

# Charge Controller Data Manager - CCDM

## Getting Started Guide – for CCDM 0.0.2 beta

---

### Table of Contents

Table of Contents .....	1
Overview .....	2
Initial Setup .....	2
System Requirements .....	2
Installing Python .....	3
Win10:.....	3
Linux / Raspbian:.....	7
Checking PIP .....	8
Win10:.....	8
Linux / Raspbian:.....	9
Installing Pillow .....	9
Win10:.....	9
Linux / Raspbian:.....	10
Placing CCDM onto your Device .....	11
Networking.....	11
Initial Configuration of CCDM .....	11
Start CCDM.....	13
Win10:.....	13
Linux / Raspbian:.....	15
Finishing Up.....	15
Appendix .....	16
CCDM UI Screens .....	16
Overview Tab .....	16
Misc Tab .....	17
Config Tab .....	18

## Overview

Welcome to Charge Controller Data Manager (aka CCDM). CCDM is a small python application written to gather data from Midnite Solar Classic series charge controllers, display the data on a UI (user Interface) as well as log some data into a daily log files in simple text form. While this Getting Started guide might seem lengthy, fear not, anyone with some basic computer skills should have no problem getting CCDM running if they take the time to work through this guide carefully. This guide sets out to help prepare your device for CCDM, teach you a bit about python and walk you through configuring, starting up and navigating through the UI (user Interface) of CCDM.

## Initial Setup

Before we dive into this setup guide, let us make some basic assumptions. You likely have this guide because you downloaded the Charge Controller Data manager – CCDM – master package from our GitHub repository. From here on in we will refer to this folder and all its contents as the “CCDM Package”. We’re also going to assume you have a host device (PC, raspberry pi, etc.) you’d like to place CCDM onto to collect data from your Charge Controller. Both this device and your charge controller are powered-up and networked together. If you can say yes to all these basic assumptions, then we are ready to begin.

## System Requirements

CCDM is a lightweight package requiring minimal resources from a host device and can be installed on virtually any device that can also run python and connect to a TCP/IP network. The CCDM package itself requires approximately 3 MB of hard drive space. To date CCDM has been tested on the following device Operating Systems:

Win 10

Linux Mint 17

Raspbian Stretch

CCDM also requires Python, PIP and Pillow to be installed on the device, we will explore how to install these items in this guide. At this point you may be wondering what is Python, PIP and Pillow and why are they needed for CCDM?

**Python** is an interpreted, high level and general – purpose programming language. In python, the program itself is written in script files. These script files are then run by an interpreter that does the job of “running the code” files (IE processing the scripts) on your device. This interpreter is what you are going to install on your device in order to run python code (IE run

CCDM). Python was invented by Guido Van Rossum in the late '80s – early 90s. Since CCDM is written in python, you are required to install python on the device you want to run CCDM on.

**PIP** is the standard python package manager for the python environment. Python packages are like “add-ons” to extend what python can do. Anytime you want to install a package for python, you will be using a version of pip. So, what packages do we need to install in our python environment you may be asking? This leads us nicely into **Pillow**.

**Pillow** is a package of library files for use in python to work with images in python programs. Pillow is a fork (a divergent project) from **PIL** which stands for **Python Imaging Library**. PIL is now considered an out of date, unsupported project and Pillow has become what is considered the replacement (extension is a better word) of the original PIL project. PIL was originally authored by Fredrik Lundh in 1995. Starting around mid 2010, the Pillow project has been developed, from Lundh’s original work, by Alex Clark and various other contributors. CCDM uses the Pillow library to handle some images on it’s UI (user Interface). Therefore, like Python itself, Pillow must be installed on the device you want to run CCDM on.

So, let us get installing!

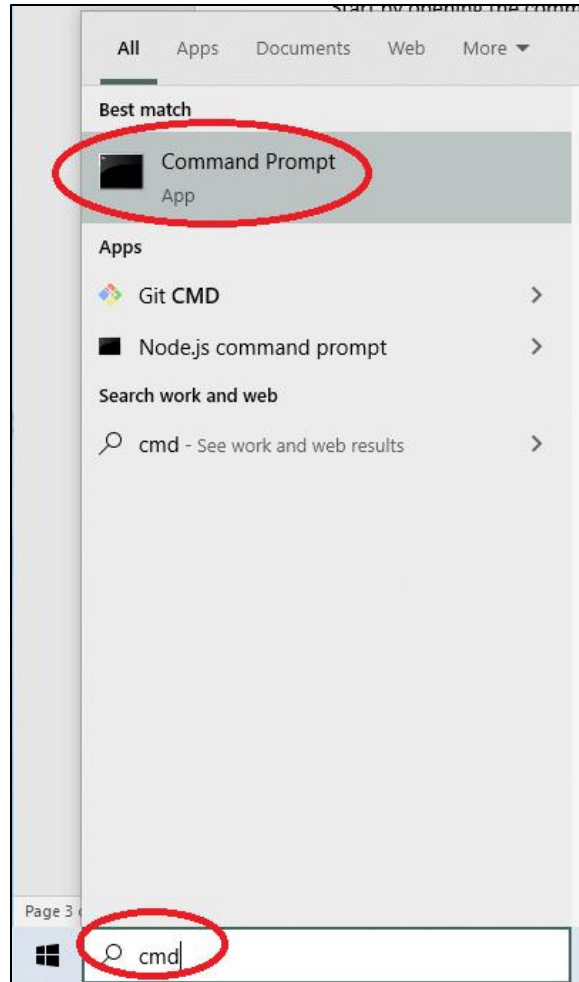
## Installing Python

CCDM is written in python, therefore the device you intend to run CCDM on must have python installed on it. Follow the appropriate section below for the operating system you’re working with.

**Note:** CCDM has been tested using Python Versions 3.4 through 3.9

### Win10:

Before we assume we need to install python, let’s check your device to see if a version of python version is already installed. Start by opening the command prompt, you can do so by entering “cmd” (with-out the quotes) in the start menu and then selecting the Command Prompt app, such as seen on this screen shot:



The command prompt window will open. At the command prompt, you can check to see if Python is already on your Windows device by entering “python3 - - version” (without the quotes) such as shown in this screen shot:

```
Command Prompt
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\katie>python3 --version
Python 3.9.1

C:\Users\katie>
```

In the above example we start off at C:\Users\katie>, your system will be different and that's ok. Also in the above example you can see where we've entered "python3 --version", and in the case above the system returned "Python 3.9.1". If your system returns any python version 3.4.x or above then python is all ready to go and you may proceed to the "Checking PIP" section. If your system returned nothing, an error or an older version of python, or if you'd like to upgrade, then you must install / upgrade python.

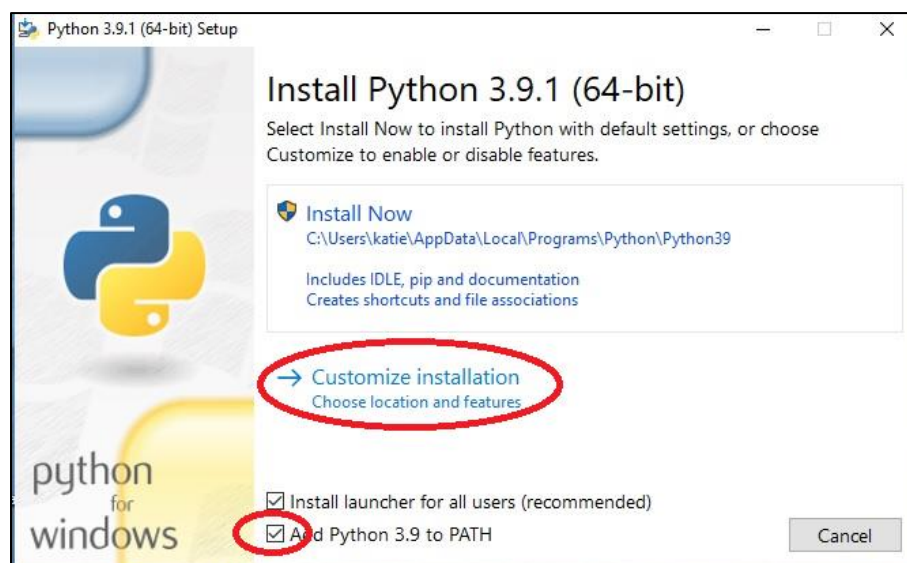
To install the latest version of Python for Windows (3.9.1 at time of writing), visit the following website and download proper installer for your OS:

<https://www.python.org/downloads/release/python-391/>

The recommended Windows installer is at the bottom of the webpage, select 32 or 64 bit depending on your requirements. Once the installer is downloaded (it will likely end up in your Downloads folder), launch it and follow the instructions to install python. Here are a few screen shots of 3.9.1 being installed to help you with your installation.

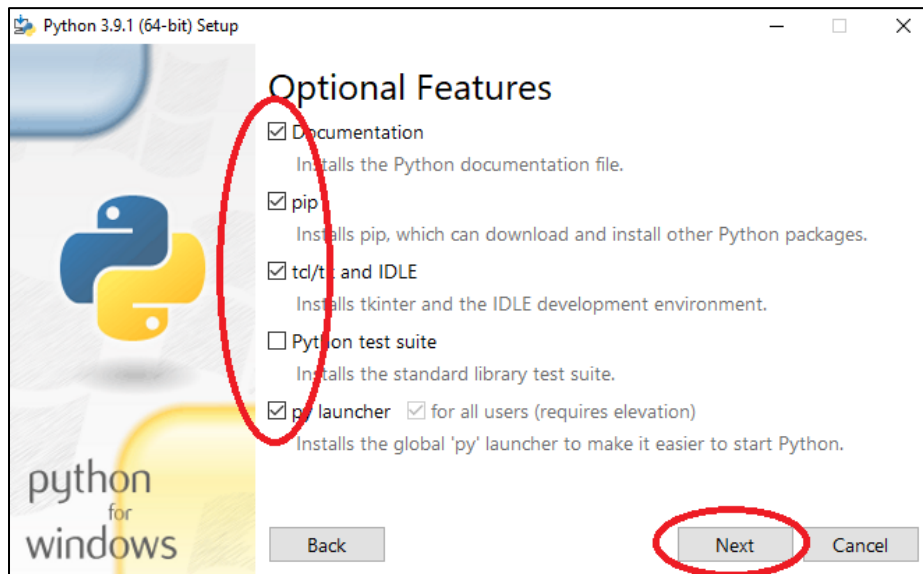
**NOTE:** It is suggested to NOT use the Windows Store to install python as you will not have control over some of the preferences when installing in this manner.

This is the initial dialog screen that will open when you launch the Python for windows installer. Be sure to select "Add Python 3.9 to PATH" and then click on "Customize installation".

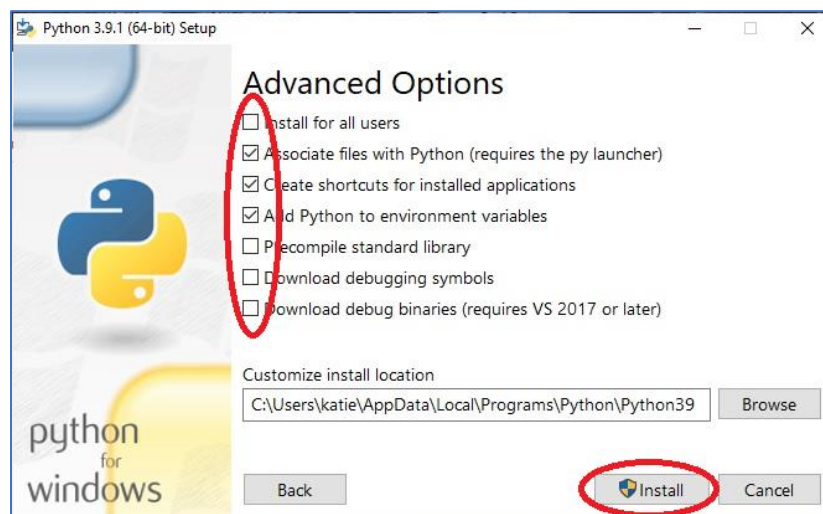


After clicking the Customize installation as above, an "Optional Features" dialog will open, select options as shown in the screen shot below (note the "for all users" is greyed in this

example but may not but greyed for you). Once you have selected the options, click the “Next” button.



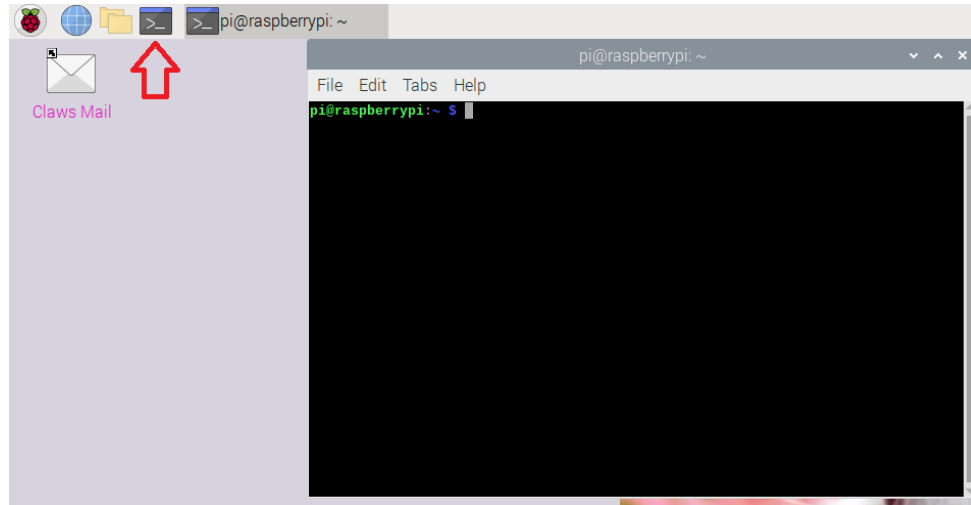
Finally, you are brought to the “Advanced Options” dialog. Select the check boxes as shown in the following screen shot, then click the “Install” button. Python will now install!



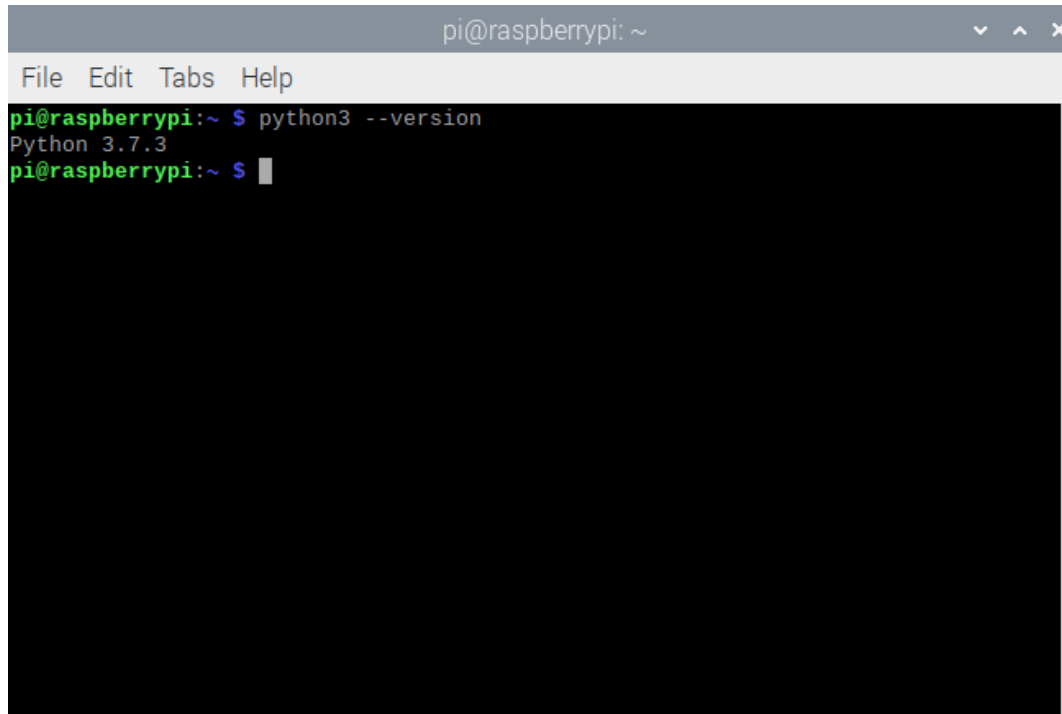
Once the installation of python is complete, it is a good idea to go back to the command prompt and perform a python version check again as done previously. This will ensure the installation was successful. If the command prompt check is successful, reward yourself with a tea or coffee. Then move onto Checking PIP.....

## Linux / Raspbian:

Typically, Linux distributions come with python already installed. However, we will check to confirm as well as see what version of python is installed. Start by opening Bash / Terminal as shown in this screen shot example:



Bash / Terminal window will open. You can check if python is installed and what version it is by entering “python3 - -version” (with-out quotes) as shown in this screen shot:



In the above example we start off at `pi@raspberrypi:~ $`, your system may be different here, that's ok. Also in the above example you can see where we've entered “python3 - - version”,

and in the case above the system returned “Python 3.7.3”. If your system returns any python version 3.4.x or above, then python is all ready to go and you may proceed to the Checking PIP section. If your system returned nothing, an error or an older version of python, then you must install / upgrade python.

To install the latest version of Python in Linux / Raspbian enter:

```
sudo apt update
```

Then:

```
sudo apt install python3
```

Once the installation of python is complete, it is a good idea to perform a python version check again as done previously. This will ensure the installation was successful. If the command prompt check is successful, reward yourself with a tea or coffee. Then move onto Checking PIP.....

## Checking PIP

In python, “extras” or “add-ons” (so to speak) are called packages. The standard python installation does not come with all possible “extras” installed by default. This keeps the python installation as small as possible. However, from time to time, we may need or want to add some extras to our python installation. To do so we use something called *pip*.

PIP is the standard package manager for the Python Environment. There are three versions of pip, the three versions are: *pip*, *pip2* and *pip3*. PIP2 and PIP3 are used to manage packages on Python 2.x.x versions and Python 3.x.x versions, respectively (pip2 for python 2 installs and pip3 for python 3 installs). PIP (not pip2 or pip3) will perform package management on both python 2.x.x and Python 3.x.x installations. Since CCDDM will be using Python 3.x.x we can use either PIP or PIP3 to manage packages.

As we did in the Installing Python section, let’s see what your device has before installing.

### Win10:

During the Windows PIP *should* have installed when you installed python itself. However, we should confirm this. To confirm pip is installed, open the windows command prompt again, then type “pip3 --version”. If pip3 is installed, you will get a version of pip3 returned. If you did not get this list, or got an error, then you need to repeat the [Installing Python](#) section and ensure pip is selected during the installation.



Linux / Raspbian:

To confirm pip is installed, open Bash / Terminal again, then type:

```
pip3 --version
```

If pip3 is installed, you will get a version of pip3 returned. If you get an error message, use the following commands in Bash / Terminal to install pip:

```
sudo apt update
```

Then:

```
sudo apt install python3-pip
```

After installing, it's a good idea to check the version again to confirm the install.

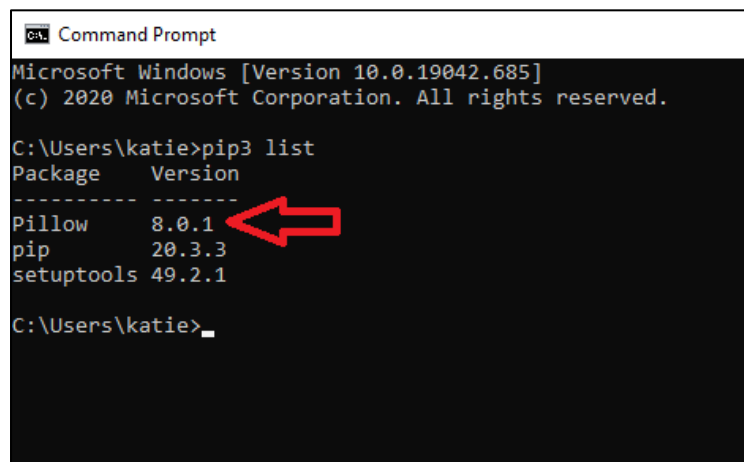
## Installing Pillow

As mentioned previously, Pillow is a package that can be added to Python for programmers to use as a set of tools to work with images. Think of Pillow as an “extra” or “add-on” that we are going to add to the standard python installation.

**Note:** CCDM has been tested using Pillow Versions 2.3.0 through 8.0.1

Win10:

If you'd like to check to see if the Pillow package is already installed on your Win10 device, you can enter the following into the command prompt:



```
Command Prompt
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\katie>pip3 list
Package      Version
-----
Pillow       8.0.1
pip          20.3.3
setuptools   49.2.1

C:\Users\katie>
```

As you can see Pillow shows up in the list of pip3 installed packages, the version is also listed. If Pillow did not show up in the list, you will have to install it. To do so enter the following at the command prompt. An installation process should start, and pillow will be installed for you. The following is a screen shot example:

```
Command Prompt
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\katie>pip3 install pillow
Collecting pillow
  Downloading Pillow-8.0.1-cp39-cp39-win_amd64.whl (2.1 MB)
    | 2.1 MB 544 kB/s
Installing collected packages: pillow
Successfully installed pillow-8.0.1

C:\Users\katie>
```

As always, after the installation it is a good idea to re-check the pip3 list to ensure it now shows up.

#### Linux / Rapsbian:

If you'd like to check to see if the Pillow Package is already installed on your Linux / Raspbian device, you can enter the following in Bash / Terminal:

```
Terminal
file_server@TRD-EMACLAP1 ~ $ pip3 list
apt-clone (0.2.1)
Brlapi (0.6.1)
chardet (2.2.1)
colorama (0.2.5)
command-not-found (0.3)
defer (1.0.6)
dirspec (13.10)
html5lib (0.999)
httplib2 (0.8)
language-selector (0.1)
louis (2.5.3)
lxml (3.3.3)
mugshot (0.2.5)
oauthlib (0.6.1)
oneconf (0.3.7.14.04.1)
pexpect (3.1)
Pillow (2.3.0)
pip (1.5.4)
piston-mini-client (0.7.5)
pycrypto (2.6.1)
pycurl (7.19.3)
pygobject (3.12.0)
PyICU (1.5)
python-apt (0.9.3.5ubuntu3)
python-debian (0.1.21-nmu2ubuntu2)
pyxdg (0.25)
```

As you can see Pillow shows up in the list of pip3 installed packages, also listed is the version. If Pillow did not show up in the list, you will have to install it. To do so follow these simple steps.

Again, in Bash / Terminal, enter:

```
python3 -m pip install --upgrade pip
```

Then:

```
python3 -m pip install --upgrade Pillow
```

After the installation, it is a good idea to re-check the pip3 list to ensure it now shows up.

## Placing CCDM onto your Device

After you downloaded and unzipped the CCDM package from our GitHub repository you have been left with a main folder called "Charge-Controller-Data-Manager-CCDM-master". In the folder are some contents such as other folder and files. Leaving all the contents as is, copy and paste the entire "Charge-Controller-Data-Manager-CCDM-master" folder to your device's desktop. You may rename the "Charge-Controller-Data-Manager-CCDM-master" folder to something shorter, like "CCDM" for example, but do not rename, move or delete any of the contents in this folder.

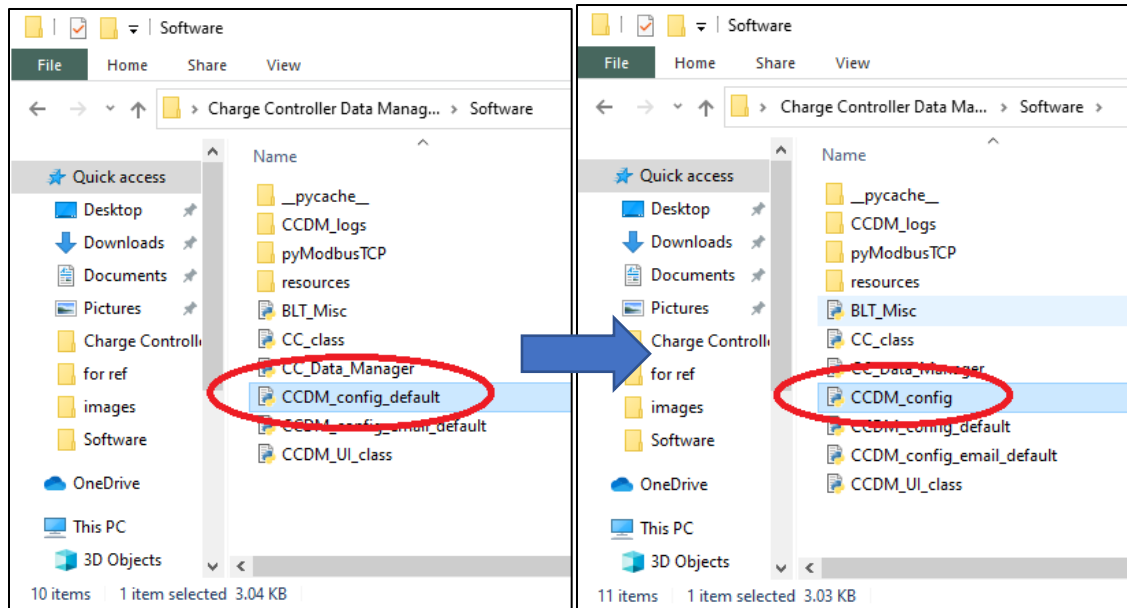
## Networking

As previously mentioned in the Initial Setup section, be sure your Charge Controller and what ever device you are running CCDM on (PC, Rasp Pi, etc.) are networked together and able to communicate. Use Ping or similar networking tools to ensure proper communication infrastructure is in place. Going into detail about networking is beyond the scope of this Getting Started Guide. Therefore, if you are having trouble networking your device and your charge controller you will need to consult the Charge Controller's documentation as well as other online resources regarding the topic of Networking.

## Initial Configuration of CCDM

In the CCDM Package, there will be a folder called "Software".

In this Software folder, you will find a file called "CCDM\_config\_default.py". CCDM is configured through this configuration file.



Make a copy of this file and rename it to “CCDM\_config.py” (or use a previously configured version of this file).

Open this CCDM\_config.py in any text editor (notepad as an example).

Find the line that looks like the following (starts with “CC\_IP\_ADDRS”):

```

CCDM_config_default - Notepad
File Edit Format View Help
##----- General Configuration Data file -----

CCDM_VERSION = '0.0.2 - beta'

##----- General Consts -----

#Set amount of Charge Controllers for CCDM to Poll
CC_COUNT = 1          #Note: Later to be read automatically based upon UDP annunciations
#Maximum Count (leave at 4)
MAX_CC_COUNT = 4

#Controller Names
CC_NAMES = ["Charge Controller 1", "Charge Controller 2", "Charge Controller 3", "Charge Controller 4"]

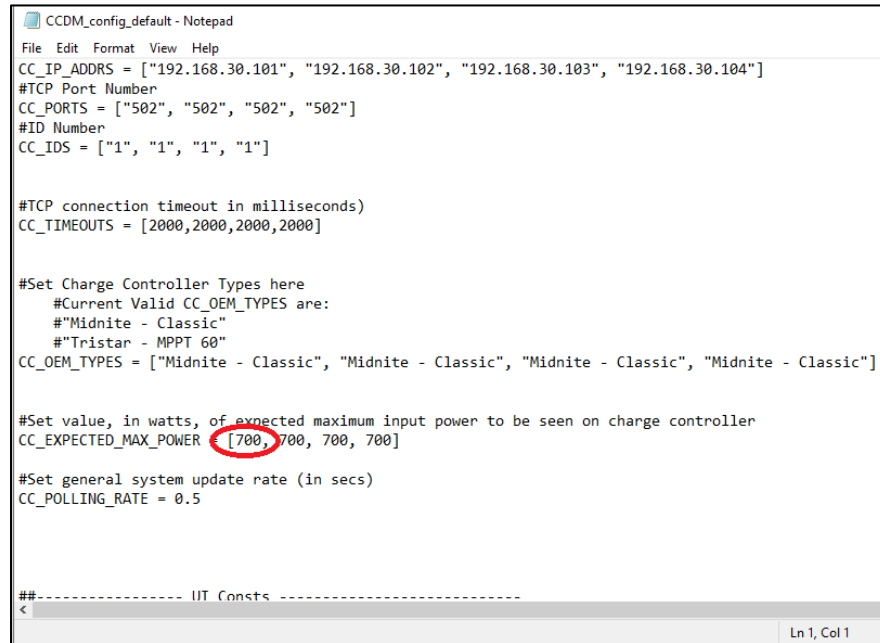
#TCP/IP Address
CC_IP_ADDRS = ["192.168.30.101", "192.168.30.102", "192.168.30.103", "192.168.30.104"]
#TCP Port Number
CC_PORTS = ["502", "502", "502", "502"]
#ID Number
CC_IDS = ["1", "1", "1", "1"]

#TCP connection timeout in milliseconds)
CC_TIMEOUTS = [2000,2000,2000,2000]

```

Change the first address (default is - "192.168.30.101") to whatever IP address your Charge Controller resides at on your network. **Be sure to leave the quotes in this case.** Leave the other three addresses alone.

Now scroll down to find a line that begins with "CC\_EXPECTED\_MAX\_POWER". Edit the first value to match your systems expected maximum input power in watts.



```
CCDM_config_default - Notepad
File Edit Format View Help
CC_IP_ADDRS = ["192.168.30.101", "192.168.30.102", "192.168.30.103", "192.168.30.104"]
#TCP Port Number
CC_PORTS = ["502", "502", "502", "502"]
#ID Number
CC_IDS = ["1", "1", "1", "1"]

#TCP connection timeout in milliseconds)
CC_TIMEOUTS = [2000,2000,2000,2000]

#Set Charge Controller Types here
#Current Valid CC_OEM_TYPES are:
#"Midnite - Classic"
#"Tristar - MPPT 60"
CC_OEM_TYPES = ["Midnite - Classic", "Midnite - Classic", "Midnite - Classic", "Midnite - Classic"]

#Set value, in watts, of expected maximum input power to be seen on charge controller
CC_EXPECTED_MAX_POWER = [700, 700, 700, 700]

#Set general system update rate (in secs)
CC_POLLING_RATE = 0.5

##----- IIT Consts -----
<
```

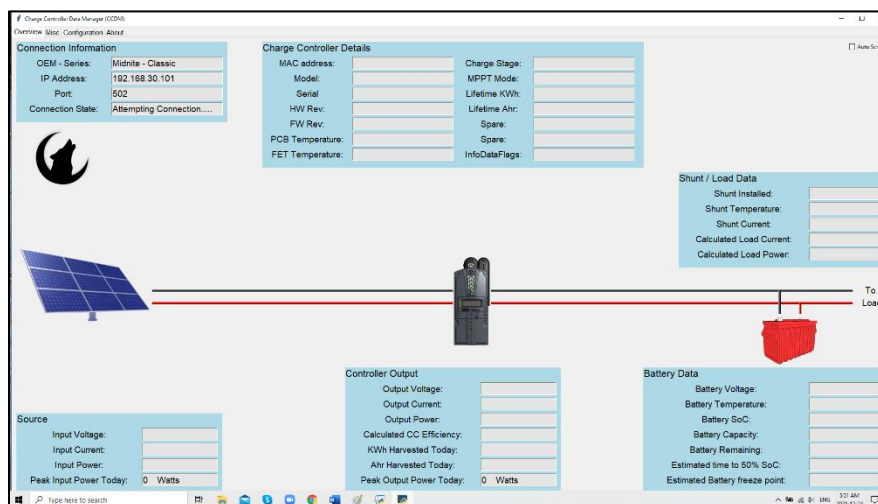
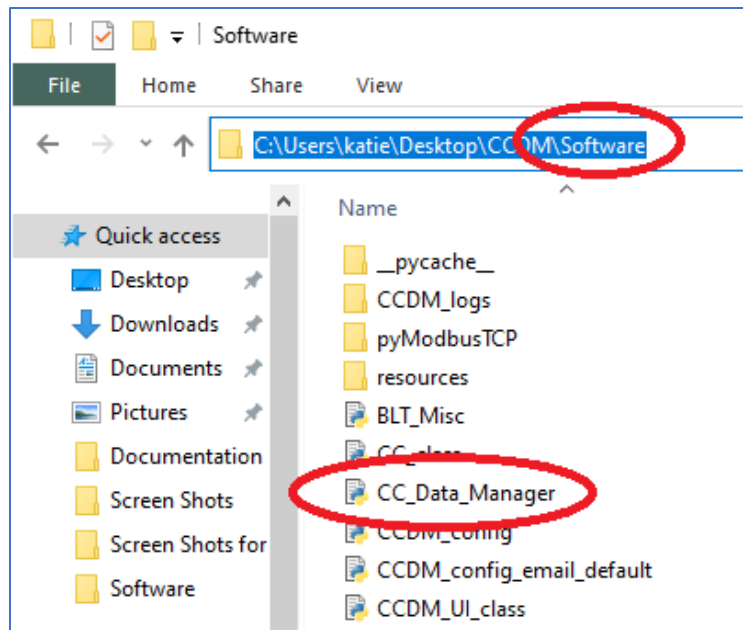
Save the file and then close the text editor.

## Start CCDM

If all the above went well, you should be ready to start CCDM!

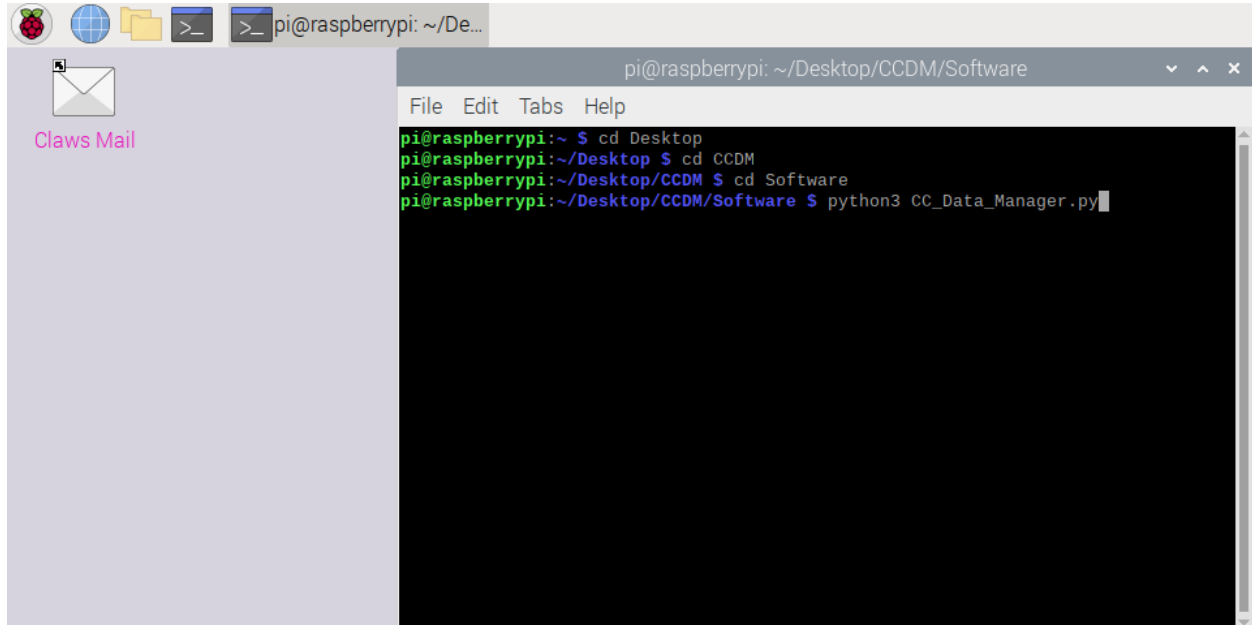
### Win10:

Again in the CCDM package, under the Software folder, locate a file called "CC\_Data\_Manager.py", double click it. CCDM should launch and start trying to read data from your charge controller via the IP address you configured in the previous step.



## Linux / Raspbian:

In Linux/Raspbian it is suggested to start CCDM via Bash / Terminal. Open Bash / Terminal and navigate to Desktop -> CCDM -> Software. Once navigated to the proper location, enter “python3 CC\_Data\_Manager.py” (with-out quotes):

A screenshot of a Raspberry Pi desktop environment. The desktop background is light purple. On the left side, there is a dock with icons for a Raspberry Pi logo, a globe, a folder, and a terminal icon. Below the dock, there is a pink envelope icon labeled "Claws Mail". In the center, a terminal window is open. The terminal title bar reads "pi@raspberrypi: ~/Desktop/CCDM/Software". The terminal content shows the following commands and their outputs:

```
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ cd CCDM
pi@raspberrypi:~/Desktop/CCDM $ cd Software
pi@raspberrypi:~/Desktop/CCDM/Software $ python3 CC_Data_Manager.py
```

## Finishing Up

Hopefully by following the above sections you were able to get CCDM off the ground and gathering data from your Charge Controller. If CCDM is running correctly, each evening at 11:59pm a daily log file (in text form) will be placed into the “CCDM\_logs” folder in the CCDM package.

From here, relax and enjoy watching your Charge Controller do its job. Also, have a look through the following Appendix sections for more information about CCDM.

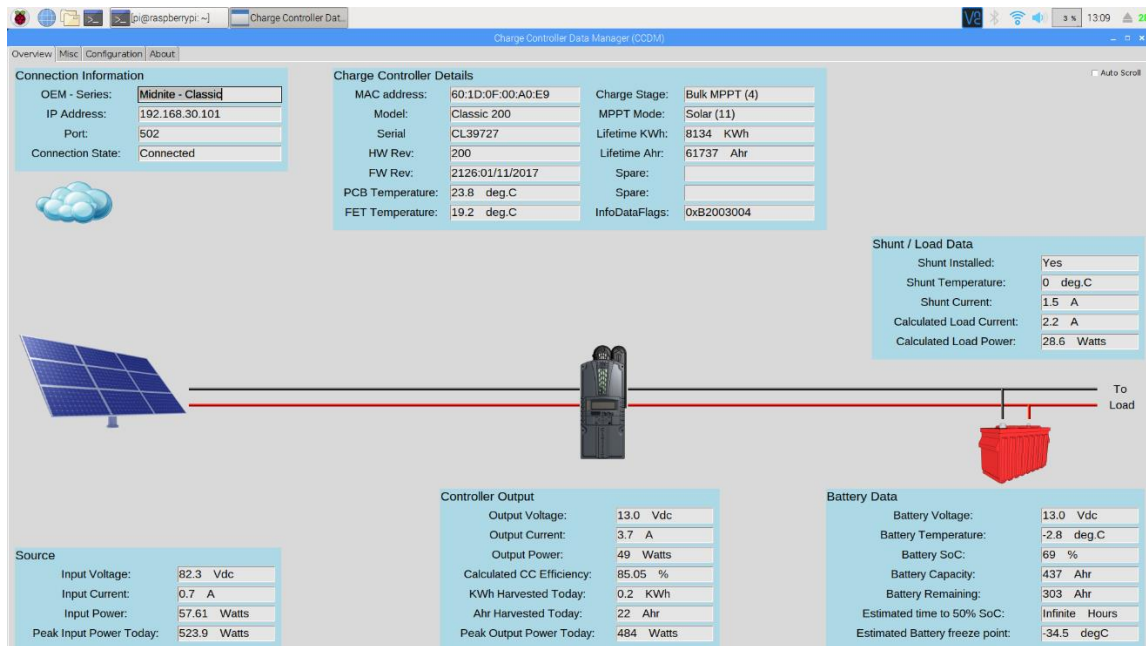
## Appendix

### CCDM UI Screens

Here is a quick explanation of what to expect when navigating through CCDM.

#### Overview Tab

The overview tab gives you just that, a general overview of your off-grid system.



In the top RH corner of this tab (and all other tabs) is an “Auto Scroll” check box. Selecting this check box enables an auto scrolling feature where CCDM will scroll through all of the Tabs automatically.

On this Overview tab you will also see a number of widgets, here is a break down of each widgets function:

**Connection Information Widget:** Shows some general information about the charge controller type, what IP address CCDM is trying to find the charge controller at, as well as the application port number. Finally, a status of the connection is displayed.

**Charge Controller Details Widget:** Displays some detailed information about your Charge Controller

**Source Widget:** This widget displays information about incoming energy. Above this widget is an image of the source type. If the Charge Controller’s MPPT mode is configured as solar or Legacy P&O the image will show a solar array.

**Controller Output Widget:** Shows some general data about what the Charge Controllers Output looks like, including a peak power output for the day. The Calculated CC Efficiency field is simply found by  $(\text{Output Power} / \text{Input Power}) * 100$ .

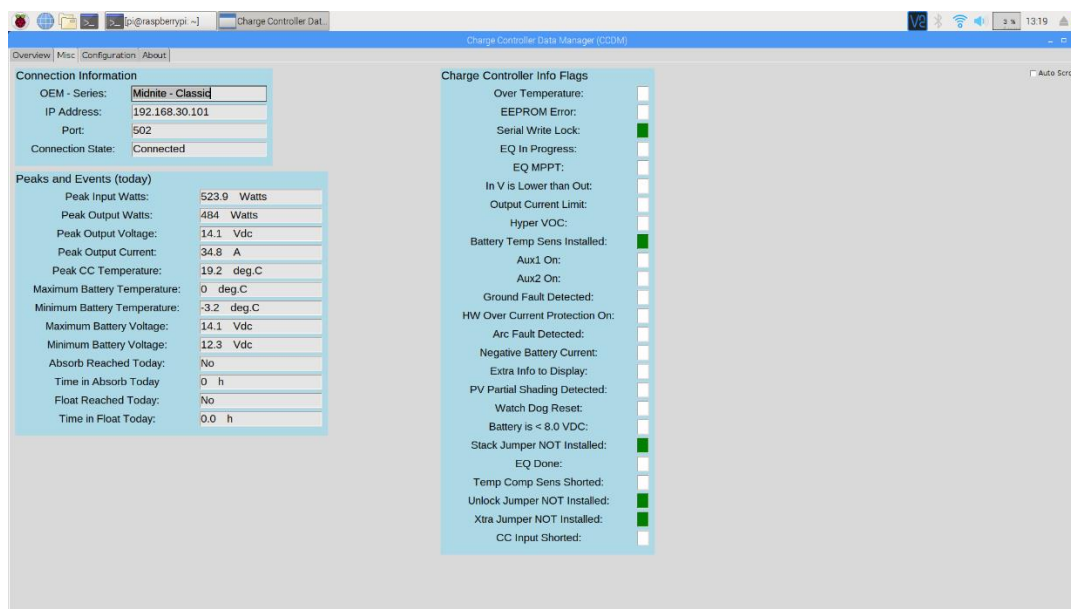


**Shunt / Load Data Widget:** This widget displays data regarding Shunt measurements and calculate load information. CCDM will automatically detect if your midnite classic system has a shunt installed. The outcome of this detection is shown in the “Shunt Installed” field. The “Shunt Installed” filed will show “No” and all other fields in this widget will simply display “NA”.

**Battery Data Widget:** Displays information about the battery. If no shunt is detected in the system then all fields except the “Battery Voltage” filed will display “NA”.

## Misc Tab

The Misc tab displays some peak values and events noticed during the day as well as gives a summary of various flags internal to the charge controller.



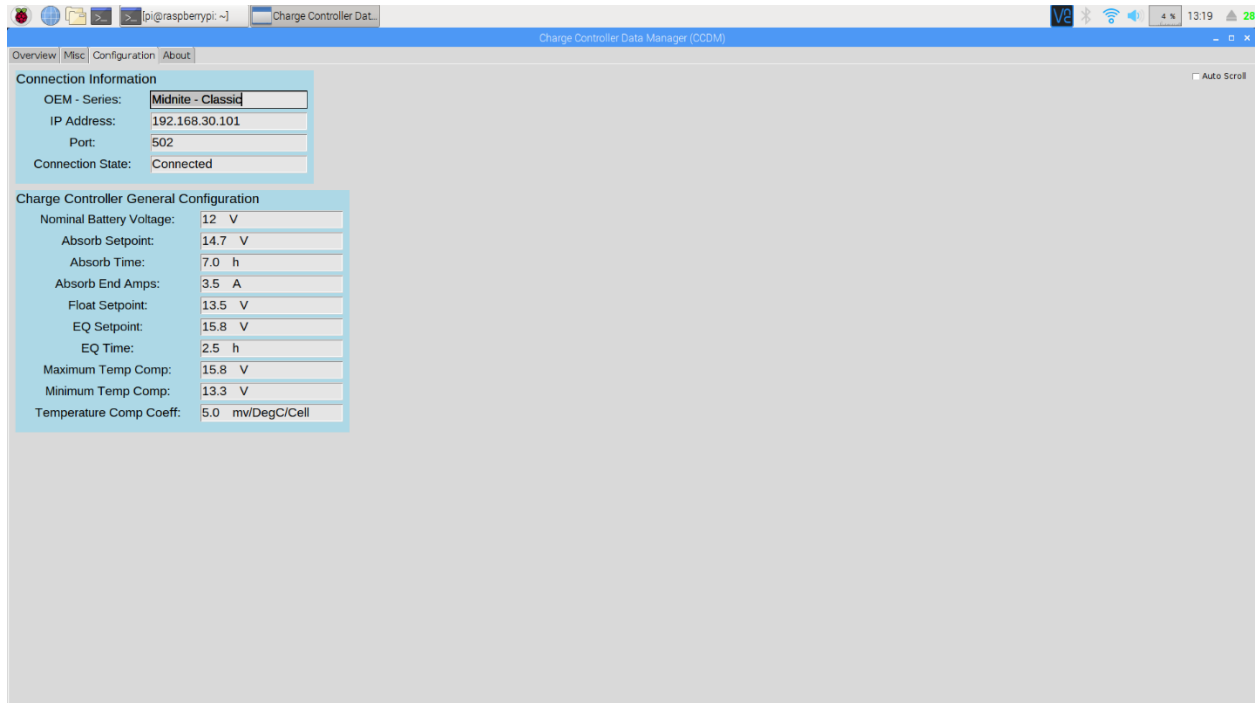
**Connection Information Widget:** Same as in the previous tab

**Peaks and Events (today) Widget:** Displays some peak values and other events that have happened during the day. These values are reset at 11:59pm or if CCDM is restarted (future versions will retain this data during a restart).

**Charge Controller Info Flags Widget:** Displays the status of the Info Flags contained in the Charge Controller, refer to the Charge Controllers documentation for further information about what each flag means.

## Config Tab

The Config tab shows you a summary of how your Charge Controller is configured. Future versions will display some information on how CCDM itself is configured (based on CCDM\_config.py file).



**Connection Information Widget:** Same as in the previous tabs

**Charge Controller General Configuration Widget:** Displays general data of how your Charge Controller is configured.