Lab 1: Debugging a Raspberry Pi Internet of Things Flask Application

| Author	**Dave Glover, Microsoft Cloud Developer Advocate**
Platforms	Linux, macOS, Windows, Raspbian Buster
Tools	Visual Studio Code Insiders Edition
Language	Python
Date	As of August, 2019

Follow me on Twitter @dglover

PDF Lab Guide

You may find it easier to download and follow the PDF version of the Debugging Raspberry Pi Internet of Things Flask App hands-on lab guide.

Introduction

In this hands-on lab, you will learn how to create and debug a Python web application on a Raspberry Pi with Visual Studio Code and the Remote SSH extension. The web app will read the temperature, humidity, and air pressure telemetry from a BME280 sensor connected to the Raspberry Pi.
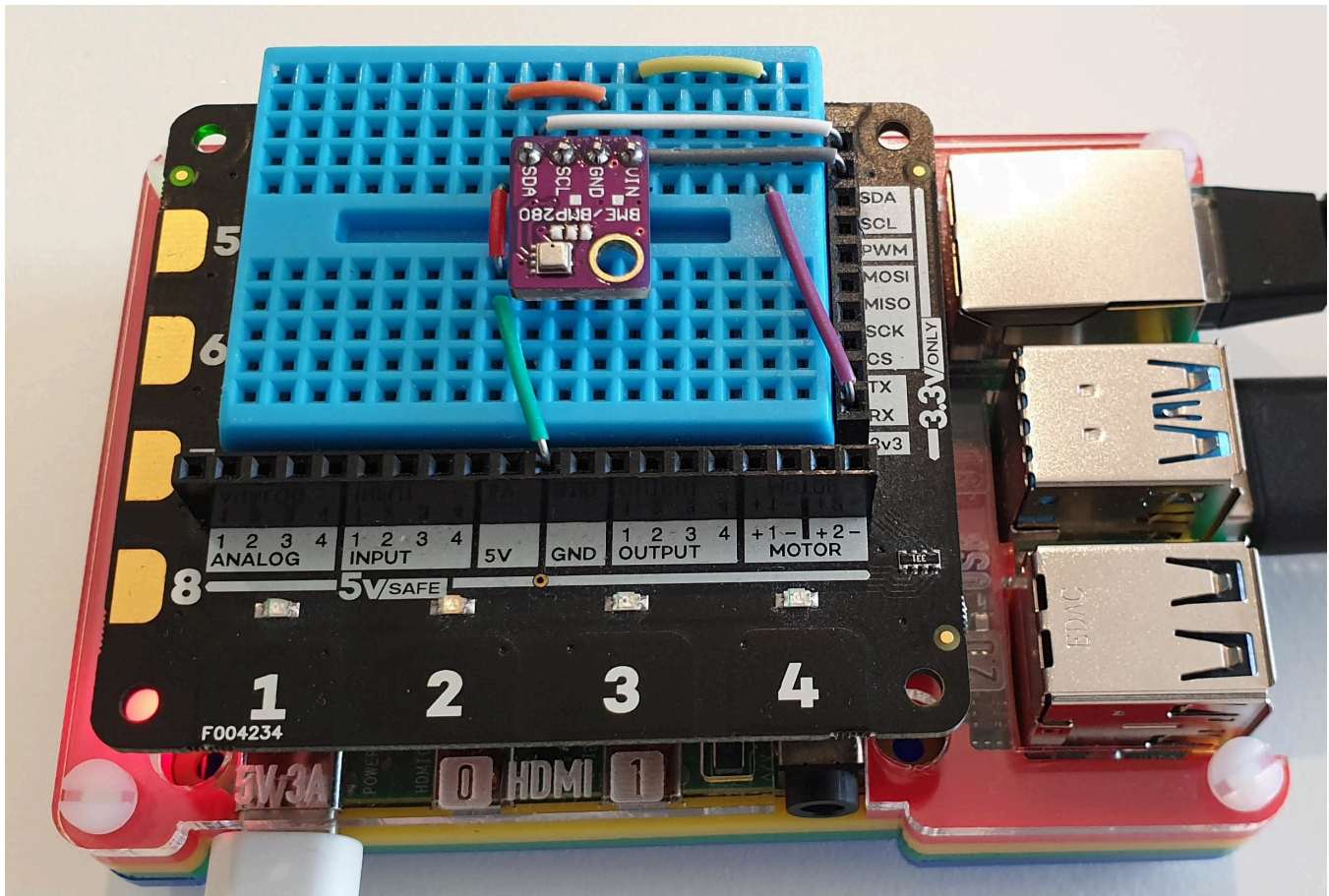
Remote Development using SSH

The Visual Studio Code Remote - SSH extension allows you to open a remote folder on any remote machine, virtual machine, or container with a running SSH server and take full advantage of Visual Studio Code's feature set. Once connected to a server, you can interact with files and folders anywhere on the remote filesystem.

No source code needs to be on your local machine to gain these benefits since the extension runs commands and other extensions directly on the remote machine.
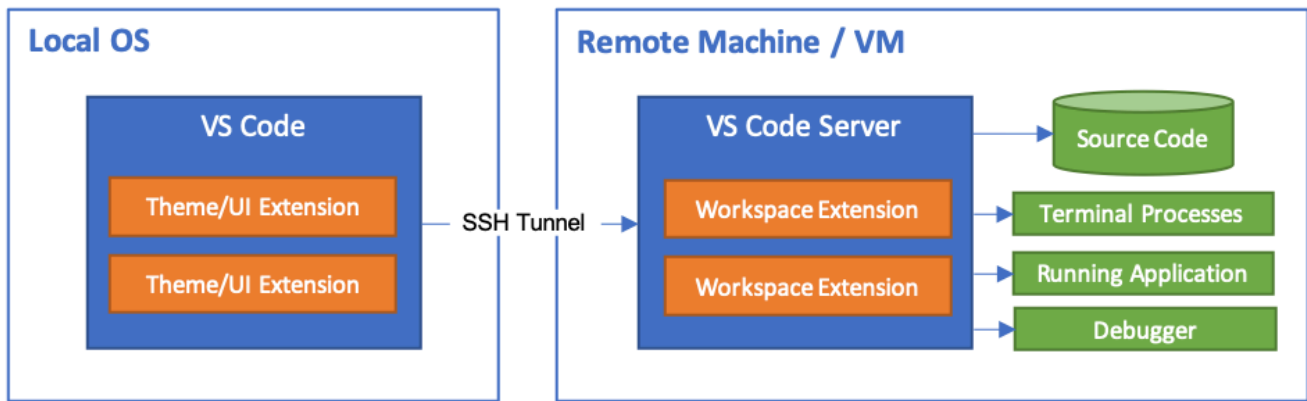
CircuitPython

[CircuitPython](#), an [Adafruit](#) initiative, is built on the amazing work of Damien George and the MicroPython community. CircuitPython adds hardware support for Microcontroller development and simplifies some aspects of MicroPython. MicroPython and by extension, CircuitPython implements version 3 of the Python language reference. So, your Python 3 skills are transferrable.

CircuitPython runs on over 60 **Microcontrollers** as well as the **Raspberry Pi**. This means you build applications that access GPIO hardware on a Raspberry Pi using CircuitPython libraries.

The advantage of running CircuitPython on the Raspberry Pi is that there are powerful Python debugging tools available. If you have ever tried debugging applications on a Microcontroller, then you will appreciate it can be painfully complex and slow. You resort to print statements, toggling the state of LEDs, and worst case, using specialized hardware.

With Raspberry Pi and CircuitPython, you build and debug on the Raspberry Pi, when it is all

working you transfer the app to a CircuitPython Microcontroller. You need to ensure any libraries used are copied to the Microcontroller, and pin mappings are correct. But much much simpler!

This hands-on lab uses CircuitPython libraries for GPIO, I2C, and the BME280 Temperature/Pressure/Humidity sensor. The CircuitPython libraries are installed on the Raspberry Pi with pip3.

```
pip3 install adafruit-blinka adafruit-circuitpython-bme280
```

Software Installation

This hands-on lab uses Visual Studio Code. Visual Studio Code is a code editor and is one of the most popular **Open Source** projects on GitHub. It runs on Linux, macOS, and Windows.

1. Visual Studio Code Insiders Edition
   As at August 2019, **Visual Studio Code Insiders Edition** is required as it has early support for Raspberry Pi and Remote Development over SSH.
2. Remote - SSH Visual Studio Code Extension

For information on contributing or submitting issues see the Visual Studio GitHub Repository. Visual Studio Code documentation is also Open Source, and you can contribute or submit issues from the Visual Studio Documentation GitHub Repository.

Raspberry Pi Hardware

If you are attending a workshop, then you can use a shared network-connected Raspberry Pi. You can also use your own network-connected Raspberry Pi for this hands-on lab.

You will need the following information from the lab instructor.

1. The **Network IP Address** of the Raspberry Pi
2. Your assigned **login name** and **password**.

SSH Authentication with private/public keys

Setting up a public/private key pair for SSH authentication is a secure and fast way to authenticate from your computer to the Raspberry Pi. This is needed for this hands-on lab.

SSH for Linux and macOS

From a Linux or macOS **Terminal Console** run the following commands:

1. Create your key. This is typically a one-time operation. **Take the default options**.

   ```
   ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa_python_lab
   ```

2. Copy the public key to the Raspberry Pi.

   ```
   ssh-copy-id -i ~/.ssh/id_rsa_python_lab <login@Raspberry IP Address>
   ```

   For example:

   ```
   ssh-copy-id -i ~/.ssh/id_rsa_python_lab dev99@192.168.1.200
   ```

3. Test the SSH Authentication Key

   ```
   ssh -i ~/.ssh/id_rsa_python_lab <login@Raspberry IP Address>
   ```

   A new SSH session will start. You should now be connected to the Raspberry Pi **without** being prompted for the password.

4. Close the SSH session. In the SSH terminal, type exit, followed by ENTER.

SSH for Windows

Install the **OpenSSH Client for Windows**

Windows 10 (1809+) Users

From PowerShell as Administrator

```
Add-WindowsCapability -Online -Name OpenSSH.Client
```

From Windows Settings

1. To install OpenSSH, start from **Settings** then go to **Apps** > **Apps and Features** > **Manage Optional Features**.
2. Scan this list to see if OpenSSH client is already installed. If not, then at the top of the page select **Add a feature**, then:
3. To install the OpenSSH client, locate "OpenSSH Client", then click "Install".
4. Once the installation completes, return to Apps > Apps and Features > Manage Optional Features and you should see the OpenSSH component(s) listed.

Windows 7 or older Windows 10 Systems

For earlier Windows, install Git for Windows.

Create an SSH Key from Windows

From a Windows **Command Prompt** run the following commands:

1. Create your key. This is typically a one-time operation. **Take the default options**.

   ```
   ssh-keygen -t rsa -b 4096 -f %USERPROFILE%\.ssh\id_rsa_python_lab
   ```

2. Copy the public key to your Raspberry Pi

```
SET REMOTEHOST=<login@Raspberry IP Address>

SET PATHOFIDENTITYFILE=%USERPROFILE%\.ssh\id_rsa_python_lab

scp %PATHOFIDENTITYFILE% %REMOTEHOST%:~/tmp.pub

ssh %REMOTEHOST% "mkdir -p ~/.ssh && chmod 700 ~/.ssh && ^
cat ~/tmp.pub >> ~/.ssh/authorized_keys && ^
chmod 600 ~/.ssh/authorized_keys && rm -f ~/tmp.pub"
```

3. Test the SSH Authentication Key

```
ssh -i %USERPROFILE%\.ssh\id_rsa_python_lab <login@Raspberry IP Address>
```

A new SSH session will start. You should now be connected to the Raspberry Pi **without** being prompted for the password.

4. Close the SSH session. In the SSH terminal, type **exit**, followed by **ENTER**.

Trouble Shooting SSH Client Installation

- Remote Development using SSH
- Installing a supported SSH client

Configure Visual Studio Code Remote SSH Development

We need to tell Visual Studio Code the IP Address and login name we will be using to connect to the Raspberry Pi.
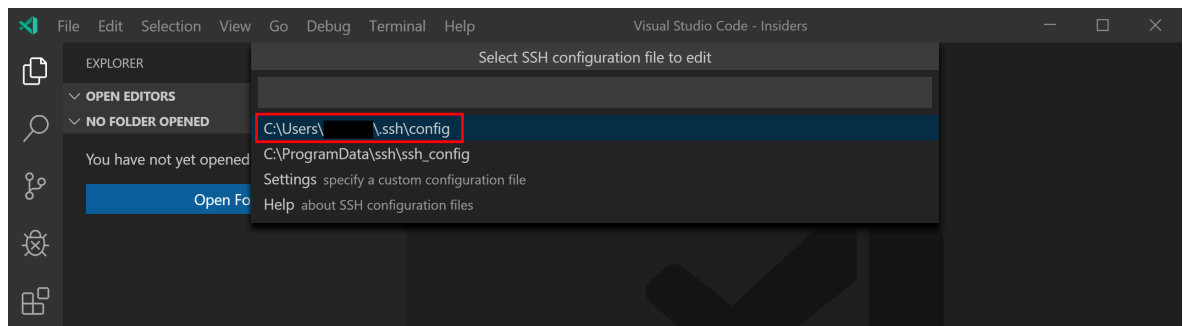
1. Start Visual Studio Code Insiders Edition
2. Click the **Open Remote Windows** button. You will find this button in the bottom left-hand corner of the Visual Studio Code window.
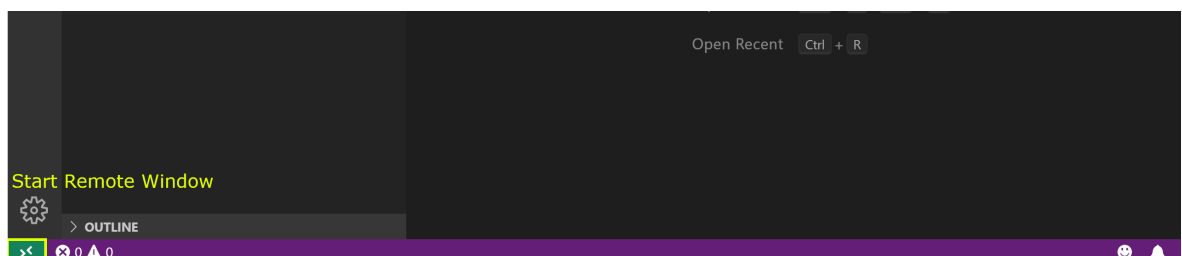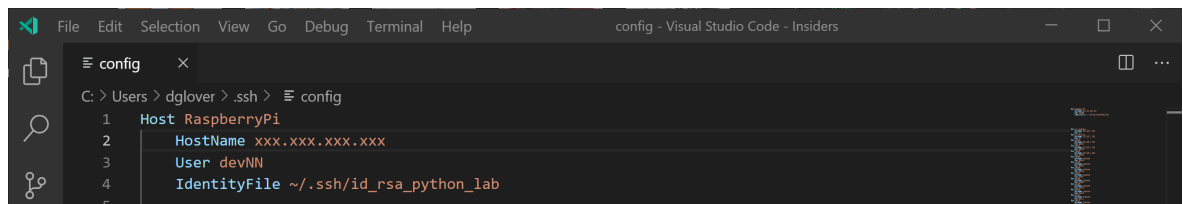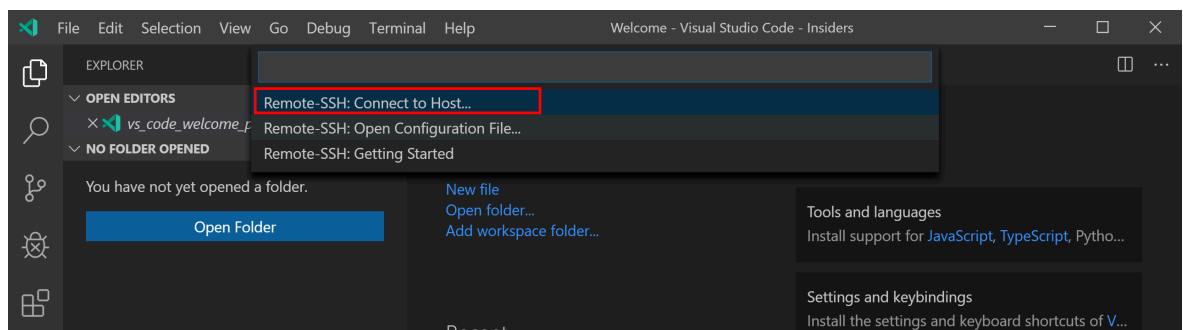

3. Select **Open Configuration File**
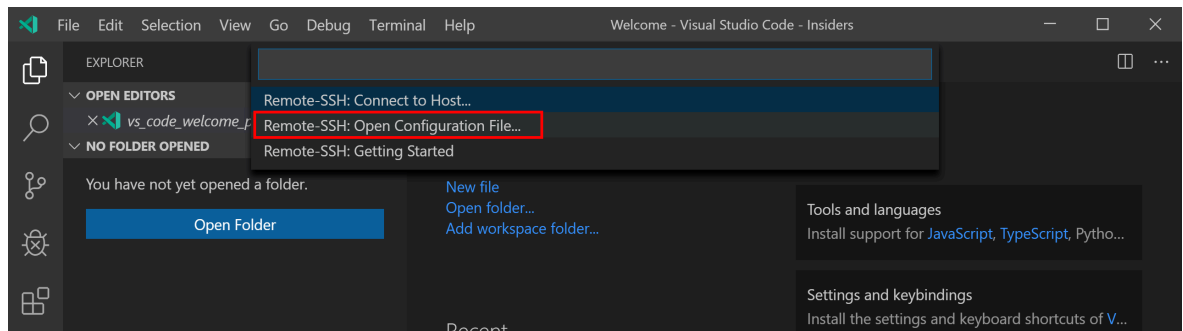
4. Select the user .ssh config file

5. Set the SSH connection configuration. You will need the Raspberry Pi **IP Address**, the Raspberry Pi **login name**, and finally set the **IdentityFile** field to **~/.ssh/ id_rsa_python_lab**. Save these changes (Ctrl+S).

6. Click the Open Remote Windows button (bottom left) then select **Remote SSH: Connect to Host**
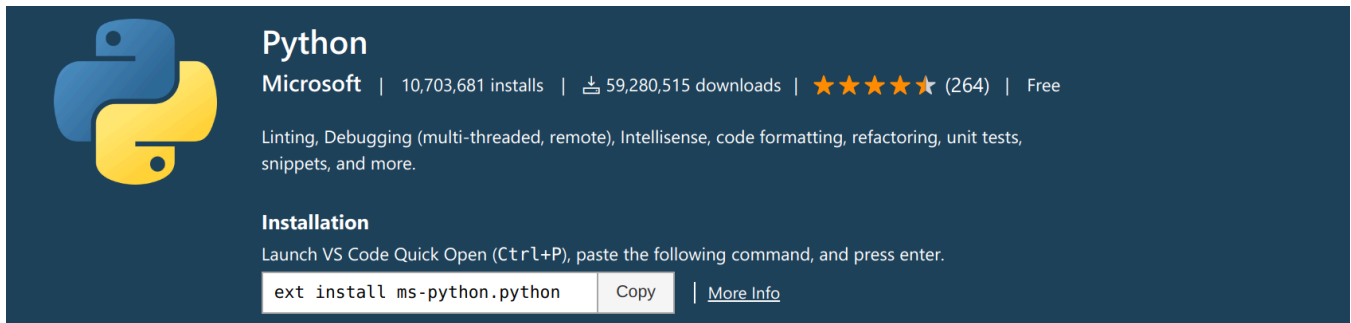

7. Select the host **RaspberryPi** configuration
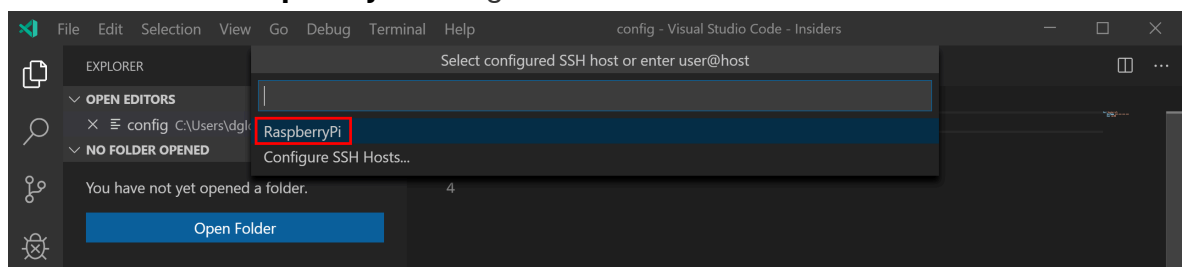
It will take a moment to connect to the Raspberry Pi.

Install the Python Visual Studio Code Extension

Launch Visual Studio Code Quick Open (Ctrl+P), paste the following command, and press enter:

```
ext install ms-python.python
```

See the Python Extension page for information about using the extension.

# Open the Lab 1 SSH Debug Project

From **Visual Studio Code**, select **File** from the main menu, then **Open Folder**. Navigate to and open the **github/Lab1-ssh-debug** folder.

1. From Visual Studio Code: File -> Open Folder
2. Navigate to **github/Lab1-ssh-debug** directory
3. Open the **app.py** file and review the contents
4. Set a breakpoint at the first line of code in the **show_telemetry** function (**now = datetime.now()**) by doing any one of the following:
   ◦ With the cursor on that line, press F9, or,
   ◦ With the cursor on that line, select the Debug > Toggle Breakpoint menu command, or, click directly in the margin to the left of the line number (a faded red dot appears when hovering there). The breakpoint appears as a red dot in the left margin:
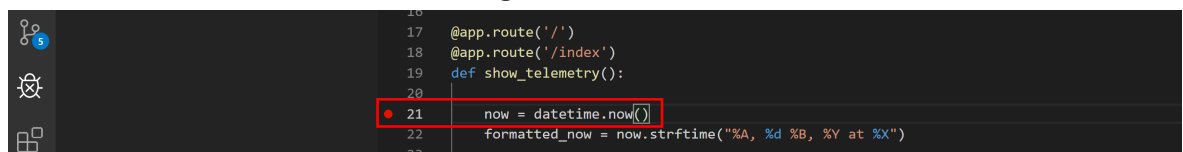
5. Review the **debug** options.
    1. Switch to Debug view in Visual Studio Code (using the left-side activity bar).
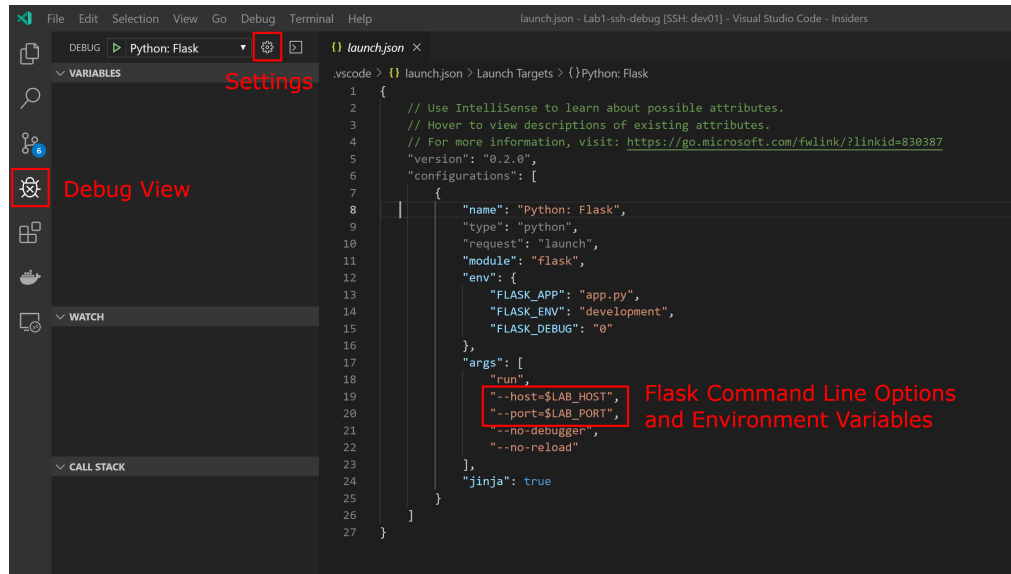
    2. Click the **Settings** button which will open the **launch.json** file.
    3. The **launch.json** file defines how the Flask app will be started, and what Flask Command Line parameters will be passed on at startup.

       There are two environment variables used in the launch.json file. These are **LAB_HOST** (which is the IP Address of the Raspberry Pi), and **LAB_PORT** (a random TCP/IP Port number between 5000 and 8000). These environment variables are set by the .bashrc script which runs when you connect to the Raspberry Pi with Visual Studio Remote SSH.

6. Press F5 (or click the Run icon) to launch the **Python: Flask** debug configuration. This will start the Web Application on the Raspberry Pi in debug mode.
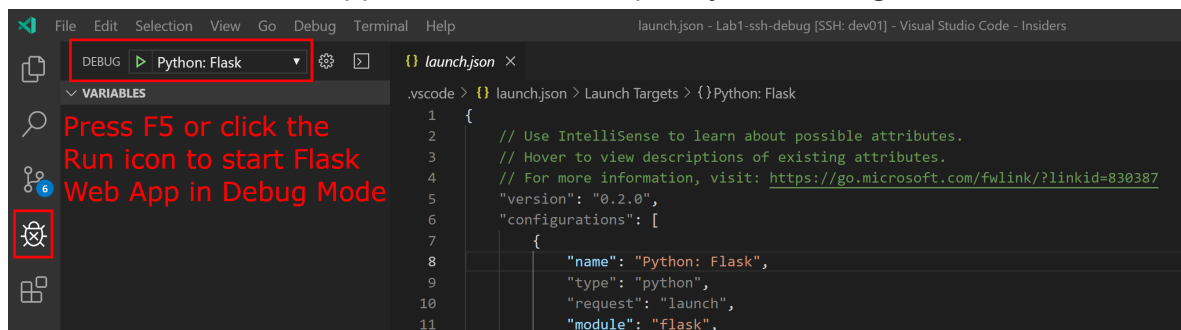
7. Ctrl+click the Flask Web link in the Visual Studio Terminal Window. This will launch your desktop Web Browser.

```
24          html = render_template('index.html', title=title,
25                                  temperature=temperature, pressure=pressure,
26                                  humidity=humidity)
27
28          return html
29
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL                                          2: Python Debug Conso ▼

ocess/attach_linux_x86.so: cannot open shared object file: No such file or directory
 * Serving Flask app "app.py"
 * Environment: development
 * Debug mode: off
 * Running on http://192.168.1.3:7346/ (Press CTRL+C to quit)
```

⊗ 0 ⚠ 0   ▶ Python: Flask (Lab1-ssh-debug)   Ln 15, Col 32   Spaces: 4   UTF-8

8. Next switch back to Visual Studio Code. The code execution has stopped at the breakpoint you set.

Debug actions

Once a debug session starts, the **Debug toolbar** will appear at the top of the editor window.

A debugging toolbar (shown above) will appear in Visual Studio Code. It has the following options:

1. Pause (or Continue, F5),
2. Step Over (F10)
3. Step Into (F11),
4. Step Out (Shift+F11),
5. Restart (Ctrl+Shift+F5),
6. and Stop (Shift+F5).

Using the Debugger

Next, we are going to **Step Into** (F11) the code using the debugging toolbox. Observe the debugger steps into the **Telemetry** class and calls the **render_telemetry** method.

Variable Explorer

1. As you step through the **render_telemetry** method you will notice that Python variables are displayed in the **Variables Window**.
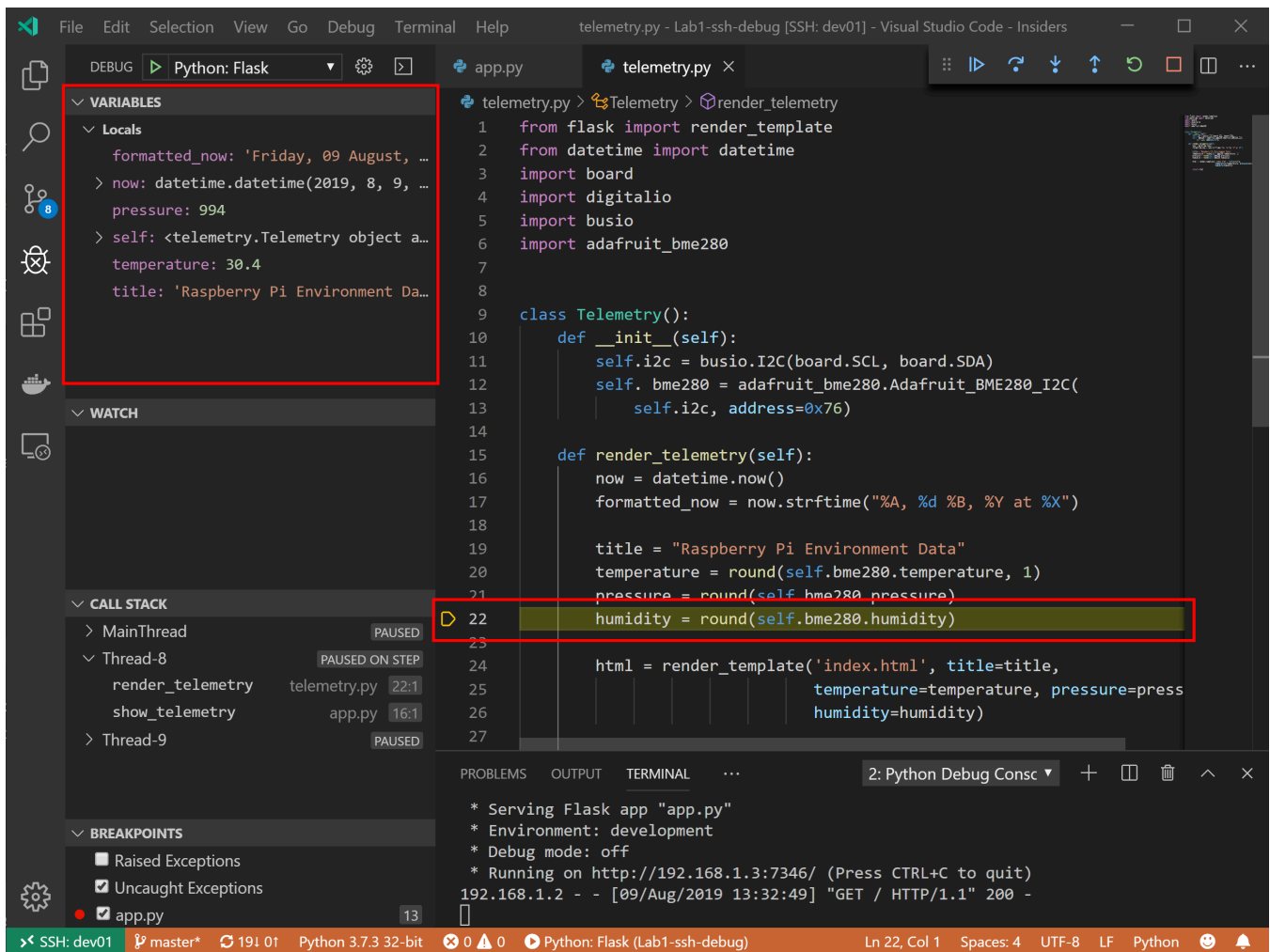
2. Right mouse click a variable, and you will discover you can change, copy, or watch variables. Try to change the value of a variable. **Hint**, double click on a variable value.

3. Press F5 to resume the Flask App, then switch back to your web browser and you will see the temperature, humidity, and pressure Sensor data displayed on the web page.

Raspberry Pi Environment Data

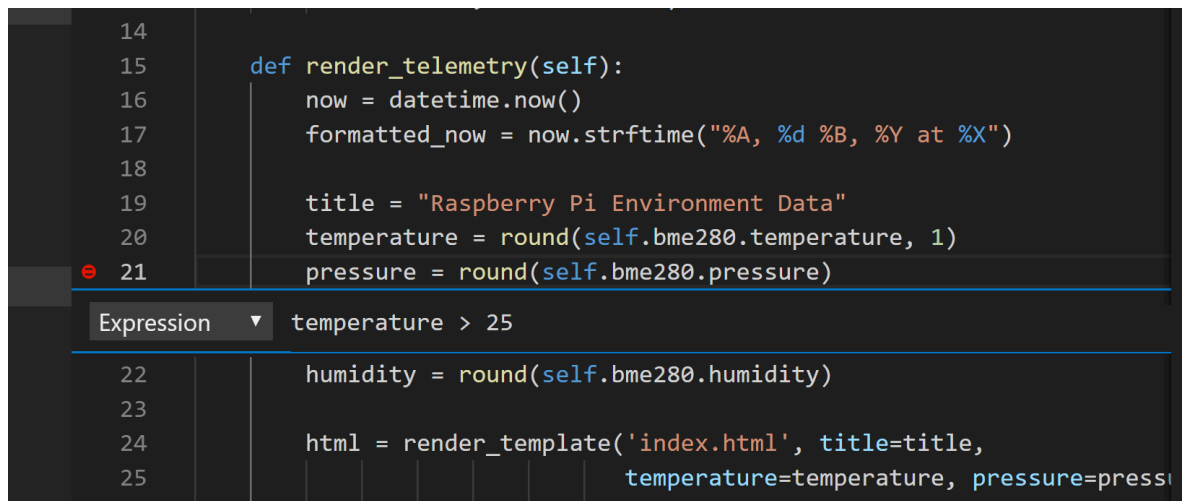| Telemetry	Value
Temperature	31.7 C
Humidity	17 %
Pressure	995 hPa

4. Press the **Refresh** button on your web browser. The Flask app will again stop at the

breakpoint in Visual Studio Code.

Experiment with Debugger Options

Things to try:

1. Review the Visual Studio Code Python Tutorial
2. Review the Python Flask tutorial
3. Review the Visual Studio Code Debugging Tutorial
4. Try to change the value of a variable from the Visual Studio Code **Variable Window**.
5. Try setting a **conditional** breakpoint

```
14
15        def render_telemetry(self):
16            now = datetime.now()
17            formatted_now = now.strftime("%A, %d %B, %Y at %X")
18
19            title = "Raspberry Pi Environment Data"
20            temperature = round(self.bme280.temperature, 1)
21            pressure = round(self.bme280.pressure)
   Expression    ▼   temperature > 25
22            humidity = round(self.bme280.humidity)
23
24            html = render_template('index.html', title=title,
25                                   temperature=temperature, pressure=press
```

6. Try the Visual Studio Code **Debug Console**. This will give you access to the Python REPL, try printing or setting variables, importing libraries etc.

```
print(temperature)

temperature = 24

import random
random.randrange(100, 1000)
```

```
 15        def render_telemetry(self):
 16            now = datetime.now()
 17            formatted_now = now.strftime("%A, %d %B, %Y at %X")
 18
 19            title = "Raspberry Pi Environment Data"
 20            temperature = round(self.bme280.temperature, 1)
 21            pressure = round(self.bme280.pressure)
 22            humidity = round(self.bme280.humidity)
 23
 24            html = render_template('index.html', title=title,
 25                                   temperature=temperature, pressure=pressure,
 26                                   humidity=humidity)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
 * Serving Flask app "app.py"
 * Environment: development
 * Debug mode: off
 * Running on http://192.168.1.3:7346/ (Press CTRL+C to quit)
print(temperature)
None
30.6
```

Type a Python command into the command line

```
> print(temperature)
```

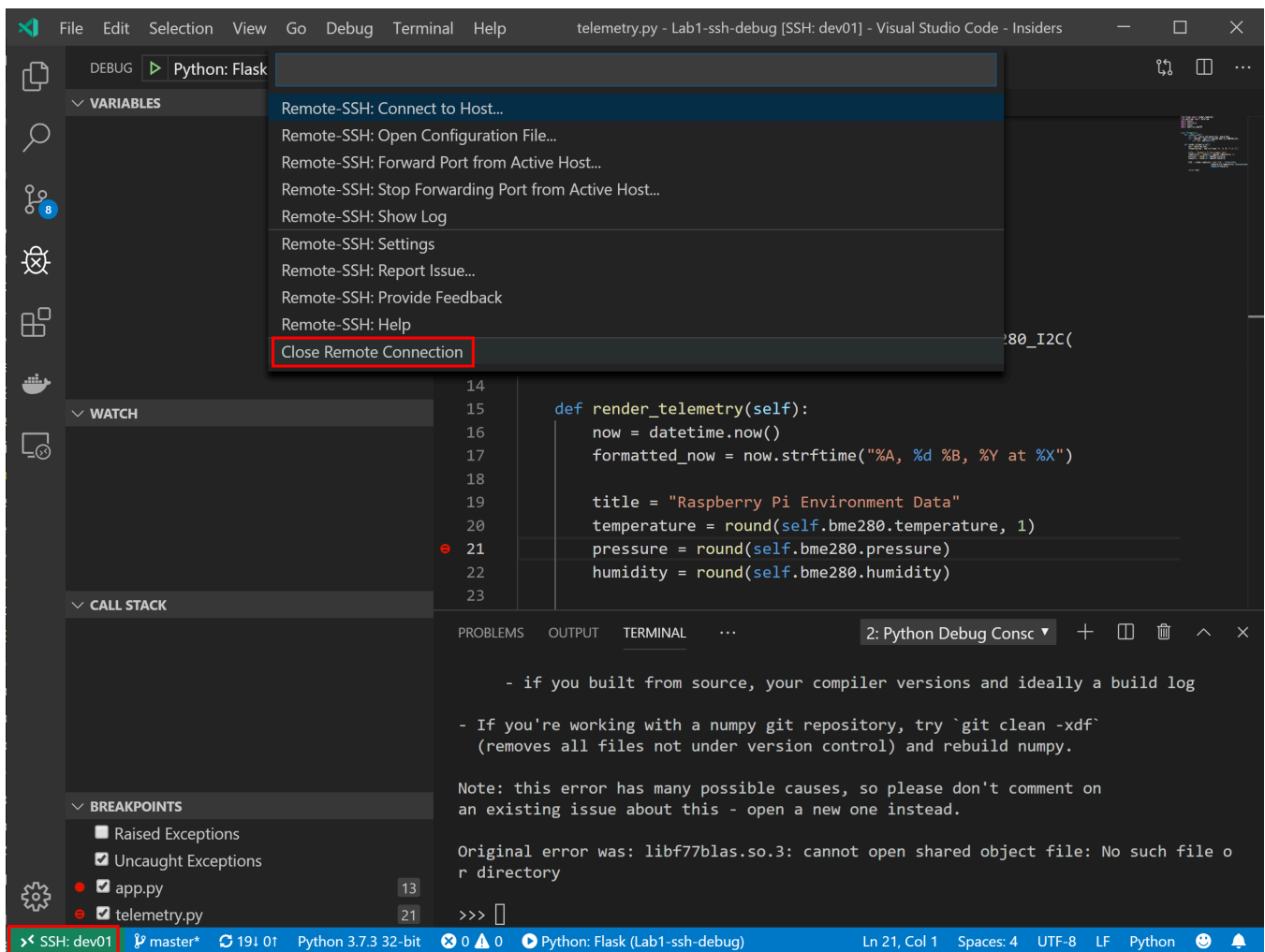⊗ 0 ⚠ 0   ▶ Python: Flask (Lab1-ssh-debug)

Update the Flask Template

1. Update the Flask **index.html** template to display the current date and time.
2. Rerun the Flask app.

Closing the Visual Studio Code Remote SSH

From Visual Studio Code, **Close Remote Connection**.

1. Click the **Remote SSH** button in the bottom left-hand corner and select **Close Remote Connection** from the dropdown list.

Finished

finished

References

- [Visual Studio Code](#)
- [Python](#)
- [Raspberry Pi](#)
- [Flask](#)