NIST Laboratory Environment Monitoring and Alert System (LEMAS)

Documentation for building, usage, and configuration

Michael Braine, michael.braine@nist.gov

August 5, 2017

Revision 1.0 established for LEMASdist v1.00 package

What is this, and why should I read it?

This documentation outlines device setup and customization options should a user desire to perform changes to the controls that dictate when a MMS message or email is sent to lab users. It assumes no understanding of Python3 syntax, Python libraries that are used, or Linux commands; however, the reader should be, at a minimum, comfortable on a command line interface and know what a raspberry (raz͵berē,'raz͵b(ə)rē – an edible soft fruit related to the blackberry, consisting of a cluster of reddish-pink drupelets) looks like. The curator, in the context within, is the current maintainer of all the monitoring devices (primary curator) and could include additional members whom are designated to receive the same messages as the primary curator.

Components required for a new device

<components list>

<picture of setup>

Using a SDXC microSD card (typically larger than 16 GB capacity, but includes some 16 GB cards) will require exFAT32 formatting.

The current revision of LEMASrun.py was tested to work with the above components, and this document written with the assumption that the device will be using the above parts list. A Raspberry Pi 3 Model B (hereby RPi for the rest of the text) is not required, but knowledge of Python3 (or Python2, if you prefer that flavor) and the native OS will be required for setup, especially if running the program on boot is desired.

Files required for manual installation, provided by the curator in folder LEMASdist

Filename	*Description*	*Raspbian Install Location*
networkconfig.sh	Bash script to configure network on boot	/home/pi/LEMASdist
LEMASRun.py	Python3 script for logging T3311, alerts	/home/pi/LEMASdist
LEMASRunQuiet.py	Python3 script for logging T3311, no alerts	/home/pi/LEMASdist
LEMASRunLoud.py	Python3 script for logging T3311, alerts	/home/pi/LEMASdist
NoContact.list	list of names to not send messages to	/home/pi/LEMASdist
LabSettings.py	user-configurable settings file	/home/pi/LEMASdist

launch-LEMAS	executable Bash script to begin log	/home/pi/Desktop
Contacts.py(.gpg)	GNU_gpg encrypted, contacts list	/home/pi/LEMASdist
testmsgdate.csv	date to send test message to users	/home/pi/LEMASdist
config.txt	configuration for Adafruit display	/boot
lightdm.conf	configuration for disabling screensaver	/etc/lightdm

Logic for when a message is sent

A message is sent from any single device under one of three conditions:

1) When the set environment has exceeded limits, with a graph
2) When the set environment has returned within limits for a period of time, with a graph
3) When the set environment has exceeded limits, and the environment (T or RH) changes in increments of TincSet or RHincSet (LabSettings.py) until the environment returns within limits for a period of time, with a graph

In the first case, the device will send a message with an attached graph to its laboratory owners should temperature or humidity limits be exceeded. These limits can be easily changed, detailed in section *Configuring the device for a lab with existing settings*.

In the second case, the device will send a message with an attached graph to its laboratory owners should temperature or humidity conditions return within limits after a period of time. These limits are the same limits that decide when a message is sent under the third condition and can be easily changed, detailed in section *Configuring the device for a lab with existing settings*.

In the third case, when the environment is outside of the setpoints, the device will send a message for each occasion the environment changes by TincSet or RhincSet. E.g., the temperature upper limit is 25 deg. C, and TincSet is 5 deg. C, there will be a message generated at 30 deg. C, 35, deg. C, etc, until the temperature returns below 25 deg. C.

Device setup from a blank microSD card

Required to install Raspbian to a microSD card is a microSD card reader, or adapter for SD-to-microSD.

Installing Raspbian OS from a Windows Machine

1. Navigate a web browser to https://www.raspberrypi.org/downloads/raspbian/ and download the latest full Raspbian ZIP image for Raspberry Pi 3.
2. Navigate a web browser to https://etcher.io, download and install the latest Etcher release.
3. Insert the microSD card into the computer. Launch etcher. Select the downloaded Raspbian image file, select the drive containing the microSD card, select Flash!.
4. Once Raspbian has been written to the microSD card and verified by Etcher, remove the microSD card from the computer and insert it into the Raspberry Pi. Connect the Raspberry Pi to a computer monitor (*NOT THE ADAFRUIT DISPLAY*, the RPi must be configured because the display does not have onboard video codec feedback. Running the installer BASH will configure the Adafruit display). Supply power to the Raspberry Pi to through the micro USB cable. The RPi will boot directly into the default user “pi” with default password “raspberry”.

Automated installation of LEMASrun.py with BASH script: LESAinstaller.sh

1. Activate the US-based keyboard layout by navigating through the start menu (raspberry icon > Preferences > Mouse and Keyboard). Click the Keyboard tab, click on the Keyboard Layout button, select United States under Country, and English (US) under Variant.
2. Connect the device to the internet with ethernet and use the network icon on the taskbar to connect to the NIST Research Network. Ask the OU IT Security Officer or group IT Security Representative for assistance if needed. Do note that you will need access to repositories external to NIST at least once when setting up the REN connection. Alternatively, the NIST Guest WiFi is suitable for updates since the Raspberry Pi 3 Model B has a built-in WiFi chip. Similarly, use the network icon to connect to NIST Guest WiFi.
3. On a flash drive, load the LEMASdist folder provided by the curator. Plug the flash drive into the RPi on any USB slot.
4. Open a terminal with CTRL+ALT+T and change directory (cd) to the directory on the flash drive containing LEMASdist. Typically, the flash drive is location in /media/pi/<flash drive name>/LEMASdist/, so the command will be:
   ```
   cd /media/pi/<flash drive name>/LEMASdist/
   ```
   while replacing <flash drive name> with name of the flash drive.
5. Run the install.sh script with the command:
   ```
   sh install.sh
   ```
   This command executes the BASH script that copies files/creates directories needed for the LEMASrun.py program, performs Raspbian OS updates, and downloads Python3 libraries, and changes permissions for related files. Respond with y to any y/n prompts, confirming the extra disk space usage. Enter the passphrase to decrypt the Contacts.py.gpg file. Seek the curator if the pass phrase is required.
6. Add the networkconfig.sh BASH script to crontab, a daemon scheduler that runs on boot.
   In a terminal, launch crontab scheduler:
   ```
   sudo crontab -e
   ```
   In crontab, scroll to the last line and add a new line with the text:
   ```
   @reboot sh /home/pi/LEMASdist/networkconfig.sh >/home/pi/logs/cronlog 2>&1
   ```
   This is the scheduler script that runs the BASH script on boot to launch the environment monitoring program.

The Raspberry Pi will begin logging and sending messages once on the REN upon restart; however, the Python logging script will need configured for the lab it will be monitoring in order for the device to identify itself correctly and send messages under the desired outage conditions. See sections *Configuring the device for a lab with existing settings*, *Adding new labs and new users*, and *Adding new lab settings*, as appropriate.

Manual installation of NIST Laboratory Environment SMS Alert System

Updating Raspbian OS and Python3

1. Activate the US-based keyboard layout by navigating through the start menu (raspberry icon > Preferences > Mouse and Keyboard). Click the Keyboard tab, click on the Keyboard Layout button, select United States under Country, and English (US) under Variant.
2. Connect the device to the internet with Ethernet and use the network icon on the taskbar to connect to the NIST Research Network. Ask the OU IT Security Officer or group IT Security Representative for assistance if needed. Do note that you will need access to

repositories external to NIST at least once when setting up the REN connection. Alternatively, the NIST Guest WiFi is suitable for updates, since the Raspberry Pi 3 Model B has a built-in WiFi chip. Similarly, use the network icon to connect to NIST Guest WiFi.

3. In Raspbian, open a terminal either by clicking the button on the taskbar or navigating through the start menu (raspberry icon > Accessories > Terminal). Type the following command in the terminal and press enter:
   sudo apt-get update
   This command will ask for a list of updates of all installed packages that are managed by the apt-get package manager, including Raspbian OS and Python 3. Once the list has been updated, type the following command in the terminal and press enter:
   sudo apt-get upgrade
   This command will download and install the updates. Reply y to any y/n prompts.
4. (optional, for freeing extra disk space)In the terminal, enter:
   `sudo apt-get remove minecraft-pi -y && sudo apt-get remove bluej -y && sudo apt-get remove geany -y && sudo apt-get remove greenfoot -y && sudo apt-get remove nodered -y && sudo apt-get remove scratch -y && sudo apt-get remove scratch2 -y && sudo apt-get remove sense-hat -y && sudo apt-get remove sonic-pi -y`
   These commands will remove unused programs preloaded on Raspbian.
5. In the terminal, enter:
   `sudo apt-get autoclean`
   This command will clean up any files that may no longer be used due to the update processes.
6. In the terminal, enter:
   sudo apt-get install python3-pyserial python3-numpy python3-matplotlib
   This command installs the PySerial, NumPy, and matplotlib libraries for Python3 for serial interfacing, numerical math operations similar to Matlab, and matrix plotting library, respectively. Note that the python3 designation here is not a Linux command to run Python, it is asking for the libraries in the repositories to download that are built for Python3.
7. In the terminal, enter:
   sudo python3 /usr/bin/pip install minimalmodbus
   This will install the minimalmodbus library for Modbus RTU communications with Python's pip installer for Python 3.

Configuring the Raspberry Pi 3

8. On a flash drive, load the LEMASdist folder provided by the curator. Plug the flash drive into the RPi on any USB slot. Open file manager with the folder icon on the taskbar, or navigating through the start menu (raspberry icon > Accessories > File Manager).
9. If the dropdown menu does not already read "Places", click the dropdown that reads "Directory Tree" and change to "Places". Access the flash drive and copy/paste the LEMASdist folder onto the desktop.
10. In the terminal, use the copy command (cp) to move the config.txt file to /boot/:
    sudo cp home/pi/Desktop/LEMASdist/config.txt /boot/config.txt
    The config.txt file will allow the Adafruit display to work with the RPi upon restarting with the display connected and powered. If tilde (~) does not appear, rather another symbol, then the keyboard needs changed to English (US) (see Step 9).
11. In the terminal, enter the following command to change audio output from over HDMI to the 3.5mm jack. This will prevent potential issues when using the Adafruit display:
    sudo amixer cset numid=3 1
12. In the terminal, enter the following command:
    sudo cp home/pi/Desktop/LEMASdist/lightdm.conf etc/lightdm/lightdm.conf
    This copies a file which will deactivate the blank screen screensaver so that graphed results can always be viewed without mouse/keyboard action.

13. Add the networkconfig.sh BASH script to crontab, a daemon scheduler that runs on boot.
    In a terminal, launch crontab scheduler:
    sudo crontab -e
    In crontab, scroll to the last line and add a new line with the text:
    @reboot sh /home/pi/LEMASdist/networkconfig.sh
    This is the scheduler script that runs the BASH script on boot to configure the research network connection.
14. Add a lxterminal session to launch at user login by opening the autostart file. In a terminal:
    sudo leafpad /home/pi/.config/lxsession/LXDE-pi/autostart
    This will open the autostart script file. On the last line, add a new line that reads:
    @lxterminal -e sudo /usr/bin/python3.4 /home/pi/LEMASdist/LEMASrun.py
    This will instruct the system to open a terminal and execute the logging program upon user login. Press Ctrl+s to save and Ctrl+q to quit leafpad.

The Raspberry Pi will begin logging and sending messages once on the REN upon restart; however, the Python logging script will need configured for the lab it will be monitoring in order for the device to identify itself correctly and send messages under the desired outage conditions. See sections *Configuring the device for a lab with existing settings*, *Adding new labs and new users*, and *Adding new lab settings*, as appropriate.

Setting file ownership and permissions

Open a terminal (CTRL+ALT+T) and use the following commands to change file ownership (chown) and file permissions (chmod):

```
sudo chmod u=rwx,g=rx,o=rx /home/pi/LEMASdist/networkconfig.sh

sudo chown pi:pi /home/pi/LEMASdist/testmsgdate.py

sudo chown pi:pi /home/pi/LEMASdist/LabSettings.py

sudo chown pi:pi /home/pi/LEMASdist/NoContact.list

sudo chown pi:pi /home/pi/Desktop/launch-LEMAS

sudo chmod u=rwx,g=rx,o=rx /home/pi/Desktop/launch-LEMAS

sudo chown pi:pi /home/pi/Desktop/NIST-LEMAS.pdf

sudo chown pi:pi /home/pi/Desktop/T3311manual.pdf

sudo chown pi:pi /home/pi/Desktop/T3311specs.pdf

sudo chown pi:pi /home/pi/Desktop/T3311protocols.pdf
```

Decrypting the Contacts File

Open a terminal (CTRL+ALT+T) and use the following command to begin decryption of the Contacts.py.gpg file:

```
gpg /home/pi/LEMASdist/Contacts.py.gpg
```

Enter the passphrase provided by the curator to initialize decryption. Contacts.py will be added to the LEMASdist folder.

Configuring the device for a lab with existing settings

If a configuration does not exist for the lab, then the following steps must be skipped and section *Adding new lab settings* followed first.

1. In the terminal, execute the following command:
   leafpad ~/LEMASdist/LabSettings.py
   This command opens the file containing lab settings in the leafpad text editor.
2. In the leafpad text editor, navigate to the line of text that reads: labID = '219/G032'. This line will need updated to reflect the lab that the device will be monitoring. The format of the labID variable is, inside single quotes (') <building number>/<lab number>. For example, if the device will be used in building 220, room B113, the line must read:
   labID = '220/B113'
   Once the change is made, press Ctrl+S to save the change, then Ctrl+Q to quit leafpad.
3. Properly reboot the RPi by performing Start > Restart, or opening a terminal (CTRL+ALT+T) and using the command `sudo reboot`.

The Raspberry Pi will now begin logging upon boot.

Adding users to an existing lab

If a configuration exists for a lab, one must only edit Contacts.py. Note that Contacts.py.gpg must first have been decrypted into Contacts.py.

1. Open Contacts.py in a text editor:
   `leafpad ~/LEMASdist/Contacts.py`
2. Find the `labusers` dictionary that contains the desired lab. Add a comma after the last user, before the closing square bracket ( ]) and add the name of the user in single quotes. Their name spelling and capital letter usage must match their name in the `allcontacts` dictionary.

Adding new labs and assigning contacts

If the contacts already exist, then section the *Adding New Contacts* can be ignored. Otherwise, the contacts need added and *Adding new contacts* must be followed as well. Note: Contacts.py.gpg must first have been decrypted into Contacts.py.

1. If Contacts.py is not already opened in leafpad, in a terminal, execute the following command:
   leafpad ~/LEMASdist/Contacts.py
   This command opens the lab monitoring program in the leafpad tex
2. To add a new lab, navigate to the line that reads:
   #labusers['<building>/<room>'] = ['user1', 'user2']
   This is the line that needs edited. Delete the symbol for comment line (#) and replace the text inside the brackets (<>), including the brackets, with the necessary information. The names entered for users must match those in the `allcontacts` dictionary.

Adding new contacts

If a lab does not exist for the contacts, follow section *Adding new labs* first. Note: Contacts.py.gpg must first have been decrypted into Contacts.py.

1. Open Contacts.py in leafpad:
   `leafpad ~/LEMASdist/Contacts.py`
2. To add a new contact, navigate to the line that reads:
   `#allcontacts['user1'] = ['<number>@<carrier>', '<email>@<domain>']`
   This is the line that needs edited. Delete the symbol for comment line (#) and replace the

text inside the bracks (<>), including the brackets, with the necessary information. The spelling and capital letter usage for the user must match that in the `labusers` dictionary.

Adding new lab settings

This section is intended for when a device is added to a lab for which there are no settings. The settings are contained in LabSettings.py, set as the Tcontrols and RHcontrols. Adding users to a lab, or adding a new lab, is detailed in section *Adding new labs and new users*. The new lab should be added first.

1. In the terminal, execute the following command:
   leafpad ~/LEMASdist/LabSettings.py
   This command opens the lab monitoring program in the leafpad text editor.
2. On the last line of Tcontrols, enter a new line below. Enter into the line the format for Tcontrols:
   ['<building>/<room>'] = [<minimum temperature>, <maximum temperature>]
3. Remove any less, than, greater than symbols. For example, for a lab in building 219, G032 with minimum temperature of 19 deg. C and maximum temperature of 21 deg. C, the line will read as follows:
   Tcontrols['219/G032'] = [19, 21]
4. On the last line of RHcontrols, enter a new line below. Enter into the line the format for RHcontrols
   RHcontrols['<building>/<room>'] = [<minimum humidity>, <maximum humidity>]
5. Remove any less, than, greater than symbols. For example, for a lab in building 219, G032 with minimum humidity of 30 %RH and maximum humidity of 60 %RH, the line will read as follows:
   RHcontrols['219/G032'] = [30, 60]

Setting a date and time for a test message

This section is intended for assigning a date and time for sending a test message to all users. This assumes the intended device is set up with users and outage conditions.

1. In Raspbian, on the device to send a test message from, open a terminal. Use the following command to open the configuration file:
   sudo leafpad /home/pi/LEMASdist/testmsgdate.py
2. Edit the TestmsgDate variable with the date and time, in 24-hour format, a test message is desired to be sent. Surround the date and time with single quotes (‘). Use the following examples to input the date and time in the correct format:
   TestmsgDate = ‘February, 01 2118 18:30:00’
   or
   TestmsgDate = ‘August, 22 1986 06:30:00’
   Both of these settings will send a test message to all users of the device at 6:30 PM or 6:30 AM, respectively, on their respective month, day, and year. The latter will not send any messages at all because the date has already passed…hopefully. If it hasn’t, then a time paradox must have occurred because the author of this document and logging program has not yet been born. It is very important to note that setting the date and time as February 1, 2118 18:30:00 may not have the intended results. Notice the absent 0 in the day of the month. *Do not* enter any other lines, there must only be the first line containing the date and time.
   This format *must* be followed in order for the date and time to be parsed correctly. For reference, the strftime format is:

"%B %d, %Y %H:%M:%S"

The website: http://strftime.org/ details the meaning of each symbol.

Appendices

Appendix A: Linux Boot Script for LEMAS using crontab: networkconfig.sh

networkconfig.sh must be copied to /home/pi/LEMASdist/networkconfig.sh and scheduled with crontab.

networkconfig.sh file contents:

```
#!/bin/sh
#networkconfig.sh
#   Tested with Raspbian GNU/Linux 8 (Jessie) on Raspberry Pi 3 Model B
#
#////////////////////////////////////////////////////////////////////////////
## envlauncher.sh Notes
#   Authored by: Michael Braine, Physical Science Technician, NIST, Gaithersburg, MD
#       PHONE: 301 975 8746
#       EMAIL: michael.braine@nist.gov (use this instead of phone)
#
#   Purpose: BASH script to configure Raspberry Pi network settings on boot. Needs scheduled with crontab daemon.
#           @reboot sh /home/pi/LEMASdist/networkconfig.sh
#
#////////////////////////////////////////////////////////////////////////////
## References
#   -none
#
## Change log from v1.00 to v1.00
#   July 26, 2017
#
#   ver 1.00 - initial version

#configure network
sudo ifconfig eth0 129.6.171.181
sudo ifconfig eth0 netmask 255.255.255.224
sudo route add default gw 129.6.171.190 eth0

alias python=python3.4
```

Appendix B: Common carrier gateways

See https://martinfitzpatrick.name/list-of-email-to-sms-gateways/ for a more complete list.

```
# Carrier                          Email to SMS Gateway
# Alltel                    [10-digit phone number]@message.alltel.com
#                               Example: 1234567890@message.alltel.com
# AT&T (formerly Cingular)       [10-digit phone number]@txt.att.net
#                          [10-digit phone number]@mms.att.net (MMS)
#                          [10-digit phone number]@cingularme.com
#                               Example: 1234567890@txt.att.net
# Boost Mobile             [10-digit phone number]@myboostmobile.com
#                               Example: 1234567890@myboostmobile.com
# Nextel (now Sprint Nextel)     [10-digit telephone number]@messaging.nextel.com
#                               Example: 1234567890@messaging.nextel.com
# Sprint PCS (now Sprint Nextel)  [10-digit phone number]@messaging.sprintpcs.com
#                          [10-digit phone number]@pm.sprint.com (MMS)
#                               Example: 1234567890@messaging.sprintpcs.com
# T-Mobile                      [10-digit phone number]@tmomail.net
#                               Example: 1234567890@tmomail.net
# US Cellular                   [10-digit phone number]@email.uscc.net (SMS)
#                          [10-digit phone number]@mms.uscc.net (MMS)
#                               Example: 1234567890@email.uscc.net
# Verizon                       [10-digit phone number]@vtext.com
#                          [10-digit phone number]@vzwpix.com (MMS)
#                               Example: 1234567890@vtext.com
# Virgin Mobile USA                [10-digit phone number]@vmobl.com
#                               Example: 1234567890@vmobl.com
```

Appendix C: Raspberry Pi configuration to use Adafruit 5” 800x480 HDMI Backpack display: config.txt

config.txt must be copied to /boot/config.txt and overwrite the existing file.

config.txt file contents:

```
# uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1

# uncomment this if your display has a black border of unused pixels visible
```

```
# and your display can output without overscan
#disable_overscan=1

# uncomment the following to adjust overscan. Use positive numbers if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

# uncomment to force a console size. By default it will be display's size minus
# overscan.
#framebuffer_width=1280
#framebuffer_height=720

# uncomment if hdmi display is not detected and composite is being output
hdmi_force_hotplug=1

# uncomment to force a specific HDMI mode (here we are forcing 800x480!)
hdmi_group=2
hdmi_mode=1
hdmi_mode=87
hdmi_cvt=800 480 60 6 0 0 0

max_usb_current=1

# uncomment to force a HDMI mode rather than DVI. This can make audio work in
# DMT (computer monitor) modes
#hdmi_drive=2

# uncomment to increase signal to HDMI, if you have interference, blanking, or
# no display
#config_hdmi_boost=4

# uncomment for composite PAL
#sdtv_mode=2
```

```
#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800

# for more options see http://elinux.org/RPi_config.txt
```

Appendix D: Automated installation BASH script: LESAinstaller.sh

```
#!/bin/sh
#install.sh
#   Tested with Raspbian GNU/Linux 8 (Jessie) on Raspberry Pi 3 Model B
#
#////////////////////////////////////////////////////////////////////////////
## install.sh Notes
#   Authored by: Michael Braine, Physical Science Technician, NIST, Gaithersburg, MD
#       PHONE: 301 975 8746
#       EMAIL: michael.braine@nist.gov (use this instead of phone)
#
#   Purpose: BASH script to copy and create files/folders for LEMASrun.py
#
#////////////////////////////////////////////////////////////////////////////
## References
#
#////////////////////////////////////////////////////////////////////////////
## Change log from v1.02 to v1.03
#   November 26, 2017
#
#   ver 1.03 - added decryption for Contacts.py.gpg
#
##////////////////////////////////////////////////////////////////////////////


#ask for labID
read -p "What will be the labID? e.g. 219/G032: " labID
labIDf=$(echo $labID | sed -n -e 's/\//_/p')                    #replaced / with _ to prevent confusing filenames with
directory path


#update OS
echo " "
echo "Updating Raspberry Pi"
```

```
sudo apt-get update -y
sudo apt-get upgrade -y

echo " "
echo "Removing nonessential Raspbian programs"
sudo apt-get remove minecraft-pi -y
sudo apt-get remove bluej -y
sudo apt-get remove geany -y
sudo apt-get remove greenfoot -y
sudo apt-get remove nodered -y
sudo apt-get remove scratch -y
sudo apt-get remove scratch2 -y
sudo apt-get remove sense-hat -y
sudo apt-get remove sonic-pi -y
sudo apt-get autoremove -y
echo "Raspberry Pi update done"

#install python3 libraries
echo " "
echo "Updating/installing Python3 libraries"
sudo apt-get install python3-serial                    #install or update pyserial
sudo apt-get install python3-numpy                     #install or update numpy
sudo apt-get install python3-matplotlib                #install or update matplotlib
sudo python3 /usr/bin/pip install minimalmodbus              #install or update minimalmodbus

#install LEMAS files
echo " "
echo "Copying LEMAS files"
CWD=$(pwd)
sudo mkdir /home/pi/LEMASdist                          #create LEMAS directories
sudo mkdir /home/pi/LEMASdist/tmpimg
sudo cp $CWD/LEMASRun.py /home/pi/LEMASdist/                   #copy program files to LEMAS directory
sudo cp $CWD/testmsgdate.py /home/pi/LEMASdist/
sudo cp $CWD/LabSettings.py /home/pi/LEMASdist/
sudo cp $CWD/Contacts.py.gpg /home/pi/LEMASdist/
sudo cp $CWD/networkconfigs/$labIDf'_'networkconfig.sh /home/pi/LEMASdist/networkconfig.sh
sudo cp $CWD/NoContact.list /home/pi/LEMASdist/
```

```
sudo cp $CWD/launch-LEMAS /home/pi/Desktop/

sudo cp $CWD/LEMASpi/config.txt /boot/                    #move system files to operate with adafruit display
and disable screensaver

sudo cp $CWD/LEMASpi/lightdm.conf /etc/lightdm

sudo amixer cset numid=3 1                                #disable audio output through HDMI


#copy manuals for Comet System T3311

echo " "

echo "Copying instrument manuals"

sudo cp $CWD/LEMASmanuals/NIST-LEMAS.pdf /home/pi/Desktop/

sudo cp $CWD/LEMASmanuals/T3311manual.pdf /home/pi/Desktop/

sudo cp $CWD/LEMASmanuals/T3311specs.pdf /home/pi/Desktop/

sudo cp $CWD/LEMASmanuals/T3311protocols.pdf /home/pi/Desktop/


#copy pdf for LEMAS

sudo cp $CWD/NIST-LEMAS.pdf /home/pi/Desktop/


#change permissions and ownerships

echo " "

echo "Setting up LEMAS permissions"

sudo chmod u=rwx,g=rx,o=rx /home/pi/LEMASdist/networkconfig.sh        #change permissions of bash script to
be executable

sudo chown pi:pi /home/pi/LEMASdist/testmsgdate.py

sudo chown pi:pi /home/pi/LEMASdist/LabSettings.py

sudo chown pi:pi /home/pi/LEMASdist/NoContact.list

sudo chown pi:pi /home/pi/Desktop/launch-LEMAS

sudo chmod u=rwx,g=rx,o=rx /home/pi/Desktop/launch-LEMAS

sudo chown pi:pi /home/pi/Desktop/NIST-LEMAS.pdf

sudo chown pi:pi /home/pi/Desktop/T3311manual.pdf

sudo chown pi:pi /home/pi/Desktop/T3311specs.pdf

sudo chown pi:pi /home/pi/Desktop/T3311protocols.pdf


echo " "

echo "Passphrase required to decrypt contact information"

gpg /home/pi/LEMASdist/Contacts.py.gpg

rm /home/pi/LEMASdist/Contacts.py.gpg
```

```
echo " "
echo "LEMAS installation complete"
echo "Remember: LXTerminal and Chrontab need manually set up (see NIST-LEMAS.pdf in /home/pi/Desktop)"
echo "Have fun!"
```

Appendix D: Configuration settings for disabling screensaver: lighdm.conf

lightdm.conf must be copied to /etc/lightdm/lightdm.conf and overwite the existing file.

lightdm.conf file contents:

```
#
# General configuration
#
# start-default-seat = True to always start one seat if none are defined in the configuration
# greeter-user = User to run greeter as
# minimum-display-number = Minimum display number to use for X servers
# minimum-vt = First VT to run displays on
# lock-memory = True to prevent memory from being paged to disk
# user-authority-in-system-dir = True if session authority should be in the system location
# guest-account-script = Script to be run to setup guest account
# logind-load-seats = True to automatically set up multi-seat configuration from logind
# logind-check-graphical = True to on start seats that are marked as graphical by logind
# log-directory = Directory to log information to
# run-directory = Directory to put running state in
# cache-directory = Directory to cache to
# sessions-directory = Directory to find sessions
# remote-sessions-directory = Directory to find remote sessions
# greeters-directory = Directory to find greeters
#
[LightDM]
#start-default-seat=true
#greeter-user=lightdm
#minimum-display-number=0
#minimum-vt=7
#lock-memory=true
#user-authority-in-system-dir=false
#guest-account-script=guest-account
#logind-load-seats=false
#logind-check-graphical=false
```

#log-directory=/var/log/lightdm

#run-directory=/var/run/lightdm

#cache-directory=/var/cache/lightdm

#sessions-directory=/usr/share/lightdm/sessions:/usr/share/xsessions

#remote-sessions-directory=/usr/share/lightdm/remote-sessions

#greeters-directory=/usr/share/lightdm/greeters:/usr/share/xgreeters

#

Seat defaults

#

type = Seat type (xlocal, xremote)

xdg-seat = Seat name to set pam_systemd XDG_SEAT variable and name to pass to X server

pam-service = PAM service to use for login

pam-autologin-service = PAM service to use for autologin

pam-greeter-service = PAM service to use for greeters

xserver-command = X server command to run (can also contain arguments e.g. X -special-option)

xserver-layout = Layout to pass to X server

xserver-config = Config file to pass to X server

xserver-allow-tcp = True if TCP/IP connections are allowed to this X server

xserver-share = True if the X server is shared for both greeter and session

xserver-hostname = Hostname of X server (only for type=xremote)

xserver-display-number = Display number of X server (only for type=xremote)

xdmcp-manager = XDMCP manager to connect to (implies xserver-allow-tcp=true)

xdmcp-port = XDMCP UDP/IP port to communicate on

xdmcp-key = Authentication key to use for XDM-AUTHENTICATION-1 (stored in keys.conf)

unity-compositor-command = Unity compositor command to run (can also contain arguments e.g. unity-system-compositor -special-option)

unity-compositor-timeout = Number of seconds to wait for compositor to start

greeter-session = Session to load for greeter

greeter-hide-users = True to hide the user list

greeter-allow-guest = True if the greeter should show a guest login option

greeter-show-manual-login = True if the greeter should offer a manual login option

greeter-show-remote-login = True if the greeter should offer a remote login option

user-session = Session to load for users

allow-user-switching = True if allowed to switch users

allow-guest = True if guest login is allowed

guest-session = Session to load for guests (overrides user-session)

session-wrapper = Wrapper script to run session with

```
# greeter-wrapper = Wrapper script to run greeter with
# guest-wrapper = Wrapper script to run guest sessions with
# display-setup-script = Script to run when starting a greeter session (runs as root)
# display-stopped-script = Script to run after stopping the display server (runs as root)
# greeter-setup-script = Script to run when starting a greeter (runs as root)
# session-setup-script = Script to run when starting a user session (runs as root)
# session-cleanup-script = Script to run when quitting a user session (runs as root)
# autologin-guest = True to log in as guest by default
# autologin-user = User to log in with by default (overrides autologin-guest)
# autologin-user-timeout = Number of seconds to wait before loading default user
# autologin-session = Session to load for automatic login (overrides user-session)
# autologin-in-background = True if autologin session should not be immediately activated
# exit-on-failure = True if the daemon should exit if this seat fails
#
[SeatDefaults]
xserver-command=X -s 0 -dpms
#type=xlocal
#xdg-seat=seat0
#pam-service=lightdm
#pam-autologin-service=lightdm-autologin
#pam-greeter-service=lightdm-greeter
#xserver-command=X
#xserver-layout=
#xserver-config=
#xserver-allow-tcp=false
#xserver-share=true
#xserver-hostname=
#xserver-display-number=
#xdmcp-manager=
#xdmcp-port=177
#xdmcp-key=
#unity-compositor-command=unity-system-compositor
#unity-compositor-timeout=60
#greeter-session=example-gtk-gnome
greeter-hide-users=false
#greeter-allow-guest=true
#greeter-show-manual-login=false
```

```
#greeter-show-remote-login=true
#user-session=default
#allow-user-switching=true
#allow-guest=true
#guest-session=
#session-wrapper=lightdm-session
#greeter-wrapper=
#guest-wrapper=
#display-setup-script=
#display-stopped-script=
#greeter-setup-script=
#session-setup-script=
#session-cleanup-script=
#autologin-guest=false
autologin-user=pi
#autologin-user-timeout=0
#autologin-in-background=false
#autologin-session=UNIMPLEMENTED
#exit-on-failure=false

#
# Seat configuration
#
# Each seat must start with "Seat:".
# Uses settings from [SeatDefaults], any of these can be overriden by setting them in this section.
#
#[Seat:0]

#
# XDMCP Server configuration
#
# enabled = True if XDMCP connections should be allowed
# port = UDP/IP port to listen for connections on
# key = Authentication key to use for XDM-AUTHENTICATION-1 or blank to not use authentication (stored in keys.conf)
#
# The authentication key is a 56 bit DES key specified in hex as 0xnnnnnnnnnnnnnn.  Alternatively
# it can be a word and the first 7 characters are used as the key.
```

```
#
[XDMCPServer]
#enabled=false
#port=177
#key=

#
# VNC Server configuration
#
# enabled = True if VNC connections should be allowed
# command = Command to run Xvnc server with
# port = TCP/IP port to listen for connections on
# width = Width of display to use
# height = Height of display to use
# depth = Color depth of display to use
#
[VNCServer]
#enabled=false
#command=Xvnc
#port=5900
#width=1024
#height=768
#depth=8
```

Appendix E: Settings file for Laboratories: LabSettings.py

```
## ////////////////////////ONLY CHANGE THIS BETWEEN LABS\\\\\\\\\\\\\\\\\\\\\\\\\\

#assigned lab for this device, format is <building number>/<lab number>, set as a string
labID = '219/G032'
#set communications port. On Raspbian for USB-to-serial it should be /dev/ttyUSB0 (check dmesg immediately after connecting USB), default = /dev/ttyUSB0
T3311port = '/dev/ttyUSB0'

## ////////////////////////////USER PREFERENCES\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

#display pixels per inch for 5", 800x480 display, default is 186 px/in^2
dpi_set = 186
#number of minutes before sending message for return to normal status (for ensuring environment is not oscillating around limits), default is 10 minutes
normalstatus_wait = 10
```

#temperature incremental alert, for when temperature is constantly changing. deg. C. system sends message every increment of this setting out of spec.

#e.g., upper spec is 25, setting is 5, outage message sent at 25, incremental message sent at 30, 35, 40, 35, 30, 35, 30, etc.

TincSet = 5

#humidity incremental alert, for when humdity is constantly changing. %RH. system sends message every increment of this setting out of spec.

RHincSet = 10

Graphing settings

#number of points to record per hour, default is 40 pt/hr (every 90 seconds)

pts_per_hr = 40

#length of time to graph into past, hours, default is 12 hours

graph_time = 12

#number of points between each x-axis tickmark, default is 20 points per tick

tick_spacing = 20

#amount of space to graph above maximum temperature in graph time range, deg. C

graphTmax = 0.5

#amount of space to graph below minimum temperature in graph time range, deg. C

graphTmin = 0.5

#amount of space to graph above maximum humidity in graph time range, %RH. 100 sets maximum displays humidity to 100 %RH with no autoscaling

graphRHmax = 1

#amount of space to graph below minimum humidity in graph time range, %RH. 100 sets minimum displays humidity to 0 %RH with no autoscaling

graphRHmin = 1

#fontsize for graph label text, Temperature (deg.C) amd Humidity (RH) text

FontsizeLabel = 16

#fontsize for graph y-axis humidity and temperature values

FontsizeYticks = 5

#fontsize for graph shared x-axis time values

FontsizeXticks = 5

#linewidth of plots of temperature and humidity

GraphLinewidth = 2.0

Conditions for reacquiring data from sensor to prevent graphing bad reads

#if change in temperature from previous is greater than this, reread temperature, deg. C

rereadT = 0.5

#if change in humidity from previous is greater than this, reread humidity, %RH

```
rereadRH = 1


## ////////////////////LABORATORY TEMPERATURE CONTROLS\\\\\\\\\\\\\\\\\\\\\\\\\\\\

# Lab temperature controls

#If these ever need changed, the format for additional labs and temperature controls using a dictionary is:
Tcontrols['<building>/<room>'] = [<minimum temperature>, <maximum temperature>]

#Dictionary list of temperature controls. Units are degrees Celsius

#Add new temperature controls below the last line

Tcontrols = {}                                          #initialize empty dictionary

Tcontrols['219/A027'] = [19.7, 20.2]

Tcontrols['219/F030'] = []

Tcontrols['219/F031'] = []

Tcontrols['219/F032'] = []

Tcontrols['219/F033'] = []

Tcontrols['219/F034'] = [18.0, 22.0]

Tcontrols['219/F036'] = [18.0, 22.0]

Tcontrols['219/G021'] = [19.5, 20.5]

Tcontrols['219/G022'] = []

Tcontrols['219/G023'] = [19.5, 20.5]

Tcontrols['219/G024'] = []

Tcontrols['219/G026'] = []

Tcontrols['219/G027'] = [19.7, 20.2]

Tcontrols['219/G032'] = [18.0, 22.0]


Tcontrols['220/A08'] = [19.5, 20.5]

Tcontrols['220/A12'] = [19.5, 20.5]

Tcontrols['220/A16'] = []

Tcontrols['220/A58'] = []

Tcontrols['220/B05'] = [19.5, 20.5]

Tcontrols['220/B07'] = [19.5, 20.5]

Tcontrols['220/B13'] = []

Tcontrols['220/C03'] = [19.0, 21.0]

Tcontrols['220/C05'] = [17.5, 21.0]


Tcontrols['221/A39'] = []


Tcontrols['237/F035'] = []

#Tcontrols['<building>/<room>'] = [<minimum temperature>, <maximum temperature>]
```

```
## ////////////////////LABORATORY HUMIDITY CONTROLS\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

# Lab humidity controls

#If these ever need changed, the format for additional labs and temperature controls using a dictionary is:
RHcontrols['<building>/<room>'] = [<minimum humidity>, <maximum humidity>]

#Dictionary list of humidity controls. Units are percent relative humidity (0 - 100)

#Add new humidity controls below the last line

RHcontrols = {}                                                    #initialize empty dictionary


RHcontrols['219/A027'] = [20.0, 60.0]

RHcontrols['219/F030'] = []

RHcontrols['219/F031'] = []

RHcontrols['219/F032'] = []

RHcontrols['219/F033'] = []

RHcontrols['219/F034'] = [30.0, 58.0]

RHcontrols['219/F036'] = [30.0, 58.0]

RHcontrols['219/G021'] = [20.0, 60.0]

RHcontrols['219/G022'] = []

RHcontrols['219/G023'] = [20.0, 60.0]

RHcontrols['219/G024'] = []

RHcontrols['219/G026'] = []

RHcontrols['219/G027'] = [20.0, 60.0]

RHcontrols['219/G032'] = [30.0, 58.0]


RHcontrols['220/A08'] = [0, 50]

RHcontrols['220/A12'] = [0, 50]

RHcontrols['220/A16'] = []

RHcontrols['220/A58'] = []

RHcontrols['220/B05'] = [0, 50]

RHcontrols['220/B07'] = [0, 50]

RHcontrols['220/B13'] = []

RHcontrols['220/C03'] = [30.0, 60.0]

RHcontrols['220/C05'] = [30.0, 60.0]


RHcontrols['221/A39'] = []


RHcontrols['237/F035'] = []

#RHcontrols['<building>/<room>'] = [<minimum humidity>, <maximum humidity>]
```

Appendix F: NIST Lab Environment SMS Alert System script: LEMASrun.py

```
#LEMASRun.py

#   Tested with Python 3.6.1 (Anaconda 4.4.0 stack) on Linux Mint 18.2 Sonya Cinnamon, Python 3.4.2, on Raspbian Linux

#

#////////////////////////////////////////////////////////////////////////////

## LEMASRun.py Notes

#   August, 2017

#   Authored by: Michael Braine, Physical Science Technician, NIST, Gaithersburg, MD

#       PHONE: 301 975 8746

#       EMAIL: michael.braine@nist.gov (use this instead of phone)

#

#   Purpose

#       continuously read temperature and humidity from CometSystems T3311, send notification via text/email with graph attached to lab users if temperature or humidity is outside of specified limits

#       log temperature and humidity to <month><YYYY>-all.env.csv

#       log temperature and humidity outages to <month><YYYY>-outage.env.csv

#

#////////////////////////////////////////////////////////////////////////////

## References

# smtp.nist.gov is at ip: 129.6.16.94

#

#   T3311 manual (T3311manual.pdf)

#   T3311 protocols (T3311protocols.pdf)

#

#   Common cell carrier gateways

#   mostly complete list: https://martinfitzpatrick.name/list-of-email-to-sms-gateways/

#

##////////////////////////////////////////////////////////////////////////////

## Change log from v1.07 to v1.08

#   November 2, 2017

#

#   ver 1.08    - added importing of contacts from seperate file, Contacts.py

#

#////////////////////////////////////////////////////////////////////////////


## Used libraries

import smtplib, time, os, csv, datetime, copy
```

```
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
import minimalmodbus
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

## Import user's labsettings
#program_directory = '/home/pi/LEMASdist'
program_directory = '/home/braine/BraineCode/LEMASmaster/LEMASdist'
os.chdir(program_directory)
import LabSettings
import Contacts
import testmsgdate                                  #import test message date with each loop, in case test
message date changes
os.chdir(program_directory+'/tmpimg')

#////////////////////////Import LabSettings\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
## import settings from LabSettings.py
labID = LabSettings.labID
T3311port = LabSettings.T3311port
pts_per_hr = LabSettings.pts_per_hr
graph_time = LabSettings.graph_time
tick_spacing = LabSettings.tick_spacing
dpi_set = LabSettings.dpi_set
normalstatus_wait = LabSettings.normalstatus_wait
Tcontrols = LabSettings.Tcontrols
RHcontrols = LabSettings.RHcontrols
TincSet = LabSettings.TincSet
RHincSet = LabSettings.RHincSet
graphTmax = LabSettings.graphTmax
graphTmin = LabSettings.graphTmin
graphRHmax = LabSettings.graphRHmax
graphRHmin = LabSettings.graphRHmin
FontsizeLabel = LabSettings.FontsizeLabel
FontsizeYticks = LabSettings.FontsizeYticks
```

```
FontsizeXticks = LabSettings.FontsizeXticks

GraphLinewidth = LabSettings.GraphLinewidth

rereadT = LabSettings.rereadT

rereadRH = LabSettings.rereadRH

allcontacts = Contacts.allcontacts

labusers = copy.deepcopy(Contacts.labusers[labID])

TestmsgDate = testmsgdate.TestmsgDate

TestmsgDate = datetime.datetime.strptime(TestmsgDate, "%B %d, %Y %H:%M:%S")


print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Starting NIST Laboratory Environment Monitoring and Alert
System (LEMAS), v1.08, October 2017')

print('\n\nMichael Braine, August 2017\nCurator contact: michael.braine@nist.gov')

print('\n\nLoud Mode')


#//////////////////////////Outage Parameter Setup\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

#set up parameters

graph_pts = round(graph_time*pts_per_hr)                    #maximum number of recent points to plot

sleeptimer = (pts_per_hr / 60 / 60)**-1                     #amount of time for system to wait until next
temperature, seconds

Tmin = Tcontrols[labID][0]                                  #get lower temperature limit for assigned lab

Tmax = Tcontrols[labID][1]                                  #get upper temperature limit for assigned lab

RHmin = RHcontrols[labID][0]                                #get lower humidity limit for assigned lab

RHmax = RHcontrols[labID][1]                                #get upper humidity limit for assigned lab

TincAlert = [Tmin - TincSet, Tmax + TincSet]

RHincAlert = [RHmin - RHincSet, RHmax + RHincSet]


labcontacts = np.array([])

for icontact in range(len(labusers)):

    labcontacts = np.append(labcontacts, allcontacts[labusers[icontact]])


envdata_directory = '/home/pi/Desktop/EnvironmentData'              #where data is stored in Linux file system,
create if doesn't exist, doesn't really ever need changed

TestmsgSent = False                                         #initialize test message has not been sent

ethoutage = False                                           #initialize with internet outage status as false

ethoutage_sent = False                                      #initialize status of messages queued under internet
outage as not sent


## NIST SMTP server credentials and settings
```

```
#email = ''
#passwd = ''
#fromaddr = 'NIST Lab Environment Alert'
#SMTPserver = 'smtp.nist.gov'
#SMTPport = 25

## GMail SMTP server settings
#SMTPserver = 'smtp.gmail.com'
#SMTPport = 587

#//////////////////////////Function Definitions\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
#define message sending functions
def SendMessage(toaddress, message):                                    #method for sending regular messages
    msg = MIMEMultipart()                                               #define msg as having multiple components
    msg['Subject'] = 'DMG Alert: Curator Status Message'
    msg['From'] = 'dmgalert@nist.gov'
    msg['To'] = ','.join(toaddress)
    text = MIMEText(message)
    msg.attach(text)

    server = smtplib.SMTP('129.6.16.94', 25)
    server.sendmail('dmgalert@nist.gov', toaddress, msg.as_string())
    server.quit()

def SendMessageMMS(toaddress, message, img_path):                       #method for sending messages with image
attached
    msg = MIMEMultipart()                                               #define msg as having multiple components
    msg['Subject'] = 'DMG Alert: Environment Outage'
    msg['From'] = 'dmgalert@nist.gov'
    msg['To'] = toaddress
    text = MIMEText(message)
    msg.attach(text)
    img_file = open(img_path, 'rb').read()
    image_attach = MIMEImage(img_file)
    msg.attach(image_attach)

    server = smtplib.SMTP('129.6.16.94', 25)
```

```
    server.sendmail('dmgalert@nist.gov', toaddress, msg.as_string())
    server.quit()


#external to NIST network testing
# def SendMessage(toaddress, message):                                  #method for sending messages
#     server = smtplib.SMTP('smtp.gmail.com', 587)
#     username = input('Login: ')
#     passwd = input('Password: ')
#     server.ehlo()                                                     #not needed for NIST
#     server.starttls()                                                 #not supported by smtp.nist.gov, required for
smtp.gmail.com
#     server.login(username, passwd)                                    #not required by smtp.nist.gov, required for
smtp.gmail.com
#     server.sendmail('Test', toaddress, message)
#     server.quit()


#define sensor connection and reading functions
def ConnectT3311():                                                   #Connect to instrument T3311 method, modbus RTU
protocol
    T3311_obj = minimalmodbus.Instrument(T3311port, 1)
    print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Connecting to T3311...')
    time.sleep(5)


    try:                                                              #Read before setting baudrate solves the checksum error when
reading from the T3311...can't explain why
        T3311_obj.read_register(48,1)
    except Exception:
        pass
    T3311_obj.serial.baudrate = 9600                                  #set baudrate, default = 9600
    time.sleep(5)


    try:
        T3311_obj.read_register(48,1)
    except Exception:                                                 #try another read and reset method if above failed
        pass
        T3311_obj = minimalmodbus.Instrument(T3311port, 1)
        T3311_obj.seria.baudrate = 19200
        time.sleep(5)
```

```
        try:
            T3311_obj.read_register(48,1)
        except Exception:
            T3311_obj.serial.baudrate = 9600
            time.sleep(5)


    return T3311_obj
T3311_obj = ConnectT3311()                                  #connect to instrument


## Definitions for temperature and humidity measurement
def ReadTemperature():                                      #read T3311 address for temperature
    temptemp = (T3311_obj.read_register(48, 1)-32)*5/9          #read temperature, convert from deg. F to
deg. C
    return round(temptemp, 3)


def ReadHumidity():                                         #read T3311 address for humidity
    temphumid = T3311_obj.read_register(49, 1)
    return round(temphumid, 3)


#////////////////////////Variable Initialization\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
## Initialize variables and figure setup
currenttime = np.array([])                                  #initialize empty lists
axestime = np.array([])
temperature = []
humidity = []
tf_alert_T = []
tf_alert_RH = []
plt.ion()                                                   #activate interactive plotting
fig = plt.figure(num=1, figsize=(4.2,2.2), dpi=dpi_set)             #get matplotlib figure ID, set figure size
gs = gridspec.GridSpec(2,3)
gs.update(hspace=0.05)
ax1 = plt.subplot(gs[0,:])
ax2 = plt.subplot(gs[1,:])
fig.subplots_adjust(left=0.06, right=1, top=0.98, bottom=0.14)
fig.canvas.toolbar.pack_forget()
if graphRHmax == 100:
    graphRHmax = 100
```

```
else:
    graphRHmax = max(humidity) + graphRHmax
if graphRHmin == 100:
    graphRHmin = 0
else:
    graphRHmin = min(humidity) - graphRHmin


#//////////////////////Initial Environment Data\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
## Initial temperature, humidity, and time values
#used for inside the loop, checking for bad readings
#check requires at least two values, second acquired in the loop
#initial temperature
try:
    temperature.append(ReadTemperature())                    #read T3311 modbus address for temperature
except Exception:                                            #reestablish connection if failed
    T3311_obj = ConnectT3311()
    try:
        temperature.append(ReadTemperature())                #read T3311 modbus address for temperature
    except Exception:
        T3311_obj = ConnectT3311
        try:
            temperature.append(ReadTemperature())
        except Exception:
            print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Communications with instrument failed')


#initial humidity
try:
    humidity.append(ReadHumidity())                          #read T3311 modbus address for humidity
except Exception:                                            #reestablish connection if failed
    T3311_obj = ConnectT3311()
    try:
        humidity.append(ReadHumidity())                      #read T3311 modbus address for humidity
    except Exception:
        T3311_obj = ConnectT3311
        try:
            humidity.append(ReadHumidity())
        except Exception:
```

```
        print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Communications with instrument failed')


#initial time data
currenttime = np.append(currenttime, time.strftime("%Y-%m-%d %H:%M:%S"))        #get current system time (yyyy mm dd hh mm ss)
axestime = np.append(axestime, time.strftime("%H:%M"))


#////////////////////////////Eternal Loop\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
## Measure temperature and humidity for all eternity
while True:
    ti = time.time()                                                #begin/reset active timer----------------------------------------------------------------------------
    # print('NIST Laboratory Environment Monitoring and Alert System (LEMAS), v1.06, October 2017')
    # print('\n\nMichael Braine, August 2017\nCurator contact: michael.braine@nist.gov\n\nLogging began:'+starttime)


    #//////////////////////////Instrument Communications\\\\\\\\\\\\\\\\\\\\\\\\\\\\
    #read temperature
    try:
        temperature.append(ReadTemperature())                           #read T3311 modbus address for temperature
    except Exception:                                               #reestablish connection if failed
        T3311_obj = ConnectT3311()
        try:
            temperature.append(ReadTemperature())                       #read T3311 modbus address for temperature
        except Exception:
            T3311_obj = ConnectT3311
            try:
                temperature.append(ReadTemperature())
            except Exception:
                print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Communications with instrument failed')


    #if suspected bad sensor read, read temperature again
    if abs(temperature[-1] - temperature[-2]) > rereadT:
        time.sleep(10)
        try:
            temperature[-1] = ReadTemperature()                         #read T3311 modbus address for temperature
        except Exception:
            T3311_obj = ConnectT3311()
            try:
```

```
            temperature[-1] = ReadTemperature()                    #read T3311 modbus address for temperature
        except Exception:
            T3311_obj = ConnectT3311
            try:
                temperature[-1] = ReadTemperature()
            except Exception:
                print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Communications with instrument failed')


time.sleep(1)
#read humidity
try:
    humidity.append(ReadHumidity())                                #read T3311 modbus address for humidity
except Exception:                                                  #reestablish connection if failed
    T3311_obj = ConnectT3311()
    try:
        humidity.append(ReadHumidity())                            #read T3311 modbus address for humidity
    except Exception:
        T3311_obj = ConnectT3311
        try:
            humidity.append(ReadHumidity())
        except Exception:
            print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Communications with instrument failed')

#if suspected bad sensor read, read humidity again
if abs(humidity[-2] - humidity[-2]) > rereadRH:
    time.sleep(10)
    try:
        humidity[-1] = ReadHumidity()                              #read T3311 modbus address for humidity
    except Exception:
        T3311_obj = ConnectT3311()
        try:
            humidity[-1] = ReadHumidity()                          #read T3311 modbus address for humidity
        except Exception:
            T3311_obj = ConnectT3311
            try:
                humidity[-2] = ReadHumidity()
            except Exception:
```

```
            print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Communications with instrument failed')


    #get and format time
    currenttime = np.append(currenttime, time.strftime("%Y-%m-%d %H:%M:%S"))    #get current system time (yyyy mm dd hh mm ss)
    axestime = np.append(axestime, time.strftime("%H:%M"))


    #remove oldest data points from memory that no longer need graphed
    if len(temperature) >= graph_pts:
        del temperature[0]
        del humidity[0]
        currenttime = np.delete(currenttime, 0)
        axestime = np.delete(axestime, 0)


    #////////////////////////////////Update Graphs\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
    time_vec = range(len(axestime))


    #plot temperature
    ax1 = plt.subplot(gs[0,:])
    plt.cla()
    plt.plot(time_vec, temperature, 'r-', linewidth=GraphLinewidth)
    plt.plot(time_vec, np.zeros([len(time_vec), 1])+Tmin, 'b-', linewidth=0.25)
    plt.plot(time_vec, np.zeros([len(time_vec), 1])+Tmax, 'b-', linewidth=0.25)
    ax1.set_ylim([min(temperature)-graphTmin, max(temperature)+graphTmax])      #y-axis limits
    ax1.ticklabel_format(style='plain')                                         #disable scientific notation on y-axis
    #plt.ylabel('Temperature (deg. C)', fontsize=4)
    plt.setp(ax1.get_xticklabels(), visible=False)                              #hide tickmarks, will use shared axis
    plt.grid(color='gray', alpha=0.3)
    plt.text(0.05, 0.1, 'Temperature (deg. C)', transform=ax1.transAxes, alpha=0.5, fontsize=FontsizeLabel, color='gray') #add transparent text to bottom left of first axes
    ax1.patch.set_facecolor('black')
    plt.yticks(fontsize=FontsizeYticks)
    plt.ticklabel_format(useOffset=False)


    #plot humidity with temperature's x-axis
    ax2 = plt.subplot(gs[1,:], sharex=ax1)
    plt.cla()
```

```
    plt.plot(time_vec, humidity, 'g-', linewidth=GraphLinewidth)
    plt.plot(time_vec, np.zeros([len(time_vec), 1])+RHmin, 'b-', linewidth=0.25)
    plt.plot(time_vec, np.zeros([len(time_vec), 1])+RHmax, 'b-', linewidth=0.25)
    ax2.set_ylim([graphRHmin, graphRHmax])
    ax2.ticklabel_format(style='plain')
    #plt.ylabel('Humidity (%RH)', fontsize=4)
    plt.grid(color='gray', alpha=0.3)
    plt.text(0.05, 0.1, 'Humidity (%RH) ', transform=ax2.transAxes, alpha=0.5, fontsize=FontsizeLabel, color='gray')
    ax2.patch.set_facecolor('black')
    plt.yticks(fontsize=FontsizeYticks)
    plt.ticklabel_format(useOffset=False)


    #setup xticks
    plt.xticks(np.arange(min(time_vec), max(time_vec), tick_spacing), axestime[np.arange(min(time_vec),
max(time_vec), tick_spacing)], rotation='vertical', fontsize=FontsizeXticks)
    #plt.xlabel('Time (YYYY-mm-dd hh:mm:ss)')
    #plt.tight_layout()
    plt.pause(0.001)


    #////////////////////////////////Environment Logs\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
    ## Data file management
    #Create EnvironmentData directory if it does not exist
    if not(os.path.isdir(envdata_directory)):
        os.makedirs(envdata_directory)


    ## Read csv and append to end of file
    #files get stored with month and year as filename in .csv format with .env.csv extension in envdata_directory
    monthYYYY = time.strftime("%B%Y")                                   #get month and year for title of file
    if os.path.isfile(envdata_directory+'/'+monthYYYY+'-all.env.csv'):          #use existing monthYYYY.env.csv file
        envfile = open(envdata_directory+'/'+monthYYYY+'-all.env.csv', 'a')     #open file with append properties
        envfile.write(str(currenttime[-1]))                             #add time of measurement
        envfile.write(','+str(temperature[-1]))                           #add latest temperature
        envfile.write(','+str(humidity[-1]))                            #add latest humidity
        envfile.write('\n')
        envfile.close()                                                 #close file
    else:                                                               #otherwise create new monthYYYY.env.csv
        envfile = open(envdata_directory+'/'+monthYYYY+'-all.env.csv', 'w')     #create file with write properties
```

```
        envfile.write('time,Temperature (deg. C),Humidity (%RH)\n')          #write header
        envfile.write(str(currenttime[-1]))                                  #add time of measurement
        envfile.write(','+str(temperature[-1]))                               #add latest temperature
        envfile.write(','+str(humidity[-1]))                                 #add latest humidity
        envfile.write('\n')
        envfile.close()                                                      #close file


        envfile = open(envdata_directory+'/'+monthYYYY+'-outages.env.csv', 'w') #create file with write properties
        envfile.write('time,Temperature (deg. C),Humidity (%RH),Temperature Outage?,Humidity Outage?\n') #write
header
        envfile.write('\n')
        envfile.close()                                                      #close file


    ## Log outages to a -outage.env.csv file

    #Temperature outage log

    if (temperature[-1] > Tmax) or (temperature[-1] < Tmin) and (humidity[-1] < RHmax) and (humidity[-1] > RHmin):
#if temperature outage but humidity in, log the temperature outage
        if len(tf_alert_T) != 0:
            if os.path.isfile(envdata_directory+'/'+monthYYYY+'-outages.env.csv'): #use existing monthYYYY.env.csv file
                envfile = open(envdata_directory+'/'+monthYYYY+'-outages.env.csv', 'a') #open file with append properties
                envfile.write(str(currenttime[-1]))                          #add time of measurement
                envfile.write(','+str(temperature[-1]))                       #add latest temperature
                envfile.write(','+str(humidity[-1]))                         #add latest humidity
                envfile.write(',TEMPERATURE OUTAGE,')
                envfile.write('\n')
                envfile.close()                                              #close file
            else:                                                            #otherwise create new monthYYYY.env.csv
                envfile = open(envdata_directory+'/'+monthYYYY+'-outages.env.csv', 'w') #create file with write properties
                envfile.write('time,Temperature (deg. C),Humidity (%RH),Temperature Outage?,Humidity Outage?\n') #write
header
                envfile.write(str(currenttime[-1]))                          #add time of measurement
                envfile.write(','+str(temperature[-1]))                       #add latest temperature
                envfile.write(','+str(humidity[-1]))                         #add latest humidity
                envfile.write(',TEMPERATURE OUTAGE,')
                envfile.write('\n')
                envfile.close()                                              #close file


    #humidity outage log
```

```
    if (humidity[-1] > RHmax) or (humidity[-1] < RHmin) and (temperature[-1] < Tmax) and (temperature[-1] > Tmin): #if humidity outage but temperature in, log the humidity outage
        if len(tf_alert_RH) != 0:
            if os.path.isfile(envdata_directory+'/'+monthYYYY+'-outages.env.csv'): #use existing monthYYYY.env.csv file
                envfile = open(envdata_directory+'/'+monthYYYY+'-outages.env.csv', 'a') #open file with append properties
                envfile.write(str(currenttime[-1]))                    #add time of measurement
                envfile.write(','+str(temperature[-1]))                  #add latest temperature
                envfile.write(','+str(humidity[-1]))                   #add latest humidity
                envfile.write(', ,HUMIDITY OUTAGE')
                envfile.write('\n')
                envfile.close()                                  #close file
            else:                                               #otherwise create new monthYYYY.env.csv
                envfile = open(envdata_directory+'/'+monthYYYY+'-outages.env.csv', 'w') #create file with write properties
                envfile.write('time, Temperature (deg. C),Humidity (%RH),Temperature Outage?,Humidity Outage?\n') #write header
                envfile.write(str(currenttime[-1]))                    #add time of measurement
                envfile.write(','+str(temperature[-1]))                  #add latest temperature
                envfile.write(','+str(humidity[-1]))                   #add latest humidity
                envfile.write(', ,HUMIDITY OUTAGE')
                envfile.write('\n')
                envfile.close()                                  #close file


    #temperature and humidity outage log
    if (humidity[-1] > RHmax) or (humidity[-1] < RHmin) and (temperature[-1] > Tmax) or (temperature[-1] < Tmin): #if humidity and temperature outage, log humidity and temperature outage
        if len(tf_alert_RH) != 0 or len(tf_alert_T) !=0:
            if os.path.isfile(envdata_directory+'/'+monthYYYY+'-outages.env.csv'): #use existing monthYYYY.env.csv file
                envfile = open(envdata_directory+'/'+monthYYYY+'-outages.env.csv', 'a') #open file with append properties
                envfile.write(str(currenttime[-1]))                    #add time of measurement
                envfile.write(','+str(temperature[-1]))                  #add latest temperature
                envfile.write(','+str(humidity[-1]))                   #add latest humidity
                envfile.write(',TEMPERATURE OUTAGE,HUMIDITY OUTAGE')
                envfile.write('\n')
                envfile.close()                                  #close file
            else:                                               #otherwise create new monthYYYY.env.csv
                envfile = open(envdata_directory+'/'+monthYYYY+'-outages.env.csv', 'w') #create file with write properties
                envfile.write('time,Temperature (deg. C),Humidity (%RH),Temperature Outage?,Humidity Outage?\n') #write header
                envfile.write(str(currenttime[-1]))                    #add time of measurement
```

```
            envfile.write(','+str(temperature[-1]))                    #add latest temperature
            envfile.write(','+str(humidity[-1]))                       #add latest humidity
            envfile.write(',TEMPERATURE OUTAGE,HUMIDITY OUTAGE')
            envfile.write('\n')
            envfile.close()                                            #close file


    #////////////////////Communications with outside world\\\\\\\\\\\\\\\\\\\\\\\\
    ## Update NoContact list
    NoContact = []                                                     #reinitialize No Contact list
    labusers = copy.deepcopy(Contacts.labusers[labID])                 #reinitialize labusers
    with open(program_directory+'/NoContact.list') as openedfile:
        reader = csv.reader(openedfile, delimiter=',')
        filedata = list(zip(*reader))


    #sort through list, remove NoContact entries from labusers list
    r, nNoContact = np.shape(filedata)
    for i in range(nNoContact):
        NoContact.append(filedata[0][i])
        if len(NoContact) > 0:
            if NoContact[-1] in labusers:
                del labusers[labusers.index[NoContact[-1]]]


    ## Send scheduled test message to all users
    comparetime = datetime.datetime.strptime(time.strftime("%B %d, %Y %H:%M:%S"), "%B %d, %Y %H:%M:%S")
    if (TestmsgDate-comparetime) < datetime.timedelta(0,30*60) and (TestmsgDate-comparetime) >
datetime.timedelta(0,0): #if 30 minutes prior to sending test message
        if not(TestmsgSent):                                           #if test message has not been sent
            message = 'DMGalert test: This is a scheduled test message of the NIST Laboratory Environment Monitoring
and Alert System stationed in '+labID+'. No action is needed for this message. The current system time is
'+time.strftime('%a %b %d, %Y, %I.%M %p')+'. The current environment for '+labID+' is %.2f' % temperature[-1]+'
deg. C and %.2f' % humidity[-1]+' RH.'
            plt.savefig(program_directory+'/tmpimg/outage.jpg')        #save current figure
            for naddress in range(len(labcontacts)):
                SendMessageMMS(labcontacts, message, program_directory+'/tmpimg/outage.jpg') #send test message
with attached graph
            print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Sending scheduled test message to users...')
            TestmsgSent = True
    else:                                                              #if not 30 minutes before sending test message
        TestmsgSent = False                                            #set test message to has not been sent
```

```
    ## Temperature alerts

    #initial temperature outage

    if (temperature[-1] > Tmax) or (temperature[-1] < Tmin):                #if outside of temperature range

        if (temperature[-2] > Tmax) or (temperature[-2] < Tmin):            #if previous temperature was also out of
range

            if (labstatus_T == 'normal'):                                   #if lab status was previously normal, or there was an
ethernet outage

                message = 'DMG alert: the temperature for '+labID+' is outside the allowable range [%.2f' % Tmin+', %.2f'
% Tmax+'] deg. C. At '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST the environment was %.2f' % temperature[-
1]+' deg. C and %.2f' % humidity[-1]+' RH. The number for plant is 301 975 xxxx. A message will be sent should the
temperature status change.'

                plt.savefig(program_directory+'/tmpimg/outage.jpg')         #save current figure

                for naddress in range(len(labcontacts)):

                    try:

                        SendMessageMMS(labcontacts[naddress], message, program_directory+'/tmpimg/outage.jpg')
#connect to SMTP server and send outage message with graph attached

                    except Exception:                                       #if cannot reach internet to send messages, continue logging
and send outage alert when internet connection resumes

                        if not(ethoutage): #if this is

                            ethoutage_Toutmessage = 'DMG alert: An internet outage prevented the sending of a recent
temperature event. The temperature for '+labID+' was outside the allowable range [%.2f' % Tmin+', %.2f' % Tmax+']
deg. C. At '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST the environment was %.2f' % temperature[-1]+' deg. C
and %.2f' % humidity[-1]+' RH. The number for plant is 301 975 xxxx. A message will follow should the temperature
status have changed during the internet outage.'

                        ethoutage = True

                        pass

                print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Temperature outage detected...messages sent/queued
for users and temperature status elevated')

                labstatus_T = 'warning'                                     #elevate status

            elif labstatus_T == 'warning':                                  #if temperature was already out

                tf_alert_T = [time.time()]                                  #reset temperature alert timer

    #incremental tmeperature alerts

    if (temperature[-1] > TincAlert[1]) or (temperature[-1] < TincAlert[0]):    #if temperature has changed passed the
allowed increment

        if (temperature[-2] > TincAlert[1]) or (temperature[-2] < TincAlert[0]): #if previous temperature was also passed
the allowed increment

            if (temperature[-1] > Tmin) and (temperature[-1] < Tmax):           #if temperature is within spec

                TincAlert = [Tmin - TincSet, Tmax + TincSet]                    #reset TincAlert

            elif (temperature[-1] > TincAlert[1]):                              #if temperature increased

                plt.savefig(program_directory+'/tmpimg/outage.jpg')

                message = 'DMG alert update: the temperature for '+labID+' has increased. At '+time.strftime('%a %b %d,
%Y, %I.%M %p')+' EST the environment was %.2f' % temperature[-1]+' deg. C and %.2f' % humidity[-1]+' RH'
```

```
            TincAlert[1] = TincAlert[1] + TincSet                    #set new incremental alert parameters
            TincAlert[0] = TincAlert[1] - 2*TincSet
            for naddress in range(len(labcontacts)):
                try:
                    SendMessageMMS(labcontacts[naddress], message, program_directory+'/tmpimg/outage.jpg')
#connect to SMTP server and send outage message with graph attached
                except Exception:                                    #if cannot reach internet to send messages, continue logging
and send outage alert when internet connection resumes
                    if not(ethoutage): #if this is
                        ethoutage_Toutmessage = 'DMG alert: An internet outage prevented the sending of a recent
temperature event. The temperature for '+labID+' was outside the allowable range [%.2f' % Tmin+', %.2f' % Tmax+']
deg. C. At '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST the environment was %.2f' % temperature[-1]+' deg. C
and %.2f' % humidity[-1]+' RH. The number for plant is 301 975 xxxx. A message will follow should the temperature
status have changed during the internet outage.'
                    ethoutage = True
                    pass
            print('\n'+tim
        elif (temperature[-1] < TincAlert[0]):                       #if temperature decreased
            plt.savefig(program_directory+'/tmpimg/outage.jpg')
            message = 'DMG alert update: the temperature for '+labID+' has decreased. At '+time.strftime('%a %b %d,
%Y, %I.%M %p')+' EST the environment was %.2f' % temperature[-1]+' deg. C and %.2f' % humidity[-1]+' RH'
            TincAlert[0] = TincAlert[0] - TincSet                    #set new incremental alert parameters
            TincAlert[1] = TincAlert[0] + 2*TincSet
            for naddress in range(len(labcontacts)):
                try:
                    SendMessageMMS(labcontacts[naddress], message, program_directory+'/tmpimg/outage.jpg')
#connect to SMTP server and send outage message with graph attached
                except Exception:                                    #if cannot reach internet to send messages, continue logging
and send outage alert when internet connection resumes
                    if not(ethoutage): #if this is
                        ethoutage_Toutmessage = 'DMG alert: An internet outage prevented the sending of a recent
temperature event. The temperature for '+labID+' was outside the allowable range [%.2f' % Tmin+', %.2f' % Tmax+']
deg. C. At '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST the environment was %.2f' % temperature[-1]+' deg. C
and %.2f' % humidity[-1]+' RH. The number for plant is 301 975 xxxx. A message will follow should the temperature
status have changed during the internet outage.'
                    ethoutage = True
                    pass
            print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Temperature status changed...messages sent/queued
for users and temperature status elevated')


    #temperature outage under an internet outage
    if ethoutage:                                                    #if under internet outage, try to send queued messages
        for naddress in range(len(labcontacts)):
```

```
            try:
                SendMessageMMS(labcontacts[naddress], ethout_Toutmessage, program_directory+'/tmpimg/outage.jpg') #connect to SMTP server and send outage message with graph attached
                ethoutage_sent = True                                   #set status of messages queued under ethernet outage
            except Exception:                                           #if cannot reach internet to send messages, continue logging and send outage alert when internet connection resumes
                pass


    #temperature return to normal
    if (temperature[-1] < Tmax) and (temperature[-1] > Tmin):                #if temperature is inside temperature range
        if labstatus_T == 'warning':                                        #if lab status was previously under warning
            if len(tf_alert_T) == 0:                                        #if temperature alert timer has not been set
                tf_alert_T = [time.time()]                                  #set the start of the temperature alert timer
            elif (time.time() - tf_alert_T[0]) > normalstatus_wait*60:       #if normalstatus_wait has passed since temperature alert timer has start/reset
                message = 'DMG alert update: the temperature for '+labID+' returned to normal at '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST.'
                plt.savefig(program_directory+'/tmpimg/outage.jpg')         #save current figure
                for naddress in range(len(labcontacts)):
                    try:
                        SendMessageMMS(labcontacts[naddress], message, program_directory+'/tmpimg/outage.jpg') #connect to SMTP server and send outage message with graph attached
                    except Exception:
                        if not(ethoutage):
                            ethoutage_Tinmessage = 'DMG alert update: the temperature for '+labID+' returned to normal at '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST.'
                            ethoutage = True
                print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Temperature returned to normal...messages sent/queued to users and temperature status reduced')
                labstatus_T = 'normal'                                      #reduce status
                tf_alert_T = []                                             #remove temperature alert timer
    #temperature outage under an internet outage
    if ethoutage:                                                           #if under internet outage, try to send queued messages
        for naddress in range(len(labcontacts)):
            try:
                SendMessageMMS(labcontacts[naddress], ethout_Tinmessage, program_directory+'/tmpimg/outage.jpg') #connect to SMTP server and send outage message with graph attached
                ethoutage_sent = True                                   #set status of messages queued under ethernet outage
            except Exception:                                           #if cannot reach internet to send messages, continue logging and send outage alert when internet connection resumes
                pass
```

```
    ## Humidity alerts

    #initial humidity outage

    if (humidity[-1] > RHmax) or (humidity[-1] < RHmin):                    #if outside of humidity range

        if (humidity[-2] > RHmax) or (humidity[-2] < RHmin):                #if previous humidity was also out of range

            if labstatus_RH == 'normal':                                    #if lab status was previously normal

                message = 'DMG alert: the humidity for '+labID+' is outside the allowable range: [%.2f' % RHmin+', %.2f' % RHmax+'] RH. At '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST the environment was %.2f' % temperature[-1]+' deg. C and %.2f' % humidity[-1]+' RH.'

                plt.savefig(program_directory+'/tmpimg/outage.jpg')         #save current figure

                for naddress in range(len(labcontacts)):

                    try:

                        SendMessageMMS(labcontacts[naddress], message, program_directory+'/tmpimg/outage.jpg') #connect to SMTP server and send outage message with graph attached

                    except Exception:

                        if not(ethoutage):

                            ethoutage_RHoutmessage = 'DMG alert: An internet outage prevented the sending of a recent humidity event. The humidity for '+labID+' was outside the allowable range [%.2f' % Tmin+', %.2f' % Tmax+'] deg. C. At '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST the environment was %.2f' % temperature[-1]+' deg. C and %.2f' % humidity[-1]+' RH. A message will follow should the humidity status have changed during the internet outage'

                        ethoutage = True

                print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Humidity outage detected...messages sent/queued to users and humidity status elevated')

                labstatus_RH = 'warning'                                    #elevate status

            elif labstatus_RH == 'warning':                                 #if humidity was already out

                tf_alert_RH = [time.time()]                                 #reset humidity alert timer

    #incremental humidity alerts

    if (humidity[-1] > RHincAlert[1]) or (humidity[-1] < RHincAlert[0]):        #if humidity has changed passed the allowed increment

        if (humidity[-2] > RHincAlert[1]) or (humidity[-2] < RHincAlert[0]):    #if previous humidity was also passed the allowed increment

            if (humidity[-1] > RHmin) and (humidity[-1] < RHmax):               #if humidity is within spec

                RHincAlert = [RHmin - RHincSet, RHmax + RHincSet]               #reset TincAlert

            elif (humidity[-1] > RHincAlert[1]):                                #if humidity increased

                plt.savefig(program_directory+'/tmpimg/outage.jpg')

                message = 'DMG alert update: the humidity for '+labID+' has increased. At '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST the environment was %.2f' % temperature[-1]+' deg. C and %.2f' % humidity[-1]+' RH'

                RHincAlert[1] = RHincAlert[1] + RHincSet                        #set new incremental alert parameters

                RHincAlert[0] = RHincAlert[1] - 2*RHincSet

                for naddress in range(len(labcontacts)):

                    try:
```

```
                SendMessageMMS(labcontacts[naddress], message, program_directory+'/tmpimg/outage.jpg')
#connect to SMTP server and send outage message with graph attached

              except Exception:                                    #if cannot reach internet to send messages, continue logging
and send outage alert when internet connection resumes

                if not(ethoutage): #if this is

                  ethoutage_RHoutmessage = 'DMG alert: An internet outage prevented the sending of a recent
humidity event. The humidity for '+labID+' was outside the allowable range [%.2f' % Tmin+', %.2f' % Tmax+'] deg. C.
At '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST the environment was %.2f' % temperature[-1]+' deg. C and
%.2f' % humidity[-1]+' RH. A message will follow should the humidity status have changed during the internet outage.'

                ethoutage = True

                pass

            print('\n'+tim

          elif (humidity[-1] < RHincAlert[0]):                          #if humidity decreased

            plt.savefig(program_directory+'/tmpimg/outage.jpg')

            message = 'DMG alert update: the humidity for '+labID+' has decreased. At '+time.strftime('%a %b %d, %Y,
%I.%M %p')+' EST the environment was %.2f' % temperature[-1]+' deg. C and %.2f' % humidity[-1]+' RH'

            RHincAlert[0] = RHincAlert[0] - RHincSet                    #set new incremental alert parameters

            RHincAlert[1] = RHincAlert[0] + 2*RHincSet

            for naddress in range(len(labcontacts)):

              try:

                SendMessageMMS(labcontacts[naddress], message, program_directory+'/tmpimg/outage.jpg')
#connect to SMTP server and send outage message with graph attached

              except Exception:                                    #if cannot reach internet to send messages, continue logging
and send outage alert when internet connection resumes

                if not(ethoutage): #if this is

                  ethoutage_RHoutmessage = 'DMG alert: An internet outage prevented the sending of a recent
temperature event. The temperature for '+labID+' was outside the allowable range [%.2f' % Tmin+', %.2f' % Tmax+']
deg. C. At '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST the environment was %.2f' % temperature[-1]+' deg. C
and %.2f' % humidity[-1]+' RH. The number for plant is 301 975 xxxx. A message will follow should the temperature
status have changed during the internet outage.'

                ethoutage = True

                pass

            print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Humidity status changed...messages sent/queued for
users and temperature status elevated')

        if ethoutage:                                               #if under internet outage, try to send queued messages

          for naddress in range(len(labcontacts)):

            try:

              SendMessageMMS(labcontacts[naddress], ethout_RHoutmessage, program_directory+'/tmpimg/outage.jpg')
#connect to SMTP server and send outage message with graph attached

              ethoutage_sent = True                                   #set status of messages queued under ethernet outage

            except Exception:                                       #if cannot reach internet to send messages, continue logging
and send outage alert when internet connection resumes

              pass
```

```
    #humidity return to normal
    if (humidity[-1] < RHmax) and (humidity[-1] > RHmin):                    #if humidity is inside humidity range
        if labstatus_RH == 'warning':                                        #if lab status was previously under humidity warning
            if len(tf_alert_RH) == 0:                                        #if humidity alert timer has not been set
                tf_alert_RH = [time.time()]                                  #set the start of the humidity alert timer
            elif (time.time() - tf_alert_RH[0]) > normalstatus_wait*60:       #if normalstatus_wait has passed since humidity alert timer has start/reset
                message = 'DMG alert update: the humidity for '+labID+' returned to normal at '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST.'
                plt.savefig(program_directory+'/tmpimg/outage.jpg')          #save current figure
                for naddress in range(len(labcontacts)):
                    try:
                        SendMessageMMS(labcontacts[naddress], message, program_directory+'/tmpimg/outage.jpg') #connect to SMTP server and send outage message with graph attached
                    except Exception:
                        if not(ethoutage):
                            ethoutage_RHinmessage = 'DMG alert update: the humidity for '+labID+' returned to normal at '+time.strftime('%a %b %d, %Y, %I.%M %p')+' EST.'
                        ethoutage = True
                print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Humidity returned to normal...messages sent/queued to users and humidity status reduced')
                labstatus_RH = 'normal'                                      #reduce status
                tf_alert_RH = []                                             #remove humidity alert timer
    if ethoutage:                                                            #if under internet outage, try to send queued messages
        for naddress in range(len(labcontacts)):
            try:
                SendMessageMMS(labcontacts[naddress], ethout_RHinmessage, program_directory+'/tmpimg/outage.jpg') #connect to SMTP server and send outage message with graph attached
                ethoutage_sent = True                                        #set status of messages queued under ethernet outage
            except Exception:                                                #if cannot reach internet to send messages, continue logging and send outage alert when internet connection resumes
                pass


    #reset ethoutage and ethoutage_sent statuses if queued messages were successfully sent
    if ethoutage:
        if ethoutage_sent:
            print('\n'+time.strftime("%Y-%m-%d %H:%M:%S")+' : Internet connection reestablished')
            ethoutage = False
            ethoutage_sent = False
```

```
    tf = time.time()                                                    #stop
timer----------------------------------------------------------------------------

    if sleeptimer-(tf-ti) > 0:

        time.sleep(sleeptimer-(tf-ti))                                  #sleep for sleeptimer less time taken for the above lines

#end of while
```