Pi Zero W Home Security System

Concept

This describes building a multiple-camera security system with infra-red (IR) motion detection, temperature/humidity monitoring, and water warnings to monitor heating/cooling system failure or flooding due to pipe breakage or weather. It uses three Raspberry Pi Model Zero Ws but can be expanded if additional cameras are desired. Two Pis have remotely viewable cameras to monitor the doors of the house. One has IR-based motion detection and e-mails notification of motion to multiple addresses. The Rpi_Cam_Web_Interface project provides live camera feeds on the internet, protected by passwords, and enables remote control of the motion detection software. Rapid light changes due to clouds ruled out using just the camera motion detection to send alarms, so a strategically placed motion detection unit (PIR) was used for motion detection. A temperature/humidity sensor (DHT22) monitors the temperature hourly and sends e-mail alarms if it is too hot or too cold with user selectable alarm temperatures. A separate Zero W with a dampness sensor in the basement will just send an e-mail if liquid is detected and upon startup. It is possible to add additional cameras if desired, just increase the quantity of camera kits, SD cards and power supplies as needed. Finally, Dataplicity is used to allow remote maintenance of the system. In summary:

- Camera 1 will have a camera, a PIR motion detector and a temperature/humidity sensor. It will also be the main access from outside through Dataplicity.
- Camera 2 will just have a camera.
- The water sensor system will just have the water sensor, probably in the basement or a low spot, but also runs software to keep the IP address up to date through FreeDNS.

Parts

- 2 - Raspberry Pi Zero W camera kits (Adafruit 3414 or 3515 for IR version of camera)
- 1 - Raspberry Pi Zero W (Adafruit 3400)
- 1 - Raspberry Pi Foundation Zero W case (Adafruit 3446 or another if you prefer)
- 1 – PIR sensor (Adafruit 189)
- 1 – Phantom Yoyo high sensitivity water sensor (Amazon)
- 3 - 16 GB μSD cards, class 10 or better, can be larger if desired (Amazon)
- 3 - microUSB 2.4 A power supplies (Amazon)
- 2 - 2x20 headers male-male (Adafruit 2882)
- 1 - 2x20 long header (Adafruit)
- 1 - green LED
- 1 - red LED
- 2 – 10k resistor
- 3 – 100 ohm resistor
- 10 foot length of 3 or 4 wire cable (sprinkler cable works nicely from you local hardware store)

Special things to do to prepare

- Create a special gmail account (this helps avoid spam) such as you.location@gmail.com and have a unique the password.

- Know the ssid and password for the remote router as well as the router IP address and ithe administrator username and password.

Headless software setup

To setup the wireless, edit wpa_supplicant.conf with WordPad (PC) and save the resulting file as a txt (PC). Then remove the .txt extension by renaming the file. You can have two wifi entries, one for the remote system and one for your home system to enable debugging.

Add sections like these for each network and keep the quotes.

```
network={
    ssid="your ssid"
    psk="your password"
    key_mgmt=WPA-PSK
    scan_ssid=1
}
```

save and exit. Alternatively, you can copy a working wpa_supplicant.conf and edit it to add the remote network information.

Next you need to create an empty file named ssh. Using WordPad, create a file with a single space, and save it as ssh in txt format. Then delete the .txt extension as you did for the config file above.

Start by loading the latest distribution of Stretch Lite on all 3 μSD cards with Win32DiskImager or similar software. You should carefully mark the cards. Also copy the files ssh and wpa_supplicant.conf to each /boot partition.

Note that you can do all of the following headless, you don't even need to know the IP address. I use BitVise instead of Putty, but in either case use
host: raspberrypi.local,
user: pi,
password: raspberry to log in for the very first time on each computer. Later you can use the new hostname.local and password in the same manner.

Initialize the system as usual for local preferences with raspi-config and be sure to enable the Camera interfaces on the two camera SD cards. Pick unique hostnames and you can use the same password for all three systems, but choose good passwords, preferably 12 characters long. For instance:
Hostnames: Camera1, Camera2, WetSensor
Password: make a good one, Pa$$w0rd is not a good one, nor is $3cr3T.
Also set the appropriate localization options such as language and time zone for the remote installation.

Reboot upon exiting the raspi-config and log back in with the new hostname and password. Next do a update/upgrade to bring each system up to date.

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

Install additional software

First, install some needed applications on all three μSD cards. In reality, not all this software is needed on each system, but this method is a bit simpler.

```
$ sudo apt-get install git ssmtp heirloom-mailx python3-gpiozero wiringpi
```

Set up mail notification on Cam1 and Water Sensor Pis

```
$ sudo nano /etc/ssmtp/ssmtp.conf
```

and add these lines

```
hostname=Camera1 or WetSensor
root=youremail@gmail.com
mailhub=smtp.gmail.com:587
FromLineOverride=YES
AuthUser=youremail (leave off @ and thereafter)
AuthPass=gmail password
AuthMethod=LOGIN
UseSTARTTLS=YES
UseTLS=YES
```

We also need to modify the revaliases file to prevent problems.

```
$ sudo nano /etc/ssmtp/revaliases
```

and add the following line at the end of the file.

```
root:youremail@gmail.com:smtp.gmail.com:587
```

Install Rpi_Cam_Web_Interface on the camera cards

Only the camera μSD cards need this software installation:

```
$ git clone https://github.com/silvanmelchior/RPi_Cam_Web_Interface.git
$ cd RPi_Cam_Web_Interface
$ ./install.sh
```

In the installation dialog box, set the below answers, using arrow keys, not tabs, to move between entries:

```
folder: cam1 or cam2 (capitalization is important here, these will be directory
names and can be different from the host names)
port: 9880 or 9980 (or pick your own favorite numbers)
user: pi
password: whatever you chose, stronger is better, I used the login password
apache/nginx: apache
autostart: yes
jpglink: yes
php 5/7: 7
```

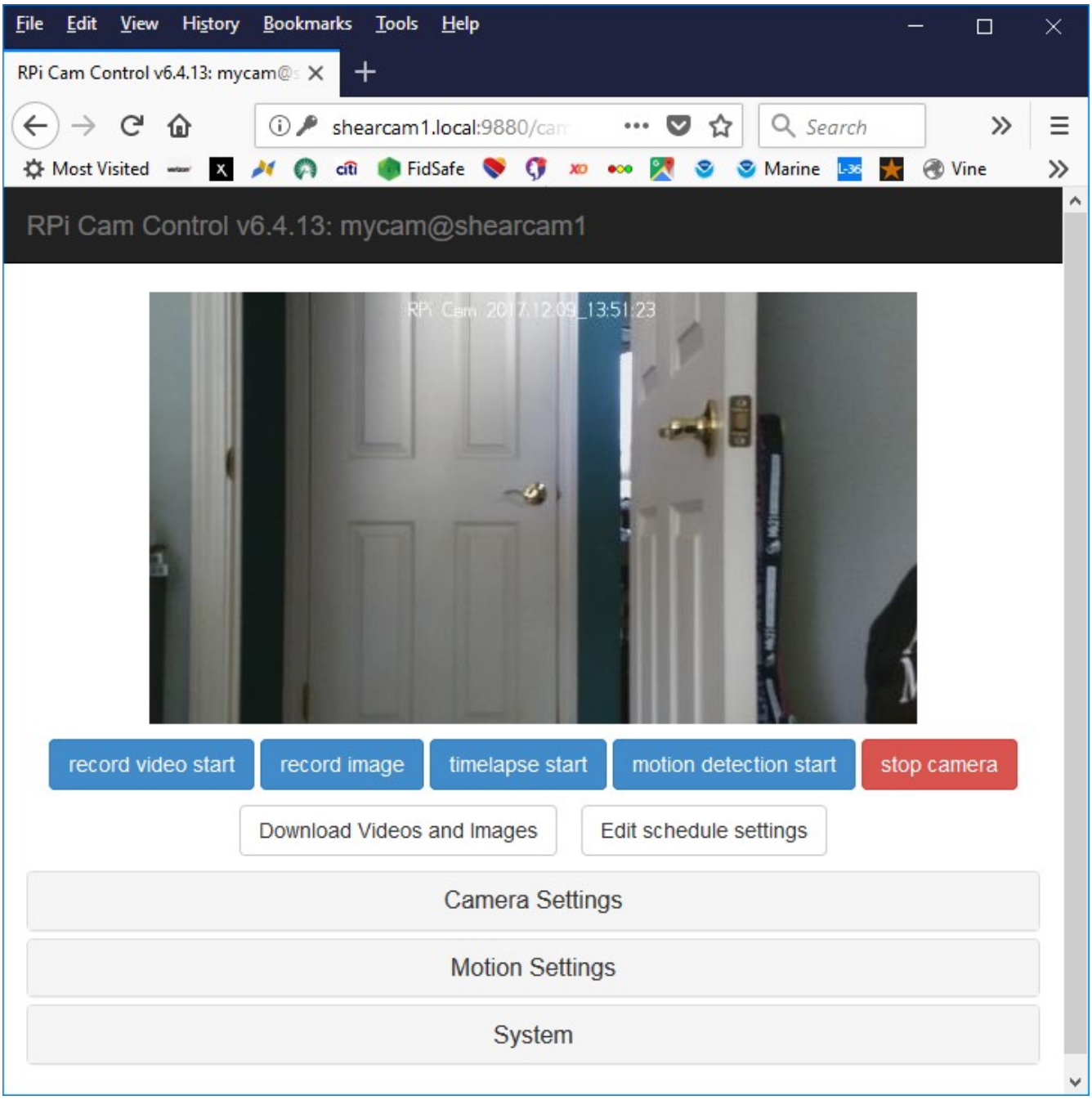
Now the system will download a bunch of software and set up the web cameras which takes a while, so be patient. If any software fails to install, you can

```
$ ./update.sh
```

which will reload things. When the download and install are done, the software will offer to start the camera. Accept this and check things from another computer's browser – these must agree with the port and folder names chosen above. Typical links are therefore:
camera1.local:9880/cam1

//camera2.local:9980/cam2

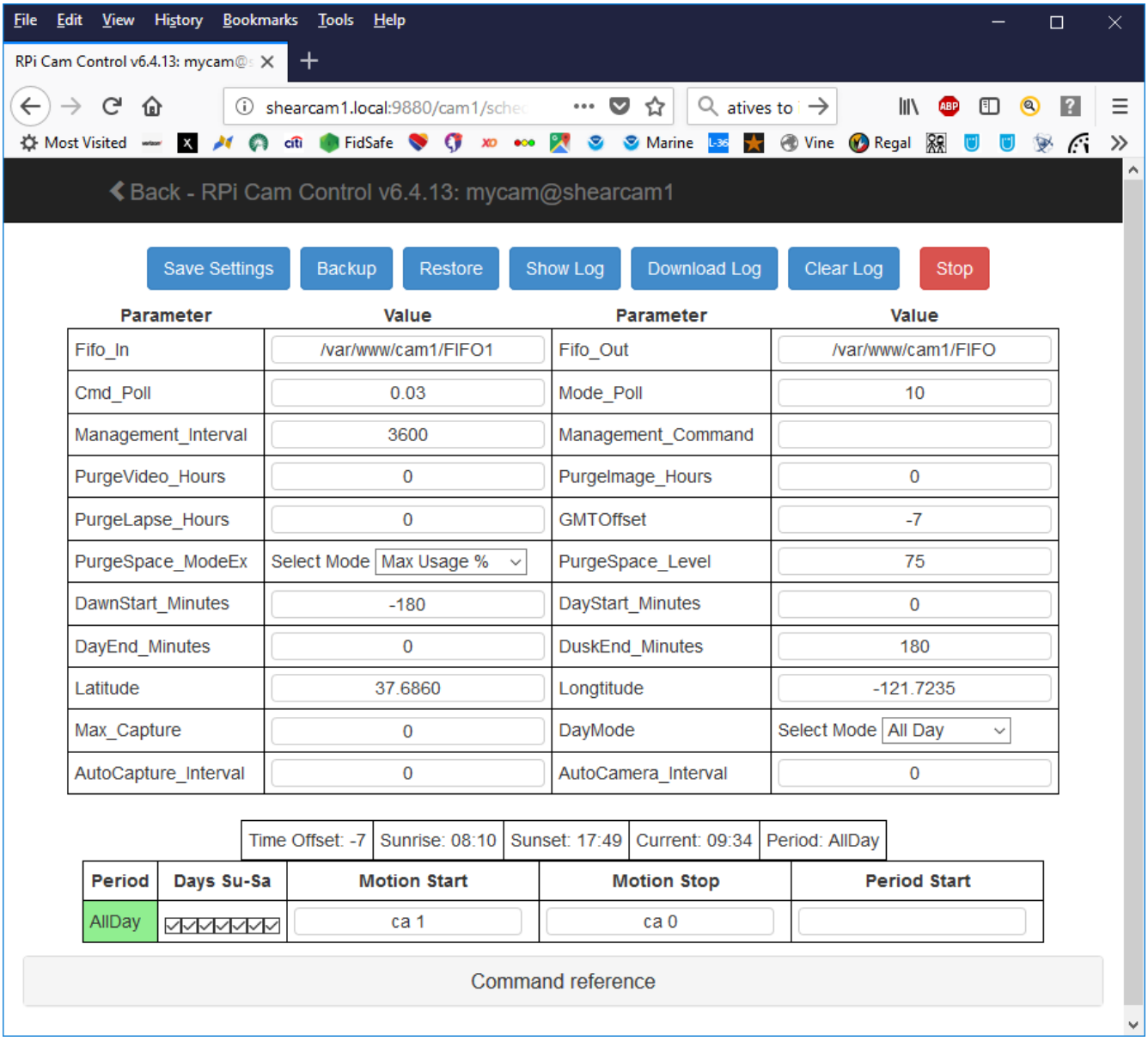
You will be asked for the username (pi) and password to log in, then the main screen will appear.



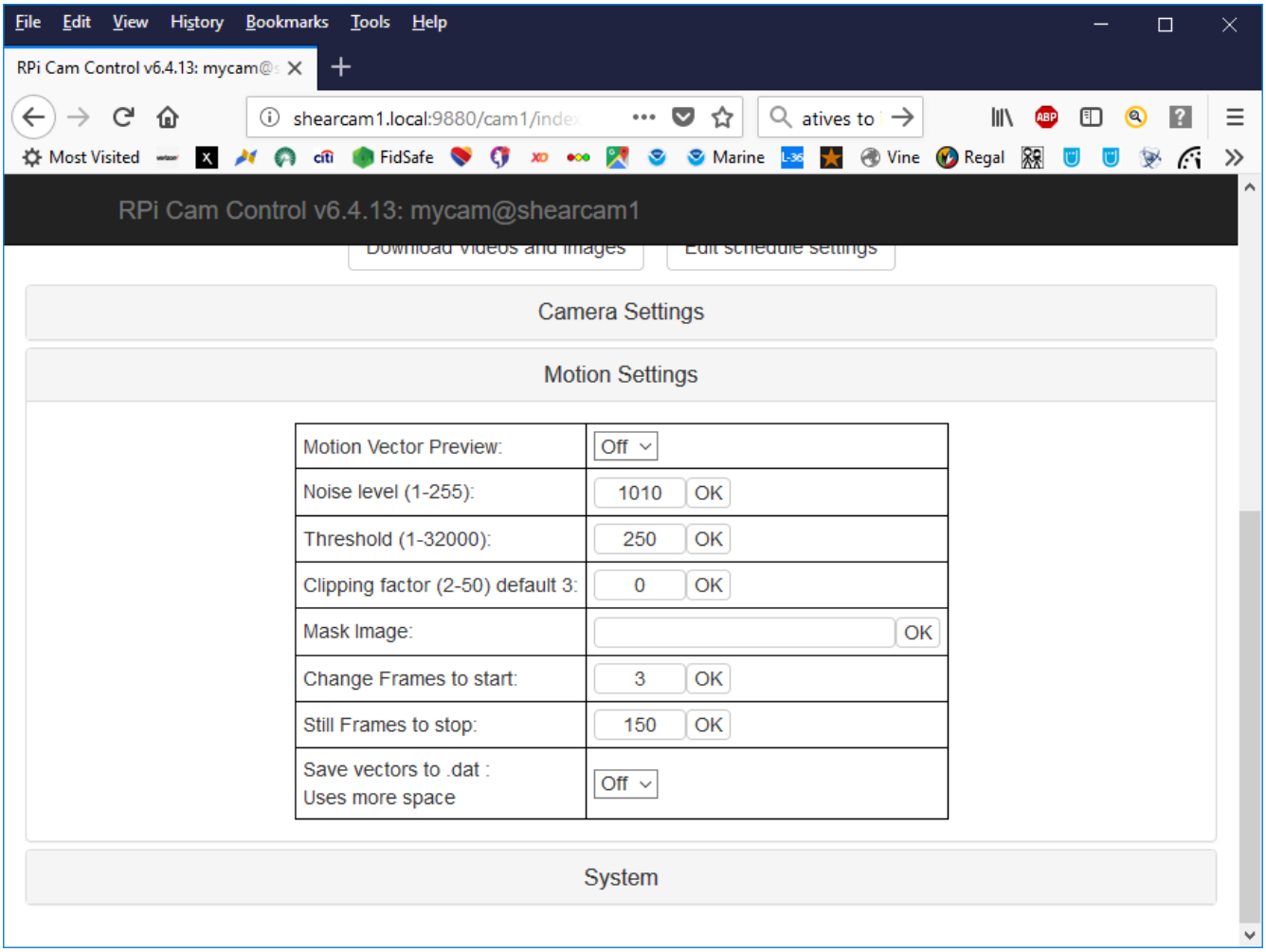
Configuring the camera system

For each camera system, while in the app from your browser, click Edit Schedule Settings Then in Purge Space Mode select Max Usage %, then choose 75% in the adjacent box. Look up your remote location Latitude and Longitude in decimal format on the web and enter these in the appropriate boxes. Note that West Longitude should be negative as should South Latitude. Also figure out your GMT offset and enter that in the appropriate box. If the sunrise/sunset settings appear correct after you save

the settings, you have it right. Western hemisphere will be negative GMT offsets. Click on Save Settings when you are finished getting all these set up.



Next click on the Camera Settings and be sure motion detect mode is Internal near the bottom of all the settings. In the Motion Settings menu, be sure the noise level is > 1000 to get into the proper algorithm (1010, the default, is good choice).



Finally, to get things started at boot time:

```
$ sudo nano /etc/raspimjpeg
```

find the autostart section and set motion_detection to true (the default is false). Save and exit and you are all set.

Motion Detection Code

For Camera 1, we also need to set up the PIR sensor and the temperature/humidity sensor. The PIR sensor uses gpiozero software to run a very simple code. This python3 program has a list of email addresses that can be as long as you like. You should also set the location name to something unique that will be clear which house the alarms are coming from.

```
#! /usr/bin/python3
# uses PIR sensor and gpiozero library to detect motion
# recommend setting PIR time constant to long to minimize multiple emails

from gpiozero import MotionSensor
import os
import time

# initialization

pirpin = 23
```

```
location = "Location Name"
email = ["youremail@gmail.com", "anotheremail@gmail.com"]
mailpart = " | heritage-mailx -s 'Location motion detected' "
mailstart = " | heritage-mailx -s 'Location motion detector started' "
pir = MotionSensor(pirpin)

# function definitions

def MOT():
    for addr in email:
        outs = "echo " + location + " motion detected at " + time.ctime() +
mailpart + addr
        os.system(outs)
        print(outs)

def NOMOT():
    print("No Motion Detected")

# let them know it is started

for addr in email:
    outs = "echo " + location + " motion detection started at " + time.ctime() +
mailstart + addr
    os.system(outs)
    print(outs)

# loop looking for motion

while True:
    pir.when_motion = lambda: MOT()
    pir.when_no_motion = lambda: NOMOT()
```

To start this program at boot time:

```
$ sudo nano rc.local
```

add this line to the end before the exit line

```
python3 /home/pi/PIR.py > /home/pi/PIR.log
```

Save and exit.

Temperature Sensor Code

Camera 1 also has the temperature/humidity sensor installed, so we need the appropriate software. Here, the language is C for fast access. The dht22.c code needs customization for your location and list of email addresses, then must be compiled. This code is executed hourly by cron and writes a log file that is emailed at the start of each month. The first line of the file shows the compilation line.

```
// gcc -Wall -std=gnu99 -o dht22 dht22.c -l wiringPi

#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <time.h>
#define MAX_TIME 85
#define DHT22PIN 7
```

```
#define ATTEMPTS 5
int dht22_val[5]={0,0,0,0,0};

int dht22_read_val()
{
    float humid=0.0;
    float tempC=0.0;
    float tempF=0.0;
    uint8_t lststate=HIGH;
    uint8_t counter=0;
    uint8_t j=0,i;
    int addr,nemail;
    char out_string[256];
    time_t current_time;
    current_time=time(NULL);
    FILE *fp;

// customize the next four lines for your needs
    float high_temp=90.0;
    float low_temp=50.0;
    const char *email[2]={"youremail1@aol.com","anotheremail@gmail.com"};
    const char *loc="Wherever";

    nemail = sizeof(email)/sizeof(email[0]);

    for(i=0;i<5;i++)
         dht22_val[i]=0;

    pinMode(DHT22PIN,OUTPUT);
    digitalWrite(DHT22PIN,LOW);
    delay(18);
    digitalWrite(DHT22PIN,HIGH);
    delayMicroseconds(40);
    pinMode(DHT22PIN,INPUT);

    for(i=0;i<MAX_TIME;i++)
    {
        counter=0;
        while(digitalRead(DHT22PIN)==lststate){
            counter++;
            delayMicroseconds(1);
            if(counter==255)
                break;
        }
        lststate=digitalRead(DHT22PIN);
        if(counter==255)
             break;

        // top 3 transistions are ignored

        if((i>=4)&&(i%2==0)){
            dht22_val[j/8]<<=1;
            if(counter>16)
                dht22_val[j/8]|=1;
            j++;
        }
    }

    // verify checksum and print the verified data
```

```
if((j>=40)&&(dht22_val[4]==((dht22_val[0]+dht22_val[1]+dht22_val[2]+dht22_val[3])&
0xFF)))
    {
        fp = fopen("/home/pi/temp.log", "a");
        humid=(256.*dht22_val[0]+dht22_val[1])/10.;
        tempC=(256.*dht22_val[2]+dht22_val[3])/10.;
        tempF=(9.*tempC/5.)+32.;
        fprintf(fp,"%5.1f\t%5.1f\t%s",tempF,humid,ctime(&current_time));
        fclose(fp);

//    send e-mail if over temperature, these are picked based on free air
temperature
//    and observation, so change the limits at your own risk

        if(tempF > high_temp) {
            for (addr=0;addr<nemail;addr++) {
                sprintf(out_string,
                "echo '%s high temp alarm at %s' | "
                "heirloom-mailx -s '%s high temperature alarm' %s",
                loc, ctime(&current_time),loc,email[addr]);
                system(out_string);
            }
        } else if (tempF < low_temp) {
            for (addr=0;addr<nemail;addr++) {
                sprintf(out_string,
                "echo '%s low temp alarm at %s' | "
                "heirloom-mailx -s ' %s low temperature alarm' %s",
                loc,ctime(&current_time),loc,email[addr]);
                system(out_string);
            }
        }
        return 1;
    }
    else
        return 0;
}

int main(void)
{
    int attempts=ATTEMPTS;
    if(wiringPiSetup()==-1)
        exit(1);
    while(attempts)
    {
        int success = dht22_read_val();
        if (success) {
            break;
        }
        attempts--;
        delay(500);
    }
    return 0;
}
```

We want to rename the temp.log file monthly and email it to someone, then start a new file. The tempupdate.sh code does this. Personalize the bolded lines with your email addresses.

```
#!/bin/bash
STR=$(date -d "last month" +"%y-%m")
mv /home/pi/temp.log /home/pi/$STR-temp.log
heirloom-mailx -a /home/pi/$STR-temp.log -s "monthly temperature log" your-email@gmail.com
heirloom-mailx -a /home/pi/$STR-temp.log -s "monthly temperature log" another-email@gmail.com
```

The temperature code will run hourly at 5 minutes after the hour, and will mail out a temperature log once a month using crontab.

```
$ sudo crontab -e
```

add the last two lines to the end of the file.

```
# m h  dom mon dow   command
5 * * * * /home/pi/dht22
1 0 1 * * /home/pi/tempupdate.sh
```

save and exit.

Water Sensor on Basement Raspberry Pi

The water sensor Pi will just send you an e-mail if it detects a problem. The Green LED means all OK, Red means water was detected. You should get an email that the system is starting whenever it gets restarted, then a wet or dry email as things occur. No port forwarding is needed for this system. Load the wetsensor.py code below onto the WetSensor Pi. Copy the wetsensor.py code from github, and edit the code with nano to set the desired email addresses that you want notified of a leak, presumably yourself and perhaps a local service watching the house for emergencies.

```
#! /usr/bin/python3

# Python program to monitor water sensor and send mail if liquid is detected
# Use gpiozero V1.4 library for simplicity
# Dry sensor is low V, wet sensor is high V (~Vcc/2), but line,
# no_line appear reversed who knows why
# Bill Ballard August 2017

from gpiozero import LED, LineSensor
import os
import time

# initialization

red = LED(17)
green = LED(22)
sensor = LineSensor(27, threshold=0.2, sample_rate=1.0)

# give a good name so you will know where the alarm is coming from

name = "Location"

# if you need to send multiple emails, simply add to this list

email = ["youremail@gmail.com", "anotheremail@att.net"]
```

```
# let them know code started

init = "echo " + time.ctime()
init = init + " | heirloom-mailx -s '" + name + " liquid detector started' "

for addr in email:
    outs = init + addr
    os.system(outs)

wet = " | heirloom-mailx -s '" + name + " liquid detected' "
dry = " | heirloom-mailx -s '" + name + " no liquid detected' "

# function definitions

def WET():
# high line
    green.off()
    red.on()
    for addr in email:
        outs = "echo " + time.ctime() + wet + addr
        os.system(outs)

def DRY():
# low line
    green.on()
    red.off()
    for addr in email:
        outs = "echo " + time.ctime() + dry + addr
        os.system(outs)

# assume dry when installed

green.on()
red.off()

# infinite loop
while True:
    sensor.when_no_line = lambda: WET()
    sensor.when_line = lambda: DRY()
```

To start the water sensor at boot:

```
$ sudo nano /etc/rc.local
```

add this line at the end.

```
python3 /home/pi/wetsensor.py > /home/pi/wetsensor.log &
```

save and exit.

View from outside

At the remote location, login to the internet wireless with another computer. In a browser, go to freedns.afraid.org. Create a free account and then create a Type A subdomain such as fap.chickenkiller.com (there are other free options than chickenkiller.com available if you prefer) Create two more subdomains such as fapcam1.chickenkiller.com and fapcam2.chickenkiller.com, also Type A. There is a maximum of 5 for free. This next step is the critical one to getting access from

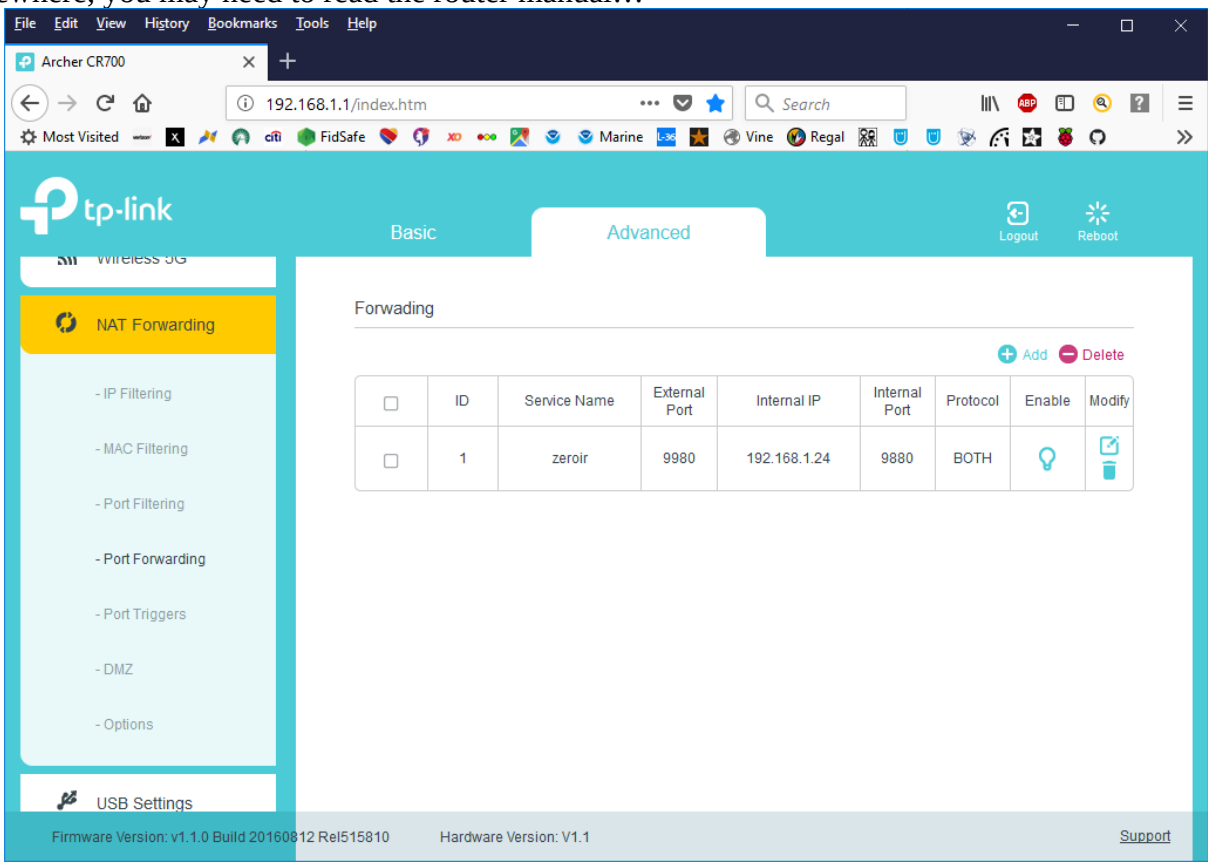
outside the local network. From the Subdomains page in FreeDNS, click on the fapcam1 entry and then click on Forward to a URL and redirect to
http://fap.chickenkiller.com:9880/cam1 (for instance)
do that for both cameras with the second forwarded to
http://fap.chickenkiller.com:9980/cam2
The initial fap.chickenkiller.com is essentially the modem, and we will redirect the ports to the different ZeroWs.

Next go to the dynamic DNS section https://freedns.afraid.org/dynamic/index.php. The last line at the bottom has a link to a quick cron example on the bottom right. Click on that and then copy the last line (probably looks like two lines). Then

```
$ sudo crontab -e
```

and paste that line in the bottom of the crontab file. Save and exit. You only need to do this on one of the computers, any one you choose. This code will continually update the variable IP address your Internet provider gives you making fap.chickenkiller.com robust to random changes in the IP address.

At the remote house, use a browser to go to 192.168.1.1, 10.0.0.1 or whatever your router IP address is. The login is often user: admin with password: password, but you ought to know this. You will need to set up port forwarding for the camera systems. This capability is generally somewhere in the advanced settings. You may need the actual IP address of the camera units which can be found in the remote location's router when you log in to do the port forward. Just set up the specific ports you selected before to allow forwarding. Each router has a bit of unique software here, but the capability is somewhere, you may need to read the router manual…



To test everything from outside the network, turn off wireless on your phone or tablet and enter fapcam1.chickenkiller.com (for instance). You should be prompted for a user name and password, then get to the control console for Rpi_Cam_Web_Interface. Check both camera systems for access. If you encounter problems, first reboot the router, if that doesn't work, double check the port forwarding.

Remote Maintenance

The best solution here appears to be Dataplicity. Go to Dataplicity.com, create an account, then add a new device. The first device is free, and fortunately we only need one. A line of code will appear to cut and paste into a terminal on the Raspberry Pi (this is simplest if you have used BitVise or Putty to open a remote terminal on the Camera1 Raspberry Pi. This only needs to be done on one of the ZeroW systems (use Camera 1), as you can ssh into the others from that one. Execute this code and you will obtain a link for your browser that will allow you to log in to the Pi remotely. In order to do anything useful, you must execute this command on the remote Pi through Dataplicity.

```
$ su pi
```

and enter the password. Then you can move around freely and do updates as well as ssh into the other ZeroWs on the remote network to update those computers. This will work even when the systems are moved to a remote location.

To access the other Raspberry Pis do ssh like this

```
$ ssh pi@camera2.local
```

You will be prompted for the password, then be in the system for updates. Use the exit command to log out.

Hardware setup

The two cameras and cases come with a cable that is the perfect length. For camera 2, connect the cable to the Pi (install the SD card first) and the camera, then snap the Pi into the base and the camera into the lid. Both should fit nicely, be sure the Pi connectors align with the case openings. Connect the top and bottom and you are done, except for coming up with a suitable way to mount the camera in its proper position.

Camera 1 will have the PIR sensor and the Temperature/Humidity sensor and therefore requires a bit more work. Solder on a 2x20 header to the Pi, ***but on the back of the board***. Also solder this type of header on the WetSensor Pi but in the normal configuration on the top of the board.

The PIR sensor can be direct wired to the Pi header or to another connector if you prefer, an old 26 pin header is quite adequate.. The PIR comes with a convenient cable. Solder the Red wire (V+) to Pin 2 (5V), the Black wire (ground) to Pin 14 (ground), and the Yellow sensor wire to pin 12 (GPIO23). ***Remember you are soldering on the back side so do your counting carefully. (It took me 3 tries to get this right, and I was trying hard.)***

Next we set up the DHT22. Bend pin 3 (1 is on the left) of the sensor out of the way or cut it off. Solder a 10k resistor from Pin 17 (3.3V) to Pin 19 (GPIO10), then attach the leftmost pin of the DHT22 sensor to Pin 17 and the second pin of the DHT22 to Pin 19. Bend the wire Pin 4 of the DHT22 to align with Pin 25 and solder it on. Now you can put the PIR cable through the cutout in the bottom of the case,

and lower the pi into the bottom, threading the DHT22 through. Snap the pi in place, then snap the camera in the top of the case and close the case.

For the WetSensor, solder a 10k resistor between the power and sensor pins at the sensor itself. Then attach the red, black and yellow cables from a long multi-conductor cable (some 3 or 4 conductor alarm wire from the hardware store is perfect) to the +, - and sensor pins of the sensor. At the Raspberry Pi end, use a small piece of perforated board to mount the LEDs and associated resistors to a section of header connector. Pin 13 (GPIO27) connects to a 100 ohm resistor, then to the sensor line. Pin 11 (GPIO17) connects to a 100 ohm resistor, then to a red LED anode (longer lead). Pin 15 (GPIO22) connects to a 100 ohm resistor then to a green led anode. The two LED cathodes and the black wire from the sensor connect to Pin9 (Ground) and the red wire from the sensor connects to Pin17 (3.3V). With a bit of care, you can make this a nice package that plugs into the Pi.

In summary, you are connecting
DHT
- Pin 17 Power +,
- Pin 21 sens (WiringPi 13),
- Pin25 ground

PIR
- Pin 4 Power +,
- Pin 12 sens (GPIO18),
- Pin 14 ground

Pinouts for GPIO viewed from the top of the board