

课 程 \_\_\_\_\_\_传感器与检测技术\_\_\_\_\_\_

项 目 \_基于CC2530实现气压传感器MS5611的数据收集

专 业 \_\_\_\_\_\_交通运输工程\_\_\_\_\_\_\_\_\_\_

班 级 \_\_\_\_\_\_交通运输工程2023\_\_\_\_\_\_

学 号 \_\_\_\_\_\_\_\_\_2231801023\_\_\_\_\_\_\_\_\_

姓 名 \_\_\_\_\_\_\_\_\_\_翁泽康\_\_\_\_\_\_\_\_\_\_\_\_

日 期 \_\_\_\_\_\_\_\_2023.12.14\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| 实  验  任  务 | 基于CC2530单片机实现气压传感器MS5611的气压数据和温度数据的读取。 |
| 实  验  报  告  内  容 | 1. 实验材料及介绍：   在你的硬件配置中，你拥有CC2530单片机、MS5611气压传感器、SmartRF04EB烧写器以及一些连接线和串口调试助手程序。以下是对每个组件的文字扩充：  （1）CC2530单片机：  CC2530是德州仪器（TI）推出的一款低功耗无线SoC芯片。它内置了一个低功耗的8051微控制器核心和一个IEEE 802.15.4标准兼容的无线通信模块。该芯片在物联网（IoT）和无线传感器网络（WSN）应用中发挥着关键作用。它支持多种无线通信协议，如Zigbee、Thread、6LoWPAN等，使其在智能家居、智能城市、工业自动化和医疗健康等领域得到广泛应用。  （2）MS5611气压传感器：  MS5611是一款高精度的数字压力传感器，由瑞士公司Measurement Specialties（现在是TE Connectivity）生产。该传感器采用最新的压力传感技术，能够测量大气压力和温度，并输出数字信号。它具有高分辨率、精准度、低功耗和快速响应的特点，适用于气象站、高度计、无人机和气压计等多种气压测量应用。  （3）SmartRF04EB烧写器：  SmartRF04EB是一款由德州仪器提供的烧写器，用于将编写好的程序加载到微控制器中。它提供了连接和烧写CC2530单片机的功能，是开发和调试嵌入式系统的重要工具。通过SmartRF04EB，你能够将设计好的固件烧录到CC2530单片机中，使其具备特定的功能和能力。  （4）连接线和串口调试助手程序：  连接线在整个系统中起着桥梁的作用，将各个组件有机地连接在一起。这些线可能包括用于电源供应、数据传输和通信的线缆。串口调试助手程序是一种能够通过串口与目标设备进行通信的工具。它使你能够实时监测、调试和分析从CC2530单片机读取的数据，为系统调试提供了便利。  2. 实验目的：  本设计采用I2C通信方式，将CC2530微控制器与MS5611数字压力传感器连接，以获取环境的大气压和温度数据。通过精心设计的程序，实现了对传感器内部寄存器值的读取与运算，最终获得准确的大气压力和温度信息。  （1）2C通信过程  首先，CC2530通过I2C总线向MS5611发送相应的指令，以读取传感器内部的寄存器值。这些寄存器中包含了用于计算大气压和温度的关键参数。通过遵循MS5611的通信协议和时序，确保了正确的配置和数据传输。  （2）CC2530与MS5611的协同工作  CC2530作为低功耗无线SoC芯片，搭载了强大的8051微控制器核心和IEEE 802.15.4标准兼容的无线通信模块。其通过I2C与MS5611进行通信，充当主设备，负责发起读取寄存器的请求和获取传感器返回的数据。这种协同工作使得CC2530能够灵活地集成在物联网和无线传感器网络中，广泛应用于智能家居、智能城市、工业自动化和医疗健康等领域。  （3）数据处理与校准  获取到的原始数据可能受到环境因素和传感器特性的影响，因此需要进行一系列的运算和校准操作。在MS5611中，通过读取PROM内存中的工厂数据和补偿参数，进行温度和气压的校准。CC2530通过精确的算法，将这些原始数据转化为准确可靠的大气压和温度数值。  实时监测与上位机通信  最后，通过串口与电脑进行通信，CC2530将获取的大气压和温度数据发送至上位机。上位机采用相应的软件，如XCOM串口调试助手，实时监测并显示CC2530读取的MS5611数据。这种实时监测机制为用户提供了对环境变化的即时响应能力，使得该设计在各种应用场景中都具备了良好的适用性。  3. 实验步骤：  3.1 操作步骤：  在完成这一系列步骤的过程中，你在设计和实施CC2530与MS5611之间的通信系统中取得了关键的进展。以下是对每个步骤的文字扩充：  1. 芯片手册寄存器了解：  在设计之初，通过深入研究CC2530和MS5611的芯片手册，你掌握了寄存器的读取方式和通讯方式。这一步是至关重要的，因为它为后续的通信协议和程序设计提供了基础。详细了解芯片手册确保了对硬件特性的全面理解，有助于编写有效的通信代码。  2. 串口通信调试：  在将系统连接到电脑之前，你通过调试串口通信下位机代码，成功实现了与电脑的串口通信，并能够发送字符串。这一步验证了串口通信的可靠性，为后续的数据传输奠定了基础。成功发送字符串是通信系统正常工作的明确迹象。  3. 烧录软件烧写程序：  利用烧入线将电脑与CC2530相连接，并通过专门的烧录软件将程序烧写进CC2530。这一步是将事先设计好的程序加载到微控制器中，使其能够执行特定的功能。烧录成功后，CC2530已准备好执行与MS5611的通信任务。  4. I2C通信设置：  将MS5611传感器与CC2530进行I2C通信，这是实现大气压和温度数据读取的关键步骤。通过USB转TTL线，将电脑与CC2530连接，为监控和调试提供了便利。I2C通信是一种有效的串行通信协议，使得CC2530能够与MS5611传感器进行高效的数据交换。  5. 上位机观察：  最后，通过上位机-串口调试助手（如XCOM），你能够实时观察CC2530读取的MS5611大气压和温度数据。这种实时监测机制为系统调试和性能评估提供了实用的工具。通过分析上位机显示的数据，你可以验证通信系统的稳定性和准确性，确保它能够在各种环境条件下可靠地运行。  这些步骤的顺利完成表明你对嵌入式系统设计和通信协议有着深入的理解，为进一步的开发和优化奠定了坚实的基础。   1. 关键算法：   这是一段基于CC2530单片机的嵌入式C代码，用于与MS5611气压传感器通信，并通过串口输出温度和气压数据。以下是对该代码的简要解释：  1. 引入头文件：  ```c  #include <iocc2530.h>  #include <stdio.h>  #include "./uart/hal\_uart.h"  #include "./iic/iic.h"  #include "ms5611.h"  ```  这里引入了一些头文件，包括与CC2530相关的头文件、串口通信和I2C通信的头文件，以及MS5611的头文件。  2. 宏定义和变量定义：  ```c  #define uchar unsigned char  #define uint unsigned int  #define uint8 uchar  #define uint16 uint  #define TRUE 1  #define FALSE 0  ```  这里定义了一些宏和数据类型的别名。  3. \*\*LED控制和延时函数：\*\*  ```c  #define LED1 P1\_0  #define LED2 P1\_1  void delay\_us(uint32 delay\_us);  void delay\_ms(uint16 delay\_ms);  ```  定义了LED引脚和延时函数，用于控制LED状态和实现微秒级和毫秒级的延时。  4. LED初始化函数：  ```c  void InitLed(void)  ```  初始化LED引脚的方向。  5. 主函数：  ```c  void main(void)  ```  主函数包含了初始化串口、LED和MS5611，然后进入一个无限循环。在循环中，通过`get\_ms5611\_data()`获取MS5611传感器的温度和气压数据，然后通过串口输出。  6. 串口接收中断函数：  ```c  #pragma vector = URX0\_VECTOR  \_\_interrupt void UART0\_ISR(void)  ```  定义了串口接收中断函数，用于接收字符。  7.处理串口接收的字符和控制LED的状态  ```c  /\*  if (RT\_flag == 1) //接收  {  if (temp != 0)  {  if ((temp != '\r') && (counter < 3)) //'\r‘回车键为结束字符 //最多能接收3个字符  {  receive\_buf[counter++] = temp;  }  else  {  RT\_flag = 3; //进入led设置状态  }  if (counter == 3)  RT\_flag = 3;  temp = 0;  }  }  if (RT\_flag == 3) //led状态设置  {  U0CSR &= ~0x40; //禁止接收  receive\_buf[2] = '\0';  // prints(receive\_buf); prints("\r\n");  if (receive\_buf[0] == '1')  {  if (receive\_buf[1] == '1')  {  LED1 = 0;  prints("led1 on\r\n");  }  else if (receive\_buf[1] == '0')  {  LED1 = 1;  prints("led1 off\r\n");  }  }  else if (receive\_buf[0] == '2')  {  if (receive\_buf[1] == '1')  {  LED2 = 0;  prints("led2 on\r\n");  }  else if (receive\_buf[1] == '0')  {  LED2 = 1;  prints("led2 off\r\n");  }  }  U0CSR |= 0x40; //允许接收  RT\_flag = 1; //恢复到接收状态  counter = 0; //指针归0  }  \*/  ```  这段注释掉的代码主要是处理串口接收的字符和控制LED的状态。具体功能如下：  1. 串口字符接收：  - 当`RT\_flag`为1时，表示正在接收状态。  - 如果接收到的字符不为零，且不是回车符 '\r'，同时计数器 `counter` 小于3，将字符存储在接收缓冲区 `receive\_buf` 中。  - 当接收到回车符或计数器达到3时，将 `RT\_flag` 置为3，表示进入LED状态设置。  2. LED状态设置：  - 当 `RT\_flag` 为3时，表示LED状态设置状态。  - 禁止串口接收。  - 将接收缓冲区的内容按规定格式解析，根据内容控制 LED1 或 LED2 的状态。  - 允许串口接收，将 `RT\_flag` 置为1，恢复到接收状态。  实验结果：  91167331d751b9d87d2caca246f4bef  691dc8ba06c08008aaa8a5129adbcde  1bdd0cae33397def21b8f80bd5ec404  图1 实验结果  由结果可以看出，传感器读出的数据较为准确，且可以实时测量环境气压和温度。 |
| 教  师  评  定 |  |