2017

自动灌溉系统

北京市第五十七中学

陆君睿

树莓派与自动灌溉系统

摘　要

在我们的校园中，有各种各样的植物与我们相陪伴。正所谓“养花容易护花难”在寒暑假的时间，面临着缺少人来看管，开学的时候植物枯死的问题。面对这样一个在每个班级中普遍存在的问题，我便萌生了一个创造一款可以自动，定时定量并且适用于不同种类植物的浇花装置，并且开始实施。

关键词： 智能；节能；自动化；树莓派；单片机

目　录

第一章　引言

现如今在我们的身边无不充斥着高科技，全自动化的设备使我们的生活变得更加便利。

本课题主要描述我用当前比较热门的物联网设备“树莓派”为装置的中央处理器，通过细致的编程和其余元件来完成寒暑假无人值守时的植物浇水问题，并从此扩展到可以为整栋楼的植物或大面积的花园的自动灌溉系统。

第二章　系统结构及原理

3.1 系统结构

3.1.1　YL-69 土壤检测器

将土壤检测器插入土壤中，即可获取当前土壤的信息。

获取到的信息将通过 DOUT 接口（见图一）传入树莓派，即可使树莓派得到土壤湿度是否符合标准的信息（具体原理见 3.2）。

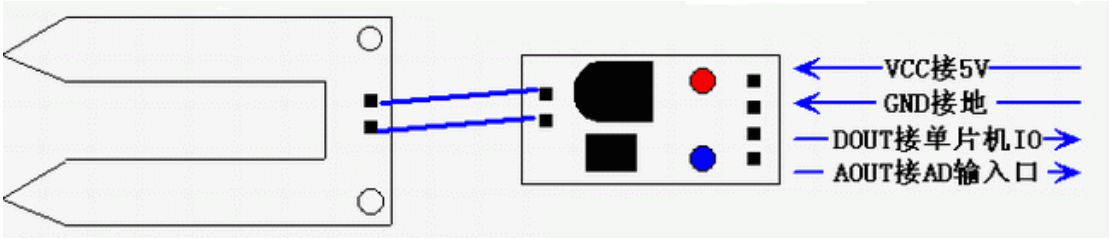


图 1

3.1.2　L298N 电机驱动模块

这是 L298N 的图片（见图 2）：

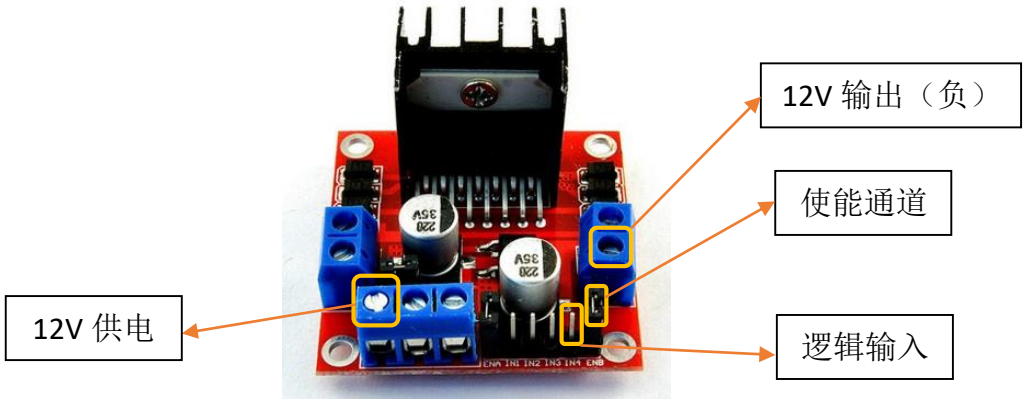


图 2

由图，最左侧的接口是外部给驱动供电所用；逻辑输入和使能则是接入树莓派的输出口，逻辑输入并始终保持低电平，两个使能端若接受高电平则电机开始工作，反之则停止工作。这样便让程序可以控制电机运作了。

3.1.3 直流 12V 水泵

我使用的直流 12V 水泵是市面上比较常见的水泵连接简单，功率足够大，价格便宜是良好的灌溉工具。

下面是水泵的简图（见图 3）：

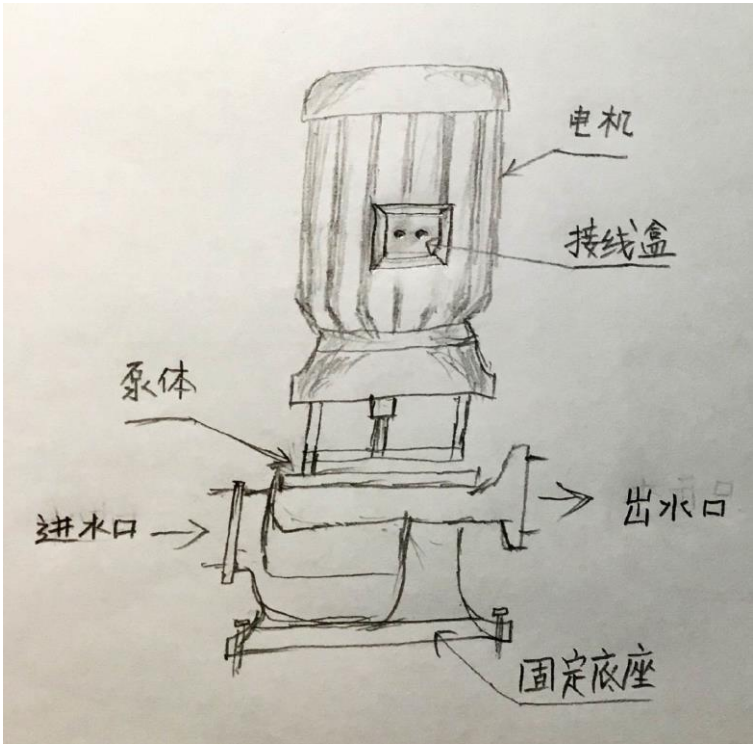


图 3

3.1.4 摄像头

摄像头（见图 4）可以用来记录植物浇水后的情况，让使用者可以看到植物的状态，并确认程序正常工作，植物健康生长。

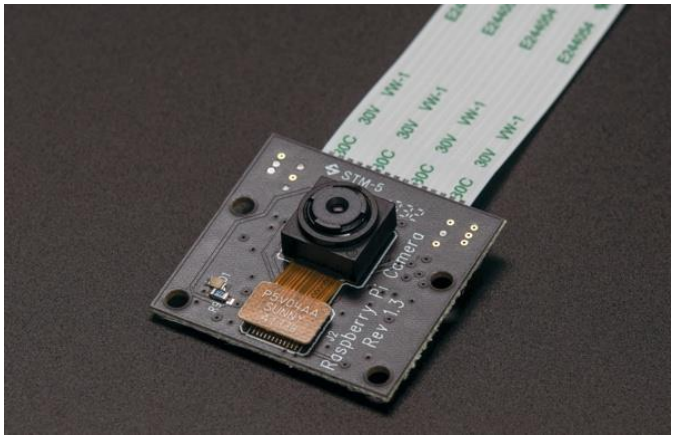


图 4

3.2 系统原理

下面是目前整个系统（忽略开始和终止）的流程图（见图 5）：

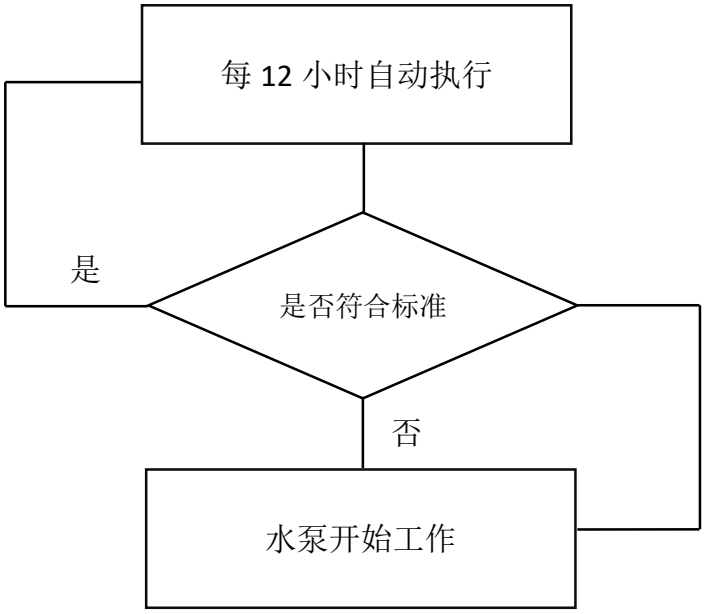


图 5

其实系统流程很简单，下面是主体代码及解释：

```
def watering_on():
    rp.output(13,rp.LOW)
    rp.output(37,rp.HIGH)
    rp.output(40,rp.HIGH)
def watering_off():
    rp.output(37,rp.LOW)
    rp.output(40,rp.LOW)
```

定义两个函数：通过控制在 3.1.2 中提到的两个使能接口（在 37 和 40 口上），及逻辑输入接口（接在 13 口上）的高电平（HIGH）或低电平（LOW），分别控制浇水的开始和结束。

```
def take_picture_write_logfile():
    a =
time.strftime('%Y-%m-%d-%H:%M:%S',time.localtime(time.time()))
    call(['raspistill -o YourFlower'+ a +'.png -q 100 -e png -
n'],shell = True)
    msg = "echo '" + a + "   Watering proccess has been complete
successfully!' >> message.log"
    call([msg],shell = True)
```

定义函数：获取当前时间作为文件名，并用摄像头拍摄照片记录在系统的根目录下（/root），并将程序成功执行的信息写入 message.log 日志文件中。

```
def clean():
    rp.cleanup()
```

清理引脚信息。

至此函数定义结束。

```
if __name__ == '__main__':
    rp.setwarnings(False)
    clean()
    start()
```

程序入口点并执行基本函数。

```
Cin = [12]
Cout = [13,37,40]
init_out(Cout)
init_in(Cin)
```

定义两个参数列表作为输入或输出接口，并执行函数。

```
active = True
count = 0
while active:
    i = rp.input(12)
    if i == 1:
        if count == 0:
            watering_on()
            print('The watering proccess is running at the moment!')
            count +=1
            continue
        if count != 0:
            continue
```

程序从 12 引脚（3.1.1 中土壤检测器接入的口），并判断值是否等于 1，如果为真，那么执行函数，开始浇水，并打印“程序正在运行!”

```
        if i == 0:
            watering_off()
            print('Proccess has been completed!\nGetting
picture...\nWriting into logfile...')
            take_picture_write_logfile()
            print('Done!')
            active = False
    clean()
```

浇水过程中不停检测，直到其值等于 1，执行函数，停止浇水，打印“程序执行完毕!”并且为植物拍照，写入日志。最后的 clean 就表示整个系统的结束。

以上就是这个系统的主体 Python 脚本，为了优化内存使用，计时器被从程序中剥离出来，在系统命令行下执行：

```
crontab –e
1 /12 * * * python3 SF_last.py
```

后一句命令的意思就是每隔 12 个小时的第 1 分钟执行根目录下的 SF_last.py，也就是上面所说的脚本。

我们还可以通过 SSH 或者 VNC 用手机或者电脑去手动执行脚本；用 Filezilla 查看记录下来的照片。

第三章　思考和改进

虽然这个作品可以实现基本的自动灌溉，但是还是存在诸多问题等待改进，例如：

（1） 成本略高

（2） 适用范围不广

（3） 可维护性不强

（4） 功能单一

（5） ……

针对这些问题目前有一个改进方案：添加高级别服务器，从网络上抓取各种各样植物的需水量和需水周期，并保存至其数据库中，各终端通过摄像头拍摄植物形态，上传给服务器，经服务器识别植物种类后从数据库中调取相应数据，并制定相应浇水计划，定时将浇水的的信息回传给终端，其中央控制器改为成本更低的单片机，仅执行简单控制水泵及检测湿度命令，并将执行结果无线传给服务器，照片和条目储存在服务器下，服务器再将其实时数据下发至安卓或苹果等手机终端，这样就加强了系统的实时监控能力。

通过以上手段就可以实现规模化，同时也可以降低成本，可对整栋楼的绿植或农业灌溉使用，其流程（忽略开始和终止）基本如下（见图 6）：

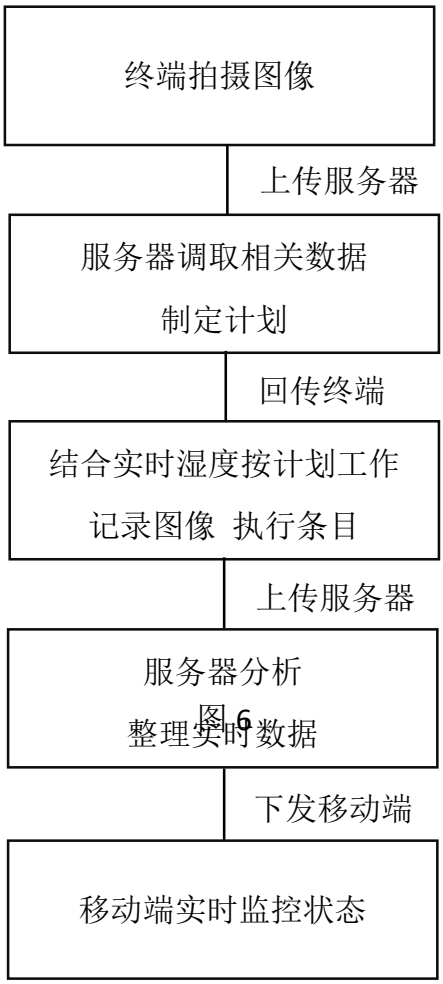

终端拍摄图像
上传服务器
服务器调取相关数据
制定计划
回传终端
结合实时湿度按计划工作
记录图像 执行条目
上传服务器
服务器分析
整理实时数据
下发移动端
移动端实时监控状态

图 6

图 6

致　谢

本课题得以完成离不开同学和家长帮助，也离不开老师精神上的支持和鼓励，在此对在研究过程中提供支持的老师、同学和家长表示衷心的感谢！

参考文献

[1] RPi.GPIO 使用手册

http://blog.csdn.net/qq_35893742/article/details/53428679 2016.12.2

[2] 沃尔弗拉姆·多纳特 Python 树莓派编程 机械工业出版社 2016.10

[3] http://bbs.elecfans.com/jishu_576648_1_1.html 2016.4.3

[4] http://www.chuangke.com/a/mokuaiku/dianjimokuai/2015/1112/197.html

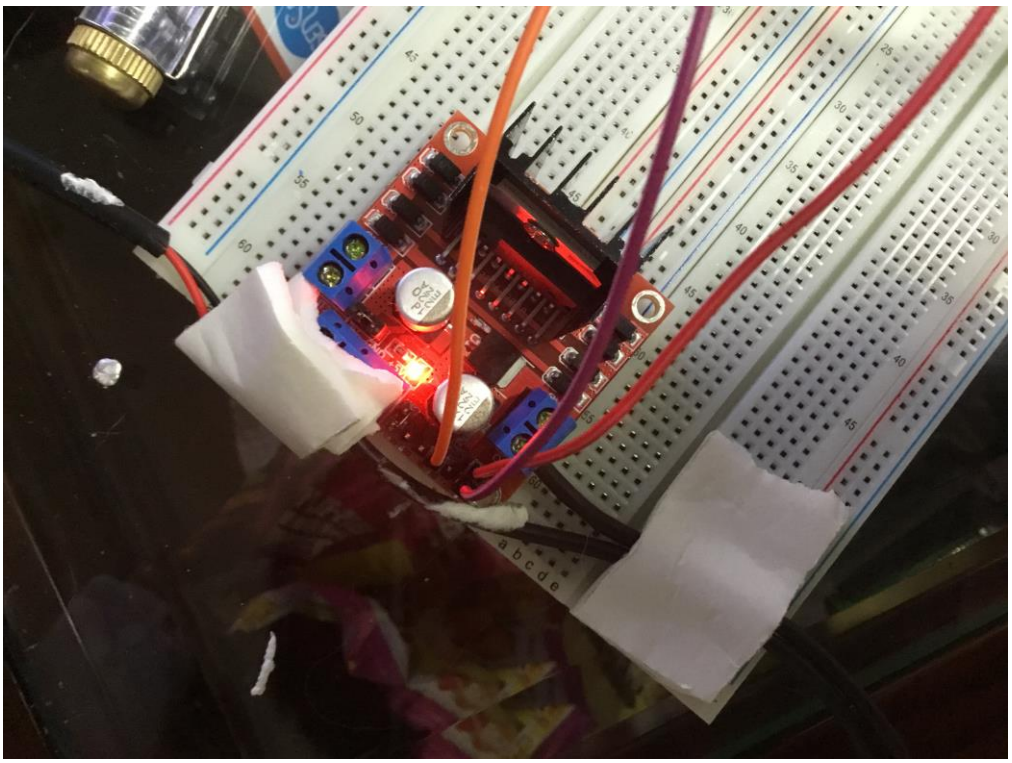
2015.11.12

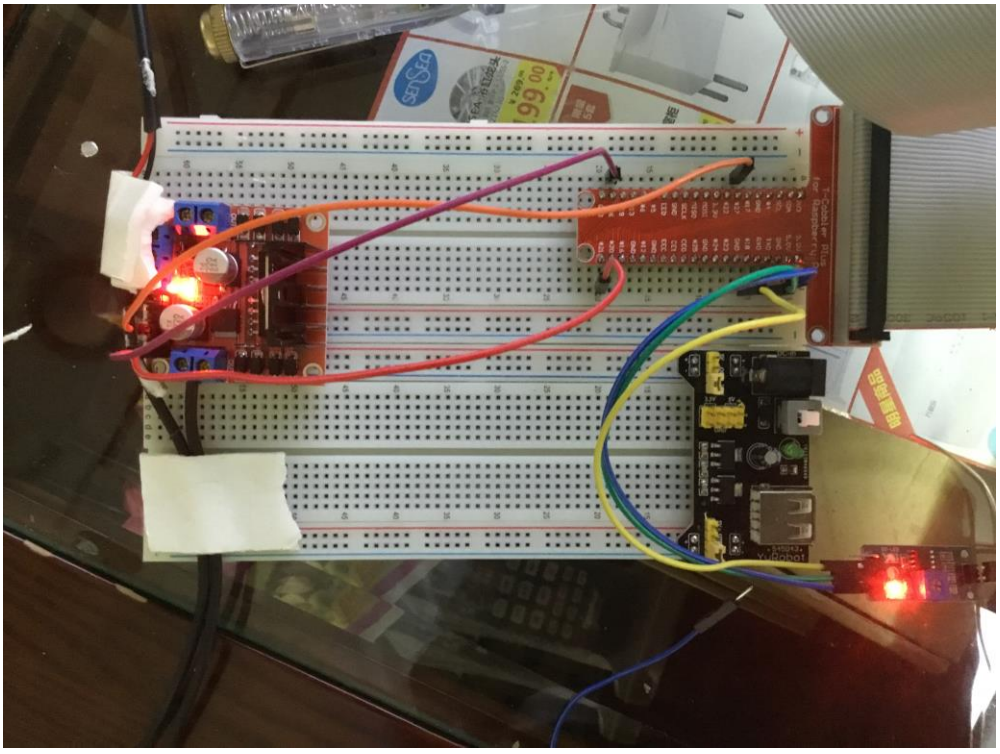
[5] 树莓派摄像头模块应用程序文档翻译

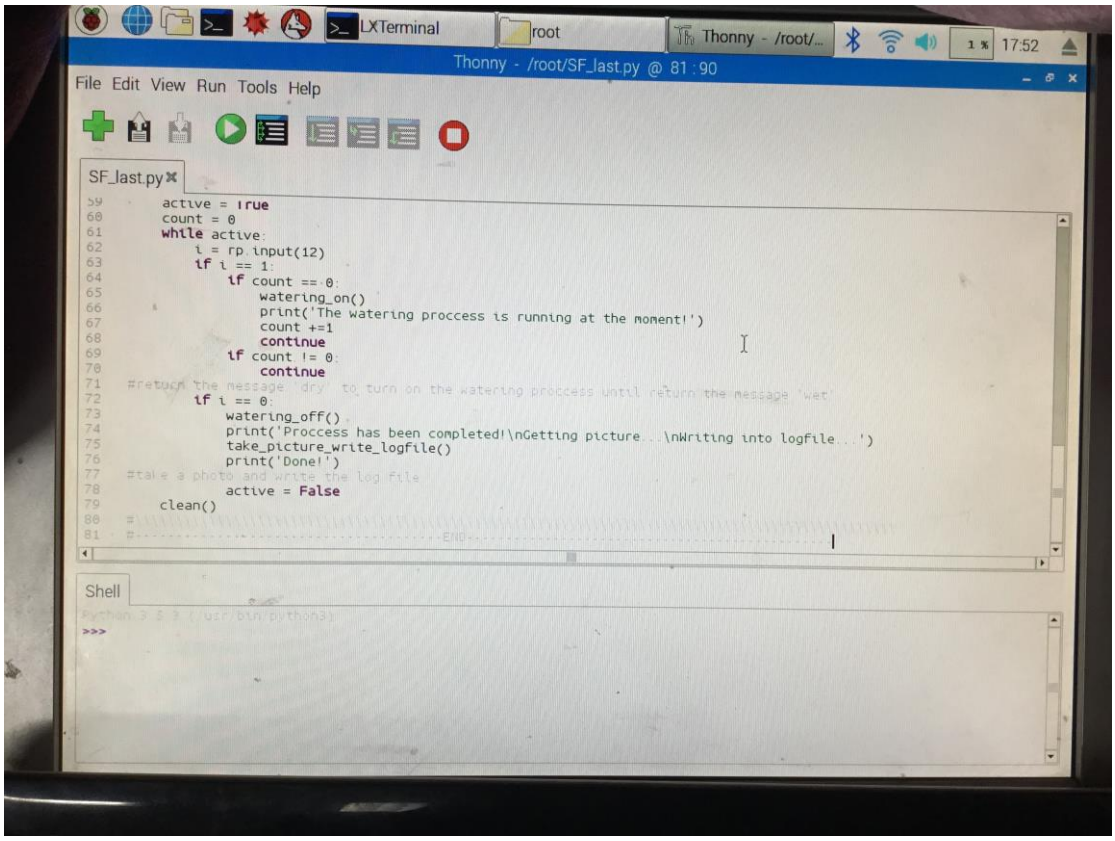
http://shumeipai.nxez.com/2014/09/21/raspicam-documentation.html

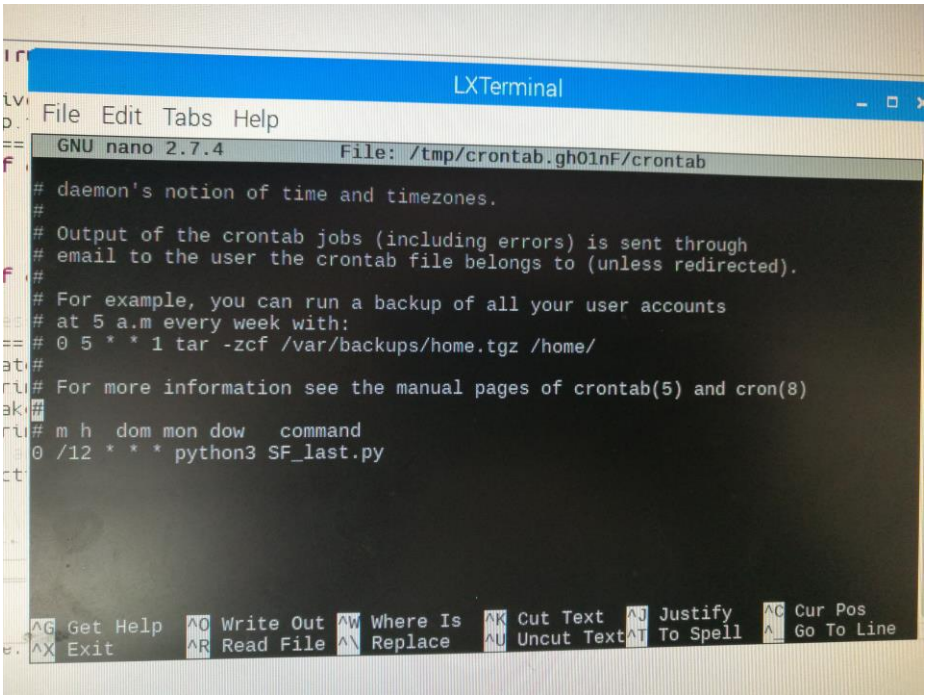
2013.5.24

附录 1　原始资料






LXTerminal
root
Thonny - /root/...
17:52
Thonny - /root/SF_last.py @ 81 : 90
File Edit View Run Tools Help
SF_last.py
active = True
count = 0
while active:
i = rp_input(12)
if i == 1:
if count == 0:
watering_on()
print('The watering proccess is running at the moment!')
count +=1
continue
if count != 0:
continue
#return the message 'dry' to turn on the watering proccess until return the message 'wet'
if i == 0:
watering_off()
print('Proccess has been completed!\nGetting picture...\nWriting into logfile...')
take_picture_write_logfile()
print('Done!')
#take a photo and write the log file
active = False
clean()
Shell
>>>


LXTerminal
File Edit Tabs Help
GNU nano 2.7.4 File: /tmp/crontab.ghO1nF/crontab
daemon's notion of time and timezones.
#
Output of the crontab jobs (including errors) is sent through
email to the user the crontab file belongs to (unless redirected).
#
For example, you can run a backup of all your user accounts
at 5 a.m every week with:
0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
For more information see the manual pages of crontab(5) and cron(8)
#
m h dom mon dow command
0 /12 * * * python3 SF_last.py
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line


Hisense