



# 基于Arduino单片机的MIMU数据采集系统设计

作者：M.Guo

组织：EE.JSSVC

时间：2020.3.10

版本：0.00

*Make things really work and others want them!*

# 目录

<b>1</b>	<b>采集系统硬件设计</b>	<b>1</b>
1.1	Arduino UNO单片机 . . . . .	1
1.2	MIMU模块(GY-9250) . . . . .	2
1.3	蓝牙模块(HC-08) . . . . .	3
1.4	各模块的电路连接 . . . . .	5
1.5	小结 . . . . .	8
<b>2</b>	<b>采集系统软件设计</b>	<b>9</b>
2.1	软件开发平台及配置 . . . . .	9
2.2	算法流程及程序源文件的编排 . . . . .	11
2.3	程序调试与下载 . . . . .	14
2.4	小结 . . . . .	16
<b>3</b>	<b>测试与结论</b>	<b>18</b>
3.1	串口数据发送MIMU传感器数据 . . . . .	18
3.2	上位机接收并显示MIMU传感器数据 . . . . .	19
3.3	小结 . . . . .	22

# 第一章 采集系统硬件设计

## 1.1 Arduino UNO单片机

Arduino 是一款便捷灵活、方便上手的开源电子原型平台，于 2005 年冬季由一个欧洲开发团队开发。Arduino 生态中包括多种开发板(各种型号的 Arduino 板)、模块、扩展板和软件 (Arduino IDE)，能够支持多种操作系统：Windows、Mac 和 Linux。

Arduino 能通过各种各样的传感器来感知环境，通过控制灯光、马达和其他的装置来反馈、影响环境。开发人员开发了简单的函数，还有许多应用库，这样就不用直接去操作寄存器了，使得没有很好的单片机基础的人员也可以使用 Arduino 做出自己想要的东西。

根据微控制器型号的不同，可用的不同Arduino 板如表1.1所示，

表 1.1: 不同微控制器型号下的部分 Arduino 板

板名称	工作电压	时钟	数字 I/O	模拟输入	PWM	UART
基于ATMEGA328微控制器的Arduino板						
Arduino Uno R3	5V	16MHz	14	6	6	1
Arduino Uno R3 SMD	5V	16MHz	14	6	6	1
Arduino mini 05	5V	16MHz	14	8	6	1
Red Board	5V	16MHz	14	6	6	1
Arduino Ethernet	5V	16MHz	14	6	6	1
基于ATMEGA32u4微控制器的Arduino板卡						
Arduino Leonardo	5V	16MHz	20	12	7	1
Pro micro 5V/16MHz	5V	16MHz	14	6	6	1
基于ATMEGA2560微控制器的Arduino板卡						
Arduino Mega 2560 R3	5V	16MHz	54	16	14	4
Mega Pro Mini 3.3V	3.3V	8MHz	54	16	14	4
基于AT91SAM3X8E微控制器的Arduino板卡						
Arduino Mega 2560 R3	3.3V	84MHz	54	12	12	4

不同型号的 Arduino 板拥有不同的性能参数，其中 Arduino Uno R3 是最适合入门且功能齐全使用量最多的 Arduino 开发板，如图 1.1 所示。

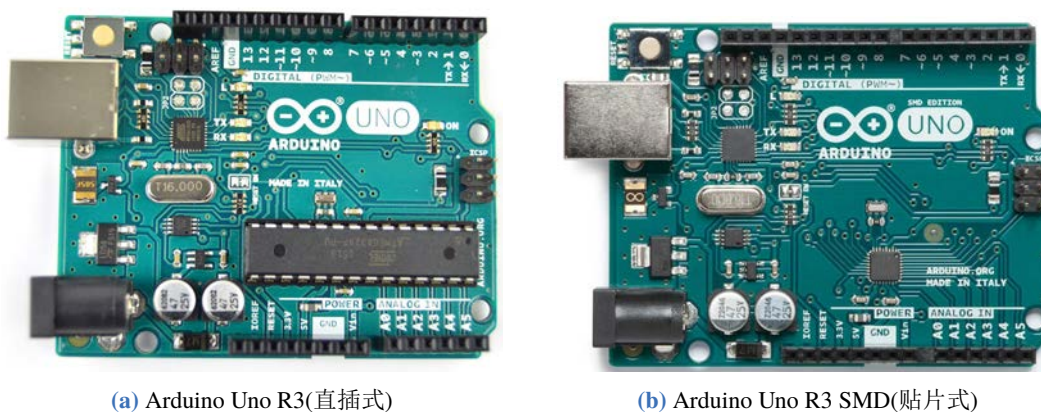


图 1.1: Arduino UNO R3 模块

本文选用的单片机即为 Arduino UNO R3 开发板，如前所述，其本质是基于 ATmega328P 微控制器的单片机开发板。它有 14 个数字输入/输出引脚(其中 6 个可用作 PWM 输出)，6 个模拟输入脚，一个 16MHz 晶体振荡器，一个 USB 接口，一个 DC 电源插座，一个 ICSP header 和一个复位按钮。只需使用 USB 线将其连接到计算机，或者使用 AC-to-DC 适配器或电池为其供电即可开始使用。该开发板的其他参数如下表 1.2 所示，

表 1.2: Arduino Uno R3 的部分参数

工作电压	输入电压(推荐)	输入电压(限制)	I/O 电流	Flash	SRAM	EEPROM
5V	7~12V	6~20V	20mA	32KB	2KB	1KM

Arduino UNO 板拥有非常丰富的资源，但在本文的开发过程中主要使用该开发板的 I2C 通信以及 TTL 串口通信的相关资源，如如表 1.3 所示。

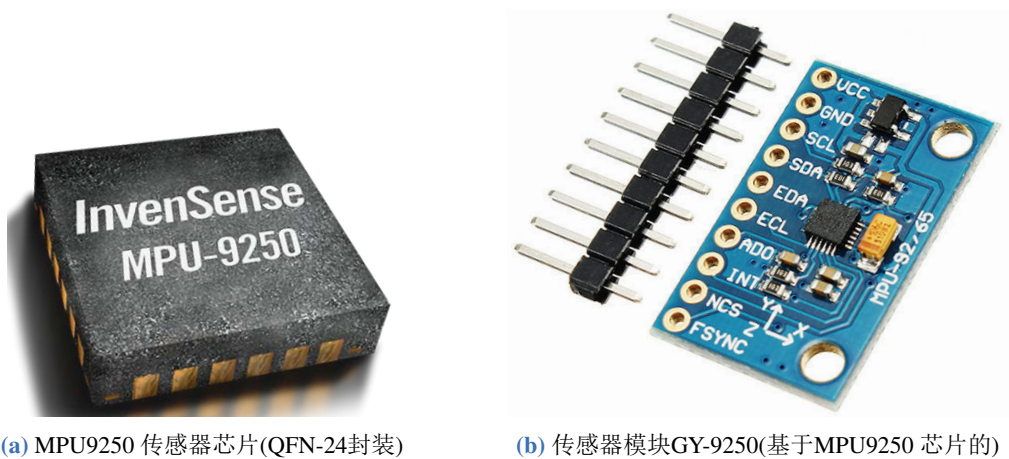
表 1.3: Arduino 板 I2C 和 UART 通信的管脚定义

资源类型	I2C		UART	
通信引脚	SDA	SCL	Rx	Tx
管脚	A4	A5	pin0	pin1

## 1.2 MIMU模块(GY-9250)

MPU9250 是一种非常流行的空间运动传感器芯片(图 1.2a)，可以获取 3 轴角速度/转角速度/磁场分量信息。MPU9250 是一个 QFN 封装的复合芯片，它由 2 部分组成。一组是具有 16 位 AD 输出的三轴加速度计和具有 16 位 AD 输出的三轴陀螺仪，另一组是 AKM 公司的 AK 8963 三轴磁力计，具有 6 位磁力计 AD 输出。该测量模块可实现精密的慢速和快速运动跟踪，提供给客户全量程的可编程陀螺仪参数选择( $\pm 250$ ,  $\pm 500$ ,  $\pm 1000$ ,  $\pm 2000^\circ/\text{s}$ )，可编程的加速度参数选择  $\pm 2g$ ,  $\pm 4g$ ,  $\pm 8g$ ,  $\pm 16g$ ，以及最大磁力计可达到  $\pm 4800\mu\text{T}$ 。





(a) MPU9250 传感器芯片(QFN-24封装) (b) 传感器模块GY-9250(基于MPU9250 芯片的)

图 1.2: MPU9250 传感器及测量模块

本文选用图1.2b中的惯性测量模块GY-9250，该模块是基于芯片 MPU9250，采用3~5V电源供电，通信方式为标准I2C/SPI通信协议，适合快速搭建电路，其引脚说明如表1.4。

表 1.4: GY-9250GY-9250引脚说明

引脚符号	内容
VCC	电源正(3~5V)
GND	地
SCL	I2C串行时钟线/SPI串行时钟端口
SDA	I2C串行数据线/SPI串行数据输入
EDA	连接其他I2C设备的主机数据口
ECL	给I2C设备提供主时钟
AD0/SD0	I2C器件地址选择器/SPI串行数据输出
INT	中断引脚(可以用于唤醒主芯片)
NCS	片选
FSYNC	数字同步输入帧，不用请接地

### 1.3 蓝牙模块(HC-08)

为了能够将采集到的 MIMU 数据以无线的方式实时地传输给上位机，本文选用了一款无线蓝牙模块，型号为 HC-08。

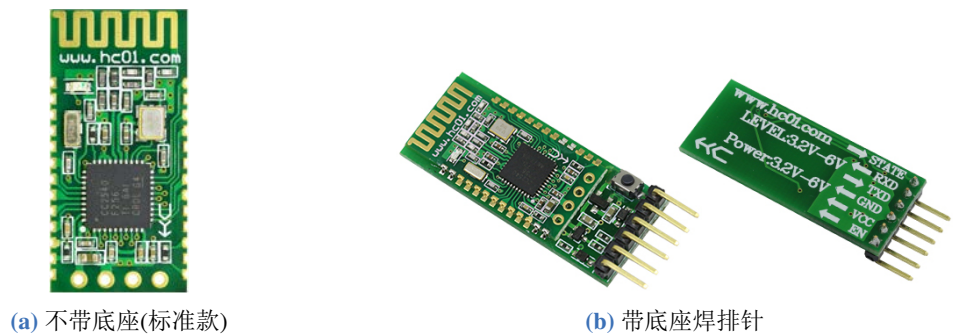


图 1.3: 蓝牙无线串口透传模块

图1.3为低功耗蓝牙模块，也称为 BLE 模块(Bluetooth Low Energy Module)，型号为 HC-08，最大的特点是成本和功耗的降低，应用于实时性要求较高的产品。图中左侧模块是不带底座的标准款，右侧为带底座焊排针的版本(本文中使用的版本)，其各引脚定义如表1.5，

表 1.5: HC 08 蓝牙模块管脚定义

KEY	VCC	GND	TXD	RXD	STATE
清除主机 配对记忆	电源脚 (3.2~6 V)	公共地	UART输出 (TTL电平)	UART输入 (TTL电平)	蓝牙连接状态 (高:连接; 低:未连接)

HC-08 蓝牙模块为主从一体，使用4.0 协议，可以设置为一个主机一个从机，建立一对一连接通信。本文中，图1.3中的蓝牙模块负责将采集到的 MIMU 数据发送给上位机，因为上位机是PC机，因此 HC-08 蓝牙模块需借助 HC-08-USB 蓝牙虚拟串口与电脑相连，图1.4即为 USB 转蓝牙串口模块。

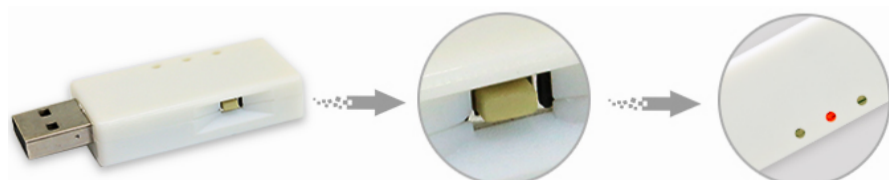


图 1.4: USB 转蓝牙串口模块

USB 转蓝牙串口模块侧面有一个按键KEY，同时内置三色状态灯，各自的含义如下：

- **黄灯：** 数据指示灯
  - ①**闪烁：** 模块内部串口有数据通过时，比如发送AT指令或者串口透传；
  - ②**熄灭：** 无数据通过；
- **红灯：** 多功能指示灯
  - ①**按键指示灯：** 当用户按下模块侧面的按键KEY，红灯会亮起(高亮度，按键按下多久，红灯就亮多久)；
  - ②**USB 挂起指示灯：** 模块装好驱动之前，红灯会长亮(普通亮度)，装好驱动后插入PC的USB端口，红灯不亮，如果没有数据通信(例如没有打开串口助手软件)，

几秒后红灯会亮起。此时，打开串口助手，并开启模块对应的端口，红灯会熄灭。关掉串口助手，10秒钟左右，红灯会再次亮起。

③**蓝牙模块复位指示灯**：HC-08蓝牙模块每次复位，红灯会亮 10ms 左右(HC-08的复位时间)。此功能可以很方便用来检测 HC-08 模块哪些 AT 指令是需要复位模块才能生效的。

• **蓝灯：** 蓝牙模块指示灯

①**慢闪**：无配对记忆模块未连接状态；

②**快闪**：有配对记忆模块未连接状态；

③**长亮**：连接配对成功

- **按键KEY**：对应HC-08模块34引脚KEY，按下KEY键时，红灯会亮。HC-08-USB上电后不连接即进入 AT 指令模式。按 KEY 键一下，可清楚模块配对记忆(默认为主机模式)。

经测试，在默认设置的情况下，图1.3和图1.4中的模块在上电之后可以自动完成连接。

## 1.4 各模块的电路连接

数据采集系统中各子模块的连接关系如示意图1.5所示。其中，锂电池给 Arduino UNO 单片机供电，单片机再通过 5V 引脚分别给 MPU9250 和蓝牙模块 HC-08 供电。此外，Arduino UNO 单片机通过 SCL 和 SDA 与 MPU9250 实现 I2C 通信，

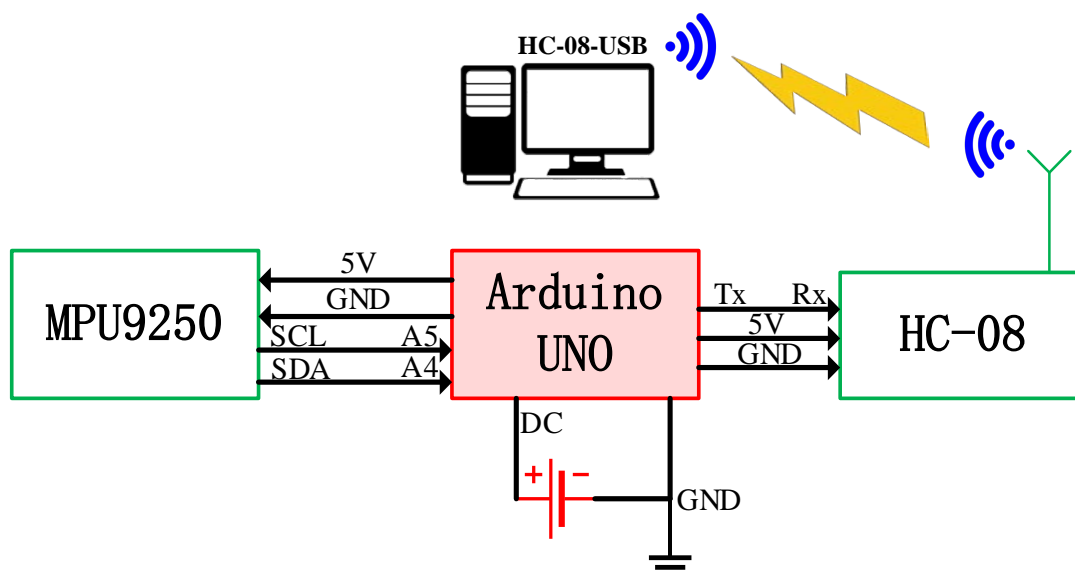


图 1.5: 各子模块的硬件连接示意图

上图仅为连接关系的示意图，详细的电路原理图如图1.6和1.7所示。

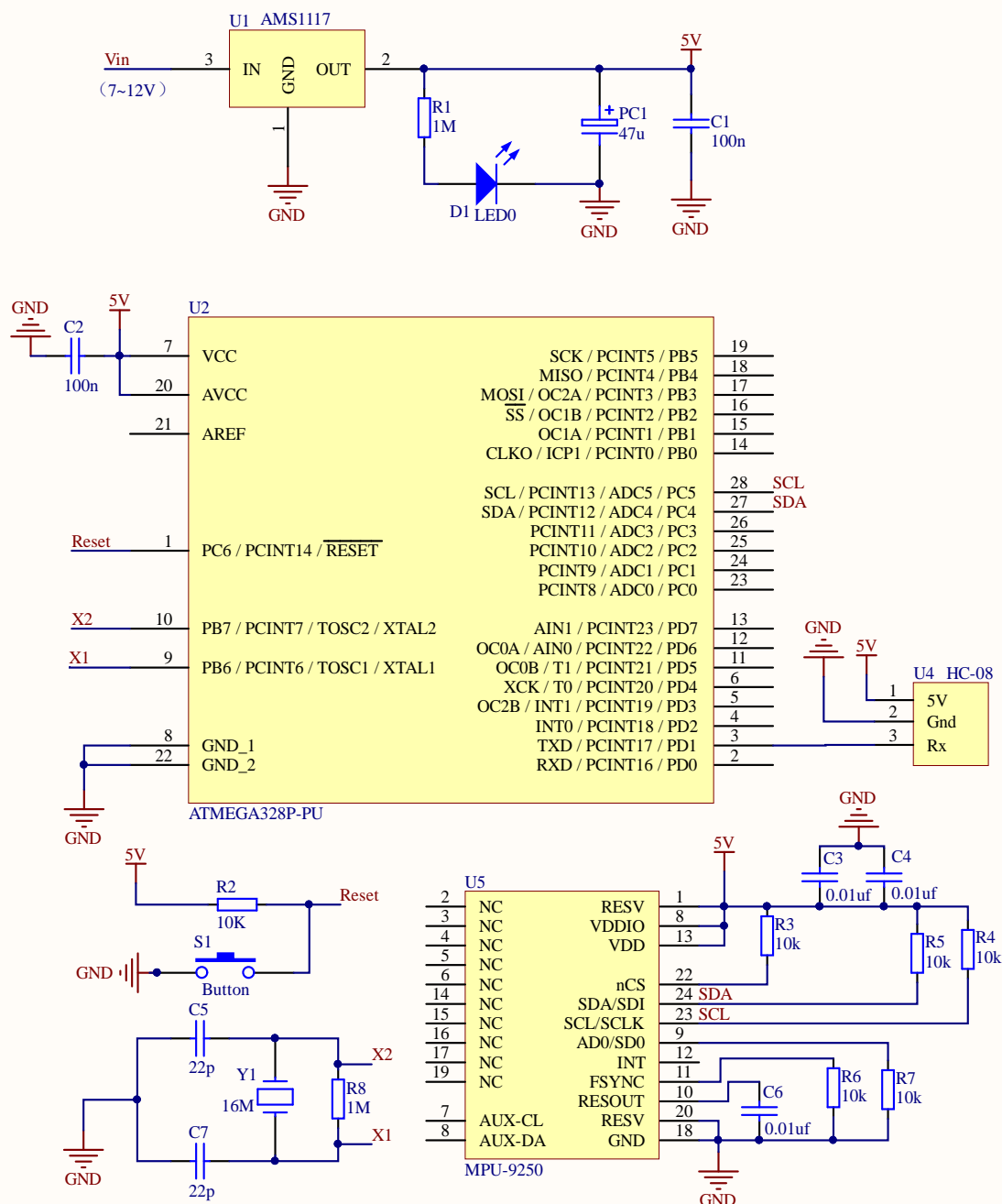


图 1.7: 各子模块的硬件连接PCB原理图

上图中，MPU9250 模块的引脚 VCC、GND、SCL 和 SDA 分别连接 Arduino UNO 的 5V、GND、A4 和 A5 端。Arduino UNO 通过 A4、A5 端分别与惯性测量模块 MPU9250 的 SDA 和 SCL 端口连接。Arduino UNO 通过 TXD(PD1) 端连接蓝牙模块 HC-08 的 RX 端，将采集到的数据经过预处理之后，发送给上位机。



单片机 Arduino UNO、MIMU 传感器 MPU9250、蓝牙通信模块 HC-08 完成电路连接之后的硬件实物如图1.8所示，

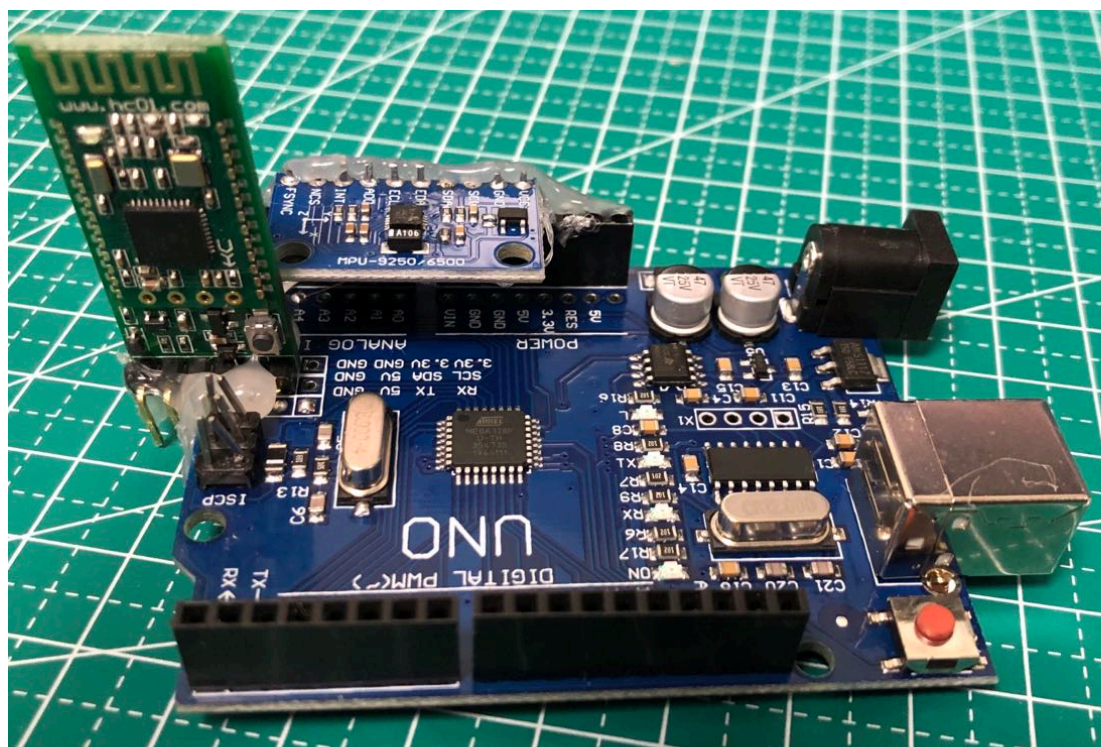


图 1.8: 硬件连接图(未通电)

数据采集系统连接 7.4V 锂电池的效果如图1.9所示，

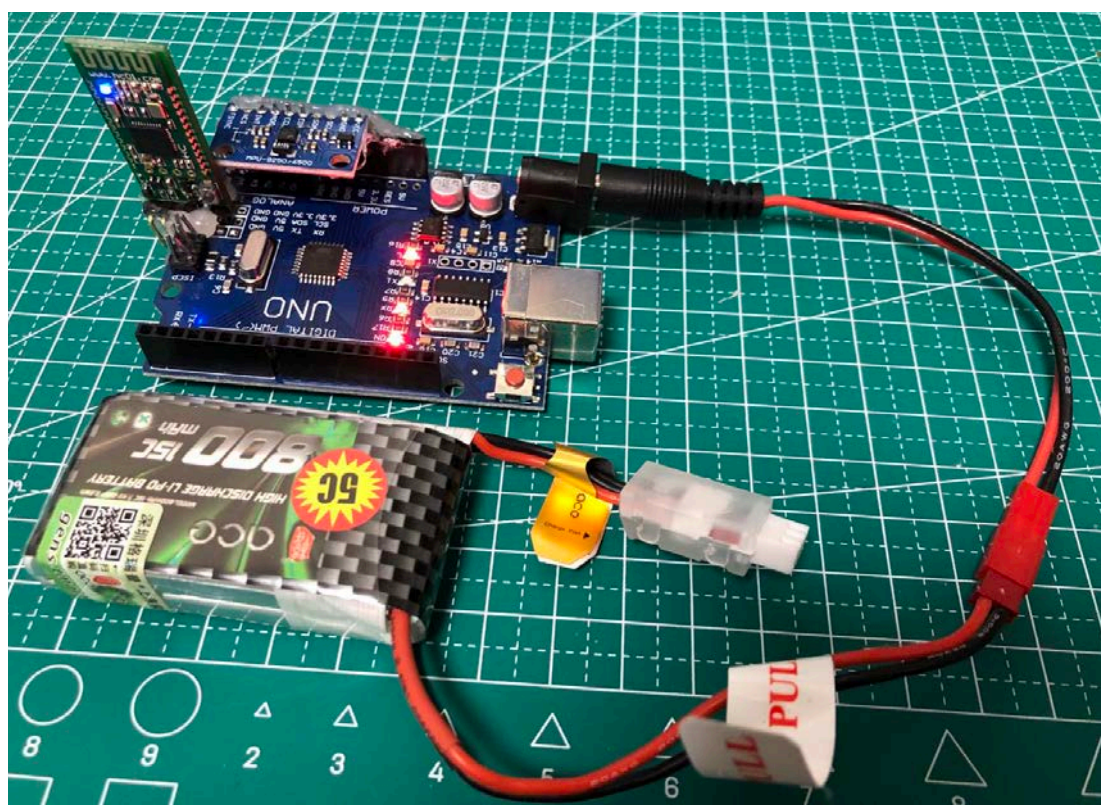


图 1.9: 硬件连接图(通电)

## 1.5 小结

本章对所要开发的 MIMU 数据采集系统的硬件结构进行了分析，内容包括：

- (1) 介绍了 Arduino UNO 单片机、MIMU 模块 (GY-9250) 和蓝牙模块 (HC-08) 的基本知识，各自的功能、引脚定义等。
- (2) 详细阐述了系统中各子模块的电路连接方式，并完成 MIMU 数据采集系统的搭建。

## 第二章 采集系统软件设计

### 2.1 软件开发平台及配置

Arduino 系列单片机均的软件开发平台称为 Arduino IDE(集成开发环境)，用于将计算机代码写入并上传到物理板，该开发平台界面如图2.1所示，Arduino IDE 使用 C++ 的简化版本，使其更容易学习编程。

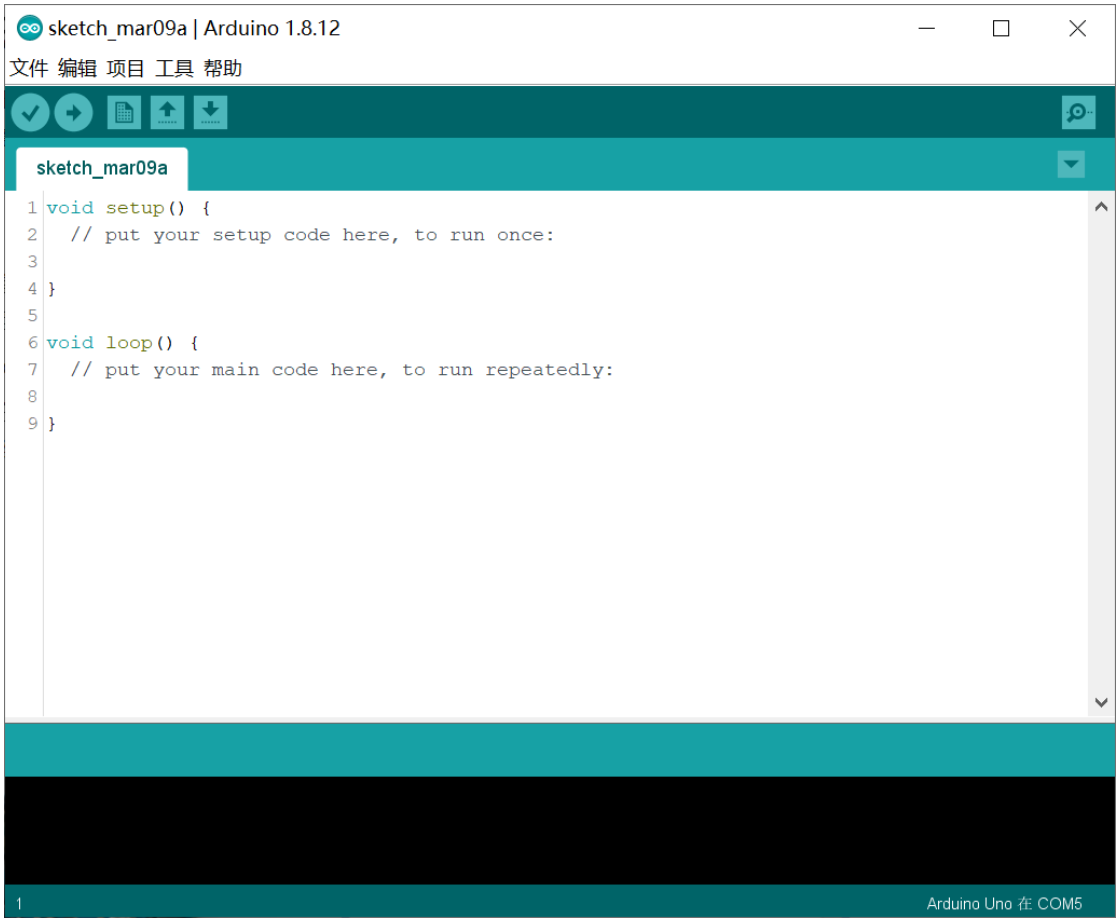


图 2.1: Arduino 单片机的软件开发平台

通过 USB 将 PC 和 Arduino 连接之后，安装好驱动软件，在软件开发之前，需要进行配置。首先是选择对应的主板，单击“工具”选型，选择本次开发所对应的开发板类型“Arduino UNO”，如图2.2所示。

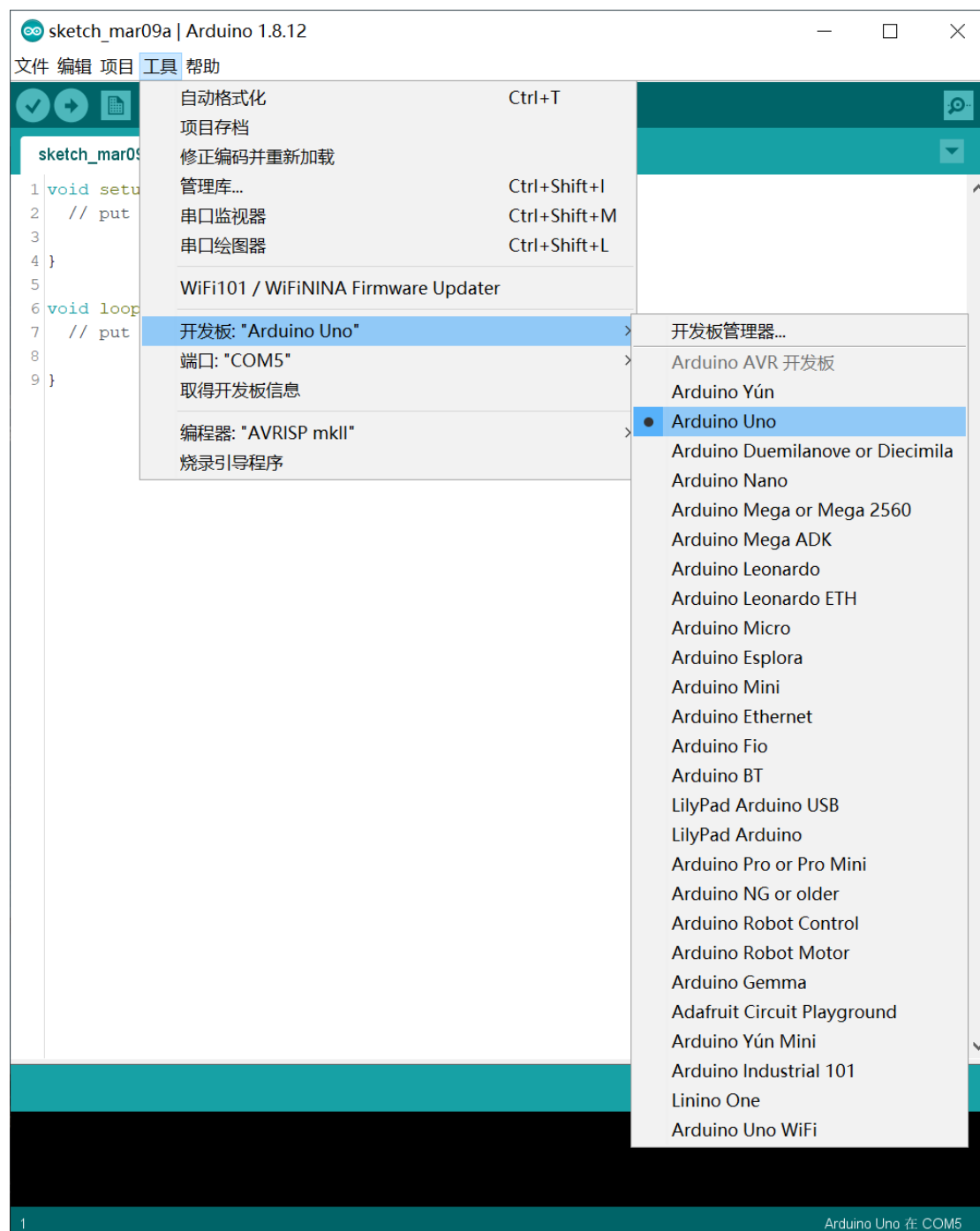


图 2.2: 选择对应的开发板类型

其次是选择对应的串口号，单击“工具”选型，选择本次开发所对应的串行端口“COM5”，如图2.3所示。

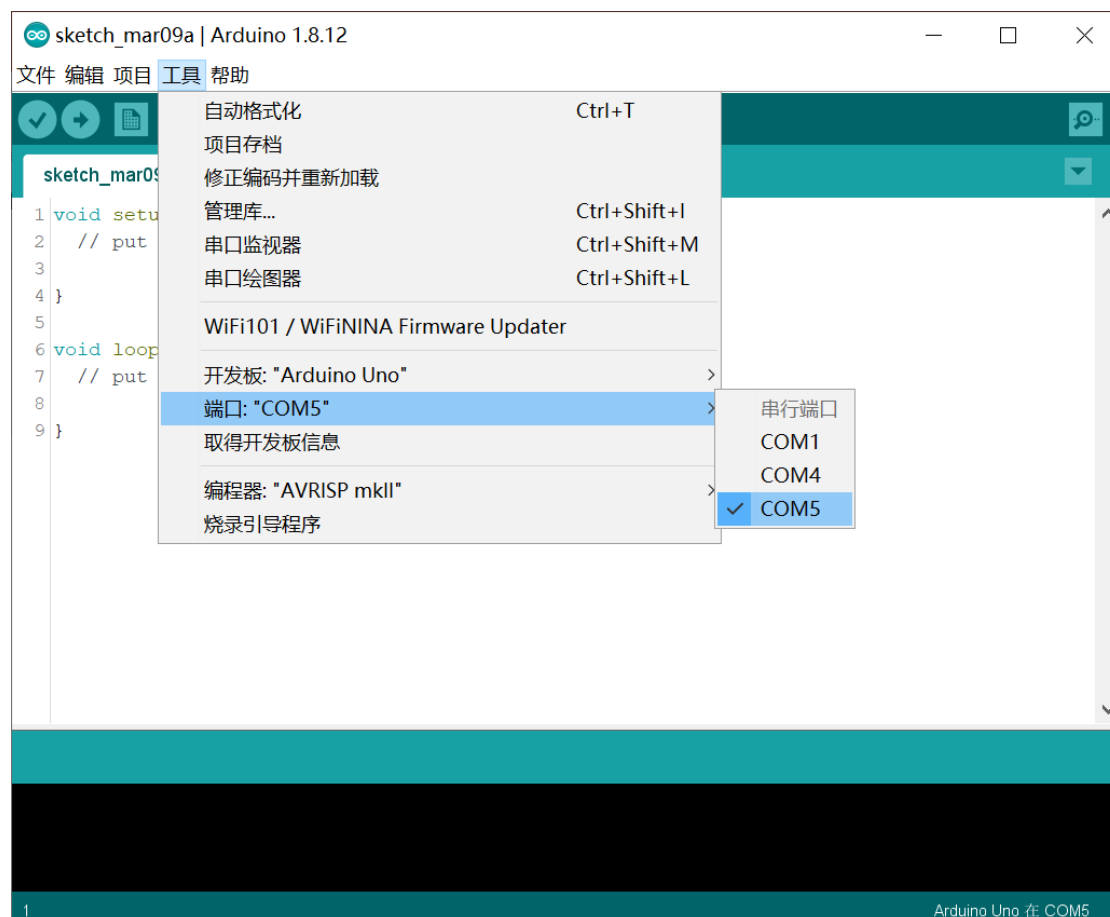


图 2.3: 选择对应的串行端口号

完成前面两步 IDE 的配置工作之后，即可新建项目，开始软件的编写。

## 2.2 算法流程及程序源文件的编排

Arduino IDE 中算法的流程如图2.4所示，其中初始化部分将声明一些全局变量，`void setup()` 函数在单片机上电之后只运行一次，调用函数库中的子函数，实现单片机相关资源的配置工作。`void loop()` 函数将核心循环运行，它将调用子函数读取传感器数据，并发送给上位机。



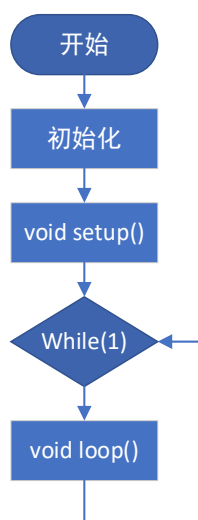


图 2.4: Arudino IDE 算法流程框架

在 Arduino IDE 中新建一个项目，另存为“Arduino Code MPU 9250.ino”，并将其保存在一个同名文件内。接着，在该文  
“MPU9250.cpp”和“MPU 9250.h”  
图 2.5 所示。其中前两个源文件是 MPU  
配置和数据读取，后两个源文件是数据  
包。这四个源文件<sup>1</sup>均为已事先开发完成



图 2.5: 程序源文件的导入

在“Arduino\_Code\_MPU9250.ino”源文件包含三个部分，第 I、II 两部分只运行一次，负责程序的初始化工作，第 III 部分一直循环运行。第 I 个部分如下所示，将刚刚导入的两个 .h 头文件包含进来，同时声明相关数据。

Listing 2.1: Arduino\_Code\_MPU9250.ino 源文件(第 I 部分)

```

1 #include "MPU9250.h"
2 #include "DataProcess.h"
3
4 /* an MPU9250 object with the
5 MPU-9250 sensor on I2C bus 0
  
```

<sup>1</sup>Github链接: <https://github.com/MinhuanGuo/MPU9250-AHRS-Arduino.git>

```

6 with address 0x68 */
7 MPU9250 IMU(Wire,0x68);
8 int status;
9 // Static data:
10 static float accX,accY,accZ;
11 static float gyrX,gyrY,gyrZ;
12 static float magX,magY,magZ;
13 static float tempreture;
14 static uchar decodePacket[20] = {0x09,
15     0x01,0x12,0x03,0x84,0xF3,0x44,
16     0x01,0x8A,0x13,0x04,0x03,0x34,
17     0x31,0x02,0x03,0x56,0x23,0x17,
18     0x00};
19 static int de_Length = 20;
20 static int en_Length = 23;
21 static uchar encodePacket[23];
22 const float _r2d = 180.0f/3.14159265359f;
23 static int sensorValue = 0;
24 static float voltage = 0.0;

```

第 II 部分如下所示，功能是初始化串口，同时建立与 MPU9250 的数据通讯。

**Listing 2.2:** Arduino\_Code\_MPU9250.ino 源文件(第 II 部分)

```

25 void setup() {
26     // serial to display data
27     Serial.begin(115200);
28     while(!Serial) {}
29
30     // start communication with IMU
31     status = IMU.begin();
32
33     if (status < 0) {
34         Serial.println("IMU_initialization_unsuccessful");
35         Serial.println("Check_IMU_wiring_or_try_cycling_power");
36         Serial.print("Status:");
37         Serial.println(status);
38         while(1) {}
39     }
40     /***** Configuration of MPU9250 *****/
41     // setting the accelerometer full scale range to +/-4G
42     IMU.setAccelRange(MPU9250::ACCEL_RANGE_4G);
43     // setting the gyroscope full scale range to +/-2000 deg/s
44     IMU.setGyroRange(MPU9250::GYRO_RANGE_2000DPS);
45     // setting DLPF bandwidth to 184 Hz

```

```

46 IMU.setDlpfBandwidth(MPU9250::DLPF_BANDWIDTH_184HZ);
47 // setting SRD to 1 for a 500 Hz update rate
48 IMU.setSrd(1);
49 /*****
50 /*****
51 //Initial Process
52 InsertChecksum(decodePacket,20);
53 EncodePacket(decodePacket,de_Length, encodePacket,en_Length);
54 }

```

第 III 部分为循环工作，第 62 行处将读取 MPU9250 的数据，第 63~71 行将原始数据转变成各轴的数据，并完成数据格式的转换，第 73 行到 80 行将数据再次打包通过串口发送出去，第 82 行为延时功能，保证整个循环部分的程序运行频率在 200 Hz 左右。

**Listing 2.3:** Arduino\_Code\_MPU9250.ino 源文件(第 III 部分)

```

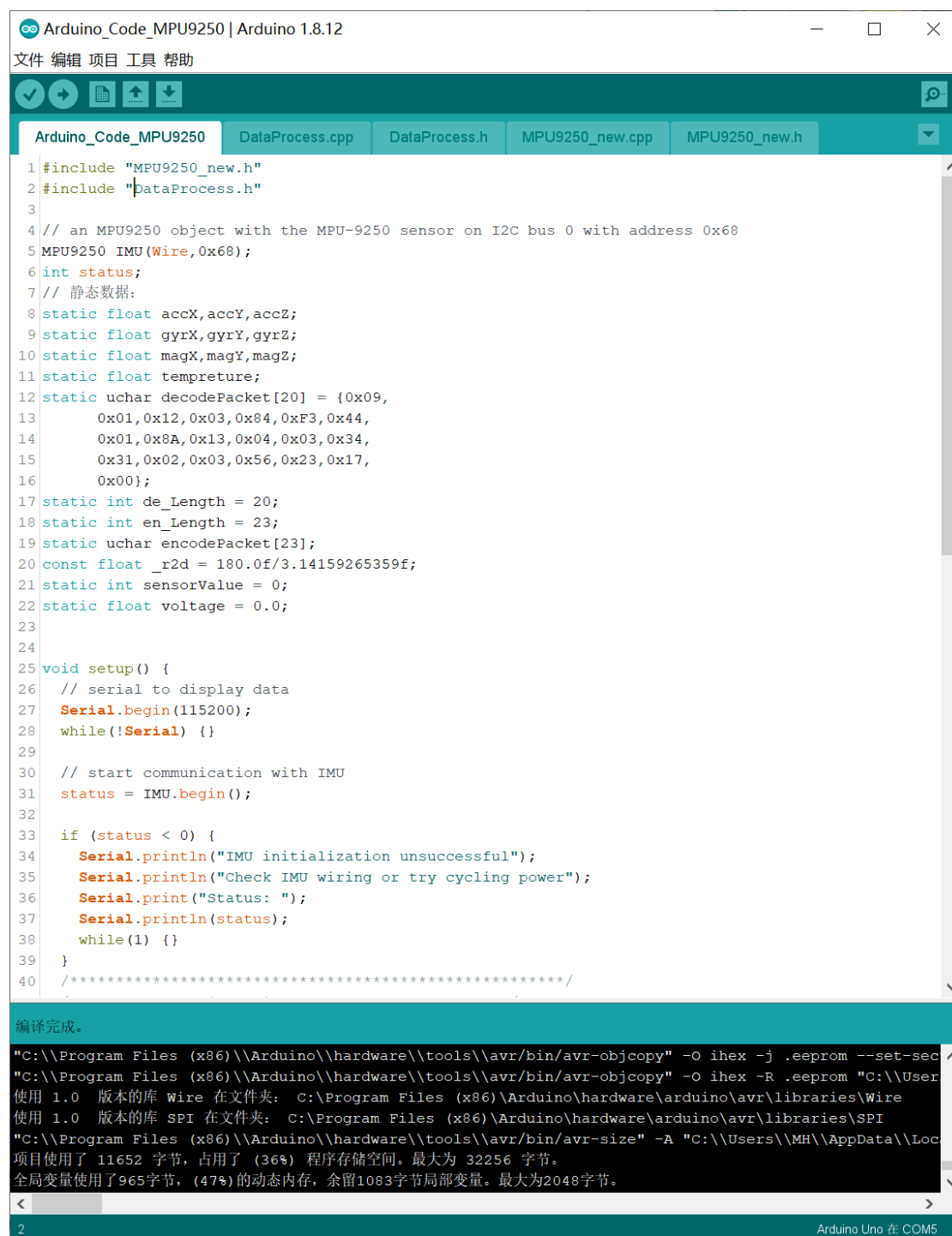
60 void loop()
61 {
62     IMU.readSensor();// read the sensor
63     accX = 1/9.0*IMU.getAccelX_mss();//change unit to g
64     accY = 1/9.0*IMU.getAccelY_mss();
65     accZ = -1/9.0*IMU.getAccelZ_mss();
66     gyrX = _r2d*IMU.getGyroY_rads();//change unit to deg/s
67     gyrY = _r2d*IMU.getGyroX_rads();//Change direction of Gx and Gy
68     gyrZ = -1.0f*_r2d*IMU.getGyroZ_rads();
69     magX = IMU.getMagX_uT();
70     magY = IMU.getMagY_uT();
71     magZ = IMU.getMagZ_uT();
72
73     DataIntoPacket(decodePacket,
74                     gyrX,gyrY,gyrZ,
75                     accX,accY,accZ,
76                     magX,magY,magZ);
77
78     InsertChecksum(decodePacket,20);
79     EncodePacket(decodePacket,de_Length, encodePacket,en_Length);
80     Serial.write(encodePacket,23);//Send from serial port
81
82     delay(3); //Trying to make it work at 200Hz
83 }

```


## 2.3 程序调试与下载

源程序开发过程中，可通过单击 IDE 中的编译按钮(✓)来检查是否有错误，图2.6为

编译完成的截图。



**图 2.6:** 程序编译过程截图

编译完成之后，可通过单击 IDE 中的编译按钮()将编译后的文件上传至 Arduino 单片机中，图2.7即为上传成功的截图。



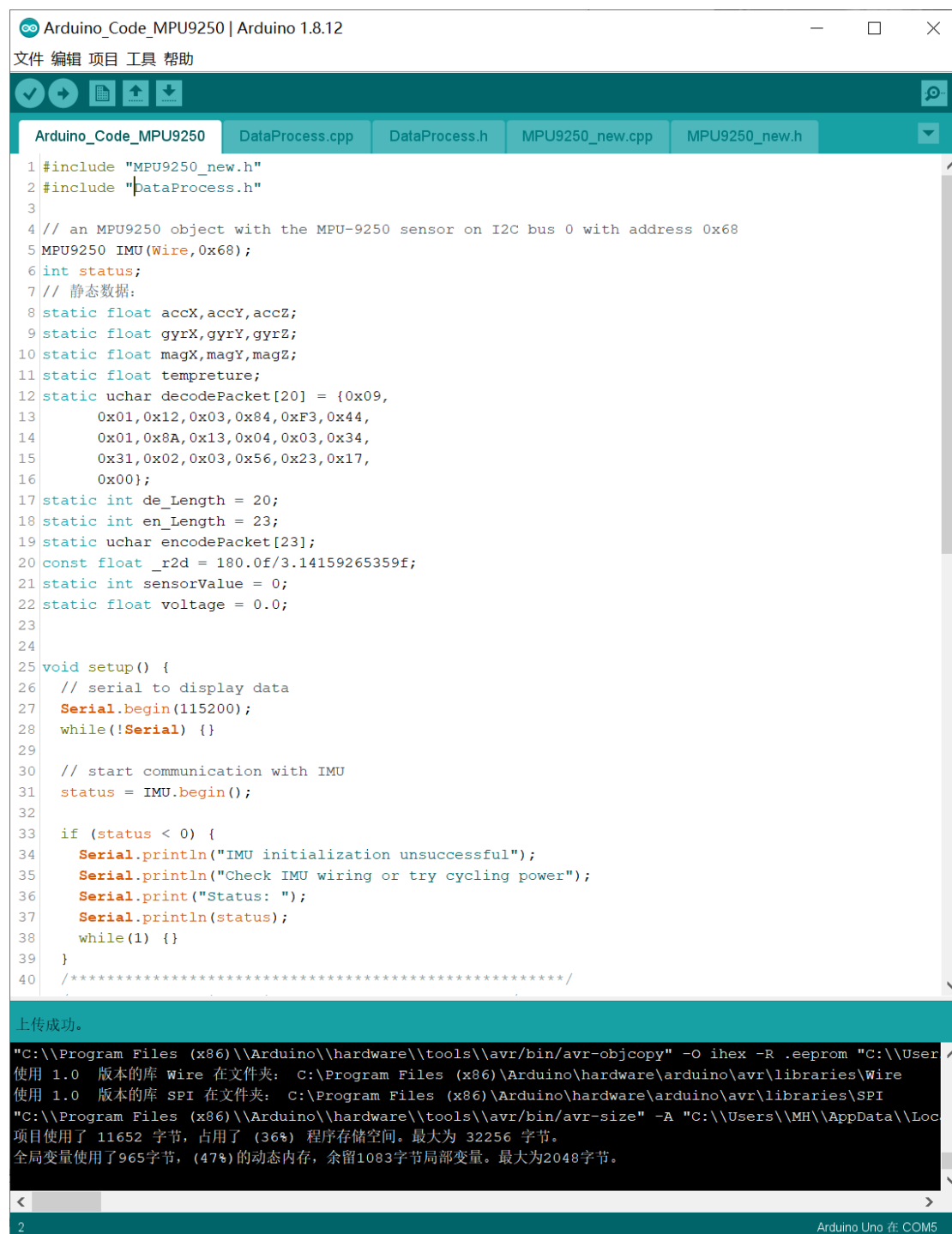


图 2.7: 编译后的文件上传至数据采集系统截图

## 2.4 小结

本章对前面所设计的 MIMU 数据采集系统进行了相关配置以及软件编排, 内容包括:

- (1) 通过 AT 命令, 配置蓝牙模块 HC-08 的波特率。
- (2) 开发软件, 通过 Arduino 的 I2C 接口读取 MIMU 传感器的数据。



- (3) 开发软件，通过 Arduino 的 UART 串口发送数据。

## 第三章 测试与结论

### 3.1 串口数据发送MIMU传感器数据

数据采集系统上电之后，使用 DS213 迷你示波器的通道 A 读取蓝牙模块数据发送端的波形。具体来说，底线夹子连接数据采集模块中的 GND，通道 A 的探头连接到电路被测点，也就是蓝牙模块 HC-08 的 RX 引脚，该引脚接收 Arduino UNO 通过串口发送来的数据，并将该数据以无线的方式发送给插入 PC 的 HC-USB 模块。测试过程如图 3.1 所示。

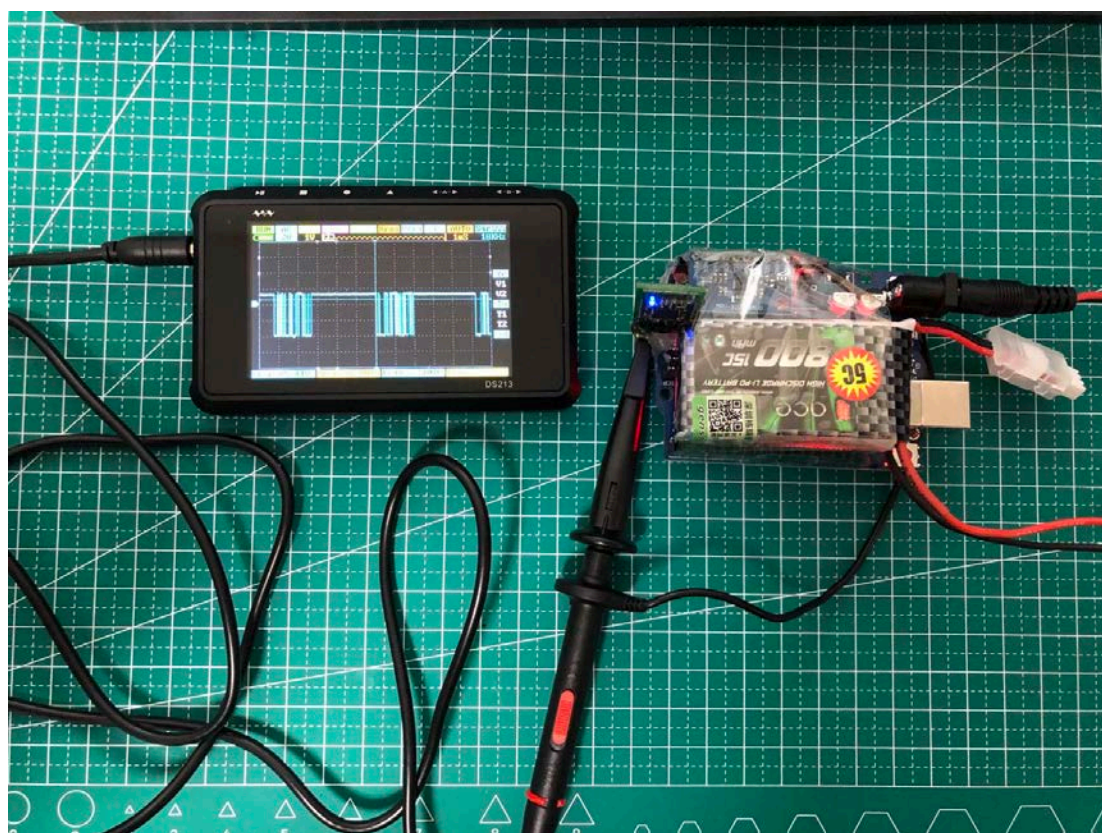


图 3.1: 使用示波器读取串口数据发送端波形

图 3.2 为蓝牙模块 RX 引脚通过示波器测量的截图，其中纵轴表示 RX 引脚的电压值，每小格代表 2 V 的电压，因此 RX 引脚电压上下变化的幅值为 5V 左右。横轴表示时间，每小格代表 1 ms 的时间间隔。从图中可以看出，当串口没有数据经过时，RX 引脚的电压持续保持高电位。当串口中数据经过时，RX 引脚的电压将会产生上下波动，每一次发送数据所需时间大约为 2 小格，即 2 ms，相邻两次发送数据的时间间隔大约为 5 小格，即 5 ms。

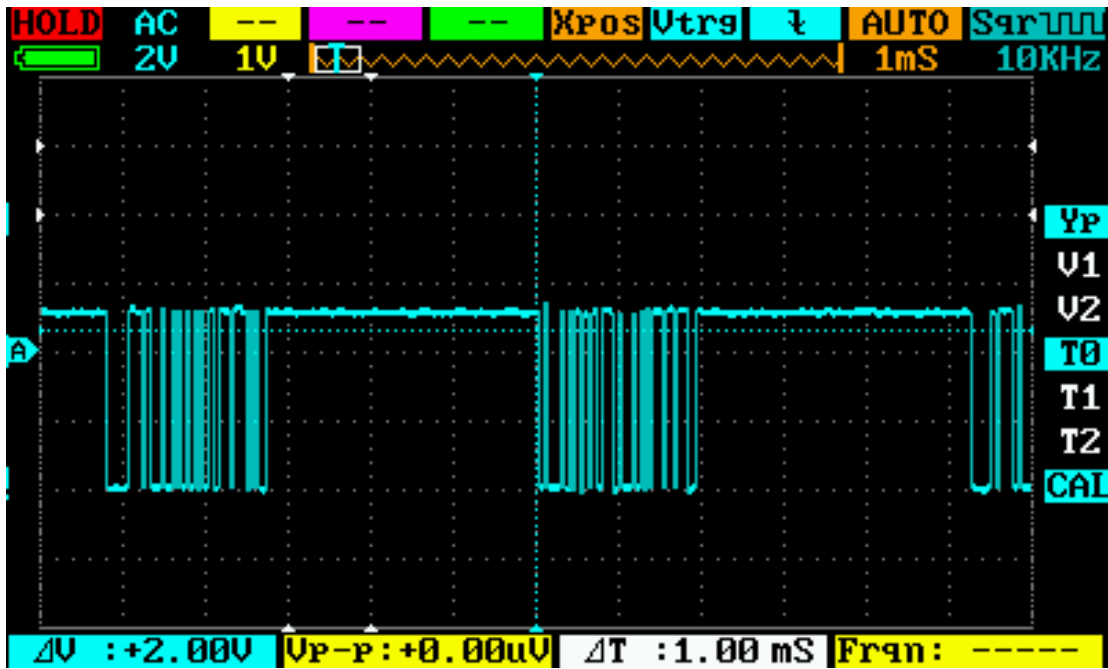


图 3.2: 串口发送数据示波器截图(TTL 电平)

由此可知，设计的 MIMU 数据采集模块已经能够将采集到的数据按照预设的频率发送出去。

### 3.2 上位机接收并显示MIMU传感器数据

为了进一步检验发送出去的 MIMU 数据是否正确，还需要从上位机的角度进行测试。作为上位机，PC 安装有基于 C# 开发的 MIMU 数据监控软件“IMU Monitor”，在接收到 MIMU 数据包之后可以负责数据包的解码、计算和显示。

图3.3为数据监控软件“IMU Monitor”的主界面，打开该软件之后，选择 HC-08-USB 对应的端号 (COM4)，选择波特率 115200，单机“Connect”按钮，即可实现上位机与数据采集模块的无线蓝牙通信，此时按钮“Connect”变为“Connected”。

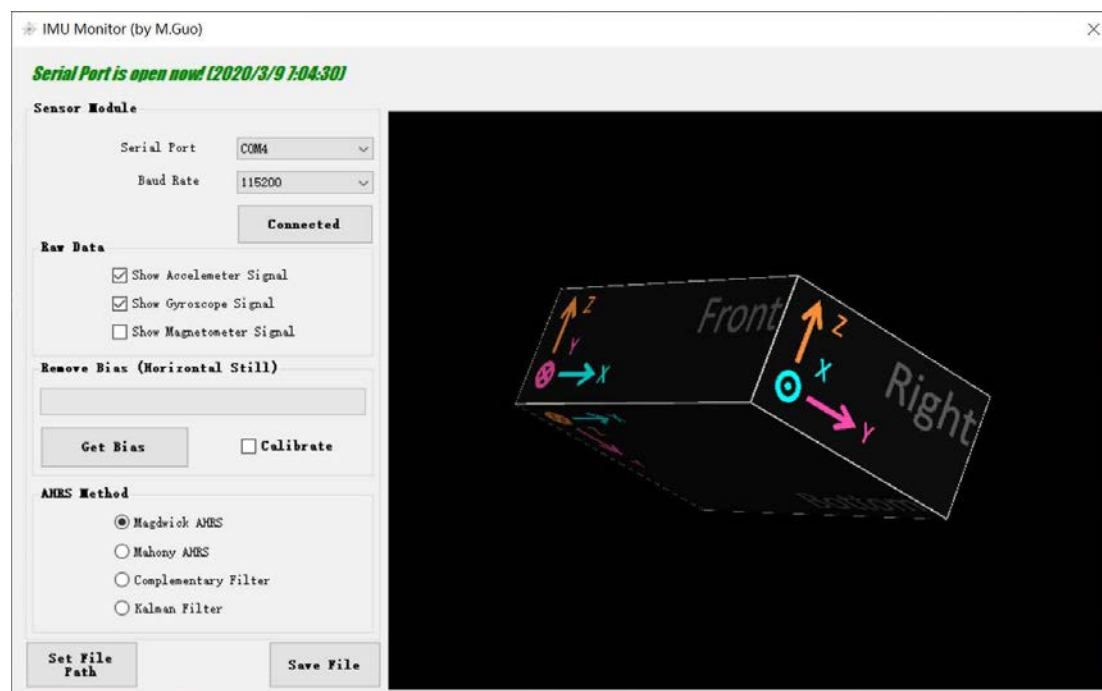


图 3.3: 数据监控软件“IMU Monitor”的主界面

上图中，“IMU Monitor”主界面的右半区为三维姿态展示区，当选中左侧下方“AHRS Method”区域中的“Magdwick AHRS”后，三维姿态展示区中的立方体将依据内置的算法，实时随着 MIMU 数据采集模块的转动而转动。

同时勾选主界面 Raw Data 区域中的“Show Accelerometer Signal”和“Show Gyroscope Signal”，可以打开两个新的窗口，分别通过波形曲线的方式，实时显示三轴加速度计的数据和三轴陀螺仪的数据，如图3.4和图3.5所示。

其中，图3.4为加速度计的三轴输出，“红-绿-黄”三种不同颜色的曲线分别表示加速度计的“x-y-z”轴的数据，单位为 g。

其中，图3.5为陀螺仪的三轴输出，“绿-红-黄”三种不同颜色的曲线分别表示陀螺仪的绕“x-y-z”轴的角速度数据，单位为 deg/s。

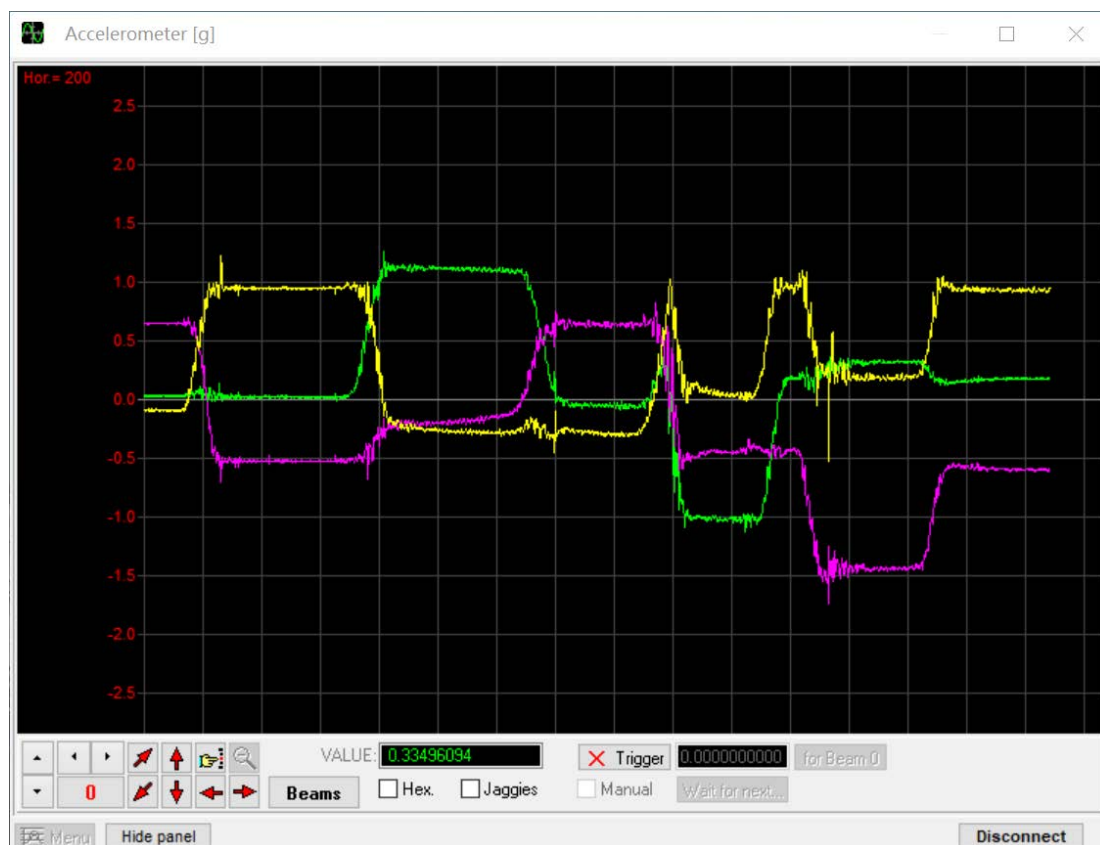


图 3.4: 数据监控软件“IMU Monitor”——三轴加速度计数据曲线

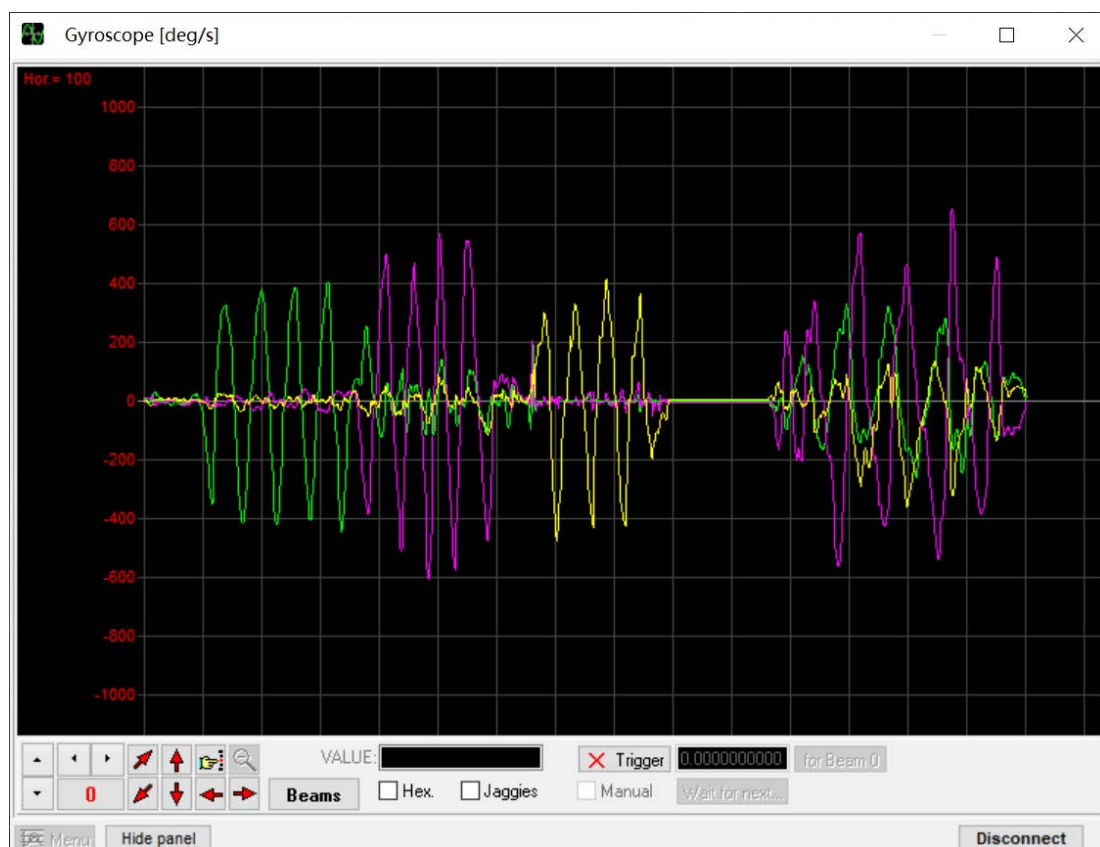


图 3.5: 数据监控软件“IMU Monitor”——三轴陀螺仪的数据曲线



### 3.3 小结

本章对前面所设计的基于 Arduino 单片机的 MIMU 数据采集系统进行了测试，内容包括：

- (1) 用示波器读取 MIMU 数据采集系统串口发送端的波形，通过对波形的分析可知所设计的 MIMU 数据采集系统能够按照给定的设置向上位机发送数据。
- (2) 借助 HC-08-USB 模块，通过在 PC 上已经开发好的上位机软件“IMU Monitor”来接收数据采集系统发送来的数据，结果显示数据收发通讯正常，并且能够通过预置的算法实时显示出传感器的各轴数据波形。