基于 RS485 的网络通信协议的设计与实现

1．实验目的

模拟工业网络，设计通信协议并实现。

2. 实验设备

1 台计算机，3 台单片机，USB 转 485 模块 1 个，TTL 转 485 模块 3 个，485 通讯线缆若干。

3. 实验要求

3.1 通信协议要求

（1）定时数据通信：每秒钟每台计算机分别交换 10、20、30、40、50 个字节的数据；

（2）广播通信：每十秒中，发布广播数据对各个计算机的时钟进行同步；

（3）错误检验：CRC 错误检验，有错误时，应采用相应的错误处理程序；

（4）网络管理：随时获知网络中各节点的工作状态，当有节点故障退出网络或新的节点加入网络时，能够记录网络状态；

（5）数据记录：在任一个网络节点，都可以实时记录本节点的数据，并以曲线形式观察当前和历史数据及节点的工作状态；

（6）闭环控制：模型计算机、检测计算机、控制计算机、监控计算机。

注：根据本设计实际优化，将定时通信周期从 1s 降低到 400ms，同时将接收固定字节数据改为了自适应，但最长不超过 50 个字节。

3.2 编程要求

（1）编程语言采用 VB、VC 等常用语言；

（2）各节点计算机运行不同的自编软件，能够充分展示所设计的通信协议及闭环控制中各环节的运行情况。

4. 实验分析

4.1 物理层的简单介绍

RS485 最大无中继传输距离为 1200 米，最大传输速率为 10Mbps。RS-485 是目前最常用的物理连接方式之一，使用已经很成熟。

4.2 网络拓扑结构

本网络使用的是总线型的网络拓扑结构，使用的通信协议是串行链路上自行

设计的通讯协议，其物理层使用的是 RS485。

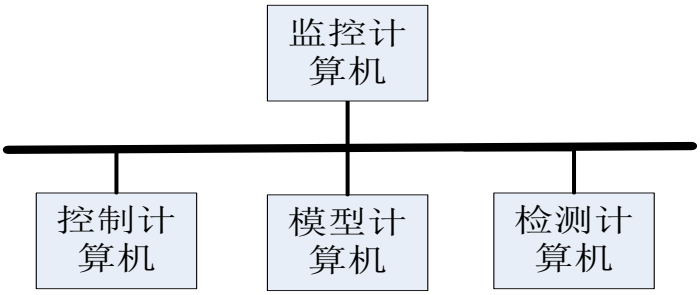

图 1 网络连接图

各个站点和计算机的作用：

（1）主站：对三台从机进行 400ms/T 的轮询访问，监测其运行状态，收集来自控制计算机、模型计算机、检测计算机的输出信息进行显示记录并对模型机的输出进行波形绘制，对从机的功能性要求进行应答处理，比如从机数据校验失败要求主机重发。

（2）控制计算机：产生输入的设定波形信息，并且内部含有 PID 控制器，产生控制量；

（3）模型计算机：将控制计算机产生的控制信息作为输入信号，经过模型后产生输出信号；

（3）检测计算机：将模型计算机的输出信息作为输入信号，经过传输变送后，产生反馈信号；

4.3 通信的具体实现

4.3.1 定时通信和闭环控制的实现

主机对三个从机进行 400ms/T 的轮询访问，每 100ms 访问一个从机，最后第 100ms 用于时间同步，每 25 个周期既 10s 同步一次。对从机进行轮询访问时，若从机未开始工作或数据未刷新，则只进行状态应答用于主机确认从机运行状态。当主机对从机设置好参数，从机开始正常运行时，每个从机每周期都有 100ms 的时间接收主机的数据并对自身要输出的数据进行刷新，主机轮询时，若数据已刷新则对数据进行读取记录并转发给下一个从机。整个控制系统工作流程如下：

1. 主机设置控制计算机激励，PID 等参数。
2. 控制计算机接收到参数数据，并产生控制量数据
3. 主机轮询控制计算机，得到控制量数据，进行记录后转发给模型计算机。
4. 模型计算机接收到控制量并产生输出量数据。

5. 主机轮询模型计算机，得到的输出量数据，进行记录、绘制波形并转发给检测计算机。
6. 检测计算机接收到输出量数据，并产生反馈量。
7. 主机轮询检测计算机，得到的反馈量数据，进行记录并转发给控制计算机。
8. 控制计算机接收到反馈量数据，并产生新的控制量数据开始下个周期的循环。

4.3.2 广播通信实现

主机设置100ms定时器，以400ms/T对三个从机进行轮询访问，前三个100ms对三台从机进行轮询，最后第 100ms，主机进行计次，在第 25 个周期既 10s 时主机将当前时间广播至三个从机。其报文格式在下文进行介绍。

4.3.3 错误检测实现

错误检测不仅包括 CRC 校验码与数据的不匹配，还包括从机在规定的超时时间内没有将响应帧回馈到主站等。

当数据帧的最后4字节的CRC校验码与根据数据帧计算出的CRC校验码不一致时，从机回复错误帧说明 CRC 校验有误，请求主机再次发送。

对于从机在规定的超时时间内没有将响应帧发回主机，主机进行计次，连续三个周期访问均无应答时，判定该从机离线，并广播停机报文，使其余设备停机。

4.3.4 网络管理实现

主机利用轮询时从机的回复来判定从机是否在线，无论收到状态回复，还是数据回复均认为该从机在线，并加入网络列表。若三次轮询某从机均无应答则认为该从机离线。

4.3.5 数据记录实现

在模型计算机、检测计算机、控制计算机上都可以实时的记录本节点的信息。监控计算机用来记录所有站点的信息。并且随着总线上的数据的交换，数据在不断的实时更新。

4.4 帧的格式

在本次实验中用到的帧主要分为两种：

1. 主机给从机发送数据
2. 从机给主机发送数据

4.4.1 主机给从机发送数据

该部分地址码、功能码如下：

地址码：

A 检测计算机；B 控制计算机；C 模型计算机；D 广播；

功能码：

a 轮询；b 时间同步；c 参数设置；d 发送数据；e 设备停机；

主要分为 2 种格式，

1. 主机设置从机参数、发送数据和时间同步共用一种格式。
   a) 主机设置从机参数，参数为激励、PID 数据。参数数据采用固定帧格式每个参数 6 个字节共四个参数，加上地址码 1 个字节，功能码 1 个字节，CRC 校验码 4 个字节，终止符 2 个字节共 32 个字节。参数的 6 个字节中第一个字节为符号位，1 表示正，2 表示负；后五个字节为数据具体数据。
   b) 主机发送数据，数据采用自适应帧格式发送，既报文中无数据长度，由从机自行接收，但数据长度最多不超过 50 个字节，否则从机拒绝接收。数据格式与参数格式相同，第一位为符号位，1 为正 2 为负剩余树为实际数据。
   c) 时间数据采用固定格式发送，格式为<时:分:秒>共 8 个字节。

   设置参数与时间同步的地址码为广播‘D’，功能码分别为‘c’和‘b’；发送数据对应的地址码为目标从机，功能码为‘d’。均采用 ASCII 码发送。其格式如下：

   地址码 + 功能码 + 数据 + CRC + 回车符 +换行符
2. 主机给从机发送功能性指令，既轮询与停机指令。该指令对应的报文无数据，轮询指令对应的地址码为目标从机，功能码为‘a’；停机指令对应的地址码为广播地址‘D’，功能码为‘e’，格式如下

   地址码 + 功能码 + 回车符 +换行符

4.4.2 从机给主机发送数据

该部分地址码、功能码如下：

地址码：

0A 检测计算机；0B 控制计算机；0C 模型计算机；

功能码：

00 回传数据；01 参数设置成功；02 参数设置失败；03 状态应答；

04 数据 CRC 校验失败；

主要分为 2 种格式，

1. 回传数据，回传数据时报文中会给定数据长度占用 2 个字节，数据占用 N 个字节，地址码占用 2 个字节，功能码占用 2 个字节，CRC 校验码占用 4 个字节，停止符占用 2 个字节共 12+N 个字节。字节总数要求小于等于 50 个字节否则不予接收。其格式如下：

   地址码 + 功能码 + 数据长度 + 数据 + CRC + 回车符 +换行符

2. 从机给主机发送功能性指令，及参数设置成功与否的状态回复，轮询时的状态应答，CRC 校验失败时要求主机重新发送。格式如下：

   地址码 + 功能码 + 回车符 +换行符

4.5 闭环控制的总线周期设计

4.5.1 闭环控制结构图

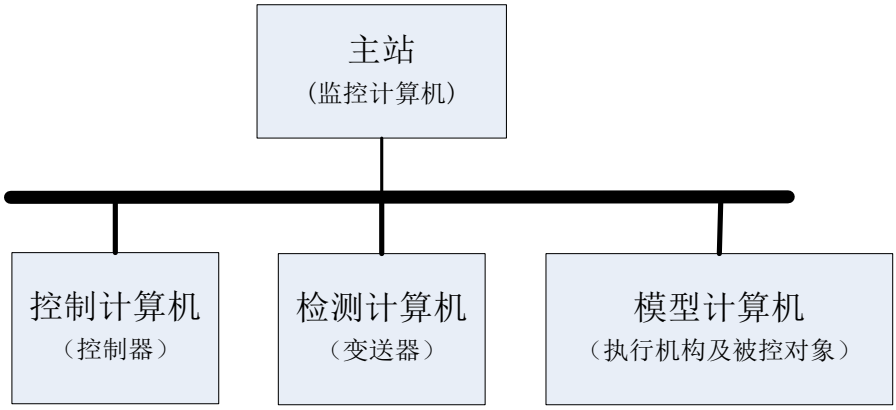

图 2 闭环控制总线拓扑图

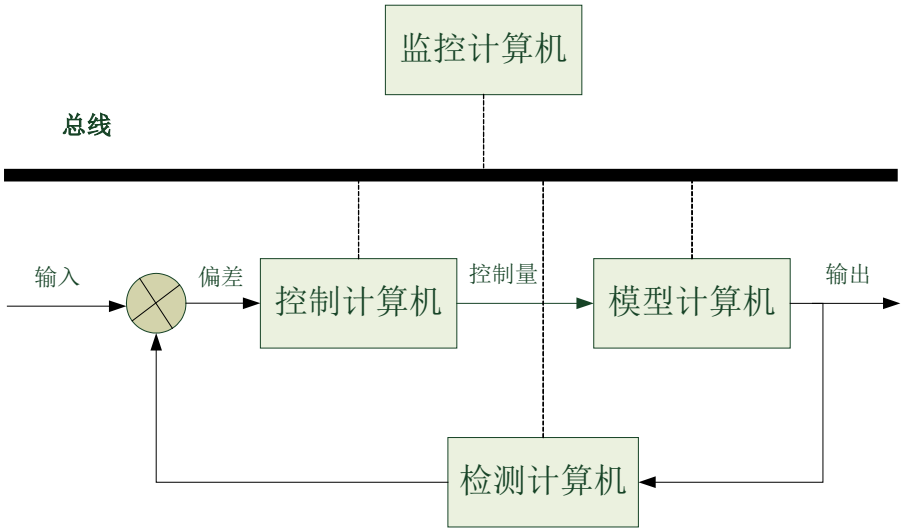

图 3 闭环控制回路结构图

（1）本次设计将监控计算机设计为主站，主要完成控制计算机，检测计算机，模型计算机的通信和数据交换，同时进行网络管理和时钟同步。

（2）模型计算机作为从站，相当于控制回路中的执行机构和被控对象，根据所建立的模型输出实际的物理量，供后续检测计算机检测处理。

（3）检测计算机作为从站，相当于控制回路中的变送器，采集模型计算机的输出，并将物理信号转化为可用的模拟量信号，传递给控制计算机，进行运算（PID 算法）。

（4）控制计算机作为从站，相当于控制回路中的控制器，根据控制算法，计算出控制量，通过主站将控制量发给模型计算机，从而改变模型计算机的输出。

（5）完整的控制为：检测计算机将从模型计算机采集回来的数据给到控制计算机，控制计算机经过运算，给出控制量，作用到模型计算机，从而改变模型计算机的输出，如此循环往复，构成闭环控制。

4.5.2 控制回路宏周期

控制回路宏周期表：

表 1 控制回路宏周期表

计算机名称	执行内容	开始时间(ms)
监控计算机（主机）	主机设置控制计算机激励，PID 等参数	0
控制计算机（从机）	控制计算机接收到参数数据，并产生控制量数据	
监控计算机（主机）	轮询控制计算机,将得到的输出量转发模型计算机	100
模型计算机（从机）	接收到控制量数据，产生输出量	

监控计算机（主机）	轮询模型计算机,将得到的输出量转发检测计算机	200
检测计算机（从机）	接收到输出量数据，产生反馈量	
监控计算机（主机）	轮询检测计算机,将得到的反馈量转发控制计算机	300
控制计算机（从机）	接收到反馈量数据，产生新的控制量	
监控计算机（主机）	周期计数，待计数值达到 25 时，广播时间	400
监控计算机（主机）	轮询控制计算机,将得到的输出量转发模型计算机	500

5.编程实现

主机使用 Python 语言编程，从机使用 C 语言编程。

5.1 主站 1（监控计算机）

主站要实现串口、波特率、校验码、数据位、停止位、数据长度、激励、PID 等参数的设置；由于主站作为监控计算机，对从站的当前状态进行监控，并通过控件的设计进行显示；当主站接收到从站的数据时，在相应的位置进行显示，方便及时进行调控；为了直观的观察闭环情况，画出给定值与当前输出值，进行比较后将误差发送给控制计算机。

编程前导入需要使用的包，Pyserial 包专门用于串口操作，Pyqt5 包用于设计 GUI，pyqtgraph 包用于数据的曲线图显示。

首先新建 ui_demo_1.py 文件，用于创建图形用户界面，本次设计的图形用户界面主要包含五个部分及若干个 pushbutton 按钮。

1. 串口设置部分

主要用于检测串口，并设置串口参数，包含检测串口按钮，串口选择列表，波特率选择，数据位选择，校验位选择，停止位选择等。利用 QtWidgets.QGroupBox()函数创建一个组合框，组合框里利用 QtWidgets.QFormLayout()函数创建一个表单布局，将上述功能逐个添加进表单布局。最后设置好每个菜单下拉选择框里的参数，调整好布局，如图 4。

图 4

2. 参数设置部分

主要用于设置 PID 及激励参数。同上先创建组合框再创建表单布局，表单布局里设置标签及对应的文本行 QtWidgets.QLineEdit()，对文本行设置对应的校验器 QDoubleValidator(self)，其中激励校验器规定激励文本行的输入范围为 0-9999 且可保留 1 位小数，PID 参数输入范围为 0-99 且可保留三位小数。如图 5.

图 5

3. 从机状态部分

该部分用于实时显示从机状态，通过更改表单布局里从机标签对应的文本实现，通过代码运行时设置的定时器实时刷新从机的状态，如图 6。

图 6

4. 三个从机输出的文本显示部分

用于显示从机回传的数据，分别创建三个文本框用于显示。如图 7

图 7

5. 模型计算机数据的曲线显示部分

先将回传的数据放入一个动态数组中，利用 pyqtgraph 包中的 pg.PlotWidget（）函数创建一个图像显示的区域，后期利用定时器对区域内曲线进行实时刷新，达到动态显示的目的。如图 8

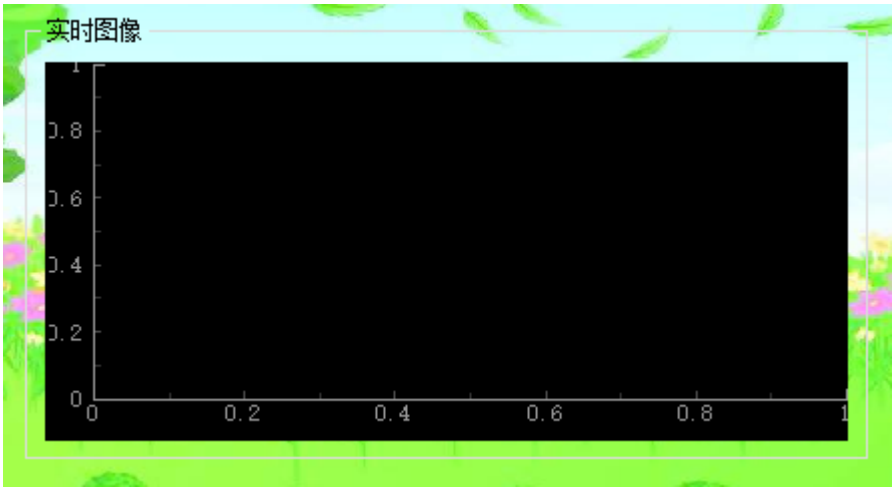

图 8

再创建需要的若干个功能按钮，例如暂停，继续，保存数据，清空数据等 GUI 的设计就完成了。

接下来第二步，创建 pyserial_demo.py 文件用于编写具体运行程序和每个控件对应的槽，既控件对应的函数。整个软件的运行代码由两部分构成，一部分是每个控件对应的执行代码，另一部分是自运行的代码。

控件对应的执行代码主要有以下几个用户自定义函数：

1. CRC 校验的部分

   包括 calculateCRC(datalist)，calculateonebyte(databyte, tempcrc)，两个函数。将要检测的数据队列输入 calculateCRC 函数，函数将调用 calculateonebyte 函数对队列里的每个数据进行计算，最后返回计算得出的 CRC 校验码。

2. 数据的格式化部分

   dataformatting(data,len,flag):函数将得到或者需要发送的数据进行编码方便数据的传输，单片机接收到数据时再对该数据进行对应的解码即可还原原始数据。编码方式为将 float 型数据扩大 1000 倍，再在数据首位根据数据正负设置 1 或者 2 的标志位，达到利用 ascii 码传输后方便接收使用的目的。例如数据为 120.001 则编码后将要传输的数据为 1120001。

3. 串口操作部分

   打开串口按钮对应的函数为 port_open(self):，关闭串口对应的函数为 port_close(self):，利用 Pyserial 模块里的函数对串口进行打开和关闭操作。

4. 参数设置部分

   参数设置区的运行按钮对应 PID_data_send(self):函数，对获取到的 PID 和激励参数进行编码并发送，收到从机发回的参数设置成功消息时将参数显示在从机输出的数据显示区。PID_data_cancel(self):函数对应停止按钮，按下此按钮后，会对从机广播停机指令，并停止主机的数据转发功能。

5. 功能按钮部分

   绘图区的清除按钮对应的 DrawArea_clear(self):函数用于清除绘图区的图像；从机数据接收区的 receive_data_clear(self):函数用于清空接受区的数据显示；Continue(self):函数对应‘继续’按钮对应暂停后继续运行的功能；Pause(self):函数对应暂停按钮，可以使从机暂停工作；

receive_data_save(self):函数对应保存按钮，用于保存从机接收区的数据，将数据保存在该 exe 执行文件的同目录下。

自运行的代码分为轮询数据的发送代码和数据的接收代码。

轮询发送的代码由一个 100ms 的定时器循环自启运行，对应的函数为 data_send(self):，该函数每 100ms 执行一次，轮流给三个从机发送轮询指令，并且将该软件的所有刷新和状态检测都挂接在该定时器下，实现每 100ms 对从机状态进行刷新，对实时图像进行刷新显示。

数据的接收代码对应的函数为 data_receive(self):函数，该函数由一个 2ms 的定时器循环自启运行，每隔 2ms 对串口进行检测，无数据时忽略，有数据是对数据进行接收，解码，显示，绘图，如此时转发功能开启数据转发功能，则将接收到的该从机的数据发送给控制循环里的下一个从机。

最后将每个控件与其对应的槽进行绑定，主机部分程序就设计完毕了。

5.2 从站 2（控制计算机）

控制计算机主要是控制算法的实现，在实验中设计了 PID 控制算法，在界面上分别输入比例、积分、微分相应的参数，实现对模型的控制，并对接收到的数据在界面输出。

该从机代码的编写需要 Keil5 软件创建工程文件，导入需要用的到 HAL 库，本次设计由于单片机仅用到串口收发功能，所以工程文件直接对正点原子 STM32F103 系列的串口收发例程进行修改，并编写需要的用户自定义函数得到。

首先编写串口接收对应的协议，设置 16 位的寄存器 USART_RX_STA 进行状态标记。该寄存器的 bit11 为地址标志位接收到该从机的地址 B 或者广播地址 D 时置 1；bit15~bit12 以及 bit8 和 bit7 为功能标志位分别在功能码为接收参数、轮询、接收数据、接收广播、停机、CRC 校验错误时置 1；bit9 为接收到回车符标志位，bit10 为接收到换行符标志位，逻辑是接收到回车符时检测若下个字符为换行符则接收完成，若不是则接收失败清空接收区域；剩下的 bit6~bit0 作为接收计数。接收完成后数据将会保存在一个字节型数组 USART_RX_BUF[]中，协议设计每次接收数据<=50 个字节。否则拒绝接收。

程序主循环的逻辑是，每 2ms 对标志寄存器 USART_RX_STA 的 bit10 进行检测，若有数据接收成功便对数据进行解码，用到的用户自定义函数为 DataConversion(int len,bool FLAG,int output)，解码后得到实际数据，然后对该数

据利用函数 Crc(int LEN,u8 CRCBUFFER[],int j)进行 CRC 校验，校验失败则发送对应功能码，校验成功则根据对应功能，参数设置或者接收数据进行对应的处理。PID 校正的功能主要由 PID_realize()和 PID_init()两个函数实现，PID 控制器采用增量式进行计算

incrementSpeed=pid.Kp*(pid.err-pid.err_next)+pid.Ki*pid.err+pid.Kd*(pid.err-2*pid.err_next+pid.err_last);

其中：

incrementSpeed ：本次计算得出的增量

pid.ActualSpeed：本次输出=上一次输出 pid.ActualSpeed+本次增量 incrementSpeed

pid.Kp，pid.Ki，pid.Kd：PID 参数

pid.err ：本次误差量 =设定值 pid.SetSpeed-上一次实际输出 pid.ActualSpeed

pid.err_next：前一次误差量

pid.err_last：上前一次误差量

5.3 从站 3（模型计算机）

模型计算机主要是设计一个模型，为了便于观察和计算，使用了大家都熟悉的一阶惯性模型，具体模型如下：

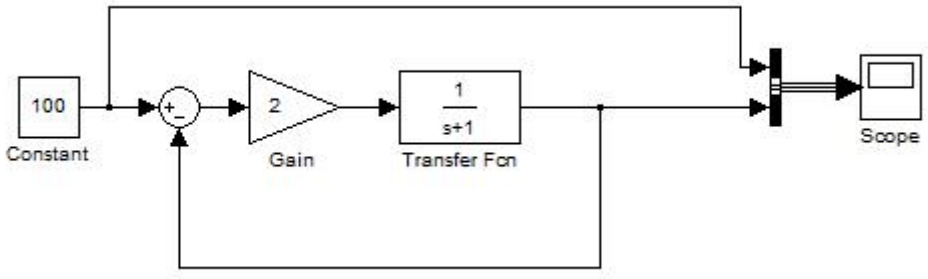

图 9　控制模型的设计

把控制计算机发送过来的数据作为输入，经过一节惯性模型的处理后输出，最后发送出去。

从站 2 串口接收的协议与程序对应的流程不尽相同。在模型的设置方面，模拟的被控对象为一阶惯性系统。采用增量式进行计算

modle.SystemOutput = (modle.InertiaTime * modle.ResultValueBack + modle.SampleTime * modle.PidOutput) / (modle.SampleTime + modle.InertiaTime);

modle.SystemOutput ：本次输出

modle.ResultValueBack：上一次输出

modle.InertiaTime ：惯性时间常数 默认设置为 3

modle.SampleTime：采样时间 默认设置为 0.1

5.4 从站 4（检测计算机）

检测计算机的功能是对模型计算机的输出进行显示，便于观察。因而，设计界面较为简单，除了相应的参数设置外，对接收到的数据进行输出即可。

程序逻辑与其他从站逻辑相同，不同点在于，接收到数据并解码校验完成后直接对数据再编码进行回传。

6.调试

6.1 通讯

设计难点在于实时发送数据与接收数据。特别是发送数据的频率，需确保在主从数据能正确传输，从机能有充分的时间处理数据的情况下尽可能减短发送频率，增加通讯的实时性。，经过反复调试，最终确定参数值。最终通讯频率设置为 100ms/次。

6.2 数据的处理

数据在发送和接收时进行格式化处理，处理方式已在第五部分编程实现的主机代码介绍中做了详述，数据格式化处理后以 ascii 码的方式传输，从机收到数据后依照格式逐字节接收，CRC 校验完成后即可正确收到数据。

7.结果及分析

7.1 结果

主站（监控器）界面显示：

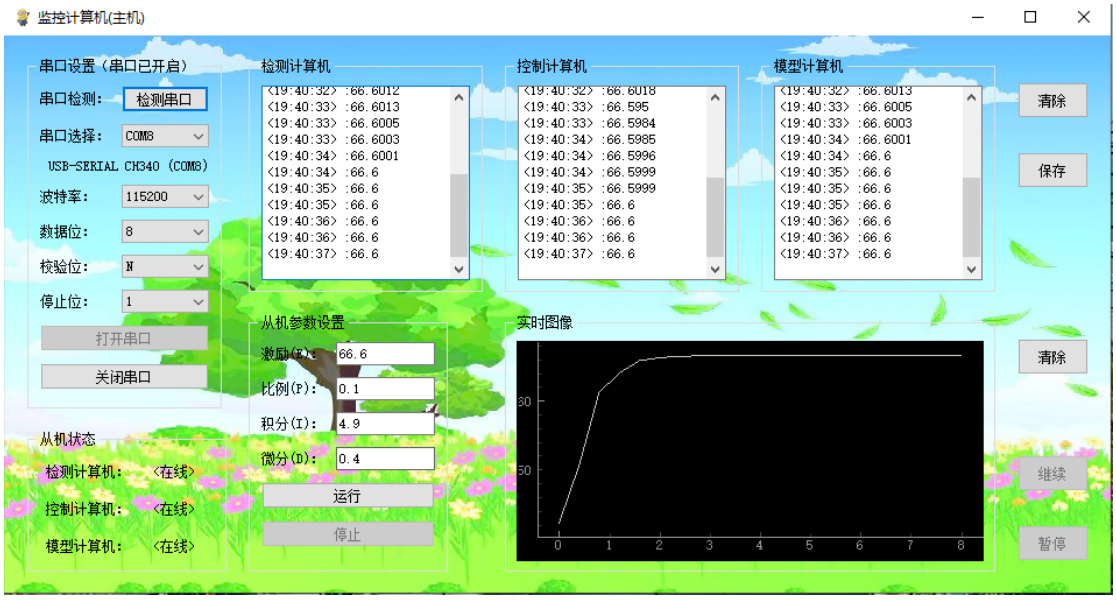

图 10 监控计算机的结果显示

Matlab 仿真结果如下：

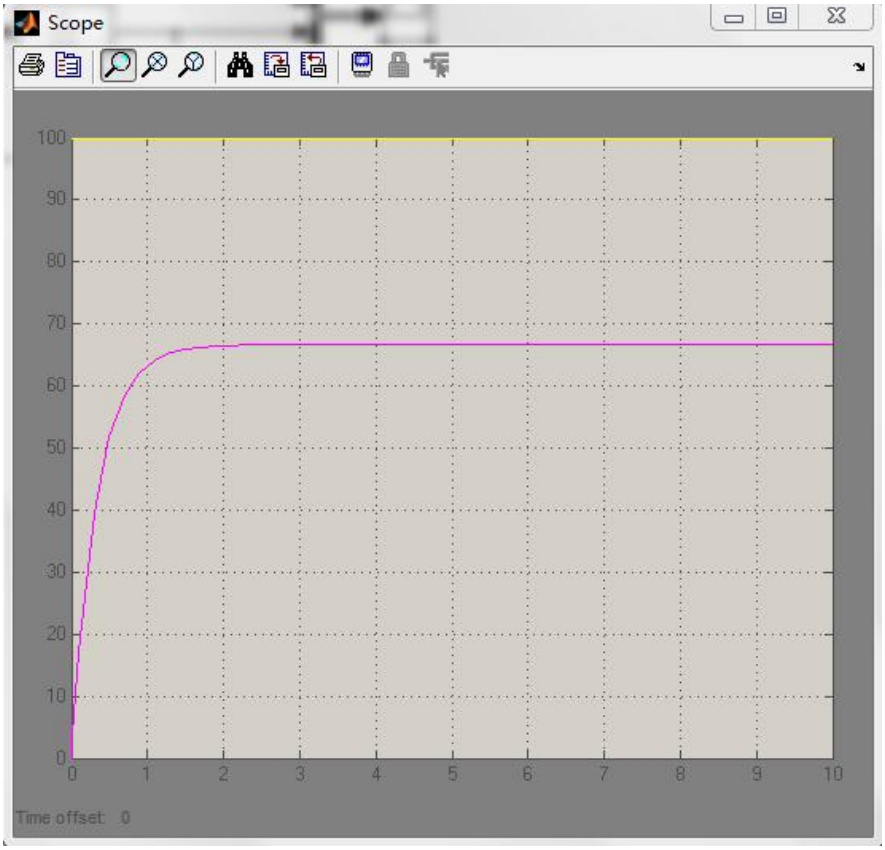

图 11 matlab 仿真结果

通过对仿真结果（图 11）与实际的 485 网络闭环输出得到图（图 9）对比，其输出的波形图是基本一致的，从而验证了 RS485 闭环网络的正确性。

7.2 分析

本次设计的软件在实际的反复测试中均能正常和稳定的运行，经过后期的长时间 Debug，优化操作设置，目前基本能较好地完成设计要求，但仍然有不足点，例如在运行时某从机串口收发端突然悬空，会影响其它从机的通讯；软件运行时若非法强行拔出设备，软件会崩溃只能再重启等等已知却未能修复的 BUG。这还需要更深入的学习 PYTHON 语言的使用，多多练习。相信通过不断地学习和练习，这些问题终会解决。