MIRA Mirror Heater Automation

# Abstract

The 36” mirror needs to be protected from condensation. We will monitor temperature, humidity and dewpoint temperature in order to alert the telescope operator and automatically turn on a mirror heater.

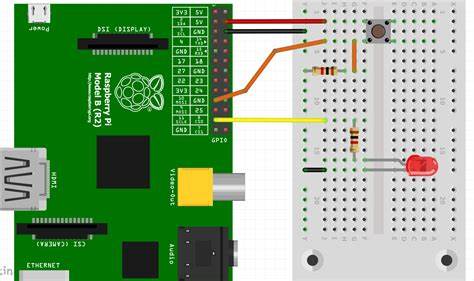
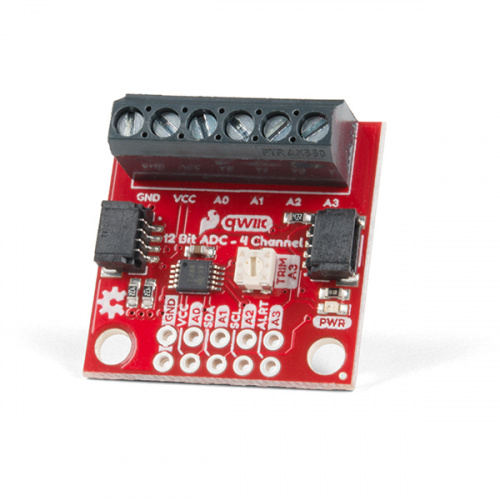
# Document History

|  |  |  |
| --- | --- | --- |
| Date | Who | What |
| 10 Aug 2020 | Glenn Nelson | Created |
| 2 Sep 2020 |  | Added install instructions |
| 8 June 2021 |  | Revise for clarity? |
|  |  |  |

# Requirements

* Environmental measurements every 5 minutes.
* Ambient dewpoint temperature measured or calculated inside the telescope “dome”.
* Dewpoint temperature near the mirror is measured.
* Mirror temperature is measured.
* Geist Watchdog 100 is used for some environmental monitoring.
* Software is configured from web page to display status: safe, warning, critical.
* Background color of web page indicates status.
* Mirror heater can be manually or automatically controlled.
* Web page updated at least every 5 minutes.

# Ideas and Questions

* There is a popular means of drawing circuits for Raspberry Pis, Arduinos and other such computers, called [Fritzing](https://fritzing.org/). The software is basically CAD for breadboards. Apparently, it can generate layouts for a circuit board that can be produced for $10 to $20 in singles. We might want to use this instead of wiring circuits ourselves. Here is an example of a Fritzing diagram with a pushbutton and LED. This can be used to generate a physical circuit board.  
  
* Raspberry Pi does not have A/D inputs. If analog sensors are actually required (there may be a digital substitute), then we can get an external A/D board or could use Arduinos for the analog devices. Here is one [A/D board from Sparkfun](https://learn.sparkfun.com/tutorials/qwiic-12-bit-adc-hookup-guide). It is 12 bit with 4 input channels. 

# Design and Implementation

Sensor data will be collected at least every five minutes and used to alert the operator of current conditions and turn on the mirror heater when appropriate. Sensor status will be available in a web page; the status will also be logged to a file whenever sensor readings are made. The operator will be able to override mirror heater automatic settings as needed.

There is a Geist environmental monitor (Geist Watchdog 100) that has temperature, humidity and dewpoint temperature sensors both in the device and as external sensors. Geist data can be retrieved by a single JSON request or by SNMP requests (not easy).

Other temperature, humidity and dew sensors will be connected to a computer. Although Arduinos have been used at MIRA, the requirement that Geist be employed, that other sensors will be used, and that a web interface is available argue for far more capability. Therefore a Raspberry Pi running Linux is chosen.

The Raspberry Pi controller is henceforth referred to as **Mira-Pi1**.

## Raspberry Pi Details

Login info: username:pi, password:starbright-10

The pi user has sudo permission.

You might want to generate a RSA key-pair for login; then you can ignore password login. Either Putty on Windows or other programs on Linux and Mac can generate these keys.

Connect to ethernet and login with SSH. Or connect monitor to HDMI, keyboard and mouse to USB, then modify WiFi setup for local WiFi network, by running **sudo raspi-config**.

This is a Raspberry Pi 4, functionally no different than a Pi 3, it’s just more powerful.

Popular text editors are *vi* and *nano*.

See section 6 for information about the Python software.

File list with **ls** command is color-coded; depending on your terminal window you may have trouble reading some filenames. If so, use command **lnc** (list with no color).

## Mirror Heater

* AC relay switch is used to run mirror heaters. Relay is controlled by low voltage DC.
* Relay is turned on automatically when certain environment conditions are met.
* Mira-Pi1 may send email with status updates (June 2021: not currently enabled)..
* The monitoring program on mira-pi1 is written in Python. It runs as a cron job every 5 minutes (or other interval if desired).
* The monitor program appends to a log file. See section 5.4.
* The status display will also confirm that the most recent log is within the expected time frame and display an alert if not (if sensors have stopped reporting).

### Heater Override

There are commands that can override the mirror heater. See **pi\_ints.py** for comments about this functionality.

Login to the Pi. Go to home directory, just type **cd**.

There are 3 shell scripts. Execute one of these scripts when needed. (NOTE: If you cannot read what **ls** command shows you, then use **lnc** command).

The scripts are:

* **console\_relay\_off.sh:** override status and turn off for 15 minutes
* **console\_relay\_on.sh:** override status and turn on for 30 minutes
* **console\_relay\_resume.sh:** remove manual overrides, revert to automatic status control

### Environmental Conditions

The web page is yellow when mirror temp is less than 5°C from dewpoint or ambient humidity > 60%.

The web page is flashing red when mirror temp is less than 2°C from dewpoint or humidity > 80%.

The web page has a green background if not red or yellow.

### Mirror Heater Controls

The mirror heater is turned on if condition red. It may be automatically turned off after 30 minutes.

The heater is switched by an AC control relay from Digital Loggers (<http://iotrelay.com>). This is an AC power strip that can be controlled by GPIO signal from a Raspberry Pi.

### Manual Override

Two pushbuttons are hooked up to enable the operator to override automatic response.

One pushbutton turns the heater on and keeps it on for a period of 30 minutes (configurable time).

One pushbutton turns the heater off and keeps it off for a period of 30 minutes (configurable time).

The pushbuttons are monitored by a continually running Python program that responds to interrupts. This program is separate from the programs that read and log the sensors.

## Web Page

* Web page displays status of the mirror heater: on/off. It also shows how much time remains if it is turned on. It also shows whether the heater was manually or automatically turned on.
* Cron job is run every 5 minutes to collect environment data from Geist and other sensors.
* Raspberry Pi also runs a continual program to service interrupts from the mirror heater pushbutton overrides.
* Mira-pi1 also hosts an Apache web server for displaying the environment and controlling the mirror heater.
* The web server on mira-pi1 presents environmental conditions measured by Geist and by other sensors. The page is written with PHP so that it can parse the environment logs and change the display status (color and other information).
* There is also a web page for setting conditions that trigger alerts. For example when dewpoint temperature approaches the mirror temperature.

## Sensors

Geist WD100 (Geist Watchdog 100) has internal and external sensors for temperature, dewpoint temperature and relative humidity. These values can be read with a single HTTP request that returns JSON; see Appendix for sample.

Mira-pi1 has additional sensors connected to it. These supplement the WD100.

Mira-pi1 runs the Python program that queries the Geist WD100 and writes environmental data logs.

The data is logged every 5 minutes. JSON from Geist is parsed with a Python program and a summary is written to a log file. The log file is used by other applications to notify the telescope operator and to operate the mirror heater. See later sections for more information.

|  |  |  |  |
| --- | --- | --- | --- |
| Platform | Sensor | Name | Location |
| Geist WD100 | Temp | G100/T | Out there somewhere |
|  | Humidity | G100/H |  |
|  | Dewpoint | G100/D |  |
| Geist GTHD | Temp | GTHD/T | Out there somewhere |
|  | Humidity | GTHD/H |  |
|  | Dewpoint | GTHD/D |  |
| Raspberry Pi | DHT22 Temp | PI/DHT/T | Mirror cell |
|  | DHT22 Humidity | PI/DHT/H | Mirror cell |
|  | DS18B20 Temp | PI/DS18/T | Mirror |
|  |  |  |  |

The mirror status is determined by an algorithm that uses dewpoint to determine when the mirror heater needs to be turned on. The heater should be turned on when the mirror surface temperature approaches dewpoint from above (if mirror temperature is less than dewpoint, condensation has already occurred). Therefore:

If mirror\_temp is close to dewpoint\_temp near mirror, then turn heater on.

How should the sensors be used to determine when the heater is turned on and off? What do we do if some sensors are offline? Ideally we measure or calculate the dewpoint temperature near the mirror and compare to the mirror temperature. However, what if some sensors are offline? The following table helps design the algorithm.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sensor | Online | | | | | |
| G100/T | ✓ | ✓ | ✓ | ✓ | ✓ |  |
| G100/H | ✓ | ✓ | ✓ | ✓ | ✓ |  |
| G100/D | ✓ | ✓ | ✓ | ✓ | ✓ |  |
| GTHD/T | ✓ | ✓ | ✓ | ✓ |  |  |
| GTHD/H | ✓ | ✓ | ✓ | ✓ |  |  |
| GTHD/D | ✓ | ✓ | ✓ | ✓ |  |  |
| PI/DHT/T | ✓ | ✓ |  |  | ✓ | ✓ |
| PI/DHT/H | ✓ | ✓ |  |  | ✓ | ✓ |
| PI/DS18/T | ✓ |  | ✓ |  | ✓ | ✓ |
| *Algorithm Uses* | All 3 PI sensors | Unknown | GTHD and PI/DS18/T | G100 and GTHD | All 3 Pi sensors | All 3 Pi sensors |
| Extra Info |  |  |  |  |  |  |

### Sensor Summary

Pi sensors are all that are required for controlling the mirror heater.

Pi sensors are on long cables and it is sometimes possible that we get bad or no data from them.

If one or all of the Pi sensors are offline, then we substitute the Geist sensors.

If the G100 (Geist chassis) is offline, then the GTHD will be offline.

## Log File of Environment

Log files of the Geist sensors and mira-pi1 sensors keep a record of all readings (every 5 minutes by default). These are rotating log files and the oldest is only a few days old. Therefore if a permanent record is desired, another process should archive these logs.

A line in the log file looks like this:

INFO : 2020-07-22 15:45:03,I:Geist WD100,V:humidity,59,V:dewpoint,55.03,V:temperature,69.91,I:GTHD,V:humidity,55,V:dewpoint,55.49,V:temperature,72.57

The log file is written with Python logging library, so it always starts with the “level”, in this case it is “INFO:”. Next is the date-time from the Geist WD100. After this, letter “I” for instrument, followed by a sequence of values that are designated with letter “V”. In this sample line there are two instruments, the “Geist WD100” rack unit and the “GTHD” external sensors. The values are humidity, dewpoint and temperature. The humidity is relative. The temperature units are Fahrenheit.

There will be additional sensors on mira-pi1 and those values are written to a different log file.

## Hardware and Wiring

A Raspberry Pi has many GPIO (general purpose I/O) pins for digital I/O. It also hosts SPI and I2C interfaces. There is no A/D input, but multi-channel A/D converter boards are readily available to connect to GPIO; see section 4. In addition, there are 4 USB ports, one or two HDMI (video) outputs, wired Ethernet and WiFi.

### Sensors

AM2302 (or DHT22) is a temperature and humidity sensor. It has temperature accuracy of ±0.5 °C and humidity accuracy of ±2.5% over 0% to 100%. It needs a pullup resistor on the data pin: 4.7K to 10K. Each AM2302 needs a dedicated GPIO pin – unfortunate. You must install Python package *Adafruit\_DHT* to use this sensor (note: already installed)

LM34 is a temperature sensor. It is an analog device with a linear calibrated response that requires A/D converter to read it.

DS18B20 is a digital temperature sensor. It has accuracy of ±0.5 °C over -10°C to +85°C. It is connected with the 1-wire device protocol. It needs a pullup resistor on the data pin: 4.7K to 10K. Use GPIO4 for the 1-wire interface. Any number of DS18B20 can be connected in parallel, but of course you only use one pullup on the data line. Each DS18B20 has a unique address and you need to catalog these addresses so that you know which one you are currently reading. You can purchase a string of 6 sensors spaced one foot apart that are wired in parallel; it is approximately $80.

#### Enabling 1-Wire Interface

The 1-wire interface is used for DS18B20 temp sensors. It needs to be enabled. I have already done this, but here is the procedure.

See this page for a good tutorial: <https://learn.adafruit.com/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing/ds18b20>. Or just follow these instructions:

You need to enable 1-wire with raspi-config:

sudo raspi-config

Select “interfacing options”, and then enable 1-wire.

sudo reboot

Now the 1-wire kernel modules should be enabled. Check that they are running:

lsmod | grep -I w1

You should see w1\_gpio, but may not see w1\_therm until a DS18B20 is connected to GPIO4.

### AC Relay Control

We will use an AC power control from Digital Loggers (iotrelay.com). It can be turned on by the 3.3V output of a GPIO pin.

There are two pushbuttons to override the AC relay automatic settings. One button turns it on and keeps it on for an interval that is configurable (defaults to 30 minutes). Another button turns it off for an interval that is configurable (defaults to 30 minutes).

### LEDs for Status

The status conditions are green, yellow and red. For convenience we will have three colored LEDs that can be lit by Mira-Pi1, but the software can be configured to disable these – important, since Mira-Pi1 will be located by the telescope.

### Hookup Details

I do not have the capability to generate wiring diagrams, so I’ll just describe it here. (Note: I might be able to generate both wiring diagram and a physical circuit board, see section 4).

DS18B20 temperature sensors data line is connected to GPIO4, the standard pin for 1-wire devices on Raspberry Pi. No pullup is required, but I use 4.7K to the data line. Plus is connected to 3.3V, negative to GND.

One DHT22 data line is connected to GPIO5. No pullup is required, but I use 4.7K to the data line. Plus is connected to 3.3V, negative to GND.

Status LEDs are green, yellow, red. 330Ω resistors are connected, respectively to GPIO 25, 24, 23. The LED plus is connected to the resistor, negative to GND.

The AC/DC relay is connected to GPIO17; no pullup is required.

The two pushbuttons are connected to GPIO 19 and 26; no pullup is required, but I use 10K.

## Pi Software Setup

The software is written in Python. First create a Python3 virtual environment in home dir:

cd ~

python3 -m venv mirror-py3

To proceed with dev and test, you must activate this environment:

source ~/mirror-py3/bin/activate

Now some python packages may need to be installed. (Note: I’ve probably already done this).

Drivers for DHT22 are supplied by Adafruit. Hmm, that’s disturbing, but fortunately nothing special is required to install.

sudo pip3 install Adafruit\_DHT

pip install Rpi.GPIO

Python programs are found in /home/pi/Projects/geist\_wd100. This software is available from GitHub: <https://github.com/dr-glenn/geist_wd100>

The web page is *mirror.php*. It is also stored in Projects/geist\_wd100, but the live version is /var/www/html/mirror.php.

## Configuration Settings File

Python programs will read some settings from pi\_settings.py if the file exists. These settings control the conditions for mirror status: green, yellow, red. The settings also list emails for notification of status.

In the absence of the pi\_settings.py, here are the default configuration settings:

[This section not completed]

# Installation

Setting up at OOS.

Plug sensors into the Pi-HAT board as shown in picture.

Next connect the Pi to Ethernet. You will need to discover the network address. If you have ready access to the router, you will use it to find the new Raspberry PI, named **mira-pi1**. The Ethernet MAC address is dc:a6:32:8a:81:61.

If you do not have access to the router, connect a keyboard and mouse to USB and use the min-HDMI cable to connect to a monitor.

Plug the Pi into power. In less than a minute I will be up.

Find the IP address, either from the router or from the console.

Login credentials: username: **pi**, password: **starbright-10** (yes, a very bright star).

You should then set a static IP associated with the MAC address by logging into the router (this can be done later, but you need to know the IP address of the Pi).

With IP address you can always SSH to the Pi. There’s little need for an actual console.

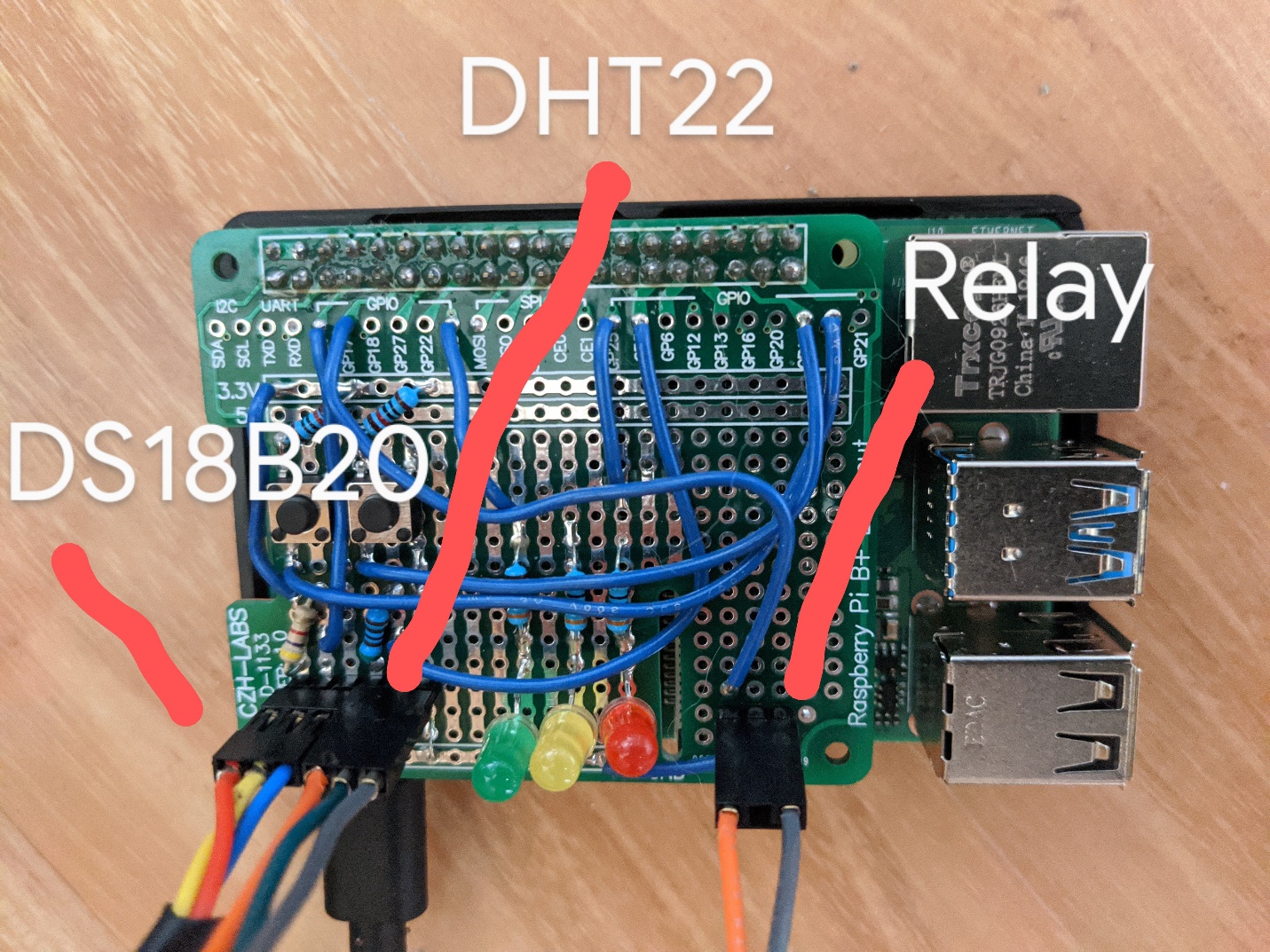
The mirror heater control software should all be running.

You need to change one value in pi\_sensors.py. Edit file and change FAKE\_STATUS=False.

Web page is <Pi address>/mirror.php

## Sensor Prototype Board

This board sits on top of the Raspberry Pi. There are 3 connectors: DS18B20 temperature sensor, DHT22 temperature and humidity, and relay controller. Pay attention to the orientation of the connectors, red or orange is ***plus***.



## Rules

The algorithms should be detailed elsewhere in this document, but I’ll rehash them here.

Status is red, yellow or green. Status is displayed on web page and by LEDs on the Pi. The yellow LED is not hooked up (will be in the future).

A cron job on the Pi runs the status checker every 5 minutes. It queries the Geist sensors and the Pi sensors.

Status Red: the heater turns on until status changes to yellow or green.

Status Yellow: a warning to the operator

Status Green: all’s well

Status Grey: some sensor is not functioning

The heater can be manually turned on and off by pushbuttons on the Pi HAT board. When turned on the red LED should light up.

When manually turned on it should stay on for 30 minutes, overriding the status as determined by the sensors. In other words, environmental changes cannot turn off the heater if it was manually turned on.

Another pushbutton on the Pi Hat can manually turn off the heater. This also overrides the environment status.

On the web page the status, “red/yellow/green” is calculated from the sensors/ However, the table displays the relay state correctly if you used manual override.

### Algorithm for Status

The Pi sensors return temperature in °C and humidity in percent. If a sensor is offline the code substitutes a value of -999.

The following sensors are available:

Pi DS18B20: this is a temperature-only sensor that is deployed to the mirror.

Pi DHT22: this is a temperature and humidity sensor that is deployed in the mirror cell.

Geist WD100: this has temperature, humidity and dewpoint instruments in the main chassis.

GTHD: this is an external Geist temperature, humidity and dewpoint sensor.

#### Algorithm updated 6 Feb 2021.

There are four possible conditions for the mirror. They are designated as grey, green, yellow or red. In the software these are assigned values of 0, 1, 2 or 3.

Grey status means that a sensor is offline or returned a “bad” value.

Green means that conditions are “safe”.

Yellow means that we should be observant of changes toward condensation.

Red means that condensation is possible and the mirror heater will be turned on.

If both Pi sensors are online, the mirror dewpoint is calculated using DS18B20 temperature and DHT22 humidity.

If only the Pi DHT22 sensor is online, the mirror cell dewpoint is calculated using DHT22 temperature and humidity.

The max(status) from mirror and mirror cell determines the overall status. For example, if values of 2 (yellow) and 3 (red) are calculated, red wins.

When status is able to be determined from Pi sensors, the Geist status values are ignored.

If status cannot be calculated from Pi sensors, status calculation is done using the Geist values. Once again the max(status) wins.

## Software Repository

The software is kept on GitHub at <https://github.com/dr-glenn/geist_wd100>.

Please be sure you understand Git and GitHub at least a little before you make changes to the software on the Pi. Please ask for help first!

To make changes, you will need to clone this repository. The best approach is that you have your own GitHub account and clone to your account. Then you login into the Pi and clone from your own clone. If you don’t do this, you won’t be able to maintain a proper history.

## Web Page

The single web page is **mirror.php** and it is also stored in the GitHub repository. On the Pi it is installed in /var/www/html. You will need sudo permission to edit the file.

## Software Maintenance

When you login with SSH, you land in directory /home/pi/Projects/geist\_wd100. Almost all code is here. You might want to edit *pi\_sensors.py*, *pi\_hw.py* or *geist\_pi.py*. Preferred editor is **vi** or **vim**.

In *pi\_hw.py* look for FAKE\_STATUS. Change to **False** when you are ready to make the controller functional. When FAKE\_STATUS=True, it will change the status every 5 minutes and ignore the sensor readings.

The heater algorithm is named **calc\_status** and is found in *pi\_sensors.py*.

# Appendix

## Notes from Jeff Larson

the transistors are temp sensors

lm34 thees are the ones Gary has been using

we want to use this type to mount direct to mirror

pdf attached easy to setup on analog input

the little white things with 2 pins are dew detectors

they get conductive when wet we are going to put 1 on the fork ,,,the coldest part of the telescope

web page background color

flash red fast if

dew sensor on fork wet or

humidity 80% or

mirror temp within 2-3 deg C of dewpoint

web page background color

slow flashing yellow if

mirror temp within 5 deg C of dewpoint or

60% humidity

web page background color

solid green if

none of the above conditions met

we will have 3 dewpoint "temp + humidity"

1 is on Geist rack mount not in calcs yet

1 on telescope Geist remote not in calcs yet

1 the small white one DHT22 in mirror cell using rasberry

to calc dewpoint in mirror cell

show on web page humidity + temp + dewpoint

and mirror temp

Bruce spent a pile of $$ on the Geist so he wants the data

from the 2 sensors remote and built in to show data

on webpage humidity + temp + dewpoint for both

those sensors on Geist

they are not in calcs yet but could play a part in decision making later on we have to watch data for awile

b4 we include or not to calcs to shut down and turn on heaters

if at anytime any of the flashing red conditions are met

turn on heaters

the other things are thermocouplers

need op amp to make work

too much monkeying around

## Revised algorithm, 28 January 2021

In the past week OOS experienced very cold temps and high humidity. The mirror heater stayed on for hours at a time which worried Bruce and Jeff. Therefore Bruce suggests the following change, excerpt from his email.

We've learned two things: the sensors can sometimes go off the reservation and, if we're too cautious, we can have the heater on for too long.  On the first point, it appears that when the sensor goes haywire, it goes really crazy.

I treat this as a state...that is, every cycle, the state the heater should be in (on or off) is evaluated w/o regard to the past .

part 1

RELAYTEST= OFF

IF(ds 18b20-0 Temp is >-18 and < 50 ) and (dht22 humidity is >0 and <150) {presumed good}

   calculate dewpoint from these value and  (ds18 Temp - 2°) ≤ dewpoint

OR

IF (dht22  Temp is >-18 and < 50 ) and (dht22 humidity is >0 and <150) {presumed good}

   calculate dewpoint from these value and  (dht22 Temp - 2°) ≤ dewpoint

THEN

   RELAYTEST= ON

ENDIF                   end part one

part2

IF NOT(dht22 humidity is >0 and <150)

     OR

        IF NOT (ds 18b20-0 Temp is >-18 and < 50 ) AND NOT (ds 18b20-0 Temp is >-18 and < 50 )

THEN

     RELAYTEST= OFF {inside numbers are bad, must use outside}

     IF ((Geist Temp- 2°) ≤ Geist dewpoint) OR (GTHD Temp- 2°) ≤ GTHD dewpoint)

     THEN

         RELAYTEST= ON

     ENDIF

ENDIF                             end part 2

END of calculations

set relay to RELAYTEST

CYCLE

END

Notes: part one is the expected case and if is used, then part 2 logic will not be used since the sensors are good.  If the inside humidity or both temp sensors are dead, then we'll go with the outside tests.  I don't see that cycling them does much good as that requires knowing how much heating/cooling is going on INSIDE the mirror cell which depends on things like delta T and other things we know little about.  Maybe we'll get sexy later.

Please go through the logic & make sure I've got the it all correct.  With a few time constants, presumably part 1 this will keep the mirror and mirror cell temperature at 2° above dewpoint or better. No turnoff delta Temp is needed.