树莓派开发

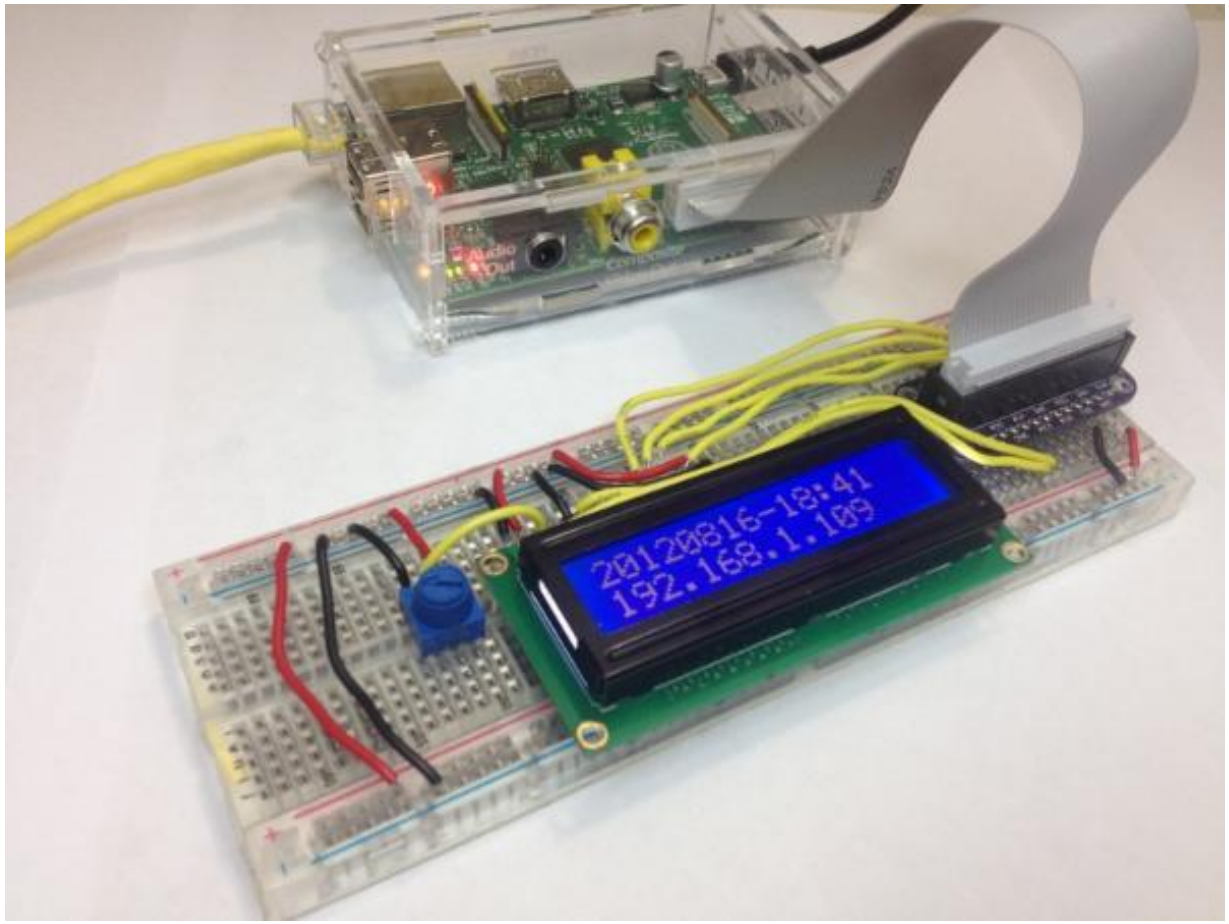
18 树莓派控制LCD液晶屏

一、项目准备

- 本节将详细介绍如何用树莓派的六个通用端口（GPIO）来连接一个廉价的HDD44780的小型LCD。
- 当然也有用I2C或是UART来连接LCD的， 但是使用GPIO是最直接的方法。
- 这种方法的几个优势：
  - 廉价的LCD，可以实现简单的现实应用，而不需要很昂贵的设备
  - 不需要I2C的驱动器
  - 不会占用树莓派仅有的USB口

一、项目准备

- 下图是用Python代码控制显示的时间日期以及IP地址。
- 如果你的树莓派运行在Headless模式下（Headless模式是系统的一种配置模式。
- 在该模式下系统缺少了显示设备、键盘或鼠标），能有个小的显示屏显示IP地址可是很有吸引力的。



一、项目准备

- 以下是完成本次教程的必要硬件
  - 一个标准的16×2的LCD

- ***Adafruit Pi Cobbler*** （树莓派GPIO的扩展设备，这里是以Cobbler为例，当然也可以用树莓派的breakout）

二、连接Cobbler到LCD上

- 任何一个拥有16个引脚的LCD基本上都是用HD44780控制器来控制的。 这种类型的LCD的引脚都拥有相同的输入输出功能，所以比较容易使用。
- LCD采用的是并行接口，这就意味着树莓派需要提供多个引脚来控制它。本项目中我们会用到树莓派的4个数据引脚（4位模式）和两个控制引脚。
- 数据引脚可以直接传输数据到LCD上， 这里我们只让LCD处于写模式，不读取任何数据。

二、连接Cobbler到LCD上

- 寄存器的选择引脚有两种用途。
  - 当设置为低位时，它可以发送指令到LCD（比如显示的位置或是清空屏幕），可理解为命令寄存器。
  - 当设置为高位的时候，它使得LCD转为数据模式并且将数据传输到屏幕上。
  - 读/写引脚在这里会被设置成低位（写模式），因为我们只是想让LCD作为一个输出设备。

三、LCD 各个引脚的定义

01、Ground

02、VCC - **5v not 3.3v**

03、Contrast adjustment (VO) from potentiometer

04、Register Select (RS). RS=0: Command, RS=1: Data

05、Read/Write (R/W). R/W=0: Write, R/W=1: Read (**we won't use this pin**)

06、Clock (Enable). Falling edge triggered

07-10 Bit0 -Bit3 **(Not used in 4-bit operation)**

11-14 Bit4-Bit 7

15、Backlight LED Anode (+)

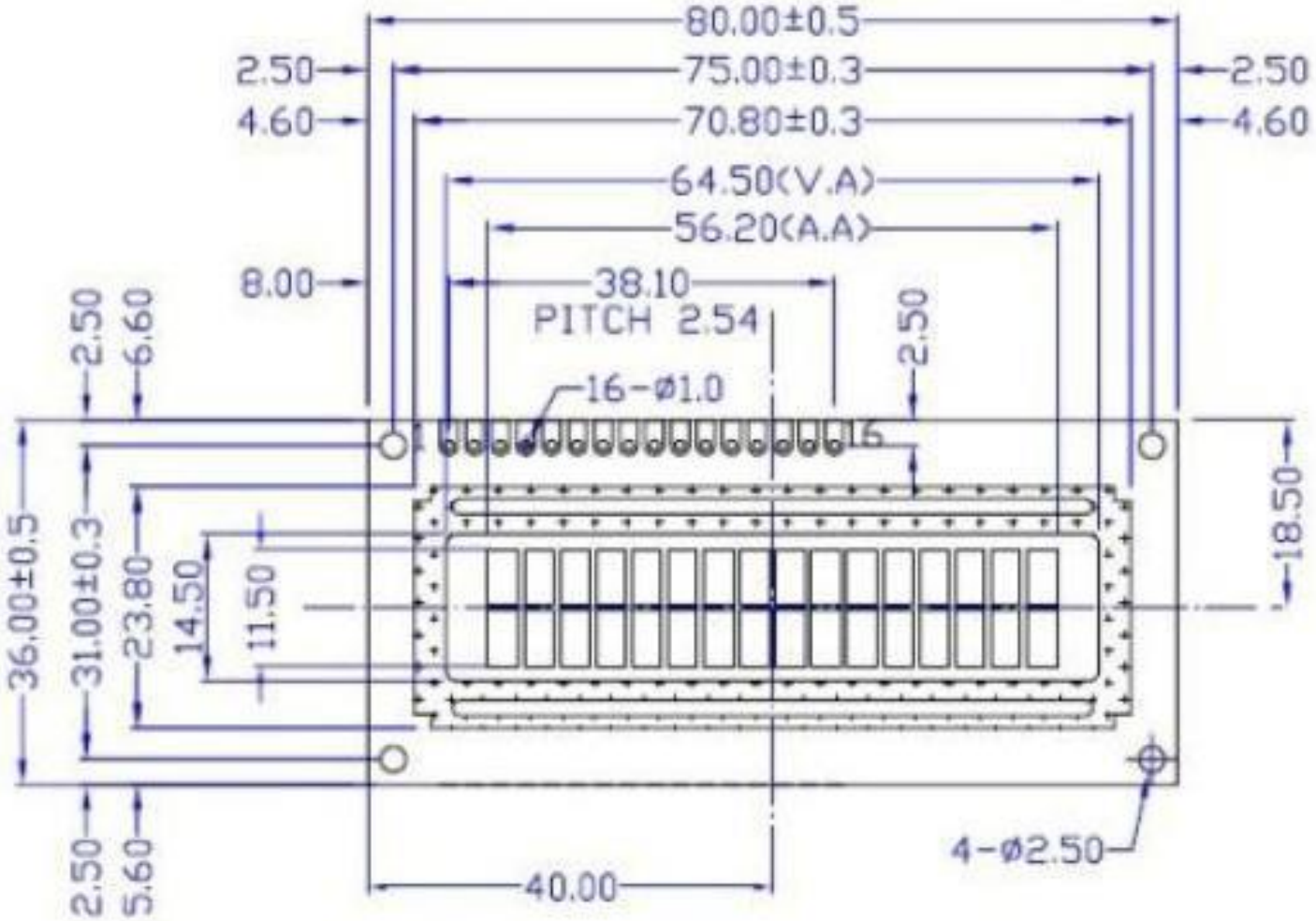
16、Backlight LED Cathode (-)

三、LCD 各个引脚的定义

编号	符号	引脚说明	编号	符号	引脚说明
1	VSS	电源地	9	D2	数据
2	VDD	电源正极	10	D3	数据
3	VL	液晶显示偏压	11	D4	数据
4	RS	数据/命令选择	12	D5	数据
5	R/W	读/写选择	13	D6	数据
6	E	使能信号	14	D7	数据
7	D0	数据	15	BLA	背光源正极
8	D1	数据	16	BLK	背光源负极

三、LCD 各个引脚的定义

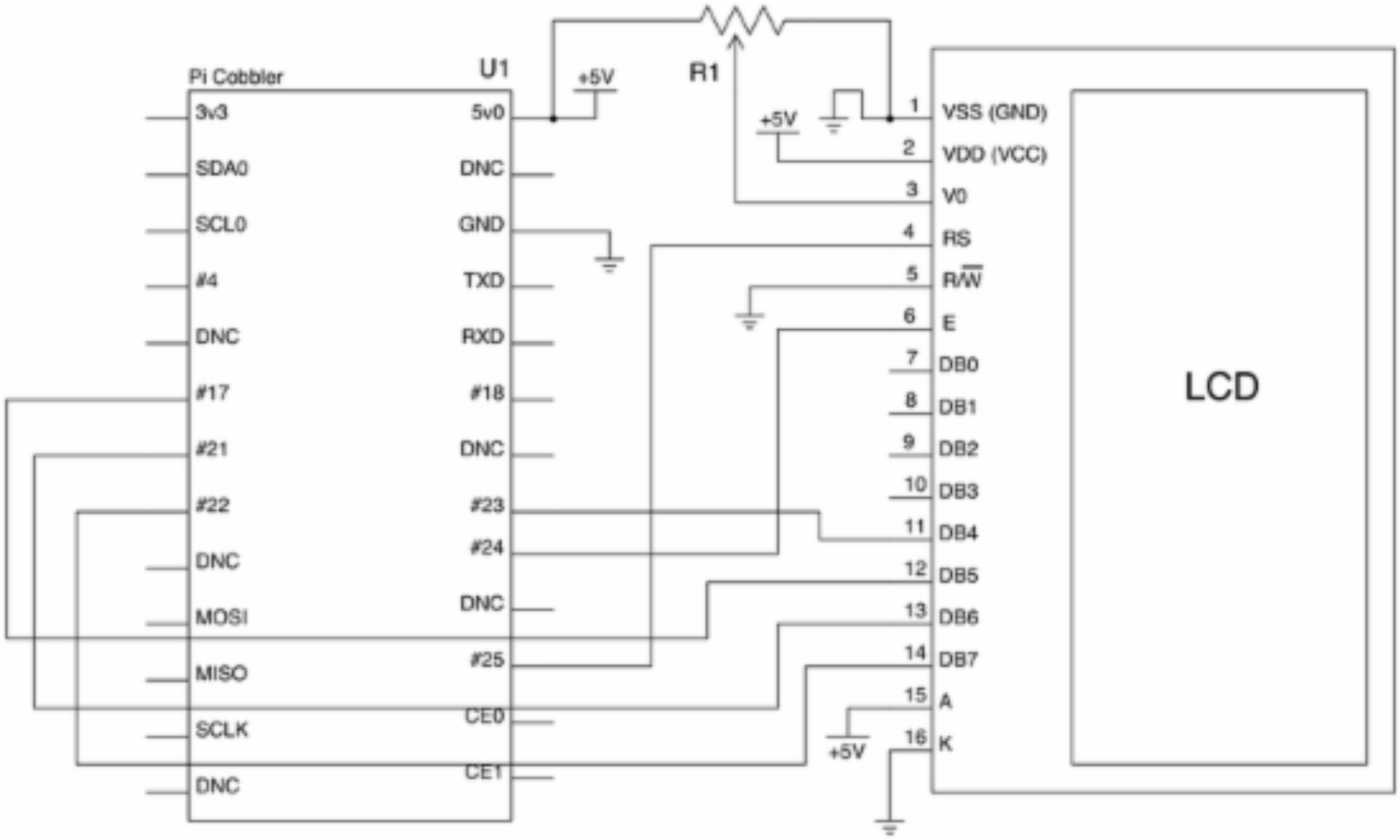
三、LCD 各个引脚的定义



三、LCD 各个引脚的定义

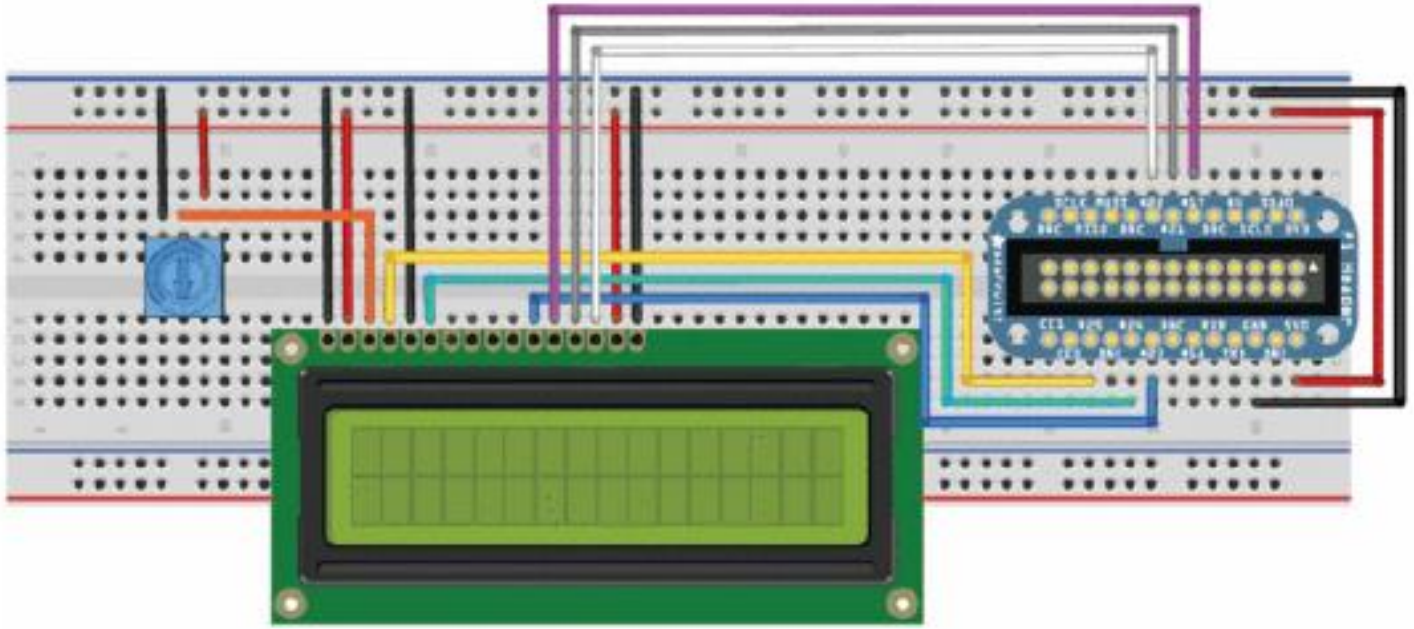
- 在连接这些引脚之前，先确认你的LCD的背光是否可以正常工作，背光应为LED的背光，因为这只需要10－40mA的功率。
- 但是若为EL的背光就需要200mA以上的功率了。EL背光的LCD往往会便宜些但是用起来比较难操作，确保你的LCD不是EL背光，否则会将整个树莓派的功率拖下来。
- 还有一些LCD的LED背光没有自带的稳压电阻，所以在连接前要去确定好你的LCD是否需要加载额外的电阻来保证背光LED正常工作。

四、原理图


Pi Cobbler
U1
3v3
SDA0
SCL0
#4
DNC
#17
#21
#22
DNC
MOSI
MISO
SCLK
DNC
5v0
DNC
GND
TXD
RXD
#18
DNC
#23
#24
DNC
#25
CE0
CE1
+5V
R1
+5V
+5V
1 VSS (GND)
2 VDD (VCC)
3 V0
4 RS
5 R/W
6 E
7 DB0
8 DB1
9 DB2
10 DB3
11 DB4
12 DB5
13 DB6
14 DB7
15 A
16 K
LCD

五、连接

- 首先将Cobber（没有使用Cobber的话，就要接更多的线了）的电源引脚连接到面包板的供电轨上。
- +5V的用红线连接到红线轨上（这里连接3.3V也可以）
- GND用黑线连接到蓝线轨上。



五、连接

为了能使数据传到LCD上，我们将进行以下的连接。

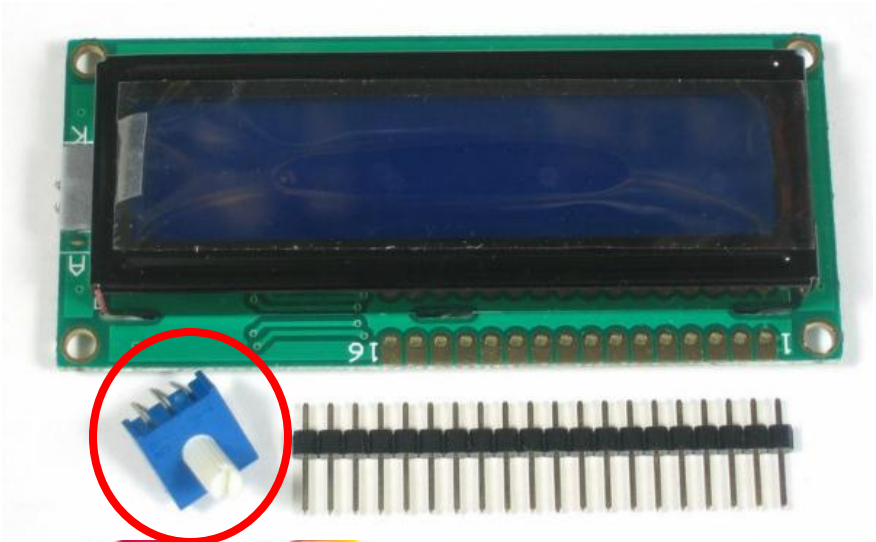
1. LCD的Pin 1脚接地(黑线)
2. LCD的Pin 2脚接 +5V(红线)
3. LCD的Pin 3脚接到分压器的中间位置（橙线）
4. LCD的Pin 4脚接到Cobber的 #25位 （黄线）
5. LCD的Pin 5脚接地（黑线）
6. LCD的Pin 6脚接到Cobber的#24位
7. LCD的Pin 7，8，9，10什么都不接
8. LCD的Pin 11脚接 Cobber的 ＃23位（蓝线）
9. LCD的Pin 12脚接 Cobber的 ＃17位 （紫线）
10. LCD的Pin 13脚接 Cobber的 ＃21位 （灰线）（推荐这里连接＃18位）
11. LCD的Pin 14脚接 Cobber的 ＃22位 （白线）
12. LCD的Pin 15脚接 +5V（红线）
13. LCD的Pin 16脚接地 （黑线）
14. 分压器左边的引脚接地（黑线），右边的引脚接+5V（红线）。

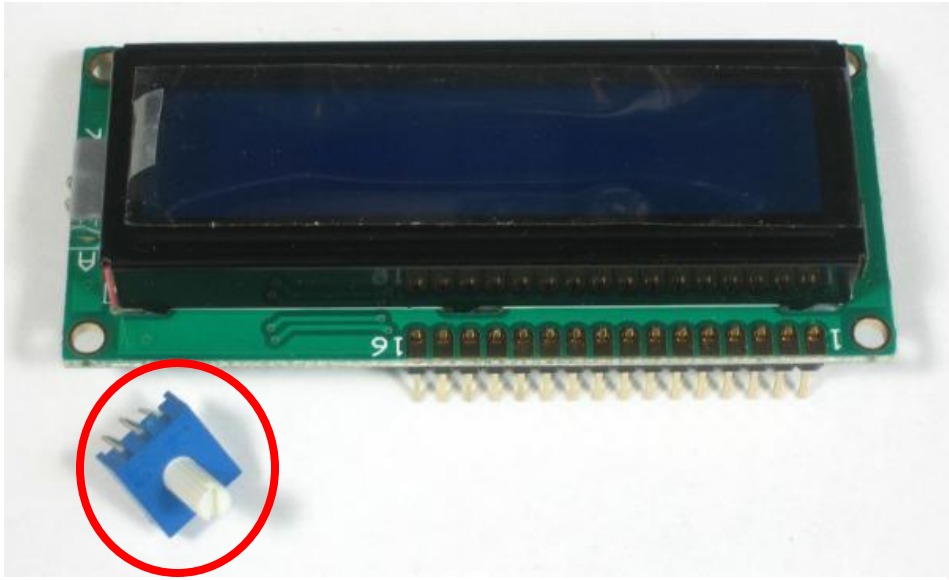
六、有关5V LCD与3.3V Pi的问题

- 树莓派配置的通用接口（GPIO）为3.3V，但是我们的LCD是需要5V配电的设备。
- 如果我们仅仅是用LCD做树莓派的输出设备的话，连接5V的引脚当然没有问题。
- 所以我们这里不使用Cobbler上3.3V的Pin口，并且我们将LCD上的RW（读写）脚接地，这样就避免了LCD向树莓派发送+5V的信号。

七、准备LCD

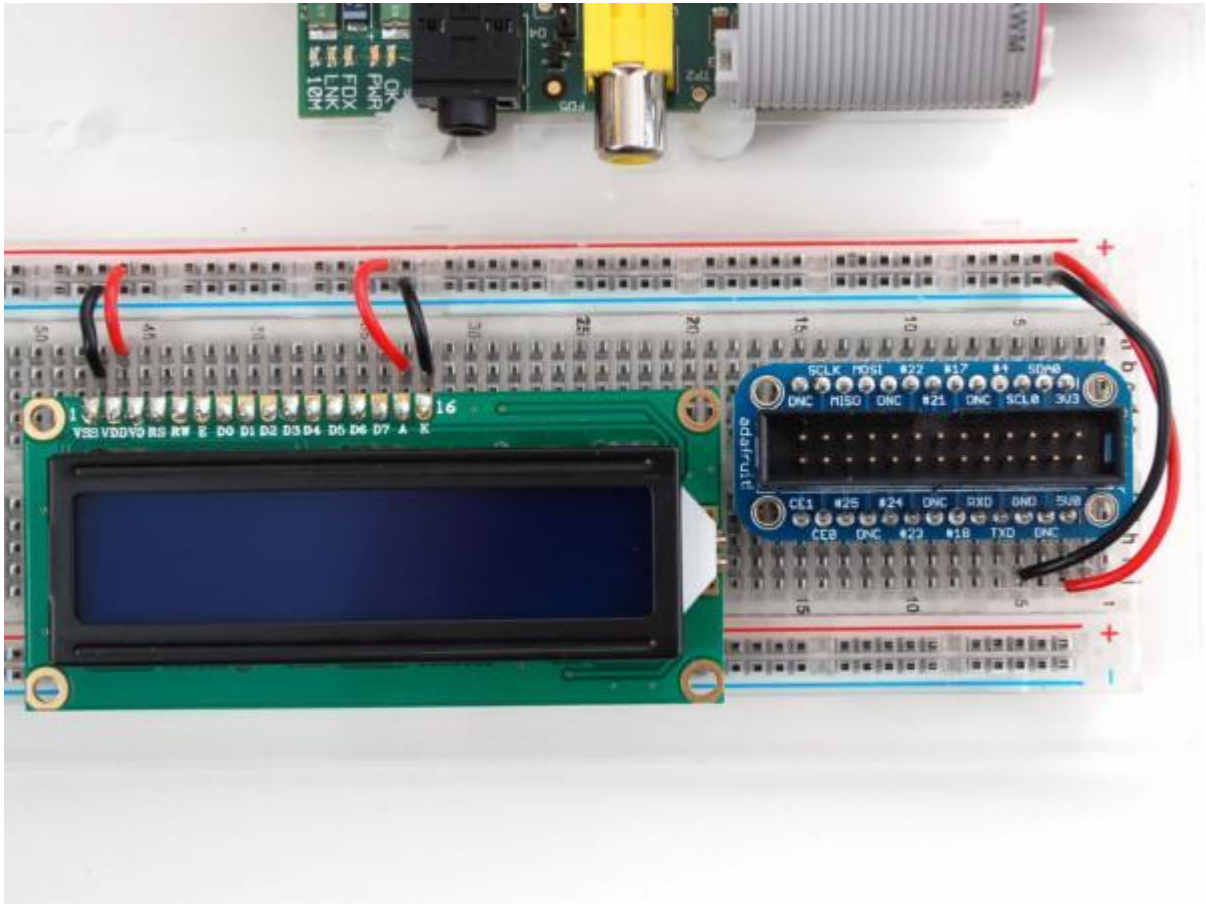
- 在开始前，确认你有一组 16引脚的接线排和一个阻值为10K的分压器（电位器，红圈内）。
- 我们给同学们提供的LCD是焊好16脚引脚的，所以接线排不需要。电位器实际也是可以省掉的，后面介绍。





七、准备LCD

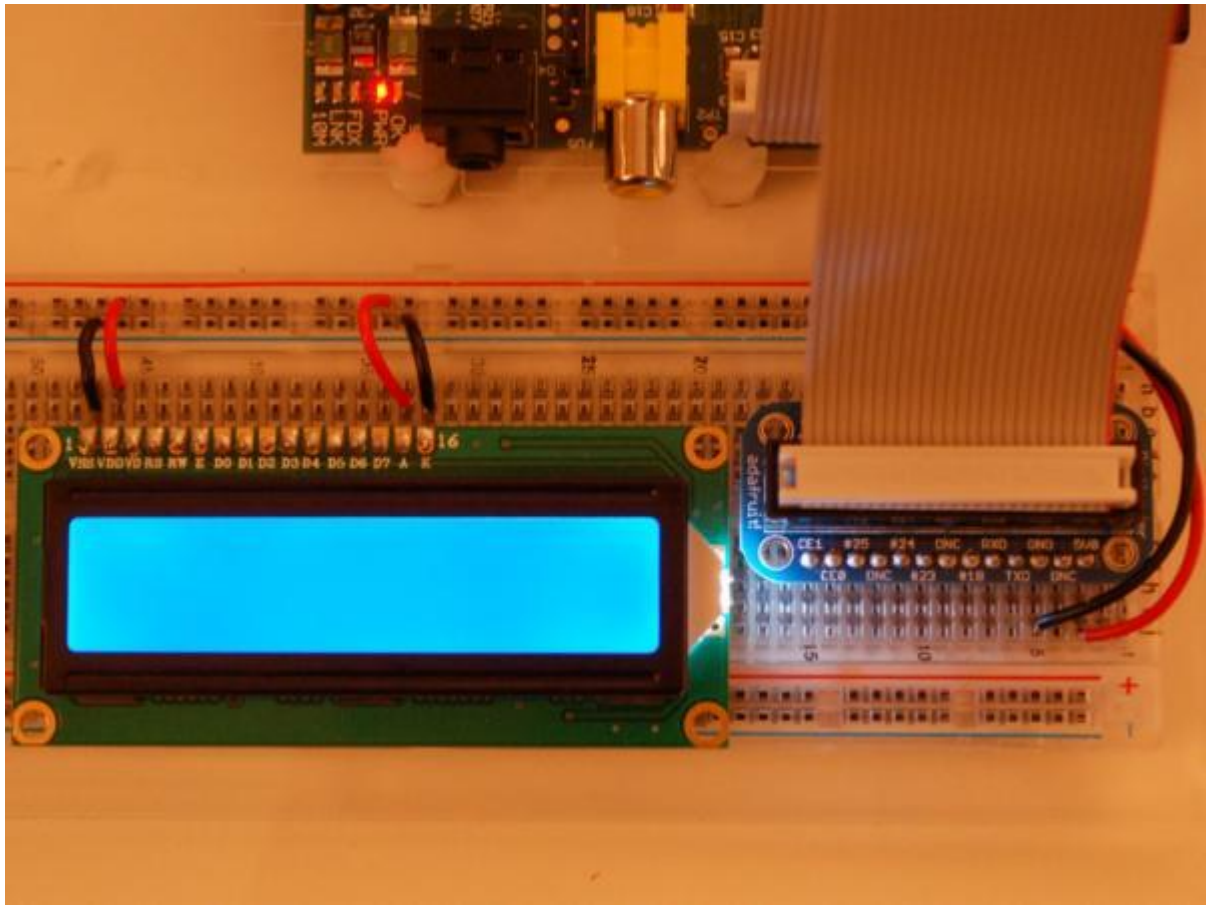
- 首先将Cobbler上的+5V引脚跟GND引脚连接到面包板上。
- 接着如图连接LCD的Pin1脚、Pin2脚、Pin15脚和Pin16脚连接到面包板的供电轨上。
- 这个时候LCD的背光应该就亮了
- 如果没有亮，请检查你的线路是否连接正常。



七、准备LCD

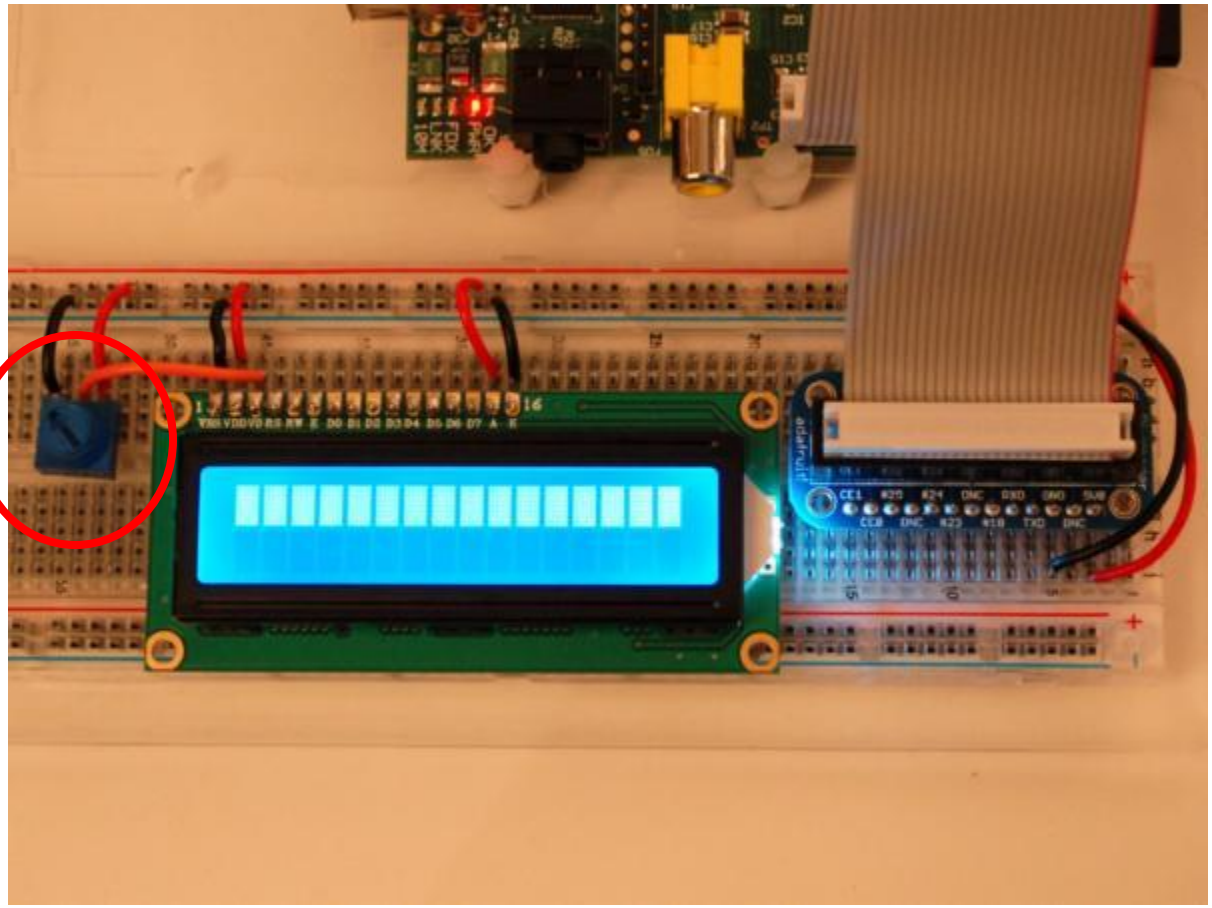
- 首先将Cobbler上的+5V引脚跟GND引脚连接到面包板上。
- 接着如图连接LCD的Pin1脚、Pin2脚、Pin15脚和Pin16脚连接到面包板的供电轨上。

- 这个时候LCD的背光应该就亮了
- 如果没有亮，请检查你的线路是否连接正常。



七、准备LCD

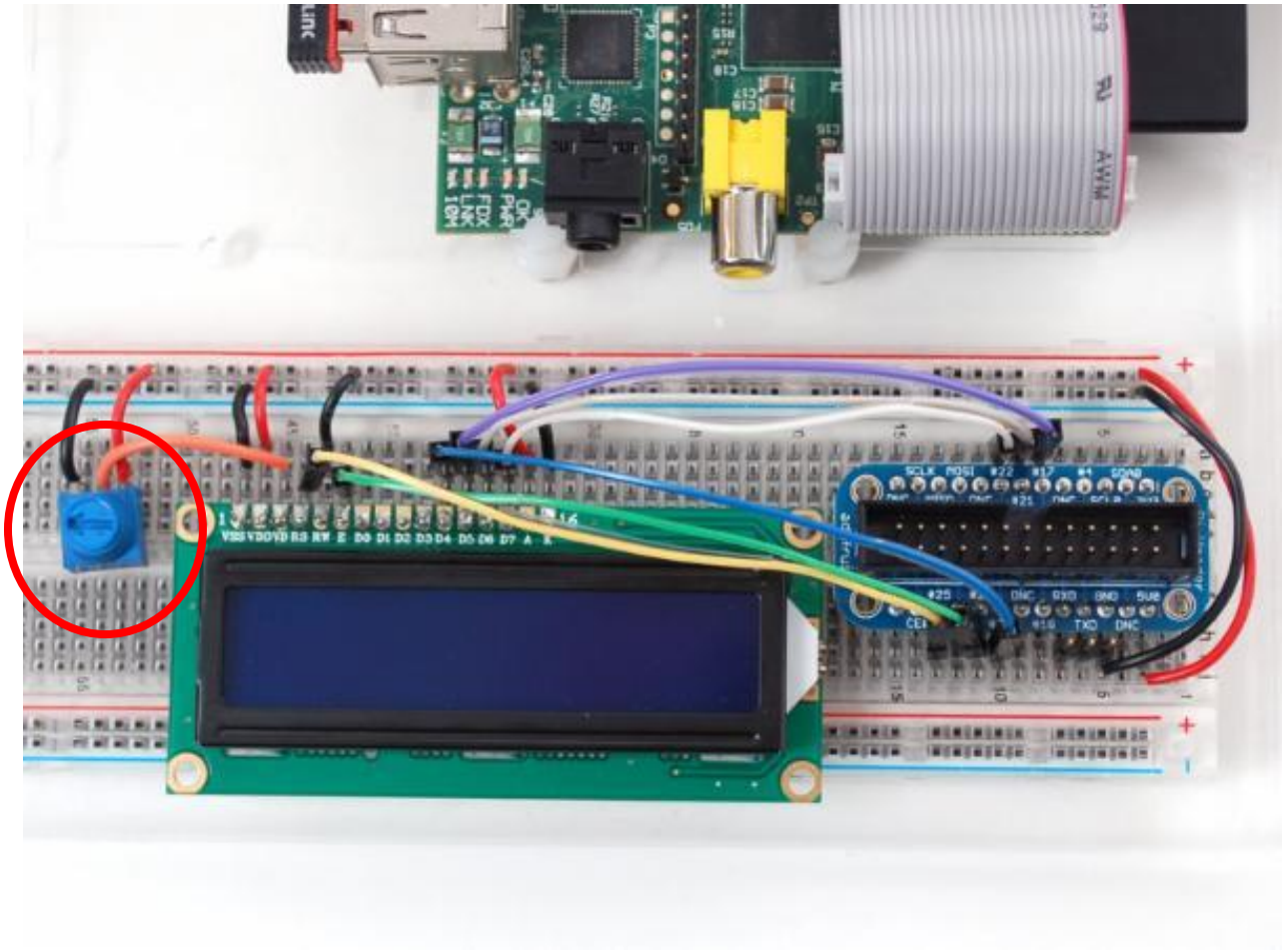
- 接着，将分压器中间的引脚按图中所示连接到LCD的Pin脚3上，其他两个引脚分别连接5V电源和地线。
- 扭动分压器（电位器）直到LCD的第一行显示出方块来。
- 如果看不到，检查一下线路是否连接正确。
- 没有电位器可以拿不同的电阻试验，直到合适的阻值（约10K）。



七、准备LCD

- 按照电路图所示完成LCD最后的接线：

RS（Pin 4脚）
RW（Pin 5脚）
EN（Pin 6脚）
D4（Pin 11脚）
D5（Pin 12脚）
D6（Pin 13脚）
D7（Pin 14脚）



八、准备Python

- 必要的Python包，这是大家已经熟悉的过程。
- 本项目是基于Debian的Wheezy系统的，必须要安装以下组件才能使用树莓派的GPIO口。
- 安装python（2.x）的最新开发套件：

  sudo apt - get install python - dev
- 安装如下组件：

  sudo apt-get install python-setuptools

  sudo easy-install –U distribute

  sudo apt-get install python –pip
- 安装 RPi.GPIO 0.3.1a
- sudo pip install rpi.gpio

九、Python脚本

- 可以在 ***Github*** 获得控制LCD的Python脚本
  https://github.com/lifanxi/rpimenu.git
- 其中包括两个文件：
  - Adafruit_CharLCD.py —该文件中包含用来控制LCD的Python类
  - Adafruit_CharLCD_IPclock_example.py — 样例程序，用来显示IP地址、日期时间。
- 第一个文件Adafruit_CharLCD.py运行后，LCD上会显示两行字符：LCD 1602 Test, 123456789ABCDEF。
- 将代码加载到树莓派上的最简单的方法就是将树莓派连上网络，然后直接通过git的clone命令来下载。

九、Python脚本

- 只要在合适的目录下（比如说/home/pi/）键入以下命令即可：

apt - get install git

git clone http : //github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git

cd Adafruit - Raspberry - Pi - Python - Code

cd Adafruit_CharLCD

十、测试

- 现在你就可以测试之前连接好的线路了，只要简单运行Python代码Adafruit_CharLCD.py即可。
- 因为这里的代码很少，它只会简单的显示出一段测试消息。
- 无论你使用的是什么型号的树莓派，这里建议大家将引脚21替换换为引脚18， 所以这里要对 Adafruit_CharLCD.py做一个小小的改动，将
  def __init__ (self , pin_rs = 25 , pin_e = 24 , pins_db = [23 , 17 , 21 , 22] , GPIO =None) :
- 修改为：
  def __init__ (self , pin_rs = 25 , pin_e = 24 , pins_db = [23 , 17 , 18 , 22] , GPIO =None) :
- 按照上述过程进行试验，整体进行很顺利，提醒一下，可以使用树莓派的Pin #18口，而不是测试代码提供者使用的#21或者#27。Raspberry Pi有Rev 1和Rev 2两个版本，它们对于PIN 13的定义是不同的。市面上现在大部分都是Rev 2版本，PIN 13对应GPIO 27。如果你的RPi是老的Rev 1版本，PIN 13对应是GPIO 21，你需要调整程序中的参数，把27改为21。

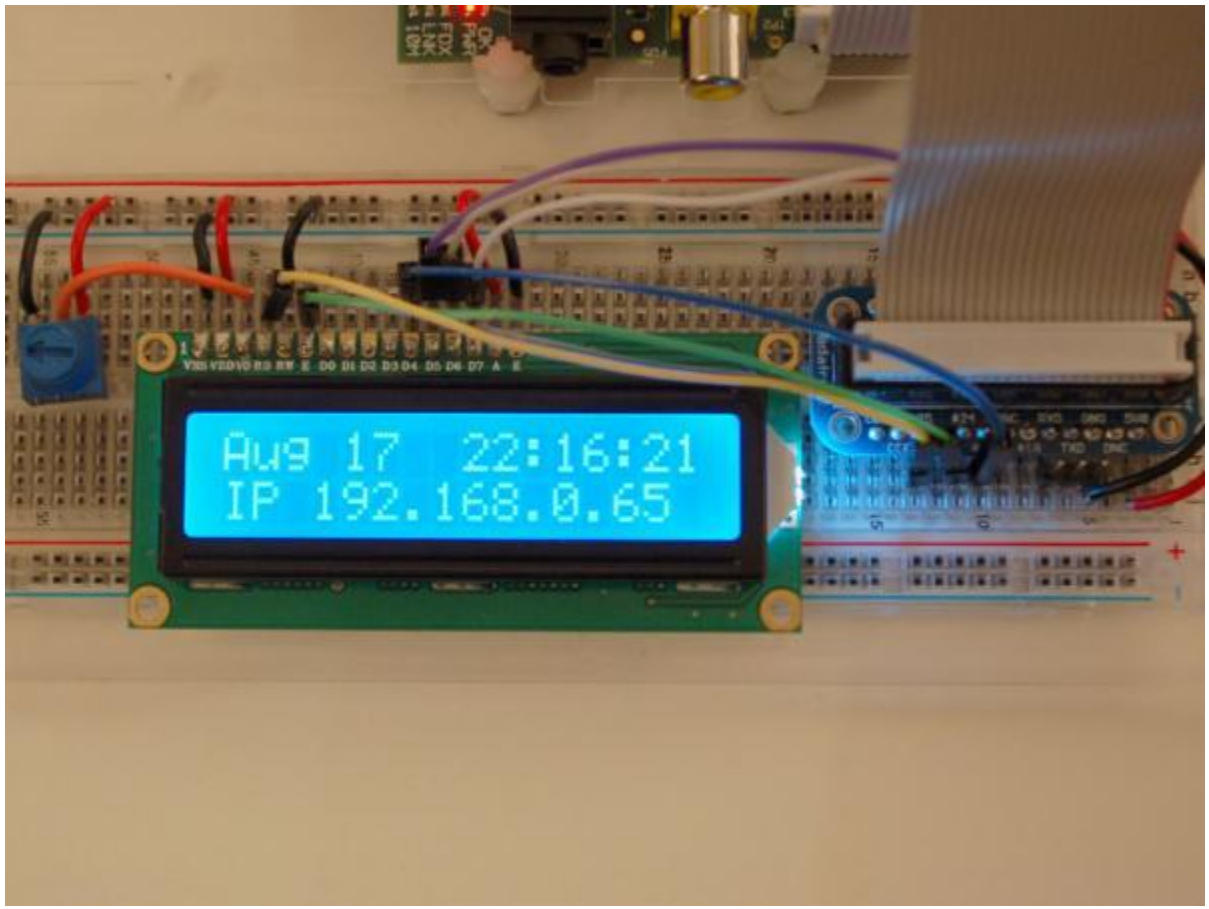
十一、IP和时钟的显示的代码

- 以下脚本的功能是显示你的IP地址，若想显示无线接口的IP地址，请将代码中的eth0替换为wlan0或者wlan1即可。

```
#!/usr/bin/python
from Adafruit_CharLCD import Adafruit_CharLCD
from subprocess import *
from time import sleep , strftime
from datetime import datetime
lcd = Adafruit_CharLCD ( )
cmd = "ip addr show eth0 | grep inet | awk '{print $2}' | cut -d/ -f1"
lcd . begin ( 16 , 1 )
def run_cmd ( cmd ) :
    p = Popen ( cmd , shell = True , stdout = PIPE )
    output = p . communicate ( ) [ 0 ]
    return output
while 1 :
    lcd . clear ( )
    ipaddr = run_cmd ( cmd )
    lcd . message ( datetime . now ( ) . strftime ( '%b %d  %H:%M:%S\n' ) )
    lcd . message ( 'IP %s' % ( ipaddr ) )
    sleep ( 2 )
```

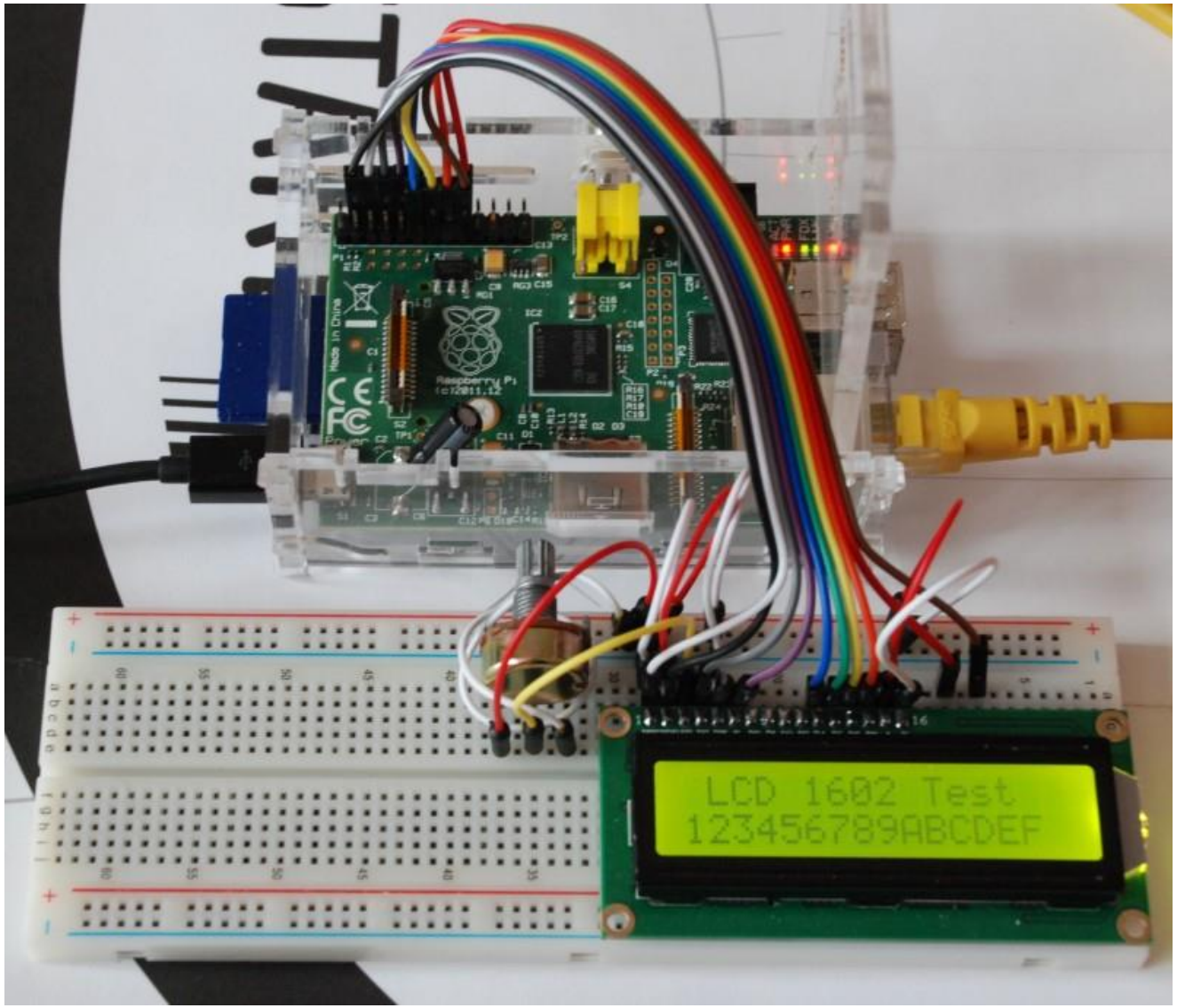
十二、运行代码

- 运行代码很简单，直接输入下列命令即可。
  sudo . / Adafruit_CharLCD_IPclock_example . Py
- 注意脚本的权限问题，可用chmod +x命令修改为可执行。
- 执行结果如图：



十二、运行代码

- 不使用Cobbler的效果



十三、让程序开机自启动

- 让树莓派启动时，都能自动运行这个Python程序
- 下面我们将设置 Adafruit_CharLCD_IPclock_example.py为开机自启动，而在关机时会自动关闭。
- 将下段代码粘贴到 /etc/init.d/lcd，注意，需要root权限才能在这个目录下执行写操作。
- 你需要相应的将路径修改为你实际保存该脚本的路径才行。修改初始化脚本的执行权限：

  sudo chmod + x / etc / init . d / lcd
- 用update-rc.d命令使系统感知lcd初始化脚本

  sudo update - rc . d lcd defaults
- 现在每次启动时，lcd也会自动启动并显示出系统的时间和IP地址到屏幕上。这样你就可以在不用屏幕显示器的情况下知道树莓派的IP地址以及何时可以连接上它。

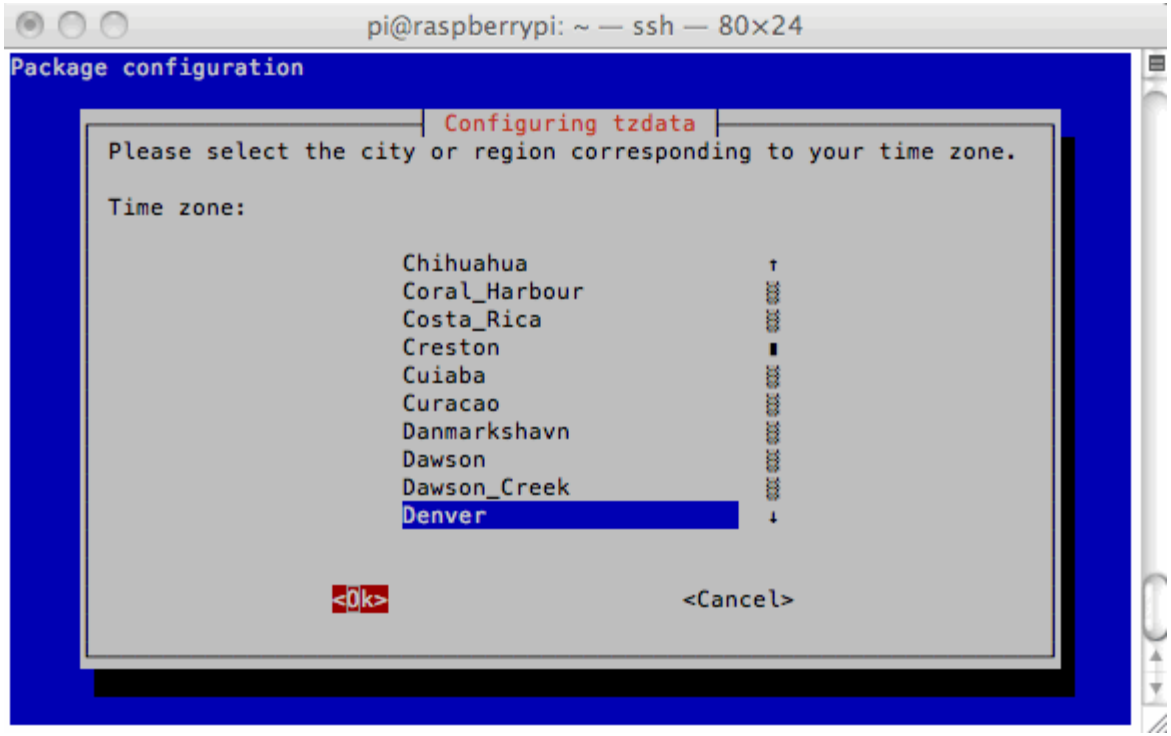
```
### BEGIN INIT INFO
# Provides: LCD - date / time / ip address
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Liquid Crystal Display
# Description: date / time / ip address
### END INIT INFO
#! /bin/sh
# /etc/init.d/lcd
export HOME
case "$1" in
     start )
          echo "Starting LCD"
          / home / pi / Adafruit - Raspberry - Pi - Python -
 Code / Adafruit_CharLCD /Adafruit_CharLCD_IPclock_example .py     2 > & 1 &
    stop )
          echo "Stopping LCD"
     LCD_PID = ` ps auxwww | grep Adafruit_CharLCD_IPclock_example .py | head-
 1 | awk '{print $2}' `
     kill - 9 $LCD_PID
     * )
          echo "Usage: /etc/init.d/lcd {start|stop}"
          exit 1
esac
exit 0
```

十四、时区

- 最后但也是最重要的是：我的树莓派是按世界统一时间（UTC）配置的，但是我想让它显示出我所在的本地时间。
- 以下命令可将树莓派设定为任意时区的本地时间，这个命令是一次性的，一旦完成设定，重启之后也不会失效。

```
sudo dpkg - reconfigure tzdata
```

- 指令输入之后会转到一个选择时间域的程序，下移光标选择你所在的时区就可以了。



十五、显示其他内容

- 如果你想让液晶屏显示些别的东西，可以参考
  - lcdmenu.py代码
  - 和Adafruit_CharLCD.py