

US-100 超声波测距模块在 Arduino 平台上的使用说明

1. 系统介绍

1.1 US-100 介绍

US-100 超声波测距模块可实现 2cm~4.5m 的非接触测距功能,拥有 2.4~5.5V 的宽电压输入范围,静态功耗低于 2mA,自带温度传感器对测距结果进行校正,同时具有 GPIO,串口(波特率 9600bps)等多种通信方式,内带看门狗,工作稳定可靠。通过模块背部的 2Pin 跳线来选择合适的模式,当拔掉跳线帽时,表示工作在 GPIO 模式下;当插上跳线帽时,表示工作在串口模式下,如图 1.1 所示。

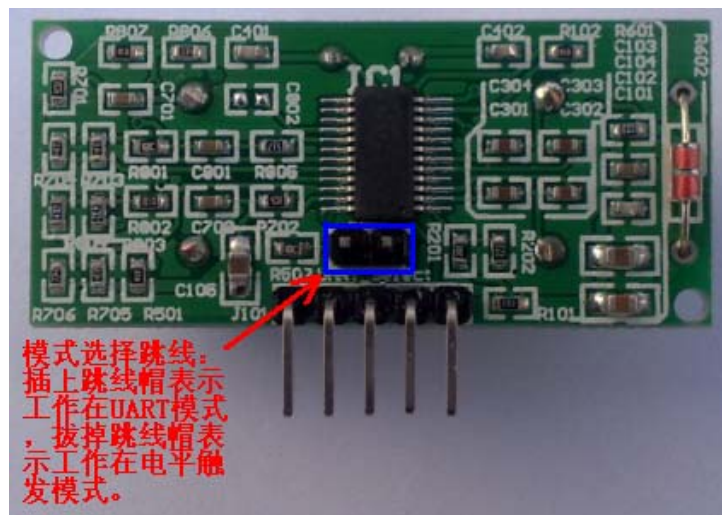


图 1.1: 模式选择跳线接口

电平触发模式下(GPIO 模式)只需要在 Trig/TX 管脚输入一个 10US 以上的高电平,US-100 便可通过 Echo 端输出一高电平,可根据此高电平的持续时间来计算距离值。即距离值为: $(\text{高电平时间} \times 340\text{m/s}) / 2$ 。此距离值已经经过温度校正,即不管温度多少,声速选择 340m/s 即可。

在串口模式下,通过 Trig/TX 管脚输入 0X55(波特率 9600),US-100 便会通过 Echo/RX 管脚输出两字节的距离值,第一个字节是距离的高 8 位(HData),第二个字节为距离的低 8 位(LData),单位为毫米。即距离值为 $(\text{HData} \times 256 + \text{LData}) \text{ mm}$ 。

在串口模式下,通过 Trig/TX 管脚输入 0X50(波特率 9600),US-100 便会通过 Echo/RX 管脚输出一个字节的温度值 (TData), 实际的温度值为 TData-45。例如通过 TX 发送完 0X50 后,在 RX 端收到 0X45,则此时的温度值为 $[69 (0X45 \text{ 的 } 10 \text{ 进制值}) - 45] = 24$ 度。

1.2 Arduino 介绍

Arduino 是源自意大利的一个开放源代码的硬件项目,该平台包括一片具备简单 I/O 功效的电路板以及一套程序开发环境。Arduino 可以用来开发可独立运作、并具互动性的电子用品,或者也可以开发出与 PC 相连的周边装置,同时能在运作时与 PC 上的软件进行沟通。Arduino 的硬体电路板可以自行焊接组装成,也可以购买已经组装好的,而整合开发环境的软体则可以自网路上免费下载与使用。通过 Arduino,可以做出很多令人惊奇的互动作品。

本文以 arduino duemilanove 2009 为例进行说明,其他 Arduino 平台的使用方法类似。

arduino duemilanove 2009 如图 1.2 所示:

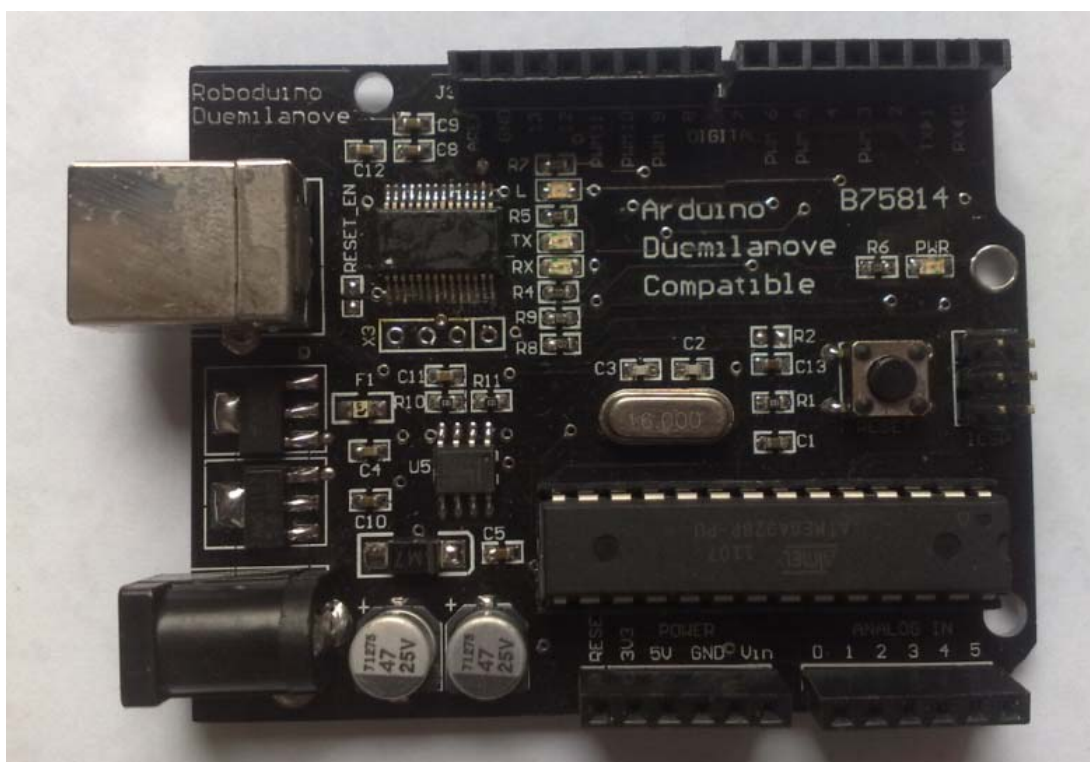


图 1.2: arduino duemilanove 2009

2. GPIO 模式下 US-100 与 Arduino 连线及例程

2.1 GPIO 模式下的连接

连接前首先将 US-100 模块背面的跳线帽拔掉。

GPIO 模式下 US-100 与 Arduino 的连接如表 2.1 和图 2.1 所示：

US-100 管脚	连接到 Arduino 对应的管脚
VCC	5V （POWER）
Trig/TX	Pin 3 （DIGITAL IO 3）
Echo/RX	Pin 2 （DIGITAL IO 2）
GND	GND
GND	GND （GND 可只连一个）

表 2.1：GPIO 模式下 US-100 与 Arduino 的连接

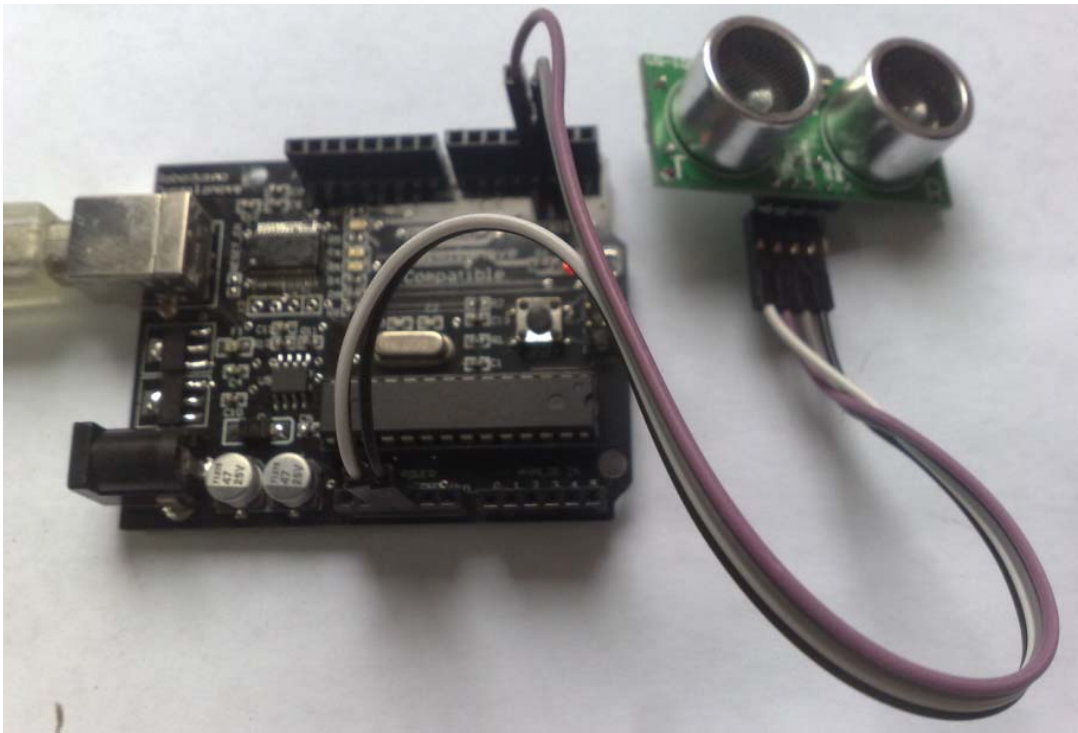


图 2.1：US-100 与 Arduino 的连接

2.2 GPIO 模式下的使用例程

```
unsigned int EchoPin = 2; //将 Arduino 的 Pin2 连接至 US-100 的 Echo/RX
```

```

unsigned int TrigPin = 3; //将 Arduino 的 Pin3 连接至 US-100 的 Trig/TX
unsigned long Time_Echo_us = 0;
unsigned long Len_mm = 0;

void setup()
{ //Initialize
  Serial.begin(9600); //测量结果将通过此串口输出至 PC 上的串口监视器
  pinMode(EchoPin, INPUT); //设置 EchoPin 为输入模式。
  pinMode(TrigPin, OUTPUT); //设置 TrigPin 为输出模式。
}

void loop()
{ //通过 Trig/Pin 发送脉冲，触发 US-100 测距
  digitalWrite(TrigPin, HIGH); //开始通过 Trig/Pin 发送脉冲
  delayMicroseconds(50); //设置脉冲宽度为 50us (>10us)
  digitalWrite(TrigPin, LOW); //结束脉冲

  Time_Echo_us = pulseIn(EchoPin, HIGH); //计算 US-100 返回的脉冲宽度
  if((Time_Echo_us < 60000) && (Time_Echo_us > 1)) //脉冲有效范围(1, 60000).
  {
    // Len_mm = (Time_Echo_us * 0.34mm/us) / 2 (mm)
    Len_mm = (Time_Echo_us*34/100)/2; //通过脉冲宽度计算距离.
    Serial.print("Present Distance is: "); //输出结果至串口监视器
    Serial.print(Len_mm, DEC); //输出结果至串口监视器
    Serial.println("mm"); //输出结果至串口监视器
  }
  delay(1000); //每秒 (1000ms) 测量一次
}

```

2.3 GPIO 模式下测试及截图

在使用时，首先在 Arduino 的开发环境中编辑源代码，编辑完后将程序进行编译，最后下载到 Arduino 开发板上，建议在下载程序到 Arduino 开发板上的过程中拔掉 US-100 与 Arduino 的连线。

待程序下载完毕，首先断掉电源，将 US-100 模块背部的跳线拔掉，然后按表 2.1 所示连接 US-100 与 Arduino 上的相应管脚。连接好后给 Arduino 开发板上电，系统便可运行。

此时打开 Arduino 开发环境中自带的串口监视器，便可查看运行的结果。

GPIO 模式下，2.2 中使用例程的测试截图如图 2.2 所示：

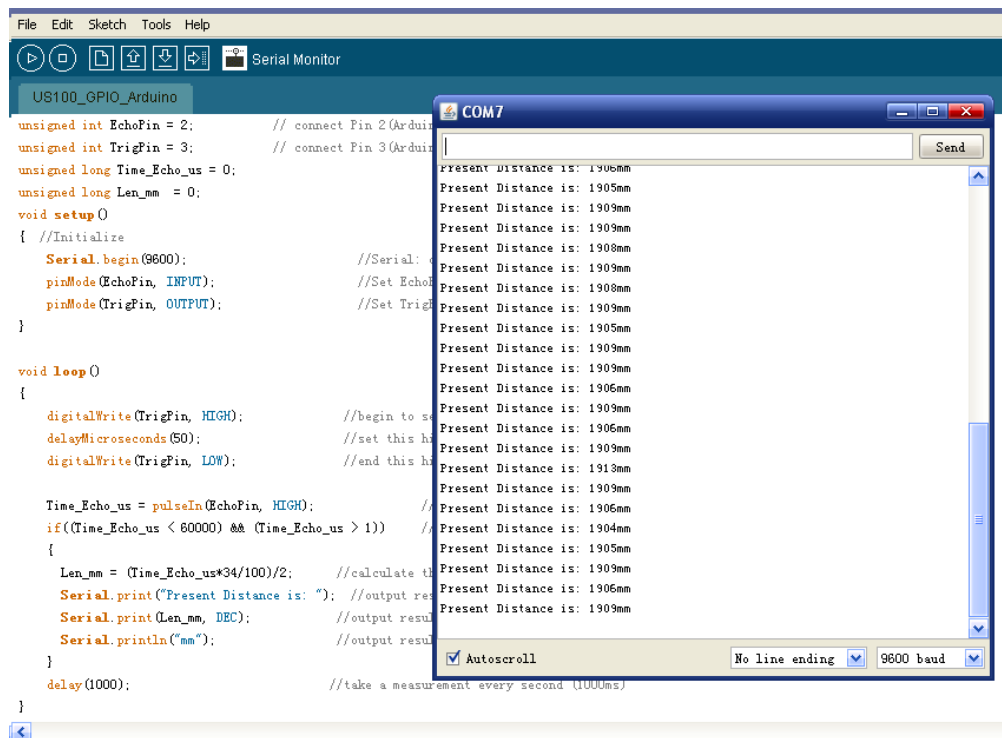


图 2.2： GPIO 模式下 US-100 测试截图

3. 串口模式下 US-100 与 Arduino 连线及例程

3.1 串口模式下的连接

连接前首先将 US-100 模块背面的跳线帽插上。

串口模式下 US-100 与 Arduino 的连接如表 3.1 和图 3.1 所示：

US-100 管脚	连接到 Arduino 对应的管脚
VCC	5V （POWER）
Trig/TX	Pin 1 （TX, DIGITAL IO 1）
Echo/RX	Pin 2 （RX, DIGITAL IO 2）
GND	GND
GND	GND（GND 可只连一个）

表 3.1： GPIO 模式下 US-100 与 Arduino 的连接

注意事项：在串口模式下，首先将程序下载到 Arduino 开发板上，然后将 Arduino 断电，将 US-100 和 Arduino 按照表 3.1 所示连接，连好后再给 Arduino 上电。

如果先将 US-100 与 Arduino 连好,再给 Arduino 下载程序,在下载程序时会出错,因为 US-100 与 Arduino 的通信和 Arduino 下载程序时使用的同一个串口,会相互干扰。

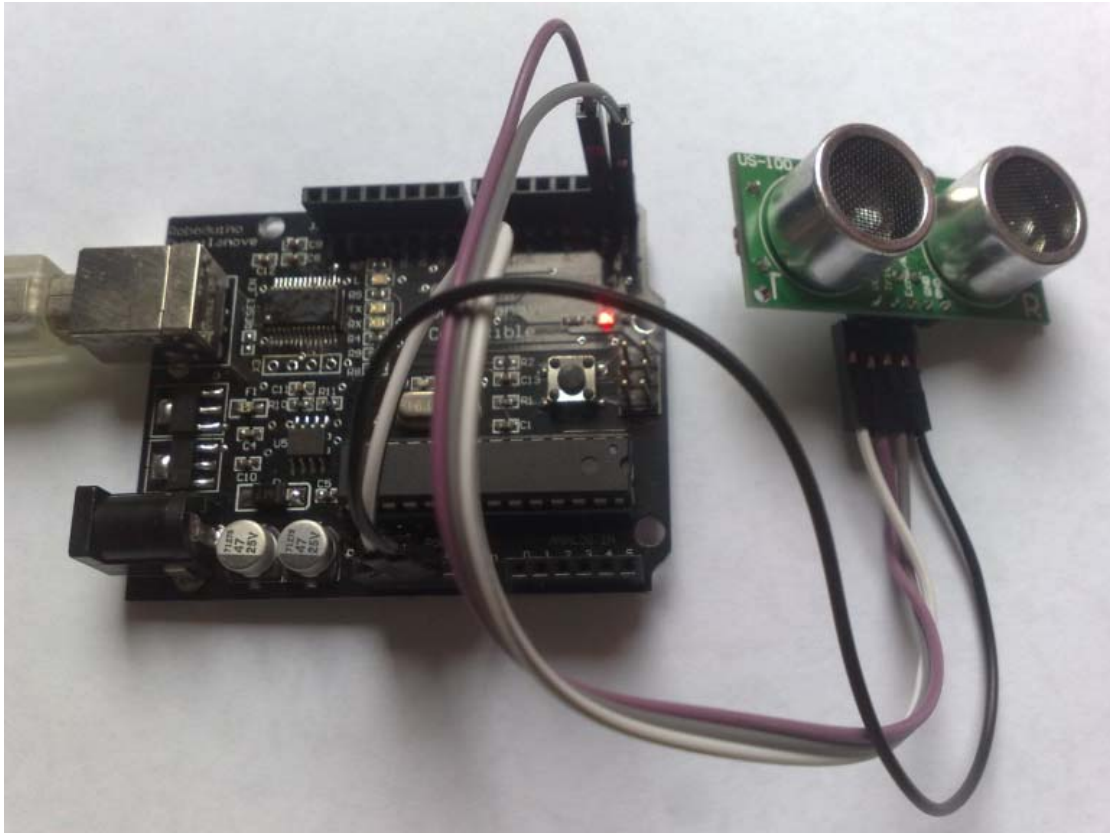


图 3.1: 串口模式下 US-100 与 Arduino 的连接

3.2 串口模式下测距使用例程

```
unsigned int HighLen = 0;
unsigned int LowLen = 0;
unsigned int Len_mm = 0;
void setup()
{
    //将 Arduino 的 RX 与 TX(Digital IO 0 和 1)分别于 US-100 的 Echo/Rx 和 Trig/Tx
    //相连,确保连接前已经使 US-100 处于串口模式。
    Serial.begin(9600); //设置波特率为 9600bps.
}

void loop()
{
    Serial.flush(); // 清空串口接收缓冲区
    Serial.write(0X55); // 发送 0X55, 触发 US-100 开始测距
    delay(500); //延时 500 毫秒
    if(Serial.available() >= 2) //当串口接收缓冲区中数据大于 2 字节
    {
        HighLen = Serial.read(); //距离的高字节
```



```

        LowLen = Serial.read();           //距离的低字节
        Len_mm = HighLen*256 + LowLen;     //计算距离值
        if((Len_mm > 1) && (Len_mm < 10000)) //有效的测距的结果在 1mm 到
10m 之间
        {
            Serial.print("Present Length is: "); //输出结果至串口监视器
            Serial.print(Len_mm, DEC);           //输出结果至串口监视器
            Serial.println("mm");                //输出结果至串口监视器
        }
    }
    delay(500);                                //等待 500ms
}

```

3.3 串口模式下测温使用例程

```

int Temperature45 = 0;

void setup()
{ //将 Arduino 的 RX 与 TX (Digital IO 0 和 1) 分别于 US-100 的 Echo/Rx 和 Trig/Tx
相连，确保连接前已经使 US-100 处于串口模式。
    Serial.begin(9600); //设置波特率为 9600bps.
}

void loop()
{
    Serial.flush();           // 清空串口接收缓冲区 t
    Serial.write(0X50);       // 发送 0X50，触发 US-100 开始测温
    delay(500);               //延时 500 毫秒
    if(Serial.available() >= 1) //当串口接收缓冲区中数据大于 1 字节
    {
        Temperature45 = Serial.read(); //读出 US-100 返回的结果
        if((Temperature45 > 1) && (Temperature45 < 130)) //返回的有效值在
1 到 130 之间
        {
            Temperature45 -= 45;           //实际温度值等于返回值减 45
            Serial.print("Present Temperature is: "); //输出结果至串口监视
器

            Serial.print(Temperature45, DEC); //输出结果至串口监视器
            Serial.println(" degree centigrade."); //输出结果至串口监视器
        }
    }
    delay(500);                //等待 500ms
}

```

3.3 串口模式下测试及截图

在使用时，首先在 Arduino 的开发环境中编辑源代码，编辑完后将程序进行编译，最后下载到 Arduino 开发板上，在下载程序到 Arduino 开发板上的过程中需要拔掉 US-100 与 Arduino 的连线。

待程序下载完毕，首先断掉电源，同时确保 US-100 模块背部的跳线已经插上，US-100 工作在串口模式下，然后按表 3.1 所示连接 US-100 与 Arduino 上的相应管脚。连接好后给 Arduino 开发板上电，系统便可运行。

此时打开 Arduino 开发环境中自带的串口监视器，便可查看运行的结果。

串口模式下测距截图如图 3.2 所示：

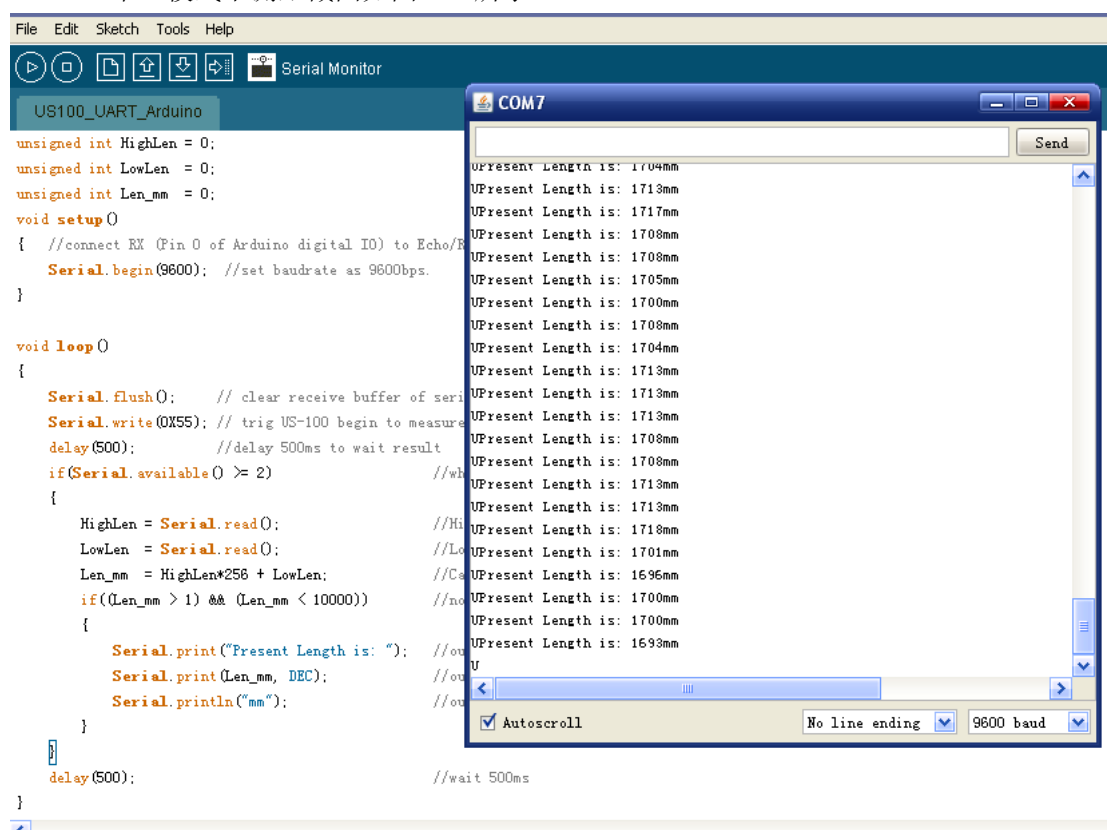


图 3.2：串口模式测距截图

从图 3.2 中可以看到，在串口测距时，每一行前多一个字符‘U’，这是因为 Arduino 与 US-100 通信和 Arduino 与 PC 机通信使用的是同一个串口，当 Arduino 向 US-100 发送 0X55 触发测距时，0X55 同时被 PC 机接收到，所以显示‘U’（U 的 ASCII 码为 0X55）。

串口模式测温截图如图 3.3 所示

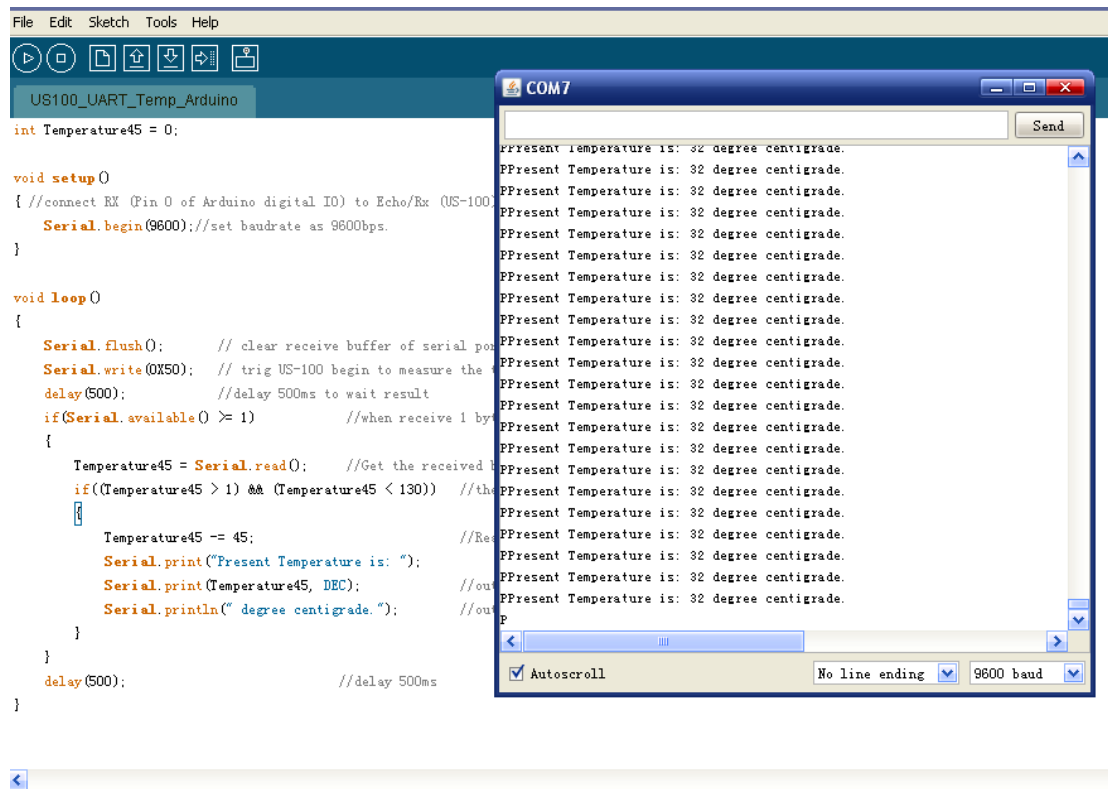


图 3.3: 串口模式测温截图

从图 3.3 中可以看到, 在串口测温时, 每一行前多一个字符 ‘P’, 这是因为 Arduino 与 US-100 通信和 Arduino 与 PC 机通信使用的是同一个串口, 当 Arduino 向 US-100 发送 0x50 触发测距时, 0x50 同时被 PC 机接收到, 所以显示 ‘P’ (P 的 ASCII 码为 0x50)。