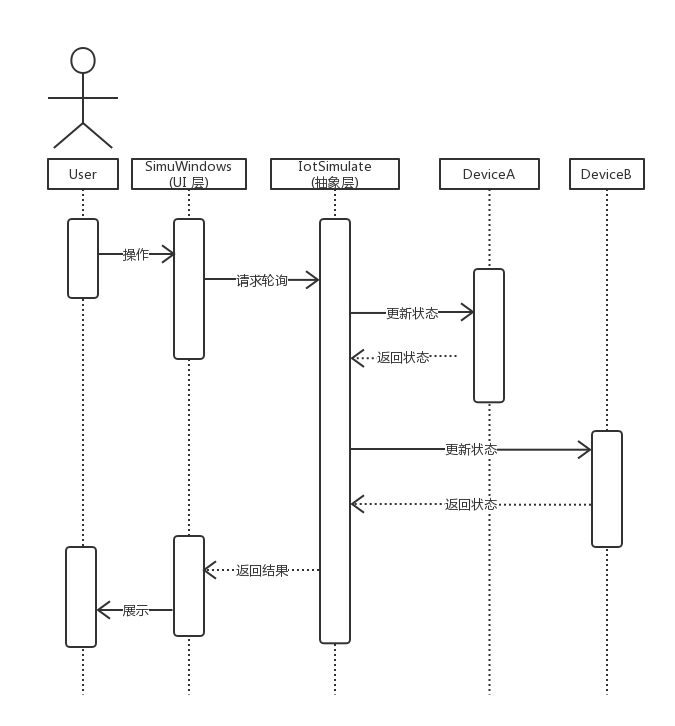
软件结构分为物理设备抽象层，UI交互层，用户通过UI交互层与物理设备抽象层进行间接交互。同时物理设备抽象层还可以作为真实世界中设备的抽象，参与到虚拟仿真平台之中。

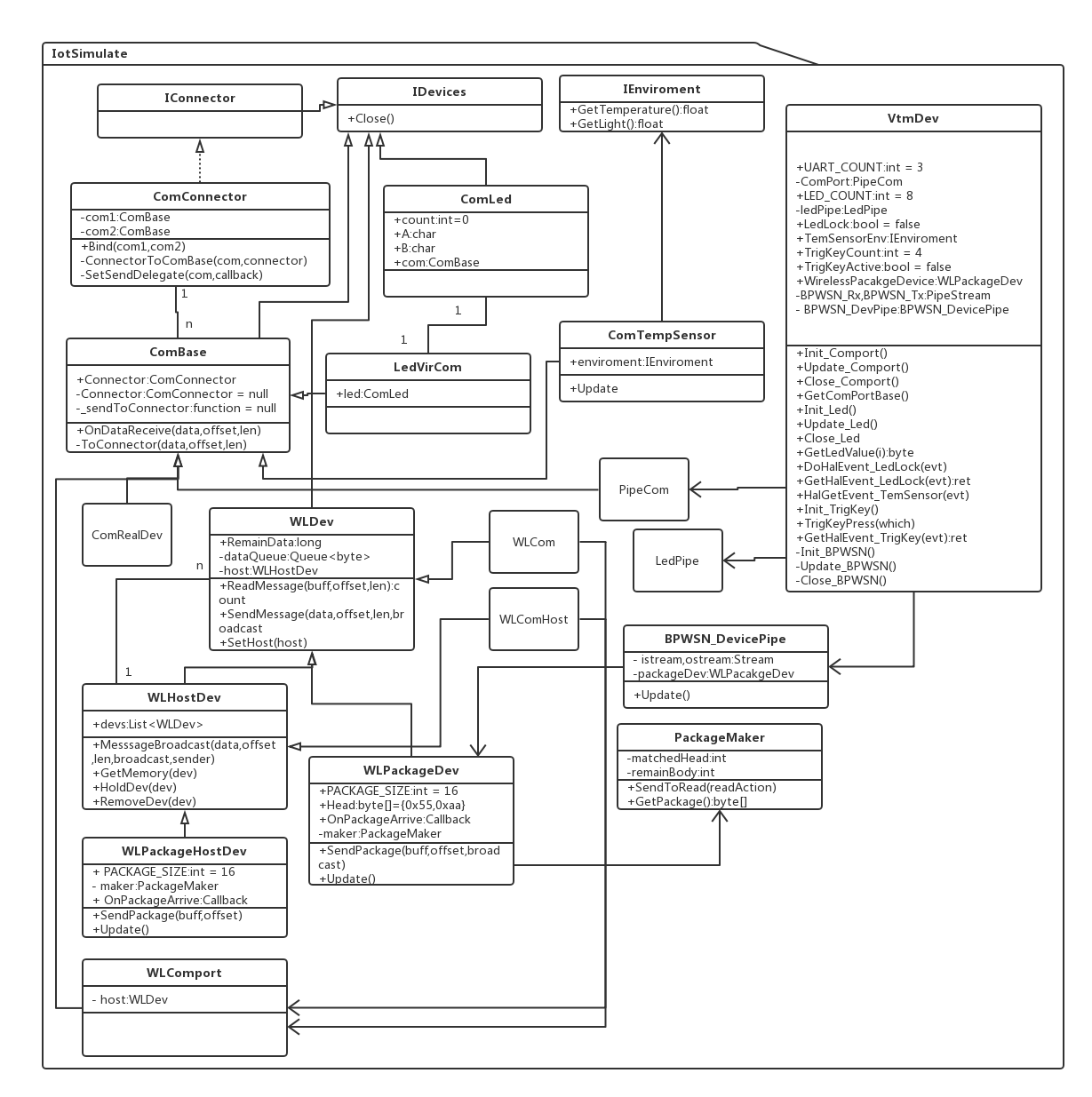
软件框架整体的时序图如下：



用户通过与UI进行交互，间接参与到抽象层物理设备之中。

3.2 硬件抽象层

抽象层的主要使用面向对象的方式进行描述，经建模开发，抽象层目前的UML类图描述如下：



其中包括：

IDevice接口 负责对物理设备进行抽象

IConnector接口 将连接线视为一种物理设备，进行抽象建模

ComBase抽象类 描述了串口设备的基本特征，所有的串口设备均需继承此类

ComConnector类 串口连接线，将两个ComBase绑定，并在Update函数中对ComBase双方的输入输出队列进行转发

ComRealDev类 具体设备，描述真实的串口设备，其输入输出队列是直接绑定到实际串口设备的。

ComLed类 具体设备，持有一个串口，将串口最后接受的字节显示出来

LedVirCom类 仅可被ComLed类访问，对外表现为ComBase，是ComLed的串口

IEnviroment接口 环境接口，用于描述一个真实或虚拟的环境

ComTempSensor类 具体设备，串口温度传感器，可持有一个IEnviroment，将感知到的温度输出至串口(温度取整+127)

WLDev类 无线网络设备，用于描述一个无线网络设备的特征

WLHostDev类 无线网络主机，可挂载多个WLDev，建立一个基于广播的星形无线网络

PackageMaker类 定义一个封包，长度16byte，以0x55AA开头，且最末尾字节为校验字节

WLPackageDev类 封包无线网络设备，此设备可发送或接收封包

WLPackageHostDev类 封包无线网络主机，对网络中的封包进行相应的主机，可以不使用

WLCom类 具体设备，将无线网转发至串口的设备

WLComHost类 具体设备，将无线网转发至串口的主机设备

VtmDev类 虚拟单片机，其构造函数需要指明一个二进制文件作为单片机的程序，其具体实现细节将在之后描述

PipeCom类 VtmDev的串口组件

LedPipe类 VtmDev的数码管模块相关类

BPWSN\_Device类 BroadcastPackageWirelessSensorNetworkDevice VtmDev的无线网模块

所有的IDevice隐含一个Update方法(由于不是必须的，没有写到接口)，Update方法通过模块自身注册DispatchedTimer执行，并可自行决定频率。

3.3 VtmDev单片机的具体实现

VtmDev单片机由VtmDev及相关类、VHClient虚拟机、VTMLib函数库、VHPipe.dll动态链接库、GCC编译器组成。

VtmDev及相关类负责将相关的操作进行封装，之后嵌入硬件抽象层。

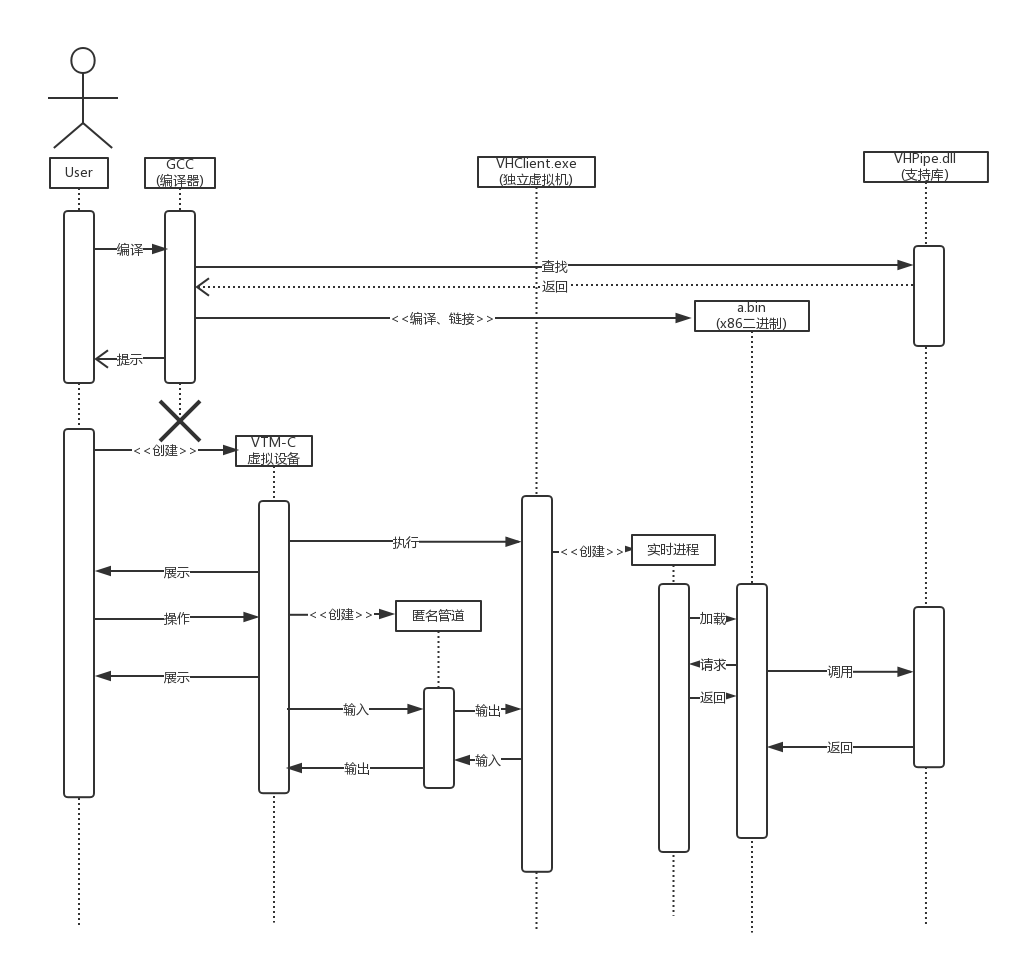
VHClient是一个独立可执行的程序，执行时需要传入固定的参数，其中包括欲加载的二进制文件路径，各种进程间通信所需的匿名管道等，在运行后将立即加载输入的二进制文件，并将交互操作以匿名管道的方式发送出去。

VHPipe.dll是一个使用C++编写的动态链接库，主要提供与VHClient.exe交互所需的各种函数接口。

VTMLib函数库是一个C语言支持库，用于提供单片机编译所需的上下文环境，包括对单片机HAL(硬件抽象层)的封装。

GCC及相关Makefile脚本、Compile.bat用于编译虚拟单片机能够解析的二进制文件。

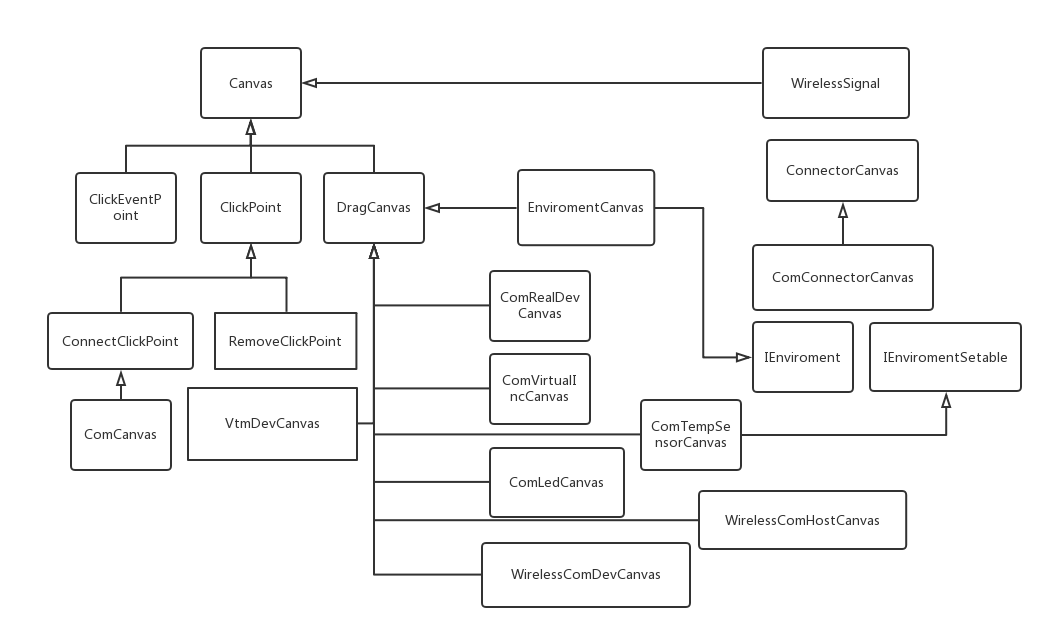
整个系统运行的时序图如下：



3.4 UI交互层

UI交互层主要使用WPF作为解决方案，将抽象层的大部分模块进行了封装，对应于特定的Canvas类，每个Canvas分别持有相应的IDevice对象，从而向用户进行展示。

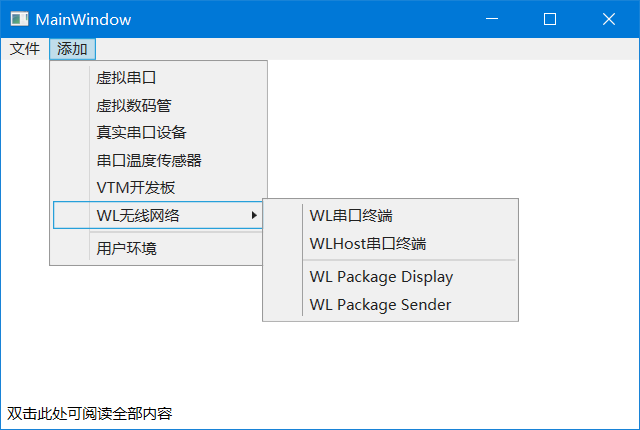
下面列出SimuWindows(UI交互层)的相关类：



4 产品

4.1 物联网仿真实验平台UI

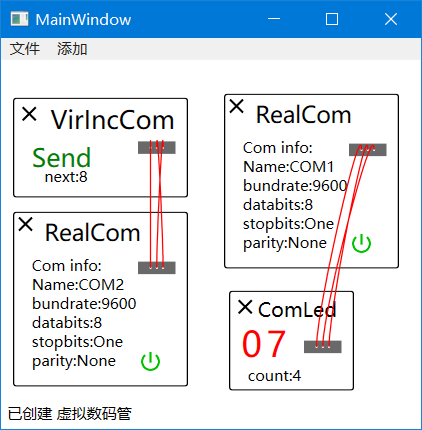
软件直接使用Canvas作为绘制工具，通过嵌套Canvas进行绘制。主窗口UI如下：



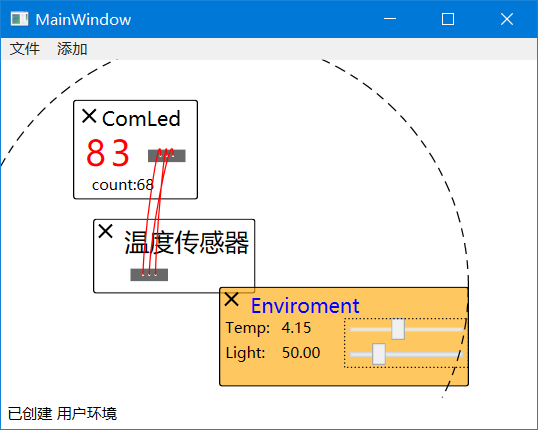
作为演示，创建虚拟数码管与虚拟的串口数据发生装置，连线可反映串口的连接方式：



现在主机上有COM1和COM2两个串口，且在物理上已经将两者所接的RX与TX线互相交叉连接，因此可以实现串口转发：



对于串口温度传感器，可以通过串口获取当前环境温度：

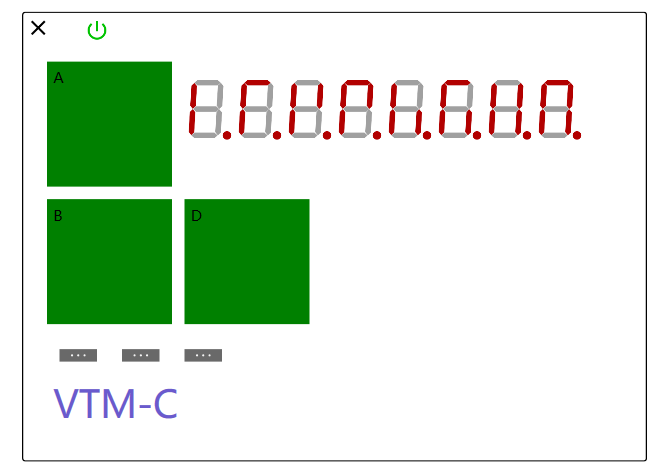


单片机模块：

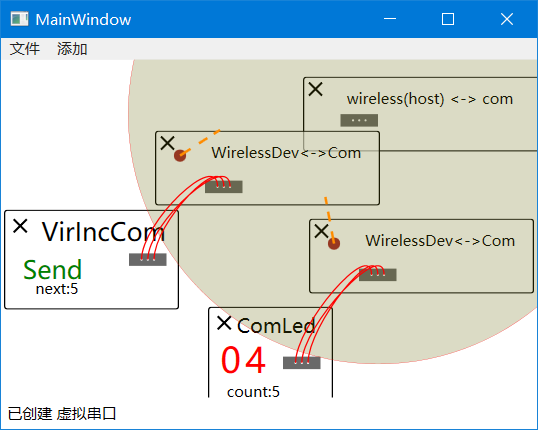
我们对以下C程序运行编译脚本：

|  |
| --- |
| #include "hal\_base.h"  #include "hal\_types.h"  #include "hal\_time.h"  #include "hal\_led.h"  int main(){  while(1){  for(int i=0;i<32;i++){  for(int j=0;j<8;j++){  HalLedSet(j,i\*8+j);  }  HalDelayMs(200);  }  }  return 0;  } |
| 编译输出：  build end  请按任意键继续. . . |

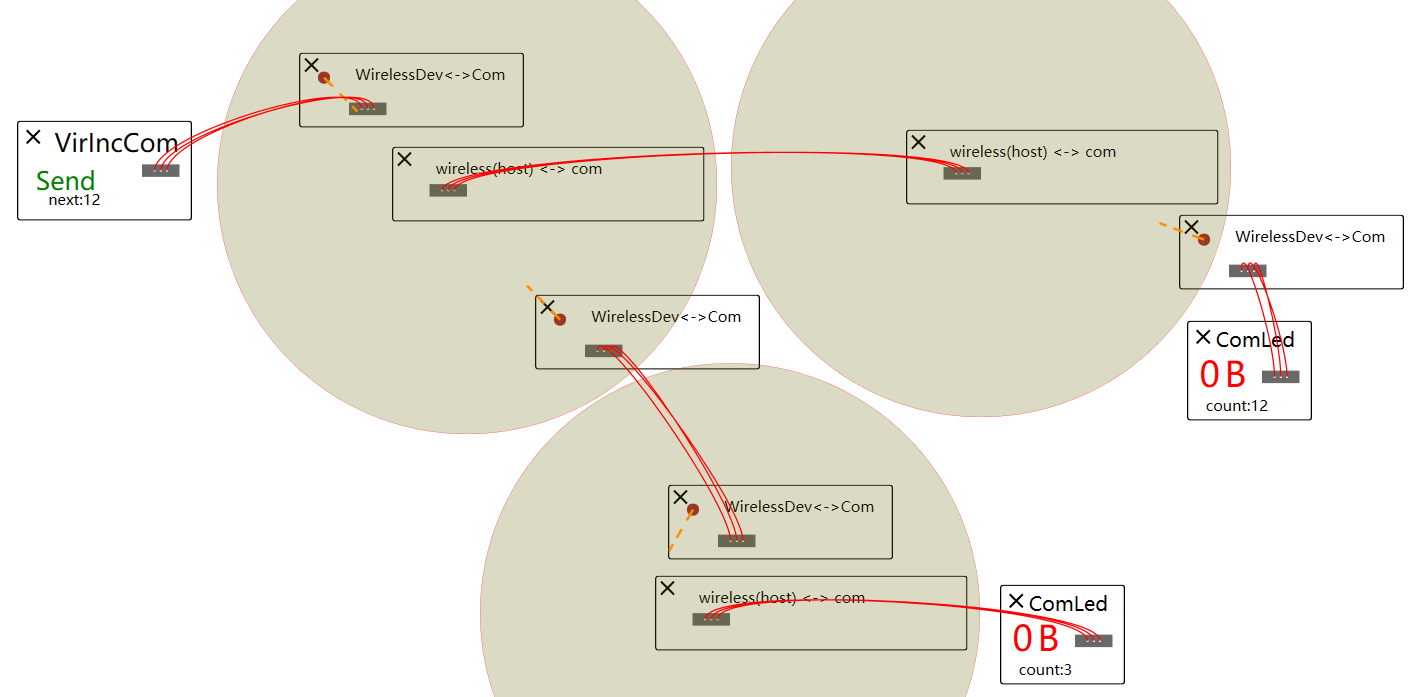
将生成的二进制装载到单片机中运行，加入LED模块并运行，结果如下：



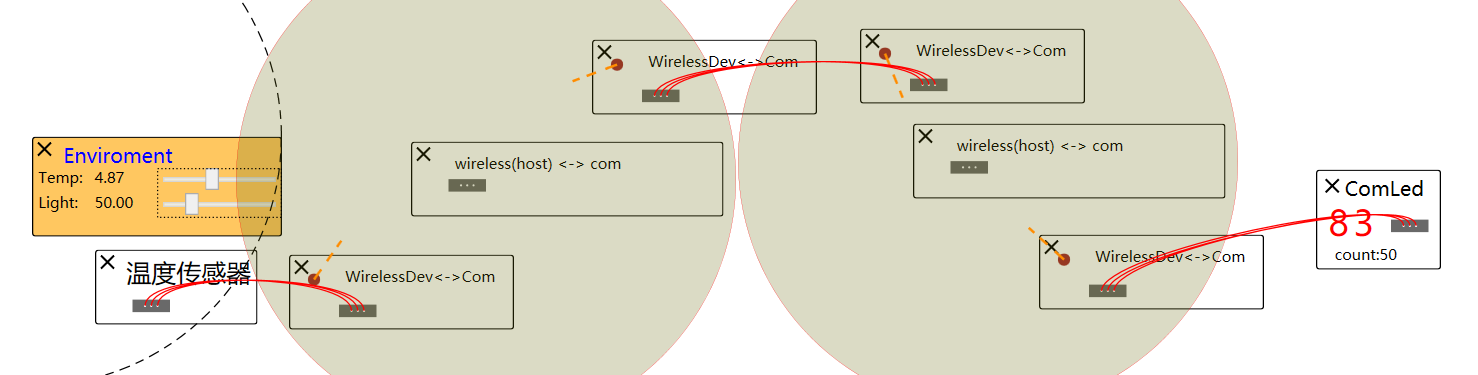
无线网络收发演示：



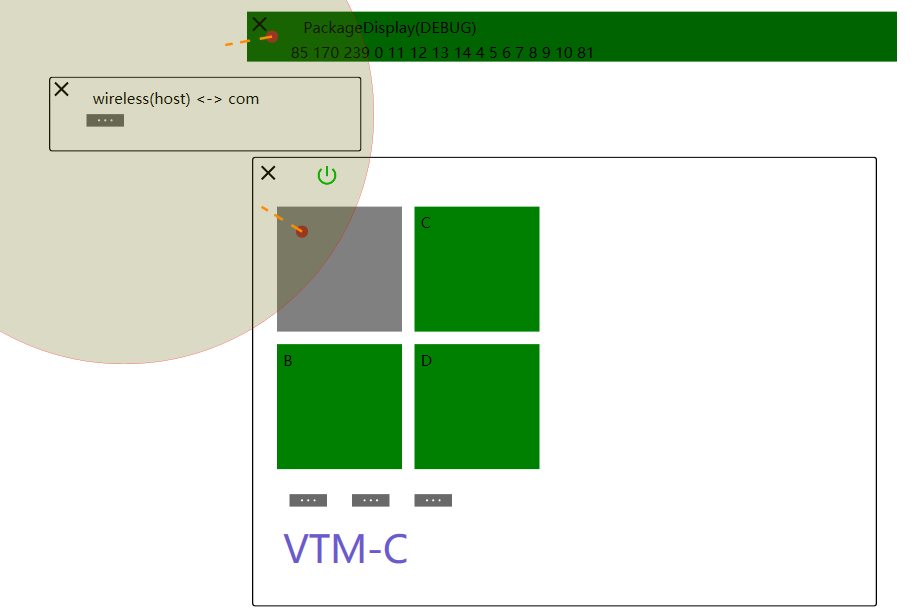
通过串口转发的无线网络：



基于无线转发的的温度传感器：



有单片机参与的无线网络：

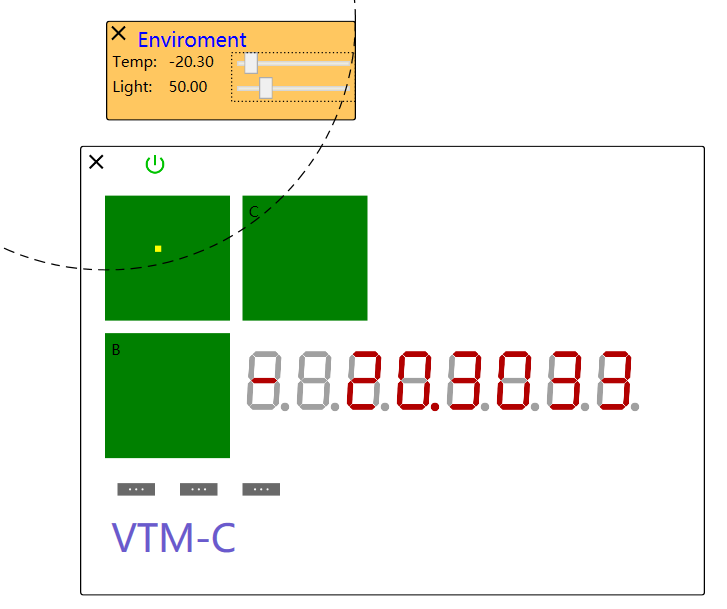


单片机接收环境温度并显示：

程序：

|  |
| --- |
| #include "hal\_base.h"  #include "hal\_types.h"  #include "hal\_time.h"  #include "hal\_led.h"  #include "hal\_sensor.h"  void HalLedNumDot(uint8 led,uint8 value){  const static unsigned char HalLedTable[]={  0x3f,0x06,0x5b,0x4f,  0x66,0x6d,0x7d,0x07,  0x7f,0x6f,0x77,0x7c,  0x39,0x5e,0x79,0x71  };  HalLedSet(led,HalLedTable[value%0x10] | 0x80);  }  void run(){  float tem;  int i;  tem = HalTemGet();    if(tem < 0){  HalLedSet(0,0x40);//-  tem = - tem;  }  else{  HalLedSet(0,0x00);  }  i = (int)tem;  tem -= i;  for(int k=0;k<3;k++){  if(k == 0){  HalLedNumDot(3 - k,i%10);  }  else{  if(i != 0)  HalLedNum(3 - k,i%10);  else  HalLedSet(3 - k,0x00);  }  i/=10;  }  for(int k=4;k<8;k++){  tem \*= 10;  HalLedNum(k,(int)tem);  tem -= (int)tem;  }  HalDelayMs(100);  }  int main(){  while(1){  run();  }  return 0;  } |
| 编译输出：  build end  请按任意键继续. . . |

运行结果：



下图展示一个较复杂的有多个单片机参与的无线网络，图中所有的设备均可通过直接或间接的方式进行交互：

