Raspberry Pi Dual Path Monitoring for Honeywell Vista Security Systems

Overview

This document describes a fun and useful "home-baked" Raspberry Pi project and who doesn't like a nice slice of pie? Some may perhaps be derisive and just "blow a raspberry", but if you take the time to read this document, it may prove to be of interest.

A security system is said to be "dual path enabled" when it has *both* IP (Internet Protocol) and cellular communication capability. Should Internet connectivity be lost, the security system can still send alerts, via a cellular communication network, typically to one or more cellular ("mobile") telephones. In this document, the acronym DPM stands for "Dual Path Monitoring".

The project consists of an open-source Python application, a Raspberry Pi SBC (single board computer), a cellular modem and an Eyezon EnvisaLink-4 ("EVL4") IP module that, in combination, deliver *self*-DPM for a Honeywell Vista security system. Consequently, no Central Monitoring Station services are required or included.

Which Cellular Modem?

A bewildering variety of cellular modems are available from several, different manufacturers. Initially, it was also unclear which communications provider would ultimately be used. Therefore, the first step was to visit the websites of AT&T, T-Mobile and Verizon (the three largest US carriers) to identify a LTE Cat-1, LTE Cat-M1, or LTE NB-IoT cellular module *already certified* on *all three* carrier's networks. See Appendix A for website links. For the DPM python application to work as designed, the target cellular modem must have *two* ports that process AT commands, which most do.

Short Message Service (SMS)

Not all Internet-of-Things (IoT) service plans provide SMS functionality. The major carriers, (AT&T, T-Mobile and Verizon) have various IoT offerings, some of which do include SMS. Other service providers, for example, Twilio, offer IoT plans with SMS. It is important to read the "fine print" about carrier coverage areas, billing rates per active Subscriber Identification Module (SIM card) and for each SMS sent and received, before deciding which service provider to use. See Appendix A for links to some relevant websites.

Complete Kits vs. Individual Components

Various Raspberry Pi cellular modem hats and even complete IoT project kits are available from vendors such as Waveshare, Sixfab and others – see Appendix A. However, after some research, for this low-budget endeavor, most components were purchased on AliExpress and eBay.

Project Hardware
- EnvisaLink-4 ("EVL4") IP interface module
- Raspberry Pi model 2B with 32GB micro SD card
- Raspberry Pi Hat with mPCIe socket for a cellular modem
- Telit LE910-NA1 mPCIe cellular modem and IoT SIM card
- pair of u.FL to SMA pigtails
- 4G LTE MIMO antenna with 15 ft of coaxial cable and SMA connectors

Project Hardware -- Pictures And Notes

4G LTE mPCIe Hat (aliexpress.com ~ $15 to $20 in 2022)
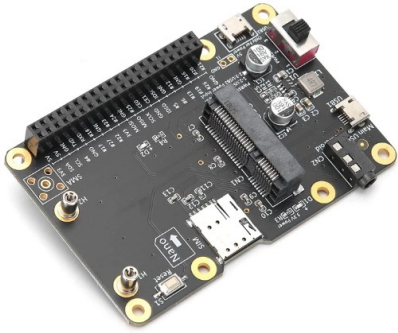
4G LTE Base Hat
Size: approx. 8.5x5.5x2.5 cm / 3.3x2.2x1 inch
For Raspberry Pi 4, 3, 2, B+, Asus Tinker Board,
Rock 64, ARTIK's Board, LattePanda
Compatible with the following mini PCI-E modules:
Quectel: EC25/EC21/ EC20 /UC20/ LTE-EP06
Sierra: AirPrime MC Series
Telit: LM960, LE910V2, HE910, LE910Cx
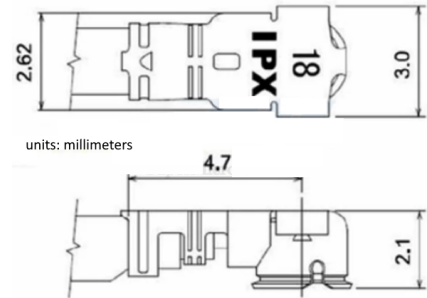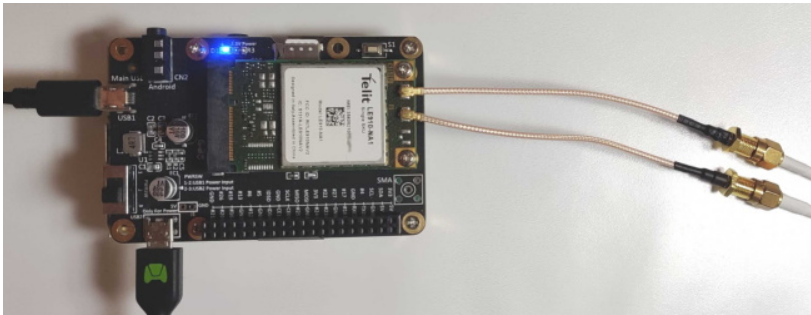Huawei: ME909s-120, ME909s-821, etc.
Simcom: SIM7100, SIM7230, etc.
ZTE: ZM8620, etc.
U-Blox: MPCI-L2 Series
Thales: PLS62W, mPLS8, mPLAS9

Telit LE910-NA1 (ebay.com, used ~ $20 in 2022)　　　**u.FL-SMA pigtails** (aliexpress.com ~ $1 each in 2022)

4G LTE Antenna (aliexpress.com ~ $10 to $15 in 2022)

Note: u.FL connectors come in different sizes !

4G LTE WiFi external panel MIMO antenna 700-2600MHz with coaxial cables of 1 to 5 meter (3 to 15 feet approx.) in length, depending on specific model. Approximately 5 inches x 9 inches x 1 inch.

EnvisaLink-4 IP module (various suppliers ~ $100 in 2022)
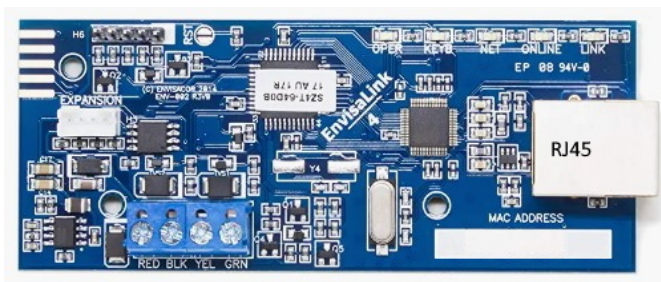
Eyezon EnvisaLink-4 ("EVL4"). This IP interface module measures about 4" long by 1.5" wide and has a single RJ45 connector for wired Ethernet connectivity.

It can be purchased at eyezon.com, amazon.com, alarmgrid.com and other online vendors.

Preparing the Raspberry Pi

This project used 32-bit "Buster" with Desktop (debian V10, kernel V5.10) as the operating system. The "lite" version is not recommended because utilities such as File Manager, Geany, Text Editor, Thonny, etc., bundled with the Desktop edition are very useful when initially building, configuring and testing the DPM application.

The 32-bit OS is compatible with all Pi models (https://www.raspberrypi.com/software/operating-systems), and for this project, the target, previously purchased, Raspberry Pi model 2B only had a 32-bit processor and 1GB of RAM, but that proved to be more than adequate.

Although Pi OS "Bullseye" (debian V11, kernel V5.15) is more current and comes with Python V3.9, it was (as of the publication date of this document) accompanied by a not-yet-patched, linux kernel issue, where `/dev/ttyACM0` does not function correctly, but must do so, in order for the DPM.py application to work as designed.

Micro SD Card Plus Raspberry Pi Imager

Purchase (e.g. from amazon.com, newegg.com, or similar) a high quality micro SD card, for example a 32GB Samsung Pro Endurance. Install Raspberry Pi Imager (https://www.raspberrypi.com/software) on your Windows/MacOS/Ubuntu computer and image Pi OS onto the micro SD card.
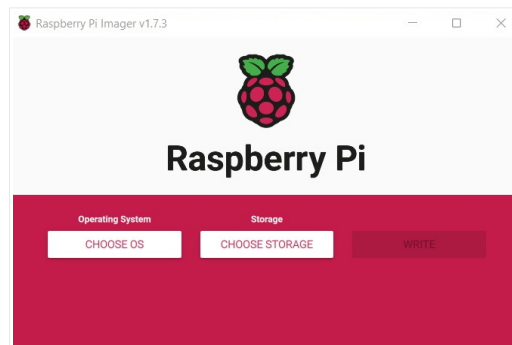
Pi OS Configuration
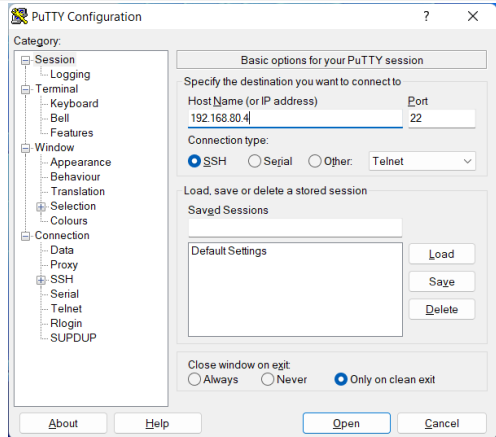
The following instructions assume the computer name of the Raspberry Pi is "`raspberrypi`" and that the user account name is "`pi1`". If different, the actual details for *your* Raspberry Pi will need to be substituted accordingly.

This project was successfully developed and tested with the following requisite software components:

- 32-bit Pi OS "Buster" (with Desktop)
- python V3.9.15
- pip V18.1
- pygtail V0.12.0
- pySerial V3.5
- luhn V0.2.0
- telnetlib
  (the python Telnet library is to be deprecated after Python V3.11 https://peps.python.org/pep-0594/#telnetlib .
  However, as an insurance policy, a copy of telnetlib source code will be stored with this project.)

| Step	Instructions / Command Line	Comments
1	image the micro SD card using Raspberry Pi Imager	32-bit Pi-OS "Buster" release with Desktop
2	Boot the Ras Pi and perform initial configuration including the download and installation of any required software updates.	Ensure there is an Internet connection !
3	`sudo raspi-config`	Enable: Boot-to-Desktop, Network-at-Boot and SSH (Secure Shell). Disable: Screen-Blanking, Splash Screen.

4	install Python V3.9.15
	
	`cd $HOME`
	
	`wget https://www.python.org/ftp/python/3.9.15/Python-3.9.15.tgz`
	
	`tar -xzvf Python-3.9.15.tgz`
	
	`cd Python-3.9.15`
	
	`./configure --enable-optimizations`
	
	`sudo make altinstall`
	
	`sudo ln -sf /usr/local/bin/python3.9 /usr/bin/python`
	
	`python -V  (should respond with Python 3.9.15)`
		As of the publication date of this document, using Python V3.10.x was problematic as seemingly, it does not include SSL or TLS and pypi.org requires https for installs, via pip.
		
		The make process for Python 3.9.15 takes about 30 to 45 minutes.
		
		(Note: each of the commands opposite should be entered as a single line)

5	`pip3 -V` `pip 18.1 from /usr/lib/python3/dist-packages/pip (python 3.7)`	check for pip3 presence and version. We'll use this version (18.1) of pip for the next step.

6	Install Pyserial and Pygtail and luhn
	
	`sudo pip3 install --target=/usr/local/lib/python3.9  Pyserial`
	
	`sudo pip3 install --target=/usr/local/lib/python3.9  Pygtail`
	
	`sudo pip3 install --target=/usr/local/lib/python3.9  luhn`
		"Pyserial" is for serial communications
		
		"Pygtail" is for 'tailing' the EVL4's syslog file
		
		"luhn" is a check-digit module.
		
		(Note: each of the commands opposite should be entered as a single line)

7	`cd $HOME`
	
	`mkdir -p DPM-EVL4/logs`
		create target deployment folders

8	verify the Pi hat and cellular modem are powered-up and `ls -l /dev/ttyACM*` to find how the modem presents itself. Add user "pi1" to the requisite group.	For example: `sudo usermod -a -G dialout pi1` see Appendix B
9	`sudo cp /etc/rsyslog.conf  /etc/rsyslog.conf.001`	Make backup copy of rsyslog.conf
10	`sudo cp /etc/rc.local  /etc/rc.local.001`	Make backup copy of rc.local
11	`sudo cp /etc/crontab   /etc/crontab.001`	Make backup copy of crontab
12	`sudo cp /boot/cmdline.txt /boot/cmdline.txt.001`	Make a backup copy of cmdline.txt
13	Modify rc.local, crontab and rsyslog.conf as described in Appendix B and finally, restart rsyslog. `sudo systemctl restart rsyslog`	Hint: run FileManager as root (`sudo pcmanfm`) and then invoke Geany to make these changes, since it is easier than using a command-line editor.

Step	Instructions / Command Line	Comments
14	Edit `/boot/cmdline.txt` and ensure it is a single line of text that includes: `fsck.mode=force fsck.repair=yes` Hint: `sudo pcmanfm` then invoke Geany to edit this file.	Forcing a file system check at each boot, with potentially some repairs, will add about 30 seconds delay (for a 32GB micro SD card), but significantly reduces the risk of the Ras Pi not booting properly at all.
15	Configure the EVL4's syslog client as described in Appendix B	

16	Copy the DPM project components from GitHub into
	
	`$HOME/DPM-EVL4`
		`cid.py`
		`config.py`
		`DPM.ini`
		`DPM-metrics.ini`
		`DPM.py`
		`DPM.sh`
		`mymodem.py`
		`pingone.py`
		`protectPassword.py`
		`scanEVL4log.py`
		`startup.py`
		`stupefy.py`
		`telnetEVL4.py`

17	Check all file permissions! See Appendix B for details.	`DPM-metrics.ini` **must** be write-able by user "`pi1`"
		  `chmod 644 DPM-metrics.ini`
		`DPM.sh` **must** be executable by user "`pi1`"
		  `chmod 744 DPM.sh`

18	Carefully edit the configuration file (`DPM.ini`) to match your specific requirements. Start with **DEBUG** as the logging level.	Hint: use Geany, via File Manager. Remember to also globally change "`pi1`" to your Ras Pi user ID.
19	Double check everything.	Including IP addresses of the EVL4 and Ras Pi, DHCP MAC to IP address mappings and firewall rules in the sub-net's router.
20	Reboot the Ras Pi and check all log files in `$HOME/DPM-EVL4/logs`	Hint: to check that DPM is running: `ps -ef	grep DPM.py	grep -v grep`
21	If it is working correctly, congratulate yourself; if not, troubleshoot!	Set the logging level in the configuration file (`DPM.ini`) to **INFO** once you've checked daily for a week or two that everything is working correctly.
22	Optionally, install Putty – can to talk to the cellular modem directly and facilitate trouble shooting. See Appendix B	`sudo apt install putty`
23	After final physical deployment of the Ras Pi into the home security system enclosure, install Putty on your Windows/Linux desktop/notebook computer. Patch into the same network subnet and communicate with the Ras Pi using SSH (secure shell) plus the "`pi1`" account and password. This will facilitate making any final changes to the `DPM.ini` configuration file and/or perusing log files. In the *example* shown to the right, the IP V4 address of the Raspberry Pi is `192.168.80.4`	

Physical Installation

In many home security systems, the control unit is housed in a sturdy metal enclosure with a lockable door, as illustrated by the Honeywell Vista-20P starter-kit, on the right.
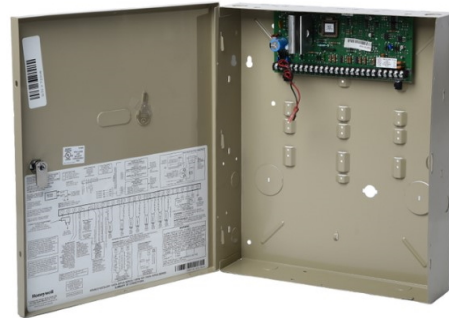
When zone-expansion modules are inside the cabinet, along with wiring and a "backup" battery, there may be little or no free space. If so, this Ras Pi project could be deployed in a small (`approx. 5 inch x 3 inch x 2 inch`) or larger, ventilated, plastic box, attached to the exterior of the main cabinet.

For better overall security, an additional empty, lockable enclosure can be procured, (Google search "Honeywell Vista enclosure only") to accommodate the Ras Pi and associated networking gear and subsequently connected by a data cable to the main box. To avoid electrical faults and/or short circuits, the Ras Pi must be mounted using plastic *standoffs* (a.k.a. "offsets", or "legs"), or against a layer of non-conductive material (e.g. plastic), *not* directly against the metal case.

Ideally, the security system should have it's own network sub-net (e.g. 192.168.nnn.nnn), supported by a small (e.g. 4-port) router, with appropriate firewall rules in effect, connected to an Internet Gateway. Use DHCP with MAC to IP address mappings, to assign consistent IP addresses to the Ras Pi and EVL4.

Wired Ethernet (e.g. Cat-5 cable) to the Internet Gateway is the more robust, reliable and secure way to establish connectivity, not least because "wireless anything" does not work well inside a metal security cabinet and external antennas are then required.

A small, pre-owned, non-WiFi router (e.g. from ebay.com) will be fine for this purpose. WiFi routers typically have some RJ45 ports for Ethernet cable connections and their WiFi can usually be turned off via the router's software configuration program, as desired. Regardless, first verify dimensions to ensure it will fit inside the cabinet, if that's to be it's final location.

Power Supply

Using a higher current (approx 5 amp, USB-style) 5V DC power supply to drive the Raspberry Pi facilitates powering the Pi Hat *and* cellular modem from a USB port on the Raspberry Pi itself. The overall system will be less likely to experience "low voltage" events on the infrequent occasions when greater current is pulled. With this configuration, when the Ras Pi is rebooted, the hat is power-cycled and therefore, so is the cellular modem. The modem has its own CPU or MCU and firmware, so rebooting daily will help reduce problems due to memory leaks and similar ailments.

Depending upon the position of the slide-switch, the hat used for this project can power the cellular modem from an independent 5V DC feed. Setting "`ModemSoftReboot = True`" in the `DPM.ini` configuration file will cause the modem to be soft-rebooted by the `startup.py` process, as invoked by the Housekeeping script (`DPM.sh`). Similarly, setting **"`TelEVL4SoftReboot = True`"** in the `DPM.ini` configuration file will cause the EVL4 IP interface module to be soft-rebooted.

Appendix A: Internet Resources

www.raspberrypi.com	Official website for Raspberry Pi hardware, software and support.
https://www.raspberrypi.com/software/	Download the Raspberry Pi Imager and other software from here
www.eyezon.com	Eyezon's website for the EnvisaLink ("EVL4") IP interface module and much more.
http://forum.eyez-on.com/FORUM/	Eyezon's User Forum (Q & A, plus much more)
https://forum.eyezon.com/viewtopic.php?t=2143&sid=6863e05baafe62ccd9b872dff308b03e	Eyezon's instructions to install and configure an EVL4 IP module on a Honeywell Vista series security system
https://forum.eyezon.com/viewtopic.php?t=301&sid=6863e05baafe62ccd9b872dff308b03e	Eyezon's EVL4 Third Party Interface ("TPI") specification
https://www.alarmgrid.com/	Alarm Grid's website with security system products and services for DIY enthusiasts.
https://www.alarmliquidators.com/	Alarm Liquidator's website with security system products and services for DIY enthusiasts.
https://www.business.att.com/content/other/ATT_Approved_Modules.xlsx	modules, chip-sets and devices certified by AT&T (download-able spreadsheet)
https://wholesale.t-mobile.com/iot-solutions/modules-chipsets-devices	modules, chip-sets and devices certified by T-Mobile
https://opendevelopment.verizonwireless.com/design-and-build/approved-modules	modules, chip-sets and devices certified by Verizon
https://sixfab.com/shop/	SixFab's store for modems, antennas, Raspberry Pi kits, etc.
https://docs.sixfab.com/	SixFab's documentation about Raspberry Pi kits and projects
https://sixfab.com/product/connect-sim-card-for-iot-projects/	SixFab's SIM offering and pricing (no SMS service as of publication date of this document)
https://www.waveshare.com/product/raspberry-pi/hats.htm	Waveshare's broad catalog of Raspberry Pi hats.
https://www.waveshare.com/wiki/Main_Page#Long-range-wireless-comm	Waveshare's wiki that documents a broad range of IoT devices (mainly surface-mounted, not mPCIe)
https://www.twilio.com/docs/iot/supersim	Twilio's "super SIM" offering
https://www.twilio.com/iot/super-sim-card-pricing	Twilio's "super SIM" pricing structure
https://www.twilio.com/docs/iot/supersim	Twilio's "super SIM" documentation with detailed tutorials about Ras Pi and other IoT projects
https://www.freedesktop.org/wiki/Software/ModemManager	Information about Modem Manager – a useful testing utility.
https://modemmanager.org	Information about Modem Manager – a useful testing utility.

Appendix B: Ras Pi Filesystem Folders, Pi OS Changes, Etc.

Cellular modems are complex devices. To avoid being trapped in an information desert, ensure you obtain a copy of the technical documentation, especially, the "AT" command set, for a cellular modem *before* actually purchasing it!

Modem Manager ("mmcli") is a useful testing tool that can provide some, albeit limited, insights about a given cellular modem. Putty can also be used to communicate directly with serial devices.

Cellular modem modules present themselves as different devices depending on operating systems (linux, windows, etc) and the specific device driver being used. For the Telit modem used in this project, ACM0 and ACM3 are serial ports, that process "AT" commands.

```
pi1@raspberrypi: $ ls -l /dev/ttyACM*

crw-rw----  1 root dialout 166,   0 mmm dd hh:mm ttyACM0
crw-rw----  1 root dialout 166,   1 mmm dd hh:mm ttyACM1
crw-rw----  1 root dialout 166,   2 mmm dd hh:mm ttyACM2
crw-rw----  1 root dialout 166,   3 mmm dd hh:mm ttyACM3
crw-rw----  1 root dialout 166,   4 mmm dd hh:mm ttyACM4
crw-rw----  1 root dialout 166,   5 mmm dd hh:mm ttyACM5

The (default) user "pi1" should be added to the "dialout" group to avoid having to run
processes accessing the cellular modem as "root" :    sudo usermod -a -G dialout pi1

pi1@raspberrypi: $ grep pi1 /etc/group | grep -v grep

adm:x:4:pi1
dialout:x:20:pi1
cdrom:x:24:pi1
sudo:x:27:pi1
audio:x:29:pulse,pi1
blah … blah … blah

For a Raspberry Pi with the default computer name ("raspberrypi") and default user name
("pi1"), home looks like this:

pi1@raspberrypi:~ $ cd $HOME
pi1@raspberrypi:~ $ pwd
/home/pi1

The default folders beneath home are shown below.  The new folder "DPM-EVL4" was created
to accommodate the python application components and its operational log files.  The date
and time stamps have been generalized here for convenience.

drwxr-xr-x 2 pi1 pi1 4096 mmm dd hh:mm Bookshelf
drwxr-xr-x 2 pi1 pi1 4096 mmm dd hh:mm Desktop
drwxr-xr-x 2 pi1 pi1 4096 mmm dd hh:mm Videos
drwxr-xr-x 2 pi1 pi1 4096 mmm dd hh:mm Public
drwxr-xr-x 2 pi1 pi1 4096 mmm dd hh:mm Pictures
drwxr-xr-x 3 pi1 pi1 4096 mmm dd hh:mm Music
drwxr-xr-x 2 pi1 pi1 4096 mmm dd hh:mm Documents
drwxr-xr-x 2 pi1 pi1 4096 mmm dd hh:mm Downloads
drwxr-xr-x 4 pi1 pi1 4096 mmm dd hh:mm DPM-EVL4
```

```
pi1@raspberrypi:~/DPM-EVL4    (the DPM application's components reside here)


-rw-r--r-- 1 pi1 pi1 22281 mmm dd hh:mm cid.py
-rw-r--r-- 1 pi1 pi1 12018 mmm dd hh:mm config.py
-rw-r--r-- 1 pi1 pi1  4426 mmm dd hh:mm DPM.ini
-rw-r--r-- 1 pi1 pi1   427 mmm dd hh:mm DPM-metrics.ini  <-----  must be writable
-rw-r--r-- 1 pi1 pi1 27157 mmm dd hh:mm DPM.py
-rwxr-xr-x 1 pi1 pi1  2901 mmm dd hh:mm DPM.sh          <-----  must be executable
-rw-r--r-- 1 pi1 pi1  8104 mmm dd hh:mm mymodem.py
-rw-r--r-- 1 pi1 pi1  1465 mmm dd hh:mm pingone.py
-rw-r--r-- 1 pi1 pi1  1498 mmm dd hh:mm protectPassword.py
-rw-r--r-- 1 pi1 pi1  3967 mmm dd hh:mm scanEVL4log.py
-rw-r--r-- 1 pi1 pi1  3389 mmm dd hh:mm startup.py
-rw-r--r-- 1 pi1 pi1  2625 mmm dd hh:mm stupefy.py
-rw-r--r-- 1 pi1 pi1 20278 mmm dd hh:mm telnetEVL4.py
drwxrwxrwx 2 pi1 pi1  4096 mmm dd hh:mm logs


pi1@raspberrypi:~/DPM-EVL4/logs   (the DPM application's log files reside here)

-rw-r--r-- 1 pi1 pi1   670 mmm dd hh:mm DPM-HK-startup-yyyymmddhhmmss.log
-rw-r--r-- 1 pi1 pi1   163 mmm dd hh:mm DPM-HK-yyyymmddhhmmss.log
-rw-r--r-- 1 pi1 pi1 24958 mmm dd hh:mm DPM-yyyymmddhhmmss.log
-rw-rw-rw- 1 pi1 pi1   211 mmm dd hh:mm EVL4.log
-rw-r--r-- 1 pi1 pi1    11 mmm dd hh:mm EVL4.offset
```

Example of /etc/crontab file that reboots the Raspberry Pi daily at midnight

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *   * * * root    cd / && run-parts --report /etc/cron.hourly
25 6   * * * root   test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.daily )
47 6   * * 7 root   test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.weekly )
52 6   1 * * root   test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.monthly )
#
0 0 * * * root reboot now
```

Example of /etc/rc.local that invokes DPM.sh post reboot
Note: for better security, do not run DPM.sh and DPM.py as "root", but as "pi1" instead.

```
#!/bin/sh -e
## rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

su pi1 -c /home/pi1/DPM-EVL4/DPM.sh

exit 0
```


Example of /etc/rsyslog.conf modifications to receive output from the EVL4's syslog client

```
# /etc/rsyslog.conf configuration file for rsyslog
blah  … blah  … blah
################
#### MODULES ####
################
module(load="imuxsock") # provides support for local system logging
module(load="imklog")   # provides kernel logging support
#module(load="immark")  # provides --MARK-- message capability

# provides UDP syslog reception
module(load="imudp")
input(type="imudp" port="514")
# provides TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")
##########################
#### GLOBAL DIRECTIVES ####
##########################
blah … blah … blah
##############
#### RULES ####
##############
# First some standard log files.  Log by facility.
# Note: the minus sign prefix turns OFF file write synchronization
auth,authpriv.*           /var/log/auth.log
*.*;auth,authpriv.none    -/var/log/syslog
#cron.*                   /var/log/cron.log
daemon.* -/var/log/daemon.log
kern.*                    -/var/log/kern.log
lpr.*                     -/var/log/lpr.log
mail.*                    -/var/log/mail.log
user.*                    -/var/log/user.log
local0.*                   /home/pi1/DPM-EVL4/logs/EVL4.log

blah … blah … blah
```

EVL4's Web-browser Interface To Configure Syslogging

Open an internet browser (Chrome/Firefox/Edge), enter the IP V4 address of the EVL4 module (192.168.80.2 in this example) and then log into the EVL4 with the credentials "user" and the EVL4's *cleartext* password.

EnvisALERTS

E n v i s a L i n k   4

Home| Network

Network Parameters

IP Address   192.168.80.002   <--------- example IP V4 address of the EVL4 IP module

Network Mask   255.255.255.0
Gateway   192.168.80.1   <----------- example IP V4 address of this subnet's router
DNS Server   192.168.1.254
DHCP Status   BOUND - -1364
Make Network Settings Static? ☐ SUBMIT

Change User Password [   ] CHANGE
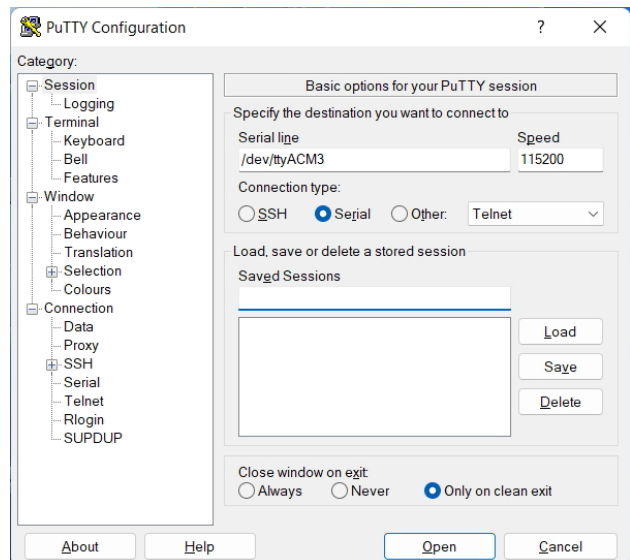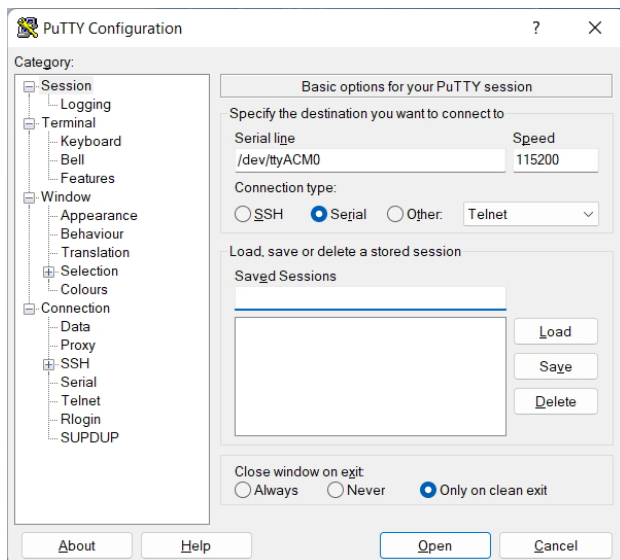
EnvisAlerts Status

Envisalerts Server   198.nnn.nnn.nnn

ONLINE

Syslog Client

Server IP Address   192.168.80.004   <--- example IP V4 address of the Raspberry Pi
                                                                              in this subnet.
Facility (16-23, 0 = OFF)   16   <------------ Facility 16 equates to local0 in rsyslog.conf
                                                                              on the Raspberry Pi
                          CHANGE

Examples of Putty talking to cellular modem serial ports

Appendix C: Generalized Sample DPM.py (INFO-level) Log File

```
yyyy-mm-dd hh:mm:ss CFG-001I Configuration object created (logging level: INFO)
yyyy-mm-dd hh:mm:ss SCN-001I ScanEVL4Log object created
yyyy-mm-dd hh:mm:ss PNG-001I PingOne object created
yyyy-mm-dd hh:mm:ss CID-001I DecodeCID object created
yyyy-mm-dd hh:mm:ss MDM-001I MyModem object created
yyyy-mm-dd hh:mm:ss TEL-001I TelnetEVL4 object created

yyyy-mm-dd hh:mm:ss DPM-001I Dual Path Monitoring activated …
yyyy-mm-dd hh:mm:ss DPM-015I sending to cell phone: 19990001212 DPM-001I Dual Path
Monitoring activated ...

yyyy-mm-dd hh:mm:ss DPM-002I Alert: FIRE, SYSTEM READY, SYSTEM TROUBLE, DISARMED !
Partition:01, New event:373 Fire trouble, Zone:Ground Floor Smoke Detectors
yyyy-mm-dd hh:mm:ss DPM-015I sending to cell phone: 19990001212 FIRE, SYSTEM READY, SYSTEM
TROUBLE, DISARMED ! Partition:01, New event:373 Fire trouble, Zone:Ground Floor Smoke
Detectors

yyyy-mm-dd hh:mm:ss DPM-002I Alert: SYSTEM READY, DISARMED ! Partition:01, Restore
event:373 Fire trouble, Zone:Ground Floor Smoke Detectors

yyyy-mm-dd hh:mm:ss DPM-002I Alert: ARMED STAY, NIGHT-STAY ! Partition:01, Close event:441
Armed Stay, User:Me
yyyy-mm-dd hh:mm:ss DPM-015I sending to cell phone: 19990001212 ARMED STAY, NIGHT-STAY !
Partition:01, Close event:441 Armed Stay, User:Me

yyyy-mm-dd hh:mm:ss DPM-002I Alert: SYSTEM READY, DISARMED ! Partition:01, Open event:441
Armed Stay, User:Me

yyyy-mm-dd hh:mm:ss DPM-004I Decoded syslog CID is: Partition:01, New event:604 Fire test,
User:Me
yyyy-mm-dd hh:mm:ss DPM-015I sending to cell phone: 19990001212 Partition:01, New event:604
Fire test, User:Me


* actual phone number has been replaced with a fictitious one.
```

Appendix D: Generalized Sample DPM.py (DEBUG-level) Log File

```
yyyy-mm-dd hh:mm:ss CFG-001I Configuration object created (logging level: DEBUG)
yyyy-mm-dd hh:mm:ss SCN-001I ScanEVL4Log object created
yyyy-mm-dd hh:mm:ss PNG-001I PingOne object created
yyyy-mm-dd hh:mm:ss CID-001I DecodeCID object created
yyyy-mm-dd hh:mm:ss MDM-001I MyModem object created
yyyy-mm-dd hh:mm:ss MDM-002I Modem connection is: Serial<id=0x764df0d0, blah … blah … blah
yyyy-mm-dd hh:mm:ss MDM-003I Modem replied "OK" to: ATE0
yyyy-mm-dd hh:mm:ss MDM-004I Modem replied "OK" to: AT+CSCS=GSM
yyyy-mm-dd hh:mm:ss MDM-005I Modem replied "OK" to: AT+CMGF=1
yyyy-mm-dd hh:mm:ss blah … blah … blah
yyyy-mm-dd hh:mm:ss TEL-001I TelnetEVL4 object created
yyyy-mm-dd hh:mm:ss TEL-002I Telnet connection:<telnetlib.Telnet object at 0x000002A945CC1330>
yyyy-mm-dd hh:mm:ss TEL-003I Reply from login to EVL4 was OK

yyyy-mm-dd hh:mm:ss DPM-001I Dual Path Monitoring activated ...
yyyy-mm-dd hh:mm:ss DPM-015I Sending to cell phone: 19990001212 DPM-001I Dual Path
Monitoring activated ...
yyyy-mm-dd hh:mm:ss MDM-010I Modem replied "OK" to: AT+CMGS
yyyy-mm-dd hh:mm:ss MDM-011I Modem sent SMS: DPM-001I Dual Path Monitoring activated ...
to 19990001212

yyyy-mm-dd hh:mm:ss DPM-006I New, unprocessed message: SYSTEM READY, AC PRESENT, DISARMED

yyyy-mm-dd hh:mm:ss MDM-019I Modem replied "OK" to: AT+CMGL="REC UNREAD"
yyyy-mm-dd hh:mm:ss TEL-005I EVL4 responded "OK" to 'stay awake' request
yyyy-mm-dd hh:mm:ss TEL-006I EVL4 responded "OK" to poll request

yyyy-mm-dd hh:mm:ss DPM-042I DPM-042I syslog captured an event that TPI has yet to:
CID=1604010010
yyyy-mm-dd hh:mm:ss DPM-004I Decoded syslog CID is: Partition:01, New event:604 Fire test,
User:Me
yyyy-mm-dd hh:mm:ss DPM-013I Red Alert - found  'fire'  in message

yyyy-mm-dd hh:mm:ss DPM-015I sending to cell phone: 19990001212 Partition:01, New event:604
Fire test, User:Me
yyyy-mm-dd hh:mm:ss MDM-010I Modem replied "OK" to: AT+CMGS
yyyy-mm-dd hh:mm:ss MDM-011I Modem sent SMS: Partition:01, New event:604 Fire test, User:Me
to 19990001212


* actual phone number has been replaced with a fictitious one.
```

Appendix E: DPM.sh Housekeeping BASH Shell Script

```bash
#!/usr/bin/bash

# Copyright (c) N. A. Inc. 2022
# This program is free software per V3, or later version, of the GNU General Public License.
# It is distributed AS-IS, WITHOUT ANY WARRANTY, or implied warranty of MERCHANTABILITY
# or FITNESS FOR A PARTICULAR PURPOSE.

# Version 1.01
# DPM.sh is a housekeeping script for the Dual Path Monitoring application,
# (DPM.py) run on a Raspberry Pi model 2, 3, or 4, with Python V3.9 or later.
# A system reboot should be scheduled to run daily after midnight via  /etc/crontab
# DPM.sh should be run automatically, post re-boot, via  /etc/rc.local
# It assumes the syslog client of the EVL4 module is configured to write to a LOCAL
# facility (e.g. LOCAL0) and  /etc/rsyslog.conf  then directs it to the actual-path
# equivalent of $LOGDIR/EVL4.log  It also assumes that DPM.ini (config file) directs the
# runtime log for DPM.py to the actual-path equivalent of $LOGDIR/DPM-yyyymmddhhmmss.log

PATH="/home/pi1/DPM-EVL4:$PATH" ; export PATH
APPDIR="/home/pi1/DPM-EVL4"
LOGDIR="/home/pi1/DPM-EVL4/logs"

# FIRST, check Ras Pi's clock and do any optional modem and EVL4 reboots
now="$(date '+%Y%m%d%H%M%S')"
STARTUPLOG="$LOGDIR/DPM-HK-startup-$now.log"
cd $APPDIR ; python $APPDIR/startup.py $STARTUPLOG 1>>$STARTUPLOG 2>&1

# Ras Pi system clock should now be correct, so let's go!
tstamp="$(date '+%Y%m%d%H%M%S')"
HKLOG="$LOGDIR/DPM-HK-$tstamp.log"
start="$(date '+%Y-%m-%d %T')"
echo $start " Start of DPM Housekeeping" > $HKLOG

# rename the prior EVL4 syslog and offset files.
var1="$(stat -c %x $LOGDIR/EVL4.log)"
var2="${var1%%'.'*}"
var3="${var2//[!0-9]/}"
mv $LOGDIR/EVL4.log  $LOGDIR/EVL4-$var3.log 1>>$HKLOG 2>&1
mv $LOGDIR/EVL4.offset  $LOGDIR/EVL4-$var3.offset 1>>$HKLOG 2>&1

# compress aged (> 2 days old) files
find $LOGDIR/DPM-*.log -type f -mtime +2 -exec gzip {} \; 1>>$HKLOG 2>&1
find $LOGDIR/EVL4-*.log -type f -mtime +2 -exec gzip {} \; 1>>$HKLOG 2>&1
find $LOGDIR/EVL4-*.offset -type f -mtime +2 -exec gzip {} \; 1>>$HKLOG 2>&1

# delete aged (> 5 days old), compressed files
find $LOGDIR/DPM-*.log.gz -type f -mtime +5 -exec rm -f {} \; 1>>$HKLOG 2>&1
find $LOGDIR/EVL4-*.log.gz -type f -mtime +5 -exec rm -f {} \; 1>>$HKLOG 2>&1
find $LOGDIR/EVL4-*.offset.gz -type f -mtime +5 -exec rm -f {} \; 1>>$HKLOG 2>&1

# create a new $LOGDIR/EVL4.log file with the requisite permissions
touch $LOGDIR/EVL4.log 1>>$HKLOG 2>&1
chmod 644 $LOGDIR/EVL4.log 1>>$HKLOG 2>&1
# restart rsyslog daemon to ensure it can write to the EVL4.log file
sudo systemctl restart rsyslog.service 1>>$HKLOG 2>&1

restart="$(date '+%Y-%m-%d %T')"
echo $restart " About to restart Dual Path Monitoring" >> $HKLOG
# invoke the Dual Path Monitoring python application
cd $APPDIR ;  nohup python $APPDIR/DPM.py 1>>$HKLOG 2>&1 &

end="$(date '+%Y-%m-%d %T')"
echo $end " End of DPM Housekeeping and restart" >> $HKLOG
```

Appendix F: Generalized Sample startup.py (DEBUG-level) Log File

```
Raspberry Pi booting with a working Internet connection:
yyyy-mm-dd hh:mm:ss CFG-001I Configuration object created (logging level: DEBUG)
yyyy-mm-dd hh:mm:ss PNG-001I PingOne object created
yyyy-mm-dd hh:mm:ss SUP-001I Internet connection OK; no Ras Pi clock adjustment will be made


Raspberry Pi booting with no Internet connection:
yyyy-mm-dd hh:mm:ss CFG-001I Configuration object created (logging level: DEBUG)
yyyy-mm-dd hh:mm:ss PNG-001I PingOne object created
yyyy-mm-dd hh:mm:ss SUP-002I NO Internet connection ...
yyyy-mm-dd hh:mm:ss MDM-001I MyModem object created
yyyy-mm-dd hh:mm:ss blah … blah … blah
yyyy-mm-dd hh:mm:ss SUP-003I Adjusted Ras Pi system date and time: ['sudo', 'date', '+"%y%m
%d %T"', '-s yymmdd hh:mm:ss']
```

Appendix G: Sending SMS Messages To DPM

The DPM application can process three types of *inbound* SMS messages, namely an Arm request (e.g. "Arm partition 1 away"), a system Status request ("Status?") and a Chime toggle ("Chime") request providing, in the DPM.ini configuration file, "ModemInboundSMS = True" and "InboundSMSCellPhones" is populated with at least one valid cell phone number. Since SMS traffic is not encrypted, a *dis*arm request is ***not*** supported.

For security reasons, an "arm only" user account should be created on the Honeywell Vista system for each user (i.e. cell phone). The following key sequences should be entered on a Honeywell keypad:

```
MasterCode + 8 + uu + ssss
        where uu is the new user's number and ssss is their security code

MasterCode + 8 + uu + # 1 + 1
        where uu is user's number and 1 means ARM only
```

To find the security code (ssss) for each cell phone number, run the DPM application in DEBUG mode, send a "Arm partition 1 away" request, via SMS (it will fail until the security system has been configured) to the wireless modem's phone number and then look in DPM's log file for:

```
TEL-014I Submitting Process Keystrokes request ^03,1$ssssnn
```

SECURITY CODES

Your installer assigned a master code that is used to perform all system functions. In addition, other security codes can be assigned for use by other users.
- Only the System Master and Partition Master can assign security codes to users.
- Users are identified by 2-digit user numbers.
- Only the System Master can change user partitions.
- In addition to a security code, each user is assigned an authority level and various system attributes. See the full User Guide for definitions of each Authority Level.
- Security codes can be used interchangeably within a partition when performing system functions (a system armed with one security code can be disarmed by a different security code), with the exception of the Guest and Arm Only codes.
- Security code programming involves these steps:
  1. Choose a user number from the set of users assigned to the partition in which the user will be operating, and assign a 4-digit security code.
  2. Assign an authority level to that user.
  3. Assign other attributes as necessary.

Assigning Security Codes and User Attributes

The following lists the various command strings for adding security codes and attributes.

NOTE: Partition Master codes (VISTA-20P Series only) apply only to those user numbers previously assigned (by the system master/installer) to the partition master's partition.

Function	Commands
Change System Master Code	System Master code + [8] + [02] + new code + new code again
Add Security Code	Master code + [8] + user no. + new user's security code The Keypad beeps once to confirm that new user was added.
Delete Security Code	Master code + [8] + [user no.] + [#] [0] The security code and all attributes* set for this user number, including any associated RF keys, are erased from the system. (*except assigned partition)
Authority Level	Master code + [8] + [user no.] + [#] [1] + auth. level Authority Levels: 0 = standard user 1 = arm only 2 = guest 3 = duress 4 = partition master (VISTA-20P Series only)
Access Group	Master Code + [8] + [user no.] + [8] [2] + group (1-8) You can assign users to a group, then set an access schedule that defines the times this group of users can operate the system. The system ignores these users outside the scheduled times.
User's Partition	System Master Code + [8] + [user no.] + [#] [3] + [0] + part. + [#] This assigns the partitions the user can access. If more than one, enter partition numbers sequentially, then press [#] to end. E.g., master code + [8] + [user no.] + [#] [3] + [0] + [1] [2] + [#] gives the user access to partitions 1 and 2 and common partition. Partition Entries: 1 = partition 1 and common 2 = partition 2 and common 3 = common partition only

Appendix H: Sample Usage of protectPassword.py On a Raspberry Pi SBC

```
python protectPassword.py

Enter your password (minimum of 8 characters): YOURPASS
Your stupefied password is:  'n637B\x08JX'   (including the encapsulating single quotation
marks!) ... now copy and paste it into its target destination.
As a final check, you entered:  YOURPASS  as your cleartext password
```

Appendix I: Document History

Version	Date	Author	Notes
1.0	2022-10-12	Not Applicable Inc. Copyright (c) 2022	This publication is free docuware per V3, or later version, of the GNU General Public License. It is distributed AS-IS, WITHOUT ANY WARRANTY, or implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
1.1	2022-10-16	Not Applicable Inc.	Fixed spelling error and standardized table borders.
1.2	2022-10-17	Not Applicable Inc.	Updated to reflect successful use of Python-3.9.15
1.3	2022-10-27	Not Applicable Inc.	Corrected spelling error
1.4	2022-11-04	Not Applicable Inc.	Added details for using SSH, post physical deployment
1.5	2022-11-15	Not Applicable Inc.	Typo correction and minor narrative change.
1.6	2025-03-13	Not Applicable Inc.	Added Chime toggle request via SMS