

Extracted from:

# Portable Python Projects

Run Your Home on a Raspberry Pi

This PDF file contains pages extracted from *Portable Python Projects*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit <http://www.pragprog.com>.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printed versions; the content is otherwise identical.

Copyright © 2022 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

The Pragmatic Bookshelf

Raleigh, North Carolina

# Run Your Home on a Raspberry Pi





# Portable Python Projects

Run Your Home on a Raspberry Pi

Mike Riley

The Pragmatic Bookshelf

Raleigh, North Carolina



Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit <https://pragprog.com>.

The team that produced this book includes:

CEO: Dave Rankin

COO: Janet Furlow

Managing Editor: Tammy Coron

Development Editor: Jacquelyn Carter

Copy Editor: L. Sakhi MacMillan

Indexing: Potomac Indexing, LLC

Layout: Gilson Graphics

Founders: Andy Hunt and Dave Thomas

For sales, volume licensing, and support, please contact [support@pragprog.com](mailto:support@pragprog.com).

For international rights, please contact [rights@pragprog.com](mailto:rights@pragprog.com).

Copyright © 2022 The Pragmatic Programmers, LLC.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

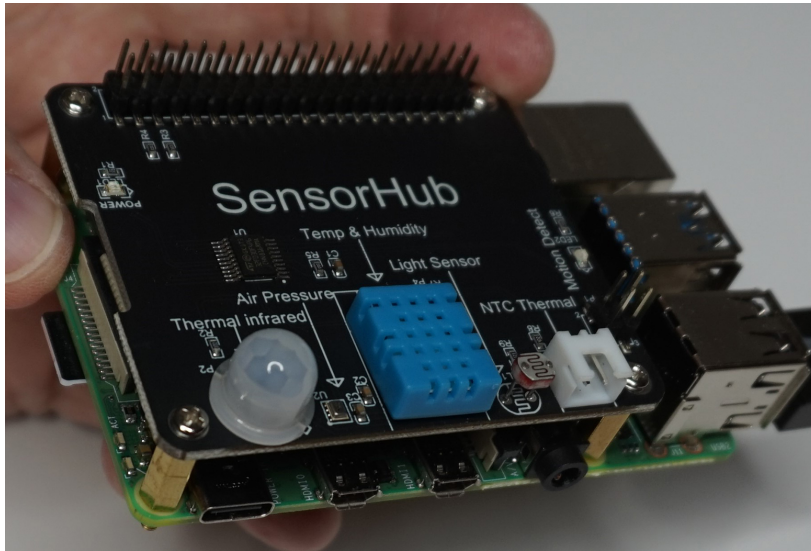
ISBN-13: 978-1-68050-859-8

Encoded using the finest acid-free high-entropy binary digits.

Book version: P1.0—February 2022

## Seating the Sensor

The SensorHub package includes four metal posts that can help secure the SensorHub more permanently to the Pi. I suggest you test your SensorHub attached to the Pi's GPIO pins first before securing it with the posts. When you're satisfied with the long-term use of the SensorHub, you can make the installation more secure by screwing the posts in place. Look at how much more secure the SensorHub sits on my Pi 4 in the next photo.



When you power on the Pi with the SensorHub correctly attached, the SensorHub's red LED power indicator located in the upper-left quadrant on top of the SensorHub will remain constantly illuminated. You may also see a blue LED near the middle-right edge of the SensorHub occasionally turn on and off. This LED lights up whenever motion is detected within the last five seconds. You can verify this by waving your hand over the board to watch this detection occur.

Since we'll be using the Philips Hue platform to turn on and off a Hue smart plug connected to a fan (or other appliance that may be better suited for your needs), we'll be calling upon the `phue` third-party Python library. Install the library into your virtual environment:

```
$ sudo pip3 install phue
```

Before we can start sending commands to the Hue bridge, we have to register it first.

## Bridge Registration

When you installed the Hue system, you needed to press the sync button on the top of the Hue bridge for your Hue smartphone app to talk to the unit. Philips has adopted a basic means of security to prevent anyone nearby from gaining unauthorized access to your Hue network. By pressing the button on the Hue bridge, you are authorizing the application requesting control access.

To do this for Python scripts running on the Pi, you first need to register with the bridge by connecting to it within thirty seconds after the Hue bridge button has been pressed. You'll also need to know the IP address of the Hue bridge on your network, which you should be able to identify from your network router's dashboard, similar to the way the Pi's IP and MAC address were identified back in [Finding the Pi on Your Network, on page ?](#).

Create the following script in your huefan directory, name it register.py, and add the lines of Hue bridge registration code to this file:

```
huefan/register.py
1 from phue import Bridge
2 huebridge = Bridge('YOUR_HUE_BRIDGE_IP_GOES_HERE')
3 huebridge.connect()
```

Let's take a quick look at what this three-line Python script does.

- ❶ Using the from Python keyword, we can import a specific class or function from a module and assign it to an instance variable. In this case, we import Bridge from the phue module for use in our script.
- ❷ We pass the IP address of our Hue bridge to Bridge and assign its result to the huebridge variable.
- ❸ As long as the button on the top of the Hue bridge was pressed a few seconds before this script is run, the Hue bridge will register our Pi and grant it the ability to control other Hue-registered devices on the network. Once you've successfully registered your Pi with the Hue bridge, there's no reason to call the connect() again unless you use a different Hue bridge or another Pi to run your Hue scripts on.

Replace the YOUR\_HUE\_BRIDGE\_IP\_GOES\_HERE with the IP address of your Hue bridge. When you're ready, press the sync button on top of your Hue bridge and to allow it to automatically register new devices asking to register with it. In our case, this new device is our Pi running this script.

Execute the `register.py` script within thirty seconds of pressing the Hue bridge button. If you fail to run the script in time, you will have to press the Hue bridge button and run the registration script again.

```
$ python3 register.py
```

If no output appears after you execute the script, then you successfully registered your Pi with the Hue bridge. You can now use your Pi and the `phue` scripts you run on it to remotely control other Hue devices on your network.

If the script failed to connect(), it will provide details in the output errors. The most frequent error for not being able to register is using an incorrect IP address for the Hue bridge. This error is usually reported at the bottom of the error trace as a `socket.gaierror: [Errno -2] Name or service not known`. Fix the IP address and connectivity problem so that you can successfully register your Pi with the Hue bridge before proceeding.

## Registered Devices

To confirm your effort was successful as well as identify the name and ID of the connected Hue smart plug, create and run the following `listdevices.py` script that will connect to your Hue bridge and enumerate a list of Hue accessories configured with the bridge:

```
huefan/listdevices.py
```

```
from phue import Bridge
```

```
1 huebridge = Bridge('YOUR_HUE_BRIDGE_IP_GOES_HERE')
2 lights = huebridge.lights
3 for light in lights:
    print(light.name)
```

Let's take a quick look at how this script connects to your Hue bridge and identifies the other Hue devices it's able to manage.

- ❶ Here we pass the IP address of our Hue bridge into the `huebridge` object instance.
- ❷ Next, we interrogate the bridge lights collection and assign those details about the Hue smart bulbs, smart plugs, and other Hue-controllable accessories to the `lights` variable.
- ❸ Finally, we use a for loop to iterate over the lights collection and assign each item to the `light` variable. Then we print the name we assigned each of those devices via the Hue smartphone app.



The output of the script will list the names of the Hue lights, smart plugs, and accessories you have registered and named with your Hue bridge. Identify the name of your Hue smart plug from the list. In my case, I named my smart plug “Fan”, a name I will use in the sample code. If you labeled your smart plug or light a different name using the Philips Hue app on your smartphone, be sure to replace 'Fan' in the code listings with the name of your Hue-labeled device. For example, here’s the output shown after running the `python3 listdevices.py` command from within my Terminal window.

A screenshot of a terminal window titled 'pi@pi: ~/projects/huefan'. The window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The command 'python3 listdevices.py' has been executed, resulting in a list of connected devices: 'Mike's Bedstand', 'Dresser', 'Family Room', 'Window', 'Fan', 'Shelf', 'Kitchen', 'Cabinet', and 'TV Light'. The prompt 'pi@pi:~/projects/huefan \$' is visible at the bottom of the output.

```
pi@pi: ~/projects/huefan
File Edit Tabs Help
pi@pi:~/projects/huefan $ python3 listdevices.py
Mike's Bedstand
Dresser
Family Room
Window
Fan
Shelf
Kitchen
Cabinet
TV Light
pi@pi:~/projects/huefan $
```

Now that we see our list of connected devices, we can turn them on or off using a single line of Python code.