编 号

本科生毕业设计（论文）

题目：用于大飞机装配的基于 MEMS 传感器平面装配监测系统

物联网工程 学 院 物联网工程 专 业

学　　号 1030613221

学生姓名 明　坤

指导教师 王　呈　讲　师

时广轶　副教授

二〇一七年六月

设计总说明

大飞机一般是指起飞总重超过100吨的运输类飞机，包括军用大型运输机和民用大型运输机，也包括一次航程达到3000公里的军用飞机或载员达到100人以上的民用客机。大飞机装配过程中，需要用支架固定各段机身，操作人员在支架旁边的型架上进行装配，此时位于飞机两侧的型架平面要求严格对准，以保证飞机固件的精确装配。但大型飞机在使用型架进行机身固件装配时，由于型架两侧轨道受力不均匀，会导致地面沉降程度有差异，使两个装配面发生偏移，互相不平行。因此，需要一套能够辅助操作人员测量型架姿态并进行相应调整的监测系统。

现有的两种型架姿态监测方案分别采用激光测试仪和静力水准仪，受限于其自身的原理，这两种方法都不便于在飞机的装配过程中进行测量。为了解决飞机装配型架的姿态监测问题，本文设计并实现了一种新型型架姿态监测系统。系统在满足高精度姿态监测的前提下，可以在飞机装配过程中实时监测型架姿态，操作人员可以通过7寸控制屏看到实时刷新的型架姿态。当型架偏离标准姿态时，该系统可以辅助操作人员调整型架姿态回到初始状态，甚至在姿态变化超过阈值时发出报警信号，通知操作人员及时停止作业。本设计主要内容包括：

（1）型架单点姿态采集设计。

采集系统包括高精度、低漂移、低噪声MEMS加速度计ADXL355和32位微控制器STM32。利用MEMS加速度计测量重力加速度在器件各个轴上的分量，STM32读取数据并计算器件各轴与水平面的夹角，从而计算出型架的姿态。

（2）姿态信息传输与可视化设计。

传输系统和可视化系统包括DTU和7寸工控组态屏。型架监测系统中，所有设备均连入RS485总线，基于modbus协议通信。7寸组态屏作为显示和控制终端，用于监测型架姿态和对倾角传感器相关参数进行设置；通过无线数据传输单元(下文简称DTU)完成姿态数据采集并上传至服务器，DTU被设置为每隔1分钟上传一次角度数据到服务器，操作人员可以通过浏览器远程查看实时和历史数据。

（3）型架多点姿态监测设计。

飞机装配使用的型架基座较大(3m*4m的平面)，采用在多个测量点分别布置传感器的方式，解决了仅测量各角顶点位置无法完全得到型架整体的真实姿态的困难。采用激光测绘仪辅助确定型架的标准姿态，并结合MEMS传感器提供系统一个标准值，解决了MEMS倾角传感器系统由于存在初始误差导致输出值与被测物的角度存在偏差的问题。

经过实地测试，依靠本监测系统的数据，修复型架姿态的精度大约在0.12mm左右，可以满足飞机装配的需求。该系统可以在装配工作期间实现对装配面的水平与倾斜监测，弥补了激光测绘仪和静力水准仪校准不便于在装配流程中实时监测的缺点，有望提高大飞机的装配效率，缩短大飞机的生产周期。

关键词：大飞机装配；MEMS 传感器；装配监测

ABSTRACT

Large aircraft generally refers to the transport aircraft that takeoff weight more than 100 tons , which include the military large transport aircraft that voyage more than 3000km,the civilian large transport aircraft that can carry more than 100 passengers. It's necessary to fixed sections of the fuselage by using the fixture in large aircraft assembly process.When the operator is assembling next to the type of rack in the fixture,it's vital to make both sides of the assembly plane rigorous alignment to ensure the precise assembly of the aircraft firmware. However,

There might produce offset on attitude of fixture when aircraft assembling,because the force applied on two sides of the track is not equal,which lead to two fixture not parallel to each other.Therefore, there is a need for a monitoring system that can assist the operator in measuring the fixture attitude and making corresponding adjustments.

The existing two types of fixture attitude monitoring program are using laser tester and static level.Because of its own principles, these two methods are not easy to use in the assembly process. In order to solve the attitude problem of aircraft assembly fixture, this paper designs and implements a new type of fixture attitude monitoring system. The system can monitor the attitude of the fixture in the process of aircraft assembly in the high precision. The operator can watch the real-time refreshing attitude of fixture through the 7-inch control screen. When the fixture deviates from the standard attitude, the system can assist the operator to adjust the fixture attitude back to the initial state, and send alarm signal to notify the operator to stop the operation in time when the attitude change exceeds the threshold. The main contents of this design include:

(1)The design of single point attitude acquisition of fixture.

The acquisition system includes high precision, low drift, low noise MEMS accelerometer ADXL355 and 32-bit microcontrollers STM32. Using the MEMS accelerometer to measure the component of the gravitational acceleration on each axis of the device, STM32 reads the data and calculates the angle between the axes of the device and the horizontal plane to calculate the attitude of the fixture.

(2) design of attitude transmission and visual about information.

The transmission system and the visualization system include DTU and 7-inch industrial control screen. All equipment of fixture attitude monitoring system are connected to the RS485 bus, and communicating by using modbus protocol. The 7-inch screen is used as the display and control terminal for monitoring the fixture attitude and setting the parameters of the tilt sensor. The attitude data is collected and uploaded to the server through the wireless data transmission unit (hereinafter referred to as DTU), and the DTU is set to upload the angle data to the server once a minute.It allows that operators can remotely view the real-time and historical data from the PC-side browser.

(3) design of multi-point attitude monitoring.

This paper puts forward a method to solve the problem that the measurement of the vertices of the corners can not be completely get real attitude of fixture by using multi-point attitude monitoring.The standard position of the model is determined by the laser surveying instrument, and then,the standard value of the system is provided with the MEMS sensor,which solves the problem that the deviation of the output value from the angle of the measured object due to the existence of some initial error.

The accuracy of the monitoring system is about 0.12mm,which is concluded after field test.It can meet the needs of aircraft assembly. The system can finish the level and tilt monitoring of the fixture attitude during assembly work, make up the shortcomings of laser mapping and static leveling and facilitate real-time monitoring in the assembly process. It is expected to improve the assembly efficiency of large aircraft and shorten of the production cycle.

Keywords: Large aircraft assembly; MEMS sensor; assembly monitoring

目 录

第 1 章 绪论

1.1 研究背景和意义

本文所述型架是专为保证大飞机装配质量而研发的辅助设备，用于在飞机机身装配过程中牢靠固定机身和辅助操作人员进行装配固件定位，从而保证飞机零部件的装配精度。用于大飞机装配的型架A、B如图1-1所示，位于图片中部带有弧形的枕状物体称为支架；位于图片的左上方和右下方的物体称为型架，每个型架都由其下方的基座和上方的操作平台构成，其中每个基座的四个顶点都有一个可调节高度的支撑柱。

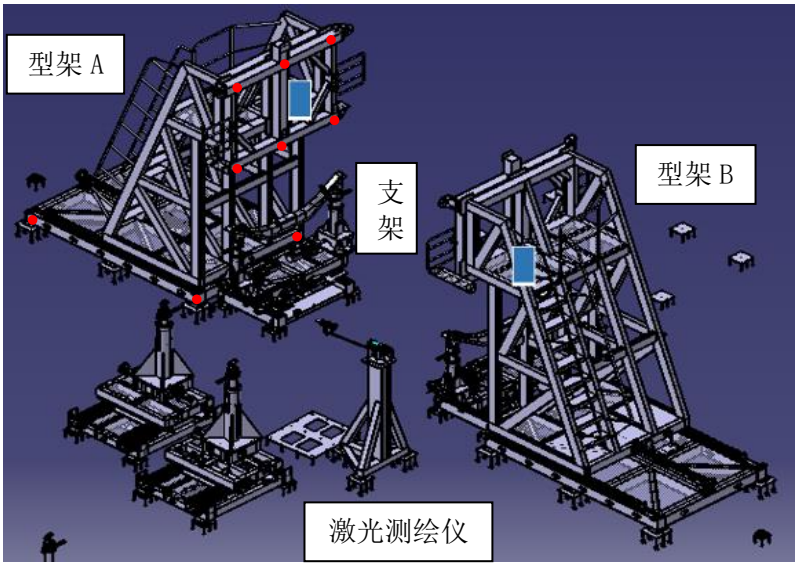


图 1-1 大飞机装配型架示意图

在大飞机进行装配时，会先将一段机身的左右两端放在支架上，然后调整型架A、B使得操作平台的前面板对准飞机机身的内部空间，之后再由操作员登上操作平台完成后续的零部件装配工作。型架是为装配流程服务的，其各部分的结构设计都与飞机结构紧密相关，例如，在装配大飞机机身的过程中需要进行铆钉装配，此时，型架上的卡板和定位器可以帮助操作人员确定安装位置。当装配完成后，再将飞机机身运离型架，转移到其他车间进行后续的机身拼接工序。

整个机身的装配过程可能持续数周，在此期间需要两个型架以高精度保持原有姿态。如果在这段长周期的装配过程中，型架自身发成了偏移，那么势必会使装配误差增大. 降低装配质量。然而，在飞机的装配施工周期中，工人的施工操作、型架自重导致的地面沉降，以及型架自身的结构微变形都可能造成型架姿态的变化。如果不能及时发现型架姿态的变化并修正，就会影响飞机机身装配的精度，严重时可能需要停工检修，甚至返工重新装配。

生产过程中往往因为经常需要搬运仪器检查型架姿态或者型架变化长时间未发现导致返工而耽误装配时间。因此，为了保证飞机装配效率，缩短装配周期，需要找到一种能够实时监测型架姿态变化的高精度传感器系统，来保证型架姿态的稳定，系统不仅需要能够实时监测，还需要在型架姿态变化超出可接受范围时发出报警信号，便于及时修正型架姿态，保证飞机装配精度。

为此，本文提出了一种基于MEMS加速度计系统的型架姿态监测解决方案，可以在保证高精度的条件下，实时测量监测型架姿态，同时又具有安装方便、测量修正周期短、价格适中等优点，有望大大降低我国飞机装配流程中消耗在保持型架姿态上的时间，从而缩短大飞机的装配周期。

1.2 现有的型架姿态监测技术解决方案

1.2.1 激光测绘

激光测绘是传统型架状态监测较常用的方案[1][2][3]，一般使用如图1-2所示的激光测绘仪，工业应用中，普遍采用瑞士徕卡公司的各类激光仪器。

图 1-2 激光测绘仪

图 1-3 激光测绘仪的使用

激光测绘仪的使用过程如图 1-3 所示，通过在特定位置提前放置反射器，再开启激光测绘仪用激光手段测量这些提前标记好的特殊点，并输出这些点相对于仪器的三维相对坐标[4]。以此为原理，提前在图 1-1 中的红点位置做好标记，定期用激光测绘仪测出各个点的三维坐标，就可以利用这组数据将型架调整到标准姿态。优点是精度高(μm 级)，测量误差小。但缺点是在装流程中测量不方便，且价格昂贵。

使用激光测绘仪，可以在飞机装配前把型架调整到标准位置。但是在飞机装配流程中，机身骨架位于两型架之间，会遮挡住标准点，此时无法用激光测绘仪测量。如果要使用激光测绘仪，还需要把装配中的飞机机身运离型架，这样就导致装配效率大大降低。如果经常测量，就可能耽误工期，但如果测量间隔的时间较长，就不能及时发现型架的姿态变化。

1.2.2 静力水准仪

静力水准仪是测量两点间或多点间相对高程变化的精密仪器[5]。主要用于大型储罐、大坝、核电站、高层建筑、基坑、隧道、桥梁、地铁等垂直位移和倾斜的监测[6] [7]。

静力水准仪是根据静止液面在重力作用下保持同一水平的特点，亦即与大地水准面相平行的水准面的原理，测量各点间的高程差，从而直接得到垂直形变或间接求得倾斜角度的变化。各测点的结构如图 1-4 所示。

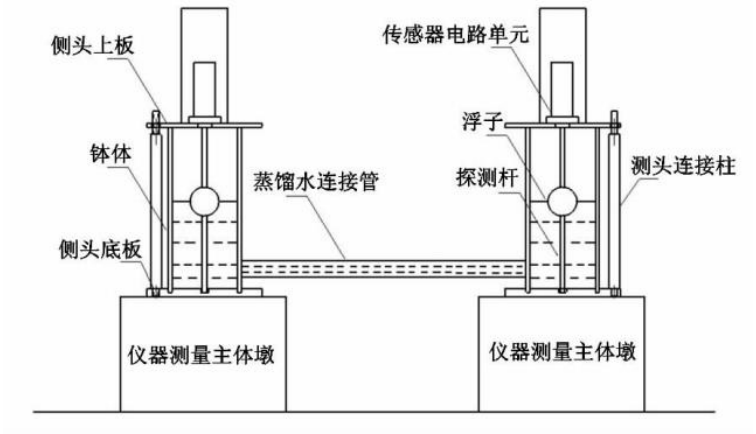


图 1-4 静力水准仪测点结构

静力水准系统至少由两个观测点组成，每个观测点安装一套静力水准仪。部署时，首先需要施工构建一个稳定的基准点平面，安装在这个平面上的静力水准仪作为基准点，随后再在各个测点安装静力水准仪，通过测量各个静力水准仪的液面高度，可以计算出每个测点相对于基准点的高程差。静力水准仪的贮液容器相互用通液管完全连通，贮液容器内注入液体，当液体液面完全静止后系统中所有连通容器内的液面应同在一个大地水准面上，此时每一容器的液位由传感器测出，即初始液位值分别为：H_1、H_2、H_3、H_4，如图示 1-5。

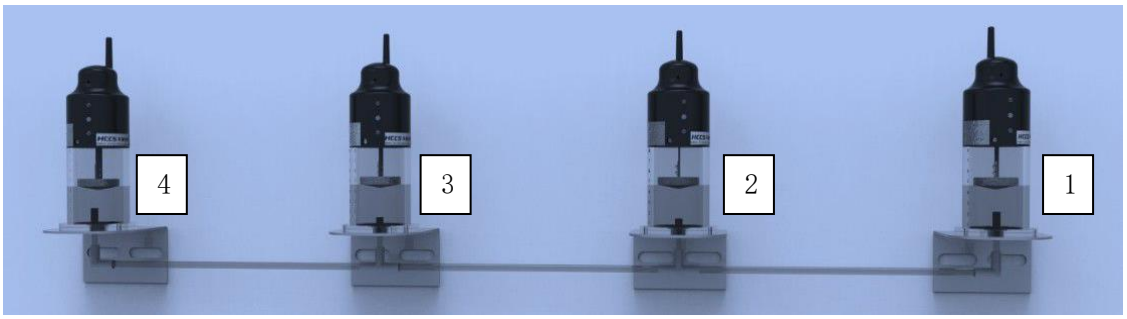


图 1-5 处于同一平面的静力水准仪

假设被测物体测点 1 作为基准点，测点 2 的地基下沉，测点 3 的地基上升，测点 4 的地基不变，当系统内液面达到平衡静止后形成新的水准面，则各测点连通容器内的新液位值分别为：H_1'、H_2'、H_3'、H_4'，如图示 1-6。

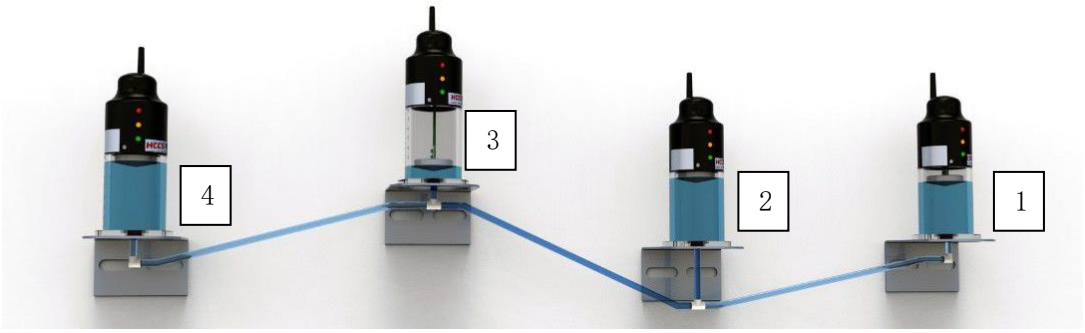


图 1-6 处于不同平面的静力水准仪

系统各测点的液位由静力水准仪传感器测得，各测点液位变化量分别计算为：$\Delta h_1 = H_1' - H_1$、$\Delta h_2 = H_2' - H_2$、$\Delta h_3 = H_3' - H_3$、$\Delta h_4 = H_4' - H_4$。其中计算结果：Δh_i 为正值表示该测点贮液容器内的液面升高，Δh_i 为负值表示该测点贮液容器内的液面降低。

如果选定测点 1 为基准点，则其它各测点相对基准点的垂直位移(沉降量)为：$\Delta H_2 = \Delta h_1 - \Delta h_2$、$\Delta H_3 = \Delta h_1 - \Delta h_3$、$\Delta H_4 = \Delta h_1 - \Delta h_4$。其中计算结果：$\Delta h_i$ 为正值表示该测点地基抬高，Δh_i 为负值表示该测点地基沉降。如果知道两测点间的水平距离 L，则两测点间相对倾斜的变化也可算得。

由以上原理，可以提前构筑一个稳定的地基平面，用于放置基准点，随后再在型架的各个支点放置静力水准仪，测量型架各个支点的沉降程度，从而指导工人进行调整，保证型架的稳定。但是由于管道中的液体存在粘滞力，需要较长的时间才能使液面保持较稳定的状态。而在飞机装配过程中，由于工人上下操作，零配件搬运等原因，会使管道中的液

面长期处于非静止状态，导致误差较大，不能实时动态监测型架姿态[8]。

1.2.4 型架姿态监测技术的应用现状

国内的型架监测的主要有两种方案，一是使用激光测绘仪测算出型架几个关键点的相对坐标，随后调整型架支撑柱使其达到标准姿态，二是安装静力水准仪直接测量型架各个支撑柱的沉降值，参考沉降值调整支撑柱达到调整型架的目的。

但这两种方法受限于其自身原理，制约了大飞机的生产效率。激光测绘仪虽然精度高，但在测量时需要保证被测点与仪器之间无遮挡物，此方法可以在飞机装配开始前把型架调整到标准位置，但当飞机装配开始后，飞机机身的一部分被固定在型架上，机身会遮挡住被测点，此时如果要使用激光测绘仪，就需要把飞机机身运离型架。飞机装配过程要求型架一直处于标准状态，按照标准需要经常测量型架姿态，在装配过程中使用激光测绘仪往往费时费力，影响工期。另一个方案中的静力水准仪精度也满足要求，但使用静力水准仪的先决条件是需要一个稳定的地基平面，用于放置基准点，否则会导致整体的测量结果不准确，因此需要在型架周围提前施工，改造装配场地，人工构建一个受沉降影响尽量小的地基。另外由于静力水准仪中灌注的液体存在粘滞力，如果它受到扰动,则需要较长的时间等待液面稳定后，才能输出正确的数值。

国内的监测方案虽然能保证精度，但也遇到了测绘修正周期长、国内相关设备缺乏、国外设备价格昂贵等问题，首选方案中高精度的激光测绘仪需要依赖国外进口，价格昂贵。

国外的技术厂商如波音、空客等已有其自己的相关解决方案，部分装配操作已经引入自动化过程，但受于限制，不可能转让给我国企业，因此，我国亟需研究具有自主知识产权的型架姿态监测解决方案。

1.3 章节安排

本文设计并实现了基于 MEMS 传感器的平面装配监测系统,可用于监测用于大飞机型架姿态，提高装配效率。MEMS 加速度计和 32 位微处理器 STM32 构成该系统的倾角传感器部分，并与 DTU 和组态屏共同连入 RS485 总线，和远端的服务器一起构成型架姿态监测系统。操作人员可以通过组态屏查看实时参数，历史数据同时保存在组态屏和服务器中。经试验，该系统具有良好的精度和操作性，能够满足飞机型架的监测需要。本文采用如下的结构，来完成对系统设计方法的论述和实验结果的分析。

第一章绪论中，本文阐述了本课题的研究背景和研究意义，并介绍了现有的各类飞机型架监测系统的解决方案，分析了其优缺点，同时介绍了相关系统应用现状。

第二章主要介绍了本文所提出的型架监测系统的总体架构，介绍了其中关键部件 MEMS 传感器的原理。同时简要介绍了使用 MEMS 加速度计计算倾角的方法。

第三章详细介绍了型架监测系统的软硬件设计方案，硬件部分介绍了：设计总指标、器件选型及相关指标和硬件原理图，软件部分介绍了：系统中传感器部分的软件设计、RS485 总线中通信协议的设计和现场工控组态屏的软件设计。

第四章介绍并分析了型架监测系统的设计难点，同时提出了该问题解决的方案。章节最后介绍了型架姿态的调整方法，即操作人员如何依靠系统输出的数据辅助纠正型架姿态。

第五章是实验结果和分析，第二章、第三章和第四章的型架姿态监测系统设计，设计测试方案进行测试，由测试结果分析并验证该系统的实际性能能否满足设计需要。

第六章是总结与展望，客观总结了本文的工作，提出了几点不足之处，并对型架姿态监测在未来的发展和应用做出展望。

第 2 章 型架姿态监测系统的分析与设计

2.1 型架姿态监测系统设计的基本方案和原理

为了解决现有方案存在的测量时间长、测量不方便、不能实时测量、价格昂贵等问题，本方案提出了采用 MEMS 传感器作为整个系统关键测量部件的思路，将若干 MEMS 传感器安装在型架上，使用 MEMS 传感器检测重力加速度，从而解算出型架姿态。MEMS 传感器具有监测频带宽(响应频率高)，便于实时监测，价格适中等优点。

监测系统的架构图如图 2-1 所示。

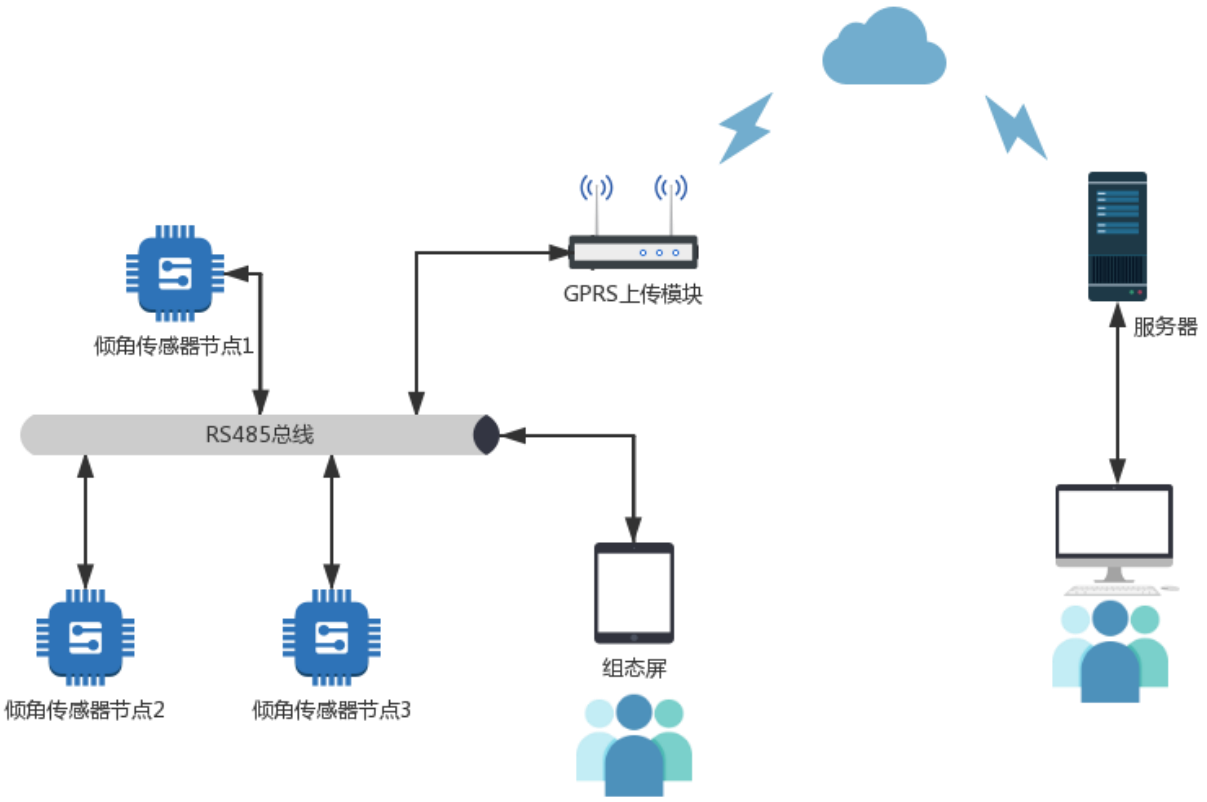


图 2-1 监测系统示意图

整个系统由 MEMS 倾角传感器、DTU、工控组态屏和后台服务器共四个部分组成，整个系统中，DTU、倾角传感器和工控组态屏共同连入 RS485 总线。其中：

(1) STM32 和 MEMS 加速度计构成 MEMS 倾角传感器，作为整个系统的测量器件，安装在型架上即可输出倾角数据，同时接收工控组态屏的命令完成工作。

(2) 工控组态屏是整个系统的显示和控制终端，操作人员通过组态屏查看倾角传感器的数据，并可使用组态屏设置倾角传感器的参数。

(3) DTU 负责记录传感器返回的倾角数据，并每间隔一分钟上传一次至服务器，后台服务器收到数据后，参照通信协议规定的数据格式解析并保存，方便操作人员使用 PC 端的浏览器查看历史数据。

2.2 MEMS 倾角传感器原理

MEMS 是微机电系统（Micro-Electro-Mechanical System）的缩写，是微电路和微机械按功能要求在芯片上的集成的工艺，属于微电子技术与机械工程结合的一种工业技术[9]。MEMS 的内部通常包括微处理器和若干获取外界信息的微型传感器，可以实现对力、声、光、热、电、磁信号的感知和处理。

MEMS 加速度传感器是目前技术非常成熟的 MEMS 产品之一。如图 2-2，与传统加速度计类似，MEMS 加速度计也是由弹簧、质量块、物理量转电信号这三大部分组成。其中，质量块(Mass)放置在芯片基座上，但不与基座紧固连接以方便移动；弹簧结构(Spring)的一部分紧固在基座上,另一部分与质量块相连；还有其余一些电容极板固定在基座上，当传感器收到加速度影响时，质量块在基座上产生位移，同时牵拉弹簧移动，从而改变各个极板间的距离，因为电容的大小受到两极板重叠面积或间距影响，此时基座上在质量块附近的电容的容值也会发生相应变化，并产生微弱的电流，经过其内部信号处理单元的采样，可以计算出加速度的大小，图 2-2 中的梳齿结构可以极大地扩大传感面积，提高测量精度，降低信号处理难度。除此以外，MEMS 加速度计还可以通过压阻式、力平衡式和谐振式等方式实现。另外通过在芯片上集成模拟、数字滤波器和其他信号调理电路，可以在进一步提高传感器的相关性能前提下，提供超低功耗、超小体积的加速度检测方案。

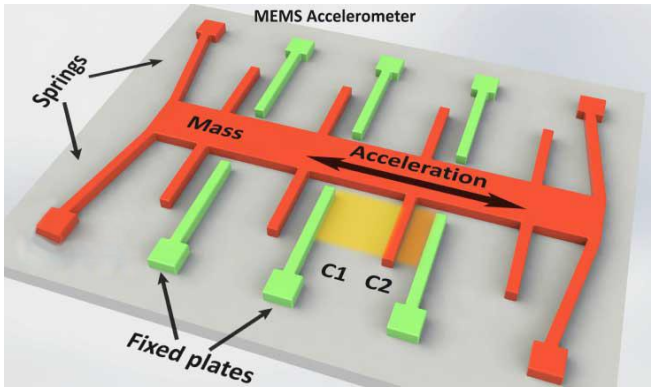


图2-2 MEMS加速度计原理示意图

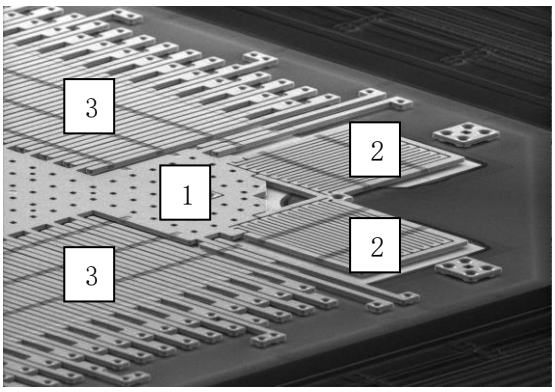


图2-3 MEMS加速度计扫描电镜照片

图 2-3 是一张关于 MEMS 加速度计的扫描电镜图片，图中的关键部分与原理图类似：

(1) 质量块，用于检测加速度从而产生位移。

(2) 类似弹簧的部件，一端固定在基座上，一端与质量块相连。

(3) 梳状电容，电容极板的两边极板一边固定在基座上，一边附着在质量快上，与质量块相连。

当外力施加在传感器上时，质量块收到惯性力发生位移，改变电容间距，导致电容容值发生变化，从而产生电信号。

基本的原理是 加速度—>质量块受到惯性力—>弹簧使质量块的位移与惯性力产生相关—>质量块的位移使梳状电容产生变化—>电信号。

可以通过输出的电信号计算传感器受到的加速度。图 2-3 中的加速度计是双轴加速度计，所以可以看到其有上下水平对称、互相垂直的两个部分，用于测量 X,Y 两个轴的加速度。

由于 MEMS 加速度计实质上是通过质量块受力后的变化来检测加速度，所以也会检测

到重力(重力加速度)，通过加速度计各轴的重力加速度分量数值，可以推算出加速度计当前的角度姿态[10][11]。

通过加速度数值解算姿态的方法如图 2-4 和 2-5

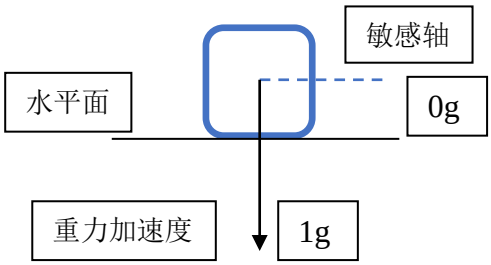


图 2-4 加速度计水平放置

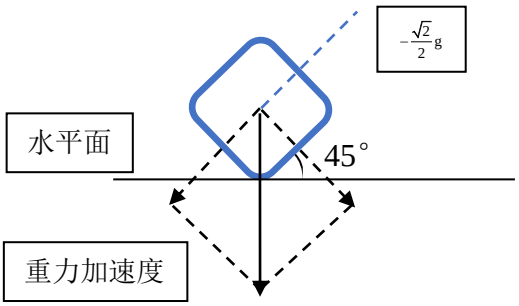


图 2-5 加速度计倾斜 45° 放置

先以单轴加速度计为例，图中的圆角矩形表示单轴 MEMS 加速度计，从其中心引出的虚线表示它的敏感轴(可以得到沿该轴向的加速度数值)。

加速度计水平放置时，如图 2-4，在竖直方向上受到 1g 的重力加速度，但在敏感轴上没有中立加速度分量，此时输出数值为 0。

加速度计倾斜 45° 放置时，如图 2-5，在敏感轴方向上受到 $\frac{\sqrt{2}}{2}$g 的重力加速度分量，此时输出数值为 $\frac{\sqrt{2}}{2}$g。设加速度计与水平面的倾角为 α，加速度的敏感轴输出为 η。如图 2-5 中的几何关系可得：

$$\alpha = \arcsin(\frac{\eta}{g})$$

双轴、三轴传感器测量角度姿态时，计算方式也与此类似：设加速度计某一轴与水平面的倾角为 θ_i，此时加速度计该轴的输出为 a_i，通过加速度计输出数据计算倾角的公式为：

$$\theta_i = \arcsin(\frac{a_i}{g})$$

第 3 章 MEMS 传感器采集系统设计

3.1 MEMS 传感器硬件设计

3.1.1 总体硬件性能设计指标

结合最初的设计指标和元器件选型后的实际指标，整个型架监测系统的电气指标和性能指标如下表所示。

电气设计指标如表 3-1 所示

表 3-1 型架监测系统电气设计指标

电气指标	最小值	典型值	最大值	单位
电源电压	+4		+40	V(DC)
工作电流		40		mA
工作温度	-40		+85	℃
接口		RS485		
传输距离			500	m

系统性能的设计指标如表 3-2 所示

表 3-2 型架监测系统整体性能设计指标

性能指标	最小值	典型值	最大值	单位
测量范围	-30		+30	°
测量轴		3		
零点漂移	-0.0086	±0.0012	+0.0086	°/°C
分辨率		0.0003		°
精度		0.006		°
输出频率		10		Hz

3.1.1 MEMS 加速度计 ADXL355

ADXL355 是 ADI 公司(Analog Devices, Inc)生产的带数字输出的低噪声、低漂移、低功耗 3 轴加速度计。其内置 20-bit ADC 和可编程数字低通、高通滤波器，并配置有 SPI 接口。需要 2.25V~3.6V 供电，可工作在-40~+125℃，测量时工作电流为 200μA。

ADXL355 的各项关键参数指标如表 3-3 所示(只节选了量程为±2.048g 的部分指标)。

表 3-3 ADXL355 相关参数

参数	测试条件/备注	最小值	典型值	最大值	单位
测量范围	用户可选		±2.048		g
			±4.096		g
			±8.192		g
非线性度	±2g		0.1		%FS
灵敏度	±2g	235520	256000	276480	LSB/g
比例因子	±2g		3.6		μg/LSB
温度导致的灵敏度变化	−40°C to +125°C		±0.01		%/°C
零点输出值	±2 g	-75	±25	+75	mg
由温度导致的零点漂移	−40°C to +125°C	-0.15	±0.02	+0.15	mg/°C
噪声密度			25		μg/ √Hz

为了获得较高的灵敏度，在使用时选择±2.048g 的量程，按照表 3-1 中的比例因子参数计算，当传感器姿态处于 0° 附近时，它的分辨能力为

$$\arcsin(\frac{0+3.9\mu g}{1g})-\arcsin(\frac{0}{1g})=0.00022°$$

但可以分辨的最小加速度值由总噪声决定，这与噪声密度和测量带宽有关。总噪声是指相对于理想输出的随机偏差，等于噪声密度与噪声带宽平方根的乘积，可以通过下面公式计算：

$$总噪声=噪声密度\times\sqrt{1.6\times带宽} \qquad (3\text{-}1)$$

其中，ADXL355 的噪声密度典型值为 25μg/ √Hz，带宽可以通过外部的数字滤波器或者启用 ADXL355 内部的滤波器进行选择。从设计上考虑，本系统没有太多动态测量的需求，只要保证适当的更新频率即可，更重要的需求是保证测量精度。那么在这一目标下，选择启用该芯片内部的低通滤波器，并配置为最低的截止频率 0.977Hz，那么带入公式(3-1)得：

$$总噪声=\frac{25\mu g}{\sqrt{Hz}}\times\sqrt{1.6\times 0.977\text{Hz}}=31.257\mu g$$

直接按照总噪声计算，可以粗略得到传感器的精度：

$$\arcsin(\frac{0+31.257\mu g}{1g})=0.00179°$$

但 MEMS 加速度计的噪声为高斯噪声且不相关，因此可通过对多个加速度计的输出求均值来降低噪声。在后续的数据处理中，进行均值滤波，可以进一步降低噪声对精度的影响。

在对 ADXL355 原理图的绘制过程中，有几点需要注意：虽然该芯片内置 LDO(低压差线性稳压器)，但为了使内部的模拟电路部分和数字电路部分(主要是 ADC)更稳定可靠地工作，一般推荐由外部性能更好的基准电压源提供工作电压和参考电压。为了进一步保证精度，设计中也使用了外部的高精度、低漂移基准电压源分别提供模拟和数字电源，

ADXL355 相关原理图如图 3-1 所示：

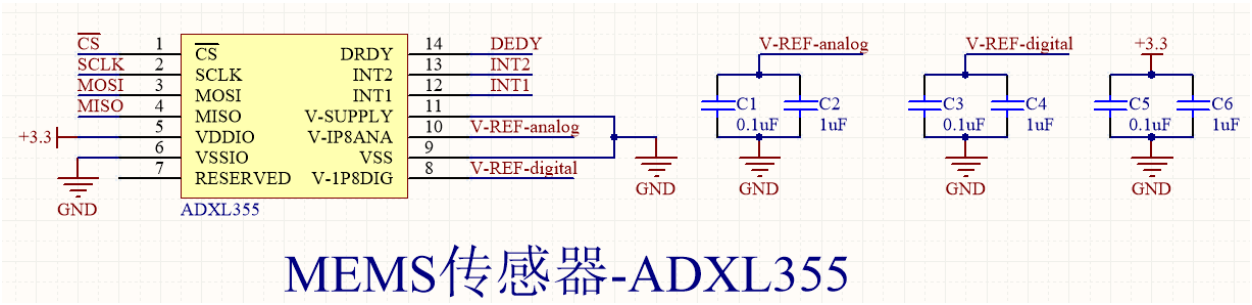


图 3-1 ADXL355 原理图

3.1.2 MCU STM32

MCU 选择 ST(意法半导体)公司生产的 STM32RBT6，该 MCU 是 ST 公司基于 ARM 公司的 Cortex™-M3 内核设计的微控制器，工作频率最高可达 72MHz，内置 128Kbyte 闪存程序存储器、20Kbyte 的 SRAM,可工作在 2.0V~3.6V 的电压下，带有 3 个 USART 接口

和两个 SPI 接口。

考虑到控制 MEMS 加速度计需要一个 SPI 接口，与外部通信需要一个 USART 接口，烧录程序需要一个 USART 接口(设置好 BOOT 位可以在 STM32 启动时通过串口烧写程序)，该 MCU 的性能和外设可以满足使用需求。

STM32 相关原理图如图 3-2 所示：

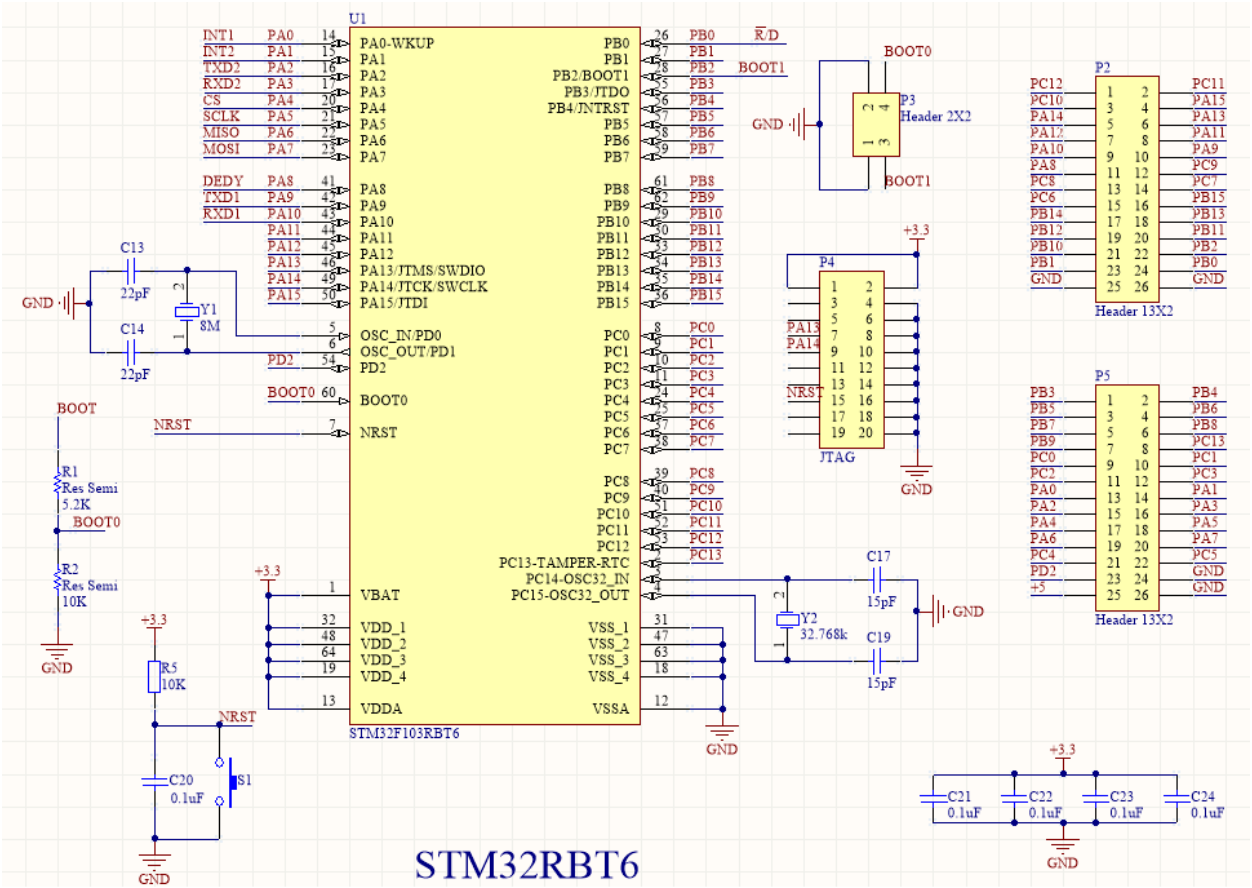


图 3-2　STM32 原理图

3.1.2 电源管理

首先考虑需要几种不同的供电电压，MCU 和 485 接口芯片都可以工作在 3.3V，MEMS 加速度计 ADXL355 需要工作在 1.8V 下。所以需要两种不同的供电电压，其中前者的供电电流需要大约是 40mA，并且不需要太高的精度。后者 ADXL355 的工作电流小于 1mA，但为保证精度，需要提供基准电压给其内置的 ADC，故需要选用高精度的电源芯片。

综上考虑，最后选择 TI 公司的 TPS7A6533-Q1 给 MCU、485 芯片供电，以及 TI 公司的 LM4132-1.8 给 ADXL355 提供高精度的 1.8V 电压。两者的性能指标分别如下表 3-4、3-5 所示。

表 3-4 TPS7A6533-Q1 的性能指标

性能指标	最小值	典型值	最大值	单位
输入电压	4		40	V
工作温度	-40		150	℃
输出电流			300	mA
精度(Vout=3.3V , Vin=Vout+1~16V)	-2%		+2%	

表 3-5 LM4132-1.8 的性能指标

性能指标	最小值	典型值	最大值	单位
输入电压			5.5	V
工作温度	-40		125	℃
输出电流			20	mA
精度	-0.05%		+0.05%	
温度系数	10		20	ppm/℃
线性调整率($\Delta V_{REF}/\Delta V_{IN}$)		30		ppm/V
负载调整率($\Delta V_{REF}/\Delta I_{LOAD}$)		25		ppm/mA
长期稳定性($\Delta V_{REF}-1000Hours$)		50		ppm

电源部分相关原理图如图 3-3 所示：

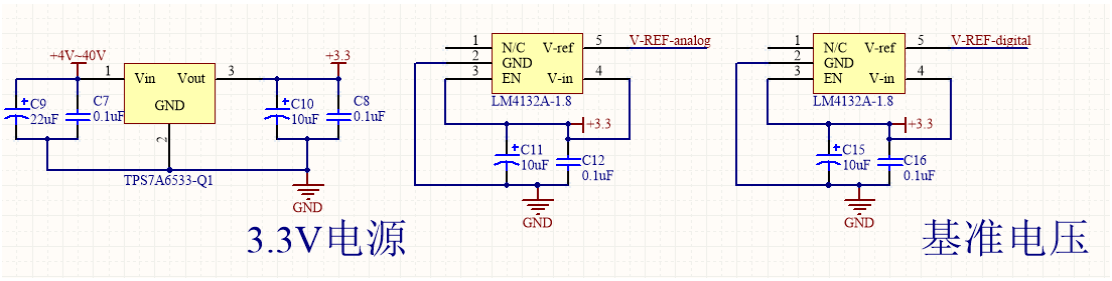


图 3-3　电源部分原理图

3.1.2 RS485 收发器 SN65HVD72

考虑到装配车间中的复杂环境、现场可能存在的电磁干扰以及需要多设备的连接，所以在系统设计之初就选择了采用 RS485 总线。RS485 收发器选用 TI 公司的 SN65HVD72，可以工作在 3.3V 电压下，最远传输距离可达 2000m，通过 STM32 控制收/发是能引脚，作为半双工的 RS485 接口使用。RS485 收发器原理图如图 3-4 所示：

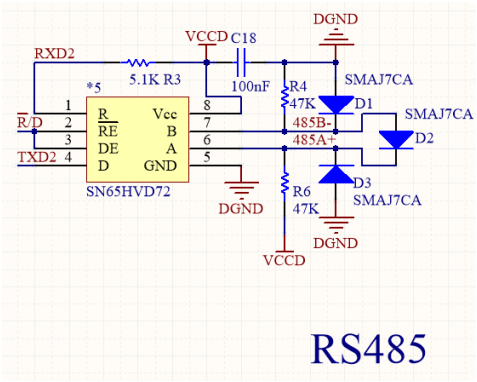


图 3-4　RS485 收发器原理图

3.2 MEMS 传感器软件设计

3.2.1 传感器软件总体设计

软件整体设计如图 3-5 所示：

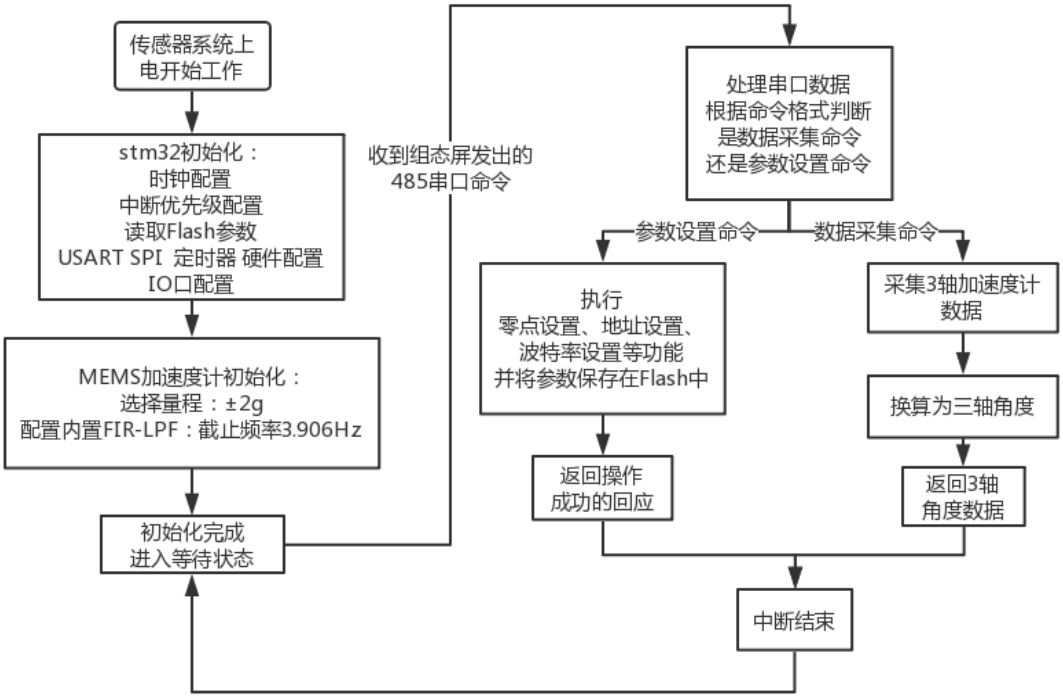


图 3-5　传感器软件流程图

传感器系统上电后，STM32 首先进行自身模块参数的配置：配置时钟，工作在 72Mhz，配置 USART 的优先级，随后读取 Flash 中的参数(如：串口通信参数、自身地址码、传感器读数修正参数等)，在使用 Flash 中的参数或默认值配置串口和 SPI 接口以及各个 IO 口。

STM32 自身模块配置完成后，开始用 SPI 接口配置 ADXL355，选择最低量程±2g，由于没有动态测量的要求，将内部的 FIR 低通滤波器的截止频率配置为 3.906Hz，以尽可能地减少 MEMS 器件内部噪声对测量精度的影响。设置完加速度计参数后进入等待状态。

当接收到 485 总线上的命令时，进入串口中断处理函数，如果是参数类的设置，则进入单独的参数设置函数，如果是数据采集命令，则采集 ADXL355 的加速度值，并将其转换为角度，并发送到 485 总线上。

3.2.2 通信协议设计

考虑到装配车间中的复杂环境、现场可能存在的电磁干扰以及需要多设备的连接，所以在系统设计之初就选择了采用 RS485 总线。RS485 总线采用查分传输，大大减少了共模干扰，并且支持多节点并联，但 RS485 只是物理层协议，想要保证数据在总线上有序的传输，还需要上层的数据链路层协议。本系统采用了现在工业总线中常用的 modbus 协议进行组网数据传输，modbus 协议是由 Modicon 公司提出的基于主-从机的数据链路层协议，用来保证整个总线上数据的有序传输，运行 modbus 协议的总线上只有一个主机，由该主机来主动发送各类命令[12]。

本系统在 modbus 总线上用 BCD 码传输数据，这种方式称为 RTU 模式，在 modbus

通信协议下，命令一般由地址码(1Byte)、功能码(1Byte)、寄存器地址码(2Byte)、寄存器数量码(2Byte)/数据域(2Byte)和 CRC 校验码(2Byte)组成。

其中，地址码用来区分发送给不同设备的信息，功能码用来区分主设备对从设备的各种操作，本系统只用到了两种功能码，分别是 03(代表读操作)和 06(代表写操作)，寄存器地址码用来决定一条命令的操作和那个寄存器相关。不同的功能码会导致寄存器地址码之后的帧结构不相同，例如：当功能码为 03 时，代表是读操作，寄存器地址码后跟随的是寄存器数量码，表示这条命令要读取寄存器地址码之后的多少个寄存器的值；当功能码为 06 时，代表是写操作，寄存器地址码后跟随的是数据域，数据与中是需要写入指定寄存器的值。

传感器系统中的部分命令如下表 3-6 所示：

表 3-6 传感器系统的部分 modbus 命令帧格式

命令功能	命令帧结构						
	地址码	功能码	寄存器高位地址	寄存器低位地址	寄存器数量/数据域高位	寄存器数量/数据域低位	CRC 校验码
读取 X 轴角度	设备地址	03	00	01	00	02	XXXX
读取 Y 轴角度	设备地址	03	00	02	00	02	XXXX
设置波特率	设备地址	06	00	01	00	02	XXXX
设置模块地址	原地址	06	00	02	00	新地址	XXXX

3.3 工控显示屏程序设计

传感器系统采用北京昆仑通态公司的 7 寸触摸组态屏作为控制、显示终端，该组态屏是一套以先进的 Cortex-A8 CPU 为核心（主频 600MHz）的高性能嵌入式一体化触摸屏。产品设计采用了 7 英寸高亮度 TFT 液晶显示屏（分辨率 800×480），四线电阻式触摸屏（分辨率 4096×4096）。同时还预装了 MCGS 嵌入式组态软件（运行版），具备强大的图像显示和数据处理功能。组态屏的产品参数如图 3-6 所示：

产品特性		外部接口	
液 晶 屏	7″ TFT	串行接口	COM1(RS232)，COM2(RS485)，可扩展(COM3，COM4)
背 光 灯	LED	USB接口	1主1从
显示颜色	65535真彩	以太网口	-
分 辨 率	800×480	CAN接口	可扩展
显示亮度	200cd/㎡	环境条件	
触 摸 屏	电阻式	存储温度	-10℃~60℃
输入电压	24±20%VDC	工作温度	0℃~45℃
额定功率	5W	工作湿度	5%~90%
处 理 器	Cortex-A8，600MHz	产品规格	
内　　存	128M	机壳材料	工业塑料
系统存储	128M	面板尺寸	226.5×163(mm)
铁电存储	可扩展	机柜开孔	215×152(mm)
SD卡存储	可扩展	认　　证	
组态软件	MCGS嵌入版	产品认证	CE/FCC

图 3-6 组态屏的产品参数

通过在专用的软件中编写程序，可以实现用该触摸屏采集数据、设置参数以及报警功

能[13]。该屏幕用 RS485 接口连入传感器总线中，并使用 modbus 协议，因此可以保证总线中数据的有序传输。本系统主要为组态屏编写了两个界面，分别是实时更新角度数据的主界面和方便历史数据会看以及保存的数据保存界面。

其中，组态屏的主界面如图 3-7 所示：

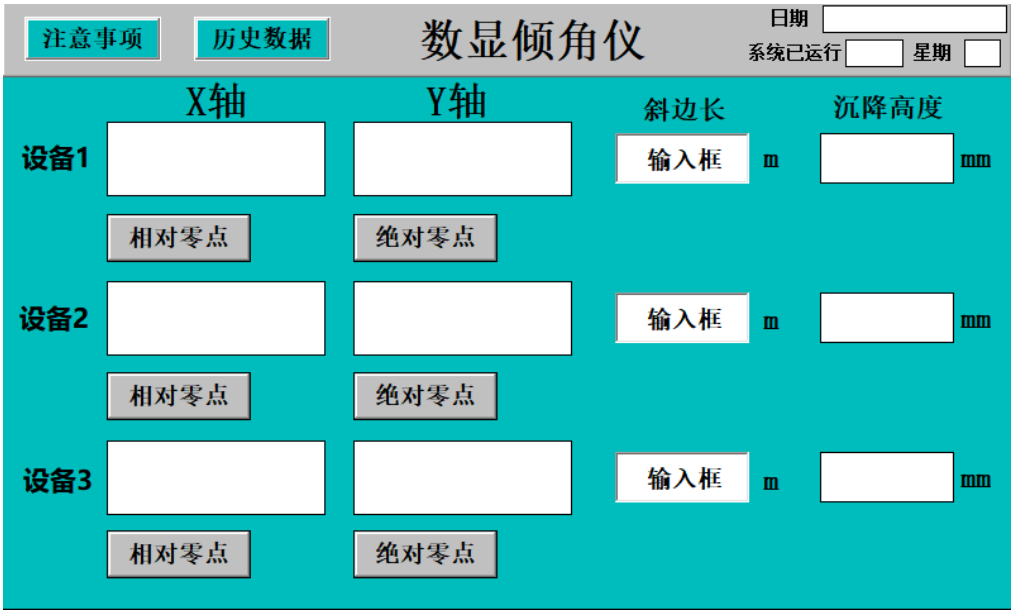


图 3-7 传感器系统主控屏幕的主界面

当组态屏处于该界面时，组态屏会用 10Hz 的速率分别轮询三个倾角传感器的 X 轴、Y 轴角度，并将数据实时更新在屏幕上。

点击左上角的历史数据，还可以进入如图 3-8 的数据保存界面

图 3-8 传感器系统主控屏幕的数据保存界面

组态屏自上电运行后，每间隔一分钟会自动保存三个传感器的 X,Y 轴数据，在右方选择开始、结束时间，左侧就会显示出这一时间段内的角度数据，再点击右侧的“执行导出到 U 盘”可以把这部分数据保存到 U 盘 excel 表格中

第 4 章 型架姿态监测与调整

4.1 型架姿态监测难点

4.1.1 型架支撑非刚性

从图 1-1 中截取单个型架模型，如图 4-1 所示

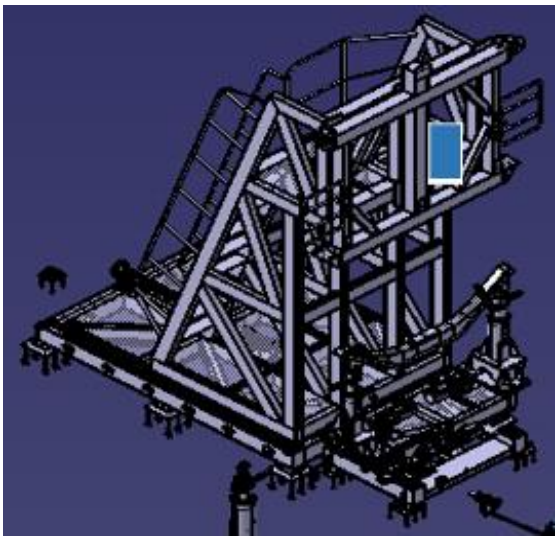


图 4-1 型架示意图

型架由基座和基座上方的操作平台组成，基座为操作台提供平整的地基，装配时，型架前方的支撑台负责支撑待装配的飞机机身，工人则站在操作平台上方为机身装配固。基座的四个角配有可独立调节升降的支撑柱，操作台直接连接在基座上方。

在装配过程中，由于各种原因，会导致基座四周产生不同程度的沉降，型架姿态监测系统通过监测型架角度姿态，辅助操作人员及时修正型架姿态至最初的状态，并在严重时发出警报。

本系统最初的设计思路是把一个双轴倾角传感器安装在基座中心，以此测量基座 X,Y 轴倾斜角度，再利用基座中心到边的距离，即可测算出型架基座四个角的沉降值。但在实测之后发现此方案并不可行，型架中的基座大概是一个 3m*5m 的金属平台，在自重和上方操作平台的影响下，整个基座平面并不是刚性的，这就导致只测量中心处的 X,Y 轴角度，不能精确计算出整个基座的沉降值。

4.1.2 配装工作会对监测带来影响

测量倾角有很多种方式，用 MEMS 加速度计测量倾角的优点是：价格适中、体积小、功耗低、便于安装、动态特性好，但缺点在于 MEMS 加速度计在检测到重力加速度的同时，也能测量到被测物体的运动加速度。操作人员在工作过程中，必然会有搬运装配固件，使用装配工具等动作，从而使型架产生额外的运动加速度，这些运动加速度也会被 MEMS 加速度计检测到，从而可能会对倾角的计算产生影响，需要在传感器系统中考虑如何尽量减少此类影响。

4.2 基于多节点测量的型架姿态监测解决方案

为了解决章节 4-1 中提出的问题，本监测系统采用了多节点测量的解决方案，并附以其他技术手段保证实时检测的准确性。

测量部分由三个独立的 MEMS 倾角传感器组成，分别安装在型架的三个位置。组态屏安装在型架附近的数据监控区，并连入 485 总线，以 10Hz 的频率实时显示三个传感器节点的数据，如果数据超过阈值则会报警。同时，485 总线上的 GPRS 上传模块，每隔 1 分钟接收一组总线上三个传感器的数据，并打包成 IP 帧，上传到服务器。工作人员可以在 PC 上通过 WEB 浏览器访问指定网址，获取某个时间段的历史数据或者整体打包下载。

3 个倾角传感器具体安装位置如图 4-3 所示，两个传感器分别安装到型架基座的两个角上，另一个传感器安装到操作平台上。

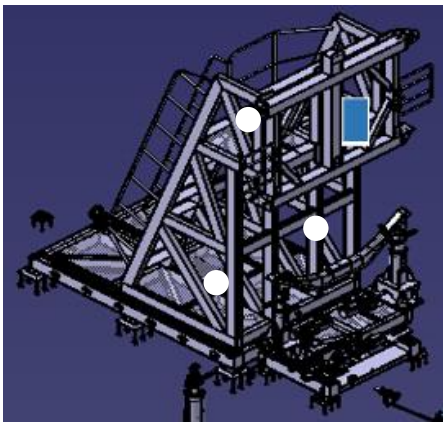


图 4-3 传感器安装位置示意图

4.3 型架姿态调整方案

由于传感器从制造到使用中存在一些不可避免的误差，例如：MEMS 加速度计的硅基板和封装的对齐误差、MEMS 元器件焊接到 PCB 电路板上的倾斜误差、PCB 电路板安装到产品外壳中的误差以及产品最终安装到被测物体上的误差。这些误差会导致测量得到的数据与物体的真实值有所偏离，对于不方便使用外部基准的被测物体，需要提前标定、修正传感器或使用一些消除安装误差的校准算法，但对于方便使用外部基准的被测物体，可以使用相对零点法。

现将传感器与被测物体固定牢固，随后将被测物体的姿态移动到由外部基准标定过的零点位置，此时让传感器记下此时读数，并将以后的数据减去此读数做处理，这种处理方式称为相对零点[14][15]。在本监测系统的使用过程中，需要先安装好各传感器，然后用激光测绘仪将型架调整到标准位置，随后使用组态屏上的相对零点功能，给各个传感器设置相对零点，此时，各个传感器的示数归零，如果型架姿态发生变化，传感器的读数也会变化。

型架姿态修正的调整策略是先调整短边，在调整长边。当传感器的读数偏离零点时，一次调整型架基座短边的两个支撑柱，使得两个传感器在长轴方向上的角度接近于零，然后再整体同时升高/降低型架基座某一长边的两个支撑柱，使得两个传感器在短轴上的读数接近于零。随后再根据安装在操作平台上的传感器的读数进行微调。

第 5 章 实验结果与分析

5.1 实验方案

5.1.1 以徕卡激光测绘系统作为基准

选择徕卡公司的全自动三维建筑测量仪 AT960 系统作为基准，这套激光测绘系统的性能参数如图 5-1 所示。

Accuracy *	$U_{x,y,z}$ = +/-15 µm + 6 µm/m

* All accuracies are specified as maximum permissible errors (MPE) and in accordance with ASME B89.4.19-2006 & ISO10360-10 using precision Leica 1.5" Red Ring Reflectors up to 60 m distance unless otherwise noted.

Angle accuracy	+/-15 µm + 6 µm/m
Distance accuracy AIFM	+/-0.5 µm/m
Dynamic lock on	+/-10 µm
Orient to Gravity (OTG)	$U_{z(OTG)}$ = +/-15 µm + 8 µm/m
Environmental working conditions	
Dust/Water	IP54 (IEC 60529)
Operating temperature	0°C to +40°C
Data output rate	1 000 points/sec
Laser safety	Class 2 IEC 60825-1 (2014-05) "Safety of laser products" EN 60825-1 (2007-10) "Safety of laser products"

图 5-1 徕卡激光测绘仪性能参数

徕卡激光测绘仪使用红色可见激光进行测量，当设备放置的倾斜角度处于±3°范围内时，仪器可以自动调平，测量系统距离被测物体小于 10m 时，测量误差小于 0.075mm。

测量点的选取如图 5-2 所示，图中左上方物体代表型架，右下方物体代表激光测绘仪，选取型架操作平台上的四个白点的位置作为测量的基准点。在装配过程中，这四个点位置的准确性直接影响着装配精度，调整型架姿态的目的也是保持这四个基准点的位置准确性。

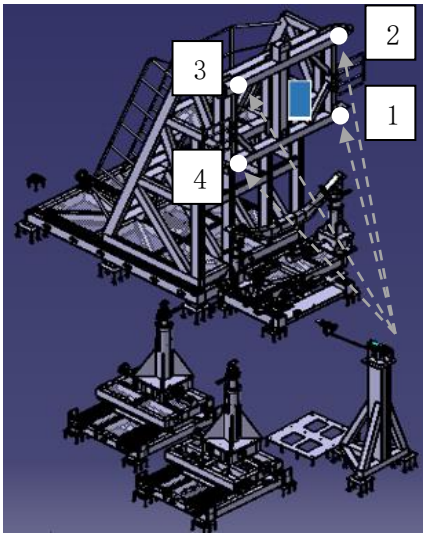


图 5-2 激光测绘仪测量点的选取

5.1.2 基于模拟沉降产生的测试监测系统

为了验证型架监测系统的性能，本设计方案提出了模拟沉降的方案。首先将型架监测系统的传感器安装到指定位置，部署组态屏，并将相关设备接入 RS485 总线。随后使用激光测绘仪将型架调整到正常姿态，具体的调整方法是参照激光测绘仪的数据，调整型架基座的四个支撑柱，使得两边同一侧的两个基准点的 X,Y 坐标相等，以此以四个基准点组成的平面垂直于水平面，保证安装精度。将型架调整至标准姿态后，将三个 MEMS 倾角传感器的当前状态设置为零点，并保存记录此时四个基准点的 X,Y,Z 坐标，设这四个基准点的坐标分别为$P_1(x_1,y_1,z_1)$、$P_2(x_2,y_2,z_2)$、$P_3(x_3,y_3,z_3)$、$P_4(x_4,y_4,z_4)$。

随后将型架基座的四个支撑柱随机升高/下降 0.1mm~1mm，之后按照章节 4-3 的调整方案，依靠监测系统的传感器读数对四个支撑柱进行调整，使三个传感器的数值重新接近于零。调整完毕后，再使用激光测绘仪测量型架操作平台上四个基准点的 X,Y,Z 三轴坐标，设此时这四个基准点的坐标为$P_1'(x_1',y_1',z_1')$、$P_2'(x_2',y_2',z_2')$、$P_3'(x_3',y_3',z_3')$、$P_4'(x_4',y_4',z_4')$。最后通过计算各个基准点两次测量结果之间的位移距离$\Delta_i=|P_iP_i'|$来检验监测系统的性能。

5.2 测试结果

先调整将型架到标准姿态，随后随机调整支撑柱，再依靠倾角传感器复原姿态。其中，随机调整和复原姿态的测试进行三组，测试得到的数据如表 5-1 所示。

表 5-1 型架监测系统测试数据

数据组别	基准点	X 坐标(mm)	Y 坐标(mm)	Z 坐标(mm)	Δ(mm)
初始姿态	P_1	-66.24	3112.90	3619.62	--
	P_2	-66.30	3112.85	5168.45	--
	P_3	1434.18	503.48	5168.39	--
	P_4	1434.13	503.52	3619.58	--
第一次模拟沉降再复原后	P_1	-66.16	3112.83	3619.58	0.1135
	P_2	-66.18	3112.80	5168.43	0.1315
	P_3	1434.09	503.40	5168.38	0.1208
	P_4	1434.05	503.42	3619.54	0.1341
第二次模拟沉降再复原后	P_1	-66.15	3112.82	3619.59	0.1240
	P_2	-66.19	3112.80	5168.47	0.1224
	P_3	1434.11	503.41	5168.40	0.0994
	P_4	1434.09	503.43	3619.56	0.1004
第三次模拟沉降再复原后	P_1	-66.33	3112.96	3619.66	0.1153
	P_2	-66.36	3112.93	5168.50	0.1118
	P_3	1434.26	503.56	5168.45	0.1280
	P_4	1434.23	503.54	3619.62	0.1095

上表中的 Δ 由 $\Delta_i = |P_i P_i'|$ 计算得到，将四个基准点的三次实验偏差绘制成曲线，得到图 5-3，其中 Pij 代表第 i 组测试中第 j 个基准点的偏差 Δ，三组实验结果表明使用型架监测系统恢复型架姿态的误差大概在 0.12mm 左右，最大不超过 0.14mm，符合系统的设计要求。

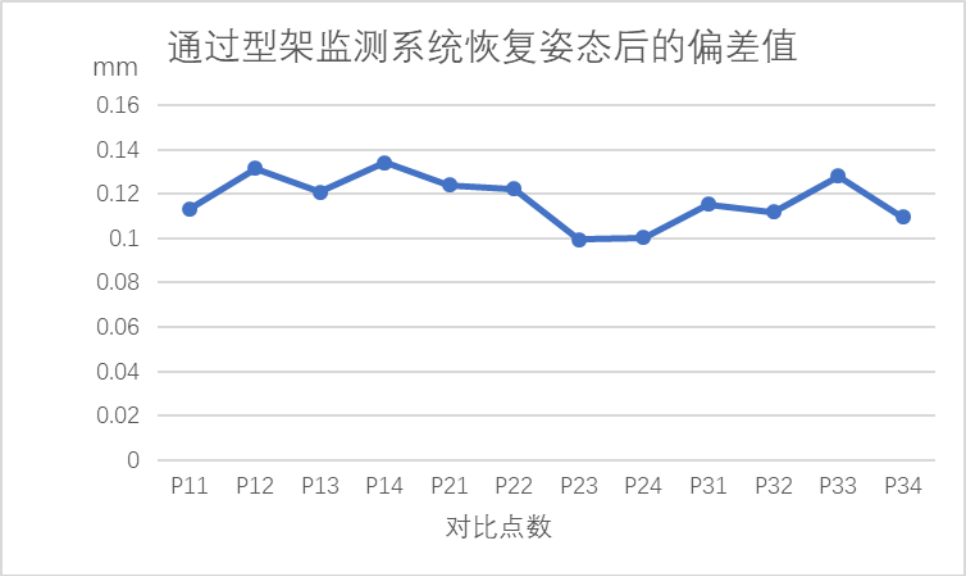


图 5-3 型架监测系统测试的偏差值结果

5.3 结果分析

当传感器姿态处于 0° 附近时，它的分辨能力为

$$\arcsin(\frac{0+3.9\mu g}{1g}) - \arcsin(\frac{0}{1g}) = 0.00022°$$

考虑到最多大约会有 31.257μg 的随机噪声，当传感器姿态处于 0° 附近时，它的精度大约为

$$\arcsin(\frac{0+31.257\mu g}{1g}) = 0.00179°$$

如果把型架基座长边的长度按 4m 计算，那么传感器系统能分辨的沉降为

$$\sin(0.00179°) \times 4000mm = 0.125mm$$

但在系统中由于没有太高的动态测量要求，所以传感器工作在较低的带宽下，并且每采集 5 组数值进行中值滤波后再输出，可以稍稍提高传感器的精度。但由于传感器存在一定的温度漂移和非线性因素，可能导致精度受到影响。综合以上因素的影响，再对比图 5-2 给出的测试结果，系统精度的测试结果跟设计精度大致匹配，在设计可接受的范围内。

第 6 章 总结与展望

6.1 总结

本文设计并实现了用于大飞机装配的基于 MEMS 传感器平面装配监测系统，系统经过初始校准后能够实时监测用于飞机装配的型架的姿态，并辅助操作人员将受到影响的型架恢复到初始姿态，或在型架姿态变化严重时发出警报。经过实地实验，系统的实时性和准确性都可以满足设计需求。但由于此方案提出不久，还没有充分的时间进行一些长期稳定性的测试。为了验证系统的鲁棒性，还需要进行一些后续的测试。

6.1.1 关于系统软硬件设计

系统在传感器硬件上选用高精度、低噪声、低漂移的 MEMS 加速度计，保证了测量精度和稳定性。现场设备均连入 RS485 总线以应对复杂的工装环境，总线通信遵照 modbus 协议，因此可兼容其他支持 modbus 的设备，方便扩展。

6.1.2 关于监测设计与实现

经过实验，监测系统的精度可以满足设计需要。设置系统中倾角传感器的参数需要在组态屏上进行，操作人员可以分别在现场的组态屏或办公室的 PC 机上查看实时数据和历史数据，不仅方便了实时监督，还可以回看型架姿态的变化历史。

6.1.3 关于型架姿态调整方法

型架姿态的调整方法目前还是需要人工操作，另外受限于产品制造、安装中产生的各种误差，目前仍然需要在整个系统正式启动前使用更高精度的激光测绘仪进行标定初始状态。

6.2 展望

虽然已经验证了系统的可行性，但系统的精度离激光测绘仪还是有一定差距，如果能在之后找到更高性能的 MEMS 加速度计器件，还有可能进一步提升系统精度。在保证精度的前提下，还需要重视传感器系统的温度漂移和长期稳定性，可以在以上方案的基础上对传感器系统的温度漂移做进一步的测试，进行更加精确的标定，或考虑加入 MEMS 陀螺仪补偿振动对测量的影响[16][17]。另外整个监测系统的姿态调整部分，目前仍然需要操作员人工操作，日后可以考虑在四个支撑柱加入步进电机，实现调整的自动化。

参考文献

[1] 刘韶光,范欢欢,范晓龙,王宏旭. 激光跟踪仪在飞机改装型架装配中的应用研究[J]. 机械制造,2015,53(3)：45-47.

[2] 王彦喜,闵俊,刘刚. 激光跟踪仪在飞机型架装配中的应用[J]. 航空制造技术,2010,(19)：92-94.

[3] 王巍,黄宇,庄建平. 激光跟踪仪在飞机装配工装制造中的应用[J]. 航空制造技术,2004,(12)：81-84.

[4] 李广云. LTD500 激光跟踪测量系统原理及应用[J]. 测绘工程,2001,10(4)：3-8.

[5] 刘湘锴,吴能森,吴承彬. 流体静力水准测量技术发展及应用分析[J]. 河南城建学院学报,2015,24(6)：23-26.

[6] 陈容,强永兴,许德明,郭俊兵. 静力水准仪在碧口水电站的应用[J]. 西北水电,2011,(1)：17-20.

[7] 孙泽信,张书丰,刘宁. 静力水准仪在运营期地铁隧道变形监测中的应用及分析[J]. 现代隧道技术,2015,52(01)：203-208.

[8] 梁振英,李明. 流体静力水准测量的精度[J]. 测绘科技动态,1992,(Z1)：7-9.

[9] 谷雨. MEMS 技术现状与发展前景[J]. 电子工业专用设备,2013,(08)：1-8.

[10] 雷凌毅,唐恭富,姚毅. 基于 MEMS 加速度计的倾斜姿态角传感器设计[J]. 兵工自动化,2014,(6)：71-73.

[11] 刘武发, 蒋蓁, 龚振邦. 基于 MEMS 加速度传感器的双轴倾角计及其应用[J]. 传感器与微系统, 2005, 24(3)：86-88.

[12] 朱小襄. ModBus 通信协议及编程[J]. 信息化研究, 2005, 31(7)：42-44.

[13] 杨新盛, 姜明晓. 基于昆仑通态 TCP7062K 汽车升降实验机监控系统[J]. 国外电子测量技术, 2012(11)：53-54.

[14] 白普俊, 薛娜, 刘松涛,等. 基于激光追踪仪的精密转台角度标定方法[J]. 华南理工大学学报 (自然科学版), 2016, 44(1):100-107.

[15] 刘春学, 郑哲恩, 孙坚,等. 六自由度工业机器人本体标定实验的研究[J]. 机床与液压, 2017(5).

[16] 徐哲,刘云峰,董景新. MEMS 加速度计温漂预测补偿模型[J]. 中国惯性技术学报,2012,(5)：601-604.

[17] Madgwick S O H, Harrison A J L, Vaidyanathan R. Estimation of IMU and MARG orientation using a gradient descent algorithm[C]//Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on. IEEE, 2011： 1-7.

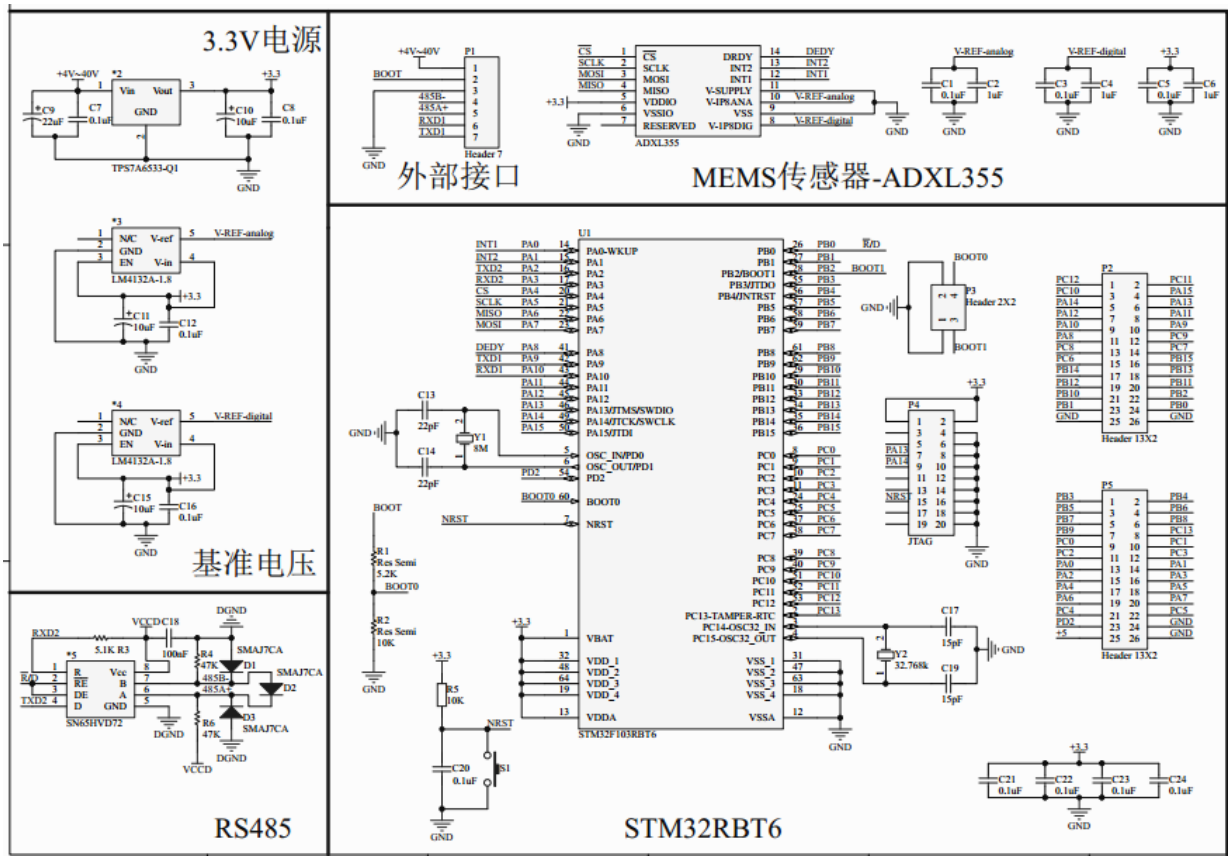
致　谢

感谢江南大学四年来对我的辛苦培育，特别感谢物联网工程学院为我提供了良好的学习环境、感谢领导、老师们四年来对我无微不至的关怀和指导，让我得以在这四年中学到很多有用的知识。在此，我还要感谢在班里同学和朋友，感谢你们在我遇到困难的时候帮助我，给我支持和鼓励，感谢你们。

特别感谢我的校内指导王呈老师和企业指导时广轶老师，在本系统开发中给予我悉心指导，从系统开发到结束，是他们给我鼓励与指引使我能够克服重重困难，将系统完善，在此谨向王呈、时广轶老师致以诚挚的谢意和崇高的敬意。谢谢!

附　录

电路原理图：



代码：

1.ADXL355 的端口配置函数

```
#define ADXL355_SPI                        SPI1
#define ADXL355_SPI_CLK                    RCC_APB2Periph_SPI1
#define ADXL355_SPI_SCK_PIN                GPIO_Pin_5
#define ADXL355_SPI_SCK_GPIO_PORT          GPIOA
#define ADXL355_SPI_SCK_GPIO_CLK           RCC_APB2Periph_GPIOA
#define ADXL355_SPI_SCK_SOURCE             GPIO_PinSource5
#define ADXL355_SPI_SCK_AF                 GPIO_AF_5
#define ADXL355_SPI_MISO_PIN               GPIO_Pin_6
#define ADXL355_SPI_MISO_GPIO_PORT         GPIOA
#define ADXL355_SPI_MISO_GPIO_CLK          RCC_APB2Periph_GPIOA
#define ADXL355_SPI_MISO_SOURCE            GPIO_PinSource6
#define ADXL355_SPI_MISO_AF                GPIO_AF_5
#define ADXL355_SPI_MOSI_PIN               GPIO_Pin_7
#define ADXL355_SPI_MOSI_GPIO_PORT         GPIOA
#define ADXL355_SPI_MOSI_GPIO_CLK          RCC_APB2Periph_GPIOA
#define ADXL355_SPI_MOSI_SOURCE            GPIO_PinSource7
#define ADXL355_SPI_MOSI_AF                GPIO_AF_5
```

```
#define ADXL355_SPI_CS_PIN                      GPIO_Pin_4
#define ADXL355_SPI_CS_GPIO_PORT                GPIOA
#define ADXL355_SPI_CS_GPIO_CLK                 RCC_APB2Periph_GPIOA
#define ADXL355_SPI_INT1_PIN                    GPIO_Pin_0
#define ADXL355_SPI_INT1_GPIO_PORT              GPIOA
#define ADXL355_SPI_INT1_GPIO_CLK               RCC_APB2Periph_GPIOA
#define ADXL355_SPI_INT1_EXTI_LINE              EXTI_Line0
#define ADXL355_SPI_INT1_EXTI_PORT_SOURCE EXTI_PortSourceGPIOA
#define ADXL355_SPI_INT1_EXTI_PIN_SOURCE    EXTI_PinSource0
#define ADXL355_SPI_INT1_EXTI_IRQn              EXTI0_IRQn
#define ADXL355_SPI_INT2_PIN                    GPIO_Pin_1
#define ADXL355_SPI_INT2_GPIO_PORT              GPIOA
#define ADXL355_SPI_INT2_GPIO_CLK               RCC_APB2Periph_GPIOA
#define ADXL355_SPI_INT2_EXTI_LINE              EXTI_Line1
#define ADXL355_SPI_INT2_EXTI_PORT_SOURCE EXTI_PortSourceGPIOA
#define ADXL355_SPI_INT2_EXTI_PIN_SOURCE    EXTI_PinSource1
#define ADXL355_SPI_INT2_EXTI_IRQn              EXTI1_IRQn



#define ADXL355_WHO_AM_I_ADDR           0x0F
#define ADXL355_CTRL_REG1_ADDR          0x20
#define ADXL355_CTRL_REG2_ADDR          0x21
#define ADXL355_CTRL_REG3_ADDR          0x22
#define ADXL355_CTRL_REG4_ADDR          0x23
#define ADXL355_CTRL_REG5_ADDR          0x24
#define ADXL355_REFERENCE_REG_ADDR      0x25
#define ADXL355_OUT_TEMP_ADDR           0x26
#define ADXL355_STATUS_REG_ADDR         0x27
#define ADXL355_OUT_X_L_ADDR            0x28
#define ADXL355_OUT_X_H_ADDR            0x29
#define ADXL355_OUT_Y_L_ADDR            0x2A
#define ADXL355_OUT_Y_H_ADDR            0x2B
#define ADXL355_OUT_Z_L_ADDR            0x2C
#define ADXL355_OUT_Z_H_ADDR            0x2D
#define ADXL355_FIFO_CTRL_REG_ADDR      0x2E
#define ADXL355_FIFO_SRC_REG_ADDR       0x2F
#define ADXL355_INT1_CFG_ADDR           0x30
#define ADXL355_INT1_SRC_ADDR           0x31
#define ADXL355_INT1_TSH_XH_ADDR        0x32
#define ADXL355_INT1_TSH_XL_ADDR        0x33
#define ADXL355_INT1_TSH_YH_ADDR        0x34
```

```
#define ADXL355_INT1_TSH_YL_ADDR          0x35
#define ADXL355_INT1_TSH_ZH_ADDR          0x36
#define ADXL355_INT1_TSH_ZL_ADDR          0x37
#define ADXL355_INT1_DURATION_ADDR        0x38


#define I_AM_ADXL355          ((uint8_t)0xD4)

#define ADXL355_MODE_POWERDOWN          ((uint8_t)0x00)
#define ADXL355_MODE_ACTIVE             ((uint8_t)0x08)

#define ADXL355_OUTPUT_DATARATE_1       ((uint8_t)0x00)
#define ADXL355_OUTPUT_DATARATE_2       ((uint8_t)0x40)
#define ADXL355_OUTPUT_DATARATE_3       ((uint8_t)0x80)
#define ADXL355_OUTPUT_DATARATE_4       ((uint8_t)0xC0)

#define ADXL355_X_ENABLE                ((uint8_t)0x02)
#define ADXL355_Y_ENABLE                ((uint8_t)0x01)
#define ADXL355_Z_ENABLE                ((uint8_t)0x04)
#define ADXL355_AXES_ENABLE             ((uint8_t)0x07)
#define ADXL355_AXES_DISABLE            ((uint8_t)0x00)

#define ADXL355_BANDWIDTH_1             ((uint8_t)0x00)
#define ADXL355_BANDWIDTH_2             ((uint8_t)0x10)
#define ADXL355_BANDWIDTH_3             ((uint8_t)0x20)
#define ADXL355_BANDWIDTH_4             ((uint8_t)0x30)

#define ADXL355_FULLSCALE_250                ((uint8_t)0x00)
#define ADXL355_FULLSCALE_500                ((uint8_t)0x10)
#define ADXL355_FULLSCALE_2000               ((uint8_t)0x20)


#define ADXL355_BlockDataUpdate_Continous    ((uint8_t)0x00)
#define ADXL355_BlockDataUpdate_Single       ((uint8_t)0x80)


#define ADXL355_BLE_LSB                      ((uint8_t)0x00)
#define ADXL355_BLE_MSB                      ((uint8_t)0x40)
```

```
#define ADXL355_HIGHPASSFILTER_DISABLE          ((uint8_t)0x00)
#define ADXL355_HIGHPASSFILTER_ENABLE           ((uint8_t)0x10)

#define ADXL355_INT1INTERRUPT_DISABLE           ((uint8_t)0x00)
#define ADXL355_INT1INTERRUPT_ENABLE        ((uint8_t)0x80)

#define ADXL355_INT2INTERRUPT_DISABLE           ((uint8_t)0x00)
#define ADXL355_INT2INTERRUPT_ENABLE        ((uint8_t)0x08)

#define ADXL355_INT1INTERRUPT_LOW_EDGE          ((uint8_t)0x20)
#define ADXL355_INT1INTERRUPT_HIGH_EDGE         ((uint8_t)0x00)



#define ADXL355_BOOT_NORMALMODE                 ((uint8_t)0x00)
#define ADXL355_BOOT_REBOOTMEMORY               ((uint8_t)0x80)



#define ADXL355_HPM_NORMAL_MODE_RES             ((uint8_t)0x00)
#define ADXL355_HPM_REF_SIGNAL                  ((uint8_t)0x10)
#define ADXL355_HPM_NORMAL_MODE                 ((uint8_t)0x20)
#define ADXL355_HPM_AUTORESET_INT               ((uint8_t)0x30)

#define ADXL355_HPFCF_0                 0x00
#define ADXL355_HPFCF_1                 0x01
#define ADXL355_HPFCF_2                 0x02
#define ADXL355_HPFCF_3                 0x03
#define ADXL355_HPFCF_4                 0x04
#define ADXL355_HPFCF_5                 0x05
#define ADXL355_HPFCF_6                 0x06
#define ADXL355_HPFCF_7                 0x07
#define ADXL355_HPFCF_8                 0x08
#define ADXL355_HPFCF_9                 0x09


#define ADXL355_CS_LOW()        GPIO_ResetBits(ADXL355_SPI_CS_GPIO_PORT,
ADXL355_SPI_CS_PIN)
#define ADXL355_CS_HIGH()       GPIO_SetBits(ADXL355_SPI_CS_GPIO_PORT,
ADXL355_SPI_CS_PIN)
```

```
    void ADXL355_Init(ADXL355_InitTypeDef *ADXL355_InitStruct);
    void ADXL355_RebootCmd(void);

    void ADXL355_INT1InterruptCmd(uint8_t InterruptState);
    void ADXL355_INT2InterruptCmd(uint8_t InterruptState);
    void ADXL355_INT1InterruptConfig(ADXL355_InterruptConfigTypeDef
*ADXL355_IntConfigStruct);
    uint8_t ADXL355_GetDataStatus(void);

    void ADXL355_FilterConfig(ADXL355_FilterConfigTypeDef *ADXL355_FilterStruct);
    void ADXL355_FilterCmd(uint8_t HighPassFilterState);
    void ADXL355_Write(uint8_t* pBuffer, uint8_t WriteAddr, uint16_t NumByteToWrite);
    void ADXL355_Read(uint8_t* pBuffer, uint8_t ReadAddr, uint16_t NumByteToRead);
```

2.ADXL355 的 SPI 配置函数

```
    void SPI_init(void)
    {

      GPIO_InitTypeDef GPIO_InitStructure;
      SPI_InitTypeDef       SPI_InitStructure;
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO | RCC_APB2Periph_GPIOA |
                             RCC_APB2Periph_GPIOB, ENABLE);
      RCC_APB2PeriphClockCmd(ADXL355_SPI_CLK ,ENABLE);

      GPIO_StructInit(&GPIO_InitStructure);
      GPIO_InitStructure.GPIO_Pin =ADXL355_SPI_CS_PIN ;
      GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_Init(ADXL355_SPI_CS_GPIO_PORT, &GPIO_InitStructure);

      GPIO_SetBits(ADXL355_SPI_CS_GPIO_PORT,ADXL355_SPI_CS_PIN);

      GPIO_StructInit(&GPIO_InitStructure);
      GPIO_InitStructure.GPIO_Pin
=ADXL355_SPI_SCK_PIN|ADXL355_SPI_MISO_PIN|ADXL355_SPI_MOSI_PIN;
      GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF_PP;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_Init(ADXL355_SPI_MOSI_GPIO_PORT, &GPIO_InitStructure);
```

```
    SPI_InitStructure.SPI_Direction=SPI_Direction_2Lines_FullDuplex;
    SPI_InitStructure.SPI_Mode=SPI_Mode_Master;
    SPI_InitStructure.SPI_DataSize = SPI_DataSize_16b;
    SPI_InitStructure.SPI_CPOL=SPI_CPOL_High;
    SPI_InitStructure.SPI_CPHA=SPI_CPHA_2Edge;
    SPI_InitStructure.SPI_NSS=SPI_NSS_Soft;
    SPI_InitStructure.SPI_BaudRatePrescaler=SPI_BaudRatePrescaler_256;
    SPI_InitStructure.SPI_FirstBit=SPI_FirstBit_MSB;
    SPI_InitStructure.SPI_CRCPolynomial=7;
    SPI_Init(ADXL355_SPI, &SPI_InitStructure);
    SPI_Cmd(ADXL355_SPI,ENABLE);
}
```

3.ADXL355 初始化配置函数

```
void ADXL355_init(void)
{
     SPI_init();
    ADXL355_write_byte(0x1E,0x00);
    ADXL355_write_byte(0x1F,0x00);
    ADXL355_write_byte(0x20,0x00);
    ADXL355_write_byte(0x21,0x00);
    ADXL355_write_byte(0x22,0x00);
    ADXL355_write_byte(0x23,0x00);

    ADXL355_write_byte(0x24,0x01);
    ADXL355_write_byte(0x25,0x01);
    ADXL355_write_byte(0x26,0x2B);
    ADXL355_write_byte(0x27,0x00);
    ADXL355_write_byte(0x28,0x09);
    ADXL355_write_byte(0x29,0xFF);
    ADXL355_write_byte(0x2A,0x80);
    ADXL355_write_byte(0x2C,0x0F);
    ADXL355_write_byte(0x2D,0x08);
    ADXL355_write_byte(0x2E,0x80);
    ADXL355_write_byte(0x2F,0x00);
    ADXL355_write_byte(0x31,0X0B);
    ADXL355_write_byte(0x38,0x00);
 }
```

4.ADXL355 的读写函数

```
u8 ADXL355_read_byte(u8 add)
{
       GPIO_ResetBits(ADXL355_SPI_CS_GPIO_PORT ,ADXL355_SPI_CS_PIN);
```

```
    SPI_I2S_SendData(ADXL355_SPI,(add|0x80)<<8|0x00);

    while(SPI_I2S_GetFlagStatus(ADXL355_SPI,SPI_I2S_FLAG_TXE)==RESET);

    while(SPI_I2S_GetFlagStatus(ADXL355_SPI, SPI_I2S_FLAG_RXNE)==RESET);

    GPIO_SetBits(ADXL355_SPI_CS_GPIO_PORT ,ADXL355_SPI_CS_PIN);

    return SPI_I2S_ReceiveData(ADXL355_SPI)&0xff;

  }
void ADXL355_write_byte(u8 add,u8 val)
{
    GPIO_ResetBits(ADXL355_SPI_CS_GPIO_PORT ,ADXL355_SPI_CS_PIN);
    SPI_I2S_SendData(ADXL355_SPI,add<<8|val);

    while(SPI_I2S_GetFlagStatus(ADXL355_SPI,SPI_I2S_FLAG_TXE)==RESET);

    while(SPI_I2S_GetFlagStatus(ADXL355_SPI, SPI_I2S_FLAG_RXNE)==RESET);

    GPIO_SetBits(ADXL355_SPI_CS_GPIO_PORT ,ADXL355_SPI_CS_PIN);
    SPI_I2S_ReceiveData(ADXL355_SPI)&0xff;

  }
void ADXL355_ReadXYZ(float *g)
{
    uint8_t BUF[6];
    int16_t temp;

    BUF[0] = ADXL355_read_byte(0x32);
    BUF[1] = ADXL355_read_byte(0x33);
     delay_ms(1);
    BUF[2] = ADXL355_read_byte(0x34);
    BUF[3] = ADXL355_read_byte(0x35);
     delay_ms(1);
    BUF[4] = ADXL355_read_byte(0x36);
    BUF[5] = ADXL355_read_byte(0x37);
      delay_ms(1);

    temp = (BUF[1] << 8) + BUF[0];
    if(temp < 0)
        temp = -temp;
```

```
        g[0] = (float)(temp * 3.9);
        temp = (BUF[3] << 8) + BUF[2];
        if(temp < 0)
            temp = -temp;
        g[1] = (float)(temp * 3.9);
        temp = (BUF[5] << 8) + BUF[4];
        if(temp < 0)
            temp = -temp;
        g[2] = (float)(temp * 3.9);
    }
```