毕业设计（论文）检测系统

文本复制检测报告单(全文标明引文)

№:BC20230516120058103690089 72　　检测时间:2023-05-16 12:00:58

篇名：基于VL6180的接近检测系统设计

作者：高生洋(B19030405)

指导教师：路纲

检测机构：洛阳理工学院

文件名：毕业论文.docx

检测系统：毕业设计（论文）检测系统（毕业设计（论文）管理系统）

检测类型：毕业设计论文

检测范围：中国学术期刊网络出版总库
中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库
中国重要会议论文全文数据库
中国重要报纸全文数据库
中国专利全文数据库
图书资源
优先出版文献库
大学生论文联合比对库
互联网资源(包含贴吧等论坛资源)
英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)
港澳台学术文献库
互联网文档资源
源代码库
CNKI大成编客-原创作品库

时间范围：1900-01-01至2023-05-16

检测结果

去除本人文献复制比：0.6%　　跨语言检测结果：-

去除引用文献复制比：0.6%　　总文字复制比：0.6%

单篇最大文字复制比：0.2%　(6834632_别样珺_基于深度学习的倾斜车牌定位研究)

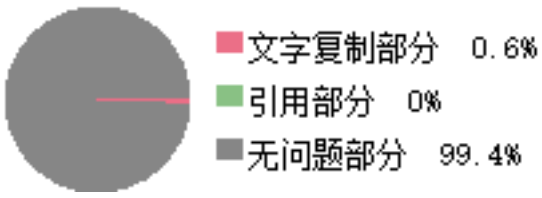
重复字数：[242]　　总段落数：[6]

总字数：[40929]　　疑似段落数：[3]

单篇最大重复字数：[73]　　前部重合字数：[173]

疑似段落最大重合字数：[144]　　后部重合字数：[69]

疑似段落最小重合字数：[29]



指标：☐ 疑似剽窃观点　☑ 疑似剽窃文字表述　☐ 疑似整体剽窃　☐ 过度引用

相似表格：0　　相似公式：检测中　　疑似文字的图片：0

1.2%(29)	1.2%(29)	中英文摘要等（总2373字）
6.4%(144)	6.4%(144)	第1章绪论（总2241字）
0%(0)	0%(0)	第2章相关技术和理论分析（总3867字）
0%(0)	0%(0)	第3章系统设计与实现（总9596字）
0.5%(69)	0.5%(69)	第4章系统测试与性能评估_第1部分（总12817字）

（注释：无问题部分 文字复制部分 引用部分）

1. 中英文摘要等 总字数：2373

相似文献列表

去除本人文献复制比：1.2%(29) 去除引用文献复制比：1.2%(29) 文字复制比：1.2%(29) 疑似剽窃观点：(0)

1 基于物联网的农业环境监测系统的研究与设计 1.2%（29）

李雪刚(导师：朱东海;黄梦醒) - 《海南大学硕士论文》- 2012-05-01 是否引证：否

原文内容

洛阳理工学院学位论文原创性声明

本人郑重声明：所呈交的毕业设计及学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人或集体，均已在文中以明确方式表明。本人完全意识到本声明的法律结果由本人承担。

作者签名：

年月日

洛阳理工学院学位论文版权使用授权书

本论文作者完全了解学校有关保留、使用毕业设计及学位论文的规定，学生在校学习期间毕业设计及论文的知识产权单位归属洛阳理工学院。同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权洛阳理工学院可以将本学位论文的全部和部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

作者签名：

指导教师签名：

年月日

基于VL6180的接近检测系统设计

摘要

本文设计了一种基于VL6180传感器的接近检测系统。该系统通过测量物体与传感器之间的距离，并将数据显示到屏幕，实现对物体的接近检测和距离测量。在硬件方面，自行设计电路使用国产乐鑫ESP32作为主控芯片，VL6180传感器作为距离检测模块，并配备了OLED显示屏等部件。在软件方面，采用C++编程语言编写程序，实现数据的实时采集和传输，并编写Python脚本进行调试和性能评估，最终设计了一种低成本、高精度、高稳定性和高实时性的接近检测系统。

此外，本文还详细介绍了硬件电路和系统软件的设计过程，为了验证系统的性能和可靠性，本文进行了全面的测试和调试。测试结果表明，该接近检测系统具有良好的稳定性、准确性和可靠性，能够满足不同场景的需求和期望。

关键词：VL6180X，ESP32，接近检测，嵌入式

Proximity Detection System Design Based on VL6180

ABSTRACT

This paper designs a proximity detection system based on the VL6180 sensor. The system achieves proximity detection and distance measurement of objects by measuring the distance between the object and the sensor and displaying the data to the screen. In terms of hardware, the self-designed circuit uses the domestic Espressif ESP32 as the main control chip, the VL6180 sensor as the distance detection module, and is equipped with OLED display and other components. In terms of software, C++ programming language is used to write the program for real-time data acquisition and transmission, and Python scripts are written for debugging and performance evaluation, finally designing a proximity detection system with low cost, high accuracy, high stability and high real-time performance.

In addition, the design process of hardware circuit and system software is detailed in this paper. To verify the performance and reliability of the system, comprehensive testing and debugging are carried out in this paper. The test results show that the proximity detection system has good stability, accuracy and reliability, and can meet the needs and expectations of different scenarios.

KEY WORDS: VL6180, ESP32, Proximity detection, Embedded

目录

前言

随着智能化、自动化和信息化的发展，人们对于自动化设备和智能化系统的需求日益增加。在此背景下，接近检测技术成为了自动化控制、机器人技术和工业生产等领域中的重要应用技术之一。接近检测技术能够实现对物体和环境的自动检测和识别，从而实现自动化控制和智能化决策，具有广泛的应用前景。近年来，随着物联网技术和人工智能技术的不断发展，越来越多的智能化系统需要具备接近检测功能，以提高智能化程度和自动化程度。

本文研究了一种基于VL6180和ESP32-PICO-D4设计的接近检测系统，该系统能够实现对物体与传感器之间的距离测量。该系统不仅具有低成本高精度、高稳定性和高实时性等特点，还能够实现数据的实时采集和传输，适用于智能家居、安防监控、无人机、自动化控制、机器人技术和工业生产等领域的应用。

在本次研究中，对该系统的硬件和软件进行了详细设计和实现，并对系统进行了全面的测试和性能评估。在系统设计和实现过程中也遇到了一些挑战，如如何保证系统的稳定性和精度，如何兼顾系统的功能和成本等。通过不断地努力和实验，最终克服了这些挑战，最后通过对系统的测试和性能评估，验证了系统的可行性和实用性。

本课题的研究成果不仅具有实际应用价值，而且具有一定的创新性和实用性。本文的主要贡献在于提出了一种基于VL6180和ESP32国产芯片的接近检测系统设计方案，实现了物体距离的测量和显示及预警。采用的研究方法是结合理论分析和实际实验相结合，具有科学性、实用性和先进性。在论文写作过程中，我们秉承着创新、务实、严谨、协作的指导思想，以实现高品质的研究成果为目标，经过多次实验和不断修改，最终完成了该论文。

2. 第1章绪论　　总字数：2241

相似文献列表

去除本人文献复制比：6.4%(144)　去除引用文献复制比：6.4%(144)　文字复制比：6.4%(144)　疑似剽窃观点：(0)

1	6834632_别样珺_基于深度学习的倾斜车牌定位研究 别样珺 - 《大学生论文联合比对库》- 2019-05-01	3.3%（73） 是否引证：否
2	公共管理视域下高职院校学生管理工作研究 李凤莲(导师：谢奉军) - 《南昌航空大学硕士论文》- 2018-06-01	2.5%（57） 是否引证：否
3	电子信息工程学院-P11614064-陈林 电子信息工程学院 - 《大学生论文联合比对库》- 2020-10-14	2.5%（55） 是否引证：否
4	1110513_陈相_计算机科学与技术 陈相 - 《大学生论文联合比对库》- 2015-05-07	1.9%（43） 是否引证：否
5	基于主题模型的微博话题检测与跟踪研究 谢黎黎(导师：黄华军) - 《中南林业科技大学硕士论文》- 2015-06-01	1.6%（36） 是否引证：否
6	基于L21范数的多图正则化非负矩阵分解算法研究 周长宇(导师：姚明海) - 《渤海大学硕士论文》- 2020-05-01	1.6%（36） 是否引证：否

原文内容

第1章绪论

1.1 研究背景和意义

近年来，工业控制和制造业快速发展，对生产效率、质量和安全性的要求越来越高。为了提高生产效率和质量，智能化检测和控制技术得到了广泛的应用。在工业自动化控制中，常常需要对工件或机器人的距离进行准确的检测，这就需要一种高精度、高速度、非接触式的距离检测技术。

目前，传统的距离检测技术主要包括机械式、光学式、超声波式等，这些传统的距离检测技术在应对现代工业应用中仍存在一些问题，如受环境干扰、易受测量目标材料影响等。因此，发展高精度、高速度、非接触式的距离检测技术已成为一种必然趋势。

近年来，基于时间飞行原理的光电式接近检测技术逐渐发展成为一种可靠、精度高、速度快、适应范围广的非接触式距离检测技术。VL6180是一款基于时间飞行原理的光电式接近检测传感器，具有高度集成、低功耗、多种测距模式等优点。本课题基于VL6180和ESP32-PICO-D4国产芯片设计并实现了一个接近检测系统，通过测量物体与传感器之间的距离，并将距离值显示在OLED显示屏上，实现了精准、远距离的接近检测。

因此，本课题旨在利用VL6180传感器和ESP32-PICO-D4，设计并实现一种高精度、高速度、非接触式的距离检测系统，为工业自动化控制提供更可靠、高效、智能化的解决方案。该系统可应用于物流、智能家居、机器人等领域，能够提高生产效率、降低生产成本，同时提高生产安全性和质量。

1.2 研究内容和方法

本课题主要研究基于VL6180传感器和ESP32-PICO-D4芯片，设计并实现一种高精度、高速度、非接触式的距离检测系统。该系统采用基于时间飞行原理的光电式接近检测技术，能够实现对工件或机器人的远距离精确检测，同时具有高度集成、低功耗、低成本等优点。具体内容包括硬件设计与实现、软件设计与实现、系统测试与性能评估等方面。

硬件设计与实现方面，主要包括电路设计与实现、PCB设计与实现等。其中电路设计需要根据VL6180传感器的原理和ESP32-PICO-D4芯片的特性，设计出符合要求的电路，确保系统能够稳定、准确的进行距离检测。PCB设计则需要将电路设计转化为PCB布局，并制作出高质量的PCB板。

软件设计与实现方面，主要包括系统架构和模块设计、软件算法设计与实现、软件测试和调试等。系统架构和模块设计需要根据系统的硬件特性，设计出合理、稳定、高效的软件架构，并实现相应的软件模块。软件算法设计和实现则需要结合VL6180传感器的测量原理，实现精准的距离检测算法。软件测试和调试则需要对软件进行全面的测试，确保系统的稳定性和可靠性。

系统测试与性能评估方面，主要包括系统测试介绍和系统测试结果分析。系统测试介绍需要详细说明测试方案和测试环境，并说明测试的目的和方法。系统测试结果分析则需要对测试结果进行详细分析和评估，评价系统的性能和稳定性，并与其他距离检测技术进行比较。

本课题的研究方法主要包括以下方面：

(1) 文献调研。对接近检测技术和相关芯片、传感器进行深入了解和研究，了解现有技术的优缺点和发展趋势。

(2) 硬件设计。根据文献调研和需求分析，选择合适的电子元器件和传感器，进行电路图设计和PCB布局设计。

(3) 软件设计。基于ESP-IDF和ArduinoIDE开发环境，使用C++语言编写程序代码，实现距离测量和数据显示功能。

(4) 系统测试与性能评估。对距离检测系统进行测试，测试结果包括距离测量精度、稳定性、响应速度等性能指标的测试和分析。

1.3 论文结构和安排

基于上述的研究内容和方法，将本文分为六个部分，内容安排如下：

第一部分：绪论。主要介绍本论文的研究背景、研究目的、研究内容和方法、研究意义和国内外研究现状等。通过对相关领域的综述和分析，引出本研究的创新点和价值，并阐述本文的研究思路、理论基础和研究方法。

第二部分：相关技术和理论分析。主要介绍本论文涉及的相关技术和理论分析。首先介绍时间飞行原理和基于时间飞行原理的光电式接近检测技术，然后详细介绍VL6180传感器和ESP32-PICO-D4芯片及其他必要元器件的特性和工作原理，最后分析和比较现有的距离检测技术。

第三部分：系统设计与实现。主要介绍距离检测系统的设计与实现。分为硬件设计与实现、软件设计与实现和系统测试与性能评估三个部分。硬件设计与实现部分介绍电路设计和PCB布局设计，软件设计与实现部分介绍系统架构和模块设计、软件算法设计与实现、软件测试和调试等，系统测试与性能评估部分介绍测试方案和测试结果分析等。

第四部分：系统测试与性能评估。主要对距离检测系统的性能进行测试分析和优化。分为系统性能分析、系统性能优化和实验分析三个部分。系统性能分析部分详细分析系统的性能指标，如测量精度、稳定性、响应速度等。系统性能优化部分针对系统存在的性能问题进行优化，提高系统的性能。实验分析部分对系统进行实验验证，并对实验结果进行分析和总结。

第五部分：结论与展望。在结论部分主要介绍距离检测系统的应用和未来展望，对本研究的成果进行总结和评价，并阐述研究的贡献和不足之处，最后探讨距离检测技术的发展趋势和未来的应用方向。

最后一部分是参考文献和附录。参考文献部分需要列出本研究所引用的所有文献，包括期刊论文、会议论文、专利、书籍等。在附录部分，包括一些与本研究有关的技术资料、数据、图表、代码等用于必要的辅助材料。

指　　标

疑似剽窃文字表述

1. 测试和分析。
1.3 论文结构和安排
基于上述的研究内容和方法，将本文分为六个部分，内容安排如下：

第一部分：绪论。主要介绍本论文的研究背景、研究目的、研究内容和方法、研究意义和国内外研究现状等。

3. 第2章相关技术和理论分析 总字数：3867

相似文献列表

去除本人文献复制比：0%(0) 去除引用文献复制比：0%(0) 文字复制比：0%(0) 疑似剽窃观点：(0)

原文内容

第2章相关技术和理论分析

2.1 接近检测技术综述

近年来，接近检测技术已经得到了广泛应用，不仅在现代工业中，而且在日常生活中也十分常见。接近检测技术主要用于检测物体的存在、位置、形状等信息，以及物体与物体之间的距离和接触状态等。常用的接近检测技术包括机械接触式检测技术、光电式接近检测技术、超声波检测技术和磁性检测技术等。其中，光电式接近检测技术因其非接触式、高精度、高可靠性、高速度等优点，已经成为目前应用最广泛的一种接近检测技术。

光电式接近检测技术主要利用光电传感器检测物体与光电传感器之间的距离，常用的光电传感器有反射式光电传感器、散射式光电传感器、吸收式光电传感器和时间飞行式光电传感器等。其中，时间飞行式（Time-of-Flight）光电传感器是一种高精度、高速度的光电传感器，能够实现毫米级别的精度和纳秒级别的响应速度。因此，在许多应用场景中，时间飞行式光电传感器已经成为首选的接近检测技术。时间飞行式（TOF）光电传感器主要利用了光脉冲在空气或其他介质中传播的速度恒定这一特性，通过测量光脉冲在传播过程中所需要的时间来计算物体与传感器之间的距离。时间飞行式光电传感器的工作原理简单、精度高、可靠性强、响应速度快，因此被广泛应用于机器人、安防、测距仪等领域。

综上所述，接近检测技术在现代工业和生活中具有重要的应用价值。其中，光电式接近检测技术因其非接触式、高精度、高可靠性、高速度等优点，已经成为目前应用最广泛的一种接近检测技术。而时间飞行式光电传感器作为其中的一种高精度、高速度的光电传感器，已经被广泛应用于机器人、安防、测距仪等领域。

2.2 VL6180传感器技术原理及应用

2.2.1 VL6180技术原理

VL6180传感器采用时间飞行式（TOF）原理进行测量，利用光脉冲在空气或其它介质中传播的速度恒定这一特性，通过测量光脉冲在传播过程中所需要的时间来计算物体与传感器之间的距离。

具体来说，VL6180传感器发射一组红外光脉冲，然后测量光脉冲在空气中传播，到达待测物体表面并被反射回传感器所需的时间。如图2-1所示，VL6180通过内置的接收器接收反射光信号，并计算物体与传感器之间的距离。由于光脉冲传播速度固定，因此测量光脉冲传播时间可以计算出物体与传感器之间的距离。VL6180传感器通过内置的TOF测量模块精确测量了光脉冲发射和接收之间的时间间隔，即“飞行时间”，然后将其转换为物体与传感器之间的距离。

图2-1 ToF 原理图

除了基本的距离测量之外，VL6180传感器还具有一些附加功能，例如灵活的测量范围、多级增益控制、环境光抑制等功能。这些附加功能可以使VL6180传感器在不同的应用场景中实现更加精准的距离测量。

2.2.2 VL6180技术特性

VL6180传感器采用了时间飞行（time-of-flight）测量原理，它具体的实现原理是使用简单的管道架构实现距离测量，从传感器发出红外脉冲到光敏元件接收到反射回来的信号，再转换为电信号，整个过程中管道架构起到了关键的作用。VL6180通过将测量电路中的多个模块组合成一个流水线，使得每个模块可以并行处理多个距离测量请求，从而大大提高了测量效率和速度[9]。具体来说，管道架构包括以下几个模块，如图2-2所示：。

图2-2 VL6180的测距管道架构

(1) 激光控制模块：控制激光器发射脉冲，保证激光在正确的时间点发射。

(2) 时钟控制模块：控制电路中的时钟，确保时间测量的精度和准确性。

(3) 光电转换模块：将接收到的光信号转换为电信号，并对信号进行放大和滤波处理，以提高信噪比。

(4) 时间测量模块：测量从激光器发射脉冲到接收到反射光的时间差，并计算出目标物体与传感器的距离。

通过将这些模块组合成一个流水线，VL6180可以实现高效的距离测量，同时还可以支持多个测量请求的并行处理。这种简单而高效的管道架构使得VL6180成为了一种非常实用的激光测距传感器，总结一下VL6180的技术特性就是：

(1) 测量范围：VL6180传感器具有灵活的测量范围，可以在0至100毫米之间进行测量。此外，它还具有高度可定制化的测量范围，可以根据具体应用场景进行调整。

(2) 精度：VL6180传感器具有高精度的距离测量功能，可以实现毫米级别的距离测量精度。该传感器内置了TOF测量模块，可以精确测量光脉冲发射和接收之间的时间间隔，从而实现高精度的距离测量。

(3) 环境适应性：VL6180传感器可以适应不同的环境光照强度，具有良好的环境适应性。它内置了环境光抑制功能，可以抑制来自外部光源的干扰，从而提高测量精度。

(4) 多级增益控制：VL6180传感器具有多级增益控制功能，可以根据不同的应用场景进行调整。在低光照环境下，传感器可以自动增加增益，提高信号强度，从而实现更加精准的距离测量。

(5) 低功耗：VL6180传感器具有低功耗的特性，可以在待机模式下消耗极少的电能。在实际应用中，这种低功耗特性可以延长传感器的使用寿命，并减少电源需求。

(6) 通信接口：VL6180传感器支持多种通信接口，包括I2C、SPI和GPIO等。这种通信接口的灵活性可以满足不同应用场景的需求，实现数据的快速传输和处理。

2.3 ESP32-PICO-D4芯片介绍及应用

ESP32-PICO-D4是一款由乐鑫科技（Espressif Systems）开发的紧凑型Wi-Fi和蓝牙系统级芯片(SoC)。它是ESP32系列芯片的一员，与其他ESP32芯片一样，它采用了TSMC的40纳米工艺，具有强大的处理能力、低功耗、高度集成的特点，可用于物联网、智能家居、工业自动化等领域的应用。

ESP32-PICO-D4芯片的主要特点如下：

(1) 双核处理器：ESP32-PICO-D4芯片内部集成了两个Xtensa® 32位LX6单元处理器核心，主频高达240MHz，可同时运行多个任务。

(2) Wi-Fi和蓝牙：ESP32-PICO-D4支持IEEE 802.11b/g/n Wi-Fi以及蓝牙v4.2 BLE协议，能够在不同的无线网络环境下实现高速、稳定的数据传输。

(3) 内存和存储：ESP32-PICO-D4芯片内部集成了520KB的SRAM，4MB的Flash存储器，并支持外部Flash和SRAM扩展。此外，该芯片还集成了SPI、SDIO/SPI、I2C、UART、I2S、IR、PWM和GPIO等多种接口。

(4) 安全性：ESP32-PICO-D4具有多种安全功能，例如支持AES、SHA-2、RSA和ECDSA等加密算法，以及支持TLS/SSL和WPA/WPA2加密协议等。

(5) 小巧封装：ESP32-PICO-D4芯片采用QFN封装，尺寸仅为7mm x 7mm，方便在小型电路板和设备中应用。

ESP32-PICO-D4芯片的特性使得它在物联网、智能家居、工业自动化等领域的应用中具有广泛的应用前景。同时，ESP32-PICO-D4还支持开源的ESP-IDF软件开发套件，使得开发者可以轻松进行软件开发和调试。

ESP32-PICO-D4的核心是ESP32芯片，ESP32-PICO-D4模组已将晶振、滤波电容、flash、RF匹配链路等所有外围器件无缝集成进封装内（原理图见附录），不再需要外围器件即可工作。由于无需外围器件，模组焊接和测试过程也可以避免，因此ESP32-PICO-D4可以大大降低供应链的复杂程度并提升管控效率。

在基于VL6180传感器的接近检测系统中，ESP32-PICO-D4芯片作为主控芯片，可以实现多种功能。首先，它可以通过内置的Wi-Fi模块实现数据传输和远程监控。其次，它具有丰富的GPIO口，可以通过GPIO口控制VL6180传感器的工作模式，并且可以通过SPI或I2C接口与VL6180传感器进行通信。另外，ESP32-PICO-D4芯片还具有PWM和ADC等功能模块，可以用于控制和采集VL6180传感器的信号。

2.4 OLED显示屏技术原理及应用

OLED有机发光二极管是一种光电材料，具有自发光的特性。OLED显示屏采用有机材料作为发光层，通过对不同材料电流的控制，产生红、绿、蓝三种基本颜色的光，并经过亮度的调节，形成彩色图像。相比传统的液晶显示屏，OLED显示屏具有更高的对比度、更广的视角、更快的响应速度和更低的能耗。

OLED显示屏由许多微小的OLED单元组成，每个OLED单元由一个发光层、一个电子传输层和两个电极组成。在一个OLED单元中，当正极电压被施加到阳极上时，电子从阴极流向阳极，并在电子传输层中与空穴结合，产生能量释放，发射出光子，形成像素的颜色。与液晶显示屏相比，OLED显示屏的亮度、颜色饱和度、对比度等方面都有更好的表现。

OLED技术因其低功耗、高对比度、广视角、高亮度等优势，被广泛应用于电子产品中，OLED显示屏可以用于显示实时测量的距离值、警报信息等。在我的课题中，我使用了0.91寸的SSD1306 OLED显示屏，它是一款低功耗、高对比度、广视角、高亮度的显示屏。它具有32x128的分辨率，可以显示简单的图标、文字、数字等，适合用于实时显示距离值。SSD1306 OLED显示屏采用I2C总线接口，可以与微控制器连接，通过编程实现控制显示内容。

第3章系统设计与实现

3.1 硬件设计与实现

本课题设计的接近检测系统主要是通过VL6180测量物体与传感器之间的距离，并将数据传输到ESP32-PICO-D4芯片上进行处理，最终通过OLED显示屏上显示出距离值。在硬件设计过程中需要考虑以下几个方面：VL6180传感器的驱动电路设计、ESP32-PICO-D4芯片的驱动电路设计、整个板载电源的设计等，最后还要考虑如何PCB布局。

3.1.1 电路设计与实现

1. 电源电路

该电源电路主要通过3.7V的外置电池或5V的直流电源为电路供电，并通过一个单刀双掷开关手动选择。两种电源都通过3.3V的稳压LDO输出稳定的板载电源，同时还使用了TP4054芯片，可以通过Type-c为外置锂电池充电。电路图如3-1所示：

图3-1 双路电源及稳压

2. 双路供电的切换设计

该电路采用单刀双掷开关来手动切换外置电池和type-c接口获取的直流电源。开关的一端连接外置电池正极，另一端连接type-c接口的正极。当开关处于外置电池侧时，外置电池为电路供电；当开关处于type-c接口侧时，type-c接口获取的直流电源为电路供电。

3. 稳压LDO的使用及参数选择

为确保电路的稳定供电，如图3-1，该电路使用了3.3V的稳压LDO。在电路中需要添加1uF的输入和输出电容来保证电路的稳定性和抗干扰能力。在选择LDO时需要考虑其最大输入电压、输出电流和线性调节率等参数，以满足电路的需求。

外置锂电池充电保护电路的设计

为保护外置锂电池，在电路中还需要使用TP4054芯片进行充电管理。TP4054芯片集成了过充保护、欠压保护和温度保护等功能，可以提高电池的安全性和可靠性。

在TP4054芯片中，充电电流是采用一个连接在PROG引脚与地之间的电阻来设定的，可以使用下面的两个公式来计算：在我的课题中使用了600mAh的电池，充电电流是0.1A，则我需要放置电阻的大小为：即，在电路中还需要添加电位器来设置充电电流，并通过电阻分压来检测充电状态。如图3-2所示。

图3-2 Type-C和TP4054

4. 板载串口调试的设计

既然为锂电池充电使用了type-c接口那么不妨利用type-c接口的强大数据传输功能，使用CP2102芯片实现串口通信功能，如图3-3所示。

图3-3 CP2102串口通信

CP2102是一种USB转UART芯片， 由Silicon Labs公司开发。它可以将UART信号转换为USB信号，实现串口通信功能，并通过USB接口与计算机进行连接。方便串口下载、打印和调试，为之后的系统测试和优化提供便利。

5. VL6180驱动电路

VL6180 需要2.8V的稳定电压才能正常工作，因此驱动电路中使用了线性稳压器将VIN转换为VL6180 IC的2.8V电源。IIC引脚连接到内置电平转换器，使它们可以在超过2.8V的电压下安全使用，同时引出了两个可编程GPIO的漏极开路输出引脚。原理图如图3-4所示：

图3-4 VL6180驱动电路

6. 其他电路

ESP32-PICO-D4正常工作需要3.3V的稳定电压，板载电源从电源电路经过3V3的LDO获取，同时为芯片设计了两个按键方便调试。OLED屏幕和VL6180可以共用一对IIC引脚，分别引入屏幕和传感器针脚的插槽，原理图如图3-5所示：

图3-5 其他电路

3.1.2 PCB设计与实现

本项目使用的是国产EDA软件：立创EDA进行的设计，包括电路图和PCB都是在上面完成。在 PCB 设计中，有许多技巧和方法可以应用来提高设计效率和准确性。PCB 布局对信号传输、散热和 EMC 抗干扰性能都有着重要影响。因此，在 PCB 设计中，需要优化布局，以确保最佳信号完整性和最小的 EMC 问题。例如，使用平面铺铜层来确保信号传输的可靠性和抑制噪声。

线宽是非常重要的参数之一。过粗或过细的线宽都可能会影响电路板的性能和可靠性。在我的设计中最高的电源电压是5V，板载负载都比较小，在考虑到电流容量、阻抗、焊盘大小和空间等情况下电源线路的线宽可以设置为10mil，其他信号线可以适当缩小至8mil。

最终本课题设计的PCB板子如下图3-6和3-7所示。

图3-6 主板PCB 实物图

图3-7 主板PCB 3D视图

设计好的PCB在嘉立创打样，从立创商城购买元器件，自己手动焊接，最终板子效果如下图：

图3-8 实物图

3.2 软件设计与实现

我的主力机是一台ArchLinux所以本课题使用的软件开发环境是ESP32基于Arduino的开发框架，该框架提供了许多简单实用的API（应用程序接口），使用了VS Code和PlatformIO作为主要工具，使用C++来编写程序代码。通过PlatformIO插件来管理Arduino库、开发板配置和项目构建。PlatformIO还支持固件更新、串口监视器、调试功能等，大大提高了项目开发的效率和可靠性。

3.2.1 系统架构和模块设计

1. VL6180驱动设计

与VL6180的通信是通过IIC总线进行的，VL6180X芯片中的一些寄存器是相关的，它们的值可能会在某些条件下相互影响。在修改这些寄存器的值时，为了避免读取到不正确的值，可以使用SYSTEM__GROUPED_PARAMETER_HOLD寄存器将这些寄存器的值锁定。在锁定状态下，芯片将不会读取与当前组相关的寄存器的值，直到锁定被解除为止。在修改与当前组相关的寄存器的值之后，必须将它的比特位设置为0，以释放这些寄存器的值。否则，芯片可能会继续使用被锁定的寄存器的旧值，导致意想不到的结果。因此，使用一些特殊的寄存器可以确保正确地修改相关寄存器的值，并避免读取到不正确的值。下表是VL6180常用的几个寄存器及其地址：

表3-1 VL6180常用寄存器

地址	注册名称	说明
0x016	SYSTEM__FRESH_OUT_OF_RESET	检查复位状态，默认为1
0x212	I2C_SLAVE__DEVICE_ADDRESS	可编程IIC地址
0x012	SYSTEM__HISTORY_CTRL	控制历史数据的存储和检索
0x015	SYSTEM__INTERRUPT_CLEAR	清除各种中断标志位
0x017	SYSTEM__GROUPED_PARAMETER_HOLD	安全锁防止数据更新中被读取
0x018	SYSRANGE__START	触发测距操作
0x019	SYSRANGE__THRESH_HIGH	设置测量距离的高阈值
0x01A	SYSRANGE__THRESH_LOW	设置测量距离的低阈值
0x062	RESULT__RANGE_VAL	存储最近一次测量的距离值
.....	.....	.....

地址注册名称说明

0x016 SYSTEM__FRESH_OUT_OF_RESET 检查复位状态，默认为1

0x212 I2C_SLAVE__DEVICE_ADDRESS 可编程IIC地址

0x012 SYSTEM__HISTORY_CTRL 控制历史数据的存储和检索

0x015 SYSTEM__INTERRUPT_CLEAR 清除各种中断标志位
0x017 SYSTEM__GROUPED_PARAMETER_HOLD 安全锁防止数据更新中被读取
0x018 SYSRANGE__START 触发测距操作
0x019 SYSRANGE__THRESH_HIGH 设置测量距离的高阈值
0x01A SYSRANGE__THRESH_LOW 设置测量距离的低阈值
0x062 RESULT__RANGE_VAL 存储最近一次测量的距离值
....

寄存器SYSTEM__FRESH_OUT_OF_RESET是VL6180X中的一个系统寄存器，用于指示芯片是否刚刚从复位状态中恢复。该寄存器包含一个比特位，如果该比特位被设置为1，则表示芯片刚刚从复位状态中恢复，如果该比特位被设置为0，则表示芯片已经经过了复位状态。

当VL6180X作为一个I2C从机时，在我的项目中需要指定其唯一的地址，以便主机可以与之通信。该地址通过写入I2C_SLAVE__DEVICE_ADDRESS寄存器来设置。默认情况下，VL6180X的I2C地址是0x29。可以通过将这个寄存器的值更改为其他值来修改此地址。。因此，使器可以确保正确地修改相关寄存器的值，并避免读取到不正确的值。

VL6180从上电到完成初始化的步骤如下图3-9：

图3-9 VL6180上电初始化

(1) 第一步检查设备状态：首先需要检查VL6180X芯片是否已经上电检查，这可以通过读取寄存器SYSTEM__FRESH_OUT_OF_RESET 来实现。如果该寄存器值等于0x01，说明芯片已经上电检查完成。

(2) 第二步加载SR设置：将SR设置加载到VL6180X芯片中，以便进行后续的距离测量。

(3) 第三步应用其他特定设置：根据具体应用需求，可能需要应用其他特定设置，例如串扰、GPIO管脚配置、最大收敛时间等。

(4) 第四步确认设置已加载：为了确保设置已经成功加载到VL6180X芯片中，需要将0x00写入SYSTEM__FRESH_OUT_OF_RESET寄存器。这可以帮助程序确定是否已加载设置。

(5) 最后一步准备测距：初始化完成后，VL6180X芯片已经可以进行距离测量。此时可以将芯片置于测量模式，并准备开始测距。

加载设置后，可以选择两种测距模式，单次模式和连续模式，在单次模式下，VL6180执行单次量程测量，完成后进入软件待机，连续模式则按照程序确定的速率执行，执行范围测量的步骤如下：

图3-10 VL6180执行范围测量

在单次模式下，测距只会在外部触发信号的情况下进行一次，即每次需要给芯片发送一个测距触发信号才能进行测距。测距完成后，芯片会输出测距结果并进入待机状态。如果需要再次进行测距，需要重新发送触发信号。

在连续模式下，芯片会以设定的测量间隔进行连续测距，每次测距完成后会输出测距结果并等待下一次测距触发。如果芯片接收到新的触发信号，则会立即开始下一次测距，否则会一直进行连续测距，直到主机发送停止命令为止。

单次模式适用于需要按需进行测距的应用场景，例如无人机避障、智能门铃等。因为单次模式只在接收到触发信号时才会进行测距，可以有效减少芯片功耗，同时也避免了频繁测距对信噪比的影响。

连续模式适用于需要实时、连续监测目标位置变化的应用场景，例如机器人导航、智能安防等。由于连续模式可以实时输出测距结果，因此可以较为精确地掌握目标位置的变化情况，同时也可以在不影响测距精度的情况下提高测量速度。不过需要注意的是，由于连续模式需要不断进行测距，因此芯片的功耗也会相应增加。

2. 系统架构设计

系统由两个主要部分组成：主机和传感器节点。主机负责控制传感器节点、收集距离数据并将其发送到OLED显示屏上。传感器节点负责测量物体与传感器之间的距离，并将距离值发送给主机。

具体来说，整个系统包括以下模块和组件：

(1) 传感器模块：包括VL6180接近传感器和I2C通信模块，用于测量物体与传感器之间的距离并将其发送给主机。

(2) 主机模块：包括ESP32-PICO-D4芯片、WiFi模块、OLED显示屏和UI控制模块，用于接收来自传感器节点的数据并将其显示在OLED屏幕上。此外，主机还可以通过WiFi连接到Internet或其他设备，以实现远程监控和控制。

(3) 软件模块：包括ESP-IDF开发框架、Arduino IDE、C语言编译器和相关库文件，用于编写和调试主机和传感器节点程序。其中，ESP-IDF框架提供了多种API，如WiFi连接、GPIO控制和OLED可视化等功能；Arduino IDE则提供了编程环境和自带的库文件，使程序编写更加方便。

(4) 数据处理模块：包括数据读取、校验和显示控制等功能，用于确保传感器节点测量数据的准确性和可靠性，并将其显示在OLED屏幕上。此外，还可以对数据进行分析和处理，以实现更复杂的应用场景。

3.2.2 软件设计与实现

根据前面的系统架构设计本课题采用如下的软件设计：

1. VL6180驱动

VL6180提供了很多寄存器，首先定义这些VL6180芯片中的常用寄存器及其地址如下，完整的定义见附件。然后定义VL6180X的C++类，包含一些成员函数和变量，用于控制VL6180传感器。

```
#define VL6180x_FAILURE_RESET -1 //复位
#define VL6180X_SYSTEM_HISTORY_CTRL 0x0012 //历史数据记录控制寄存器
#define VL6180X_SYSTEM_INTERRUPT_CLEAR 0x0015 //中断清除
#define VL6180X_SYSTEM_FRESH_OUT_OF_RESET 0x0016 //复位状态标志
#define VL6180X_SYSRANGE_START 0x0018 //距离测量启动寄存器
#define VL6180X_SYSRANGE_THRESH_HIGH 0x0019 //上限阈值
#define VL6180X_SYSRANGE_THRESH_LOW 0x001A //下限阈值
#define VL6180X_SYSALS_START 0x0038 //环境光强度测量启动
```

```
#define VL6180X_SYSALS_THRESH_HIGH 0x003A //环境光强度上限阈值
#define VL6180X_SYSALS_THRESH_LOW 0x003C //下限阈值
#define VL6180X_RESULT_ALS_VAL 0x0050 //环境光强度测量值
#define VL6180X_RESULT_RANGE_VAL 0x0062 //距离测量值
#define VL6180X_FIRMWARE_BOOTUP 0x0119 //固件启动状态
#define VL6180X_I2C_SLAVE_DEVICE_ADDRESS 0x0212 //I2C从设备地址
class VL6180X{
public:
//这里定义各种方法，详见附录
};
```

后面是一些具体功能实现的例子，例如考虑到VL6180X和OLED共用一组IIC，所以可以通过向寄存器VL6180X_I2C_SLAVE_DEVICE_ADDRESS写入新地址来解决IIC地址冲突的问题。

```
//修改IIC地址
uint8_t VL6180x::changeAddress(uint8_t old_address, uint8_t new_address){
if( old_address == new_address) return old_address;
if( new_address > 127) return old_address;
VL6180x_setRegister(VL6180X_I2C_SLAVE_DEVICE_ADDRESS, new_address);
return VL6180x_getRegister(VL6180X_I2C_SLAVE_DEVICE_ADDRESS);
}
```

VL6180x 传感器获取距离需要注意的是，在将系统距离测量寄存器设置为单发模式后，传感器会在每次测量结束后自动返回到空闲模式。因此，如果需要连续测量多个距离值，则需要在每次测量之间添加适当的延迟，并再次调用此函数以启动下一次测量。

```
uint8_t VL6180x::getDistance()
{
VL6180x_setRegister(VL6180X_SYSRANGE_START, 0x01); //启动单发模式从 0x01 改变
delay(10);
VL6180x_setRegister(VL6180X_SYSTEM_INTERRUPT_CLEAR, 0x07);
return VL6180x_getRegister(VL6180X_RESULT_RANGE_VAL);
//返回距离;
}
```

下面是VL6180x 传感器连续获取距离值的函数实现，该函数执行以下步骤：向寄存器 VL6180X_SYSRANGE_START 写入值 0x01，以启动一次新的测量。再次写入值 0x03开启连续测量模式，在连续测量模式下，距离值将被不断更新，读取寄存器 VL6180X_RESULT_RANGE_VAL 的值，并返回该值作为函数的输出。这个值就是当前测量得到的距离值。同理连续获取环境光强度值也是相同的方法实现（程序代码详见附录）。

```
uint8_t VL6180x::getDistanceContinously()
{
VL6180x_setRegister(VL6180X_SYSRANGE_START, 0x01); delay(10);
VL6180x_setRegister(VL6180X_SYSRANGE_START, 0x03); delay(10);
VL6180x_setRegister(VL6180X_SYSTEM_INTERRUPT_CLEAR, 0x07);
VL6180x_setRegister(VL6180X_SYSTEM_HISTORY_CTRL, 0x01); // 启用历史缓冲区
return VL6180x_getRegister(VL6180X_RESULT_RANGE_VAL);
}
```

在整个VL6180的驱动程序中，为VL6180X对象设计了一下的一些方法及其用途（方法的实现程序详见附录）如下表3-2

表3-2 VL6180X对象的方

方法名称	说明
getDistance	获取当前测量到的范围距离值
getAmbientLight	获取当前测量到的环境光强度值
changeAddress	修改设备IIC地址
VL6180xSetDistInt	设置范围距离中断阈值
VL6180xSetALSInt	设置环境光强度中断阈值
getDistanceContinously	连续获取范围距离值
getLastDistanceFromHistory	获取最近一次测量到的范围距离值。
VL6180xClearInterrupt	清除传感器中断标志位
.....	.....

法

方法名称说明

getDistance 获取当前测量到的范围距离值

getAmbientLight 获取当前测量到的环境光强度值

changeAddress 修改设备IIC地址

VL6180xSetDistInt 设置范围距离中断阈值

VL6180xSetALSInt 设置环境光强度中断阈值

getDistanceContinously 连续获取范围距离值
getLastDistanceFromHistory 获取最近一次测量到的范围距离值。
VL6180xClearInterrupt 清除传感器中断标志位
....
2. 系统软件
本课题是基于Arduino 开发的，实现了对 VL6180x 传感器的控制和显示。具体功能有：使用 Wire 库和 VL6180X.h 库连接和初始化 VL6180x 传感器。使用 U8g2lib 库驱动 OLED 显示屏，并在 OLED 上显示环境光强度和距离值。在串口监视器上输出当前环境光强度和距离值。根据测量到的距离值，判断距离是否合法（在 min_distance 和 max_distance 之间），并将结果在 OLED 上显示。
其中 setup() 函数用于初始化传感器和显示屏；loop() 函数用于循环读取传感器数据并更新 OLED 屏幕和串口输出。具体步骤如下：
(1) 声明全局变量 min_distance 和 max_distance，用于指定合法的距离范围。
在 setup() 函数中初始化串口、I2C 总线、OLED 显示屏和 VL6180x 传感器，读取传感器的标识信息并将其输出到串口监视器.
(2) 在 loop() 函数中，先使用 sensor.getDistance() 获取当前的距离值。
(3) 如果距离值在合法范围内，就在 OLED 上显示当前环境光强度和距离值；否则，显示“距离过近”或“距离过远”。
(4) 使用 sensor.getAmbientLight() 获取当前的环境光强度，并将其输出到串口监视器。
(5) 如果已经过了 50 毫秒，就延迟 50 毫秒。这是为了避免频繁读取传感器数据造成处理器负荷过高。
下图是本系统的软件架构：
图3-11 系统软件架构
程序代码如下：
```
... ... ...//部分重要程序，完整程序见附录
#define VL6180X_ADDRESS 0x29
#define OLED_ADDRESS 0x3C
... ... ...//详细的参数定义，见附录
U8G2_SSD1306_128X32_UNIVISION_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE, /* clock=*/ SCL, /* data=*/ SDA);
void printIdentification(struct VL6180xIdentification *temp){
//相关的调试代码，详见附录
}
void delay_ms(int ms) {//非阻塞性延迟函数，详见附录}
void setup() {//系统初始化
Serial.begin(9600);
Wire.begin();
u8g2.begin();
u8g2.enableUTF8Print();
delay(1000);
sensor.getIdentification(&identification);
printIdentification(&identification);
if(sensor.VL6180xInit() != 0){
Serial.println("VL6180初始化失败");
};
sensor.VL6180xDefautSettings();
delay(1000);
}
void loop() {
... ... ...//系统初始化相关代码，见附录
u8g2.clearBuffer();//OLED显示相关
u8g2.setFont(u8g2_font_wqy14_t_gb2312b);
u8g2.setFontDirection(0);
int distance = sensor.getDistance();
if (distance >= min_distance && distance <= max_distance) {
u8g2.setCursor(0, 15);
u8g2.print("光照：");
u8g2.print(sensor.getAmbientLight(GAIN_10));
u8g2.print(" Lux");
u8g2.setCursor(0, 30);
u8g2.print("距离：");
u8g2.print(distance);
u8g2.print(" mm");
} else if (distance < min_distance) {
```

```
u8g2.setCursor(0, 15);
u8g2.print("距离过近");
} else {
u8g2.setCursor(0, 15);
u8g2.print("距离过远");
}
u8g2.sendBuffer();
Serial.print("环境光级别 (Lux) = ");
Serial.println(sensor.getAmbientLight(GAIN_10));
Serial.print("测量距离 (mm) = ");
Serial.println(sensor.getDistance());
if (currentMillis - last_50ms >= 50) {
last_50ms = currentMillis;
Serial.println("Delay 50ms");
delay_ms(50);
}
};
```

需要注意的是，由于环境光和距离值受到环境因素的影响，在使用前校准传感器并根据实际情况调整相关参数。将在下一章节详细讲解。

5. 第4章系统测试与性能评估_第1部分　　总字数：12817

相似文献列表

去除本人文献复制比：0.5%(69)　去除引用文献复制比：0.5%(69)　文字复制比：0.5%(69)　疑似剽窃观点：(0)

1	基于工业物联网的造纸废水控制系统升级改造	0.3%（34）
	朱万浩;章盼梅;孔令棚; - 《仪表技术与传感器》- 2022-03-15	是否引证：否
2	大数据背景下信用卡诈骗罪的法律思考	0.2%（31）
	泽仁青中(导师：彭春凝) - 《西南民族大学硕士论文》- 2022-03-20	是否引证：否

原文内容

第4章系统测试与性能评估

本章主要围绕着距离检测系统的性能进行测试分析和优化。在实际应用中，距离检测系统需要具备高精度、高稳定性和高响应速度等性能指标，以满足各种应用场景的需求。因此，本章从系统性能分析、系统性能优化和实验分析三个方面对系统的性能进行了深入研究。

首先，系统性能分析部分详细分析了距离检测系统的各项性能指标，包括测量精度、稳定性、响应速度等，为后续的性能优化和实验分析奠定了基础。接着，系统性能优化部分重点关注了系统存在的性能问题，并提出了针对性的优化措施，以提高系统的性能。最后，实验分析部分通过一系列实验验证，对距离检测系统的性能指标进行了全面评估和总结，为后续的系统应用提供了科学的依据和支持。

4.1 系统性能测试与分析

通过合理的测试和调试可以验证系统各项功能是否正常运行。包括距离测量、数据传输、显示等方面。只有在经过充分测试和调试后，才能保证系统能够准确、稳定地运行，并提供良好的使用体验。测试和调试还可以验证接近检测系统在不同环境下的可靠性。由于不同环境下的光照、温度、湿度等因素可能会影响传感器测量数据的准确性和稳定性，因此需要通过测试和调试来确定系统的工作范围，以及在不同环境下需要采用的特定算法和参数。针对我的课题，应从三个指标入手，测量精度、响应速度和系统稳定性。

测量精度包括测量误差、重复性、精度等内容。在测试中，通过多次测量同一距离值，并记录每次测量的结果来计算测量误差。重复性则是指在同样的测量条件下，多次测量结果的相对差异程度。

为了测试系统的精度，我准备了两把尺子，将其放置在固定的位置，接着，使用已知距离对目标进行测量，记录下多次测量结果，最终从结果中分析系统的精度。在之前写的程序中为了方便调试顺便把每次的测量结果输出到了串口，刚好可以拿到这个数据进行分析。于是我又写了一个简单的测试程序，使用Python实现，用PySerial库来读取串口数据，用matplotlib库来绘制散点图[10]。程序详见附录。

试验中将传感器和物体之间的距离人为固定在20mm处，经过200次系统测距得到的数据如下图：

图4-1 精度测试散点图

在结果中可以看到200次测距的平均值（绿色虚线）和之前设定的20mm及其接近，红色虚线是平均值加减标准差，可以体现数据在平均水平附近的分散程度，至此试验结果超乎预期，距离检测的精度在±1mm。

系统响应时间是指从距离测量器接收到测量信号到在显示屏上显示结果之间的时间。我还测试了系统的刷新频率，即在不同距离值下，系统的显示屏更新速率。关于系统响应时间在测试过程中发现了一个致命的问题，在程序中使用阻塞性延迟函数，具体的将在下一节详细阐述。

稳定性是指系统在长时间运行时，保持测量精度和响应速度的能力。我对系统在不同环境条件下的稳定性进行了测试，并记录了系统在不同时间段内的性能指标变化情况。结果也很满意，在通过USB供电的试验期间，系统没有发生故障，系统稳定运行了3天（因试验条件有限，前后测试一共3天）。

功能测试：对系统的各项功能进行测试和验证，如距离测量、显示效果等；性能测试：对系统的性能进行测试和评估，如响应时间、精度和稳定性等；兼容性测试：针对不同的硬件和软件环境，测试系统的兼容性和可靠性。同时测试环境需要包括完整的硬件和软件组件，以确保系统的可测试性和真实性。

4.2 系统性能优化

根据前面的系统性能分析和测试，系统响应时间还有待优化。

在之前的系统软件中使用了Arduino中的一个库函数delay()，它会接受一个以毫秒为单位的整数值作为参数，并暂停程序的执行，直到经过指定的毫秒数。它的实现方式也很简单，将系统时钟频率除以 1000(1秒) 得到每毫秒时钟周期数，在延迟指定时间时，循环运行计数器等待指定的毫秒数。但是用在我的程序中，延迟的那段时间，其他进程无法运行，会导致测距频率的变慢，在工业上一些高速高频的场合中影响很大。

所以要考虑使用非阻塞延迟技术，刚好有一个millis函数，它会获取自系统启动以来经过的毫秒数，并将其存储到 startMillis 变量中。可以通过不断遍历系统启动时间来判别经过的毫秒数，因此程序可以同时执行其他任务而不会被阻塞，进而提高数据的采样频率。优化后的非阻塞延迟函数如下：

```
unsigned long lastMillis = 0; // 上一次调用时间戳
int delayTime = 1000; // 延迟时间（毫秒）
bool nonBlockingDelay() {
unsigned long currentMillis = millis();
if (currentMillis - lastMillis >= delayTime) {
lastMillis = currentMillis;
return true; // 延迟已结束
} else {
return false; // 延迟未结束
}
}
```

在主程序中，可以使用条件语句检查延迟是否已结束：

```
// 等待延迟结束
if (nonBlockingDelay()) {
// 执行下一步操作
}
```

测试和调试还可以验证接近检测系统在不同环境下的可靠性。由于不同环境下的光照、温度、湿度等因素可能会影响传感器测量数据的准确性和稳定性，因此需要通过长时间的测试和调试来确定系统在不同环境下需要采用的特定算法和参数。

结论

在本研究中，我设计了一种基于VL6180传感器的接近检测系统，并详细介绍了其硬件和软件的设计过程。通过对该系统进行全面的测试和调试，我发现该系统具有良好的稳定性、准确性和可靠性，在多个场景中能够实现对物体的接近检测和距离测量。

总的来说，该系统具有以下优点：

(1) 低成本：使用国产乐鑫ESP32作为主控芯片，采用VL6180传感器等低成本部件，使得整个系统的制造和维护成本较低。

(2) 高精度：通过采用高精度的VL6180传感器和数据处理算法，可以实现对物体的精确测量和定位，可测量精度在±1mm。

(3) 高稳定性：通过合理的电路设计和软件架构，使得该系统具有较高的工作稳定性和抗干扰能力。

(4) 高实时性：采用C++编程语言编写程序，实现数据的实时采集和传输，配备OLED显示屏等部件，使得该系统具有较高的实时性。

基于以上优点，我们相信该接近检测系统可以应用于多个领域中，为实现智能化自动化生产、安防监控、智能家居等提供技术支持。当然，该系统仍存在一些不足之处，例如测量距离受物体颜色和材质等因素的影响，还有待进一步优化和改进。我们将继续进行研究，进一步完善该系统的性能和应用场景，为实现智慧城市和工业4.0等提供更加优质的技术方案。

……

谢辞

行文至此，我的毕业设计也告一段落了，这是我本科阶段学习的一个重要组成部分，也是我对专业知识的实践性探索。在这个过程中，我不仅学到了理论知识，也获得了实践经验和技能，对于今后的学习和工作都有着积极的影响。在本篇论文的研究过程中，我受到了许多人的帮助和支持，在此向他们表示由衷的感谢。这是我第一次以论文致谢的方式向帮助和关心我的人表达感谢，想说的话太多太多了。

要感谢我的论文导师路纲老师，从论文选题开始，他对我的论文写作进行了专业、耐心、认真、负责地指导，多次提出宝贵的修改意见，本文在老师的指导下得以成搞。要感谢我的父亲母亲，能够始终理解我支持我，只愿父母身体健康，希望能快快成长，替父母分忧。要感谢我的朋友们，年少时的朋友是彼此的青春的收藏家，他们在我烦恼伤心时安慰我，在我收获时肯定我，带给我无数的快乐。

爱牡丹花都，爱古都洛阳。学子街8号，洛阳理工学院，感谢相遇，愿桃李满天下。追风赶月莫停留，平芜尽处是春山。

参考文献

[1] 张晶, 曾宪云. 嵌入式系统概述[J]. 电测与仪表, 2002, 39(4):4.

[2] A·E·波利奇. 光学距离检测:, CN110554400A[P]. 2019.

[3] 何立民. 单片机应用技术选编(10)[J]. 单片机与嵌入式系统应用, 2003(07):67-67.

[4] 官枫林，习友宝. 一种嵌入式芯片IP核-I2C接口的仿真验证方法[J]. 大众科技,2011(10):2.
[5] 刘科，谢敬辉，李卓. 被动式光学测距误差分析[J]. 光学技术，2005，31(4):3.
[6] 邱宏斌. 一种基于ESP8266模块的物联网设计思路[J]. 电子世界，2017(7):1.
[7] 曹振民陈年生马强武凌武婧. 基于ESP8266的无线控制电路设计[J]. 工业控制计算机，2017，030(001):68-69.
[8] 盛文利. 单片机C语言的精确延时程序设计[J]. 单片机与嵌入式系统应用，2004，000(010):67-69.
[9] 佚名. 意法半导体推出新款近距离传感器VL6180X[J]. 电子设计工程，2014，22(21):1.
[10] 黎右翼，薛双喜，周蓉. 一种基于Python的串口调试方法及系统:，CN202111335678.1[P]. 2022.

附录一

主控电路

VL6180驱动板电路

附录二

代码程序

```
// VL6180.h
#ifndef VL6180X_h_
#define VL6180X_h_
#if ARDUINO < 100
#include <WProgram.h>
#else
#include <Arduino.h>
#endif
#include <Wire.h>
#define VL6180x_FAILURE_RESET -1 //复位
#define VL6180X_IDENTIFICATION_MODEL_ID 0x0000 //传感器型号标识
#define VL6180X_IDENTIFICATION_MODEL_REV_MAJOR 0x0001 //传感器型号主版本号
#define VL6180X_IDENTIFICATION_MODEL_REV_MINOR 0x0002 //传感器型号次版本号
#define VL6180X_IDENTIFICATION_MODULE_REV_MAJOR 0x0003 //传感器模块主版本号
#define VL6180X_IDENTIFICATION_MODULE_REV_MINOR 0x0004 //传感器模块次版本号
#define VL6180X_IDENTIFICATION_DATE 0x0006 //传感器制造日期
#define VL6180X_IDENTIFICATION_TIME 0x0008 //传感器制造时间
#define VL6180X_SYSTEM_MODE_GPIO0 0x0010 //GPIO0
#define VL6180X_SYSTEM_MODE_GPIO1 0x0011 //GPIO1
#define VL6180X_SYSTEM_HISTORY_CTRL 0x0012 //历史数据记录控制寄存器
#define VL6180X_SYSTEM_INTERRUPT_CONFIG_GPIO 0x0014 //中断配置GPIO
#define VL6180X_SYSTEM_INTERRUPT_CLEAR 0x0015 //中断清除
#define VL6180X_SYSTEM_FRESH_OUT_OF_RESET 0x0016 //复位状态标志
#define VL6180X_SYSTEM_GROUPED_PARAMETER_HOLD 0x0017 //参数锁定
//SysRange类别，用于配置和控制距离测量相关的参数和操作。
#define VL6180X_SYSRANGE_START 0x0018 //距离测量启动寄存器
#define VL6180X_SYSRANGE_THRESH_HIGH 0x0019 //上限阈值
#define VL6180X_SYSRANGE_THRESH_LOW 0x001A //下限阈值
#define VL6180X_SYSRANGE_INTERMEASUREMENT_PERIOD 0x001B //距离测量之间的时间
#define VL6180X_SYSRANGE_MAX_CONVERGENCE_TIME 0x001C //距离测量收敛时间
#define VL6180X_SYSRANGE_CROSSTALK_COMPENSATION_RATE 0x001E //交叉干扰补偿率
#define VL6180X_SYSRANGE_CROSSTALK_VALID_HEIGHT 0x0021 //交叉干扰有效高度
#define VL6180X_SYSRANGE_EARLY_CONVERGENCE_ESTIMATE 0x0022 //早期距离测量估计值
#define VL6180X_SYSRANGE_PART_TO_PART_RANGE_OFFSET 0x0024 //芯片间距离偏差补偿
#define VL6180X_SYSRANGE_RANGE_IGNORE_VALID_HEIGHT 0x0025 //不计入有效高度控制寄存器
#define VL6180X_SYSRANGE_RANGE_IGNORE_THRESHOLD 0x0026 //不计入阈值的控制寄存器
#define VL6180X_SYSRANGE_MAX_AMBIENT_LEVEL_MULT 0x002C //最大环境光强度乘数
#define VL6180X_SYSRANGE_RANGE_CHECK_ENABLES 0x002D //距离测量检查使能
#define VL6180X_SYSRANGE_VHV_RECALIBRATE 0x002E //垂直扫描高压再校准
#define VL6180X_SYSRANGE_VHV_REPEAT_RATE 0x0031 //垂直扫描高压重复速率
//SysAls类别，用于配置和控制环境光强度测量相关的参数和操作。
#define VL6180X_SYSALS_START 0x0038 //环境光强度测量启动
#define VL6180X_SYSALS_THRESH_HIGH 0x003A //环境光强度上限阈值
#define VL6180X_SYSALS_THRESH_LOW 0x003C //下限阈值
#define VL6180X_SYSALS_INTERMEASUREMENT_PERIOD 0x003E //环境光强度测量之间的时间
#define VL6180X_SYSALS_ANALOGUE_GAIN 0x003F //环境光强度模拟增益
#define VL6180X_SYSALS_INTEGRATION_PERIOD 0x0040 //环境光强度积分周期
//这些寄存器存储了距离测量和环境光强度测量的实际结果，可以被读取并进一步处理或显示。
#define VL6180X_RESULT_RANGE_STATUS 0x004D //距离测量状态
```

```
#define VL6180X_RESULT_ALS_STATUS 0x004E //环境光强度测量状态
#define VL6180X_RESULT_INTERRUPT_STATUS_GPIO 0x004F //GPIO中断状态
#define VL6180X_RESULT_ALS_VAL 0x0050 //环境光强度测量值
#define VL6180X_RESULT_HISTORY_BUFFER 0x0052 // 历史数据缓存
#define VL6180X_RESULT_RANGE_VAL 0x0062 //距离测量值
#define VL6180X_RESULT_RANGE_RAW 0x0064 //距离测量原始值
#define VL6180X_RESULT_RANGE_RETURN_RATE 0x0066 //距离返回速率
#define VL6180X_RESULT_RANGE_REFERENCE_RATE 0x0068 //距离参考速率
#define VL6180X_RESULT_RANGE_RETURN_SIGNAL_COUNT 0x006C //距离返回信号计数
#define VL6180X_RESULT_RANGE_REFERENCE_SIGNAL_COUNT 0x0070 //距离参考信号计数
#define VL6180X_RESULT_RANGE_RETURN_AMB_COUNT 0x0074 //距离返回环境光强度计数
#define VL6180X_RESULT_RANGE_REFERENCE_AMB_COUNT 0x0078 //距离参考环境光强度计数
#define VL6180X_RESULT_RANGE_RETURN_CONV_TIME 0x007C //距离返回测量收敛时间
#define VL6180X_RESULT_RANGE_REFERENCE_CONV_TIME 0x0080 //距离参考测量收敛时间
#define VL6180X_READOUT_AVERAGING_SAMPLE_PERIOD 0x010A //读取平均采样周期
#define VL6180X_FIRMWARE_BOOTUP 0x0119 //固件启动状态
#define VL6180X_FIRMWARE_RESULT_SCALER 0x0120 //固件结果缩放系数
#define VL6180X_I2C_SLAVE_DEVICE_ADDRESS 0x0212 //I2C从设备地址
#define VL6180X_INTERLEAVED_MODE_ENABLE 0x02A3 //交错模式使能
enum vl6180x_als_gain {
GAIN_20 = 0, // Actual ALS Gain of 20
GAIN_10, // Actual ALS Gain of 10.32
GAIN_5, // Actual ALS Gain of 5.21
GAIN_2_5, // Actual ALS Gain of 2.60
GAIN_1_67, // Actual ALS Gain of 1.72
GAIN_1_25, // Actual ALS Gain of 1.28
GAIN_1 , // Actual ALS Gain of 1.01
GAIN_40, // Actual ALS Gain of 40
};
struct VL6180xIdentification
{
uint8_t idModel;
uint8_t idModelRevMajor;
uint8_t idModelRevMinor;
uint8_t idModuleRevMajor;
uint8_t idModuleRevMinor;
uint16_t idDate;
uint16_t idTime;
};
class VL6180x
{
public:
//使用默认地址初始化库
VL6180x(uint8_t address);
uint8_t VL6180xInit(void);
void VL6180xDefautSettings(void);
//获取当前测量到的范围距离值
uint8_t getDistance();
//获取当前测量到的环境光强度值
float getAmbientLight(vl6180x_als_gain VL6180X_ALS_GAIN);
//从传感器中读取标识信息，并将其存储在结构体变量中。
void getIdentification(struct VL6180xIdentification *temp);
//更改传感器的I2C地址
uint8_t changeAddress(uint8_t old_address, uint8_t new_address);
//设置范围距离中断阈值。
void VL6180xSetDistInt(uint8_t lowThres, uint8_t highThres);
//设置环境光强度中断阈值。
void VL6180xSetALSInt(vl6180x_als_gain VL6180X_ALS_GAIN, uint16_t lowThres, uint16_t highThres);
//连续获取范围距离值。
uint8_t getDistanceContinously();
//获取最近一次测量到的范围距离值。
```

```
uint8_t getLastDistanceFromHistory();
//连续获取环境光强度值。
float getAmbientLightContinously(vl6180x_als_gain VL6180X_ALS_GAIN);
//获取最近一次测量到的环境光强度值。
float getLastAmbientLightFromHistory(vl6180x_als_gain VL6180X_ALS_GAIN);
//清除传感器中断标志位。
void VL6180xClearInterrupt(void);
private:
//初始化时给定的地址。
int _i2caddress;
//从指定寄存器地址中读取一个字节的值。
uint8_t VL6180x_getRegister(uint16_t registerAddr);
//从指定寄存器地址中读取两个字节的值并将其存储为16位整数。
uint16_t VL6180x_getRegister16bit(uint16_t registerAddr);
//向指定寄存器地址中写入一个字节的值。
void VL6180x_setRegister(uint16_t registerAddr, uint8_t data);
//将给定的16位整数值写入到指定寄存器地址中。
void VL6180x_setRegister16bit(uint16_t registerAddr, uint16_t data);
};
#endif
//VL6180.cpp
#include "VL6180X.h"
VL6180x::VL6180x(uint8_t address)
// 初始化库{
Wire.begin(); // Arduino Wire 库初始化器
_i2caddress = address; //设置默认通讯地址
}
uint8_t VL6180x::VL6180xInit(void){
uint8_t data; //用于临时数据存储
data = VL6180x_getRegister(VL6180X_SYSTEM_FRESH_OUT_OF_RESET);
if(data != 1) return VL6180x_FAILURE_RESET;
VL6180x_setRegister(0x0207, 0x01);
VL6180x_setRegister(0x0208, 0x01);
VL6180x_setRegister(0x0096, 0x00);
VL6180x_setRegister(0x0097, 0xfd);
VL6180x_setRegister(0x00e3, 0x00);
VL6180x_setRegister(0x00e4, 0x04);
VL6180x_setRegister(0x00e5, 0x02);
VL6180x_setRegister(0x00e6, 0x01);
VL6180x_setRegister(0x00e7, 0x03);
VL6180x_setRegister(0x00f5, 0x02);
VL6180x_setRegister(0x00d9, 0x05);
VL6180x_setRegister(0x00db, 0xce);
VL6180x_setRegister(0x00dc, 0x03);
VL6180x_setRegister(0x00dd, 0xf8);
VL6180x_setRegister(0x009f, 0x00);
VL6180x_setRegister(0x00a3, 0x3c);
VL6180x_setRegister(0x00b7, 0x00);
VL6180x_setRegister(0x00bb, 0x3c);
VL6180x_setRegister(0x00b2, 0x09);
VL6180x_setRegister(0x00ca, 0x09);
VL6180x_setRegister(0x0198, 0x01);
VL6180x_setRegister(0x01b0, 0x17);
VL6180x_setRegister(0x01ad, 0x00);
VL6180x_setRegister(0x00ff, 0x05);
VL6180x_setRegister(0x0100, 0x05);
VL6180x_setRegister(0x0199, 0x05);
VL6180x_setRegister(0x01a6, 0x1b);
VL6180x_setRegister(0x01ac, 0x3e);
VL6180x_setRegister(0x01a7, 0x1f);
VL6180x_setRegister(0x0030, 0x00);
```

```
return 0;
}
//修改IIC地址
uint8_t VL6180x::changeAddress(uint8_t old_address, uint8_t new_address){
if( old_address == new_address) return old_address;
if( new_address > 127) return old_address;
VL6180x_setRegister(VL6180X_I2C_SLAVE_DEVICE_ADDRESS, new_address);
return VL6180x_getRegister(VL6180X_I2C_SLAVE_DEVICE_ADDRESS);
}
//VL6180x 传感器获取距离
uint8_t VL6180x::getDistance()
{
VL6180x_setRegister(VL6180X_SYSRANGE_START, 0x01); delay(10);
VL6180x_setRegister(VL6180X_SYSTEM_INTERRUPT_CLEAR, 0x07);
return VL6180x_getRegister(VL6180X_RESULT_RANGE_VAL);
//返回距离;
}
//VL6180x 传感器获取环境光强度
float VL6180x::getAmbientLight(vl6180x_als_gain VL6180X_ALS_GAIN){
//注意：高半字节应设置为 0x4，即对于 1.0 的 ALS 增益，写入 0x46
VL6180x_setRegister(VL6180X_SYSALS_ANALOGUE_GAIN, (0x40 | VL6180X_ALS_GAIN)); //设置ALS增益
//开始ALS测量
VL6180x_setRegister(VL6180X_SYSALS_START, 0x01);
delay(100);
VL6180x_setRegister(VL6180X_SYSTEM_INTERRUPT_CLEAR, 0x07);
//从传感器中检索原始 ALS 值
unsigned int alsRaw = VL6180x_getRegister16bit(VL6180X_RESULT_ALS_VAL);
//获取积分周期进行计算，我们每次都这样做，以防有人更改它。
unsigned int alsIntegrationPeriodRaw = VL6180x_getRegister16bit(VL6180X_SYSALS_INTEGRATION_PERIOD);
float alsIntegrationPeriod = 100.0 / alsIntegrationPeriodRaw ;
//从 Appnotes 计算实际 LUX
float alsGain = 0.0;
switch (VL6180X_ALS_GAIN){
case GAIN_20: alsGain = 20.0; break;
case GAIN_10: alsGain = 10.32; break;
case GAIN_5: alsGain = 5.21; break;
case GAIN_2_5: alsGain = 2.60; break;
case GAIN_1_67: alsGain = 1.72; break;
case GAIN_1_25: alsGain = 1.28; break;
case GAIN_1: alsGain = 1.01; break;
case GAIN_40: alsGain = 40.0; break;
}
float alsCalculated = (float)0.32 * ((float)alsRaw / alsGain) * alsIntegrationPeriod;
return alsCalculated;
}
// --- VL6180x 传感器读取寄存器值的私有函数 --- //
uint8_t VL6180x::VL6180x_getRegister(uint16_t registerAddr)
{
uint8_t data;
Wire.beginTransmission( _i2caddress ); // 启动 I2C
Wire.write((registerAddr >> 8) & 0xFF); //写入 I2C 缓冲区中。
Wire.write(registerAddr & 0xFF); //LSB
Wire.endTransmission(false); //结束 I2C 传输但不释放总线，以便后续读取数据。
Wire.requestFrom( _i2caddress , 1);//读取一个字节的数据。
data = Wire.read(); //从 I2C 缓冲区中读取刚刚请求的数据，并将其存储在变量 data 中。
return data;
}
//从传感器设备中读取两个字节的数据，并将其合并成一个 16 位的数据，返回该数据作为函数的输出。
uint16_t VL6180x::VL6180x_getRegister16bit(uint16_t registerAddr)
{
uint8_t data_low;
```

```
uint8_t data_high;
uint16_t data;
Wire.beginTransmission( _i2caddress );
Wire.write((registerAddr >> 8) & 0xFF);
Wire.write(registerAddr & 0xFF);
Wire.endTransmission(false);
Wire.requestFrom( _i2caddress, 2);
data_high = Wire.read();
data_low = Wire.read();
data = (data_high << 8)|data_low;//合并成一个 16 位的数据，存储在变量 data 中。
```

6. 第4章系统测试与性能评估_第2部分　　总字数：10035

相似文献列表

去除本人文献复制比：0%(0)　去除引用文献复制比：0%(0)　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容

```
return data;
}
//将一个字节的数据写入传感器设备的寄存器中。
void VL6180x::VL6180x_setRegister(uint16_t registerAddr, uint8_t data)
{
Wire.beginTransmission( _i2caddress );
Wire.write((registerAddr >> 8) & 0xFF);
Wire.write(registerAddr & 0xFF);
Wire.write(data);
Wire.endTransmission();
}
//将一个 16 位的数据分为高低两个字节，并将它们依次写入传感器设备的寄存器中。
void VL6180x::VL6180x_setRegister16bit(uint16_t registerAddr, uint16_t data)
{
Wire.beginTransmission( _i2caddress );
Wire.write((registerAddr >> 8) & 0xFF);
Wire.write(registerAddr & 0xFF);
uint8_t temp;
temp = (data >> 8) & 0xff;
Wire.write(temp);
temp = data & 0xff;
Wire.write(temp);
Wire.endTransmission();
}
//设置距离阈值中断
void VL6180x::VL6180xSetDistInt(uint8_t lowThres, uint8_t highThres){
VL6180x_setRegister(VL6180X_SYSTEM_INTERRUPT_CONFIG_GPIO, 0x03);。
VL6180x_setRegister(VL6180X_SYSRANGE_THRESH_LOW, lowThres); //设置距离低阈值
VL6180x_setRegister(VL6180X_SYSRANGE_THRESH_HIGH, highThres);//设置距离高阈值
}
/* VL6180x 传感器设置环境光强度阈值中断的函数实现，该函数接受三个参数，VL6180X_ALS_GAIN 表示 ALS 增益
, lowThres 表示环境光强度低阈值，highThres 表示环境光强度高阈值。 */
void VL6180x::VL6180xSetALSInt(vl6180x_als_gain VL6180X_ALS_GAIN, uint16_t lowThres, uint16_t highThres){
unsigned int alsIntegrationPeriodRaw = VL6180x_getRegister16bit(VL6180X_SYSALS_INTEGRATION_PERIOD);
float alsIntegrationPeriod = 100.0 / alsIntegrationPeriodRaw ;
float alsGain = 0.0;
switch (VL6180X_ALS_GAIN){
case GAIN_20: alsGain = 20.0; break;
case GAIN_10: alsGain = 10.32; break;
case GAIN_5: alsGain = 5.21; break;
case GAIN_2_5: alsGain = 2.60; break;
```

```
case GAIN_1_67: alsGain = 1.72; break;
case GAIN_1_25: alsGain = 1.28; break;
case GAIN_1: alsGain = 1.01; break;
case GAIN_40: alsGain = 40.0; break;
}
uint16_t alsLowThresRaw = (uint16_t)((float)lowThres * alsGain) / (alsIntegrationPeriod *0.32);
uint16_t alsHighThresRaw = (uint16_t)((float)highThres * alsGain) / (alsIntegrationPeriod *0.32);
Serial.println(alsLowThresRaw);
Serial.println(alsHighThresRaw);
VL6180x_setRegister(VL6180X_SYSTEM_INTERRUPT_CONFIG_GPIO, 0x18);
VL6180x_setRegister16bit(VL6180X_SYSALS_THRESH_LOW, alsLowThresRaw);
VL6180x_setRegister16bit(VL6180X_SYSALS_THRESH_HIGH, alsHighThresRaw);
}
/* VL6180x 传感器连续获取距离值的函数实现。 */
uint8_t VL6180x::getDistanceContinously()
{
VL6180x_setRegister(VL6180X_SYSRANGE_START, 0x01); delay(10);
VL6180x_setRegister(VL6180X_SYSRANGE_START, 0x03);
delay(10);
VL6180x_setRegister(VL6180X_SYSTEM_INTERRUPT_CLEAR, 0x07);
VL6180x_setRegister(VL6180X_SYSTEM_HISTORY_CTRL, 0x01); // 启用历史缓冲区
return VL6180x_getRegister(VL6180X_RESULT_RANGE_VAL);
// return distance;
}
/* VL6180x 传感器从历史缓冲区中读取最后一个距离值的函数实现。 */
uint8_t VL6180x::getLastDistanceFromHistory()
{
uint8_t lastDist;
uint16_t histVal;
histVal = VL6180x_getRegister16bit(VL6180X_RESULT_HISTORY_BUFFER);
lastDist = histVal>>8; // 16 位历史缓冲区 0 的高位字节包含最后一个范围
return lastDist;
}
/* VL6180x 传感器连续获取环境光强度值的函数实现。 */
float VL6180x::getAmbientLightContinously(vl6180x_als_gain VL6180X_ALS_GAIN)
{
VL6180x_setRegister(VL6180X_SYSALS_ANALOGUE_GAIN, (0x40 | VL6180X_ALS_GAIN)); //设置ALS增益
//开始ALS测量
VL6180x_setRegister(VL6180X_SYSALS_START, 0x01); delay(100);
VL6180x_setRegister(VL6180X_SYSALS_START, 0x03);
delay(100);
VL6180x_setRegister(VL6180X_SYSTEM_INTERRUPT_CLEAR, 0x07);
VL6180x_setRegister(VL6180X_SYSTEM_HISTORY_CTRL, 0x03); // 为 ALS 启用历史缓冲区
//从传感器中检索原始 ALS 值
unsigned int alsRaw = VL6180x_getRegister16bit(VL6180X_RESULT_ALS_VAL);
//获取积分周期进行计算，我们每次都这样做，以防有人更改它。
unsigned int alsIntegrationPeriodRaw = VL6180x_getRegister16bit(VL6180X_SYSALS_INTEGRATION_PERIOD);
float alsIntegrationPeriod = 100.0 / alsIntegrationPeriodRaw ;
float alsGain = 0.0;
switch (VL6180X_ALS_GAIN){
case GAIN_20: alsGain = 20.0; break;
case GAIN_10: alsGain = 10.32; break;
case GAIN_5: alsGain = 5.21; break;
case GAIN_2_5: alsGain = 2.60; break;
case GAIN_1_67: alsGain = 1.72; break;
case GAIN_1_25: alsGain = 1.28; break;
case GAIN_1: alsGain = 1.01; break;
case GAIN_40: alsGain = 40.0; break;
}
float alsCalculated = (float)0.32 * ((float)alsRaw / alsGain) * alsIntegrationPeriod;
return alsCalculated;
```

```
}
/* VL6180x 传感器从历史缓冲区中读取最后一个环境光强度值的函数实现。*/
float VL6180x::getLastAmbientLightFromHistory(vl6180x_als_gain VL6180X_ALS_GAIN)
{
//从历史缓冲区中获取原始ALS
unsigned int alsRaw = VL6180x_getRegister16bit(VL6180X_RESULT_HISTORY_BUFFER);
//获取积分周期进行计算，我们每次都这样做，以防有人更改它。
unsigned int alsIntegrationPeriodRaw = VL6180x_getRegister16bit(VL6180X_SYSALS_INTEGRATION_PERIOD);
float alsIntegrationPeriod = 100.0 / alsIntegrationPeriodRaw ;
float alsGain = 0.0;
switch (VL6180X_ALS_GAIN){
case GAIN_20: alsGain = 20.0; break;
case GAIN_10: alsGain = 10.32; break;
case GAIN_5: alsGain = 5.21; break;
case GAIN_2_5: alsGain = 2.60; break;
case GAIN_1_67: alsGain = 1.72; break;
case GAIN_1_25: alsGain = 1.28; break;
case GAIN_1: alsGain = 1.01; break;
case GAIN_40: alsGain = 40.0; break;
}
float alsCalculated = (float)0.32 * ((float)alsRaw / alsGain) * alsIntegrationPeriod;
return alsCalculated;
}
//VL6180x 传感器清除中断标志位的函数实现。该函数调用 VL6180x_setRegister() 函数，向寄存器
VL6180X_SYSTEM_INTERRUPT_CLEAR 写入值 0x07，以清除所有中断标志位。
void VL6180x::VL6180xClearInterrupt(void){
VL6180x_setRegister(VL6180X_SYSTEM_INTERRUPT_CLEAR, 0x07);
}
//main.cpp
#include <Arduino.h>
#include <Wire.h>
#include <VL6180X.h>
#include <U8g2lib.h>
#define VL6180X_ADDRESS 0x29
#define OLED_ADDRESS 0x3C
int min_distance = 15;
int max_distance = 150;
VL6180xIdentification identification;
VL6180x sensor(VL6180X_ADDRESS);
U8G2_SSD1306_128X32_UNIVISION_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE, /* clock=*/ SCL, /* data=*/
SDA);
void printIdentification(struct VL6180xIdentification *temp){
Serial.print("设备 ID = ");
Serial.println(temp->idModel);
Serial.print("设备 Rev = ");
Serial.print(temp->idModelRevMajor);
Serial.print(".");
Serial.println(temp->idModelRevMinor);
Serial.print("修订 Rev = ");
Serial.print(temp->idModuleRevMajor);
Serial.print(".");
Serial.println(temp->idModuleRevMinor);
Serial.print("生产日期 = ");
Serial.print((temp->idDate >> 3) & 0x001F);
Serial.print("/");
Serial.print((temp->idDate >> 8) & 0x000F);
Serial.print("/1");
Serial.print((temp->idDate >> 12) & 0x000F);
Serial.print(" 阶段: ");
Serial.println(temp->idDate & 0x0007);
Serial.print("生产时间 (s)= ");
```

```
Serial.println(temp->idTime * 2);
Serial.println();
Serial.println();
}
void delay_ms(int ms) {
unsigned long startMillis = millis();
while (millis() - startMillis < ms);
}
void setup() {
Serial.begin(9600);
Wire.begin();
u8g2.begin();
u8g2.enableUTF8Print();
delay(1000);
sensor.getIdentification(&identification);
printIdentification(&identification);
if(sensor.VL6180xInit() != 0){
Serial.println("VL6180初始化失败");
};
sensor.VL6180xDefautSettings();
delay(1000);
}
void loop() {
static unsigned long last_50ms = 0;
static unsigned long last_100ms = 0;
unsigned long currentMillis = millis();
u8g2.clearBuffer();
u8g2.setFont(u8g2_font_wqy14_t_gb2312b);
u8g2.setFontDirection(0);
int distance = sensor.getDistance();
if (distance >= min_distance && distance <= max_distance) {
u8g2.setCursor(0, 15);
u8g2.print("光照：");
u8g2.print(sensor.getAmbientLight(GAIN_1)*1000);
u8g2.print(" Lux");
u8g2.setCursor(0, 30);
u8g2.print("距离：");
u8g2.print(distance);
u8g2.print(" mm");
} else if (distance < min_distance) {
u8g2.setCursor(0, 15);
u8g2.print("距离过近");
} else {
u8g2.setCursor(0, 15);
u8g2.print("距离过远");
}
u8g2.sendBuffer();
Serial.print("环境光级别 (Lux) = ");
Serial.println(sensor.getAmbientLight(GAIN_1)*1000);
Serial.print("测量距离 (mm) = ");
Serial.println(sensor.getDistance());
/* if (currentMillis - last_50ms >= 50) {
last_50ms = currentMillis;
Serial.println("Delay 50ms");
delay_ms(50);
} */
};
####jingdu.py
#用于检测串口并记录输出，并生成散点图
#使用了matplotlib、numpy、pyserial等python库，需要加载usbserial模块 `sudo modprobe usbserial`
#PySerial库来读取串口数据，matplotlib库来绘制散点图
```

```
import serial
import matplotlib.pyplot as plt
import warnings
import numpy as np
# 设置串口参数
ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)
# 定义空列表存储距离数据
distances = []
# 读取100次距离数据
for i in range(100):
# 从串口读取一行数据
line = ser.readline().decode().strip()
# 检查是否为有效数据
if line.startswith("测量距离 (mm) = "):
# 解析距离数值并添加到列表中
distance = float(line.replace("测量距离 (mm) = ", ""))
distances.append(distance)
# 关闭串口连接
ser.close()
# 计算平均值和标准差
mean = np.mean(distances)
std = np.std(distances)
# 设置字体MiSans
plt.rcParams['font.sans-serif'] = ['MiSans']
# 调整y轴范围
plt.ylim(min(distances)-10, max(distances)+10)
# 绘制散点图，并设置点的大小为5
plt.scatter(range(1, len(distances)+1), distances, s=5)
# 绘制平均值和标准差的虚线
plt.axhline(y=mean, color='green', linestyle='--', label='平均值')
plt.axhline(y=mean+std, color='red', linestyle='--', label='标准差')
plt.axhline(y=mean-std, color='red', linestyle='--')
# 添加图例
plt.legend()
plt.title("100次距离测量")
plt.xlabel("测量次数")
pl
```

位号	名称	制造商	封装规格
C1,C2,C3	1uF电容	SAMSUNG(三星)	C0603
C4	100nF电容	SAMSUNG(三星)	C0603
C5	100nF电容	SAMSUNG(三星)	C0603
CN1	PH2.0针座卧贴 2P	HDGC(华德共创)	CONN-SMD_2P
D1	肖特基二极管	TWGMC(台湾迪嘉)	SOD-123
H1	排针4P	XFCN(兴飞)	HDR-TH_4P-P2.54
H2	排针7P	XFCN(兴飞)	HDR-TH_7P-P2.54
LDO1	3.3V LDO	MICROCHIP(美国微芯)	SOT-23-5
LED1,LED2	发光二极管	EVERLIGHT(亿光)	LED0603-RD
Q1	场效应管	TWGMC(台湾迪嘉)	SOT-23-3
Q2,Q3	S8050三极管	WPMtek(维攀微)	SOT-23-3
R1,R2,R3,R4,R9	10kΩ	RALEC(旺诠)	R0603
R5	3kΩ	UNI-ROYAL(厚声)	R0603
R6,R7,R8	10kΩ	PANASONIC(松下)	R0603
SW1	滑动开关	XKB Connectivity(中国星坤)	SW-SMD
SW2,SW3	轻触开关	SHOU HAN(首韩)	SW-SMD
TP4054	TP4054	TOPPOWER(南京拓微)	SOT-23
U4	CP2102	SILICON LABS(芯科)	WQFN-28
U5	ESP32-PICO-D4	ESPRESSIF(乐鑫)	LGA-48_L7.0
USB1	TYPE-C母座卧贴	kinghelm(金航标)	USB-C-SMD-16P

```
t.ylabel("距离 (mm)")
# 保存散点图为png图片
plt.savefig('distance_scatter.png', dpi=300)
plt.show()
```

附录三

表1 主控BOM清单
位号名称制造商封装规格
C1,C2,C3 1uF电容 SAMSUNG(三星) C0603
C4 100nF电容 SAMSUNG(三星) C0603
C5 100nF电容 SAMSUNG(三星) C0603
CN1 PH2.0针座卧贴 2P HDGC(华德共创) CONN-SMD_2P
D1 肖特基二极管 TWGMC(台湾迪嘉) SOD-123
H1 排针4P XFCN(兴飞) HDR-TH_4P-P2.54
H2 排针7P XFCN(兴飞) HDR-TH_7P-P2.54
LDO1 3.3V LDO MICROCHIP(美国微芯) SOT-23-5
LED1,LED2 发光二极管 EVERLIGHT(亿光) LED0603-RD
Q1 场效应管 TWGMC(台湾迪嘉) SOT-23-3
Q2,Q3 S8050三极管 WPMtek(维攀微) SOT-23-3
R1,R2,R3,R4,R9 10kΩ RALEC(旺诠) R0603
R5 3kΩ UNI-ROYAL(厚声) R0603
R6,R7,R8 10kΩ PANASONIC(松下) R0603
SW1 滑动开关 XKB Connectivity(中国星坤) SW-SMD
SW2,SW3 轻触开关 SHOU HAN(首韩) SW-SMD
TP4054 TP4054 TOPPOWER(南京拓微) SOT-23
U4 CP2102 SILICON LABS(芯科)

位号	名称	制造商	封装规格
C1,C2	1uF	SAMSUNG(三星)	C0603
C3,C4	100nF	SAMSUNG(三星)	C0603
C5	4.7uF	SAMSUNG(三星)	C0603
H1	排针7P	XFCN(兴飞)	HDR-TH_7P-P2.54
Q1,Q2	AO3401三极管	TWGMC(台湾迪嘉)	SOT-23-3
R1,R2,R3,R4	10kΩ	PANASONIC(松下)	R0603
R5,R6	47kΩ	PANASONIC(松下)	R0805
R7,R8	1kΩ	PANASONIC(松下)	R0805
U1	VL6180	ST(意法半导体)	LGA-12
U2	2.8V LDO	TI(德州仪器)	TSOT-23

WQFN-28
U5 ESP32-PICO-D4 ESPRESSIF(乐鑫) LGA-48_L7.0
USB1 TYPE-C母座卧贴 kinghelm(金航标) USB-C-SMD-16P
表2 驱动板BOM清单
位号名称制造商封装规格
C1,C2 1uF SAMSUNG(三星) C0603
C3,C4 100nF SAMSUNG(三星) C0603
C5 4.7uF SAMSUNG(三星) C0603
H1 排针7P XFCN(兴飞) HDR-TH_7P-P2.54
Q1,Q2 AO3401三极管 TWGMC(台湾迪嘉) SOT-23-3
R1,R2,R3,R4 10kΩ PANASONIC(松下) R0603
R5,R6 47kΩ PANASONIC(松下) R0805
R7,R8 1kΩ PANASONIC(松下) R0805
U1 VL6180 ST(意法半导体) LGA-12
U2 2.8V LDO TI(德州仪器) TSOT-23