信息工程系

教

案

课程名称：传感器应用技术

教　　师：吴永娜

总　学　时：54

理论学时：18

实训学时：36

上课班级：物联网 241、物联网（自主）241、物联网（3+）241、物联网（三二分段）251

授课学期：2025-2026-1

一、课程基本信息

课程名称：传感器应用技术

课程类别：专业核心课

课内学时数：30

课程实验学时数：24

适用的专业范围及层次：全日制专科电子信息工程技术、物联网等专业

学分：3

先修课程：数电、模电、单片机、C 语言程序设计

考核方式：考查

二、课程简介

《传感器应用技术》是高职物联网应用技术专业的一门核心专业课程，旨在通过理论讲授与实践操作相结合的方式，使学生全面掌握传感器技术的基本原理、分类、特性、选型、应用以及调试维护等关键技能。传感器作为物联网系统感知层的核心部件，是连接物理世界与数字世界的桥梁，对于实现数据的自动采集、转换与传输至关重要。本课程不仅深入剖析了各类常用传感器的工作原理，如温度传感器、压力传感器、光传感器、加速度传感器等，还结合物联网应用实例，讲解了传感器在智能家居、工业自动化、环境监测、医疗健康等领域的应用场景与解决方案，旨在培养学生的专业技能和解决实际问题的能力。

三、课程性质与教学目的

课程性质：

《传感器应用技术》是一门集理论性、实践性和创新性于一体的专业课程，既强调传感器基础理论知识的扎实掌握，又注重实践操作能力和创新思维的培养。课程内容紧密对接物联网行业发展需求，注重知识的前沿性和实用性，为学生后续学习物联网系统集成、数据分析与处理等课程打下坚实基础。

教学目的：

1．知识目标：使学生深入理解传感器的基本概念、工作原理、性能参数及分类方法；掌握常见传感器的结构、特性及应用领域；了解传感器技术的发展趋势及最新成果。

2．技能目标：通过课程实验、项目实训等环节，培养学生传感器选型、电路设计、数

据采集与处理、系统调试与维护等专业技能；提高学生动手能力和团队协作能力，使学生能够独立完成基于传感器的物联网应用项目设计与实施。

3．素质目标：

职业素养：强化学生的责任意识、安全意识和质量意识，培养良好的职业道德和工匠精神。

4．创新思维：激发学生的创新思维，鼓励学生在传感器应用领域探索新技术、新方法，提升解决复杂问题的能力。

课程思政元素融入：

1．爱国情怀：结合我国传感器技术的发展历程和成就，引导学生认识到科技自立自强的重要性，激发爱国情怀和民族自豪感。

2．工匠精神：通过讲述传感器研发与应用中的精益求精、追求卓越的故事，培养学生的工匠精神，强调细节决定成败，质量是企业的生命线。

3．社会责任：讨论传感器技术在环境保护、公共安全、健康医疗等领域的应用，引导学生思考科技发展的社会责任，树立可持续发展的观念，促进人与自然和谐共生。

4．团结协作：在团队合作项目中，强调沟通协调的重要性，培养学生的团队协作精神，认识到在物联网技术快速发展的今天，只有团结协作才能共创辉煌。

第 1 章 传感器基础知识

一、课时：4 课时

二、教学目标：

知识与技能：

1、学生能够理解传感器的定义、分类及其在物联网等领域的重要性；

2、掌握传感器的主要性能指标（灵敏度、线性度、迟滞、重复性、分辨率等）的概念及测试方法；

3、了解测量误差的分类、表示方法及减少误差的策略，以及测量不确定度的基本概念和评定方法。

素质目标：

1、通过案例分析、小组讨论和实验操作，培养学生分析问题、解决问题的能力；

2、引导学生运用所学知识进行实践操作，提升动手能力和团队合作精神。

情感态度与价值观：

1、激发学生对科技进步的自豪感和探索未知的热情；

2、培养学生严谨的科学态度和“精益求精”的工匠精神，增强对测量误差控制重要性的认识。

三、重难点：

重点：

1、传感器的定义；

2、传感器的性能指标概念及测试方法；

3、测量误差与不确定度的理解与应用。

难点：如何准确理解和区分不同类型的测量误差，以及如何在实践中有效减少误差。

四、教学方法：讲授法、演示法、实物展示法、分组讨论法

五、教学内容及步骤：

1、导入：什么是传感器？日常生活中应用到传感器的场景有哪些?

2、新授：（1）传感器的概念：传感器是把非电学物理量转换成易于测量、传输、处理的电学量（如电压、电流、电容等）的一种组件，起自动控制作用。

（2）传感器应用的一般模式

（3）常见传感器的元件

3、案例分析

分析一个具体的传感器应用案例，讨论其中的性能指标测试方法、误差来源及控制措施。

4、分组讨论

分组讨论：每组选择一个传感器性能指标或误差类型，讨论其在实际应用中的影响及改进策略。

思政引导：鼓励学生从工匠精神角度出发，思考如何减少误差，提升测量精度。

5、总结提升

学生汇报实验结果，分享学习心得。

教师总结本课知识点，强调思政元素在传感器技术学习中的应用。

6、课后练习题

六、教学反思：

课后需反思学生对传感器性能指标和误差控制的理解程度，以及思政元素融入的效果，以便调整后续教学策略。

七、思政元素提炼：

科技进步的自豪感、精益求精的工匠精神

第 2 章 液化器泄漏检测电路

一、课时：6 课时

分配建议：理论讲授（含演示）2 课时，实验操作与仿真 4 课时

二、教学目标

知识与技能：

1、掌握 MQ2 传感器的工作原理及其在液化气泄漏检测中的应用。

2、理解并熟悉 LMO16L 液晶显示屏的显示原理及编程方法。

3、学会设计 A/D 转换电路，实现模拟信号到数字信号的转换。

4、掌握基于 MQ2 传感器的液化气泄漏报警电路的设计、搭建与调试。

5、能够编写并调试程序，实现浓度检测、报警阈值设置及报警功能。

素质目标：

1、培养学生的动手实践能力和问题解决能力，通过电路设计与调试，提升技术应用的综合素养。

2、增强学生的团队合作与沟通能力，在实验中学会分工合作、相互学习。

情感与态度与价值观目标：

1、激发学生对电子工程技术的兴趣与热爱，培养严谨的科学态度和创新精神。

2、强化安全意识，理解液化气泄漏检测的重要性，树立保护环境和人身安全的

责任感。

三、教学重难点

重点：

1、MQ2 传感器的原理及其在液化气检测中的应用。

2、LMO16L 液晶显示屏的编程与显示控制。

3、A/D 转换电路的设计与应用。

难点：

1、LMO16L 的复杂编程逻辑及调试技巧。

2、电路中程序的设计，确保各模块协同工作，实现精准检测与报警。

四、教学方法

1、讲授法：通过理论讲解，阐述 MQ2 传感器、LMO16L 显示屏及 A/D 转换的原理。

2、演示法：利用多媒体教学工具，展示电路设计、编程过程及仿真效果。

3、实验法：组织学生进行电路搭建、程序编写、仿真调试等实践活动，加深理解。

4、讨论法：鼓励学生分组讨论，分享实验心得，共同解决实验中遇到的问题。

五、教学内容及步骤

1、理论讲授

（1）讲解 MQ2 传感器的工作原理、技术参数及应用场景。

（2）介绍 LMO16L 液晶显示屏的显示原理、编程接口及常用指令。

（3）分析 A/D 转换电路的基本原理及设计方法。

2、实验准备与演示

（1）演示液化气泄漏检测电路的设计思路与仿真过程。

（2）展示程序编写的基本框架与关键代码。

3、实验操作

（1）学生分组进行电路搭建，包括 MQ2 传感器、LMO16L 显示屏、A/D 转换模块等。

（2）编写并调试程序，实现浓度检测、阈值设置及报警功能。

（3）进行电路仿真，验证设计与程序的正确性。

4、结果检查与总结

各组展示实验结果，分享成功经验与遇到的问题。

教师点评，总结实验中的共性问题与解决策略。

5、课后拓展

六、教学反思

1、反思教学过程中学生对理论知识的接受程度，是否需要调整讲解方式或增加案例分析。

2、分析学生在实验操作中遇到的难点，思考如何更有效地进行个别指导或集体辅导。

3、总结本次教学在激发学生学习兴趣、培养实践能力和团队协作精神方面的成效与不足。

七、思政元素提炼

1、强调安全生产的重要性，通过液化气泄漏检测电路的学习，引导学生树立安全第一的意识，关注日常生活和工作环境中的安全隐患。

2、培养学生的社会责任感，认识到技术应用的背后是对社会、环境及人类安全的责任与担当。

3、弘扬创新精神，鼓励学生在掌握基础知识的同时，勇于探索未知领域，不断挑战自我，为科技进步和社会发展贡献力量。

第 3 章 铂热电阻温度测量电路

一、课时：2 课时

二、教学目标

知识与技能目标：

1、掌握铂热电阻的基本原理：理解铂热电阻如何随温度变化而改变电阻值的物理机制。

2、熟悉铂热电阻的应用场景：了解铂热电阻在温度测量领域中的广泛应用。

3、了解铂热电阻的技术参数：识别并解释关键技术参数如电阻温度系数、温度范围等。

4、掌握铂热电阻温度测量电路的设计：能够设计基于桥式电路的铂热电阻测温系统，包括选择合适的元件和配置电路。

5、掌握 pT100 铂热电阻的电路输出电压与温度的计算关系：通过理论分析和实验验证，掌握输出电压与温度之间的转换公式。

素质目标：

1、培养学生的实践动手能力和创新思维，通过电路设计与调试提升解决问题的能力。

2、强化学生的团队合作与交流能力，在小组讨论和实验中促进相互学习与支持。

情感与态度与价值观目标：

1、激发学生对工程技术的兴趣与热爱，培养严谨的科学态度和持续学习的精神。

2、强调工程伦理与社会责任感，引导学生在技术应用中考虑其对社会、环境的影响。

三、教学重难点

重点：

1、铂热电阻的原理及其测温原理。

2、铂热电阻温度测量电路的设计与实施。

难点：

1、pT100 铂热电阻的电路输出电压与温度之间复杂关系的理解与应用。

2、桥式电路的精确调试与误差分析。

四、教学方法

讲授法：系统讲解铂热电阻的原理、技术参数及电路设计理论。

演示法：通过多媒体展示和实物演示，直观展示铂热电阻及其测温电路的工作过程。

实验法：组织学生进行电路搭建、调试与数据分析，增强实践能力。

实物展示法：展示实际应用中的铂热电阻测温设备，增强感性认识。

五、教学内容及步骤

1、引入新课

（1）简述温度测量的重要性及铂热电阻的优势。

（2）展示铂热电阻实物，激发学生兴趣。

2、铂热电阻原理与技术参数

（1）讲授铂热电阻的测温原理。

（2）介绍关键技术参数，通过图表展示其特性。

3、电路设计

（1）讲解桥式电路测温原理与设计思路。

（2）分组进行电路仿真设计，包括元件选择、电路搭建等。

（3）演示软件设计过程，指导学生编写代码并生成 hex 文件。

4、实验调试与数据分析

（1）指导学生进行电路调试，包括硬件连接、软件编译与仿真。

（2）调试过程中跟踪数据，分析误差来源。

（3）验证输出电压与温度的关系，记录并处理实验数据。

5、总结与反馈

（1）分析实验结果，总结电路设计与调试中的经验教训。

（2）小组展示与讨论，分享成功经验与遇到的挑战。

（3）教师点评，提出改进建议。

六、教学反思

1、反思教学过程中的亮点与不足，如学生对桥式电路理解程度的差异、实验操作中遇到的问题等。

2、分析教学方法的有效性，考虑是否需要调整教学策略以适应不同学生的学习需求。

3、总结学生反馈，为后续课程提供改进方向。

七、思政元素提炼

工匠精神：通过精细的电路设计与调试过程，培养学生精益求精、追求卓越的工匠精神。

科学精神：强调科学原理的理解与应用，培养学生尊重科学、勇于探索的科学精神。

社会责任感：在介绍铂热电阻应用时，引导学生思考其对社会生产、环境保护等方面的作用，培养学生的社会责任感。

团队协作：在实验与讨论环节中，鼓励学生相互帮助、共同进步，培养团队合作精神。

第 4 章 大气压力测量电路（可选）

一、课时：3 课时

二、教学目标

知识与技能目标：

1、掌握大气压力传感器的种类与原理：了解不同类型大气压力传感器的工作原理及其特点。

2、深入理解 MPX4115 大气压力传感器的工作原理：详细解析 MPX4115 的工作机制及其在测量大气压力中的应用。

3、了解 MPX4115 的技术参数：熟悉 MPX4115 的主要技术参数，并能根据需求选择合适的传感器。

4、掌握大气压力测量电路的设计：能够设计包含直流稳压电源、大气压力传感器、A/D 转换、单片机及数码管显示等部分在内的完整测量电路。

5、掌握程序的编写：能够编写控制单片机读取传感器数据、进行 A/D 转换并在数码管上显示结果的程序。

素质目标：

1、培养学生的系统设计与集成能力，通过电路与程序的结合实现复杂功能的测量系统。

2、提升学生的问题解决能力，面对电路设计与调试中的挑战能够灵活应对。

情感与态度与价值观目标：

激发学生对传感器技术及应用的兴趣，培养探索未知、勇于创新的精神。

强调团队合作的重要性，在设计与调试过程中促进同学间的相互学习与支持。

三、重难点

重点：

1、MPX4115 大气压力传感器的工作原理。

2、大气压力测量电路的整体设计与实现。

3、程序的编写与调试。

难点：

1、4 位一体共阳数码管的编程与调试，确保显示准确无误。

2、系统中各部分的协同工作与故障排查。

四、教学方法

讲授法：系统讲解大气压力传感器的种类、原理及 MPX4115 的详细工作原理。

演示法：通过多媒体展示和实物演示，直观展示大气压力测量电路的工作过程。

实验法：组织学生进行电路搭建、程序编写与调试，增强实践能力。

实物展示法：展示实际应用中的大气压力测量设备，增强感性认识。

五、教学内容及步骤

1、引入新课（第 1 课时）

（1）介绍大气压力测量的重要性及应用场景。

（2）简述大气压力传感器的种类与选择原则。

2、大气压力传感器 MPX4115 详解（第 2 课时）

（1）详细讲解 MPX4115 的工作原理、技术参数及应用特点。

（2）展示 MPX4115 实物，增强直观认识。

3、电路设计与仿真（第 3-4 课时）

（1）设计大气压力测量电路，包括直流稳压电源、大气压力传感器电路、A/D 转换电路、单片机电路及数码管显示电路。

（2）使用仿真软件进行电路仿真，验证设计方案的可行性。

（3）分组进行电路设计，教师巡回指导。

4、软件设计与编程（第 5 课时）

（1）讲解单片机编程基础，包括编程语言、开发环境等。

（2）编写控制程序，实现传感器数据的读取、A/D 转换及数码管显示功能。

（3）指导学生进行程序编写，解决编程过程中遇到的问题。

5、调试与总结（第 6 课时）

（1）组织学生进行电路与程序的联合调试，确保系统正常运行。

（2）调试过程中逐步跟踪数据，分析并解决出现的问题。

（3）分析总结实验结果，讨论设计过程中的经验教训。

（4）小组展示与讨论，分享成功经验与遇到的挑战。

六、教学反思

1、反思教学过程中的难点突破情况，如 4 位一体共阳数码管编程的掌握程度。

2、分析学生对大气压力传感器原理及电路设计理解程度的差异，考虑是否需要调整教学策略。

3、总结学生反馈，评估教学效果，为后续课程提供改进方向。

七、思政元素提炼

科学精神：通过大气压力测量电路的设计与调试，培养学生严谨的科学态度和实事求是的精神。

创新精神：鼓励学生在电路设计与程序编写中勇于创新，探索更高效、更准确的测量方法。

环保意识：介绍大气压力测量在气象观测、环境保护等领域的应用，培养学生的环保意识和社会责任感。

团队合作：在实验与讨论环节中，强调团队合作的重要性，促进学生之间的相互学习与支持。

第 5 章 房间湿度测量电路（可选）

一、课时：3 课时

二、教学目标：

知识与技能目标：

1、掌握 HS1101 湿度传感器的工作原理：能够深入理解 HS1101 传感器如何检测环境湿度，并熟悉其输出信号的特性。

2、理解 555 定时器的工作原理及应用：能够明确 555 定时器的基本电路结构，掌握其在不同工作模式下的工作原理，并能将其应用于湿度测量电路中。

3、学会生成 555 输出频率与湿度的关系表：学生能够根据电路设计，通过实验或计算，推导出 555 定时器输出频率与湿度值之间的对应关系，并制作关系表。

4、掌握房间湿度测量电路的设计方法：能够独立完成从需求分析、元件选型到电路布局、连接的全过程，设计出能够准确测量并显示房间湿度的电路。

5、编写与调试湿度测量程序：学生能够编写控制程序，实现数据的读取、处理及显示功能，并进行调试以确保系统稳定运行。

素质目标：

1、培养系统设计与集成能力：通过电路设计与程序编写的实践，提升学生的系统设计与集成能力，使其能够综合运用所学知识解决实际问题。

2、增强实验与调试技能：在实验与调试过程中，锻炼学生的动手能力和问题解决能力，使其能够独立完成实验任务并有效排查故障。

3、培养创新思维：鼓励学生在电路设计与程序编写中勇于创新，尝试不同的设计方案和算法，以优化系统性能。

情感态度与价值观目标：

1、激发学习兴趣：通过介绍湿度测量在日常生活和工业生产中的应用实例，激发学生对传感器技术及电子电路设计的学习兴趣。

2、培养严谨的科学态度：在实验与调试过程中，要求学生严格遵守实验规程，认真记录实验数据，培养其严谨的科学态度和实事求是的精神。

3、增强团队合作意识：在小组实验与讨论中，鼓励学生相互协作、共同完成

任务，培养其团队合作意识和集体荣誉感。

4、树立环保意识：引导学生关注环境湿度对生活质量的影响，树立环保意识，积极参与环境保护活动。

三、重难点：

重点：

1、HS1101 传感器的工作原理及应用。

2、555 定时器在湿度测量中的应用及其输出频率与湿度的关系。

3、基于硬件与软件的系统集成设计，包括电路设计与程序设计。

难点：

1、理解并实现 555 定时器输出频率与湿度值的精确对应关系。

2、系统调试过程中故障排查与性能优化。

四、教学方法：

1、讲授法：通过理论讲解，使学生掌握基本概念和原理。

2、演示法：利用多媒体、实物或仿真软件演示电路工作过程和程序执行流程。

3、实验法：组织学生进行电路搭建、程序编写及调试实验，增强实践能力。

4、实物展示法：展示成品电路或模型，帮助学生直观理解系统构成和工作原理。

五、实验步骤

1.电路仿真设计：设计一个简单的房间湿度测量电路，能检测到室内的相对湿度，并将检测数值显示出来。由 3 部分构成：

a.湿度测量部分；

b.运算处理部分；

c.显示电路部分；

2.软件设计

3.编译连接，生成 hex 文件

4. 调试仿真

5.检查结果，逐步跟踪直至结果正确

六、分析总结

1、学生反思：要求学生撰写实验报告，总结实验过程中的收获、遇到的问题及解决方法。

2、教师总结：教师对实验过程进行回顾，点评学生表现，指出普遍存在的问题和改进建议。

3、拓展讨论：引导学生探讨湿度测量技术在日常生活和工业生产中的应用实例，激发学生的创新思维和实践能力。

七、思政元素提炼

引导学生关注环境湿度对生活质量的影响，以及湿度测量技术在环保领域的应用。通过讨论和分析，增强学生的环保意识，激发他们为改善环境质量、保护生态环境贡献自己力量的社会责任感。

实训项目部分

（每项目 2 课时）

实训项目一 磁检测传感器应用

一、实训目的

学习磁检测传感器的使用方法

二、实训设备

硬件：IOT-L01-05 型物联网综合实训箱 1 台，磁检测传感器，串口线。

软件：Keil，STC_ISP_V479，串口调试工具。

三、实训原理

源码路径：配套光盘\源代码\传感器原理及应用\实训一 磁检测传感器

hex 文件路径：配套光盘\源代码\传感器原理及应用\可执行程序

\Cikong.hex

3.1 磁检测传感器介绍

磁检测传感器使用的是干簧管。干簧管(Reed Switch)也称舌簧管或磁簧开关，是一种磁敏的特殊开关。它通常有两个软磁性材料做成的、无磁时断开的金属簧片触点，有的还有第三个作为常闭触点的簧片。这些簧片触点被封装在充有惰性气体(如氮、氦等)或真空的玻璃管里，玻璃管内平行封装的簧片端部重叠，并留有一定间隙或相互接触以构成开关的常开或常闭触点。干簧管比一般机械开关结构简单、体积小、速度高、工作寿命长；而与电子开关相比，它又有抗负载冲击能力强等特点，工作可靠性很高。

干簧管可以作为传感器用，用于计数，限位等等。例如，有一种自行车公里计，就是在轮胎上粘上磁铁，在一旁固定上干簧管构成的。把干簧管装在门上，可作为开门时的报警用，也可作为开关使用。干簧管的外形如图 3.1 所示。

图 3.1　干簧管

3.2　磁检测传感器的电路图

磁控传感器原理图，以及磁控传感器与 STC12C5A16S2 单片机的连接原理如图 3.2 所示。

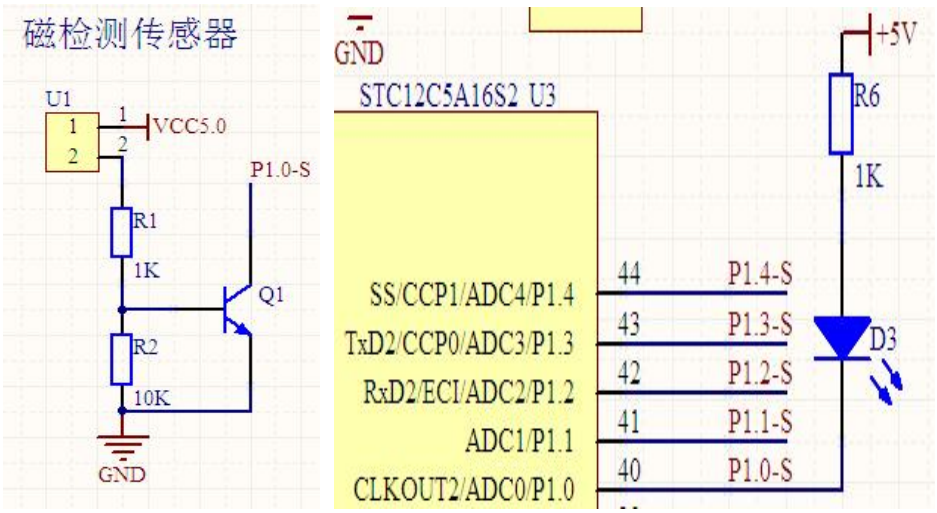


图 3.2　磁检测传感器电路图

当没有磁性物质靠近磁检测传感器时，磁检测传感器开路，Q1 不导通。

当有磁性物质靠近磁检测传感器的时候，磁检测传感器导通，则在 Q1 的基极得到使得 Q1 导通的电压，即 Q1 导通，电源从 R6--D3--Q1--地形成电流回路，则 D3 亮，同时 P1.0 为低电平。

四、实训过程

4.1　编写实训源代码文件

4.1.1　打开 KeilC51 集成开发环境，创建 Cikong 工程。

4.1.2　配置 Cikong 工程的参数。

4.1.3　编写 Cikong 源代码。

4.1.4　编译工程文件，生成可执行 Cikon.hex 文件。

4.1.5　将可执行 Cikon.hex 文件下载进 STC12C5A16S2 单片机底板。

4.1.6　将磁控传感器插入下载完程序的单片机底板。

4.1.7　将插入磁控传感器的节点重新上电。

4.2　实训源代码解析

Main.c **源代码**

```
/*************************************************************/
//晶振频率：11.0592MHz
//文件名　：Main.c
//功能说明：磁检测传感器读取实训
//变更记录：2013.05.02
//变更内容：新建造
/*************************************************************/

#include <STC12C5A60S2.h>


#define       BUF_LENTH  128     //定义串口接收缓冲长度
unsigned char     uart1_wr;          //写指针
unsigned char     uart1_rd;          //读指针
unsigned char     xdata RX0_Buffer[BUF_LENTH];    //接收缓冲
unsigned char flag;
unsigned char i;
bit       B_TI;                    //发送完成标志
sbit  P1_0 = P1^0;                 //定义 P1.0 端口
```

```
void  uart1_init(void);
void Uart1_TxByte(unsigned char dat);
void Uart1_String(unsigned char code *puts);
void delay_ms(unsigned char ms);

/*************** 用户定义参数 ****************************/

#define MAIN_Fosc       11059200UL
#define Baudrate0       9600UL

/*********************************************************/


/****************** 编译器自动生成，用户请勿修改 ************************************/

#define BRT_Reload          (256 - MAIN_Fosc / 16 / Baudrate0)
//Calculate the timer1 reload value ar 1T mode


/*********************************************************/

//*************************************************************************
//函数名：main(void)
//输入  ：无
//输出  ：无
//功能描述：当有磁性物质靠近磁检测传感器的时候，D3 亮，同时输出字符串“CiKong_Close”
```

```
    //            P1.0 采用准双向口工作模式
    //*************************************************************************
*******
    void   main(void)
    {
       uart1_init();//初始化串口
       while(1)
       {
          if(P1_0 == 0)//P1.0 为低电平的时候输出"CiKong_Close"
          {
             Uart1_String("CiKong_Close");
            Uart1_TxByte('\r');      //回车换行
            Uart1_TxByte('\n');
            delay_ms(50);           //延时下，调试用
          }
       }
    }



    //*************************************************************************
*******
    //函数名：uart1_init(void)
    //输入  ：无
    //输出  ：无
    //功能描述：串口初始化函数，通信参数为 9600 8 N 1
    //*************************************************************************
*******
    void   uart1_init(void)
    {
```

```
    PCON |= 0x80;     //UART0 Double Rate Enable
    SCON = 0x50;      //UART0 set as 10bit , UART0 RX enable
    AUXR |=  0x01;        //UART0 使用 BRT
    AUXR |=  0x04;        //BRT set as 1T mode
    BRT = BRT_Reload;
    AUXR |=  0x10;        //start BRT

    ES  = 1;
    EA = 1;
}
//*************************************************************************
*******
//函数名：Uart1_TxByte(unsigned char dat)
//输入  ：需要发送的字节数据
//输出  ：无
//功能描述：从串口发送单字节数据
//*************************************************************************
*******
void Uart1_TxByte(unsigned char dat)
{
     B_TI = 0;
    SBUF = dat;
    while(!B_TI);
    B_TI = 0;
}
//*************************************************************************
*******
//函数名：Uart1_String(unsigned char code *puts)
//输入  ：字符串首地址
```

```
    //输出  ：无
    //功能描述：从串口发送字符串
    //*********************************************************************
*******
    void Uart1_String(unsigned char code *puts)
    {
        for(; *puts != 0; puts++)
       {
            Uart1_TxByte(*puts);

       }
    }

    //*********************************************************************
*******
    //函数名：UART1_RCV (void)
    //输入  ：无
    //输出  ：无
    //功能描述：串口中断接收函数
    //*********************************************************************
*******
    void UART1_RCV (void) interrupt 4
    {
       if(RI)
       {
           RI = 0;
           RX0_Buffer[uart1_wr++] = SBUF;
           //if(++uart0_wr >= BUF_LENTH)   uart0_wr = 0;
           flag = 1;
```

```
    }

    if(TI)
    {
        TI = 0;
        B_TI = 1;
    }
}

void delay_ms(unsigned char ms)
{
     unsigned int i;
    do{
         i = MAIN_Fosc /1400;
        while(--i);
    }while(--ms);
}
```

4.3 实训运行效果

当有磁性物质靠近磁检测传感器的时候，D3 亮，同时输出字符串“CiKong_Close”，如图 4.1 所示。

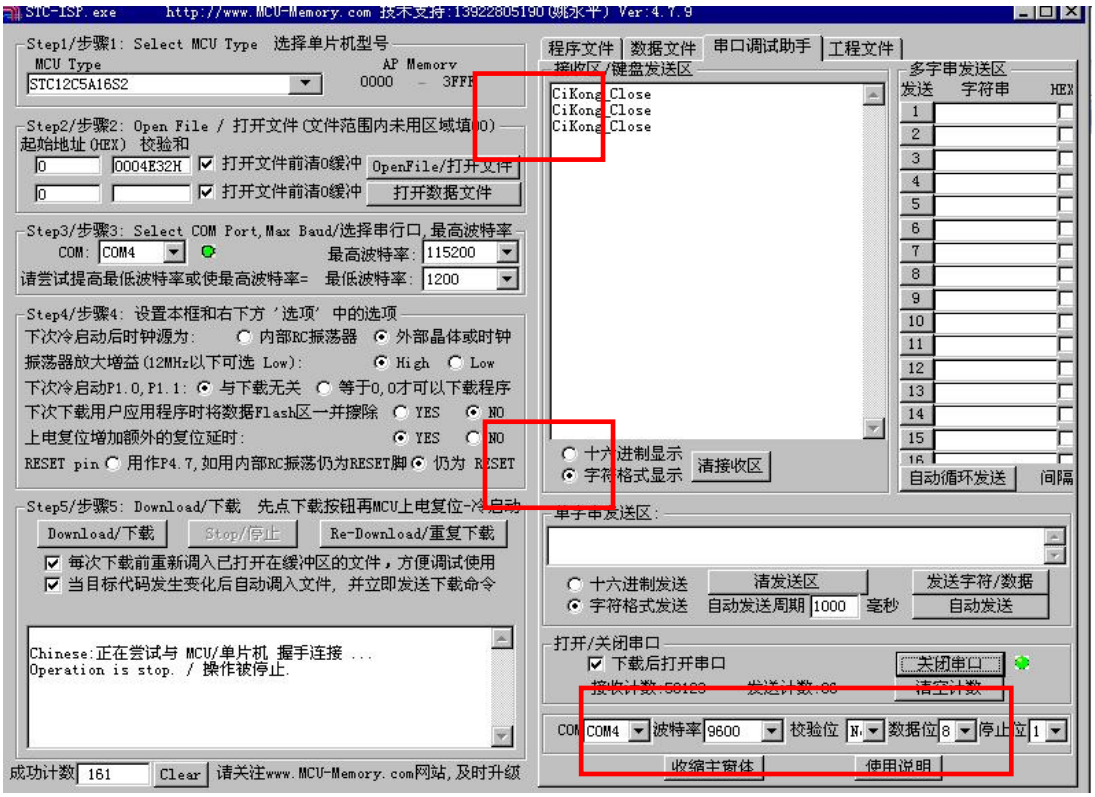


图 4.1　磁控传感器运行效果串口信息

注意：串口参数为 9600　8　N　1

实训项目二 光敏传感器应用

一、实训目的

学习光敏传感器的使用方式

二、实训设备

硬件：IOT-L01-05 型物联网综合实训箱 1 台，光敏传感器，串口线。

软件：Keil，STC_ISP_V479，串口调试工具。

三、实训原理

芯片手册：配套光盘\附件\芯片手册\光敏传感器

源码路径：配套光盘\源代码\传感器原理及应用\实训二 光敏传感器

hex 文件路径：配套光盘\源代码\传感器原理及应用\可执行程序\GuangMin.hex

3.1 光敏传感器介绍

首先我们的光敏传感器用到的是光敏电阻，光敏电阻器是一种对光敏感的元件，它的电阻值能随着外界光照强弱（明暗）变化而变化，即光敏电阻器是利用半导体光电导效应制成的一种特殊电阻器，对光线十分敏感。它在无光照射时，呈高阻状态；当有光照射时，其电阻值迅速减小。

光敏电阻器一般用于光的测量、光的控制和光电转换(将光的变化转换为电的变化)。常用的光敏电阻器硫化镉光敏电阻器，它是由半导体材料制成的。光敏电阻器的阻值随入射光线(可见光)的强弱变化而变化，在黑暗条件下，它的阻值(暗阻)可达 1~10M 欧,在强光条件(100LX)下，它阻值(亮阻)仅有几百至数千欧姆。光敏电阻器对光的敏感性(即光谱特性)与人眼对可见光(0.4~0.76) μm 的响应很接近，只要人眼可感受的光，都会引起它的阻值变化。

更加详细的资料可以参考其手册，在此不在累述。

3.2 光敏传感器的电路图

光敏传感器的电路如图 3.1 所示。

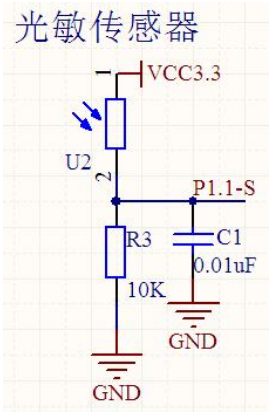


图 3.1　光敏传感器电路

一般情况下光敏电阻的暗电阻为 1M~~2MΩ，亮电阻为 1K~~15KΩ，则可以根据 P1.1 处的电压：

暗电阻下：3.3V * 10K/ (2000 K+ 10K) = 0.016V

亮电阻下：3.3V * 10K/ (15 K+ 10K) = 1.3V

我们这里使用的 STC12C5A16S2 的 ADC1（P1.1）是 10 位 ADC，根据上面的计算结果，我们可以算出亮电阻下的 ADC 值为 1.3*1024 /3.3 = 403 ，则我们可以通过采集 ADC1 的值作为临界值，当 ADC1 的值大于 400 时表明有光，小于 400 则无光。

四、实训过程

4.1　编写实训源代码文件

4.1.1　打开 KeilC51 集成开发环境，创建 GaungMin 工程。

4.1.2　配置 GaungMin 工程的参数。

4.1.3　编写 GaungMin 源代码。

4.1.4　编译工程文件，生成可执行 GaungMin.hex 文件。

4.1.5　将可执行 GaungMin.hex 文件下载进 STC12C5A16S2 单片机底板。

4.1.6　将光敏传感器插入下载完程序的单片机底板。

4.1.7　将插入光敏传感器的节点重新上电。

4.2 实训源代码解析

Main.c **源代码**

```
/*************************************************************/
//晶振频率：11.0592MHz
//文件名  ：Main.c
//功能说明：光敏传感器读取实训
//制作    ：www.frotech.com
//技术支持：020-22883196 QQ:
//变更记录：2013.05.02
//变更内容：新建造
/*************************************************************/

#include <STC12C5A60S2.h>


#define      BUF_LENTH  128    //定义串口接收缓冲长度
#define     uint unsigned int
#define     uchar unsigned char
unsigned char    uart1_wr;     //写指针
unsigned char    uart1_rd;     //读指针
unsigned char    xdata RX0_Buffer[BUF_LENTH];    //接收缓冲
unsigned char flag;
unsigned char i;
bit       B_TI;      //发送完成标志
sbit  P1_0 = P1^0;     //定义 P1.0 端口
//                          7       6       5       4           3
2    1    0   Reset Value
//sfr ADC_CONTR = 0xBC;     ADC_POWER   SPEED1   SPEED0   ADC_FLAG
ADC_START CHS2 CHS1 CHS0 0000,0000 //AD 转换控制寄存器
#define ADC_OFF()    ADC_CONTR = 0
```

```
#define ADC_ON       (1 << 7)
#define ADC_90T      (3 << 5)
#define ADC_180T (2 << 5)
#define ADC_360T (1 << 5)
#define ADC_540T 0
#define ADC_FLAG (1 << 4)       //软件清 0
#define ADC_START    (1 << 3)       //自动清 0
#define ADC_CH0      0
#define ADC_CH1      1
#define ADC_CH2      2
#define ADC_CH3      3
#define ADC_CH4      4
#define ADC_CH5      5
#define ADC_CH6      6
#define ADC_CH7      7

uint adc10_start(uchar channel);
void  uart1_init(void);
void Uart1_TxByte(unsigned char dat);
void Uart1_String(unsigned char code *puts);
void delay_ms(unsigned char ms);

/*************** 用户定义参数 ******************************/

#define MAIN_Fosc       11059200UL
#define Baudrate0       9600UL

/*********************************************************/
```

```
    /****************** 编 译 器 自 动 生 成 ， 用 户 请 勿 修 改
************************************/

    #define BRT_Reload          (256 - MAIN_Fosc / 16 / Baudrate0)
   //Calculate the timer1 reload value ar 1T mode


    /*************************************************************/

    //*************************************************************
*******
    //函数名：main(void)
    //输入  ：无
    //输出  ：无
    //功能描述：当有物体挡住光敏传感器上的光敏电阻时，ADC1(P1.1)的电压
变小
    //          当无遮挡时，ADC1 的值大于 400 时我们判定有光，且输出
“Light_Open”

    //*************************************************************
*******
    void   main(void)
    {
       uint   j;
       uart1_init();       //初始化串口
       P1ASF = (1 << ADC_CH1); //STC12C5A16S2 系列模拟输入(AD)选择
ADC1(P1.1)
       ADC_CONTR = ADC_360T | ADC_ON;
```

```
        while(1)
        {
            delay_ms(500);
            j = adc10_start(1);  //(P1.1)ADC1 转换
            if(j > 0x190)     //    当 ADC1 的值大于 400 时我们判定有光，且
输出字符串“Light_Open”
            {
               Uart1_String("Light_Open");
            }
            Uart1_TxByte('A');    //以下为按照十进制输出 ADC 值
            Uart1_TxByte('D');
            Uart1_TxByte('1');
            Uart1_TxByte('=');
            Uart1_TxByte(j/1000 + '0');
            Uart1_TxByte(j%1000/100 + '0');
            Uart1_TxByte(j%100/10 + '0');
            Uart1_TxByte(j%10 + '0');
            Uart1_TxByte(0x0d);
            Uart1_TxByte(0x0a);
        }
    }
    //*****************************************************************
*******
    //函数名：adc10_start(uchar channel)
    //输入  ：ADC 转换的通道
    //输出  ：ADC 值
    //功能描述：ADC 转换
    //*****************************************************************
*******
```

```
uint   adc10_start(uchar channel)  //channel = 0~7
{
   uint   adc;
   uchar  i;

   ADC_RES = 0;
   ADC_RESL = 0;

   ADC_CONTR = (ADC_CONTR & 0xe0) | ADC_START | channel;
   i = 250;
   do{
       if(ADC_CONTR & ADC_FLAG)
       {
           ADC_CONTR &= ~ADC_FLAG;
           adc = (uint)ADC_RES;
           adc = (adc << 2) | (ADC_RESL & 3);
           return adc;
       }
   }while(--i);
   return 1024;
}

//*************************************************************************
*******
//函数名：uart1_init(void)
//输入  ：无
//输出  ：无
//功能描述：串口初始化函数，通信参数为 9600 8 N 1
//*************************************************************************
```

```
*******
    void   uart1_init(void)
    {
       PCON |= 0x80;     //UART0 Double Rate Enable
       SCON = 0x50;      //UART0 set as 10bit , UART0 RX enable
       AUXR |=  0x01;        //UART0 使用 BRT
       AUXR |=  0x04;        //BRT set as 1T mode
       BRT = BRT_Reload;
       AUXR |=  0x10;        //start BRT

       ES  = 1;
       EA = 1;
    }
    //*************************************************************
*******
    //函数名：Uart1_TxByte(unsigned char dat)
    //输入  ：需要发送的字节数据
    //输出  ：无
    //功能描述：从串口发送单字节数据
    //*************************************************************
*******
    void Uart1_TxByte(unsigned char dat)
    {
        B_TI = 0;
       SBUF = dat;
       while(!B_TI);
       B_TI = 0;
    }
    //*************************************************************
```

```
*******
    //函数名：Uart1_String(unsigned char code *puts)
    //输入  ：字符串首地址
    //输出  ：无
    //功能描述：从串口发送字符串
    //*****************************************************************
*******
    void Uart1_String(unsigned char code *puts)
    {
        for(; *puts != 0; puts++)
       {
            Uart1_TxByte(*puts);

       }
    }

    //*****************************************************************
*******
    //函数名：UART1_RCV (void)
    //输入  ：无
    //输出  ：无
    //功能描述：串口中断接收函数
    //*****************************************************************
*******
    void UART1_RCV (void) interrupt 4
    {
       if(RI)
       {
          RI = 0;
```

```
        RX0_Buffer[uart1_wr++] = SBUF;
        //if(++uart0_wr >= BUF_LENTH)   uart0_wr = 0;
        flag = 1;
    }

    if(TI)
    {
        TI = 0;
        B_TI = 1;
    }
}

void delay_ms(unsigned char ms)
{
     unsigned int i;
    do{
         i = MAIN_Fosc /1400;
        while(--i);
    }while(--ms);
}
```

4.3 实训运行效果

如图 4.1，ADC1 的值为 312 时，即小于 400，串口没有输出“Light_Open”；当 ADC1 的值为 579，即大于 400 时，串口有输出“Light_Open”。

图 4.1　光敏传感器运行时串口信息

注意：串口的通信参数为 9600 8 N 1

实训项目三 红外对射传感器应用

一、实训目的

学习红外对射传感器工作原理

二、实训设备

硬件：IOT-L01-05 型物联网综合实训箱 1 台，带红外对射传感器，串口线。

软件：Keil，STC_ISP_V479，串口调试工具。

三、实训原理

芯片手册：配套光盘\附件\芯片手册\红外对射传感器

源码路径:配套光盘\源代码\传感器原理及应用\实训三 红外对射传感器

hex 文件路径：配套光盘\源代码\传感器原理及应用\可执行程序\HongWaiDuiShe.hex

3.1 红外对射传感器介绍

红外对射传感器使用的是槽型红外光电开关。红外光电传感器是捕捉红外线这种不可见光，采用专用的红外发射管和接收管，转换为可以观测的电信号。红外光电传感器有效地防止周围可见光的干扰，进行无接触探测，不损伤被测物体。红外光电传感器在一般情况下，有三部分构成，它们分为：发送器、接收器和检测电路。

红外对射传感器的外型如图 3.1 所示。槽型红外光电开关把一个红外光发射器和一个红外光接收器面对面地装在一个槽的两侧。发光器能发出红外光，在无阻情况下光接收器能收到光。但当被检测物体从槽中通过时，光被遮挡，光电开关便动作，输出一个开关控制信号，切断或接通负载电流，从而完成一次控制动作。槽形开关的检测距离因为受整体结构的限制一般只有几厘米。

图 3.1 红外对射传感器

3.2 光敏传感器的电路图

红外对射传感器电路如图 2，PIN1 与 PIN2 为红外发射端，PIN3 与 PIN4

为接收端，当凹槽中有物体挡住红外线时，PIN3 与 PIN4 之间截止，则 LED(D3)灭，否则亮；另外从电路实际的测量看，当无物体挡时 PIN3 与 PIN4 的电压约 2.8V,当有物体挡时电压则为 3.8V，那么用 P1.0 的 ADC0 来采集实际的 ADC 值，当大于 ADC 大于 700（0x2bc）时判定为有物体挡。

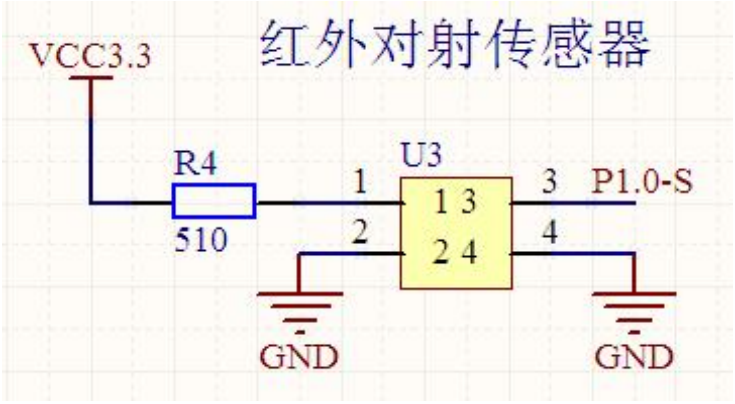


图 3.2　红外对射传感器电路

四、实训步骤

4.1　编写实训源代码文件

4.1.1　打开 KeilC51 集成开发环境，创建 HongWaiDuiShe 工程。

4.1.2　配置 HongWaiDuiShe 工程的参数。

4.1.3　编写 HongWaiDuiShe 源代码。

4.1.4　编译工程文件，生成可执行 HongWaiDuiShe.hex 文件。

4.1.5　将可执行 HongWaiDuiShe.hex 文件下载进 STC12C5A16S2 单片机底板。

4.1.6　将红外对射感器插入下载完程序的单片机底板。

4.1.7　将插入红外对射传感器的节点重新上电。

4.2　实训源代码解析

Main.c **源代码**

```
/*****************************************************************/
//晶振频率：11.0592MHz
//文件名　：Main.c
```

```
//功能说明：红外对射传感器读取实训
//制作    ：www.frotech.com
//技术支持：020-22883196 QQ:
//变更记录：2013.05.02
//变更内容：新建造
/*************************************************************/

#include <STC12C5A60S2.h>


#define      BUF_LENTH  128    //定义串口接收缓冲长度
#define     uint unsigned int
#define     uchar unsigned char
unsigned char    uart1_wr;     //写指针
unsigned char    uart1_rd;     //读指针
unsigned char    xdata RX0_Buffer[BUF_LENTH];    //接收缓冲
unsigned char flag;
unsigned char i;
bit       B_TI;      //发送完成标志
sbit  P1_0 = P1^0;   //定义 P1.0 端口
//                          7       6       5       4          3
2    1    0   Reset Value
//sfr ADC_CONTR = 0xBC;     ADC_POWER   SPEED1   SPEED0   ADC_FLAG
ADC_START CHS2 CHS1 CHS0 0000,0000 //AD 转换控制寄存器
#define ADC_OFF()    ADC_CONTR = 0
#define ADC_ON       (1 << 7)
#define ADC_90T      (3 << 5)
#define ADC_180T (2 << 5)
#define ADC_360T (1 << 5)
```

```
#define ADC_540T 0
#define ADC_FLAG (1 << 4)       //软件清 0
#define ADC_START   (1 << 3)      //自动清 0
#define ADC_CH0     0
#define ADC_CH1     1
#define ADC_CH2     2
#define ADC_CH3     3
#define ADC_CH4     4
#define ADC_CH5     5
#define ADC_CH6     6
#define ADC_CH7     7

uint adc10_start(uchar channel);
void  uart1_init(void);
void Uart1_TxByte(unsigned char dat);
void Uart1_String(unsigned char code *puts);
void delay_ms(unsigned char ms);

/*************** 用户定义参数 ****************************/

#define MAIN_Fosc       11059200UL
#define Baudrate0       9600UL

/*************************************************************/


/****************** 编 译 器 自 动 生 成 ， 用 户 请 勿 修 改
************************************/
```

```
#define BRT_Reload          (256 - MAIN_Fosc / 16 / Baudrate0)
//Calculate the timer1 reload value ar 1T mode


/*************************************************************/

//*************************************************************************
*******
//函数名：main(void)
//输入  ：无
//输出  ：无
//功能描述：当有物体挡住红外对射中间的红外线时，红外对射传感器截止，
P1.0 为高电平，
//          D3 灭，串口输出字符串“HongWai_Open”，P1.0 采用 ADC 工作
方式
//*************************************************************************
*******
void  main(void)
{
   uint   j;
   uart1_init();               //初始化串口
   P1ASF = (1 << ADC_CH0); //STC12C5A16S2 系列模拟输入(AD)选择
ADC0(P1.0)
   ADC_CONTR = ADC_360T | ADC_ON;
   while(1)
   {
       delay_ms(500);
       j = adc10_start(0);       // (P1.0)ADC0 转换
      if(j > 0x2BC)          //如果 ADC 值大于 700，即红外对射传感器
```

```
中间有外物挡住红外线，输出字符串“HongWai_Open”
            {
               Uart1_String("HongWai_Open");
            }
            Uart1_TxByte('A');     //以下为按照十进制输出 ADC 值
            Uart1_TxByte('D');
            Uart1_TxByte('0');
            Uart1_TxByte('=');
            Uart1_TxByte(j/1000 + '0');
             Uart1_TxByte(j%1000/100 + '0');
            Uart1_TxByte(j%100/10 + '0');
            Uart1_TxByte(j%10 + '0');
            Uart1_TxByte(0x0d);
            Uart1_TxByte(0x0a);
        }
    }
    //*****************************************************************
*******
    //函数名：adc10_start(uchar channel)
    //输入  ：ADC 转换的通道
    //输出  ：ADC 值
    //功能描述：ADC 转换
    //*****************************************************************
*******
    uint   adc10_start(uchar channel)  //channel = 0~7
    {
       uint   adc;
       uchar  i;
```

```
    ADC_RES = 0;
    ADC_RESL = 0;

    ADC_CONTR = (ADC_CONTR & 0xe0) | ADC_START | channel;
    i = 250;
    do{
        if(ADC_CONTR & ADC_FLAG)
        {
            ADC_CONTR &= ~ADC_FLAG;
            adc = (uint)ADC_RES;
            adc = (adc << 2) | (ADC_RESL & 3);
            return adc;
        }
    }while(--i);
    return 1024;
}

//*************************************************************************
*******
//函数名：uart1_init(void)
//输入  ：无
//输出  ：无
//功能描述：串口初始化函数，通信参数为9600 8 N 1
//*************************************************************************
*******
void  uart1_init(void)
{
    PCON |= 0x80;     //UART0 Double Rate Enable
    SCON = 0x50;      //UART0 set as 10bit , UART0 RX enable
```

```
    AUXR |=  0x01;          //UART0 使用 BRT
    AUXR |=  0x04;          //BRT set as 1T mode
    BRT = BRT_Reload;
    AUXR |=  0x10;          //start BRT

    ES  = 1;
    EA = 1;
}
//*********************************************************************
*******
//函数名：Uart1_TxByte(unsigned char dat)
//输入  ：需要发送的字节数据
//输出  ：无
//功能描述：从串口发送单字节数据
//*********************************************************************
*******
void Uart1_TxByte(unsigned char dat)
{
    B_TI = 0;
    SBUF = dat;
    while(!B_TI);
    B_TI = 0;
}
//*********************************************************************
*******
//函数名：Uart1_String(unsigned char code *puts)
//输入  ：字符串首地址
//输出  ：无
//功能描述：从串口发送字符串
```

```
    //*****************************************************************
*******
    void Uart1_String(unsigned char code *puts)
    {
        for(; *puts != 0; puts++)
       {
            Uart1_TxByte(*puts);

       }
    }

    //*****************************************************************
*******
    //函数名：UART1_RCV (void)
    //输入  ：无
    //输出  ：无
    //功能描述：串口中断接收函数
    //*****************************************************************
*******
    void UART1_RCV (void) interrupt 4
    {
       if(RI)
       {
           RI = 0;
           RX0_Buffer[uart1_wr++] = SBUF;
           //if(++uart0_wr >= BUF_LENTH)   uart0_wr = 0;
           flag = 1;
       }
```

```
    if(TI)
    {
        TI = 0;
        B_TI = 1;
    }
}

void delay_ms(unsigned char ms)
{
     unsigned int i;
    do{
         i = MAIN_Fosc /1400;
        while(--i);
    }while(--ms);
}
```

4.3 实训运行效果

使用 ADC0 的实际采集值判定是否有物体挡住红外对射传感器的凹槽，当 ADC 值大于 700 时输出字符串“HongWai_Open”。

实训项目四 红外反射传感器应用

一、实训目的

学习红外反射传感器的使用方式

二、实训设备

硬件：IOT-L01-05 型物联网综合实训箱 1 台，红外反射传感器，串口线。

软件：Keil，STC_ISP_V479，串口调试工具。

三、实训原理

芯片手册：配套光盘\附件\芯片手册\红外反射传感器

源码路径:配套光盘\源代码\传感器原理及应用\实训四 红外反射传感器

hex 文件路径：配套光盘\源代码\传感器原理及应用\可执行程序\HongWaiFanShe.hex

3.1 红外反射传感器介绍

红外反射传感器采用红外对射管 TCRT5000，其电路结构如图 3.1，其中右边 A--C 是红外发射管，左边 C--E 是红外接收管，当我们用物体在 TCRT5000 上方挡住时，红外光被反射给接收管，从而使得 C--E 导通；该红外对射管的探测范围是 0.2mm---15mm。实际电路如图 2 所示。

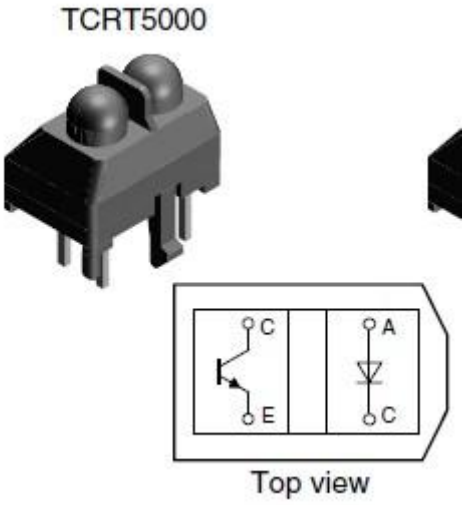


图 3.1　红外反射传感器

3.2　红外反射传感器的电路图

红外反射传感器的电路如图 3.2 所示。

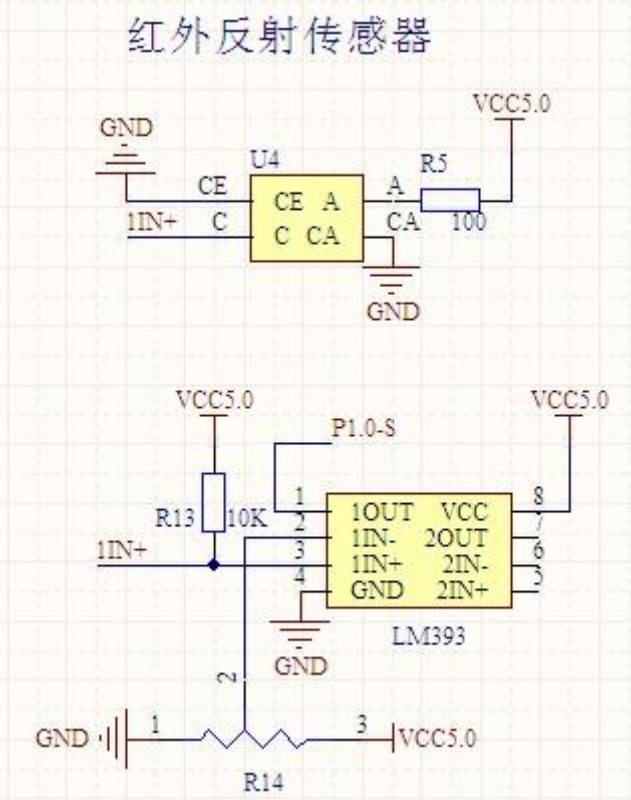


图 3.2　红外反射传感器

其中 U4 是红外对射管

LM393 是电压比较器

R14 是调节红外反射传感器测试灵敏度的，顺时针调节更灵敏。

四、实训过程

4.1　编写实训源代码文件

4.1.1　打开 KeilC51 集成开发环境，创建 HongWaiFanShe 工程。

4.1.2　配置 HongWaiFanShe 工程的参数。

4.1.3　编写 HongWaiFanShe 源代码。

4.1.4　编译工程文件，生成可执行 HongWaiFanShe.hex 文件。

4.1.5　将可执行 HongWaiFanShe.hex 文件下载进 STC12C5A16S2 单片机底板。

4.1.6　将红外反射感器插入下载完程序的单片机底板。

4.1.7　将插入红外反射传感器的节点重新上电。

4.2　实训源代码解析

Main.c 源代码

```
/*************************************************************/
//晶振频率：11.0592MHz
//文件名　：Main.c
//功能说明：红外反射传感器读取实训
//制作　　：www.frotech.com
//技术支持：020-22883196 QQ:
//变更记录: 2013.05.02
//变更内容: 新建造
/*************************************************************/

#include <STC12C5A60S2.h>


#define      BUF_LENTH  128     //定义串口接收缓冲长度
unsigned char    uart1_wr;      //写指针
unsigned char    uart1_rd;      //读指针
unsigned char    xdata RX0_Buffer[BUF_LENTH];    //接收缓冲
unsigned char flag;
unsigned char i;
bit       B_TI;           //发送完成标志
sbit  P1_0 = P1^0;        //定义 P1.0 端口

void  uart1_init(void);
void Uart1_TxByte(unsigned char dat);
void Uart1_String(unsigned char code *puts);
void delay_ms(unsigned char ms);
```

```
/*************** 用户定义参数 ******************************/

#define MAIN_Fosc       11059200UL
#define Baudrate0       9600UL

/*****************************************************************/


/****************** 编 译 器 自 动 生 成 ， 用 户 请 勿 修 改
************************************/

#define BRT_Reload          (256 - MAIN_Fosc / 16 / Baudrate0)
//Calculate the timer1 reload value ar 1T mode


/*****************************************************************/

//*************************************************************************
*******
//函数名：main(void)
//输入  ：无
//输出  ：无
//功能描述：当用白色纸放在红外反射传感器上方约 5mm 处时候，D1 亮，
同时输出字符串“Hong Wai Fan She”
//          P1.0 采用准双向口工作模式
//*************************************************************************
*******
void  main(void)
```

```
    {
       uart1_init();                    //初始化串口
       while(1)
       {
          if(P1_0 == 0)                 //P1.0 为低电平的时候输出“Hong Wai
Fan She”
          {
                Uart1_String("Hong Wai Fan She");
                Uart1_TxByte('\r');       //回车换行
                Uart1_TxByte('\n');
                delay_ms(50);           //延时下，调试用
          }
       }
    }


    //*****************************************************************
*******
    //函数名：uart1_init(void)
    //输入  ：无
    //输出  ：无
    //功能描述：串口初始化函数，通信参数为 9600 8 N 1
    //*****************************************************************
*******
    void   uart1_init(void)
    {
       PCON |= 0x80;      //UART0 Double Rate Enable
       SCON = 0x50;       //UART0 set as 10bit , UART0 RX enable
       AUXR |=  0x01;         //UART0 使用 BRT
```

```
    AUXR |=  0x04;          //BRT set as 1T mode
    BRT = BRT_Reload;
    AUXR |=  0x10;          //start BRT

    ES  = 1;
    EA = 1;
}
//*********************************************************************
*******
//函数名：Uart1_TxByte(unsigned char dat)
//输入  ：需要发送的字节数据
//输出  ：无
//功能描述：从串口发送单字节数据
//*********************************************************************
*******
void Uart1_TxByte(unsigned char dat)
{
    B_TI = 0;
    SBUF = dat;
    while(!B_TI);
    B_TI = 0;
}
//*********************************************************************
*******
//函数名：Uart1_String(unsigned char code *puts)
//输入  ：字符串首地址
//输出  ：无
//功能描述：从串口发送字符串
//*********************************************************************
```

```
*******
    void Uart1_String(unsigned char code *puts)
    {
        for(; *puts != 0; puts++)
       {
            Uart1_TxByte(*puts);

       }
    }

    //*****************************************************************
*******
    //函数名：UART1_RCV (void)
    //输入  ：无
    //输出  ：无
    //功能描述：串口中断接收函数
    //*****************************************************************
*******
    void UART1_RCV (void) interrupt 4
    {
       if(RI)
       {
           RI = 0;
           RX0_Buffer[uart1_wr++] = SBUF;
           //if(++uart0_wr >= BUF_LENTH)   uart0_wr = 0;
           flag = 1;
       }

       if(TI)
```

```
    {
        TI = 0;
        B_TI = 1;
    }
}

void delay_ms(unsigned char ms)
{
     unsigned int i;
    do{
         i = MAIN_Fosc /1400;
        while(--i);
    }while(--ms);
}
```

4.3 实训运行效果

4.3.1 实训箱上运行效果

给 B1 板上电，如果没有物体在红外对射管上方时，B1 板上的 D1 一直亮，那么请按照图 3 所示逆时针调节 R14 阻值，使得 D1 灭；然后把物体放在红外对射管上面约 5mm 处使得 D1 灯亮（如果不亮，请按照图 3 顺时针调节 R14 阻值直至 D1 亮），如图 5 所示，拿开物体后 D1 又不亮。描述得简单点就是顺时针调节更灵敏，反之则更迟钝。

在这里我们如果用白色的纸来挡红外反射管产生的红外光的话会更明显，如果用黑色纸的话基本上是没反应，这样我们把红外反射传感器当作黑白识别的传感器，比如在循迹小车上应用。

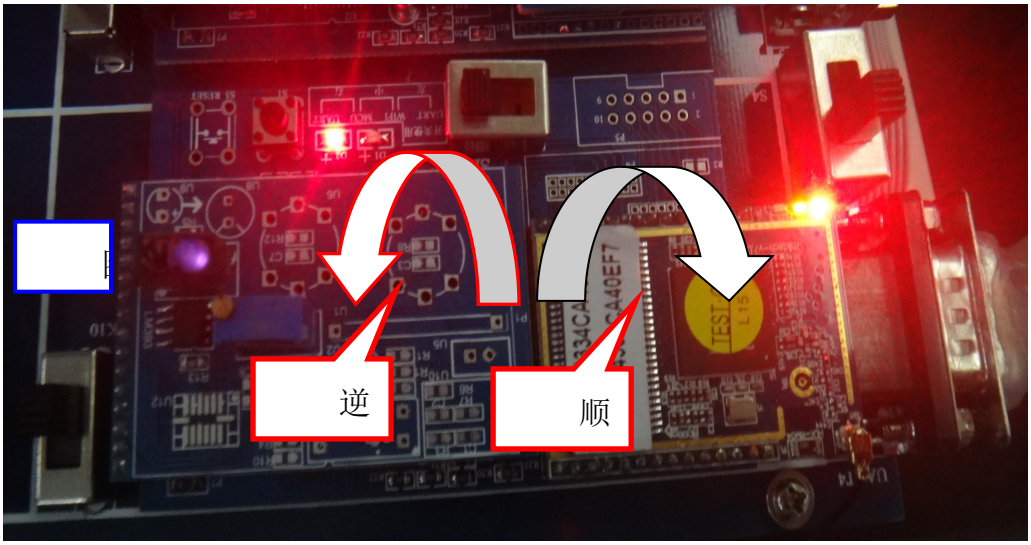


图 3　红外对射传感器实训箱上运行效果

4.3.2　串口输出信息

当使用白纸挡住红外发射器时，串口输出“Hong Wai Fan She”信息如图 4。

图 4　运行时串口信息

注意：串口的参数为 9600　8　N　1

实训项目五 火焰传感器应用

一、实验目的

学习火焰传感器的使用方式

二、实验设备

硬件：IOT-L01-05 型物联网综合实验箱 1 台，带火焰传感器，串口线。

软件：Keil，STC_ISP_V479，串口调试工具。

三、实验原理

芯片手册：配套光盘\附件\芯片手册\火焰传感器

源码路径：配套光盘\源代码\传感器原理及应用\实验五 火焰传感器

hex 文件路径：配套光盘\源代码\传感器原理及应用\可执行程序\Fire.hex

3.1 火焰传感器介绍

火焰传感器采用红外接收管来作探头，该红外接收头能接收 700--1100nm 的红外光，而火焰的能产生 900nm 左右的红外光，所以用红外接收头来探测火焰比较合适。红外接收管的结构图如图 3.1。

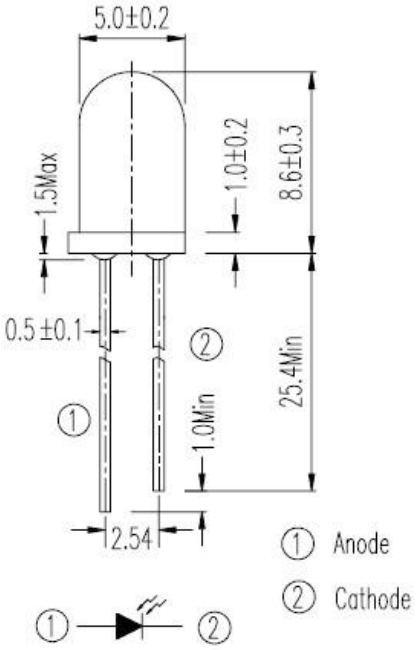


图 3.1　红外接收管型火焰传感器

3.2　电路原理图

火焰传感器的电路原理如图 3.2。

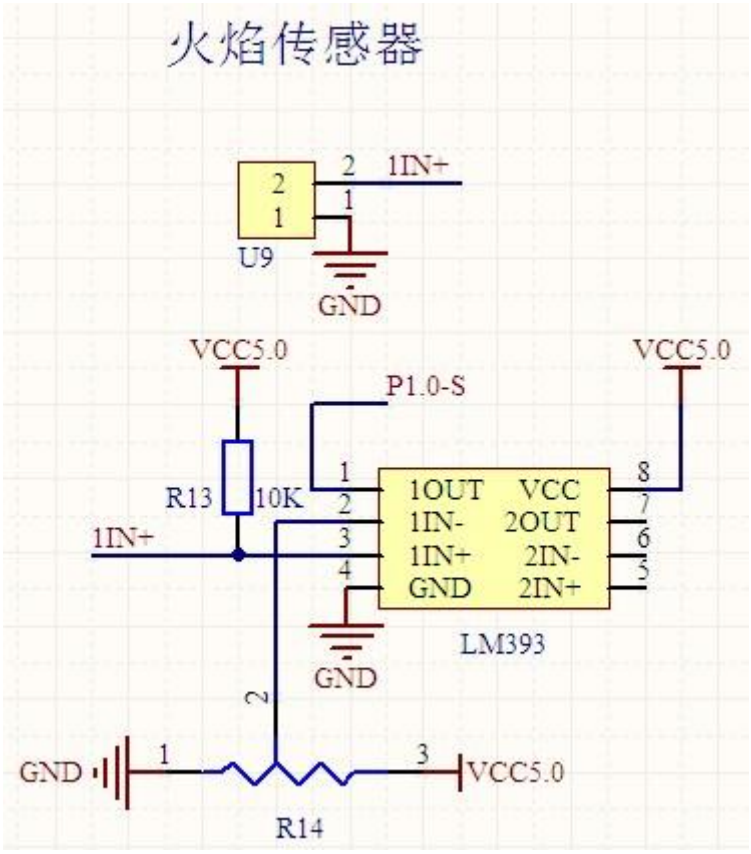


图 3.2　火焰传感器电路

U9 为红外接收管

LM393 为电压比较器

R14 为电位器，用于调节接收火焰的红外光的灵敏度。

P1.0-S 为 LM393 输出端，当检测到有火焰的时候，它的电平会变低。

四、实验过程

4.1　编写实验源代码文件

4.1.1　打开 KeilC51 集成开发环境，创建 Fire 工程。

4.1.2　配置 Fire 工程的参数。

4.1.3　编写 Fire 源代码。

4.1.4　编译工程文件，生成可执行 Fire.hex 文件。

4.1.5　将可执行 Fire.hex 文件下载进 STC12C5A16S2 单片机底板。

4.1.6　将火焰传感器插入下载完程序的单片机底板。

4.1.7　将插入火焰传感器的节点重新上电。

4.2　实验源代码解析

Main.c **源代码**

```
/*************************************************************/
//晶振频率：11.0592MHz
//文件名　：Main.c
//功能说明：火焰传感器读取实验
//制作　　：www.frotech.com
//技术支持：020-22883196 QQ:
//变更记录：2013.05.02
//变更内容：新建造
/*************************************************************/
```

```
#include <STC12C5A60S2.h>


#define      BUF_LENTH  128    //定义串口接收缓冲长度
unsigned char    uart1_wr;         //写指针
unsigned char    uart1_rd;         //读指针
unsigned char    xdata RX0_Buffer[BUF_LENTH];    //接收缓冲
unsigned char flag;
unsigned char i;
bit       B_TI;                   //发送完成标志
sbit  P1_0 = P1^0;                //定义 P1.0 端口

void  uart1_init(void);
void Uart1_TxByte(unsigned char dat);
void Uart1_String(unsigned char code *puts);
void delay_ms(unsigned char ms);

/*************** 用户定义参数 ******************************/

#define MAIN_Fosc       11059200UL
#define Baudrate0       9600UL

/*****************************************************************/


/****************** 编 译 器 自 动 生 成 ， 用 户 请 勿 修 改
************************************/

#define BRT_Reload          (256 - MAIN_Fosc / 16 / Baudrate0)
```

```
    //Calculate the timer1 reload value ar 1T mode


    /*************************************************************/

    //*************************************************************
*******
    //函数名：main(void)
    //输入  ：无
    //输出  ：无
    //功能描述：当有火焰或者灯直射火焰传感器时候，D1 亮，同时输出字符
串“Fire Now”
    //          P1.0 采用准双向口工作模式
    //*************************************************************
*******
    void   main(void)
    {
       uart1_init();            //初始化串口
       while(1)
       {
          if(P1_0 == 0)        //P1.0 为低电平的时候输出“Fire Now”
          {
                Uart1_String("Fire Now");
                Uart1_TxByte('\r');   //回车换行
                Uart1_TxByte('\n');
                delay_ms(50);       //延时下，调试用
          }
       }
    }
```

```
    //*****************************************************************
*******
    //函数名：uart1_init(void)
    //输入  ：无
    //输出  ：无
    //功能描述：串口初始化函数，通信参数为9600 8 N 1
    //*****************************************************************
*******
    void  uart1_init(void)
    {
       PCON |= 0x80;     //UART0 Double Rate Enable
       SCON = 0x50;      //UART0 set as 10bit , UART0 RX enable
       AUXR |=  0x01;        //UART0 使用BRT
       AUXR |=  0x04;        //BRT set as 1T mode
       BRT = BRT_Reload;
       AUXR |=  0x10;        //start BRT

       ES  = 1;
       EA = 1;
    }
    //*****************************************************************
*******
    //函数名：Uart1_TxByte(unsigned char dat)
    //输入  ：需要发送的字节数据
    //输出  ：无
    //功能描述：从串口发送单字节数据
    //*****************************************************************
```

```
*******
    void Uart1_TxByte(unsigned char dat)
    {
        B_TI = 0;
       SBUF = dat;
       while(!B_TI);
       B_TI = 0;
    }
    //*****************************************************************
*******
    //函数名：Uart1_String(unsigned char code *puts)
    //输入  ：字符串首地址
    //输出  ：无
    //功能描述：从串口发送字符串
    //*****************************************************************
*******
    void Uart1_String(unsigned char code *puts)
    {
        for(; *puts != 0; puts++)
       {
            Uart1_TxByte(*puts);

       }
    }

    //*****************************************************************
*******
    //函数名：UART1_RCV (void)
    //输入  ：无
```

```
    //输出  ：无
    //功能描述：串口中断接收函数
    //*****************************************************************
*******
    void UART1_RCV (void) interrupt 4
    {
       if(RI)
       {
          RI = 0;
          RX0_Buffer[uart1_wr++] = SBUF;
          //if(++uart0_wr >= BUF_LENTH)   uart0_wr = 0;
          flag = 1;
       }

       if(TI)
       {
          TI = 0;
          B_TI = 1;
       }
    }

    void delay_ms(unsigned char ms)
    {
        unsigned int i;
       do{
           i = MAIN_Fosc /1400;
          while(--i);
       }while(--ms);
    }
```

4.3　实验运行效果

4.3.1　实验箱上运行效果

如果没有火焰或者灯直射红外接收管时，B1 板上的 D1 亮，那么请按照图 3 所示逆时针调节 R14 阻值，使得 D1 灭；然后用火焰（为了安全，如用火焰测试最好是在室外，并注意安全）或者灯直射红外接收管时 D1 灯亮（如果不亮，请按照图 3 顺时针调节 R14 阻值直至 D1 亮），如图 4；拿开火焰或者灯后 D1 又不亮，那么火焰传感器测试 OK。

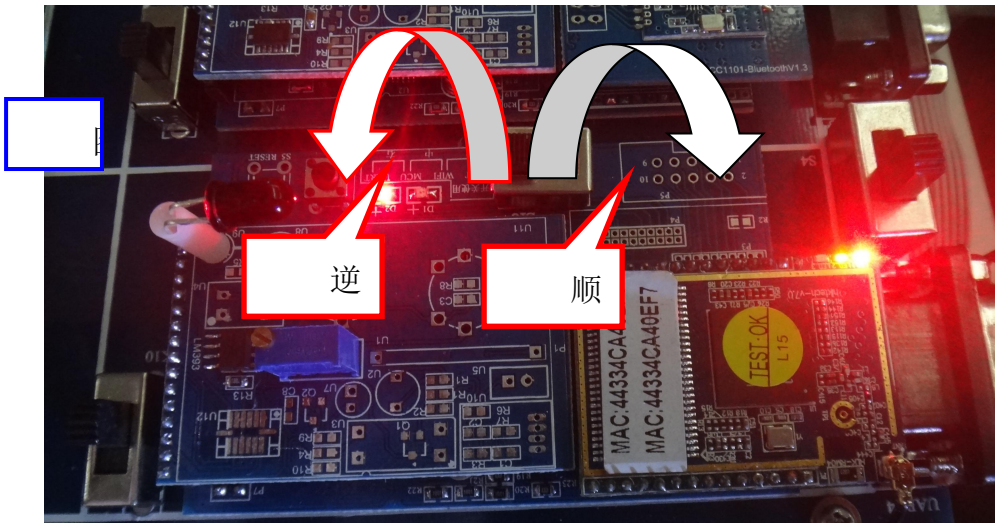


图 3　火焰传感器实验箱上运行效果

4.3.2　串口输出信息

当我们用火焰或者灯光直射火焰传感器的红外接收管时，串口接收到“Fire Now”

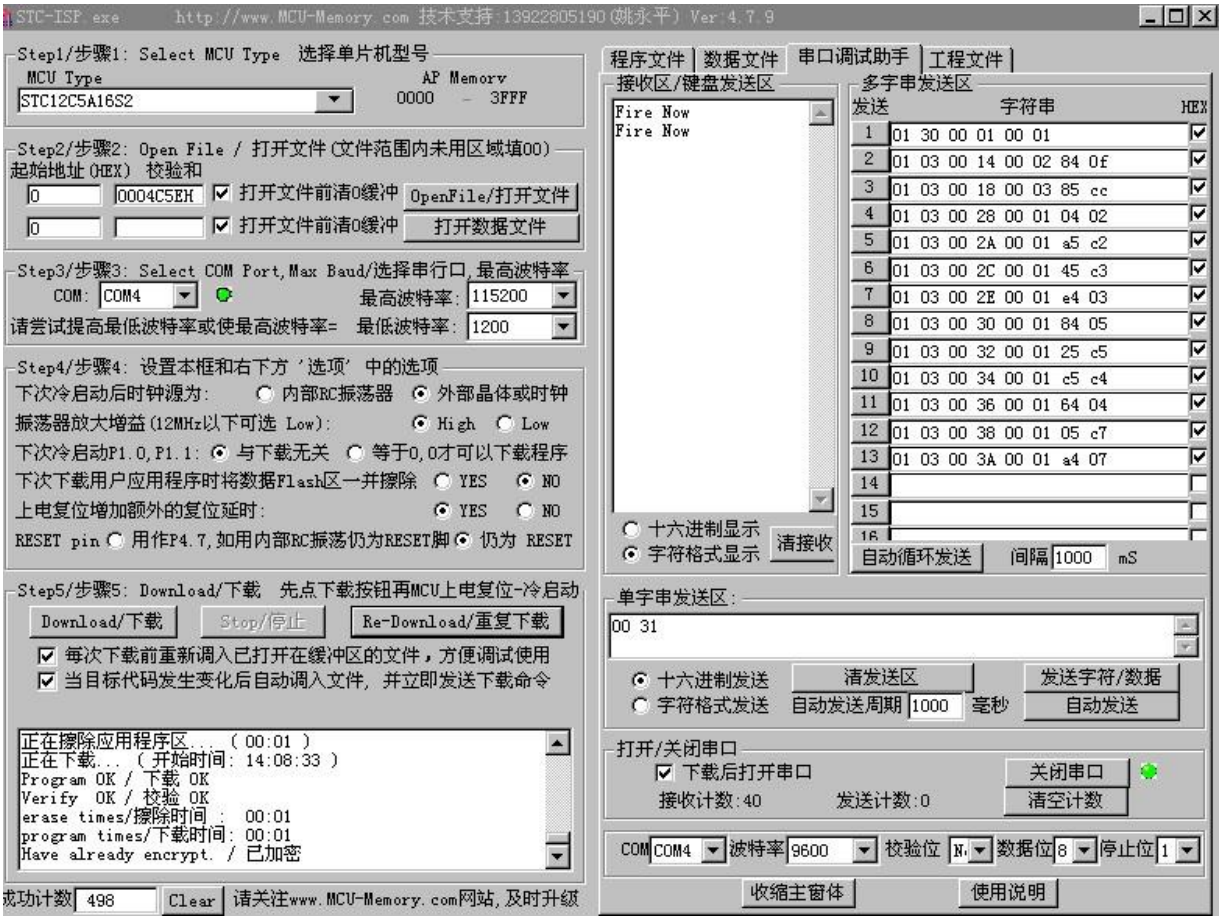


图 4　运行时串口信息

注意：串口的参数为 9600　8　N　1

实训项目六 加速度传感器应用

一、实验目的

1. 学习 IIC 通信实验
2. 学习加速度传感器 ADXL345 的加速度值读取

注意：3D 加速度传感器模块非实验箱标配传感器模块，如果学生使用的实验箱中没有该模块，请跳过此实验。

二、实验设备

硬件：IOT-L01-05 型物联网综合实验箱 1 台，带 3D 加速度传感器，串口线。

软件：Keil，STC_ISP_V479，串口调试工具。

三、实验原理

芯片手册：配套光盘\附件\芯片手册\加速度传感器 ADXL345

源码路径:配套光盘\源代码\传感器原理及应用\实验六 加速度传感器

hex 文件路径：配套光盘\源代码\传感器原理及应用\可执行程序\ADXL345.hex

3.1 3D 加速度传感器 ADXL345 简介

ADXL345 是超低功耗的 3 轴加速度计，分辨率高 13 位，测量范围可达±16g,高分辨率（3.9mg/LSB),能够测量不到 1°的角度变化，数字输出采用 16 进制补码格式，可用 ISP(3 线或 4 线）或者 IIC 通信访问。

3.2 ADXL345 电路原理图

ADXL345 传感器电路连接如图 3.1 所示。

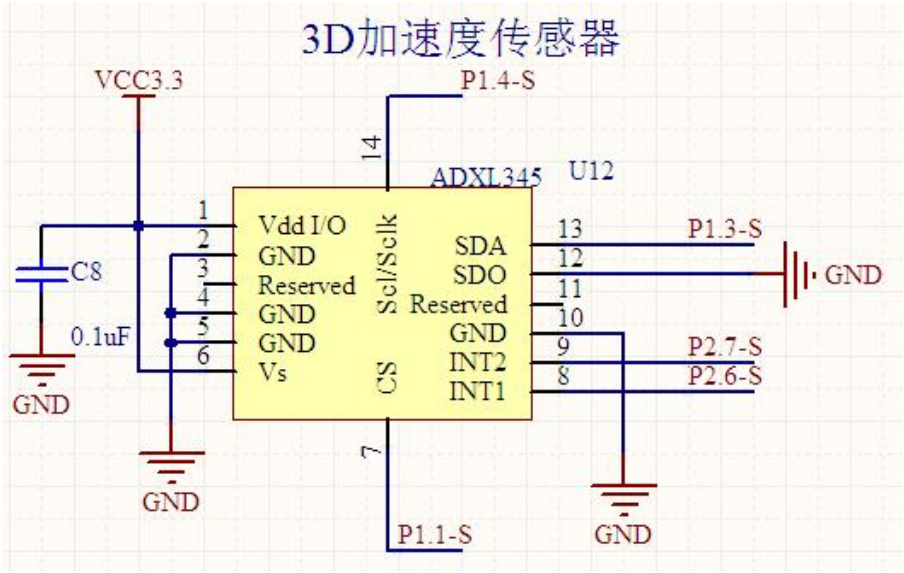


图 3.1　3D 加速度电路图

本电路采用 IIC 通信方式，ALT ADDRESS 接地（则地址为 0xA6），在操作过程中有用到的是 SDA 及 SCL。

3.3　ADXL345 在 IIC 下的时序

ADXL345 在 IIC 下的数据操作时序如图 3.2，详细的可参考其中文手册 P18。

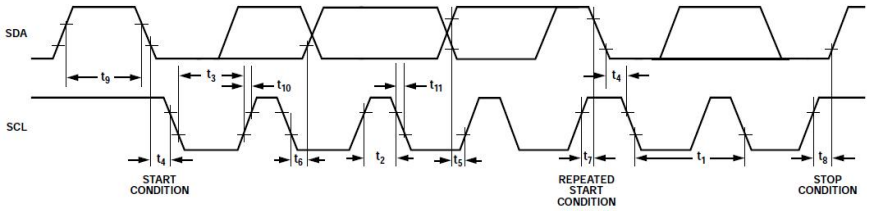


图 3.2　ADXL345 在 IIC 下的时序

3.4　ADXL345 的 IIC 数据操作

ADXL345 的 IIC 数据操作方式如图 3.3，可参考中文手册 P17。

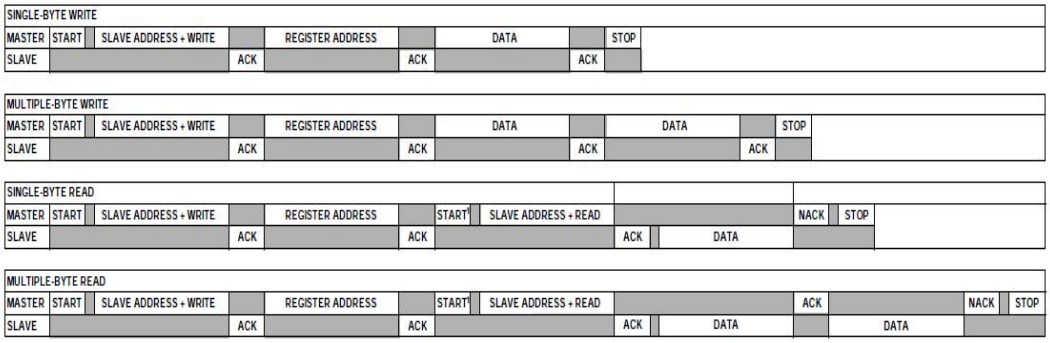


图 3.3　ADXL345 的 IIC 数据操作方式

3.5　ADXL345 的寄存器映射

ADXL345 的寄存器映射表如图 3.4 所示，具体可参考中文手册 P22

地址					
十六进制	十进制	名称	类型	复位值	描述
0x00	0	DEVID	R	11100101	器件ID
0x01 to 0x1C	1 to 28	保留			
0x1D	29	THRESH_TAP	R/W	00000000	保留，不要操作
0x1E	30	OFSX	R/W	00000000	敲击阈值
0x1F	31	OFSY	R/W	00000000	X轴偏移
0x20	32	OFSZ	R/W	00000000	Y轴偏移
0x21	33	DUR	R/W	00000000	Z轴偏移
0x22	34	Latent	R/W	00000000	敲击持续时间
0x23	35	Window	R/W	00000000	敲击延迟
0x24	36	THRESH_ACT	R/W	00000000	敲击窗口
0x25	37	THRESH_INACT	R/W	00000000	活动阈值
0x26	38	TIME_INACT	R/W	00000000	静止阈值
0x27	39	ACT_INACT_CTL	R/W	00000000	静止时间
0x28	40	THRESH_FF	R/W	00000000	轴使能控制活动和静止检测
0x29	41	TIME_FF	R/W	00000000	自由落体阈值
0x2A	42	TAP_AXES	R/W	00000000	自由落体时间
0x2B	43	ACT_TAP_STATUS	R	00000000	单击/双击轴控制
0x2C	44	BW_RATE	R/W	00001010	单击/双击源
0x2D	45	POWER_CTL	R/W	00000000	数据速率及功率模式控制
0x2E	46	INT_ENABLE	R/W	00000000	省电特性控制
0x2F	47	INT_MAP	R/W	00000000	中断使能控制
0x30	48	INT_SOURCE	R	00000010	中断映射控制
0x31	49	DATA_FORMAT	R/W	00000000	中断源
0x32	50	DATAX0	R	00000000	数据格式控制
0x33	51	DATAX1	R	00000000	X轴数据0
0x34	52	DATAY0	R	00000000	X轴数据1
0x35	53	DATAY1	R	00000000	Y轴数据0
0x36	54	DATAZ0	R	00000000	Y轴数据1
0x37	55	DATAZ1	R	00000000	Z轴数据0
0x38	56	FIFO_CTL	R/W	00000000	Z轴数据1
0x39	57	FIFO_STATUS	R	00000000	FIFO控制
					FIFO状态

图 3.4　ADXL345 的寄存器映射

3.5 通过 IIC 配置 ADXL345

在读取 ADXL345 的地址 0x32~~0x37 的加速度值之前，我们需要配置它的基本参数，

在这里我们测量范围设为±16g，13 位数据模式，即地址为 0x31 的 DATA_FORMAT 寄存器配置为 0x0b;速率设置为 100，即地址为 0x2C 的 BW_RATE 寄存器设为 0x0A;电源模式设为测量模式，即地址为 0x2D 的 POWER_CTL 设为 0x08;中断使能，地址为 0x2E 的 INT_ENABLE 寄存器设为 0x80;X 偏移量寄存器（0x1E）

设为 0xFA；Y 偏移量寄存器（0x1F）设为 0xFD;Z 偏移量寄存器（0x20）设为 0x06.配置好以后，我们就可以读取加速度值了。

这一过程详见源代码的 ADXL345 的初始化程序。

四、实验过程

4.1 编写实验源代码文件

4.1.1 打开 KeilC51 集成开发环境，创建 ADXL345 工程。

4.1.2 配置 ADXL345 工程的参数。

4.1.3 编写 ADXL345 源代码。

4.1.4 编译工程文件，生成可执行 ADXL345.hex 文件。

4.1.5 将可执行 ADXL345.hex 文件下载进 STC12C5A16S2 单片机底板。

4.1.6 将 3D 加速度传感器插入下载完程序的单片机底板。

4.1.7 将插入 3D 加速度传感器的节点重新上电。

4.2 实验源代码解析

Main.c **源代码**

```
#include <STC12C5A60S2.h>
#include <intrins.h>

/*************************************************************/
//晶振频率：11.0592MHz
//文件名  ：Main.c
//功能说明：加速度传感器 ADXL345 读取实验
//制作    ：www.frotech.com
//技术支持：020-22883196 QQ:
//变更记录：2013.05.02
//变更内容：新建造
/*************************************************************/

#define     BUF_LENTH     128       //定义串口接收缓冲长度
```

```
#define   uchar unsigned char
#define   uint unsigned int
#define     SlaveAddress   0xA6      //定义器件在 IIC 总线中的从地址,根据 ALT  ADDRESS 地址引脚不同修改
                                     //ALT  ADDRESS 引脚接地时地址为 0xA6，接电源时地址为 0x3A

unsigned char    uart1_wr;     //写指针
unsigned char    uart1_rd;     //读指针
unsigned char    xdata RX0_Buffer[BUF_LENTH];    //接收缓冲
unsigned       char                    ASCII_Buffer[10]       = {'0','1','2','3','4','5','6','7','8','9'};
unsigned char flag;
unsigned char i;

bit         B_TI;
sbit     SCL=P1^4;      //IIC 时钟引脚定义
sbit     SDA=P1^3;      //IIC 数据引脚定义

typedef unsigned char  BYTE;
typedef unsigned short WORD;

BYTE BUF[8];  //接收数据缓存区

/**********    函    数    声    明    区    **********************/

void uart1_init(void);
void Uart1_TxByte(unsigned char dat);
void Uart1_String(unsigned char code *puts);
```

```
    void Init_ADXL345(void);                    //初始化 ADXL345
    void  Single_Write_ADXL345(uchar REG_Address,uchar REG_data);   //
单个写入数据
    uchar Single_Read_ADXL345(uchar REG_Address);                   //
单个读取内部寄存器数据
    void  Multiple_Read_ADXL345();
    void Delay5us();
    void Delay5ms();
    void ADXL345_Start();
    void ADXL345_Stop();
    void ADXL345_SendACK(bit ack);
    bit  ADXL345_RecvACK();
    void ADXL345_SendByte(BYTE dat);
    BYTE ADXL345_RecvByte();
    /*********************************************************/


    /*************** 用户定义参数 ******************************/

    #define MAIN_Fosc       11059200UL
    #define Baudrate0       9600UL

    /*********************************************************/


    /****************** 编 译 器 自 动 生 成 ， 用 户 请 勿 修 改
************************************/
```

```
#define BRT_Reload          (256 - MAIN_Fosc / 16 / Baudrate0)
//Calculate the timer1 reload value ar 1T mode


/*************************************************************/

//*************************************************************************
*******
//函数名：main(void)
//输入  ：无
//输出  ：无
//功能描述：实现串口每隔 2S 输出加速度传感器 ADXL345 的 X、Y、Z 数据
//*************************************************************************
*******
void  main(void)
{
   uchar devid;

   uart1_init();          //初始化串口
   Init_ADXL345();      //初始化 ADXL345
   devid=Single_Read_ADXL345(0X00);
   while(1)
   {
       if(devid == 0xE5)     //读出的数据为 0XE5,表示正确读出器件的
ID 号
       {
           Multiple_Read_ADXL345();           //连续读出数据，存储
在 BUF 中
           for(i=0; i<6; i++)
```

```
            {
                Uart1_TxByte(BUF[i]);       //串口输出 X、Y、Z 共 6 字
节数据
            }

        }

        for(i=0; i<200; i++)      //延时
        {
            Delay5ms();
        }
    }
}


//*************************************************************************
*******
//函数名：uart1_init(void)
//输入  ：无
//输出  ：无
//功能描述：串口初始化函数，通信参数为 9600 8 N 1
//*************************************************************************
*******
void  uart1_init(void)
{
    PCON |= 0x80;      //UART0 Double Rate Enable
    SCON = 0x50;       //UART0 set as 10bit , UART0 RX enable
    AUXR |=  0x01;         //UART0 使用 BRT
    AUXR |=  0x04;         //BRT set as 1T mode
```

```
    BRT = BRT_Reload;
    AUXR |=  0x10;        //start BRT

    ES  = 1;
    EA = 1;
}

//*********************************************************************
*******
//函数名：Uart1_TxByte(unsigned char dat)
//输入  ：需要发送的字节数据
//输出  ：无
//功能描述：从串口发送单字节数据
//*********************************************************************
*******
void Uart1_TxByte(unsigned char dat)
{
     B_TI = 0;
    SBUF = dat;
    while(!B_TI);
    B_TI = 0;
}
//*********************************************************************
*******
//函数名：Uart1_String(unsigned char code *puts)
//输入  ：字符串首地址
//输出  ：无
//功能描述：从串口发送字符串
//*********************************************************************
```

```
*******
    void Uart1_String(unsigned char code *puts)
    {
        for(; *puts != 0; puts++)
       {
            Uart1_TxByte(*puts);

       }
    }

    //*****************************************************************
*******
    //函数名：UART1_RCV (void)
    //输入  ：无
    //输出  ：无
    //功能描述：串口中断接收函数
    //*****************************************************************
*******
    void UART1_RCV (void) interrupt 4
    {
       if(RI)
       {
           RI = 0;
           RX0_Buffer[uart1_wr++] = SBUF;
           //if(++uart0_wr >= BUF_LENTH)   uart0_wr = 0;
           flag = 1;
       }

       if(TI)
```

```
        {
            TI = 0;
            B_TI = 1;
        }
    }



    //*****************************************************************
*******
    //函数名：Delay5us()
    //输入  ：无
    //输出  ：无
    //功能描述：产生延时 5us
    //*****************************************************************
********
    void Delay5us()
    {
         unsigned char a;
        for(a=26;a>0;a--);


    }

    //*****************************************************************
*******
    //函数名：Delay5ms()
    //输入  ：无
    //输出  ：无
```

```
    //功能描述：产生延时 5ms
    //*****************************************************************
********
    void Delay5ms()
    {
        unsigned char a,b,c;
        for(c=7;c>0;c--)
         for(b=168;b>0;b--)
             for(a=22;a>0;a--);
    }

    //*****************************************************************
*******
    //函数名：ADXL345_Start()
    //输入  ：无
    //输出  ：无
    //功能描述：产生 IIC 时序的起始信号
    //*****************************************************************
*******
    void ADXL345_Start()
    {
        SDA = 1;                    //拉高数据线
        SCL = 1;                    //拉高时钟线
        Delay5us();                 //延时,即保证 SDA 及 SCL 为高电平
        SDA = 0;                    //产生下降沿
        Delay5us();                 //延时，参考 ADXL345 的中文技术手
册 P18 的 t4.
        SCL = 0;                    //拉低时钟线
    }
```

```
    //**********************************************************************
*******
    //函数名：ADXL345_Stop()
    //输入  ：无
    //输出  ：无
    //功能描述：产生 IIC 时序的停止信号
    //**********************************************************************
*******
    void ADXL345_Stop()
    {
        SDA = 0;                    //拉低数据线
        SCL = 1;                    //拉高时钟线
        Delay5us();                  //延时参考 ADXL345 的中文技术手册
P18 的 t8.
        SDA = 1;                    //产生上升沿
        Delay5us();                 //延时,普通延时
    }

    //**********************************************************************
*******
    //函数名：ADXL345_SendACK(bit ack)
    //输入  ：ack(0:ACK 1:NAK)
    //输出  ：无
    //功能描述：发送应答信号
    //**********************************************************************
********
    void ADXL345_SendACK(bit ack)
    {
```

```
    SDA = ack;                  //写应答信号
    SCL = 1;                    //拉高时钟线
    Delay5us();                 //延时
    SCL = 0;                    //拉低时钟线
    Delay5us();                 //延时
}

//******************************************************************
*******
//函数名：ADXL345_RecvACK()
//输入  ：无
//输出  ：ack(0:ACK 1:NAK)
//功能描述：接收应答信号
//******************************************************************
********
bit ADXL345_RecvACK()
{
    SCL = 1;                    //拉高时钟线
    Delay5us();                 //延时
    CY = SDA;                   //读应答信号
    SCL = 0;                    //拉低时钟线
    Delay5us();                 //延时

    return CY;
}

//******************************************************************
*******
```

```
    //函数名：ADXL345_SendByte(BYTE dat)
    //输入  ：发送的数据字节信息
    //输出  ：无
    //功能描述：向 IIC 总线发送一个字节数据
    //*****************************************************************
*******
    void ADXL345_SendByte(BYTE dat)
    {
        BYTE i;

        for (i=0; i<8; i++)          //8 位计数器
        {
            dat <<= 1;              //移出数据的最高位
            SDA = CY;              //送数据口
            SCL = 1;               //拉高时钟线
            Delay5us();            //延时
            SCL = 0;               //拉低时钟线
            Delay5us();            //延时
        }
        ADXL345_RecvACK();
    }

    //*****************************************************************
*******
    //函数名：ADXL345_RecvByte()
    //输入  ：无
    //输出  ：返回单字节信息
    //功能描述：从 IIC 总线接收一个字节数据
    //*****************************************************************
```

```
******
    BYTE ADXL345_RecvByte()
    {
        BYTE i;
        BYTE dat = 0;

        SDA = 1;                    //使能内部上拉,准备读取数据,
        for (i=0; i<8; i++)          //8 位计数器
        {
            dat <<= 1;
            SCL = 1;                //拉高时钟线
            Delay5us();              //延时
            dat |= SDA;              //读数据
            SCL = 0;                //拉低时钟线
            Delay5us();              //延时
        }
        return dat;
    }

    //*****************************************************************
*******
    //函数名：Single_Write_ADXL345(uchar REG_Address,uchar REG_data)
    //输入  ：寄存器地址及要写入的字节数据
    //输出  ：无
    //功能描述：向指定的寄存器写一个字节数据，过程请参考中文手册 P17
    //*****************************************************************
*******
    void Single_Write_ADXL345(uchar REG_Address,uchar REG_data)
    {
```

```
        ADXL345_Start();                    //起始信号
        ADXL345_SendByte(SlaveAddress);     //发送设备地址+写信号
        ADXL345_SendByte(REG_Address);      //内部寄存器地址,请参考中文
pdf22 页
        ADXL345_SendByte(REG_data);         //内部寄存器数据,请参考中文
pdf22 页
        ADXL345_Stop();                     //发送停止信号
    }

    //**************************************************************
*******
    //函数名：Single_Read_ADXL345(uchar REG_Address)
    //输入  ：要读取寄存器数据的地址
    //输出  ：寄存器数据
    //功能描述：从 IIC 总线读取指定寄存器字节数据，过程请参考中文手册
P17
    //**************************************************************
*******
    uchar Single_Read_ADXL345(uchar REG_Address)
    {  uchar REG_data;
        ADXL345_Start();                        //起始信号
        ADXL345_SendByte(SlaveAddress);         //发送设备地址+写信
号
        ADXL345_SendByte(REG_Address);          //发送存储单元地址,
从 0 开始
        ADXL345_Start();                        //起始信号
        ADXL345_SendByte(SlaveAddress+1);       //发送设备地址+读信
号
        REG_data=ADXL345_RecvByte();            //读出寄存器数据
```

```
    ADXL345_SendACK(1);
    ADXL345_Stop();                              //停止信号
     return REG_data;
}

//*****************************************************************
*******
//函数名：Multiple_read_ADXL345(void)
//输入  ：无
//输出  ：无
//功能描述：连续读出 ADXL345 内部加速度数据，地址范围 0x32~0x37，过
程请参考中文手册 P17
//          把从 ADXL345 读取的 X,Y,Z 共 6 个字节数据存放在 BUF[]缓存
中
//*****************************************************************
********
void Multiple_read_ADXL345(void)
{   uchar i;
    ADXL345_Start();                          //起始信号
    ADXL345_SendByte(SlaveAddress);           //发送设备地址+写信
号
    ADXL345_SendByte(0x32);                   //发送存储单元地址，
从 0x32 开始
    ADXL345_Start();                          //起始信号
    ADXL345_SendByte(SlaveAddress+1);         //发送设备地址+读信
号
    for (i=0; i<6; i++)                          //连续读取 6 个地
址数据，存储中 BUF
    {
```

```
            BUF[i] = ADXL345_RecvByte();          //BUF[0]存储 0x32 地
址中的数据
            if (i == 5)
            {
               ADXL345_SendACK(1);                //最后一个数据需要回
NOACK
            }
            else
            {
              ADXL345_SendACK(0);                 //回应 ACK
            }
        }
         ADXL345_Stop();                           //停止信号
         Delay5ms();
    }

    //**************************************************************
*******
    //函数名：Init_ADXL345()
    //输入  ：无
    //输出  ：无
    //功能描述：初始化 ADXL345，过程请参考 ADCL345 快速入门 P3 及 P6
    //**************************************************************
*******
    void Init_ADXL345()
    {
       Single_Write_ADXL345(0x31,0x0b);   //测量范围,正负 16g，13 位模
式
       Single_Write_ADXL345(0x2C,0x0A);   //速率设定为 100  参考 pdf13
```

```
页
        Single_Write_ADXL345(0x2D,0x08);    //选择电源模式为测量模式，参
考 pdf24 页
        Single_Write_ADXL345(0x2E,0x80);    //使能 DATA_READY 中断
        Single_Write_ADXL345(0x1E,0xfa);    //X 偏移量 根据测试传感器的
状态写入 pdf29 页
        Single_Write_ADXL345(0x1F,0xfd);    //Y 偏移量 根据测试传感器的
状态写入 pdf29 页
        Single_Write_ADXL345(0x20,0x06);    //Z 偏移量 根据测试传感器的
状态写入 pdf29 页
    }
```

4.3 实验运行效果

通过串口每隔 2S 输出加速度值：

一共 6 个字节：DATAX0 DATAX1 DATAY0 DATAY1 DATAZ0 DATAZ1

结果为：A1 FF 45 00 74 05，分别对应：DATAX0 DATAX1 DATAY0 DATAY1 DATAZ0 DATAZ1

当你摇动 ADXL345 所在的板子时候，相应的值会产生变化，X、Y、Z 方向请参考中文手册 P34.

串口输出结果如图 4.1 所示。

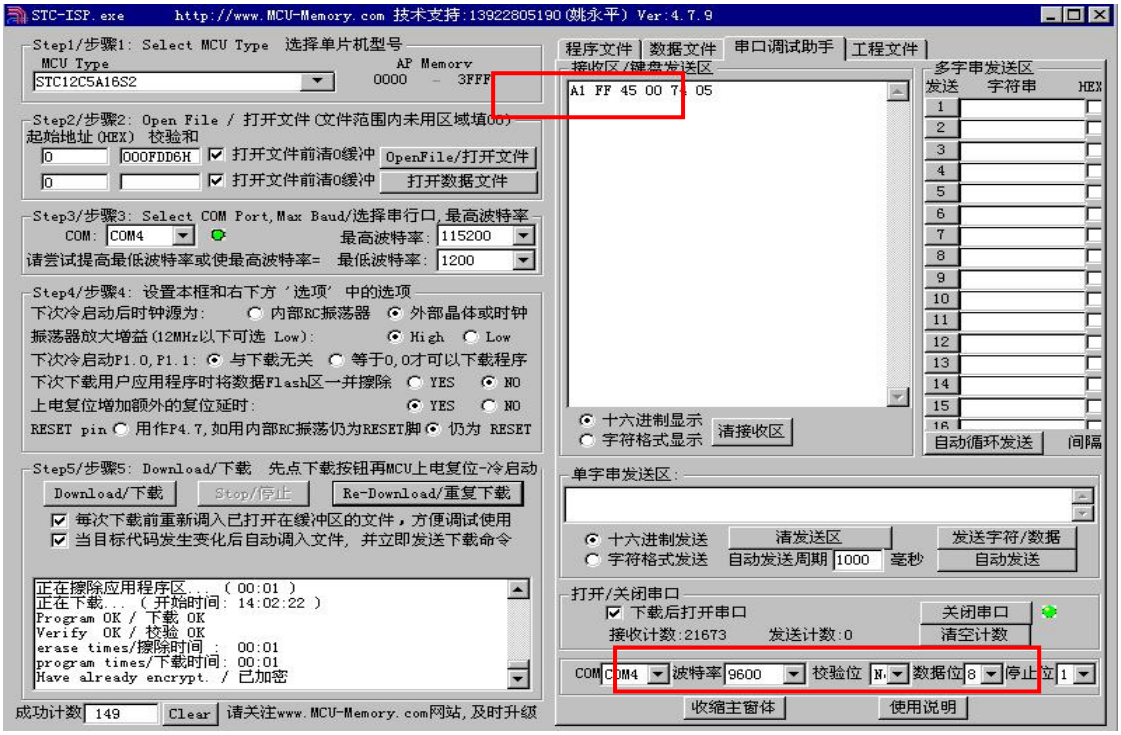


图 4.1　3D 加速度运行串口输出信息

注意：调试的时候串口参数为 9600 8 N 1.

实训项目七 结露传感器应用

一、实验目的

学习结露传感器的性能及使用方式

二、实验设备

硬件：IOT-L01-05 型物联网综合实验箱 1 台，带结露传感器，串口线。

软件：Keil，STC_ISP_V479，串口调试工具。

三、实验原理

芯片手册：配套光盘\附件\芯片手册\结露传感器

源码路径:配套光盘\源代码\传感器原理及应用\实验七 结露传感器实验

hex 文件路径：配套光盘\源代码\传感器原理及应用\可执行程序\JieLu.hex

3.1 结露传感器的介绍

结露传感器是正特性开关型元件，对低湿不敏感而仅对高湿敏感，可在直流电压下工作，响应速度快，高可靠性，广泛用于仓储、气象等行业，其实物如图 3.1.

图 3.1 结露传感器实物

供电电压：0.8V DC 安全电压

工作温度范围：1~80℃

工作湿度范围：1~100%RH

结露测试范围：94~100%RH

湿度与阻值的特性曲线图，如图 3.2。

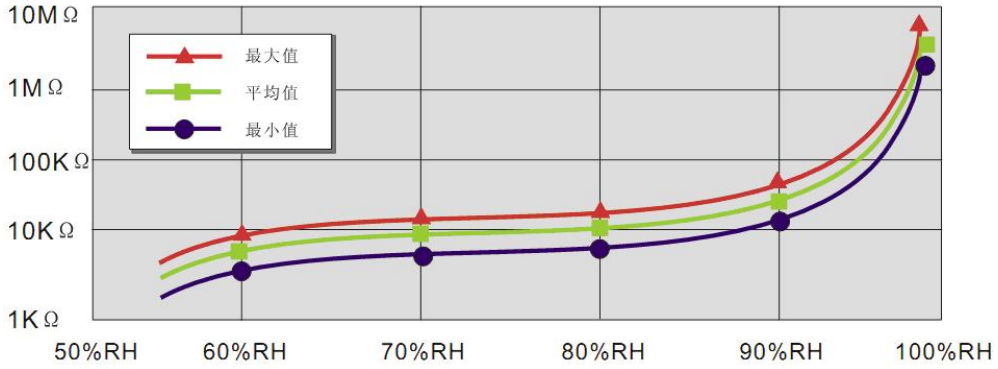


图 3.2 湿度与阻值的特性

3.2　结露传感器电路原理图

结露传感器电路如图 3.3 所示。

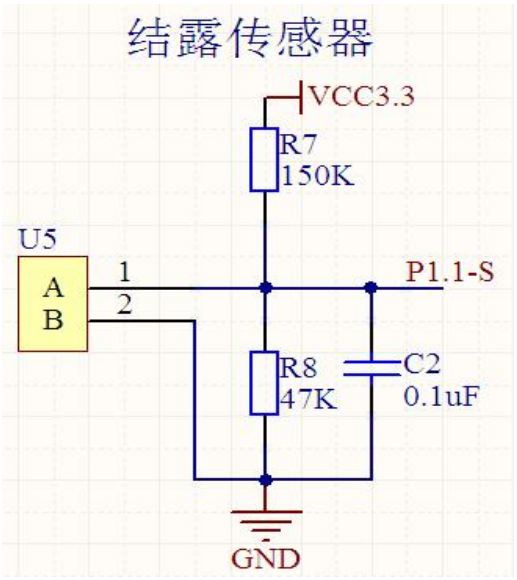


图 3.3　结露传感器原理图

从图 3.3 所示的电路图，我们可以计算出加在结露传感器 U5 上的电压为：

3.3 V* 47 K/（150K + 47K）= 0.787V，小于结露传感器的安全电压 0.8V，由特性参数表中可知，75% RH 25℃条件下，结露传感器电阻为 10K 欧姆，此时 ADC1(P1.1)为 0.171V。当湿度增加时，电阻增大，ADC1(P1.1)增大，选定一个临界值(根据实际情况选择)，比如 0.171V ，此时 AD 读数为 0.171 / 3.3＊1024 = 53，当 AD 采集的数值大于 53 时表明有结露由于实际的湿度测量还需要标准的条件来测试，所以我们这里的实验仅作为湿度大小的读取观察。

四、实验过程

4.1　编写实验源代码文件

4.1.1　打开 KeilC51 集成开发环境，创建 JieLu 工程。

4.1.2　配置 JieLu 工程的参数。

4.1.3　编写 JieLu 源代码。

4.1.4　编译工程文件，生成可执行 JieLu.hex 文件。

4.1.5　将可执行 JieLu.hex 文件下载进 STC12C5A16S2 单片机底板。

4.1.6　将结露传感器插入下载完程序的单片机底板。

4.1.7　将插入结露传感器的节点重新上电。

4.2　实验源代码解析

Main.c **源代码**

```
/*************************************************************/
//晶振频率：11.0592MHz
//文件名　：Main.c
//功能说明：结露感器读取实验
//制作　　：www.frotech.com
//技术支持：020-22883196 QQ:
//变更记录：2013.05.02
//变更内容：新建造
/*************************************************************/

#include <STC12C5A60S2.h>


#define      BUF_LENTH  128     //定义串口接收缓冲长度
#define     uint unsigned int
#define     uchar unsigned char
unsigned char    uart1_wr;      //写指针
unsigned char    uart1_rd;      //读指针
unsigned char    xdata RX0_Buffer[BUF_LENTH];    //接收缓冲
unsigned char flag;
unsigned char i;
bit       B_TI;              //发送完成标志
sbit  P1_0 = P1^0;           //定义 P1.0 端口
//                           7       6       5       4          3
```

```
2    1    0    Reset Value
    //sfr ADC_CONTR = 0xBC;     ADC_POWER   SPEED1   SPEED0   ADC_FLAG
ADC_START CHS2 CHS1 CHS0 0000,0000 //AD 转换控制寄存器
    #define ADC_OFF()    ADC_CONTR = 0
    #define ADC_ON       (1 << 7)
    #define ADC_90T      (3 << 5)
    #define ADC_180T (2 << 5)
    #define ADC_360T (1 << 5)
    #define ADC_540T 0
    #define ADC_FLAG (1 << 4)       //软件清 0
    #define ADC_START    (1 << 3)       //自动清 0
    #define ADC_CH0      0
    #define ADC_CH1      1
    #define ADC_CH2      2
    #define ADC_CH3      3
    #define ADC_CH4      4
    #define ADC_CH5      5
    #define ADC_CH6      6
    #define ADC_CH7      7

    uint adc10_start(uchar channel);
    void  uart1_init(void);
    void Uart1_TxByte(unsigned char dat);
    void Uart1_String(unsigned char code *puts);
    void delay_ms(unsigned char ms);

    /*************** 用户定义参数 ******************************/

    #define MAIN_Fosc       11059200UL
```

```
#define Baudrate0        9600UL

/*************************************************************/


/*****************  编 译 器 自 动 生 成 ， 用 户 请 勿 修 改 ***********************************/

#define BRT_Reload          (256 - MAIN_Fosc / 16 / Baudrate0)
//Calculate the timer1 reload value ar 1T mode


/*************************************************************/

//*************************************************************************
//函数名：main(void)
//输入  ：无
//输出  ：无
//功能描述：当湿度变大的时候，结露传感器的电阻值变大，则 ADC1 的值也变大
//          在这里我们以 ADC1 = 150 作为判定结露的临界值，即大于 150 时有结露
//          并且输出字符串“JieLu",小于 150 则不结露

//*************************************************************************
void   main(void)
{
```

```
    uint   j;
    uart1_init();//初始化串口
    P1ASF = (1 << ADC_CH1);   //STC12C5A16S2 系列模拟输入(AD)选择
ADC1(P1.1)
    ADC_CONTR = ADC_360T | ADC_ON;
    while(1)
    {
        delay_ms(500);
        j = adc10_start(1);     //(P1.1)ADC1 转换
        if(j > 0x96)     //当 ADC1 的值大于 150 时我们判定有结露，且输
出字符串“JieLu”
        {
           Uart1_String("JieLu");
        }
        Uart1_TxByte('A');     //以下为按照十进制输出 ADC 值
        Uart1_TxByte('D');
        Uart1_TxByte('1');
        Uart1_TxByte('=');
        Uart1_TxByte(j/1000 + '0');
        Uart1_TxByte(j%1000/100 + '0');
        Uart1_TxByte(j%100/10 + '0');
        Uart1_TxByte(j%10 + '0');
        Uart1_TxByte(0x0d);
        Uart1_TxByte(0x0a);
    }
}
//*****************************************************************
*******
//函数名：adc10_start(uchar channel)
```

```
    //输入   : ADC 转换的通道
    //输出   : ADC 值
    //功能描述：ADC 转换
    //*****************************************************************
*******
    uint   adc10_start(uchar channel)  //channel = 0~7
    {
       uint   adc;
       uchar  i;

       ADC_RES = 0;
       ADC_RESL = 0;

       ADC_CONTR = (ADC_CONTR & 0xe0) | ADC_START | channel;
       i = 250;
       do{
           if(ADC_CONTR & ADC_FLAG)
           {
               ADC_CONTR &= ~ADC_FLAG;
               adc = (uint)ADC_RES;
               adc = (adc << 2) | (ADC_RESL & 3);
               return adc;
           }
       }while(--i);
       return 1024;
    }

    //*****************************************************************
*******
```

```
    //函数名：uart1_init(void)
    //输入  ：无
    //输出  ：无
    //功能描述：串口初始化函数，通信参数为9600 8 N 1
    //*****************************************************************
*******
    void  uart1_init(void)
    {
       PCON |= 0x80;     //UART0 Double Rate Enable
       SCON = 0x50;      //UART0 set as 10bit , UART0 RX enable
       AUXR |=  0x01;        //UART0 使用BRT
       AUXR |=  0x04;        //BRT set as 1T mode
       BRT = BRT_Reload;
       AUXR |=  0x10;        //start BRT

       ES  = 1;
       EA = 1;
    }
    //*****************************************************************
*******
    //函数名：Uart1_TxByte(unsigned char dat)
    //输入  ：需要发送的字节数据
    //输出  ：无
    //功能描述：从串口发送单字节数据
    //*****************************************************************
*******
    void Uart1_TxByte(unsigned char dat)
    {
        B_TI = 0;
```

```
    SBUF = dat;
    while(!B_TI);
    B_TI = 0;
}
//*********************************************************************
*******
//函数名：Uart1_String(unsigned char code *puts)
//输入  ：字符串首地址
//输出  ：无
//功能描述：从串口发送字符串
//*********************************************************************
*******
void Uart1_String(unsigned char code *puts)
{
    for(; *puts != 0; puts++)
    {
        Uart1_TxByte(*puts);

    }
}

//*********************************************************************
*******
//函数名：UART1_RCV (void)
//输入  ：无
//输出  ：无
//功能描述：串口中断接收函数
//*********************************************************************
*******
```

```
void UART1_RCV (void) interrupt 4
{
   if(RI)
   {
      RI = 0;
      RX0_Buffer[uart1_wr++] = SBUF;
      //if(++uart0_wr >= BUF_LENTH)   uart0_wr = 0;
      flag = 1;
   }

   if(TI)
   {
      TI = 0;
      B_TI = 1;
   }
}

void delay_ms(unsigned char ms)
{
    unsigned int i;
   do{
       i = MAIN_Fosc /1400;
      while(--i);
   }while(--ms);
}
```

4.3 实验运行效果

如图 4.1 所示，当 ADC1 的值大于 150 时，我们判定有结露，否则无结露。

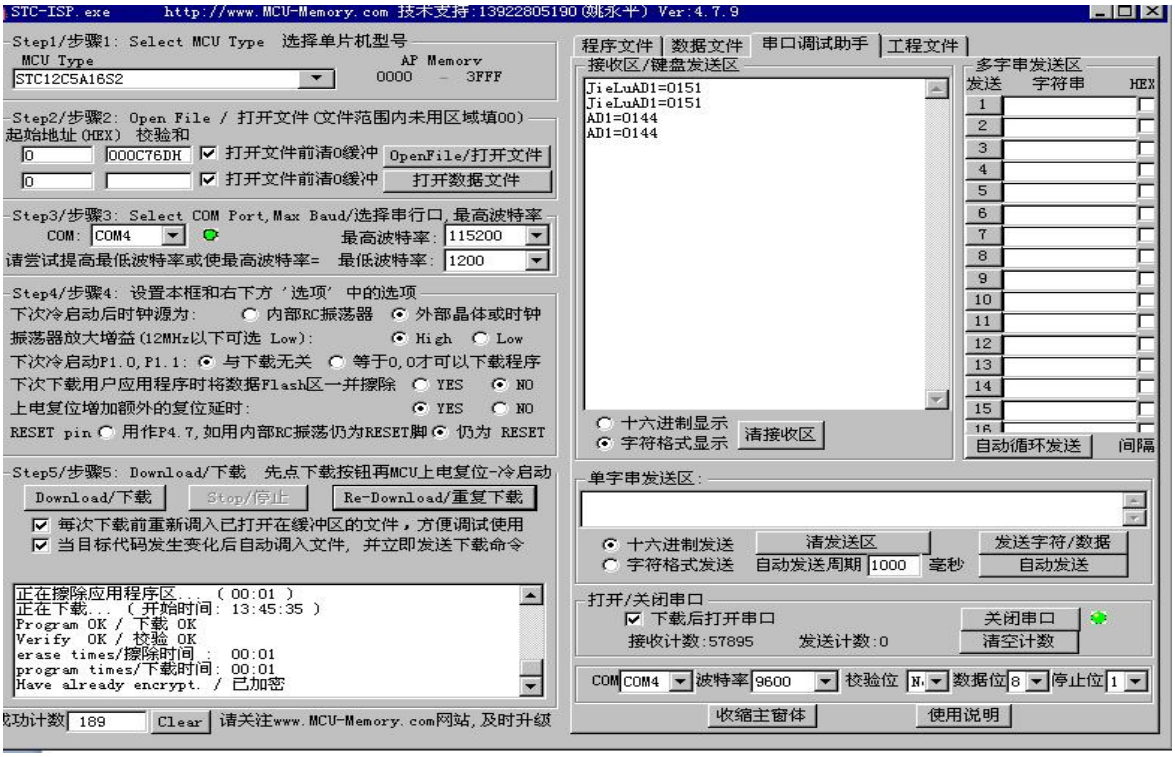


图 4.1　结露传感器运行时串口输出信息

注意：串口参数为 9600 8 N 1

实训项目八 酒精检测传感器应用

一、实验目的

学习酒精检测传感器 MQ-3 的使用方法

二、实验设备

硬件：IOT-L01-05 型物联网综合实验箱 1 台，带酒精检测传感器 MQ-3，串口线。

软件：Keil，STC_ISP_V479，串口调试工具。

三、实验原理

芯片手册：配套光盘\附件\芯片手册\酒精检测传感器

源码路径：配套光盘\源代码\传感器原理及应用\实验八 酒精检测传感器实验

hex 文件路径：配套光盘\源代码\传感器原理及应用\可执行程序\JiuJin.hex

3.1 MQ-3 的介绍

MQ-3气体传感器所使用的气敏材料是在清洁空气中电导率较低的二氧化锡(SnO_2)。当传感器所处环境中存在酒精蒸汽时，传感器的电导率随空气中酒精气体浓度的增加而增大。使用简单的电路即可将电导率的变化转换为与该气体浓度相对应的输出信号。

MQ-3气体传感器对酒精的灵敏度高，可以抵抗汽油、烟雾、水蒸气的干扰。这种传感器可检测多种浓度酒精气氛，是一款适合多种应用的低成本传感器。

元器件结构外形如图 3.1

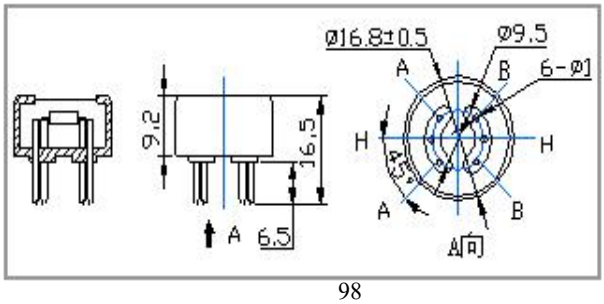
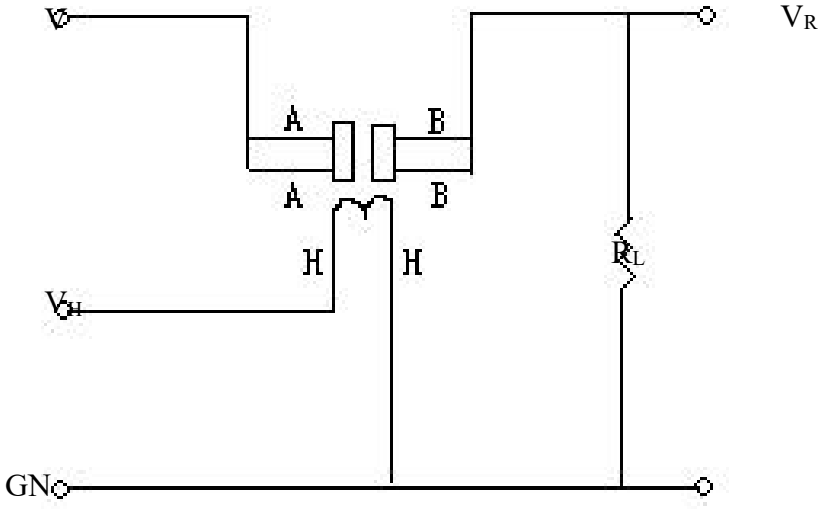


图 3.1　酒精传感器形型



D

图3.2　　酒精传感器基本测试电路

基本测试回路

图3.2是酒精传感器的基本测试电路。该传感器需要施加2个电压：加热器电压（V_H）和测试电压（V_C）。其中 V_H用于为传感器提供特定的工作温度。V_C 则是用于测定与传感器串联的负载电阻（R_L）上的电压（V_{RL}）。这种传感器具有轻微的极性，V_C需用直流电源。在满足传感器电性能要求的前提下，V_C和V_H可以共用同一个电源电路。为更好利用传感器的 性能，需要选择恰当的RL值。

酒精传感器的技术指标如表1。

表1　酒精传感器技术指标

产品型号			MQ-4
产品类型			半导体气敏元件
标准封装			胶木（黑胶木）
检测气体			酒精蒸汽
检测浓度			0.04-4mg/L酒精
标准电路条件	回路电压	V_C	≤24V　DC
	加热电压	V_H	5.0V±0.2V ACorDC
	负载电阻	R_L	可调
标准测试条件下	加热电阻	R_H	31Ω±3Ω（室温）
	加热功耗	P_H	≤900mW
	敏感体表面电阻	R_S	2KΩ-20KΩ(in 0.4mg/L酒精)

气敏元件特性	灵敏度		Rs(in air)/Rs(0.4mg/L酒精)≥5
	浓度斜率		≤0.6(R_{300ppm}/R_{100ppm}酒精)
标准测试条件	温度、湿度		20℃±2℃；65%±5%RH
	标准测试电路		Vc:5.0V±0.1V；V_H: 5.0V±0.1V
	预热时间		不少于48小时

酒精传感器灵敏度特性图 3.3 所示。

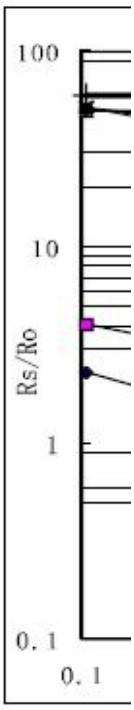


图 3.3　酒精传感器灵敏度特性图

3.2 MQ-3 酒精传感器的电路原理图

MQ-3 酒精传感器的电路原理如图 3.4 所示。

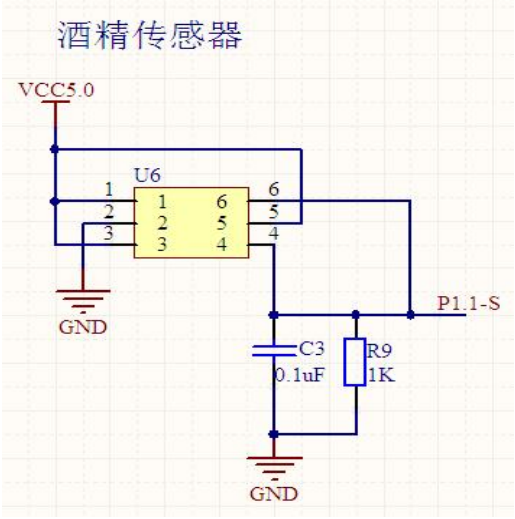


图 4　酒精传感器电路

其中 U6(MQ-3)的 PIN5 与 PIN2 为加热端，对应测试回路的 H 端；PIN1、PIN3、PIN4、PIN6 为检测回路；MQ-3 传感器的供电电压 Vc 和加热电压 Vh 都为 5V，负载电阻 R9 为 1K 欧姆。从技术指标表中可知，在 0.4mg/L 酒精中，传感器电阻 Rs 为 2K~20K，取 Rs = 12K。假设检测到酒精浓度为 10mg/L 时报警，由灵敏度特性曲线可知灵敏度为 0.12，MQ3 电阻值为 12K * 0.12 = 1.44K（Rs / Ro = 灵敏度，其中 Ro 为传感器在 0.4mg/L 酒精时的电阻值）,ADC1(P1.1) = 5V * 1K / (1K + 1.44K) = 2.00V，AD 读数为 2.00 / 3.3 * 1024 = 620，当 AD 采集的数值大于 620 时表明检测到酒精。

四、实验过程

4.1　编写实验源代码文件

4.1.1　打开 KeilC51 集成开发环境，创建 JiuJin 工程。

4.1.2　配置 JiuJin 工程的参数。

4.1.3　编写 JiuJin 源代码。

4.1.4　编译工程文件，生成可执行 JiuJin.hex 文件。

4.1.5　将可执行 JiuJin.hex 文件下载进 STC12C5A16S2 单片机底板。

4.1.6　将酒精传感器插入下载完程序的单片机底板。

4.1.7　将插入酒精传感器的节点重新上电。

4.2 实验源代码解析

Main.c 源代码

```
/*************************************************************/
//晶振频率：11.0592MHz
//文件名  ：Main.c
//功能说明：酒精检测感器读取实验
//制作    ：www.frotech.com
//技术支持：020-22883196 QQ:
//变更记录: 2013.05.02
//变更内容: 新建造
/*************************************************************/

#include <STC12C5A60S2.h>


#define      BUF_LENTH  128    //定义串口接收缓冲长度
#define     uint unsigned int
#define     uchar unsigned char
unsigned char    uart1_wr;       //写指针
unsigned char    uart1_rd;       //读指针
unsigned char    xdata RX0_Buffer[BUF_LENTH];    //接收缓冲
unsigned char flag;
unsigned char i;
bit       B_TI;                 //发送完成标志
sbit  P1_0 = P1^0;              //定义 P1.0 端口
//                          7       6       5       4         3
2    1    0   Reset Value
```

```
    //sfr ADC_CONTR = 0xBC;     ADC_POWER SPEED1 SPEED0 ADC_FLAG
ADC_START CHS2 CHS1 CHS0 0000,0000 //AD 转换控制寄存器
    #define ADC_OFF()    ADC_CONTR = 0
    #define ADC_ON       (1 << 7)
    #define ADC_90T      (3 << 5)
    #define ADC_180T (2 << 5)
    #define ADC_360T (1 << 5)
    #define ADC_540T 0
    #define ADC_FLAG (1 << 4)   //软件清 0
    #define ADC_START    (1 << 3)   //自动清 0
    #define ADC_CH0      0
    #define ADC_CH1      1
    #define ADC_CH2      2
    #define ADC_CH3      3
    #define ADC_CH4      4
    #define ADC_CH5      5
    #define ADC_CH6      6
    #define ADC_CH7      7

    uint adc10_start(uchar channel);
    void  uart1_init(void);
    void Uart1_TxByte(unsigned char dat);
    void Uart1_String(unsigned char code *puts);
    void delay_ms(unsigned char ms);

    /*************** 用户定义参数 ******************************/

    #define MAIN_Fosc       11059200UL
    #define Baudrate0       9600UL
```

```
    /*********************************************************/


    /****************** 编译器自动生成，用户请勿修改
************************************/

    #define BRT_Reload          (256 - MAIN_Fosc / 16 / Baudrate0)
    //Calculate the timer1 reload value ar 1T mode


    /*********************************************************/

    //*************************************************************
*******
    //函数名：main(void)
    //输入  ：无
    //输出  ：无
    //功能描述：检测酒精传感器 MQ-3 的 ADC 值，在这里采用 ADC1(P1.1),当
ADC1 的值
    //          大于 620 的时候我们判定有酒精，并且输出字符串“JiuJin”.
    //*************************************************************
*******
    void   main(void)
    {
       uint   j;
       uart1_init();                        //初始化串口
       P1ASF = (1 << ADC_CH1);        //STC12C5A16S2 系列模拟输入(AD)
选择 ADC1(P1.1)
```

```
        ADC_CONTR = ADC_360T | ADC_ON;
        while(1)
        {
            delay_ms(500);
            j = adc10_start(1);            //(P1.1)ADC1 转换
           if(j > 0x26C)              //    当 ADC1 的值大于 620 时我们判定
有酒精，且输出字符串“JiuJin”
           {
              Uart1_String("JiuJin");
           }
           Uart1_TxByte('A');          //以下为按照十进制输出 ADC 值
           Uart1_TxByte('D');
           Uart1_TxByte('1');
           Uart1_TxByte('=');
           Uart1_TxByte(j/1000 + '0');
           Uart1_TxByte(j%1000/100 + '0');
           Uart1_TxByte(j%100/10 + '0');
           Uart1_TxByte(j%10 + '0');
           Uart1_TxByte(0x0d);
           Uart1_TxByte(0x0a);
        }
    }
    //*************************************************************
*******
    //函数名：adc10_start(uchar channel)
    //输入  ：ADC 转换的通道
    //输出  ：ADC 值
    //功能描述：ADC 转换
```

```
    //*************************************************************
*******
    uint   adc10_start(uchar channel)  //channel = 0~7
    {
       uint   adc;
       uchar  i;

       ADC_RES = 0;
       ADC_RESL = 0;

       ADC_CONTR = (ADC_CONTR & 0xe0) | ADC_START | channel;
       i = 250;
       do{
           if(ADC_CONTR & ADC_FLAG)
           {
               ADC_CONTR &= ~ADC_FLAG;
               adc = (uint)ADC_RES;
               adc = (adc << 2) | (ADC_RESL & 3);
               return adc;
           }
       }while(--i);
       return 1024;
    }

    //*************************************************************
*******
    //函数名：uart1_init(void)
    //输入  ：无
    //输出  ：无
```

```
    //功能描述：串口初始化函数，通信参数为9600 8 N 1
    //*****************************************************************
*******
    void  uart1_init(void)
    {
       PCON |= 0x80;     //UART0 Double Rate Enable
       SCON = 0x50;      //UART0 set as 10bit , UART0 RX enable
       AUXR |=  0x01;        //UART0 使用BRT
       AUXR |=  0x04;        //BRT set as 1T mode
       BRT = BRT_Reload;
       AUXR |=  0x10;        //start BRT

       ES  = 1;
       EA = 1;
    }
    //*****************************************************************
*******
    //函数名：Uart1_TxByte(unsigned char dat)
    //输入  ：需要发送的字节数据
    //输出  ：无
    //功能描述：从串口发送单字节数据
    //*****************************************************************
*******
    void Uart1_TxByte(unsigned char dat)
    {
        B_TI = 0;
       SBUF = dat;
       while(!B_TI);
       B_TI = 0;
```

```
    }
    //*****************************************************************
*******
    //函数名：Uart1_String(unsigned char code *puts)
    //输入  ：字符串首地址
    //输出  ：无
    //功能描述：从串口发送字符串
    //*****************************************************************
*******
    void Uart1_String(unsigned char code *puts)
    {
        for(; *puts != 0; puts++)
       {
            Uart1_TxByte(*puts);

       }
    }

    //*****************************************************************
*******
    //函数名：UART1_RCV (void)
    //输入  ：无
    //输出  ：无
    //功能描述：串口中断接收函数
    //*****************************************************************
*******
    void UART1_RCV (void) interrupt 4
    {
       if(RI)
```

```
    {
        RI = 0;
        RX0_Buffer[uart1_wr++] = SBUF;
        //if(++uart0_wr >= BUF_LENTH)   uart0_wr = 0;
        flag = 1;
    }
    if(TI)
    {
        TI = 0;
        B_TI = 1;
    }
}
void delay_ms(unsigned char ms)
{
     unsigned int i;
    do{
         i = MAIN_Fosc /1400;
        while(--i);
    }while(--ms);
}
```

4.3 实验运行效果

一般情况下 ADC1 的值为 250 左右，当我们把酒精靠近 MQ-3 酒精检测传感器的时候，那么 ADC1 会很快上升，当 ADC1 的值超过 620 的时候，我们判定有酒精靠近，输出字符串“JiuJin”。运行时串口输出信息如图 4.1。

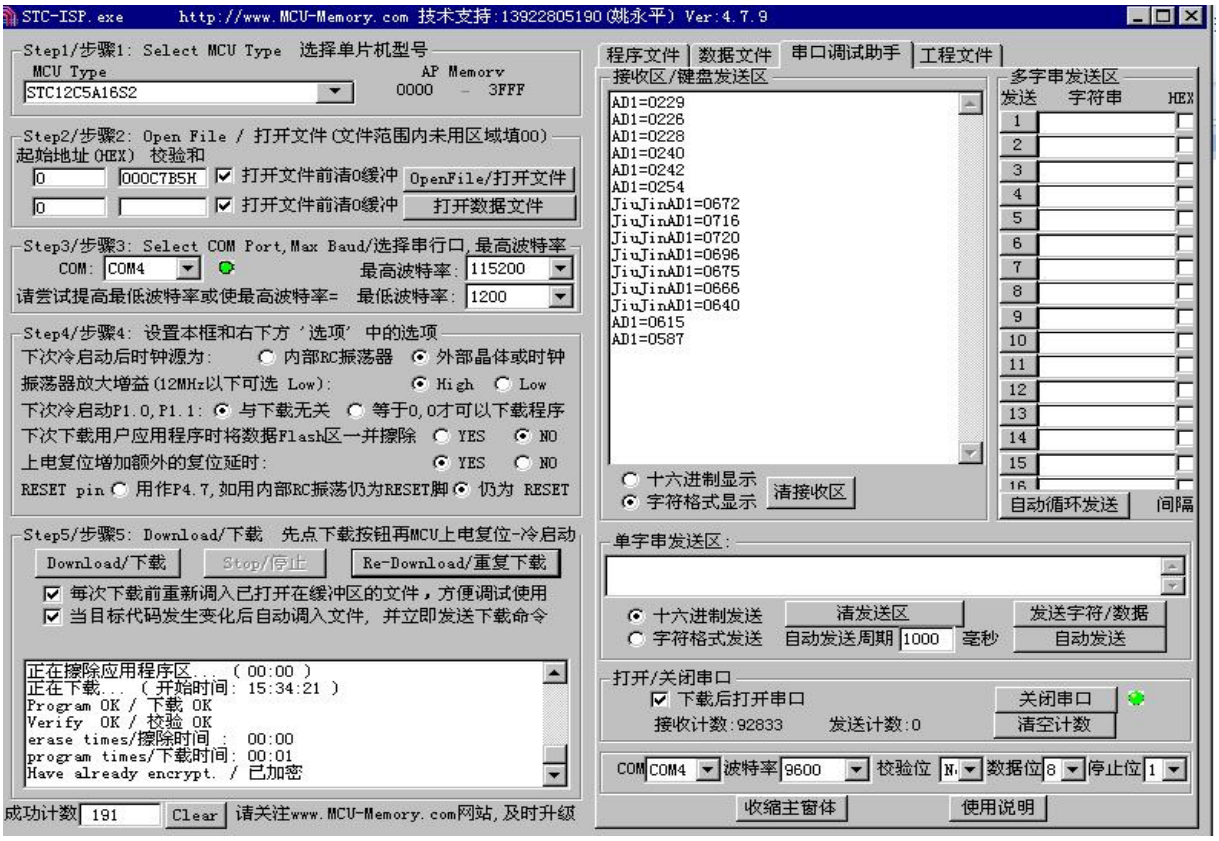


图 4.1　酒精传感器运行时串口信息

注意：不要把酒精洒在电路板或者传感器上，免得腐蚀相关电路，把酒精轻轻靠近即可。

串口参数：9600　8　N　1

实训项目九 人体感应传感器应用

一、实验目的

学习人体感应传感器的原理和使用方式。

注意：人体感应传感器模块非实验箱标配传感器模块，如果学生使用的实验箱中没有该模块，请跳过此实验。

二、实验设备

硬件：IOT-L01-05 型物联网综合实验箱 1 台，带人体感应传感器，串口线。

软件：Keil，STC_ISP_V479，串口调试工具。

三、实验原理

芯片手册：配套光盘\附件\芯片手册\人体感应传感器

源码路径：配套光盘\源代码\传感器原理及应用\实验九 人体感应传感器

hex 文件路径：配套光盘\源代码\传感器原理及应用\可执行程序\RenTi.hex

3.1 人体感应传感器介绍

人体感应传感器主要是由 RE200B 热释电红外传感器、红外热释电处理芯片 BISS0001、 菲涅尔透镜-球面三部分构成，下面分别介绍。

3.1.1 RE200B 热释电红外传感器

RE200B 热释电红外传感器是一种高灵敏度的探测元件，能以非接触式的形式检测出人体辐射的红外线能量的变化，并将其转换成电压信号输出。该传感器采用热释电材料随温 度变化的特性探测红外辐射，采用双灵敏元互补方法抑制温度变化产生的干扰，提高传 感器的工作稳定性。RE200B 广泛应用于红外线技术领域，如通讯、医疗、安保、探测等领域。实物如图 3.1 所示：

图 3.1 RE200B 热释电红外传感器

RE200B 热释电红外传感器技术参数如表 1。

表 1　RE200B 技术参数

灵敏元面积	2.0×1.0mm^2
基片材料	硅
基片厚度	0.5mm
工作波长	5-14μm
平均透过率	>75%
输出信号	>2.5V
	(420° C k 黑体 1Hz 调制频率 0.3-3.0Hz 带宽 72.5db 增益)
噪声	<200mv
	(mVp-p) (25° C)
平衡度	<20%
工作电压	2.2-15V
工作电流	8.5-24μA
	（VD=10V, RS=47kΩ,25° C）
源极电压	0.4-1.1V
	（VD=10V, RS=47KΩ,25° C）
工作温度	-30° C—+70° C
保存温度	-40° C—+80° C
视场	138°　×125°

3.1.2 红外热释电处理芯片 BISS0001

BISS0001 是一款具有较高性能的传感信号处理集成电路，它配以热释电红外传感器和少量外接元器件构成被动式的热释电红外开关。它能自动快速开启各类白炽灯、荧光灯、蜂鸣器、自动门、电风扇、烘干机和自动洗手池等装置，特别适用于企业、宾馆、商场、库房及家庭的过道、走廊等敏感区域，或用于安全区域的自动灯光、照明和报警系统。

特 点：

*CMOS 工艺

*数模混合

*具有独立的高输入阻抗运算放大器

*内部的双向鉴幅器可有效抑制干扰

*内设延迟时间定时器和封锁时间定时器

管脚说明如表 2 所示。

表 2　BISS0001 引脚说明

引脚	名称	I/O	功能说明
1	A	I	可重复触发和不可重复触发选择端。当 A 为“1”时， 允许重复触发；反之，不可重复触发
2	VO	O	控制信号输出端。由 VS 的上跳变沿触发，使 Vo 输出从 低电平跳变到高电平时视为有效触发。在输出延迟时间 Tx 之外和无 VS 的上跳变时，Vo 保持低电平状态。
3	RR1	- -	输出延迟时间 Tx 的调节端
4	RC1	- -	输出延迟时间 Tx 的调节端
5	RC2	- -	触发封锁时间 Ti 的调节端
6	R	-	触发封锁时间 Ti 的调节端

	R2	-	
7	VSS	- -	工作电源负端
8	VRF	I	参考电压及复位输入端。通常接 VDD，当接“0”时可 使定时器复位
9	VC	I	触发禁止端。当 Vc<VR 时禁止触发；当 Vc>VR 时允许 触发(VR≈0.2VDD)
10	IB	- -	运算放大器偏置电流设置端
11	VDD	- -	工作电源正端
12	2OUT	O	第二级运算放大器的输出端
13	2IN-	I	第二级运算放大器的反相输入端
14	1IN+	I	第一级运算放大器的同相输入端
15	1IN-	I	第一级运算放大器的反相输入端
16	1OUT	O	第一级运算放大器的输出端

3.1.3 菲涅尔透镜-球面

菲涅尔球面透镜实物如图 3.2 所示。

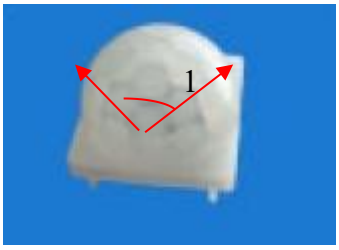


图 3.2　菲涅尔球面透镜实物

菲涅尔球面透镜技术参数如表 3 所示。

表 3　菲涅尔球面透镜技术参数

焦距	10.5
角度	100°
距离	8m

3.2 人体传感器的电路原理图

人体传感器的电路原理图，如图 3.3 所示。

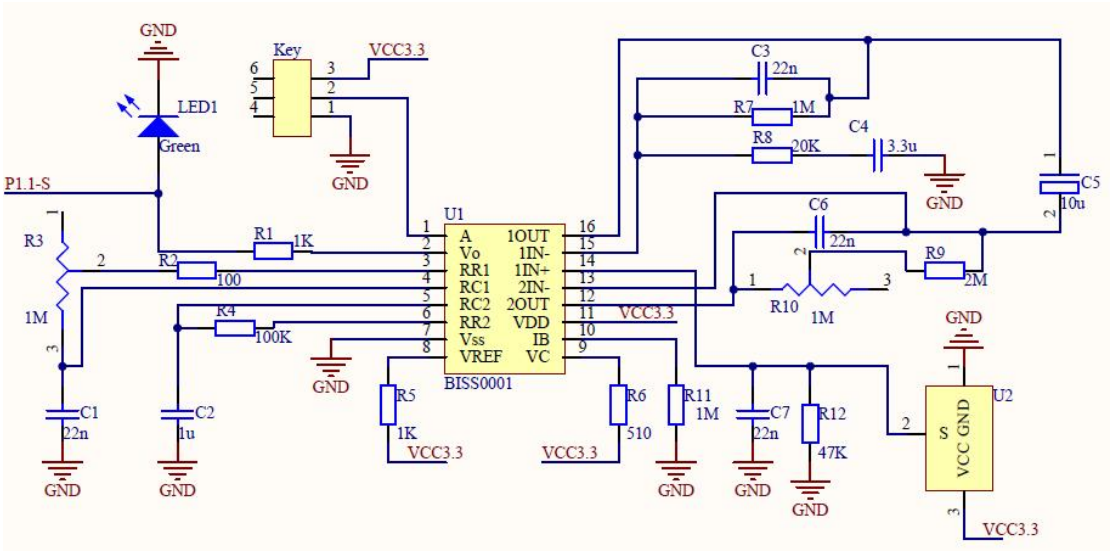


图 3.3　人体传感器电路

图中 U2 是人体感应传感器。

U1 是红外热释电处理芯片 BISS0001

当人不在人体感应传感器感应范围内时，人体感应传感器开路，1IN+没有输入。当人进入人体感应传感器感应范围时，人体感应传感器导通，则 1IN+有输入，对应 Vo 会输出高电平，Vo--R1--LED1--GND 形成电流回路，则 LED1 亮。

3.3 调节人体传感器

人体传感器 B1 板如图 3.4 所示，其中电位器 R3 是用来调节（Time）延时时间大小，顺时针旋转延时时间减小；电位器 R10 用来调节测试距离（Range）大小，顺时针旋转测试距离减小。

Key是控制触发方式的开关。人体感应传感器有下面两种触发模式：

a、不可重复触发方式:即感应输出高电平后，延时时间段一结束，输出将自动从高电

平变成低电平；

b、可重复触发方式：即感应输出高电平后，在延时时间段内，如果有人体在其感应

范围活动，其输出将一直保持高电平，直到人离开后才延时将高电平变为低电平（感应模块检测到人体的每一次活动后会自动顺延一个延时时间段，并且以最后一次活动的间为延时时间的起始点)。

当Key打到High端（电路板上有标记），为可重复触发方式，即当人体在传感器感应范围内时LED1恒亮；当Key打到Low端，为不可重复触发方式，即当人体在传感器感应范围内时，LED1亮一段时间后会自动灭。

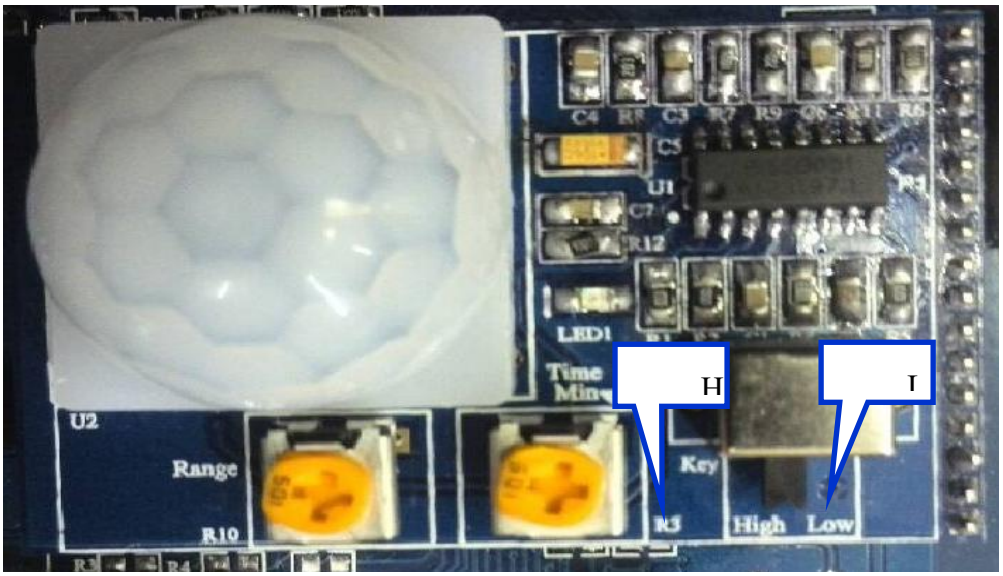


图3.4　人体传感器B1板

注意：人体感应传感器还具有感应封锁时间(默认设置:2.5S 封锁时间)：即传感器在每一次感应输出后（LED1由亮边灭之后），可以紧跟着设置一个封锁时间段，在此时间段内感应器不接受任何感应信号。可有效抑制负载切换过程中产生的各种干扰。(此时间可设置在零点几秒—几十秒钟)。

四、实验过程

4.1　编写实验源代码文件

4.1.1　打开 KeilC51 集成开发环境，创建 RenTi 工程。

4.1.2　配置 RenTi 工程的参数。

4.1.3　编写 RenTi 源代码。

4.1.4　编译工程文件，生成可执行 RenTi.hex 文件。

4.1.5　将可执行 RenTi.hex 文件下载进 STC12C5A16S2 单片机底板。

4.1.6　将人体传感器插入下载完程序的单片机底板。

4.1.7　将插入人体感应传感器的节点重新上电。

4.2　实验源代码解析

Main.c 源代码

```
/*************************************************************/
//晶振频率：11.0592MHz
//文件名  ：Main.c
//功能说明：人体感应传感器读取实验
//制作    ：www.frotech.com
//技术支持：020-22883196 QQ:
//变更记录：2013.05.02
//变更内容：新建造
/*************************************************************/

#include <STC12C5A60S2.h>


#define      BUF_LENTH  128                    //定义串口接收缓
冲长度
#define     uint unsigned int
#define     uchar unsigned char
unsigned char    uart1_wr;                      //写指针
unsigned char    uart1_rd;                      //读指针
unsigned char    xdata RX0_Buffer[BUF_LENTH];   //接收缓冲
unsigned char flag;
unsigned char i;
bit       B_TI;                                 //发送完成标志
sbit  P1_1 = P1^1;                              //定义 P1.1 端口
//                          7       6       5       4         3
2    1    0   Reset Value
//sfr ADC_CONTR = 0xBC;     ADC_POWER   SPEED1   SPEED0   ADC_FLAG
ADC_START CHS2 CHS1 CHS0 0000,0000 //AD 转换控制寄存器
#define ADC_OFF()    ADC_CONTR = 0
```

```
#define ADC_ON       (1 << 7)
#define ADC_90T      (3 << 5)
#define ADC_180T (2 << 5)
#define ADC_360T (1 << 5)
#define ADC_540T 0
#define ADC_FLAG (1 << 4)   //软件清 0
#define ADC_START    (1 << 3)   //自动清 0
#define ADC_CH0      0
#define ADC_CH1      1
#define ADC_CH2      2
#define ADC_CH3      3
#define ADC_CH4      4
#define ADC_CH5      5
#define ADC_CH6      6
#define ADC_CH7      7

uint adc10_start(uchar channel);
void  uart1_init(void);
void Uart1_TxByte(unsigned char dat);
void Uart1_String(unsigned char code *puts);
void delay_ms(unsigned char ms);

/*************** 用户定义参数 ******************************/

#define MAIN_Fosc       11059200UL
#define Baudrate0       9600UL

/*****************************************************************/
```

```
    /****************** 编 译 器 自 动 生 成 ， 用 户 请 勿 修 改
************************************/

    #define BRT_Reload          (256 - MAIN_Fosc / 16 / Baudrate0)
   //Calculate the timer1 reload value ar 1T mode


    /*************************************************************/

    //*************************************************************
*******
    //函数名：main(void)
    //输入  ：无
    //输出  ：无
    //功能描述：当人不靠近人体感应传感器时，ADC1(P1.1)的电压为零，LED1
灭，输出“No one”
    //          当人体靠近人体感应传感器时，ADC1(P1.1)的电压不为零，
LED1 亮,输出“Someone”

    //*************************************************************
*******
    void   main(void)
    {
       uint   j;
       uart1_init();               //初始化串口
       P1ASF = (1 << ADC_CH1); //STC12C5A16S2 系列模拟输入(AD)选择
ADC1(P1.1)
       ADC_CONTR = ADC_360T | ADC_ON;
```

```
    while(1)
    {
        delay_ms(500);
        j = adc10_start(1);      //(P1.1)ADC1 转换
        if(j > 0)//     当 ADC1 的值大于 0 时我们判定有人靠近，且输出
字符串“Someone”
        {
           Uart1_String("Someone");
           Uart1_TxByte(0x0d);
           Uart1_TxByte(0x0a);
        }
        else                     //没有人靠近，输出字符串“No one”
        {
           Uart1_String("No one");
           Uart1_TxByte(0x0d);
           Uart1_TxByte(0x0a);
        }
/*      Uart1_TxByte('A');        //以下为按照十进制输出 ADC 值
        Uart1_TxByte('D');
        Uart1_TxByte('1');
        Uart1_TxByte('=');
        Uart1_TxByte(j/1000 + '0');
        Uart1_TxByte(j%1000/100 + '0');
        Uart1_TxByte(j%100/10 + '0');
        Uart1_TxByte(j%10 + '0');
        Uart1_TxByte(0x0d);
        Uart1_TxByte(0x0a);
*/
    }
```

```
}
//*****************************************************************
*******
//函数名：adc10_start(uchar channel)
//输入  ：ADC 转换的通道
//输出  ：ADC 值
//功能描述：ADC 转换
//*****************************************************************
*******
uint   adc10_start(uchar channel)  //channel = 0~7
{
   uint   adc;
   uchar  i;

   ADC_RES = 0;
   ADC_RESL = 0;

   ADC_CONTR = (ADC_CONTR & 0xe0) | ADC_START | channel;
   i = 250;
   do{
       if(ADC_CONTR & ADC_FLAG)
       {
           ADC_CONTR &= ~ADC_FLAG;
           adc = (uint)ADC_RES;
           adc = (adc << 2) | (ADC_RESL & 3);
           return adc;
       }
   }while(--i);
   return 1024;
```

```
}

//*****************************************************************
*******
//函数名：uart1_init(void)
//输入  ：无
//输出  ：无
//功能描述：串口初始化函数，通信参数为 9600 8 N 1
//*****************************************************************
*******
void  uart1_init(void)
{
   PCON |= 0x80;     //UART0 Double Rate Enable
   SCON = 0x50;      //UART0 set as 10bit , UART0 RX enable
   AUXR |=  0x01;        //UART0 使用 BRT
   AUXR |=  0x04;        //BRT set as 1T mode
   BRT = BRT_Reload;
   AUXR |=  0x10;        //start BRT

   ES  = 1;
   EA = 1;
}
//*****************************************************************
*******
//函数名：Uart1_TxByte(unsigned char dat)
//输入  ：需要发送的字节数据
//输出  ：无
//功能描述：从串口发送单字节数据
//*****************************************************************
```

```
*******
    void Uart1_TxByte(unsigned char dat)
    {
        B_TI = 0;
       SBUF = dat;
       while(!B_TI);
       B_TI = 0;
    }
    //*****************************************************************
*******
    //函数名：Uart1_String(unsigned char code *puts)
    //输入  ：字符串首地址
    //输出  ：无
    //功能描述：从串口发送字符串
    //*****************************************************************
*******
    void Uart1_String(unsigned char code *puts)
    {
        for(; *puts != 0; puts++)
       {
            Uart1_TxByte(*puts);

       }
    }

    //*****************************************************************
*******
    //函数名：UART1_RCV (void)
    //输入  ：无
```

```
    //输出  ：无
    //功能描述：串口中断接收函数
    //*****************************************************************
*******
    void UART1_RCV (void) interrupt 4
    {
       if(RI)
       {
          RI = 0;
          RX0_Buffer[uart1_wr++] = SBUF;
          //if(++uart0_wr >= BUF_LENTH)   uart0_wr = 0;
          flag = 1;
       }

       if(TI)
       {
          TI = 0;
          B_TI = 1;
       }
    }

    void delay_ms(unsigned char ms)
    {
        unsigned int i;
       do{
           i = MAIN_Fosc /1400;
          while(--i);
       }while(--ms);
    }
```

4.3　实验运行效果

当人不在人体感应传感器感应范围内时，ADC1(P1.1)的电压为零，LED1 灭，输出字符串“no one”；当人进入人体感应传感器感应范围时，ADC1(P1.1)的电压不为零，LED1 亮，输出字符串“someone”，其串口信息如图 4.1。一般测量范围在 3—4m 时较灵敏，菲涅尔透镜中心处可探测距离最远，可达 8m。

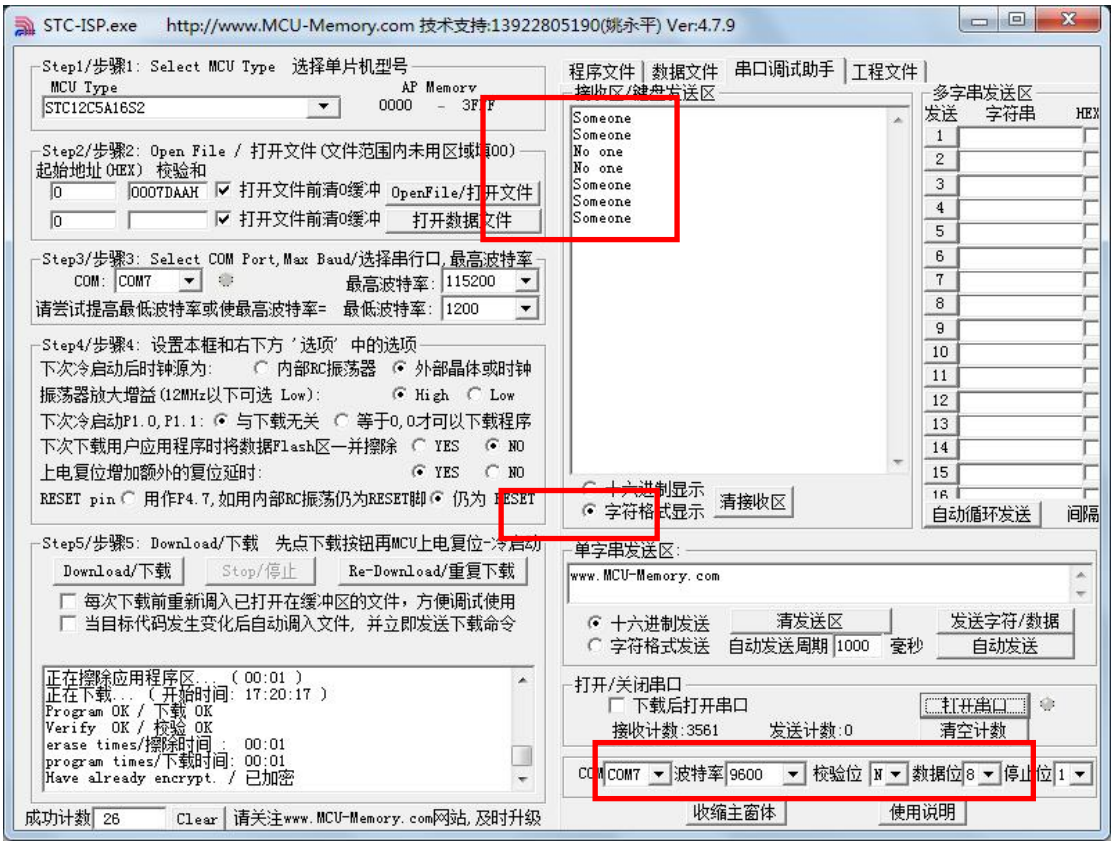


图 4.1　人体传感器运行时串口信息

注意：串口参数为 9600　8　N　1

实训项目十 声音传感器应用

一、实验目的

学习声音传感器的使用方式

二、实验设备

硬件：IOT-L01-05 型物联网综合实验箱 1 台，带声音传感器，串口线。

软件：Keil，STC_ISP_V479，串口调试工具等。

三、实验原理

源码路径：配套光盘\源代码\传感器原理及应用\实验十 声音传感器

hex 文件路径：配套光盘\源代码\传感器原理及应用\可执行程序\Sound.hex

3.1 声音传感器介绍

本声音传感器是使用驻极体电容式咪头来采集声音信号的变化量，如图 3.1 是驻极体电容式咪头的电路原理图。

FET(场效应管)咪头的主要器件，起到阻抗变换或放大的作用，

C 是一个可以通过膜片震动而改变电容量的电容，声电转换的主要部件。C1,C2 是为了防止射频干扰而设置的，可以分别对两个射频频段的干扰起到抑制作用。C1 一般是 10PF,C2 一般是 33PF,10PF 滤波 1800Mhz,33PF 滤波 900Mhz。

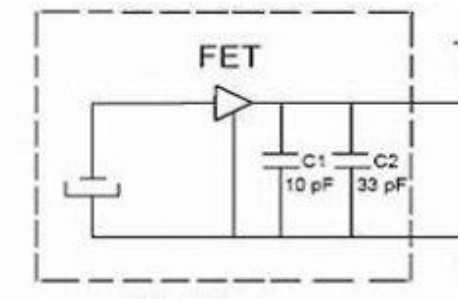


图 3.1　驻极体电容式咪头的电路原理图

由声信号到电信号的转换：

由静电学可知，对于平行板电容器，有如下的关系式：

$$C=\varepsilon \cdot S/L \qquad ①$$

即电容的容量与介质的介电常数成正比，与两个极板的面积成正比，与两个极板之间的距离成反比。另外，当一个电容器充有 Q 量的电荷，那麽电容器两个极板要形成一定的电压，有如下关系式；

$$C=Q/V \qquad ②$$

对于一个驻极体传声器，内部存在一个由振膜，垫片和极板组成的电容器，因为膜片上充有电荷，并且是一个塑料膜，因此当膜片受到声压强的作用，膜片要产生振动，从而改变了膜片与极板之间的距离，从而改变了电容器两个极板之间的距离，产生了一个Δd 的变化，因此由公式①可知，必然要产生一个ΔC 的变化，由公式②又知，由于ΔC 的变化，充电电荷又是固定不变的，因此必然产生一个ΔV 的变化。

由于这个信号非常微弱，内阻非常高，不能直接使用，在在里我们采用 LM393 来对咪头的产生的电压变化进行电压比较输出，LM393 最低输入偏差电压为 2mV，所以咪头只要有一定的变化电压输出，那么 LM393 的比较输出就能够体现出来。

3.2 声音传感器的电路原理图

人体传感器到的实际电路如图 3.2.

U8 就是咪头。

R14 是电位器，用来调节 LM393 的比较输出灵敏度。

LM393 是比较器

当有声音使得咪头产生电压输出变化时，LM393 的输出端连接 51 单片机的 P1.0 口便电平变低，实际的测试图如图 3.3 所示，一般来说如果声音越到该电平越低，那么 B1 板上的 D1 会更亮。

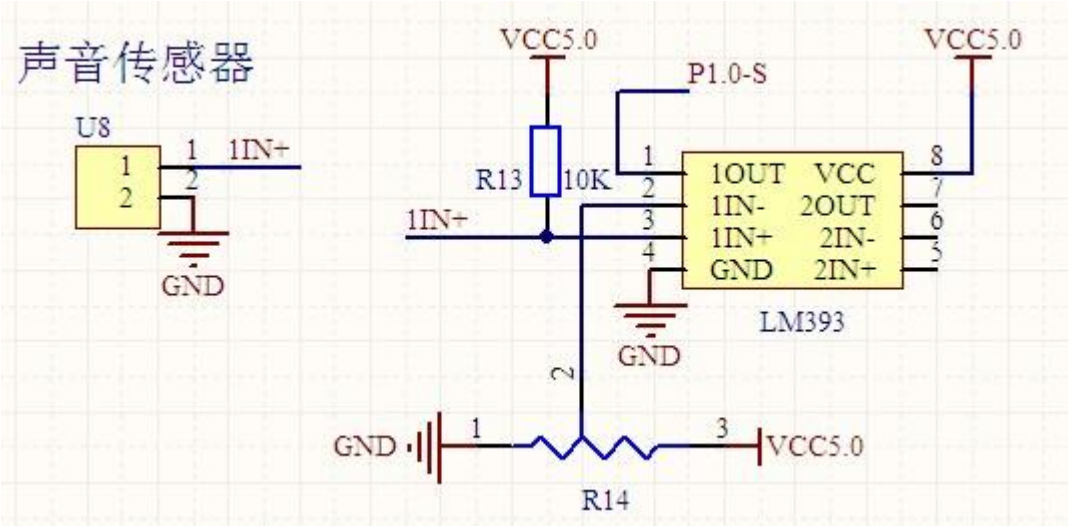


图 3.2　声音传感器电路

3.3 调节声音传感器

给 B1 板上电，如果没有比较大的声音时，B1 板上的 D1 一直亮，那么请按照图 3 所示逆时针调节 R14 阻值，使得 D1 灭；然后敲击物体产生声音（声音约为人轻轻鼓掌时声音的大小）使得 D1 灯亮（如果不亮，请按照图 3 顺时针调节 R14 阻值直至 D1 亮），如图 3 所示，声音停止后 D1 又不亮。描述得简单点就是顺时针调节更灵敏，反之则更迟钝。

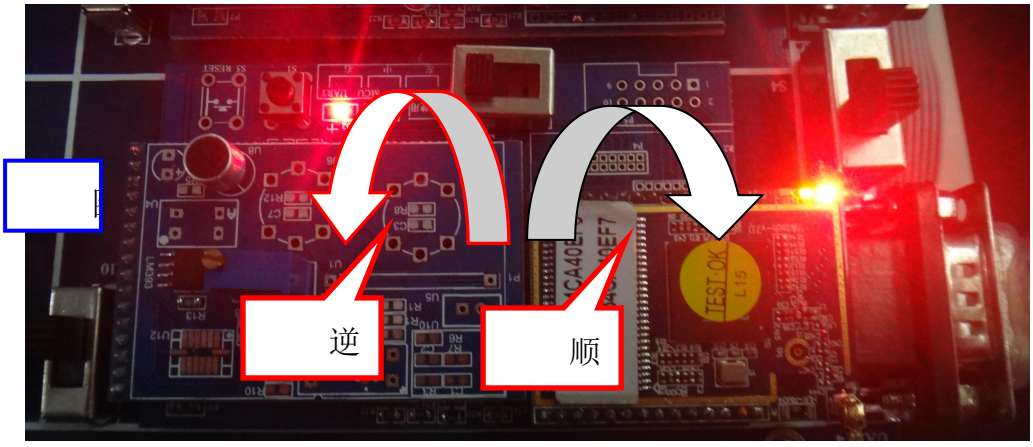


图 3.3　声音传感器的灵敏调节

四、实验过程

4.1　编写实验源代码文件

4.1.1　打开 Keil C51 集成开发环境，创建 Sound 工程。

4.1.2　配置 Sound 工程的参数。

4.1.3　编写 Sound 源代码。

4.1.4　编译工程文件，生成可执行 Sound.hex 文件。

4.1.5　将可执行 Sound.hex 文件下载进 STC12C5A16S2 单片机底板。

4.1.6　将声音传感器插入下载完程序的单片机底板。

4.1.7　将插入声音传感器的节点重新上电。

4.2　实验源代码解析

Main.c **源代码**

```
/*************************************************************/
//晶振频率：11.0592MHz
//文件名　：Main.c
//功能说明：声音传感器读取实验
//制作　　：www.frotech.com
//技术支持：020-22883196 QQ:
//变更记录: 2013.05.02
//变更内容: 新建造
/*************************************************************/

#include <STC12C5A60S2.h>
#define       BUF_LENTH  128     //定义串口接收缓冲长度
unsigned char     uart1_wr;          //写指针
unsigned char     uart1_rd;          //读指针
unsigned char     xdata RX0_Buffer[BUF_LENTH];    //接收缓冲
unsigned char flag;
unsigned char i;
```

```
bit       B_TI;                    //发送完成标志
sbit  P1_0 = P1^0;                 //定义 P1.0 端口

void   uart1_init(void);
void Uart1_TxByte(unsigned char dat);
void Uart1_String(unsigned char code *puts);
void delay_ms(unsigned char ms);

/*************** 用户定义参数 ****************************/

#define MAIN_Fosc        11059200UL
#define Baudrate0        9600UL

/*************************************************************/


/****************** 编 译 器 自 动 生 成 ， 用 户 请 勿 修 改
************************************/

#define BRT_Reload           (256 - MAIN_Fosc / 16 / Baudrate0)
//Calculate the timer1 reload value ar 1T mode


/*************************************************************/

//*************************************************************************
*******
//函数名：main(void)
//输入  ：无
```

```
    //输出   ：无
    //功能描述：当有声音在声音传感器的附近传播时候，D1 亮，同时输出字符串“Sound”
    //           P1.0 采用准双向口工作模式
    //*****************************************************************************
    void   main(void)
    {
       uart1_init();                    //初始化串口
       while(1)
       {
          if(P1_0 == 0)                 //P1.0 为低电平的时候输出“Sound”
          {
                Uart1_String("Sound");
                Uart1_TxByte('\r');       //回车换行
                Uart1_TxByte('\n');
                delay_ms(50);           //延时下，调试用
          }
       }
    }


    //*****************************************************************************
    //函数名：uart1_init(void)
    //输入   ：无
    //输出   ：无
    //功能描述：串口初始化函数，通信参数为 9600 8 N 1
```

```
    //*****************************************************************
*******
    void  uart1_init(void)
    {
       PCON |= 0x80;     //UART0 Double Rate Enable
       SCON = 0x50;      //UART0 set as 10bit , UART0 RX enable
       AUXR |=  0x01;        //UART0 使用 BRT
       AUXR |=  0x04;        //BRT set as 1T mode
       BRT = BRT_Reload;
       AUXR |=  0x10;        //start BRT

       ES  = 1;
       EA = 1;
    }
    //*****************************************************************
*******
    //函数名：Uart1_TxByte(unsigned char dat)
    //输入  ：需要发送的字节数据
    //输出  ：无
    //功能描述：从串口发送单字节数据
    //*****************************************************************
*******
    void Uart1_TxByte(unsigned char dat)
    {
        B_TI = 0;
       SBUF = dat;
       while(!B_TI);
       B_TI = 0;
    }
```

```
    //*************************************************************
*******
    //函数名：Uart1_String(unsigned char code *puts)
    //输入  ：字符串首地址
    //输出  ：无
    //功能描述：从串口发送字符串
    //*************************************************************
*******
    void Uart1_String(unsigned char code *puts)
    {
        for(; *puts != 0; puts++)
       {
            Uart1_TxByte(*puts);

       }
    }

    //*************************************************************
*******
    //函数名：UART1_RCV (void)
    //输入  ：无
    //输出  ：无
    //功能描述：串口中断接收函数
    //*************************************************************
*******
    void UART1_RCV (void) interrupt 4
    {
       if(RI)
       {
```

```
        RI = 0;
        RX0_Buffer[uart1_wr++] = SBUF;
        //if(++uart0_wr >= BUF_LENTH)   uart0_wr = 0;
        flag = 1;
    }

    if(TI)
    {
        TI = 0;
        B_TI = 1;
    }
}

void delay_ms(unsigned char ms)
{
     unsigned int i;
    do{
         i = MAIN_Fosc /1400;
        while(--i);
    }while(--ms);
}
```

4.3 实验运行效果

当我们在声音传感器附近产生声音时，就能够在串口调试输出“Sound”如图 4.1 所示，如果不能达到预期效果，请调节电位器。

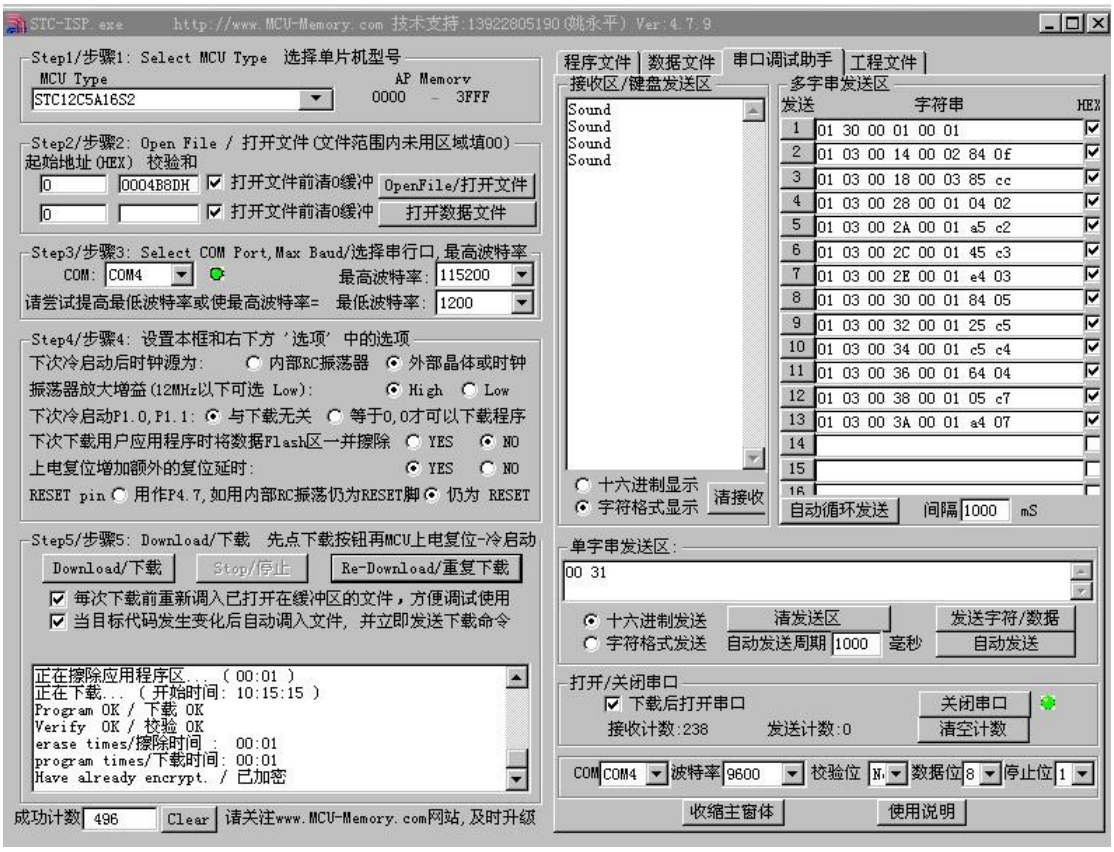


图 4.1　声音传感器运行时串口输出信息

注意：串口的参数为 9600　8　N　1

实训项目十一 温湿度传感器应用

一、实验目的

学习温湿度传感器的使用方法

二、实验设备

硬件：IOT-L01-05 型物联网综合实验箱 1 台，带温湿度传感器，串口线。

软件：Keil，STC_ISP_V479，串口调试工具。

三、实验原理

芯片手册：配套光盘\附件\芯片手册\温湿度传感器

源码路径：配套光盘\源代码\传感器原理及应用\实验十一 温湿度传感器

hex 文件路径：配套光盘\源代码\传感器原理及应用\可执行程序\AM2321.hex

3.1 AM2321 介绍

AM2321 湿敏电容数字温湿度模块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。传感器包括一个电容式感湿元件和一个高精度测温元件，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、性价比极高等优点。每个传感器都在极为精确的湿度校验室中进行校准。校准系数以程序的形式储存在单片机中，传感器内部在检测信号的处理过程中要调用这些校准系数。标准单总线接口，使系统集成变得简易快捷。超小的体积、极低的功耗，信号传输距离可达 20 米以上。产品为 3 引线（单总线接口）连接方便。AM2321 的响应时间约 2S，这比一般的温湿度传感器的响应时间要快。

3.2 温湿度传感器 AM2321 的电路原理图

AM2321 温湿度传感器的电路原理如图 3.1 所示。

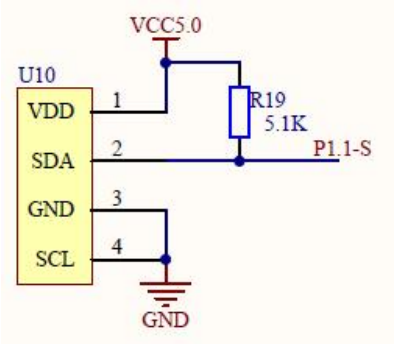


图 3.1　AM2321 温湿度传感器电路

其 SDA 口连接到 STC12C5A16S2 的 P1.1 进行单总线通信，R19 为上拉电阻，详情见其技术手册的 P4。各 PIN 的分配如表 1。

表 1　AM2321 引脚说明

PIN 号	名称	描述
PIN1	VDD	电源（3.5--5.5V）
PIN2	SDA	串行数据，双向口
PIN3	GND	地
PIN4	SCL	单总线通信模式接地

3.3　AM2321 的传感器性能

AM2321 的传感器参数如图 3.2 所示。

参数	条件	min	typ	max	单位
分辨率			0.1		%RH
			16		bit
精度[1]	25℃		±2		%RH
重复性			±0.3		%RH
互换性		完全互换			
响应时间[2]	1/e(63%)		<5		S
迟滞			<0.3		%RH
漂移[3]	典型值		<0.5		%RH/yr

图 3.2　AM2321 传感器参数 1

AM2321 传感器另外一部分参数如图 3.3 所示。

参数	条件	min	typ	max	单位
分辨率			0.1		℃
			16		bit
精度			±0.5	±1	℃
量程范围		-40		80	℃
重复性			±0.2		℃
互换性		完全互换			
响应时间	1/e(63%)		<10		S
漂移			±0.3		℃/yr

图 3.3　AM2321 传感器参数 2

3.4　AM2321 的单总线通信协议

SDA 用于微处理器与 AM2321 之间的通讯和同步如图 3.4 所示，采用单总线数据格式，一次传送 40 位数据，高位先出。具体通信时序如图 3.5 所示。

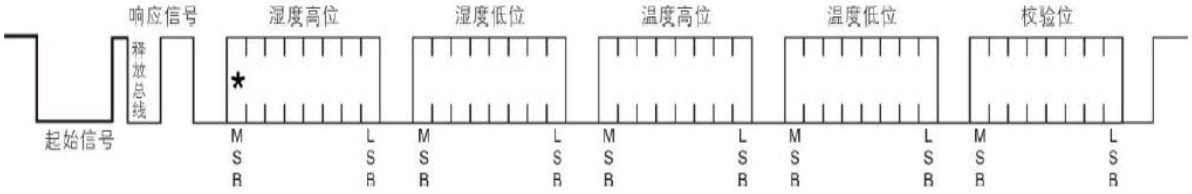


图 3.4　单总线协议时序

单总线格式定义如表 2 所示。

表 2　　单总线格式定义

名称	单总线格式定义
起始信号	微处理器把数据总线（SDA）拉低一段时间（至少 800us），通知传感器准备数据
响应信号	传感器把数据总线（SDA）拉低 80us,再拉高 80us 以响应主机的起始信号。
数据格式	收到主机起始信号后，传感器一次性从数据总线(SDA）串出 40 位数据，高位先出
湿度	湿度分辨率是 16Bit,高位在前；传感器串出的湿度值是实际湿度值

	的 10 倍
温度	温度分辨率是 16Bit,高位在前；传感器串出的温度值是实际温度值的 10 倍； 温度最高位（Bit15)等于 1 表示负温度，温度最高位（Bit15)等于 0 表示正温度； 温度除了最高位（Bit14~~Bib0)表示温度值。
校验位	校验位 = 湿度高位 + 湿度低位 + 温度高位 + 温度低位

总线数据计算举例说明：

如果接收到的 40 位数据如表 3 所示。

0000 0010	1001 0010	0000 0001	0000 1101	1010 0010
湿度高 8 位	湿度低 8 位	温度高 8 位	温度低 8 位	校验位

计算：

0000 0010 + 1001 0010 + 0000 0001 + 0000 1101 + 1010 0010 = 1010 0010（校验位）

接收数据正确，则湿度及温度的值如下：

湿度：0000 0010 1001 0010 = 0x0292H = 658, 即湿度 = 65.8%RH

温度：0000 0001 0000 1101 = 0x10DH = 269，即温度 = 26.9℃

单总线具体通信时序：

用户主机（MCU）发送一次起始信号（把数据总线 SDA 拉低至少 800μs）后，AM2321 从休眠模式转换到高速模式。待主机开始信号结束后，AM2321 发送响应信号，从数据总线 SDA 串行送出 40Bit 的数据，先发送字节的高位；发送的数据依次为湿度高位、湿度低位、温度高位、温度低位、校验位，发送数据结束触发一次信息采集,采集结束传感器自动转入休眠模式,直到下一次通信来临。单总线具体通信时序如图 5 所示。

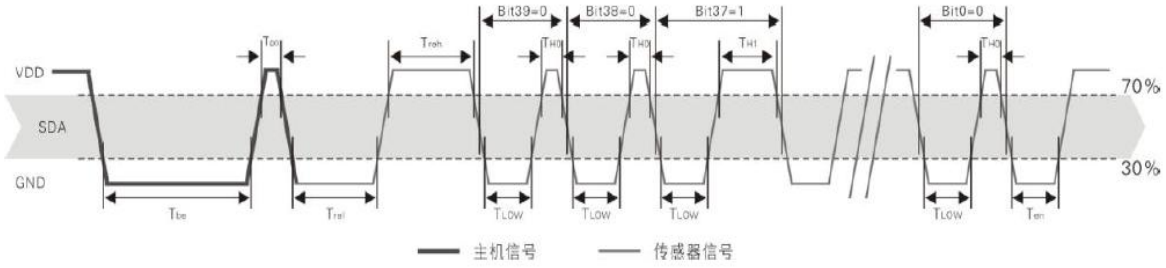


图 3.5　具体单总线通讯时序

注：主机从 AM2321 读取的温湿度数据总是前一次的测量值，如两次测量间隔时间很长，请连续读两次以第二次获得的值为实时温湿度值，同时两次读取间隔时间最小为 2S。

单总线时间表如表 4 所示。

表 4　个时间段信息

符号	参数描述	Min	Typ	Max	单位
Tbe	主机起始信号拉低时间	0.8	1	20	mS
Tgo	主机释放总线时间	20	30	200	uS
Trel	响应低电平时间	75	80	85	uS
Treh	响应高电平时间	75	80	85	uS
Tlow	信号“0”、“1”低电平时间	48	50	55	uS
Tho	信号“0”高电平时间	22	26	30	uS
Th1	信号“1”高电平时间	68	70	75	uS
Ten	传感器释放总线时间	4	50	55	u

		5			S

3.5 外部设备读取温湿度信号的步骤

步骤一：AM2321 上电后（AM2321 上电后要等待 2S 以越过不稳定状态，在此期间读取设备不能发送任何指令），测试环境温湿度数据，并记录数据，此后传感器自动转入休眠状态。AM2321 的 SDA 数据线由上拉电阻拉高一直保持高电平，此时 AM2321 的 SDA 引脚处于输入状态，时刻检测外部信号。

步骤二：微处理器的 I/O 设置为输出，同时输出低电平，且低电平保持时间不能小于 800us，典型值是拉低 1MS，然后微处理器的 I/O 设置为输入状态，释放总线，由于上拉电阻，微处理器的 I/O 即 AM2321 的 SDA 数据线也随之变高，等主机释放总线后，AM2321 发送响应信号，即输出 80 微秒的低电平作为应答信号，紧接着输出 80 微秒的高电平通知外设准备接收数据，信号传输如下图 3.6 所示。

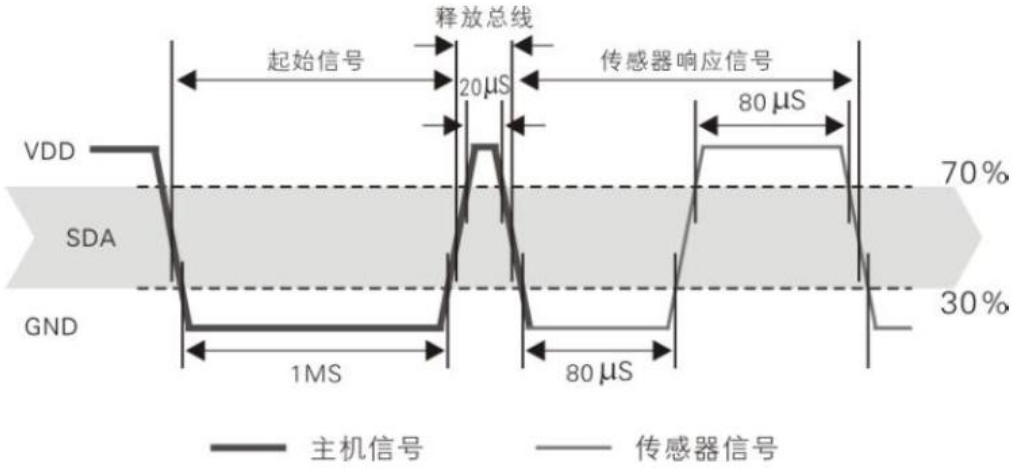


图 3.6　SDA 信号传输时序

步骤三：AM2321 发送完响应后，随后由数据总线 SDA 连续串行输出 40 位数据，微处理器根据 I/O 电平的变化接收 40 位数据。

位数据“0”的格式为： 50 微秒的低电平加 26-28 微秒的高电平；

位数据“1”的格式为： 50 微秒的低电平加 70 微秒的高电平；

位数据“0”、位数据“1”格式信号如图 3.7 所示。

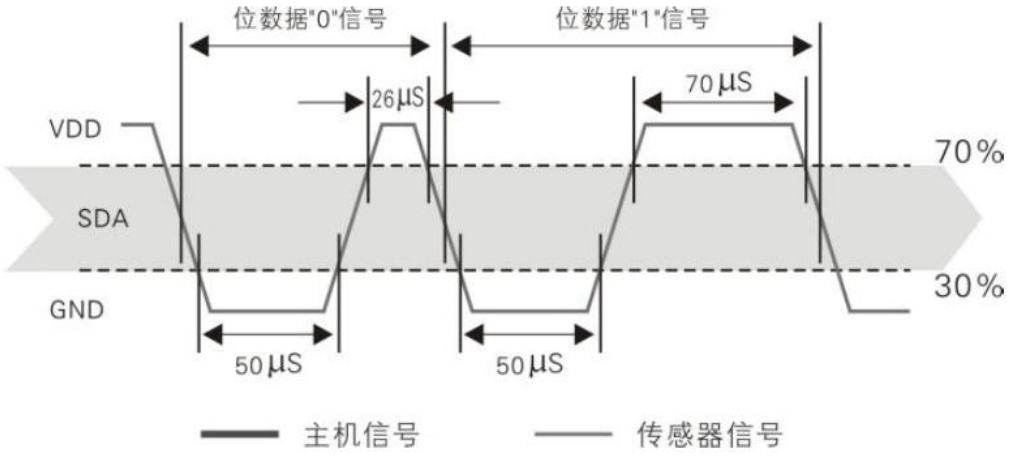


图 3.7　SDA 传输 0 和 1 时序

AM2321 的数据总线 SDA 输出 40 位数据后，继续输出低电平 50 微秒后转为输入状态，由于上拉电阻随之变为高电平。同时 AM2321 内部重测环境温湿度数据，并记录数据，测试记录结束，单片机自动进入休眠状态。单片机只有收到主机的起始信号后，才重新唤醒传感器，进入工作状态。

3.6 外部设备读取温湿度流程

外部设备读取温湿度流程如图 8 所示。

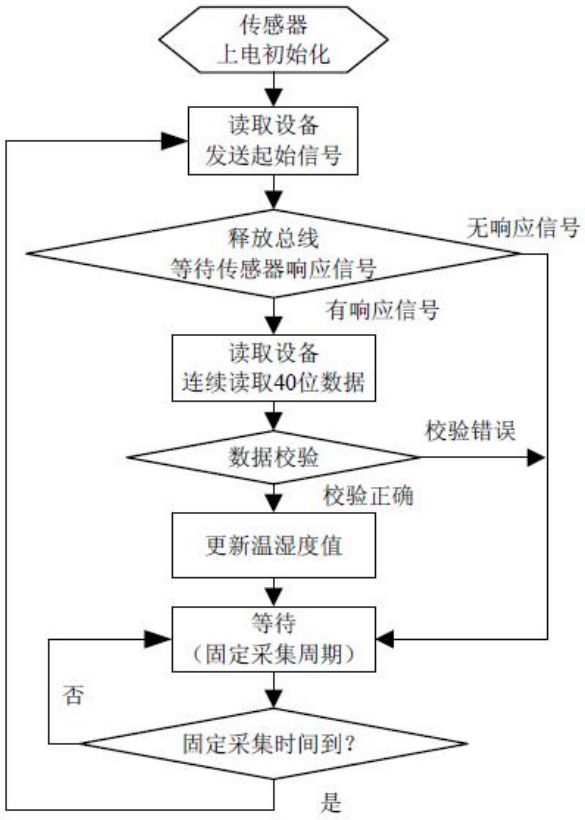


图 8　温湿度读取流程

四、实验步骤

4.1　编写实验源代码文件

4.1.1　打开 KeilC51 集成开发环境，创建 AM2321 工程。

4.1.2　配置 AM2321 工程的参数。

4.1.3　编写 AM2321 源代码。

4.1.4　编译工程文件，生成可执行 AM2321.hex 文件。

4.1.6　将可执行 AM2321.hex 文件下载进 STC12C5A16S2 单片机底板。

4.1.7　将温湿度传感器插入下载完程序的单片机底板。

4.2　实验源代码解析

Main.c **源代码**

```
#include <STC12C5A60S2.h>
```

```
#include <intrins.h>
/*************************************************************/
//晶振频率：11.0592MHz
//文件名  ：Main.c
//功能说明：温湿度传感器 AM2321 读取实验
//制作    ：www.frotech.com
//技术支持：020-22883196 QQ:
//变更记录：2013.05.02
//变更内容：新建造
/*************************************************************/

#define    BUF_LENTH 128    //定义串口接收缓冲长度
unsigned char    uart1_wr;         //写指针
unsigned char    uart1_rd;         //读指针
unsigned char    xdata RX0_Buffer[BUF_LENTH];    //定义串口接收缓冲
unsigned char flag;         //定义串口中断接收标号，即当其为 1 时表示串口介绍到数据。
unsigned char i;        //定义普通变量。

unsigned char Sensor_Data[5]={0x00,0x00,0x00,0x00,0x00};//定义温湿度传感器数据存放区。
unsigned char Sensor_Check;       //温湿度传感器校验和，判断读取的温湿度数据是否正确。
unsigned char Sensor_AnswerFlag;  //温湿度传感器收到起始标志位
unsigned char Sensor_ErrorFlag;   //读取传感器错误标志
unsigned            char            Ascii_buffer[10]            =
{'0','1','2','3','4','5','6','7','8','9'};
unsigned             char             mbus_regi[20]             =
```

```
{'H',':','0','0','.','0','%','R','H',',','T',':','0','0','.','0'};
    unsigned int RH_Data;//定义湿度值，起到中转作用,因为其数值一般大于255，所以声明为 int 类型
    unsigned int T_Data;//定义温度值，起到中转作用 ,因为其数值一般大于255，所以声明为 int 类型
    unsigned int  Sys_CNT;
    unsigned int  Tmp;

    bit       B_TI;
    sbit Sensor_SDA = P1^1;     //定义 P1.1 为 AM2321 的数据口

    /*********************      函      数      声      明      区
**************************************/
    void uart1_init(void);                      //声明串口初始化函数
    void Uart1_TxByte(unsigned char dat);       //声明串口发送单字节函数
    void Uart1_String(unsigned char code *puts); //声明串口发送字符串函数
    void delay_ms(unsigned char ms);            //声明普通延时函数
    void Clear_Data (void);                     //声明 AM2321 的数据清除函数
    void delay_3ms(void);                       //声明延时 3ms 函数
    void delay_30us(void);                      //声明延时 30us 函数
    unsigned char Read_SensorData(void);        //声明读取 AM2321 数据函数
    unsigned char Read_Sensor(void);            //声明读取 AM2321 温湿度数据函数
    /*************************************************************/
```

```
/*****用户定义参数 ，声明串口参数为 9600 8 N 1  **********/
#define MAIN_Fosc        11059200UL
#define Baudrate0        9600UL

/*************************************************************/


/*****************  编 译 器 自 动 生 成 ， 用 户 请 勿 修 改 ***********************/

#define BRT_Reload          (256 - MAIN_Fosc / 16 / Baudrate0)

/*************************************************************/


//*************************************************************************
//函数名：main(void)
//输入  ：无
//输出  ：无
//功能描述：实现 STC12C5A16S2 单片机的 P1.1 口读取 AM2321 的温湿度数据，并按照
//          ASCII 格式输出到串口界面中
//          按照"H:xx.x%RH,T:xx.x”格式每隔约 2 秒显示一次。
//*************************************************************************
void   main(void)
{
   Sensor_SDA = 1;   //SDA 数据线由上拉电阻拉高一直保持高电平，初始
```

化数据总线。

```
uart1_init();          //初始化串口，参数为 9600 8 N 1.
Uart1_String("www.frotech.com"); //串口打印输出测试字符串。
while(1)           //死循环开始
{

    delay_ms(1000);          //延时
    Read_Sensor();           // 读取传感器数据

    if(Sensor_Data[4]          ==          (Sensor_Data[0]+
Sensor_Data[1]+Sensor_Data[2]+Sensor_Data[3]))//判断温湿度校验值是否
相等，见 AM2321 数据手册 P17.
    {
      RH_Data = (Sensor_Data[0] * 256) + Sensor_Data[1];//保存
湿度值，见 AM2321 数据手册 P17
      T_Data  = (Sensor_Data[2] * 256) + Sensor_Data[3];//保存
温度值，见 AM2321 数据手册 P17

      mbus_regi[2]    = Ascii_buffer[RH_Data/100];       //取得
湿度十位值
      mbus_regi[3]    = Ascii_buffer[(RH_Data%100)/10];  //取得
湿度个位值
      mbus_regi[5]    = Ascii_buffer[RH_Data%10];     // 取 得 湿
度小数值

      mbus_regi[12]   = Ascii_buffer[T_Data/100];        //取得
温度十位值
      mbus_regi[13]   = Ascii_buffer[(T_Data%100)/10];   //取得
温度个位值
```

```
            mbus_regi[15]   = Ascii_buffer[T_Data%10];          //取得
温度小数值

            for(i = 0; i < 16; i++)
            {
                Uart1_TxByte(mbus_regi[i]);  //          按          照
"H:xx.x%RH,T:xx.x”格式发送温湿度值
            }
             Uart1_TxByte('\r');              //回车换行
             Uart1_TxByte('\n');
          }
       }
    }



    void   uart1_init(void)
    {
       PCON |= 0x80;      //UART0 Double Rate Enable
       SCON = 0x50;       //UART0 set as 10bit , UART0 RX enable
       AUXR |=  0x01;         //UART0 使用 BRT
       AUXR |=  0x04;         //BRT set as 1T mode
       BRT = BRT_Reload;
       AUXR |=  0x10;         //start BRT

       ES  = 1;
       EA = 1;
    }
```

```
void Uart1_TxByte(unsigned char dat)
{
    B_TI = 0;
   SBUF = dat;
   while(!B_TI);
   B_TI = 0;
}

void Uart1_String(unsigned char code *puts)
{
    for(; *puts != 0; puts++)
   {
        Uart1_TxByte(*puts);

   }
}



/*********************************************/
void UART1_RCV (void) interrupt 4
{
   if(RI)
   {
       RI = 0;
       RX0_Buffer[uart1_wr++] = SBUF;
       //if(++uart0_wr >= BUF_LENTH)   uart0_wr = 0;
       flag = 1;
   }
```

```
        if(TI)
        {
            TI = 0;
            B_TI = 1;
        }
    }


    //*****************************************************************
*******
    //函数名：delay_3ms(void)
    //输入  ：无
    //输出  ：无
    //功能描述：延时 3ms 函数
    //*****************************************************************
**********
    void delay_3ms(void)
    {
        unsigned char a,b;
        for(b=194;b>0;b--)
            for(a=84;a>0;a--);
    }


    //*****************************************************************
*******
    //函数名：delay_30us(void)
    //输入  ：无
    //输出  ：无
    //功能描述：延时 3ms 函数
```

```
    //**************************************************************
**********
    void delay_30us(void)
    {
        unsigned char a;
        for(a=164;a>0;a--);
    }

    void delay_ms(unsigned char ms)
    {
        unsigned int i;
       do{
            i = MAIN_Fosc /1400;
           while(--i);
       }while(--ms);
    }

    //**************************************************************
*******
    //函数名：Read_SensorData(void)
    //输入  ：无
    //输出  ：单字节的温湿度数据或者校验值
    //功能描述：读取单字节的 AM2321 的数据
    //**************************************************************
**********
    unsigned char Read_SensorData(void)
      {
       unsigned char i,cnt;
       unsigned char buffer,tmp;
```

```
        buffer = 0;
        for(i=0;i<8;i++)
        {
            cnt=0;
            while(!Sensor_SDA)           //检测上次低电平是否结束
            {
              if(++cnt >= 290)
               {
                  break;
               }
            }
                                        //延时 Min=26us Max50us 跳过数据"0"
的高电平
            delay_30us();                //延时 30us
            tmp =0;
            if(Sensor_SDA)//延时 30us 后如果数据口还是高，则该位为 1，否
则为 0，见 P19
            {
              tmp = 1;
            }
            cnt =0;
            while(Sensor_SDA)           //等待高电平 结束
            {
                if(++cnt >= 200)
                {
                  break;
                }
            }
            buffer <<=1;                //移位，使得数据的最低位准备接
```

```
收下一位
            buffer |= tmp;                    //把本次接收到的位加入到数
据中
        }
        return buffer;                        //返回单字节数据
       }

    //*****************************************************************
*******
    //函数名：Read_Sensor(void)
    //输入  ：无
    //输出  ：无
    //功能描述：读取 AM2321 的温湿度及校验值放在 Sensor_Data[]中。
    //*****************************************************************
**********
    unsigned char Read_Sensor(void)
      {
       unsigned char i;

        Sensor_SDA = 0;          //起始信号拉低，见 AM2321 数据手册 P18
       delay_3ms();              //延时 3Ms，当然一般 1ms 就可以了。
       Sensor_SDA = 1;           //拉高，释放总线
       delay_30us();               //延时 30us
       Sensor_AnswerFlag = 0;   // 传感器响应标志
       if(Sensor_SDA ==0)        //从高电平到低电平经过 30us(大于 20us)
是否为低
        {
                                 //如果为低，那么传感器发出响应信号
          Sensor_AnswerFlag = 1;    //收到起始信号
```

```
        Sys_CNT = 0;            //判断从机是否发出 80us 的低电平响应信号是否结束
        while((!Sensor_SDA))  //等待传感器响应信号 80us 的低电平结束
        {
          if(++Sys_CNT>300)   //防止进入死循环
         {
           Sensor_ErrorFlag = 1;
           return 0;
          }
        }
        Sys_CNT = 0;
                            //判断从机是否发出 80us 的高电平,如发出则进入数据接收状态
        while((Sensor_SDA))  //等待传感器响应信号 80us 的高电平结束
        {
           if(++Sys_CNT>300) //防止进入死循环
           {
             Sensor_ErrorFlag = 1;
             return 0;
           }
        }
                            // 数据接收    传感器共发送 40 位数据
                            // 即 5 个字节  高位先送   5 个字节分别为湿度高位  湿度低位  温度高位  温度低位  校验和
                            // 校验和为：湿度高位+湿度低位+温度高位+温度低位
        for(i=0;i<5;i++)
        {
          Sensor_Data[i] = Read_SensorData();
```

```
        }

    }
    else
    {
      Sensor_AnswerFlag = 0;     // 未收到传感器响应
    }
    return 1;
}
```

4.3 实验运行效果

程序运行时，串口输出的信息如图 4.1 所示。

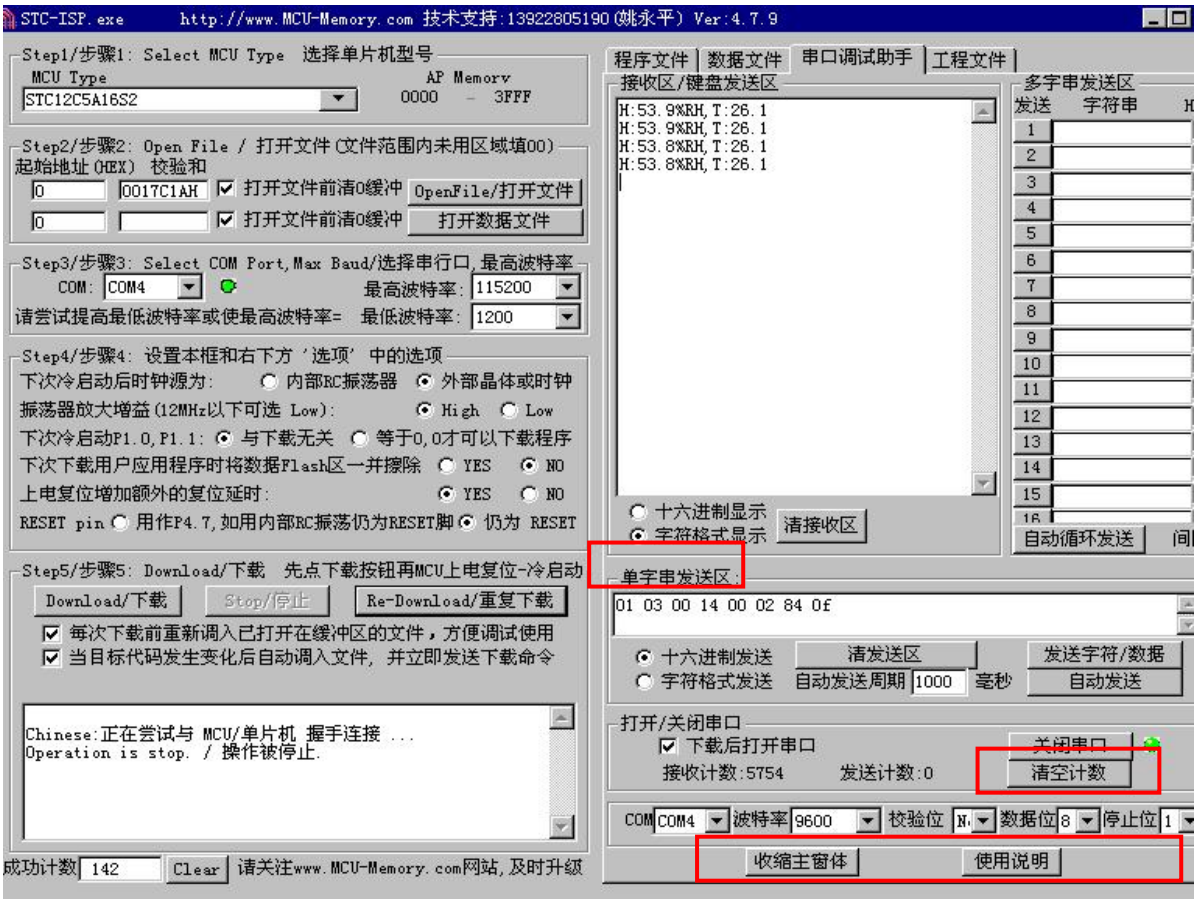


图 4.1　温湿度传感器运行时串口输出信息

注意：1. AM2321 是有保护的塑料外壳包裹的，感应元件基本是通过空气来接触外界，所以你用手直接触摸它的外壳时温度的变化可能比较小，如果你放在温差比较大的两处地方的话会看到温湿度变化明显。

2. 注意串口调试助手的参数设置:9600 8 N 1

实训项目十二 烟雾检测传感器应用

一、实验目的

学习烟雾检测传感器的原理及检测方式

二、实验设备

硬件：IOT-L01-05 型物联网综合实验箱 1 台，带烟雾传感器 MQ-2，串口线。

软件：Keil，STC_ISP_V479，串口调试工具。

三、实验原理

芯片手册：配套光盘\附件\芯片手册\烟雾检测传感器

源码路径:配套光盘\源代码\传感器原理及应用\实验十二 烟雾检测传感器实验

hex 文件路径：配套光盘\源代码\传感器原理及应用\可执行程序\YanWu.hex

3.1 烟雾检测传感器的介绍

烟雾检测传感器采用可燃气体传感器MQ-2，MQ-2气体传感器所使用的气敏材料是在清洁空气中电导率较低的二氧化锡(SnO_2)。当传感器所处环境中存在可燃气体时，传感器的电导率随空气中可燃气体浓度的增加而增大。使用简单的电路即可将电导率的变化转换为与该气体浓度相对应的输出信号。

MQ-2气体传感器对液化气、丙烷、氢气的灵敏度高，对天然气和其它可燃蒸

汽的检测也很理想。这种传感器可检测多种可燃性气体，另外烟雾中含有多种MQ-2可检测的其它，则其可作为烟雾传感器使用，是一款适合多种应用的低成本传感器。

3.1.1 元器件结构图

MQ-2型气体传感器结构如图3.1所示。

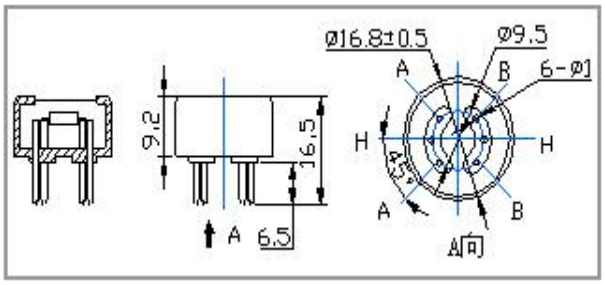


图 3.1 MQ-2 型气体传感器结构

3.1.2 基本测试回路

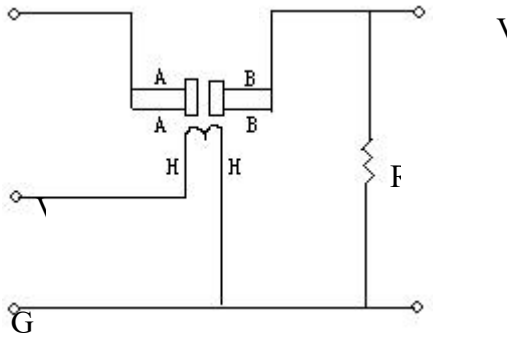


图2 MQ-2型气体传感器测试电路

图3.2是传感器的基本测试电路。该传感器需要施加2个电压：加热器电压（V_H）和测试电压（V_C）。其中 V_H用于为传感器提供特定的工作温度。V_C 则是用于测定与传感器串联的负载电阻（R_L）上的电压（V_{RL}）。这种传感器具有轻微的极

性，V_C需用直流电源。在满足传感器电性能要求的前提下，V_C和V_H可以共用同一个电源电路。为更好利用传感器的 性能，需要

选择恰当的 RL 值。

3.1.3 技术指标

MQ-2 的基本参数如图表 1。

表 1 MQ-2 技术指标

产品型号			MQ-2
产品类型			半导体气敏元件
标准封装			胶木（黑胶木）
检测气体			可燃气体、烟雾
检测浓度			300-10000ppm(可燃气体)
标准电路条件	回路电压	V_C	≤24V DC
	加热电压	V_H	5.0V±0.2V ACorDC
	负载电阻	R_L	可调
标准测试条件下	加热电阻	R_H	31Ω±3Ω（室温）
	加热功耗	P_H	≤900mW
	敏感体表面电阻	R_S	2KΩ-20KΩ(in 2000ppm C_3H_8)

气敏元件特性	灵敏度		Rs(in air)/Rs(1000ppm异丁烷)≥5
	浓度斜率		≤ 0.6($R_{3000ppm}/R_{1000ppm}$ C_3H_8)
标准测试条件	温度、湿度		20℃±2℃；65%±5%RH
	标准测试电路		Vc:5.0V±0.1V；V_H: 5.0V±0.1V
	预热时间		不少于48小时

3.1.4 灵敏度特性

MQ-2 烟雾传感器灵敏度曲线如图 3.3 所示。

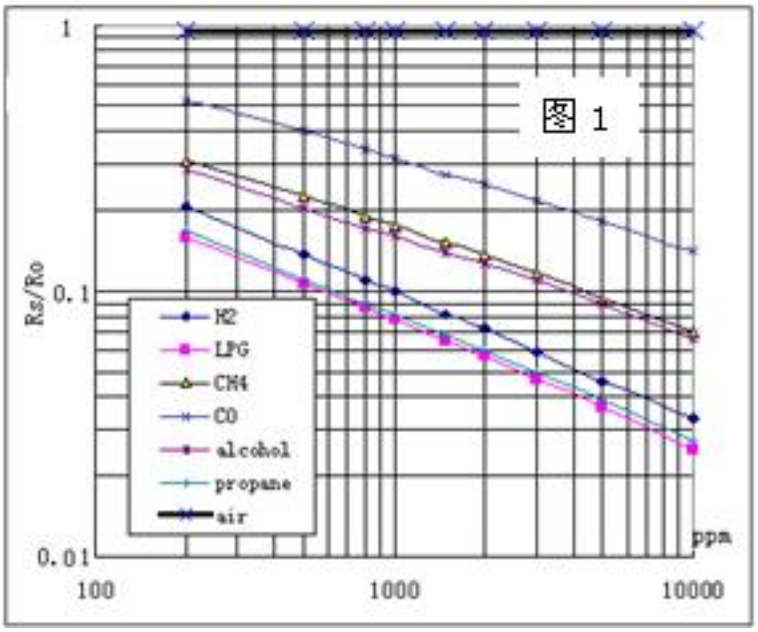


图 3.3　MQ-2 烟雾传感器灵敏度曲线

3.2 烟雾检测传感器电路原理图

MQ-2 烟雾传感器电路原理如图 3.4 所示

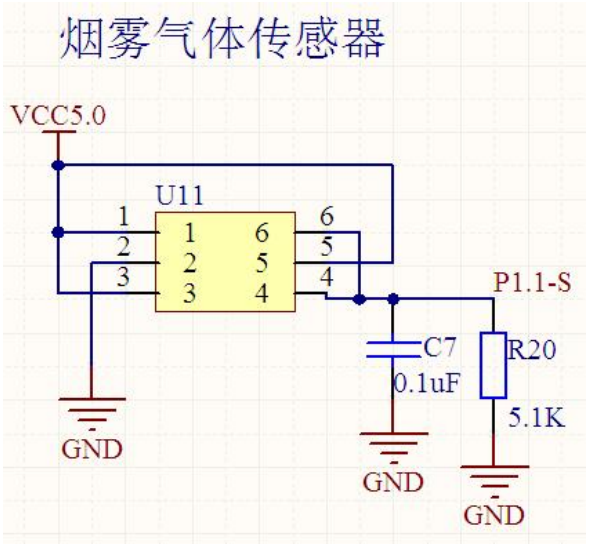


图 3.4　MQ-2 烟雾传感器电路

其中 U11(MQ-2)的 PIN5 与 PIN2 为加热电路，对应结构图中的两个 H 端；PIN1、PIN3、PIN4、PIN6 构成检测电路。

MQ-2 传感器的供电电压 Vc 和加热电压 Vh 都为 5V，负载电阻 R20 为 5.1K 欧姆。ADC1(P1.1)在清洁空气中的值以及检测到烟雾时的值需要根据实际应用情况

进行调整，以下仅为在实验条件下做的不完全的实验结果，仅供参考。在清洁空气中，ADC1 的 AD 采样值为 50 左右；在烟雾中(燃烧纸产生的烟雾或者液化气)，ADC1 的 AD 采样值为大于 85。当 AD 采集的数值大于 85 时表明检测到烟雾。

四、实验过程

4.1 编写实验源代码文件

4.1.1 打开 KeilC51 集成开发环境，创建 YanWu 工程。

4.1.2 配置 YanWu 工程的参数。

4.1.3 编写 YanWu 源代码。

4.1.4 编译从机工程文件，生成可执行 YanWu.hex 文件。

4.1.5 将可执行 YanWu.hex 文件下载进 STC12C5A16S2 单片机底板。

4.1.6 将烟雾传感器插入下载完程序的单片机底板。

4.1.7 将烟雾传感器插入下载完程序的单片机底板。

4.2 实验源代码解析

Main.c **源代码**

```
/*************************************************************/
//晶振频率：11.0592MHz
//文件名  ：Main.c
//功能说明：烟雾检测感器读取实验
//制作    ：www.frotech.com
//技术支持：020-22883196 QQ:
//变更记录：2013.05.02
//变更内容：新建造
/*************************************************************/

#include <STC12C5A60S2.h>


#define       BUF_LENTH  128     //定义串口接收缓冲长度
```

```
#define     uint unsigned int
#define     uchar unsigned char
unsigned char    uart1_wr;          //写指针
unsigned char    uart1_rd;          //读指针
unsigned char    xdata RX0_Buffer[BUF_LENTH];    //接收缓冲
unsigned char flag;
unsigned char i;
bit       B_TI;                  //发送完成标志
sbit  P1_0 = P1^0;                //定义 P1.0 端口
//                          7       6       5       4         3
2    1    0   Reset Value
//sfr ADC_CONTR = 0xBC;     ADC_POWER SPEED1 SPEED0 ADC_FLAG
ADC_START CHS2 CHS1 CHS0 0000,0000 //AD 转换控制寄存器
#define ADC_OFF()    ADC_CONTR = 0
#define ADC_ON       (1 << 7)
#define ADC_90T      (3 << 5)
#define ADC_180T (2 << 5)
#define ADC_360T (1 << 5)
#define ADC_540T 0
#define ADC_FLAG (1 << 4)          //软件清 0
#define ADC_START    (1 << 3)          //自动清 0
#define ADC_CH0      0
#define ADC_CH1      1
#define ADC_CH2      2
#define ADC_CH3      3
#define ADC_CH4      4
#define ADC_CH5      5
#define ADC_CH6      6
#define ADC_CH7      7
```

```
uint adc10_start(uchar channel);
void  uart1_init(void);
void Uart1_TxByte(unsigned char dat);
void Uart1_String(unsigned char code *puts);
void delay_ms(unsigned char ms);

/*************** 用户定义参数 ****************************/

#define MAIN_Fosc       11059200UL
#define Baudrate0       9600UL

/*****************************************************************/


/****************** 编译器自动生成，用户请勿修改
************************************/

#define BRT_Reload          (256 - MAIN_Fosc / 16 / Baudrate0)
//Calculate the timer1 reload value ar 1T mode


/*****************************************************************/

//*************************************************************************
*******
//函数名：main(void)
//输入  ：无
//输出  ：无
```

```
    //功能描述：烟雾检测传感器采用 MQ-2，当有可燃气体或者烟雾的时候，ADC1 的采样值大于
    //          85，即判定有烟雾，输出字符串“YanWu”.
    //*****************************************************************
*******
    void  main(void)
    {
       uint   j;
       uart1_init();                //初始化串口
       P1ASF = (1 << ADC_CH1);      //STC12C5A16S2 系列模拟输入(AD)选择 ADC1(P1.1)
       ADC_CONTR = ADC_360T | ADC_ON;
       while(1)
       {
           delay_ms(500);
           j = adc10_start(1);          //(P1.1)ADC1 转换
          if(j > 0x55)  //当 ADC1 的值大于 85 时我们判定有烟雾，且输出字符串“YanWu”
          {
             Uart1_String("YanWu");
          }
          Uart1_TxByte('A');            //以下为按照十进制输出 ADC 值
          Uart1_TxByte('D');
          Uart1_TxByte('1');
          Uart1_TxByte('=');
          Uart1_TxByte(j/1000 + '0');
          Uart1_TxByte(j%1000/100 + '0');
          Uart1_TxByte(j%100/10 + '0');
          Uart1_TxByte(j%10 + '0');
```

```
            Uart1_TxByte(0x0d);
            Uart1_TxByte(0x0a);
        }
    }
    //*****************************************************************
*******
    //函数名：adc10_start(uchar channel)
    //输入  ：ADC 转换的通道
    //输出  ：ADC 值
    //功能描述：ADC 转换
    //*****************************************************************
*******
    uint   adc10_start(uchar channel)              //channel = 0~7
    {
        uint   adc;
        uchar  i;

        ADC_RES = 0;
        ADC_RESL = 0;

        ADC_CONTR = (ADC_CONTR & 0xe0) | ADC_START | channel;
        i = 250;
        do{
            if(ADC_CONTR & ADC_FLAG)
            {
                ADC_CONTR &= ~ADC_FLAG;
                adc = (uint)ADC_RES;
                adc = (adc << 2) | (ADC_RESL & 3);
                return adc;
```

```
        }
    }while(--i);
    return 1024;
}

//*****************************************************************
*******
//函数名：uart1_init(void)
//输入  ：无
//输出  ：无
//功能描述：串口初始化函数，通信参数为 9600 8 N 1
//*****************************************************************
*******
void  uart1_init(void)
{
    PCON |= 0x80;     //UART0 Double Rate Enable
    SCON = 0x50;      //UART0 set as 10bit , UART0 RX enable
    AUXR |=  0x01;        //UART0 使用 BRT
    AUXR |=  0x04;        //BRT set as 1T mode
    BRT = BRT_Reload;
    AUXR |=  0x10;        //start BRT

    ES  = 1;
    EA = 1;
}
//*****************************************************************
*******
//函数名：Uart1_TxByte(unsigned char dat)
//输入  ：需要发送的字节数据
```

```
    //输出  ：无
    //功能描述：从串口发送单字节数据
    //*****************************************************************
*******
    void Uart1_TxByte(unsigned char dat)
    {
        B_TI = 0;
       SBUF = dat;
       while(!B_TI);
       B_TI = 0;
    }
    //*****************************************************************
*******
    //函数名：Uart1_String(unsigned char code *puts)
    //输入  ：字符串首地址
    //输出  ：无
    //功能描述：从串口发送字符串
    //*****************************************************************
*******
    void Uart1_String(unsigned char code *puts)
    {
        for(; *puts != 0; puts++)
       {
            Uart1_TxByte(*puts);

       }
    }
```

```
//*************************************************************************
*******
//函数名：UART1_RCV (void)
//输入  ：无
//输出  ：无
//功能描述：串口中断接收函数
//*************************************************************************
*******
void UART1_RCV (void) interrupt 4
{
   if(RI)
   {
      RI = 0;
      RX0_Buffer[uart1_wr++] = SBUF;
      //if(++uart0_wr >= BUF_LENTH)   uart0_wr = 0;
      flag = 1;
   }

   if(TI)
   {
      TI = 0;
      B_TI = 1;
   }
}

void delay_ms(unsigned char ms)
{
    unsigned int i;
   do{
```

```
        i = MAIN_Fosc /1400;
        while(--i);
    }while(--ms);
}
```

4.3 实验运行效果

当ADC1(P1.1)采集的值大于85时,表示有烟雾,输出字符串“YanWu”,如图4.1所示。

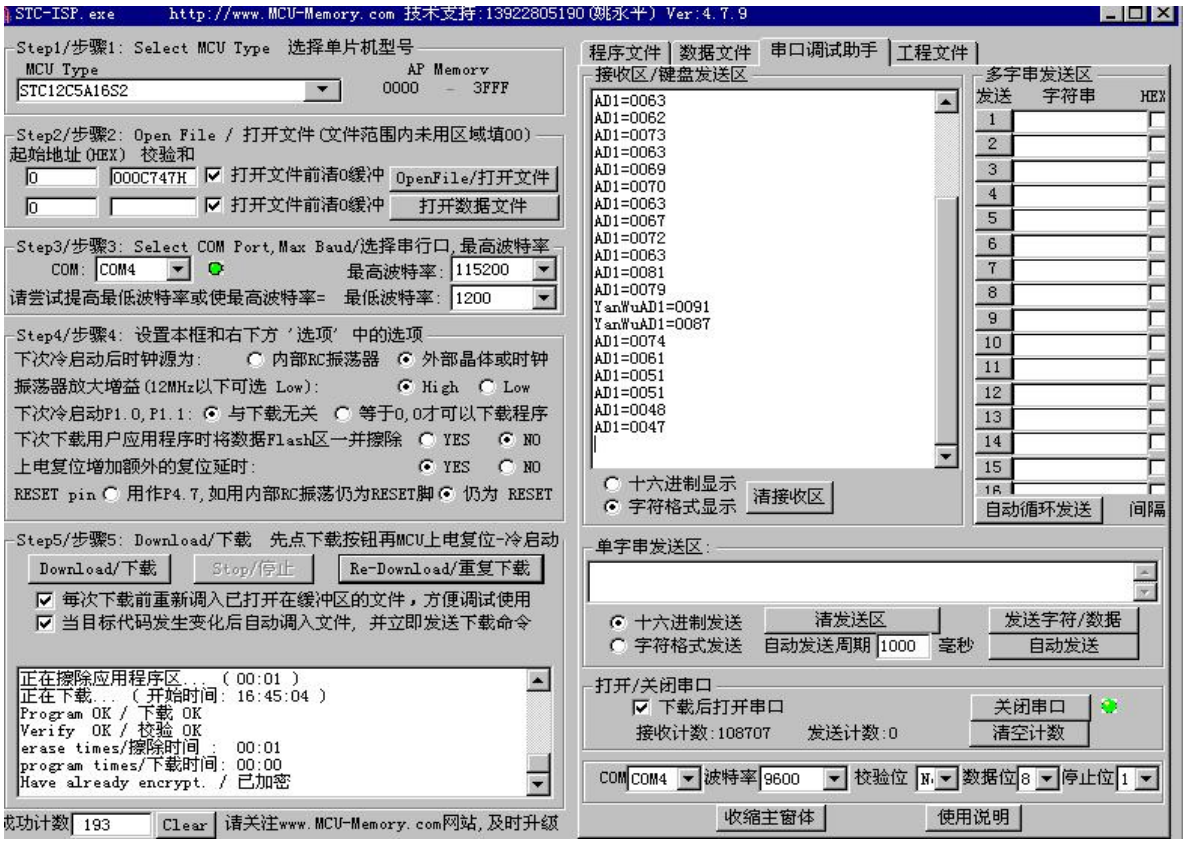


图4.1 烟雾传感器运行时串口输出信息

注意：测试的时候要避免火灾，如果室内安装有烟雾传感器的报警系统那么烟雾过大可能触发报警系统。

串口参数：9600 8 N 1