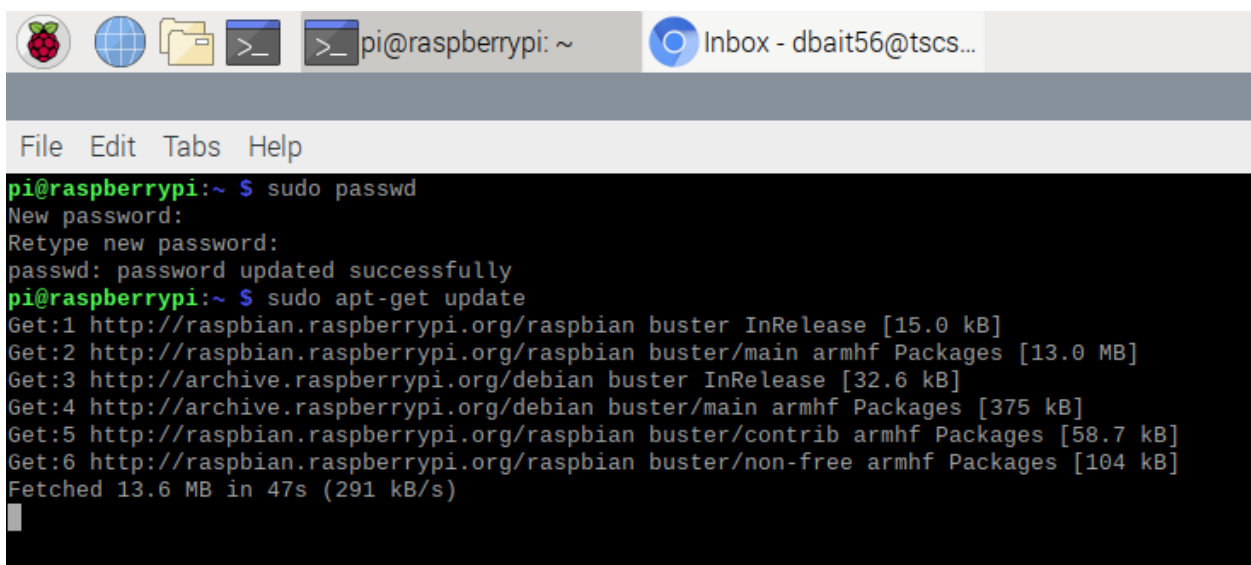# Weather Station

A Project By: Duncan Baitz



```
File  Edit  Tabs  Help
pi@raspberrypi:~ $ python3 google_spreadsheet.py
Logging sensor measurements to Duncans Weather Station every 30 seconds.
Press Ctrl-C to quit.
Temperature: 23.2 C
Humidity:    51.6 %
Wrote a row to Duncans Weather Station
Temperature: 22.8 C
Humidity:    52.9 %
Wrote a row to Duncans Weather Station
Temperature: 22.8 C
Humidity:    53.0 %
Wrote a row to Duncans Weather Station
Traceback (most recent call last):
  File "google_spreadsheet.py", line 92, in <module>
    temp = dhtDevice.temperature
  File "/home/pi/.local/lib/python3.7/site-packages/adafruit_dht.py", line 259, in temperature
    self.measure()
  File "/home/pi/.local/lib/python3.7/site-packages/adafruit_dht.py", line 243, in measure
    raise RuntimeError("Checksum did not validate. Try again.")
RuntimeError: Checksum did not validate. Try again.
pi@raspberrypi:~ $
```
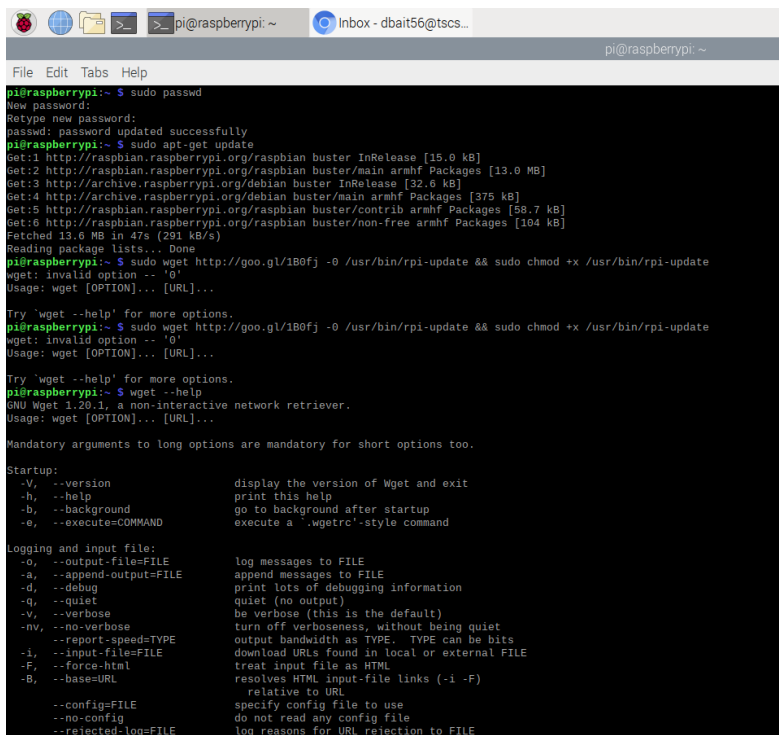
**6-22-21**

Today I restarted my project for the 5th time hoping it will turn out right this time. The first thing I did was install the operating system I used throughout the project, Raspbian. Raspbian is the most commonly used operating system because it comes as a full desktop kit. Then I had my dad connect to the internet so I could do what I needed to do. Next, I changed my password so it was not Raspberry(the default password). This is done with the command *sudo passwd*.

Now before I install anything I needed to run *sudo apt-get update*, this updates the operating system to make sure we have the latest version. Next, I checked to make sure I was running the latest version I ran a *sudo wget http://goo.gl/1B0fJ -0 /usr/bin/rpi-update && sudo chmod +x /usr/bin/rpi-update* which installs hex. Finally, I run *ifconfig* where my IP address is 192.168.0.5. I typed that into chromium to make sure it was working. That concluded the basic setup.

Then I needed to install apache. To do that I ran the following commands: sudo *apt install apache2, sudo apt install php, sudo apt install php-mysql, sudo apt install libapache2-mod-php* that will fully install apache.

Next, I installed MariaDB Server and client, to do that you run *sudo apt-get install mariadb-server, sudo apt-get install mariadb-common, and sudo apt-get install mariadb-client*. Then I went back to chromium typed in my IP address to make sure apache was up and running. After you make sure apache is up and running you need to remove the index.html file because it will cause problems installing wordpress later on. Do that with *rm /var/www/html/index.html*, type *ls* to make sure it gets removed. The command *ls* shows you all the files in the directory you are currently in.

Now we need to install wordpress. The first thing we need to do is go to the directory it will be downloaded in. Do that with the command *cd /var/www/html*. Next, I installed wordpress with the command *sudo wget http://wordpress.org/latest.zip*, this installs wordpress but to make it usable we have to unzip the latest.zip. I do this with *sudo unzip latest.zip*.

**6-23-21**

Now I needed to move what file wordpress was in. I did that with *sudo mv wordpress/\* .* then you can remove the now empty folder with *sudo rm -rf wordpress*. Next, you need to install the plugins. You can do this with *sudo chown -R www-data:www-data /var/www*, This should install all the proper plugins needed for wordpress. Since these are permalinks we need to use mod-rewrite to get proper permissions. I did that with *a2enmod rewrite*. Now to edit the main site settings. This can be done with *sudo nano /etc/apache2/apache2.conf*. You need to change only the first two overrides. But you need to make sure it's ONLY the first two that you change. If you change any of the other overrides it will cause issues and you will just have to change it back later on. To finish this step you need to restart apache. Do that with *sudo /etc/init.d/apache2 restart*. You are almost ready to see your wordpress site!

```
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See http://httpd.apache.org/docs/2.4/ for detailed information about
# the directives and /usr/share/doc/apache2/README.Debian about Debian specific
# hints.
#
#
# Summary of how the Apache 2 configuration works in Debian:
# The Apache 2 web server configuration in Debian is quite different to
# upstream's suggested way to configure the web server. This is because Debian's
# default Apache2 installation attempts to make adding and removing modules,
# virtual hosts, and extra configuration directives as flexible as possible, in
# order to make automating the changes and administering the server as easy as
# possible.

# It is split into several files forming the configuration hierarchy outlined
# below, all located in the /etc/apache2/ directory:
#
#       /etc/apache2/
#       |-- apache2.conf
#       |        `--  ports.conf
#       |-- mods-enabled
#       |        |-- *.load
#       |        `-- *.conf
#       |-- conf-enabled
#       |        `-- *.conf
#       `-- sites-enabled
#                `-- *.conf
#
#
# * apache2.conf is the main configuration file (this file). It puts the pieces
#   together by including all remaining configuration files when starting up the
#   web server.
#
# * ports.conf is always included from the main configuration file. It is
#   supposed to determine listening ports for incoming connections which can be
#   customized anytime.
#
# * Configuration files in the mods-enabled/, conf-enabled/ and sites-enabled/
#   directories contain particular configuration snippets which manage modules,
#   global configuration fragments, or virtual host configurations,
#   respectively.
#
#   They are activated by symlinking available configuration files from their
#   respective *-available/ counterparts. These should be managed by using our
```

```
^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit         ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell
```

```
  GNU nano 3.2

# error, crit, alert, emerg.
# It is also possible to configure the log level for particular modules, e.g.
# "LogLevel info ssl:warn"
#
LogLevel warn

# Include module configuration:
IncludeOptional mods-enabled/*.load
IncludeOptional mods-enabled/*.conf

# Include list of ports to listen on
Include ports.conf


# Sets the default security model of the Apache2 HTTPD server. It does
# not allow access to the root filesystem outside of /usr/share and /var/www.
# The former is used by web applications packaged in Debian,
# the latter may be used for local directories served by the web server. If
# your system is serving content from a sub-directory in /srv you must allow
# access here, or in any related virtual host.
<Directory />
        Options FollowSymLinks
        AllowOverride None
        Require all denied
</Directory>

<Directory /usr/share>
        AllowOverride None
        Require all granted
</Directory>

<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>

#<Directory /srv/>
#       Options Indexes FollowSymLinks
#       AllowOverride None
#       Require all granted
#</Directory>
```
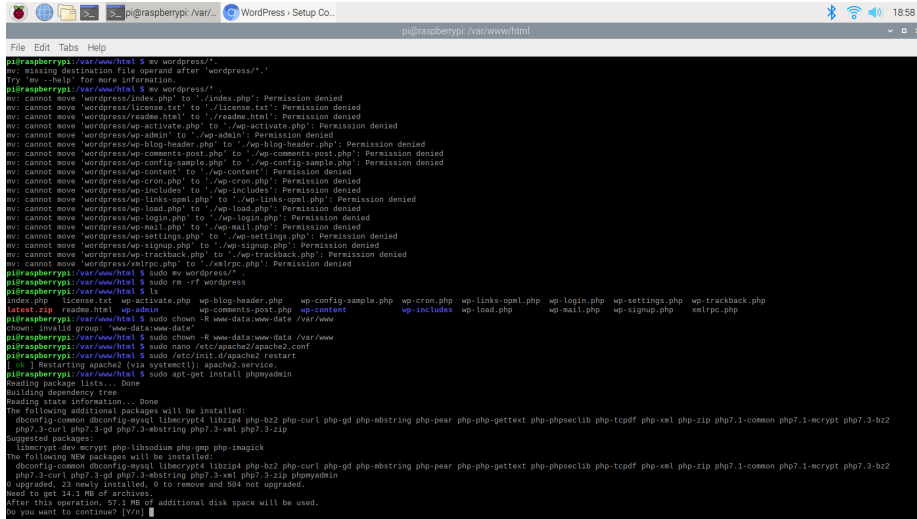
```
^G Get Help      ^O Write Out     ^W Where Is      ^K Cut Text      ^J Justify
^X Exit          ^R Read File     ^\ Replace       ^U Uncut Text    ^T To Spell
```

        If you plan to do lots of work on your weather station, you need to do the following but

not skip this entire paragraph. The wordpress database takes a lot of work to make a more
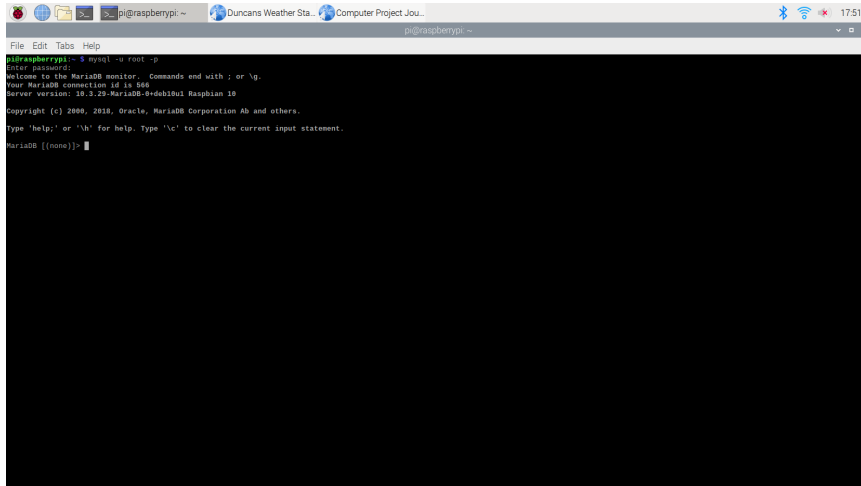
advanced website where you post quite often or if you are making multiple websites. If you are not doing this the following command will seem overkill but if you are going to be spending a lot of time on the database this will seem normal. So to keep mysql from overloading, you will need to install phpmyadmin, this will give you more organization than an overloaded mysql is. To install phpmyadmin, you will need to run *sudo apt-get install phpmyadmin*.

Now you have to sign into mysql monitor, this is where all your wordpress database information will be stored. To do that you will need to use the command *sudo mysql -u root -p* the first time you sign in it will automatically sign you in, but afterward, you will be asked for the password you set. You will need to remember that every command in the mysql monitor will need to end with a semicolon or /g for it to be executed. This is important because if you don't you will have to exit mysql monitor with *^c* to exit.
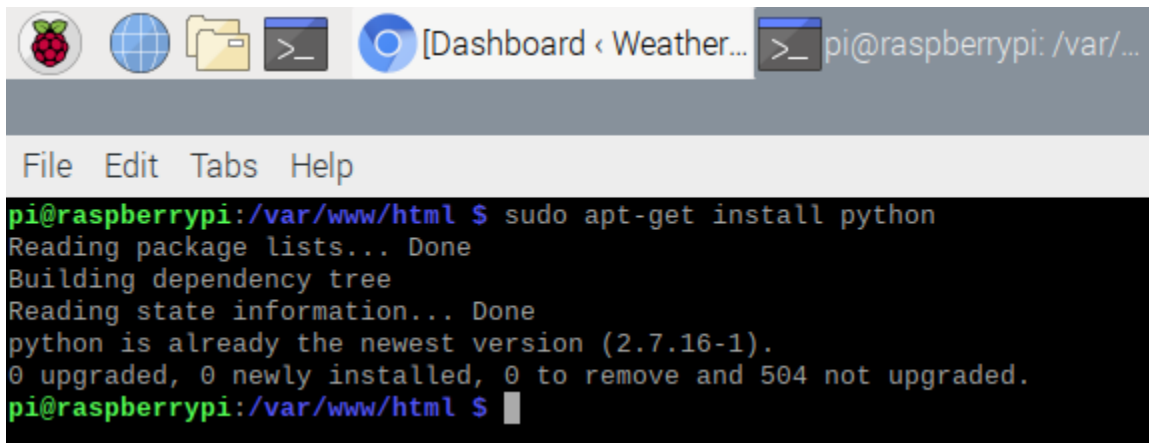
In the command window, you will need to create the database with *CREATE DATABASEWordPressDB;* this will install the database that all your sites will be made from. Next, we need to permit the database. You can do this with *grant all on WordPressDB.\* to root@localhost identified by'(your password)';* you are all finished in the mysql monitor to exit you type *exit;*. Now go to chromium and type the IP address in the browser. You will need to select the language. Then it will ask you for your database name, your username, password, and the host. Mine was WordpressDB, root, \*\*\*\*\*\*, and localhost.
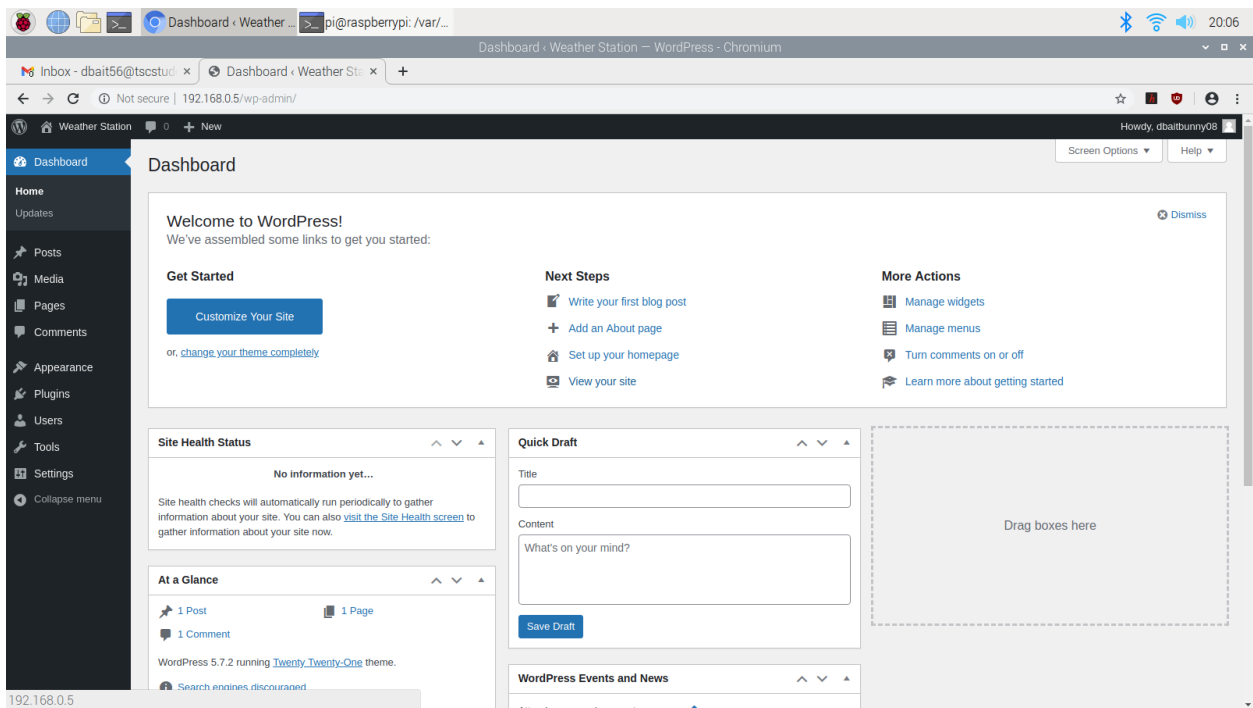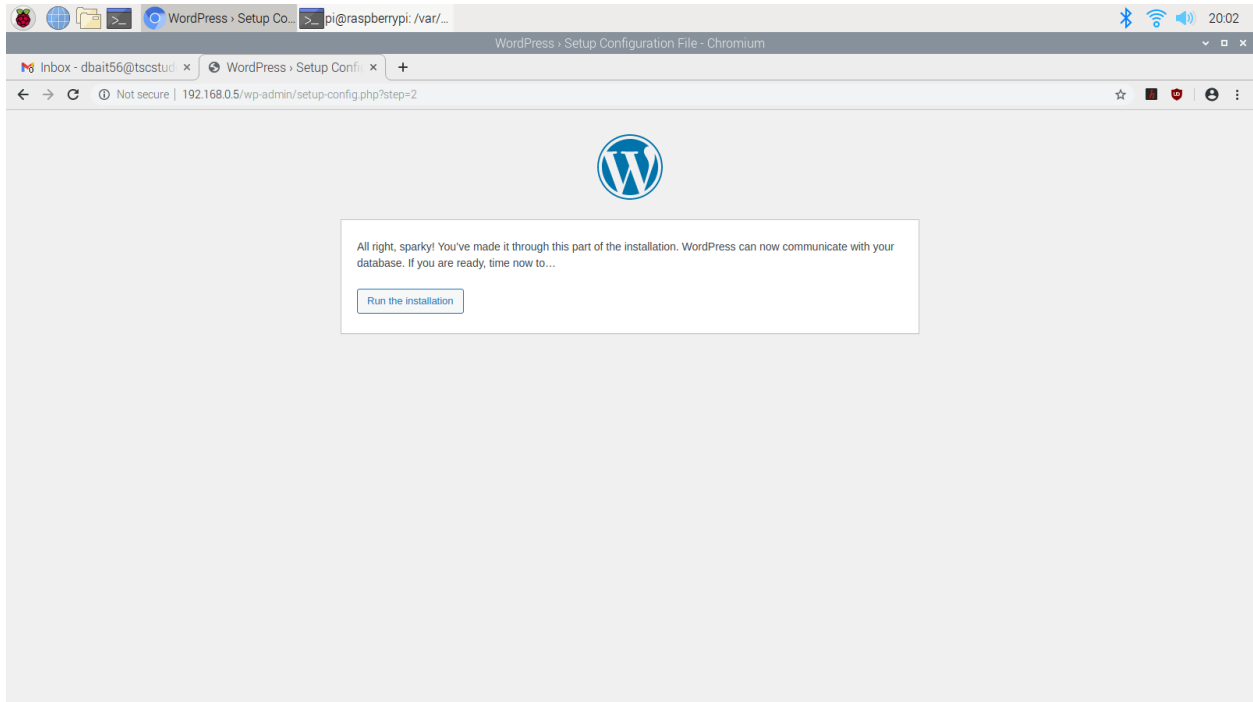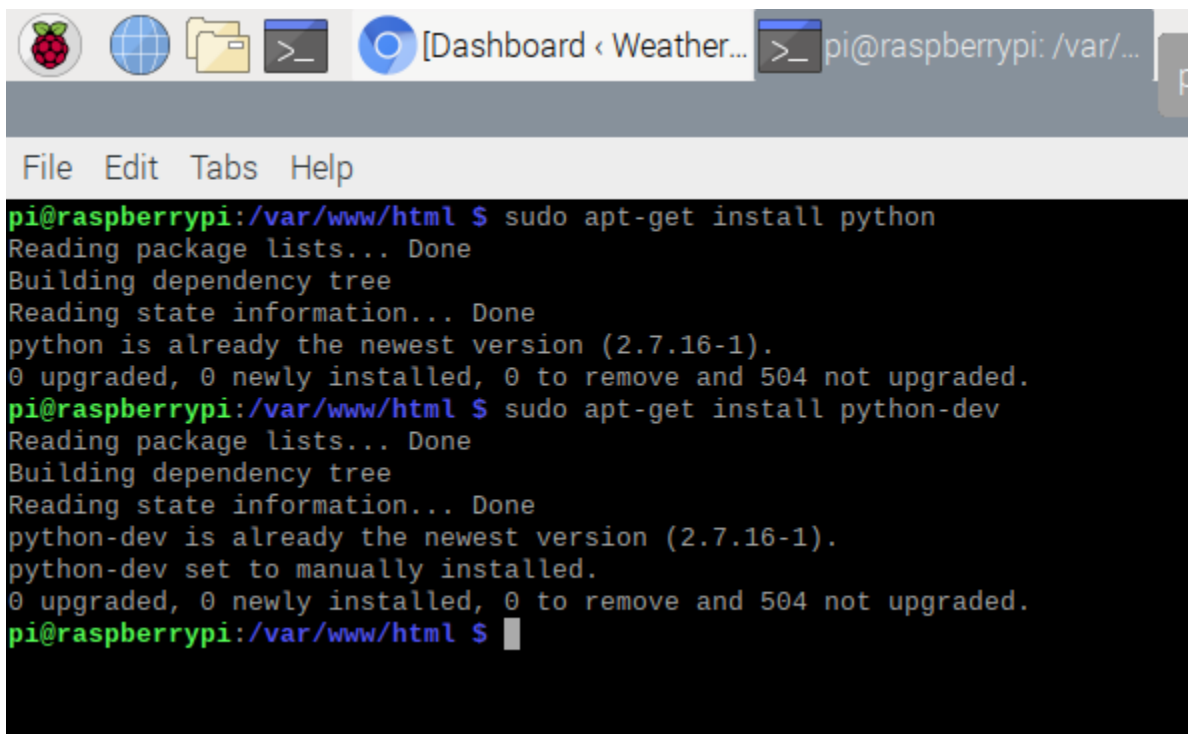
Now I had to install python, this is what the code for the sensor will use. To do that you will need to run multiple commands *sudo apt-get install python, sudo apt-get install python-dev, sudo apt-get install libmariadb-dev, sudo apt-get install build-essentials, sudo apt-get install*

*python-ossl*. Now you need to also install eash_install. Before I could do that I had to install python setup tools. Do that with *sudo apt-get install python-setuptools*. Now you can get easy_install. To do that you can use the command *sudo easy_install -U distribute*. This will update easy_install. Now you have to install the part that saves it to the wordpress database. The command you need to run to do this is *sudo easy_install MySQL-python*. That last command is essential, if you don't install it you will not be getting any readings.

File   Edit   Tabs   Help

```
Selecting previously unselected package libtasn1-6-dev:armhf.
Preparing to unpack .../08-libtasn1-6-dev_4.13-3_armhf.deb ...
Unpacking libtasn1-6-dev:armhf (4.13-3) ...
Selecting previously unselected package nettle-dev:armhf.
Preparing to unpack .../09-nettle-dev_3.4.1-1+deb10u1_armhf.deb ...
Unpacking nettle-dev:armhf (3.4.1-1+deb10u1) ...
Selecting previously unselected package libgnutls28-dev:armhf.
Preparing to unpack .../10-libgnutls28-dev_3.6.7-4+deb10u7_armhf.deb ...
Unpacking libgnutls28-dev:armhf (3.6.7-4+deb10u7) ...
Selecting previously unselected package libmariadb-dev.
Preparing to unpack .../11-libmariadb-dev_1%3a10.3.29-0+deb10u1_armhf.deb ...
Unpacking libmariadb-dev (1:10.3.29-0+deb10u1) ...
Selecting previously unselected package libtasn1-doc.
Preparing to unpack .../12-libtasn1-doc_4.13-3_all.deb ...
Unpacking libtasn1-doc (4.13-3) ...
Setting up libgnutls-openssl27:armhf (3.6.7-4+deb10u7) ...
Setting up libtasn1-doc (4.13-3) ...
Setting up libunbound8:armhf (1.9.0-2+deb10u2) ...
Setting up libgmpxx4ldbl:armhf (2:6.1.2+dfsg-4) ...
Setting up libgnutlsxx28:armhf (3.6.7-4+deb10u7) ...
Setting up libidn2-dev:armhf (2.0.5-1+deb10u1) ...
Setting up libtasn1-6-dev:armhf (4.13-3) ...
Setting up libp11-kit-dev:armhf (0.23.15-2+deb10u1) ...
Setting up libgnutls-dane0:armhf (3.6.7-4+deb10u7) ...
Setting up libgmp-dev:armhf (2:6.1.2+dfsg-4) ...
Setting up nettle-dev:armhf (3.4.1-1+deb10u1) ...
Setting up libgnutls28-dev:armhf (3.6.7-4+deb10u7) ...
Setting up libmariadb-dev (1:10.3.29-0+deb10u1) ...
Processing triggers for libc-bin (2.28-10+rpi1) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for install-info (6.5.0.dfsg.1-4+b1) ...
pi@raspberrypi:/var/www/html $ sudo apt-get install build-essentials
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package build-essentials
pi@raspberrypi:/var/www/html $ sudo apt-get install build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.6).
0 upgraded, 0 newly installed, 0 to remove and 499 not upgraded.
pi@raspberrypi:/var/www/html $ sudo apt-get install python-openssl
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-openssl is already the newest version (19.0.0-1).
python-openssl set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 499 not upgraded.
pi@raspberrypi:/var/www/html $
```

Now I needed to log back into my wordpress database to make sure everything was still working and the password that I had written down that it gave me was not working. So I tried everything I could think of. I was going to try and reset the password with an emergency reset but I was scared that it would ruin something so I didn't. I decided that I needed to take a break and do some research on another way to reset the password.

# Research!

**7-1-21**

      I finally figured out how to change my password. To do that I had to go into the mysql monitor and sign in to the wordpress database. Then it will show us the users and their passwords. To sign in to the mysql monitor you use the command *sudo mysql -u root -p*. Now it asked me for the password. Then I had to access the database through mysql, I did that with *use WordPressDB;*.

File  Edit  Tabs  Help

```
pi@raspberrypi:~ $ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 22
Server version: 10.3.29-MariaDB-0+deb10u1 Raspbian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use WordpressDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [WordpressDB]> show tables;
+-----------------------+
| Tables_in_WordpressDB |
+-----------------------+
| wp_commentmeta        |
| wp_comments           |
| wp_links              |
| wp_options            |
| wp_postmeta           |
| wp_posts              |
| wp_term_relationships |
| wp_term_taxonomy      |
| wp_termmeta           |
| wp_terms              |
| wp_usermeta           |
| wp_users              |
+-----------------------+
12 rows in set (0.001 sec)

MariaDB [WordpressDB]> SELECT ID, user_login, user_pass FROM wp_users
    -> Ctrl-C -- exit!
Aborted
pi@raspberrypi:~ $ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 23
Server version: 10.3.29-MariaDB-0+deb10u1 Raspbian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT ID, user_login, user_pass FROM wp_users;
ERROR 1046 (3D000): No database selected
MariaDB [(none)]> use WordpressDB;
```

```
       -> SELECT ID, user_login, user_pass FROM wp_users;Ctrl-C -- exit!
Aborted
pi@raspberrypi:~ $ mysql -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
pi@raspberrypi:~ $ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 32
Server version: 10.3.29-MariaDB-0+deb10u1 Raspbian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use WordpressDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [WordpressDB]> show tables;
+----------------------+
| Tables_in_WordpressDB |
+----------------------+
| wp_commentmeta       |
| wp_comments          |
| wp_links             |
| wp_options           |
| wp_postmeta          |
| wp_posts             |
| wp_term_relationships |
| wp_term_taxonomy     |
| wp_termmeta          |
| wp_terms             |
| wp_usermeta          |
| wp_users             |
+----------------------+
12 rows in set (0.001 sec)

MariaDB [WordpressDB]> UPDATE wp_users SET user_pass = 12ffc8a2e43645614f22188604f691b6 WHERE ID = 1;
ERROR 1054 (42S22): Unknown column '12ffc8a2e43645614f22188604f691b6' in 'field list'
MariaDB [WordpressDB]> exit;
Bye
pi@raspberrypi:~ $ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 33
Server version: 10.3.29-MariaDB-0+deb10u1 Raspbian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
```
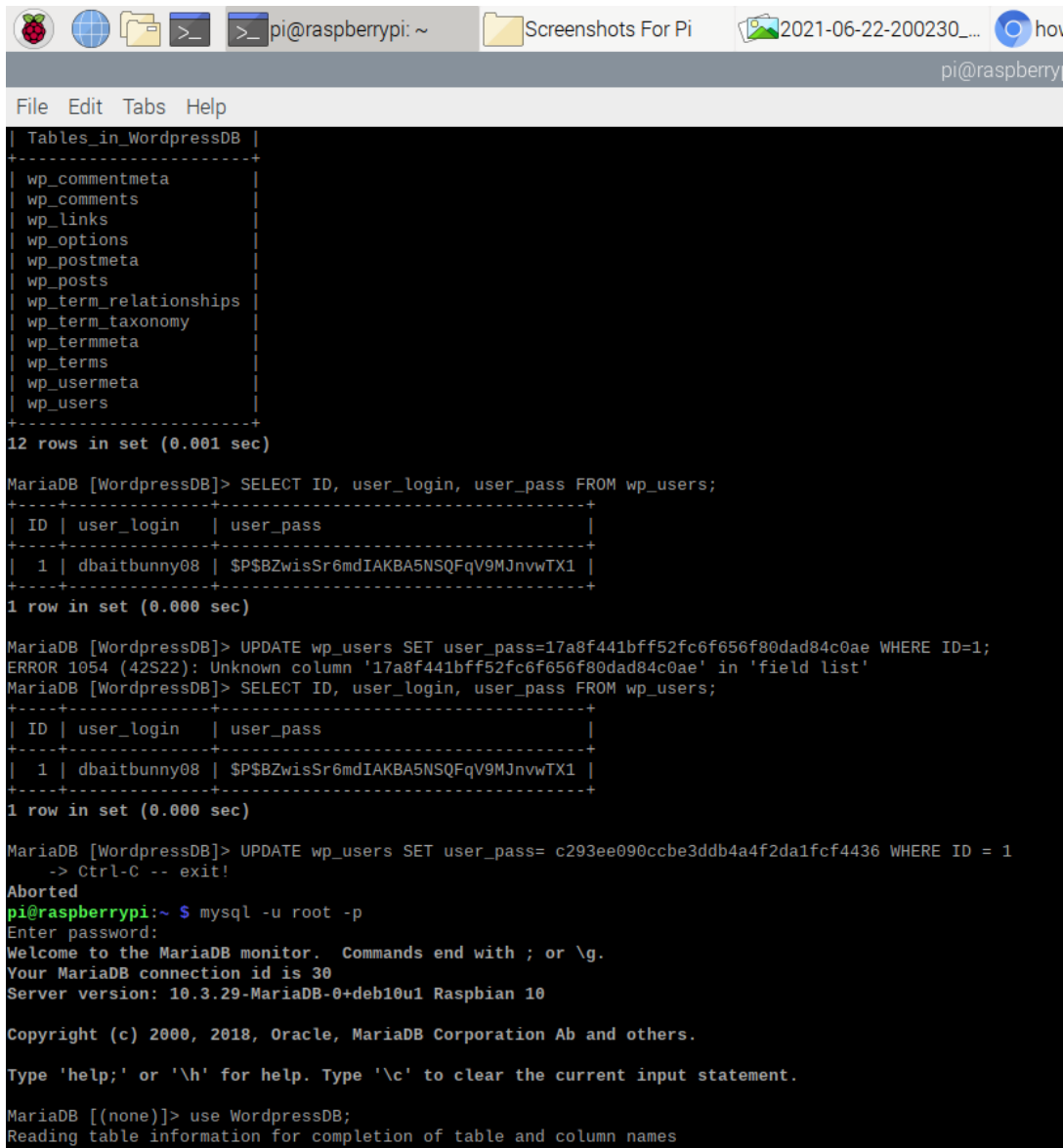
Just like the other time I was in the mysql monitor EVERY command MUST end with a
semicolon otherwise, it won't be executed. Then, I had to see which folder the users were under,
to do that I used the command *show tables;*. This shows me all the folders that the wordpress
database uses. I was looking for the folder labeled **wp_users**. This folder holds all the users and
their passwords. To see the contents in the file use the command *SELECT ID, user_login,*
*user_pass FROM wp_users;*. Please note that even though there are commas this is one
command. Now you should see a table that has two columns. In the first column should be a user
and their user ID. The second column should be the user's password. Then I tried the password

that it had given me to see if that would work, but for some reason, that password wasn't

working either. So I used the last resort and decided to change the password. To do that I had to

use a command that used an MD5 hash to create a long and complicated password that can be

used in short words. For example, if I wanted my password iloveweather then my MD5 hash

would be  10b099b4c6e6bab79c418f817d3974ea. So I would type in my password and the hash

would convert it to log you in.  To change my password I had to use the command *UPDATE*

*wp_users SET user_pass=MD5('(your password)') WHERE ID=1;*.

Now I was having trouble with downloading the library that the data should be stored in. After doing some more research, the library no longer exists and was replaced with a google drive-based code. I was so upset at this. Everything that I had been working on and failing at was not needed. Now I was not going to let this stop me. I read through the tutorial I found that would work with my sensor and got to work. Turns out this tutorial links your sensor to google sheets where it automatically logs to google sheets where you don't have to do a thing. That was kind of nice but I was still upset.

*Research!*

Now back to coding. The first thing I had to do was code the sensor. To do this I had to install a new python system that works with google docs. So I went with Circuit Python Libraries. To install this you run the command *sudo pip3 adafruit-circuitpython-dht* and *sudo apt-get install libgpiod2*. Now I had to test the library. So I used the command *sudo nano*

*dht_simpletest.py* which opens a text file. Then I copied and pasted the script for the simple test because it is 37 lines of complicated text and explaining.
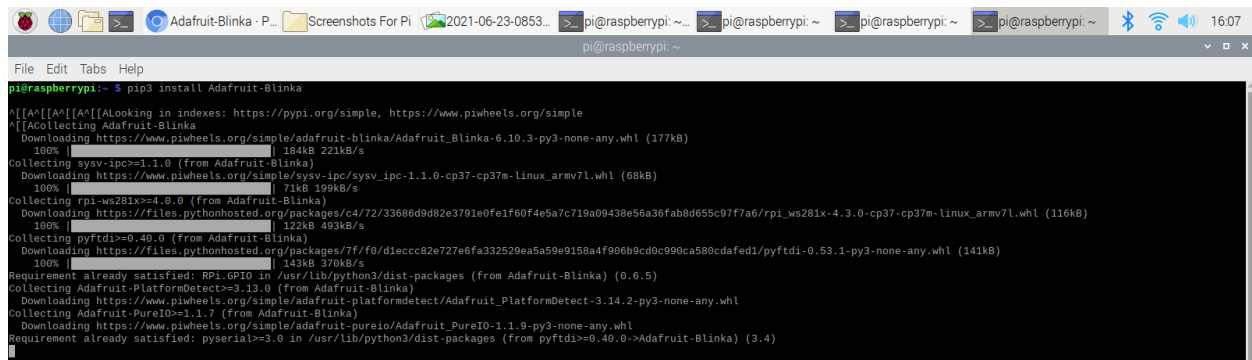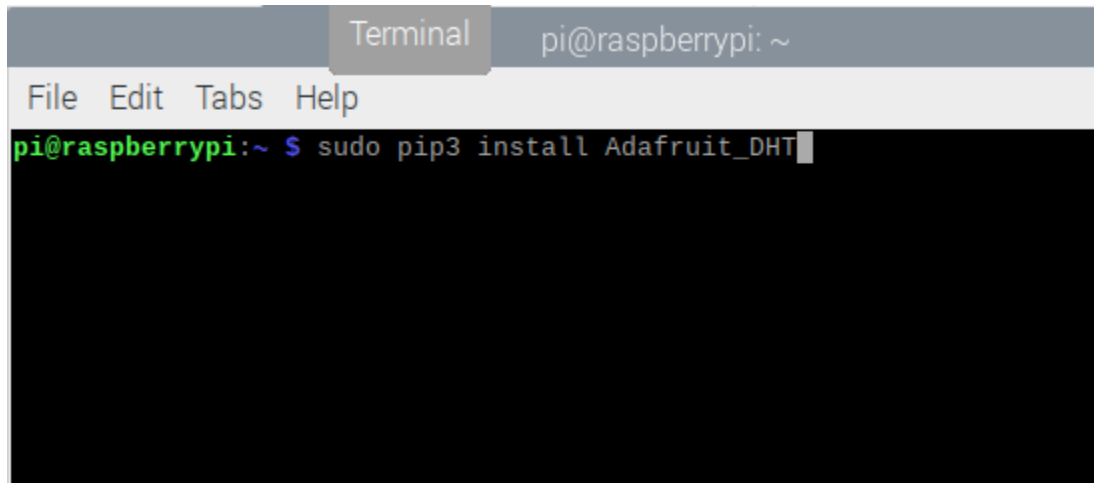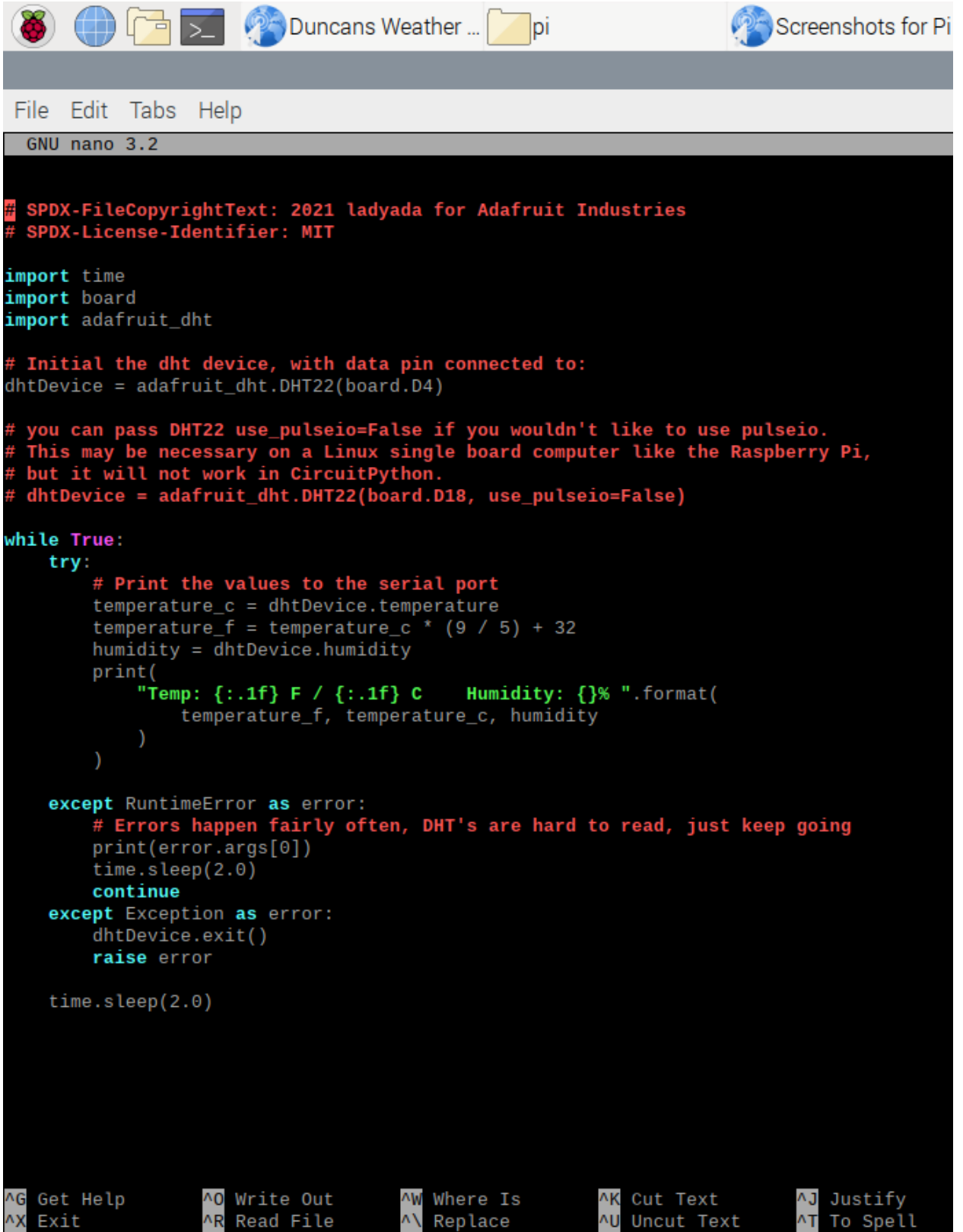




Now to wire the sensor. To do this you will need pigtails, a DHT22/AM2032 sensor, a breadboard, and a pi jumper cable. Now I had to be careful because when attaching the jumper cable if I bent any of the pins on the pi it would make it ten times harder to put on. So once it was plugged into the pi I plugged it into the breadboard. The breadboard should have at least a 30 hole long by 10 holes wide with a ditch in the middle and a positive and negative row down each side. I decided to get a 60 hole long breadboard so my wiring could be in the middle. So with the jumper in the middle I put the yellow wire to pin #4 I put the red wire to pin 3U3 and I put the black wire to negative with a pigtail from negative to ground pin.
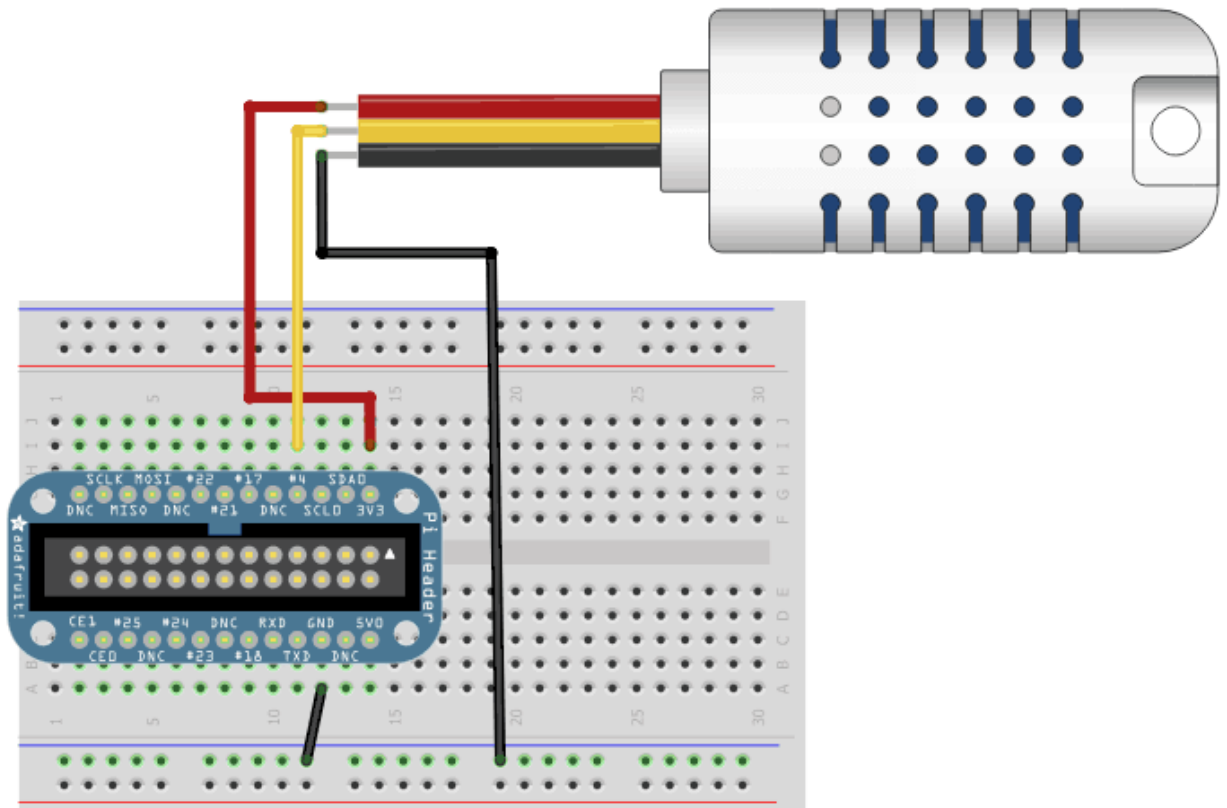
Now I can test my sensor. To do that I can use the command *python3 dht_simpletest.py*.
This should immediately start giving me data but it won't log it. To log it you will need to start
with an empty spreadsheet and delete all but one row. You can also delete all but the first three
columns. Label Column A Date/Time, label Column B Temperature C, and label Column C
Humidity %. Now I have to go to google cloud and give my pi permission to edit the spreadsheet
as a service bot. To do that I had to enable the API permissions.

File   Edit   Tabs   Help

```
  GNU nano 3.2


# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_dht

# Initial the dht device, with data pin connected to:
dhtDevice = adafruit_dht.DHT22(board.D4)

# you can pass DHT22 use_pulseio=False if you wouldn't like to use pulseio.
# This may be necessary on a Linux single board computer like the Raspberry Pi,
# but it will not work in CircuitPython.
# dhtDevice = adafruit_dht.DHT22(board.D18, use_pulseio=False)

while True:
    try:
        # Print the values to the serial port
        temperature_c = dhtDevice.temperature
        temperature_f = temperature_c * (9 / 5) + 32
        humidity = dhtDevice.humidity
        print(
            "Temp: {:.1f} F / {:.1f} C    Humidity: {}% ".format(
                temperature_f, temperature_c, humidity
            )
        )

    except RuntimeError as error:
        # Errors happen fairly often, DHT's are hard to read, just keep going
        print(error.args[0])
        time.sleep(2.0)
        continue
    except Exception as error:
        dhtDevice.exit()
        raise error

    time.sleep(2.0)
```
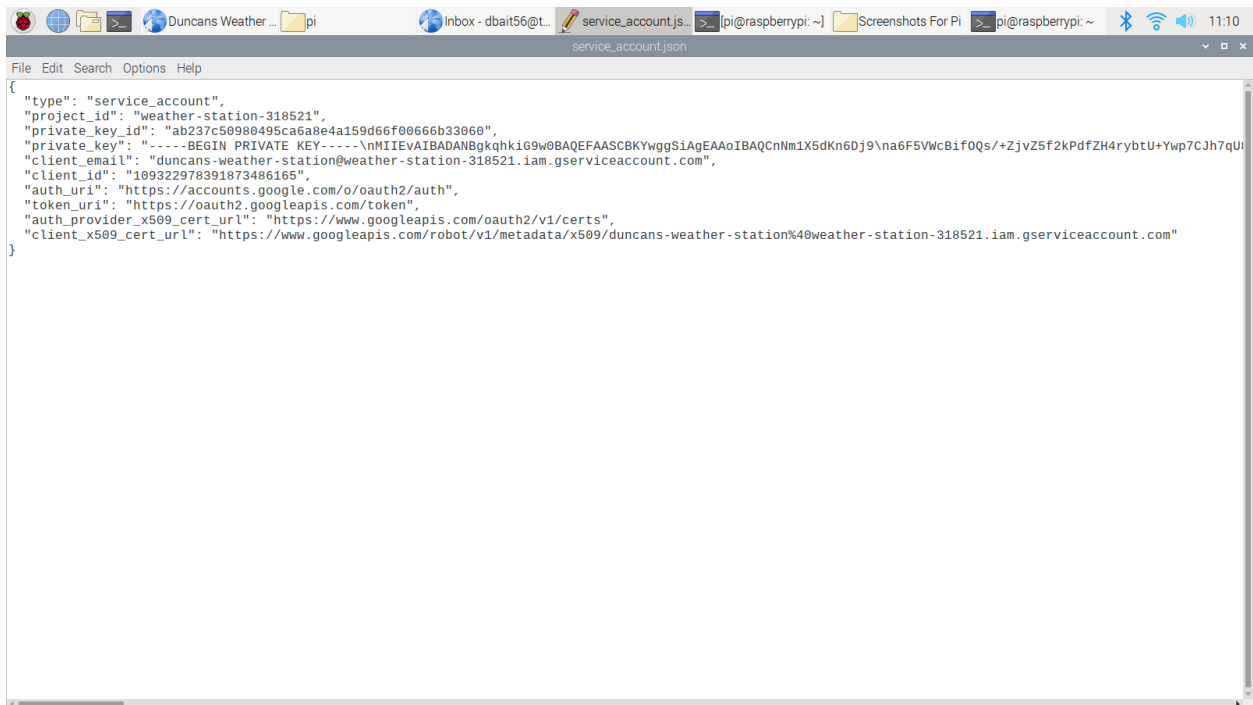
```
^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit         ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell
```

**7-2-21**
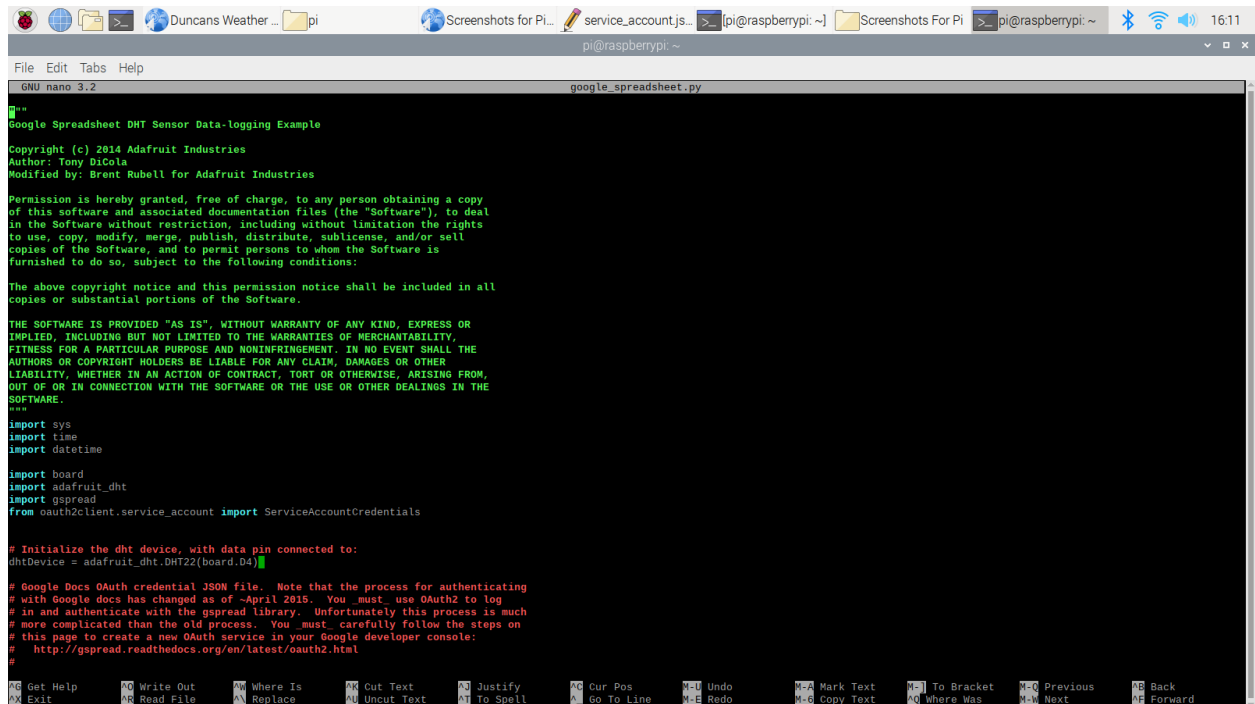
To enable API permissions all I had to do was search up in google cloud Google Drive API and Google Sheets API. Please note that due to security issues I am not allowed to show screenshots. Then I had to enable them by pressing enable. Now it gave me a .json file to download that has all the permissions in it. I named it service_account.json.



Then I copied and pasted the text for long-term recordings that will be logged into a text file called google_spreadsheet.py. To access this I used the command *sudo nano google_spreadsheet.py*. I also copied and pasted this time because I didn't want to get anything wrong and it was 127 lines. After working out the bugs in this file. I have managed to get it to log to a spreadsheet. To activate logging you just run the command *python3 google_spreadsheet.py*. As of writing this, I have successfully gotten 117 readings logged on the spreadsheet.

Now I know the code still has some bugs in it because I can't just leave it running because if

something doesn't return in time it will stop sending code to and from the sensor.



```
"""
Google Spreadsheet DHT Sensor Data-logging Example

Copyright (c) 2014 Adafruit Industries
Author: Tony DiCola
Modified by: Brent Rubell for Adafruit Industries

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
"""
import sys
import time
import datetime

import board
import adafruit_dht
import gspread
from oauth2client.service_account import ServiceAccountCredentials

# Initialize the dht device, with data pin connected to:
dhtDevice = adafruit_dht.DHT22(board.D4)

# Google Docs OAuth credential JSON file.  Note that the process for authenticating
# with Google docs has changed as of ~April 2015.  You _must_ use OAuth2 to log
# in and authenticate with the gspread library.  Unfortunately this process is much
# more complicated than the old process.  You _must_ carefully follow the steps on
# this page to create a new OAuth service in your Google developer console:
#   http://gspread.readthedocs.org/en/latest/oauth2.html
#
```

```python
        scope = ['https://spreadsheets.google.com/feeds', 'https://www.googleapis.com/auth/drive']
        credentials = ServiceAccountCredentials.from_json_keyfile_name(oauth_key_file, scope)
        gc = gspread.authorize(credentials)
        worksheet = gc.open(spreadsheet).sheet1 # pylint: disable=redefined-outer-name
        return worksheet
    except Exception as ex: # pylint: disable=bare-except, broad-except
        print('Unable to login and get spreadsheet.  Check OAuth credentials, spreadsheet name, \
        and make sure spreadsheet is shared to the client_email address in the OAuth .json file!')
        print('Google sheet login failed with error:', ex)
        sys.exit(1)


print('Logging sensor measurements to\
 {0} every {1} seconds.'.format(GDOCS_SPREADSHEET_NAME, FREQUENCY_SECONDS))
print('Press Ctrl-C to quit.')
worksheet = None
while True:
    # Login if necessary.
    if worksheet is None:
        worksheet = login_open_sheet(GDOCS_OAUTH_JSON, GDOCS_SPREADSHEET_NAME)

    # Attempt to get sensor reading.
    temp = dhtDevice.temperature
    humidity = dhtDevice.humidity

    # Skip to the next reading if a valid measurement couldn't be taken.
    # This might happen if the CPU is under a lot of load and the sensor
    # can't be reliably read (timing is critical to read the sensor).
    if humidity is None or temp is None:
        time.sleep(2)
        continue

    print('Temperature: {0:0.1f} C'.format(temp))
    print('Humidity:    {0:0.1f} %'.format(humidity))

    # Append the data in the spreadsheet, including a timestamp
    try:
        worksheet.append_row((datetime.datetime.now().isoformat(), temp, humidity))
    except: # pylint: disable=bare-except, broad-except
        # Error appending data, most likely because credentials are stale.
        # Null out the worksheet so a login is performed at the top of the loop.
        print('Append error, logging in again')
        worksheet = None
        time.sleep(FREQUENCY_SECONDS)
        continue
```

```python
    humidity = dhtDevice.humidity

    # Skip to the next reading if a valid measurement couldn't be taken.
    # This might happen if the CPU is under a lot of load and the sensor
    # can't be reliably read (timing is critical to read the sensor).
    if humidity is None or temp is None:
        time.sleep(2)
        continue

    print('Temperature: {0:0.1f} C'.format(temp))
    print('Humidity:    {0:0.1f} %'.format(humidity))

    # Append the data in the spreadsheet, including a timestamp
    try:
        worksheet.append_row((datetime.datetime.now().isoformat(), temp, humidity))
    except: # pylint: disable=bare-except, broad-except
        # Error appending data, most likely because credentials are stale.
        # Null out the worksheet so a login is performed at the top of the loop.
        print('Append error, logging in again')
        worksheet = None
        time.sleep(FREQUENCY_SECONDS)
        continue

    # Wait 30 seconds before continuing
    print('Wrote a row to {0}'.format(GDOCS_SPREADSHEET_NAME))
    time.sleep(FREQUENCY_SECONDS)
```

```python
# your OAuth2 credentials.  This file has a name like SpreadsheetData-<gibberish>.json.
# Place that file in the same directory as this python script.
#
# Now one last _very important_ step before updating the spreadsheet will work.
# Go to your spreadsheet in Google Spreadsheet and share it to the email address
# inside the 'client_email' setting in the SpreadsheetData-*.json file.  For example
# if the client_email setting inside the .json file has an email address like:
#   149345334675-md0qff5f0kib41meu20f7d1habos3qcu@developer.gserviceaccount.com
```

| | Date/Time | Tempature C | Humidity % |
|---|---|---|---|
| 1 | | | |
| 2 | 2021-07-02T14:59:22.405200 | 24.7 | 53.9 |
| 3 | 2021-07-02T15:00:06.900408 | 23.8 | 54.3 |
| 4 | 2021-07-02T15:00:37.447639 | 23.8 | 53.9 |
| 5 | 2021-07-02T15:01:07.931901 | 23.8 | 53.6 |
| 6 | 2021-07-02T15:01:38.682947 | 23.8 | 53.5 |
| 7 | 2021-07-02T15:02:09.197631 | 23.8 | 53.2 |
| 8 | 2021-07-02T15:02:39.962186 | 23.7 | 52.1 |
| 9 | 2021-07-02T15:03:10.461038 | 23.7 | 52.7 |
| 10 | 2021-07-02T15:03:40.940906 | 23.7 | 53.4 |
| 11 | 2021-07-02T15:04:11.407219 | 23.7 | 52.5 |
| 12 | 2021-07-02T15:04:41.973969 | 23.6 | 52.1 |
| 13 | 2021-07-02T15:05:12.469108 | 23.6 | 51.4 |
| 14 | 2021-07-02T16:18:00.934409 | 23.6 | 54.8 |
| 15 | 2021-07-02T16:18:31.478525 | 24 | 54.2 |
| 16 | 2021-07-02T16:19:02.478131 | 24 | 53.6 |
| 17 | 2021-07-02T16:19:33.489968 | 24 | 53.3 |
| 18 | 2021-07-02T16:20:04.041749 | 24 | 53.8 |
| 19 | 2021-07-02T16:20:34.549232 | 24 | 53.5 |
| 20 | 2021-07-02T16:21:05.316709 | 24 | 53.7 |
| 21 | 2021-07-02T16:21:36.054495 | 24.1 | 53.3 |
| 22 | 2021-07-02T16:22:06.586492 | 24.1 | 52.6 |

# Notes:

```
$ wp user list
$ wp user update 1--user.pas=$up3rstong P4$w0rd
/var/www/html
```

# Sources

Askunbuntu.com

Stackoverflow.com

Raspberryweather.com

wordpress.org/support/article/resetting-your-password/

learn.adafruit.com