

MIT AI2 204

IoT with MIT App Inventor

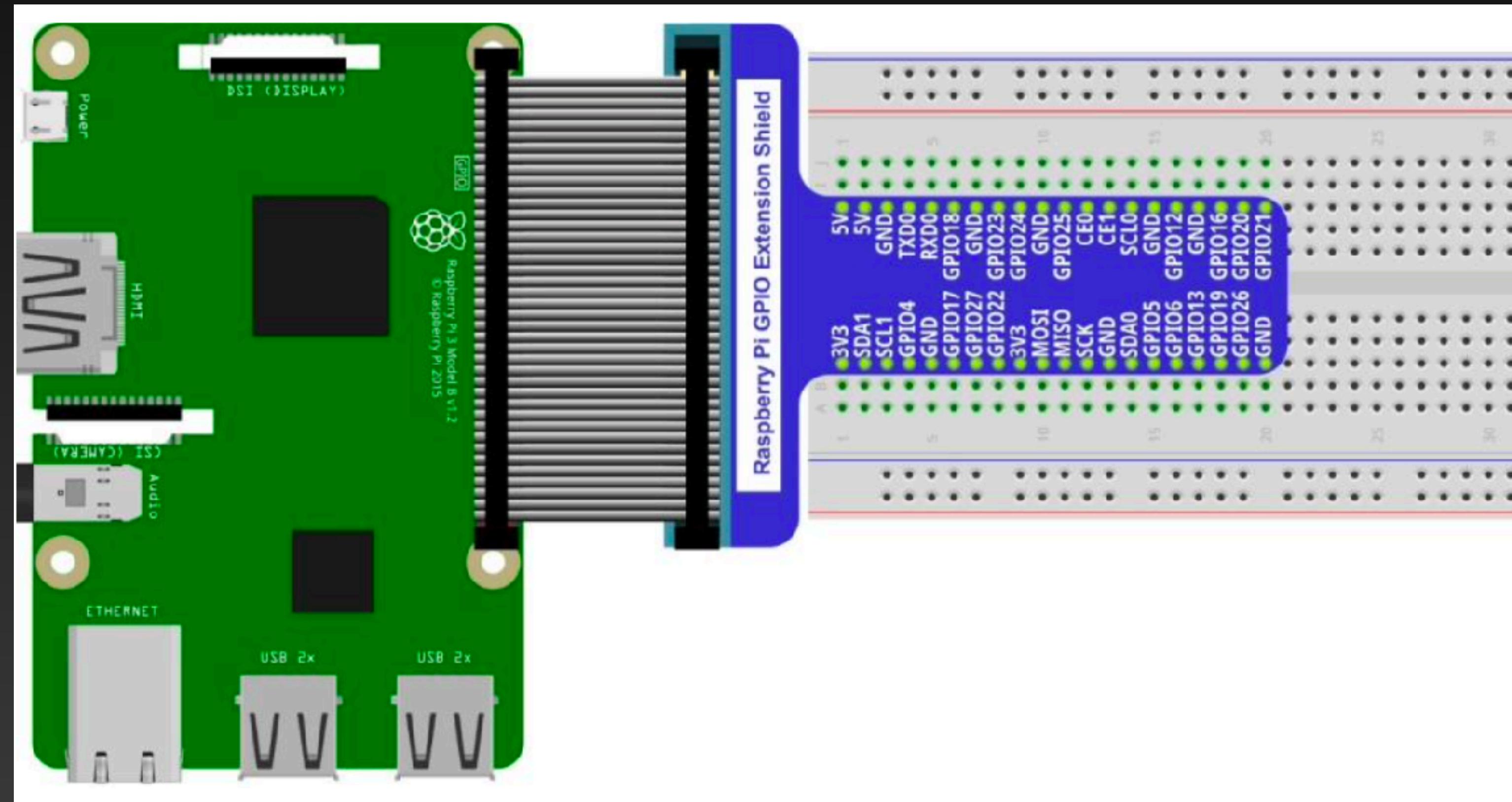
Fundamental

X. Tang

Introducing Extension Board

GPIO Extension Board

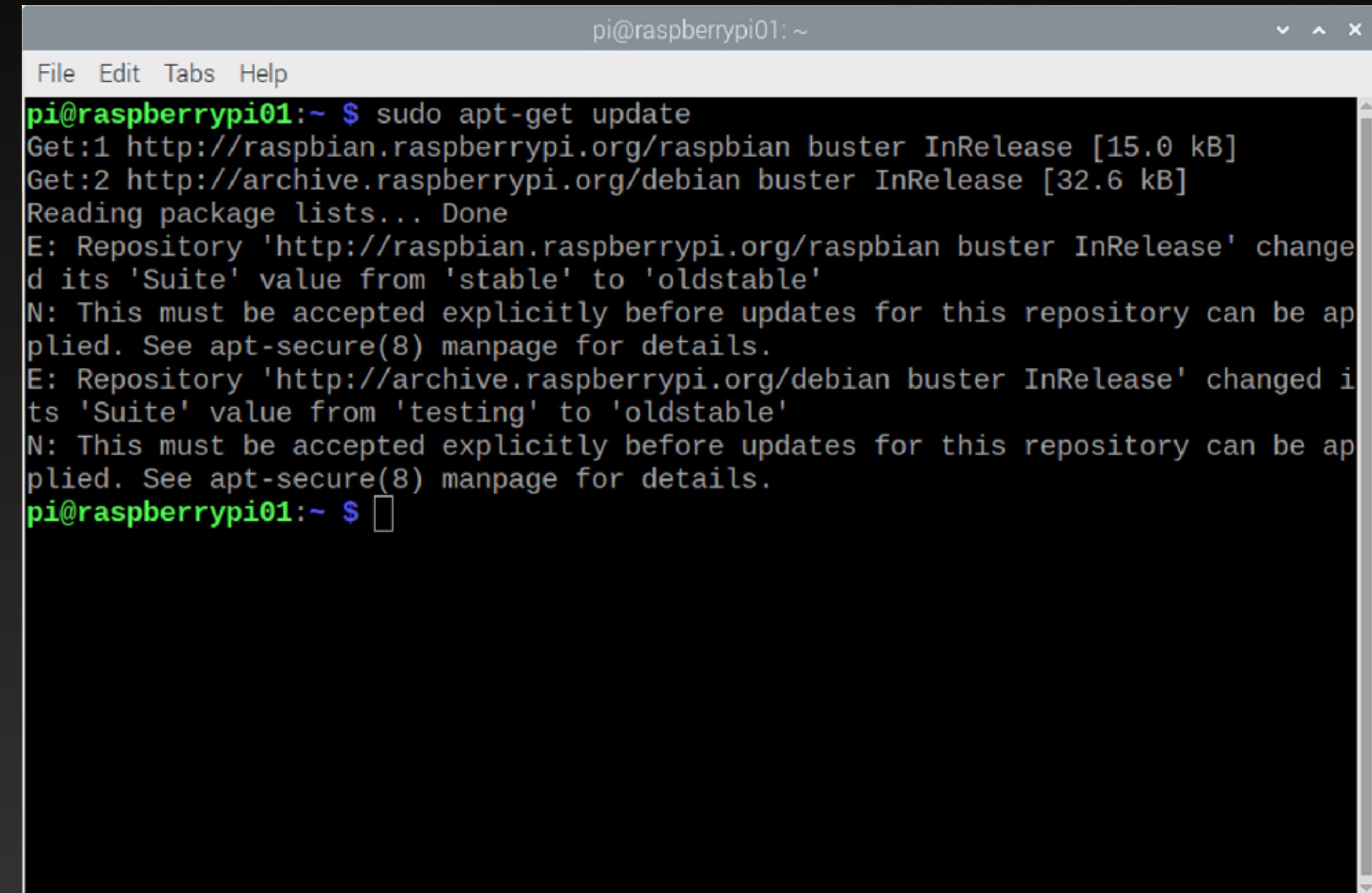
GPIO board is a convenient way to connect the RPi I/O ports to the breadboard directly. The GPIO pin sequence on Extension Board is identical to the GPIO pin sequence of RPi.



Physical Computing from Python - GPIO Library

Install GPIO Library

sudo apt-get update



A screenshot of a terminal window titled "pi@raspberrypi01: ~". The window has a standard Linux-style interface with a menu bar (File, Edit, Tabs, Help) and a title bar. The main area contains the command "pi@raspberrypi01:~ \$ sudo apt-get update" followed by its output. The output shows the system checking package lists from two repositories: "raspbian.raspberrypi.org/raspbian" and "archive.raspberrypi.org/debian". It indicates that the 'stable' suite was changed to 'oldstable' for both repositories, which requires explicit acceptance before updates can be applied. The command ends with a blank line.

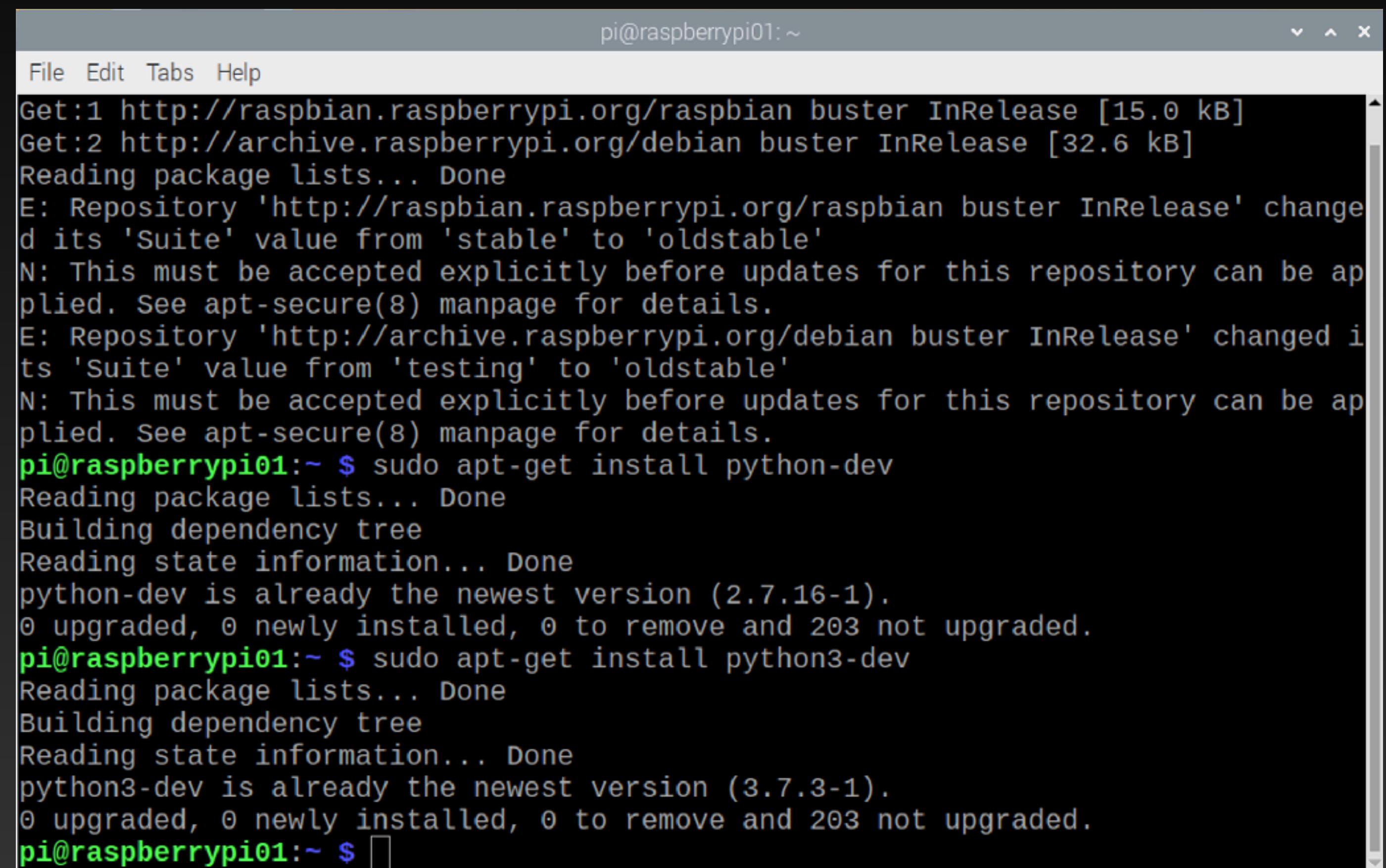
```
pi@raspberrypi01:~ $ sudo apt-get update
Get:1 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]
Get:2 http://archive.raspberrypi.org/debian buster InRelease [32.6 kB]
Reading package lists... Done
E: Repository 'http://raspbian.raspberrypi.org/raspbian buster InRelease' changed its 'Suite' value from 'stable' to 'oldstable'
N: This must be accepted explicitly before updates for this repository can be applied. See apt-secure(8) manpage for details.
E: Repository 'http://archive.raspberrypi.org/debian buster InRelease' changed its 'Suite' value from 'testing' to 'oldstable'
N: This must be accepted explicitly before updates for this repository can be applied. See apt-secure(8) manpage for details.
pi@raspberrypi01:~ $ 
```

Physical Computing from Python - GPIO Library

Install GPIO Library

```
sudo apt-get install  
python-dev
```

```
sudo apt-get install  
python3-dev
```



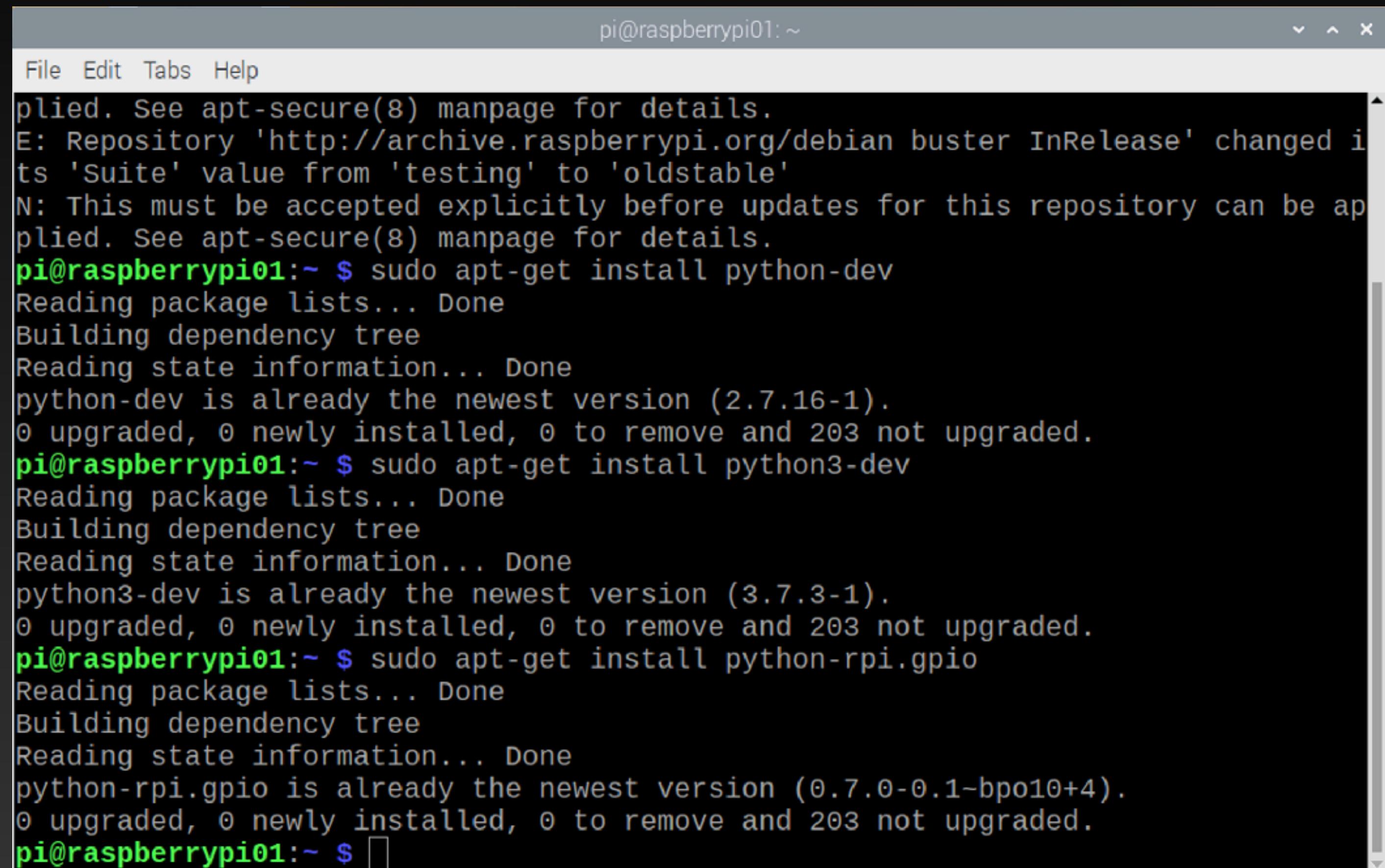
The screenshot shows a terminal window titled "pi@raspberrypi01:~". The window contains the following text:

```
pi@raspberrypi01:~  
File Edit Tabs Help  
Get:1 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]  
Get:2 http://archive.raspberrypi.org/debian buster InRelease [32.6 kB]  
Reading package lists... Done  
E: Repository 'http://raspbian.raspberrypi.org/raspbian buster InRelease' changed its 'Suite' value from 'stable' to 'oldstable'  
N: This must be accepted explicitly before updates for this repository can be applied. See apt-secure(8) manpage for details.  
E: Repository 'http://archive.raspberrypi.org/debian buster InRelease' changed its 'Suite' value from 'testing' to 'oldstable'  
N: This must be accepted explicitly before updates for this repository can be applied. See apt-secure(8) manpage for details.  
pi@raspberrypi01:~ $ sudo apt-get install python-dev  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
python-dev is already the newest version (2.7.16-1).  
0 upgraded, 0 newly installed, 0 to remove and 203 not upgraded.  
pi@raspberrypi01:~ $ sudo apt-get install python3-dev  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
python3-dev is already the newest version (3.7.3-1).  
0 upgraded, 0 newly installed, 0 to remove and 203 not upgraded.  
pi@raspberrypi01:~ $
```

Physical Computing from Python - GPIO Library

Install GPIO Library

```
sudo apt-get install  
libgpiod2 libgpiod-dev
```



A screenshot of a terminal window titled "pi@raspberrypi01: ~". The window shows the command-line interface for installing the GPIO library. The user runs three commands: "sudo apt-get install python-dev", "sudo apt-get install python3-dev", and "sudo apt-get install python-rpi.gpio". Each command outputs the package lists, dependency trees, state information, and upgrade status, indicating that the packages are already at their newest versions. The terminal window has a standard Linux-style header with "File", "Edit", "Tabs", and "Help" menu options.

```
pi@raspberrypi01:~  
File Edit Tabs Help  
plied. See apt-secure(8) manpage for details.  
E: Repository 'http://archive.raspberrypi.org/debian buster InRelease' changed its 'Suite' value from 'testing' to 'oldstable'  
N: This must be accepted explicitly before updates for this repository can be applied. See apt-secure(8) manpage for details.  
pi@raspberrypi01:~ $ sudo apt-get install python-dev  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
python-dev is already the newest version (2.7.16-1).  
0 upgraded, 0 newly installed, 0 to remove and 203 not upgraded.  
pi@raspberrypi01:~ $ sudo apt-get install python3-dev  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
python3-dev is already the newest version (3.7.3-1).  
0 upgraded, 0 newly installed, 0 to remove and 203 not upgraded.  
pi@raspberrypi01:~ $ sudo apt-get install python-rpi.gpio  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
python-rpi.gpio is already the newest version (0.7.0-0.1-bpo10+4).  
0 upgraded, 0 newly installed, 0 to remove and 203 not upgraded.  
pi@raspberrypi01:~ $
```

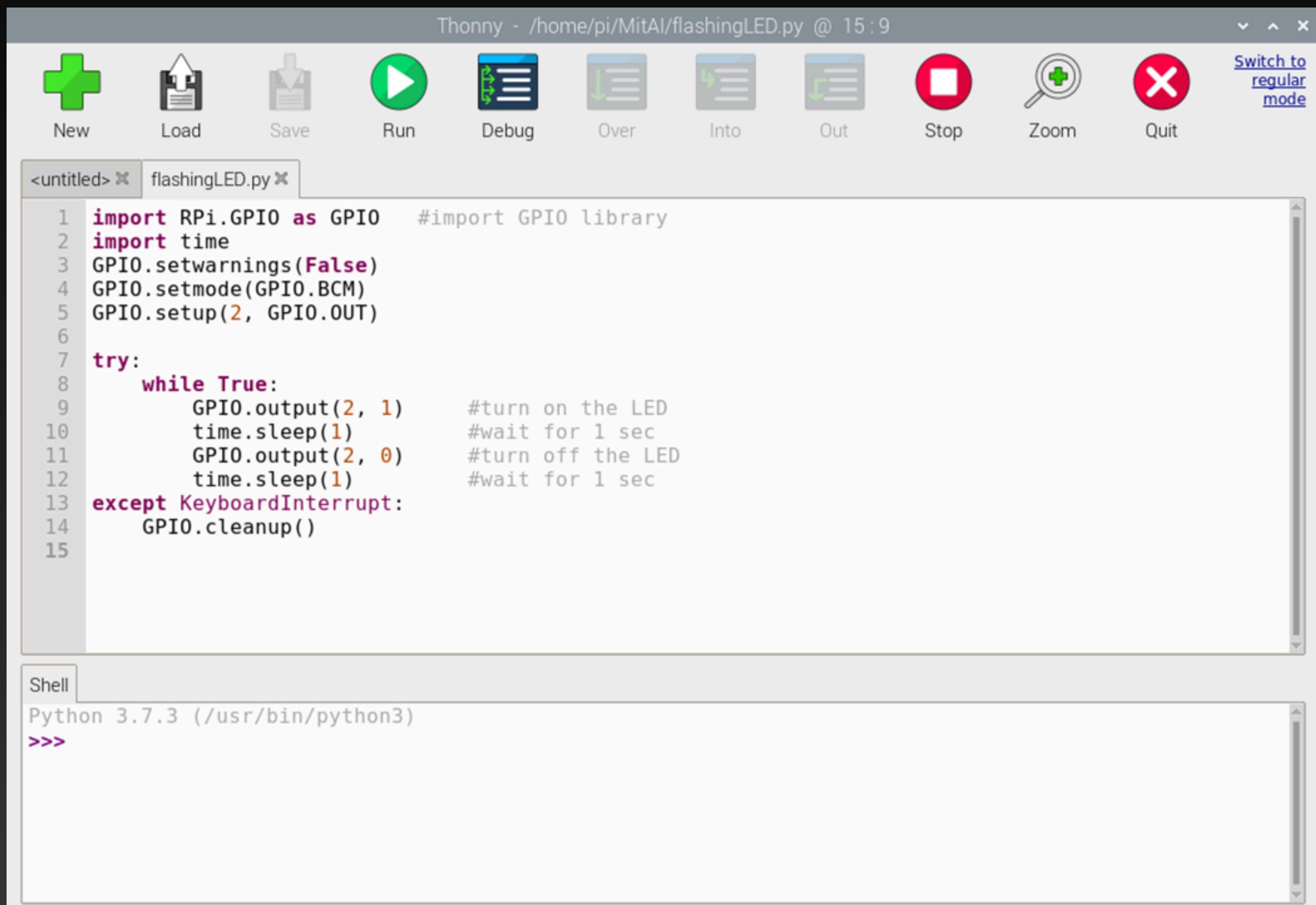
Physical Computing from Python - GPIO Library

The GPIO Library

The GPIO library is called RPi.GPIO and it should already be installed on your raspberry Pi 4/5.

When you programming in python, this library must be included in your code. The statement is:

```
import RPi.GPIO as GPIO
```



The screenshot shows the Thonny Python IDE interface. The title bar reads "Thonny - /home/pi/MitAI/flashingLED.py @ 15 : 9". The menu bar includes "File", "Edit", "Run", "Debug", "Tools", "Help", and "Switch to regular mode". The main window has two tabs: "<untitled>" and "flashingLED.py". The "flashingLED.py" tab contains the following code:

```
1 import RPi.GPIO as GPIO      #import GPIO library
2 import time
3 GPIO.setwarnings(False)
4 GPIO.setmode(GPIO.BCM)
5 GPIO.setup(2, GPIO.OUT)

6
7 try:
8     while True:
9         GPIO.output(2, 1)      #turn on the LED
10        time.sleep(1)        #wait for 1 sec
11        GPIO.output(2, 0)      #turn off the LED
12        time.sleep(1)        #wait for 1 sec
13 except KeyboardInterrupt:
14     GPIO.cleanup()
```

Below the code editor is a "Shell" tab showing the Python interpreter prompt:

```
Python 3.7.3 (/usr/bin/python3)
>>>
```

Physical Computing from Python - GPIO Library

Pin Numbering

There are two ways to refer to the GPIO pins.

1. Use the BOARD numbering, where the pin numbers on the GPIO connector of the Raspberry Pi 4/5 are used.

`GPIO.setmode(GPIO.BUILD)`

The screenshot shows the Thonny IDE interface. The title bar reads "Thonny - /home/pi/MitAI/flashingLED_Board.py @ 11:22". The menu bar includes "File", "Edit", "Run", "Debug", "Tools", "Help". The toolbar has icons for New, Load, Save, Run, Debug, Over, Into, Out, Stop, Zoom, and Quit. A "Switch to regular mode" button is also present. There are two tabs open: "flashingLED.py" and "flashingLED_Board.py". The "flashingLED_Board.py" tab contains the following code:

```
1 import RPi.GPIO as GPIO      #import GPIO library
2 import time
3 GPIO.setwarnings(False)
4 GPIO.setmode(GPIO.BOARD)
5 GPIO.setup(3, GPIO.OUT)

6
7 try:
8     while True:
9         GPIO.output(3, 1)      #turn on the LED
10        time.sleep(1)         #wait for 1 sec
11        GPIO.output(3, 0)      #turn off the LED
12        time.sleep(1)         #wait for 1 sec
13 except KeyboardInterrupt:
14     GPIO.cleanup()
```

The "Shell" tab at the bottom shows the Python interpreter prompt:

```
Shell
Python 3.7.3 (/usr/bin/python3)
>>>
```

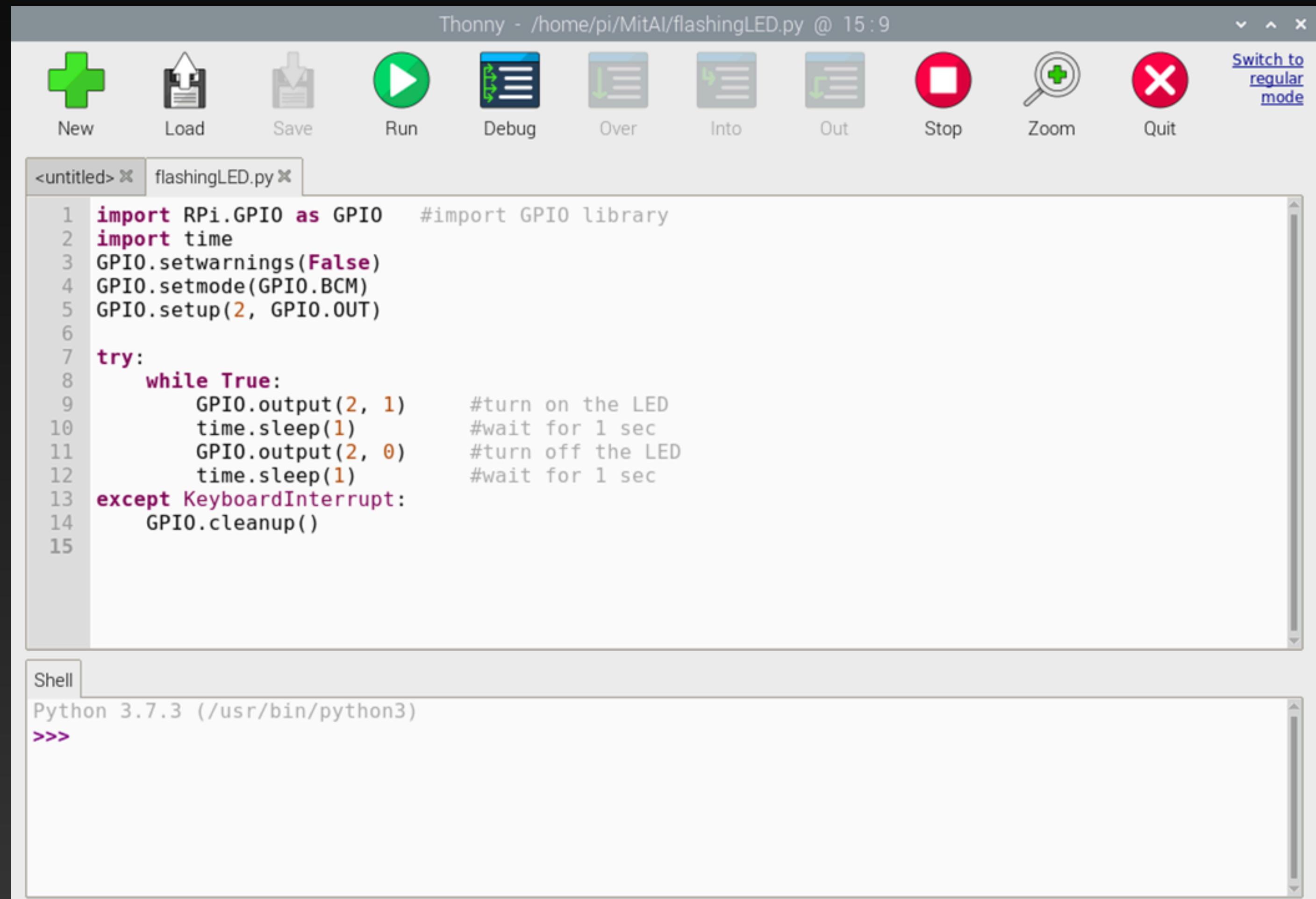
Physical Computing from Python - GPIO Library

Pin Numbering

There are two ways to refer to the GPIO pins.

2. Use the BCM, which is preferred. It uses the channel numbers refers to which pin on the board.

`GPIO.setmode(GPIO.BCM)`



The screenshot shows the Thonny IDE interface. The title bar reads "Thonny - /home/pi/MitAI/flashingLED.py @ 15 : 9". The menu bar includes "File", "Edit", "Run", "Debug", "Tools", "Help", and "Switch to regular mode". The main window has two tabs: "flashlingLED.py" (active) and "<untitled>". The code in "flashlingLED.py" is:

```
1 import RPi.GPIO as GPIO      #import GPIO library
2 import time
3 GPIO.setwarnings(False)
4 GPIO.setmode(GPIO.BCM)
5 GPIO.setup(2, GPIO.OUT)

6
7 try:
8     while True:
9         GPIO.output(2, 1)      #turn on the LED
10        time.sleep(1)        #wait for 1 sec
11        GPIO.output(2, 0)      #turn off the LED
12        time.sleep(1)        #wait for 1 sec
13 except KeyboardInterrupt:
14     GPIO.cleanup()
```

The "Shell" tab at the bottom shows the Python interpreter prompt:

```
Python 3.7.3 (/usr/bin/python3)
>>>
```

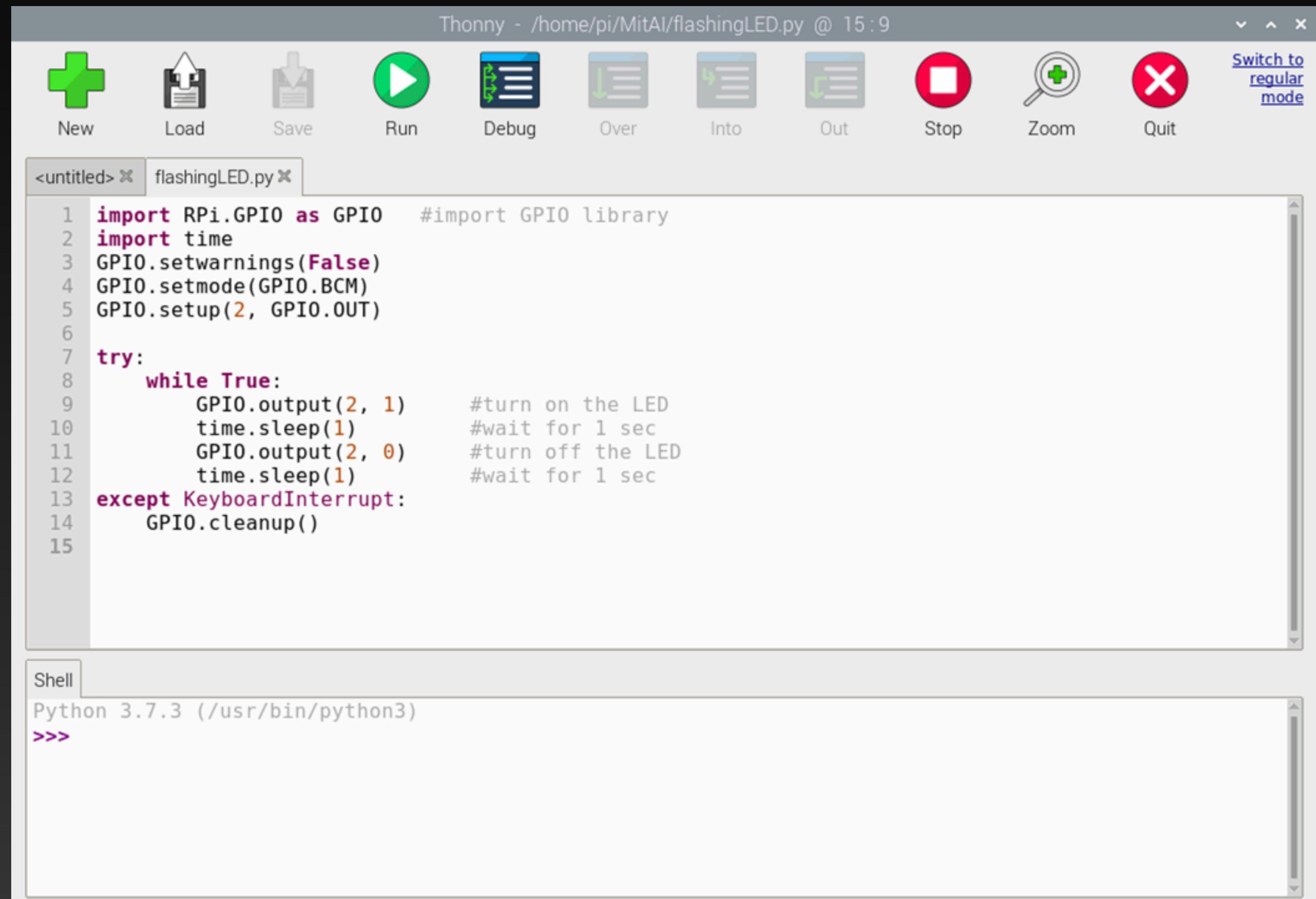
Physical Computing from Python - GPIO Library

Channel(I/O port pin) Config Input configuration

You need to configure the channels(or port pins) you are using whether they are input or output channels.

GPIO.setup(channel,GPIO.IN)

Channel refers to the channel number based on the set mode statement in previous setup.



The screenshot shows the Thonny IDE interface with a Python script named 'flashingLED.py' open. The script code is as follows:

```
1 import RPi.GPIO as GPIO      #import GPIO library
2 import time
3 GPIO.setwarnings(False)
4 GPIO.setmode(GPIO.BCM)
5 GPIO.setup(2, GPIO.OUT)

6
7 try:
8     while True:
9         GPIO.output(2, 1)      #turn on the LED
10        time.sleep(1)        #wait for 1 sec
11        GPIO.output(2, 0)      #turn off the LED
12        time.sleep(1)        #wait for 1 sec
13 except KeyboardInterrupt:
14     GPIO.cleanup()
15
```

The Thonny interface includes a toolbar with icons for New, Load, Save, Run, Debug, Over, Into, Out, Stop, Zoom, and Quit. A status bar at the top right indicates the file path and line number (Thonny - /home/pi/MitAI/flashingLED.py @ 15 : 9). Below the toolbar is a menu bar with 'Switch to regular mode'. The bottom of the window features a 'Shell' tab showing the Python version and a prompt (>>>).

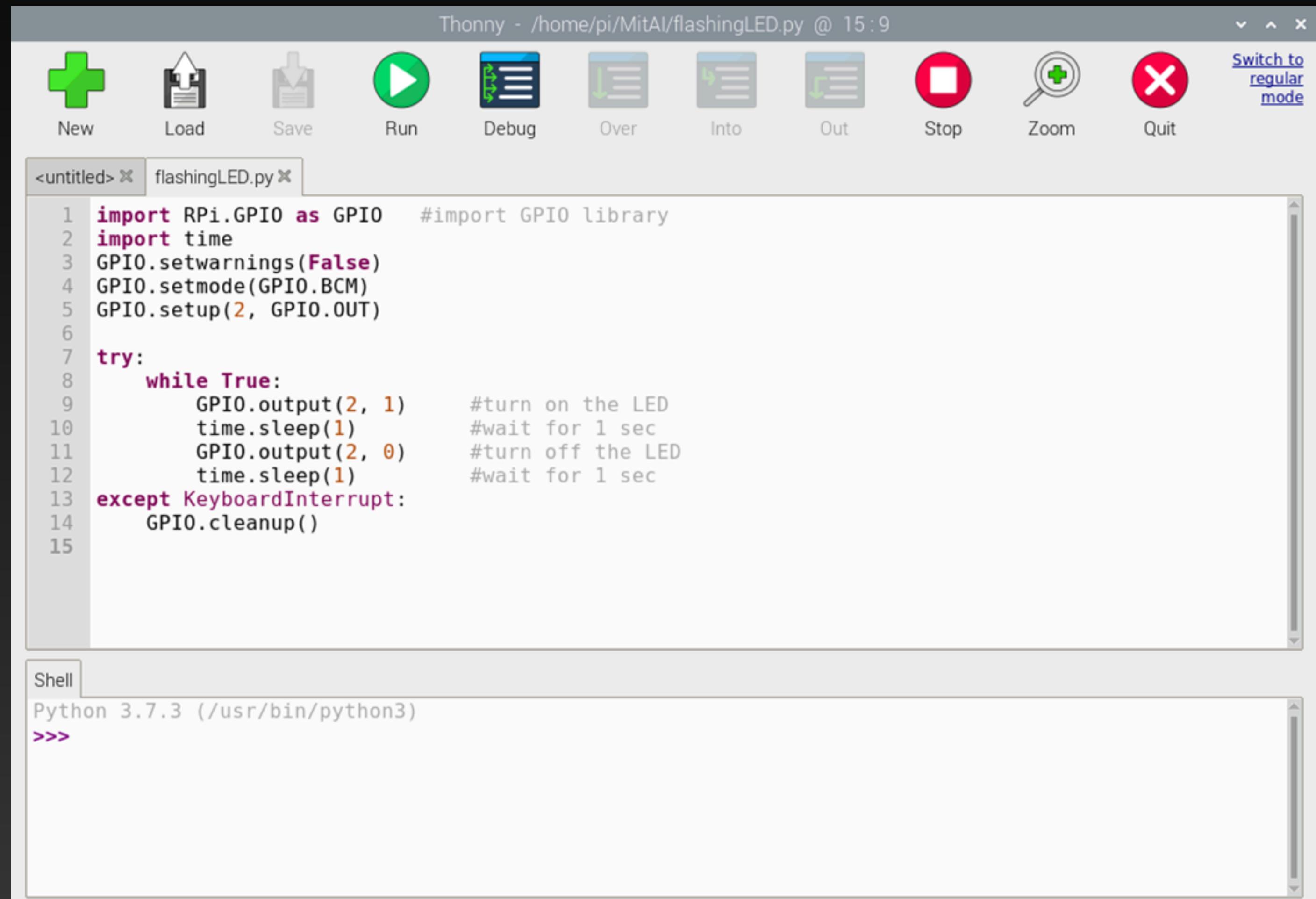
Physical Computing from Python - GPIO Library

Channel(I/O port pin) Config

Output configuration

GPIO.setup(channel,GPIO.OUT)

Channel refers to the channel number based on the set mode statement in previous setup.



The screenshot shows the Thonny IDE interface with a Python script named 'flashingLED.py' open. The script contains code to import the RPi.GPIO library, set up a GPIO pin as an output, and create a loop to alternately turn on and off a LED connected to GPIO 2, with a one-second delay between each state. A 'Shell' tab at the bottom shows the Python interpreter ready for commands.

```
1 import RPi.GPIO as GPIO      #import GPIO library
2 import time
3 GPIO.setwarnings(False)
4 GPIO.setmode(GPIO.BCM)
5 GPIO.setup(2, GPIO.OUT)

6
7 try:
8     while True:
9         GPIO.output(2, 1)      #turn on the LED
10        time.sleep(1)        #wait for 1 sec
11        GPIO.output(2, 0)      #turn off the LED
12        time.sleep(1)        #wait for 1 sec
13 except KeyboardInterrupt:
14     GPIO.cleanup()

15
```

Shell

```
Python 3.7.3 (/usr/bin/python3)
>>>
```

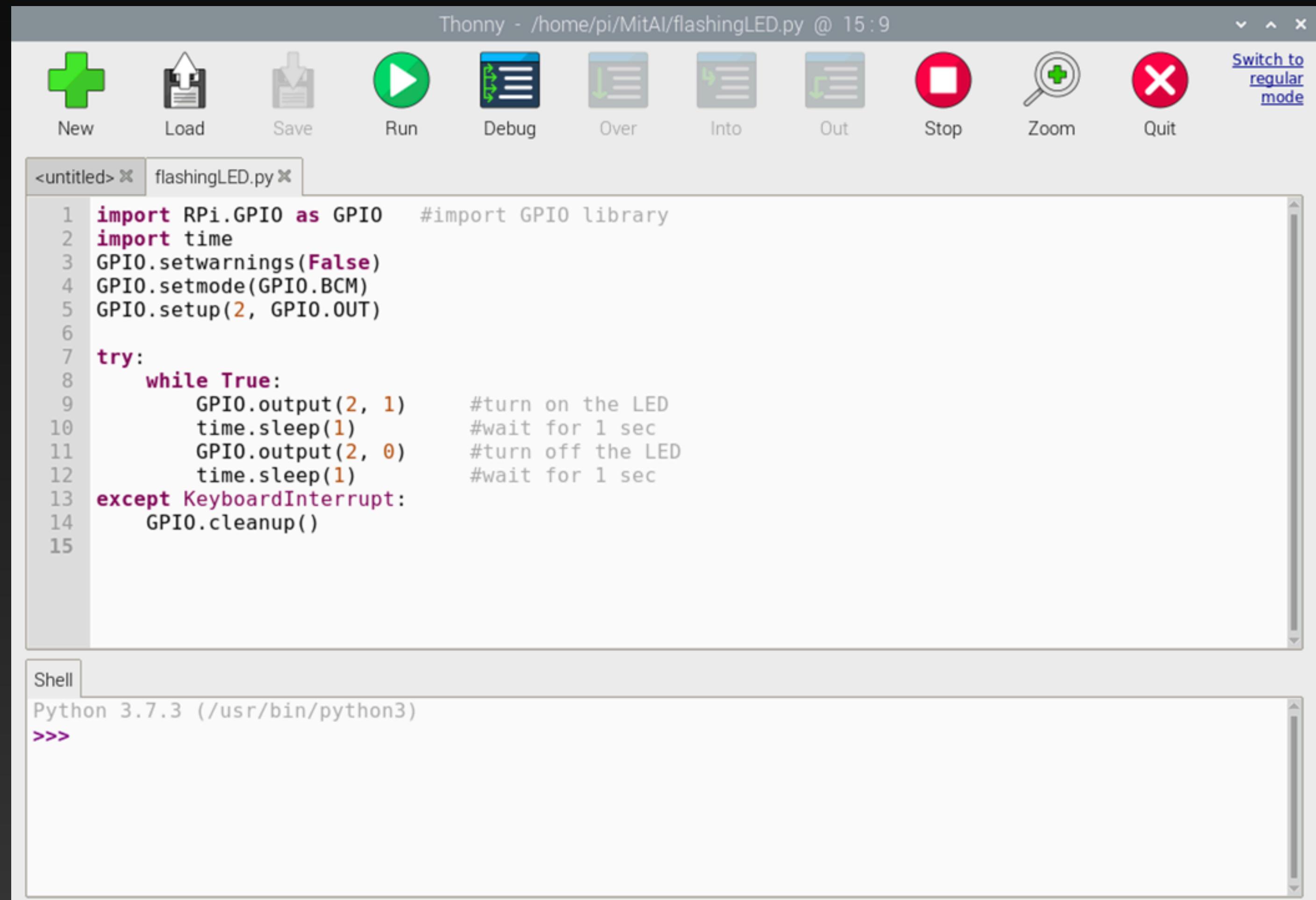
Physical Computing from Python - GPIO Library

Channel(I/O port pin) Config

Output configuration

We can specify a value for an output pin during it's setup. For example, we can config a channel as output and at the same time set its value to logic HIGH(3.3V)

```
GPIO.setup(channel,GPIO.OUT  
, initial=GPIO.HIGH)
```



The screenshot shows the Thonny IDE interface. The title bar reads "Thonny - /home/pi/MitAI/flashingLED.py @ 15 : 9". The menu bar includes "File", "Edit", "Run", "Debug", "Tools", "Help", and "Switch to regular mode". The main window has two tabs: "<untitled>" and "flashingLED.py". The "flashingLED.py" tab contains the following code:

```
1 import RPi.GPIO as GPIO      #import GPIO library  
2 import time  
3 GPIO.setwarnings(False)  
4 GPIO.setmode(GPIO.BCM)  
5 GPIO.setup(2, GPIO.OUT)  
6  
7 try:  
8     while True:  
9         GPIO.output(2, 1)      #turn on the LED  
10        time.sleep(1)        #wait for 1 sec  
11        GPIO.output(2, 0)      #turn off the LED  
12        time.sleep(1)        #wait for 1 sec  
13 except KeyboardInterrupt:  
14     GPIO.cleanup()  
15
```

Below the code editor is a "Shell" tab showing the Python environment:

```
Shell  
Python 3.7.3 (/usr/bin/python3)  
>>>
```

Physical Computing from Python - GPIO Library

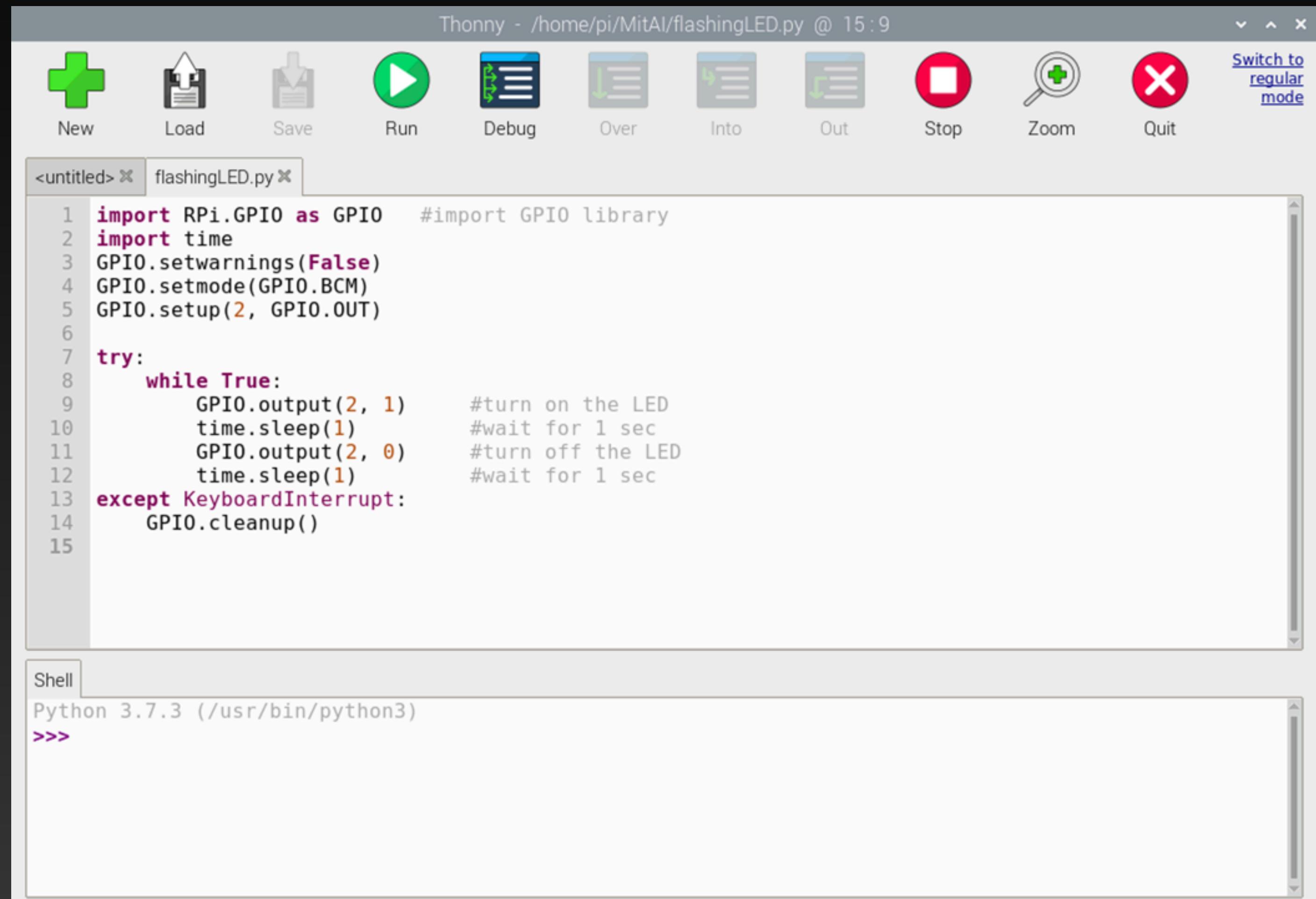
Channel(I/O port pin) Config

Output configuration

To send data to an output port pin we can use the following statement:

`GPIO.setup(channel, value)`

Where value can be 0(or `GPIO.LOW`, or `False`), or 1 (or `GPIO.HIGH`, or `True`)



The screenshot shows the Thonny IDE interface. The title bar reads "Thonny - /home/pi/MitAI/flashingLED.py @ 15 : 9". The menu bar includes "File", "Edit", "Run", "Debug", "Tools", "Help", and "Switch to regular mode". The main window has two tabs: "<untitled>" and "flashingLED.py". The code in "flashingLED.py" is as follows:

```
1 import RPi.GPIO as GPIO      #import GPIO library
2 import time
3 GPIO.setwarnings(False)
4 GPIO.setmode(GPIO.BCM)
5 GPIO.setup(2, GPIO.OUT)

6 try:
7     while True:
8         GPIO.output(2, 1)      #turn on the LED
9         time.sleep(1)          #wait for 1 sec
10        GPIO.output(2, 0)     #turn off the LED
11        time.sleep(1)          #wait for 1 sec
12 except KeyboardInterrupt:
13     GPIO.cleanup()
```

Below the code editor is a "Shell" tab showing the Python interpreter prompt:

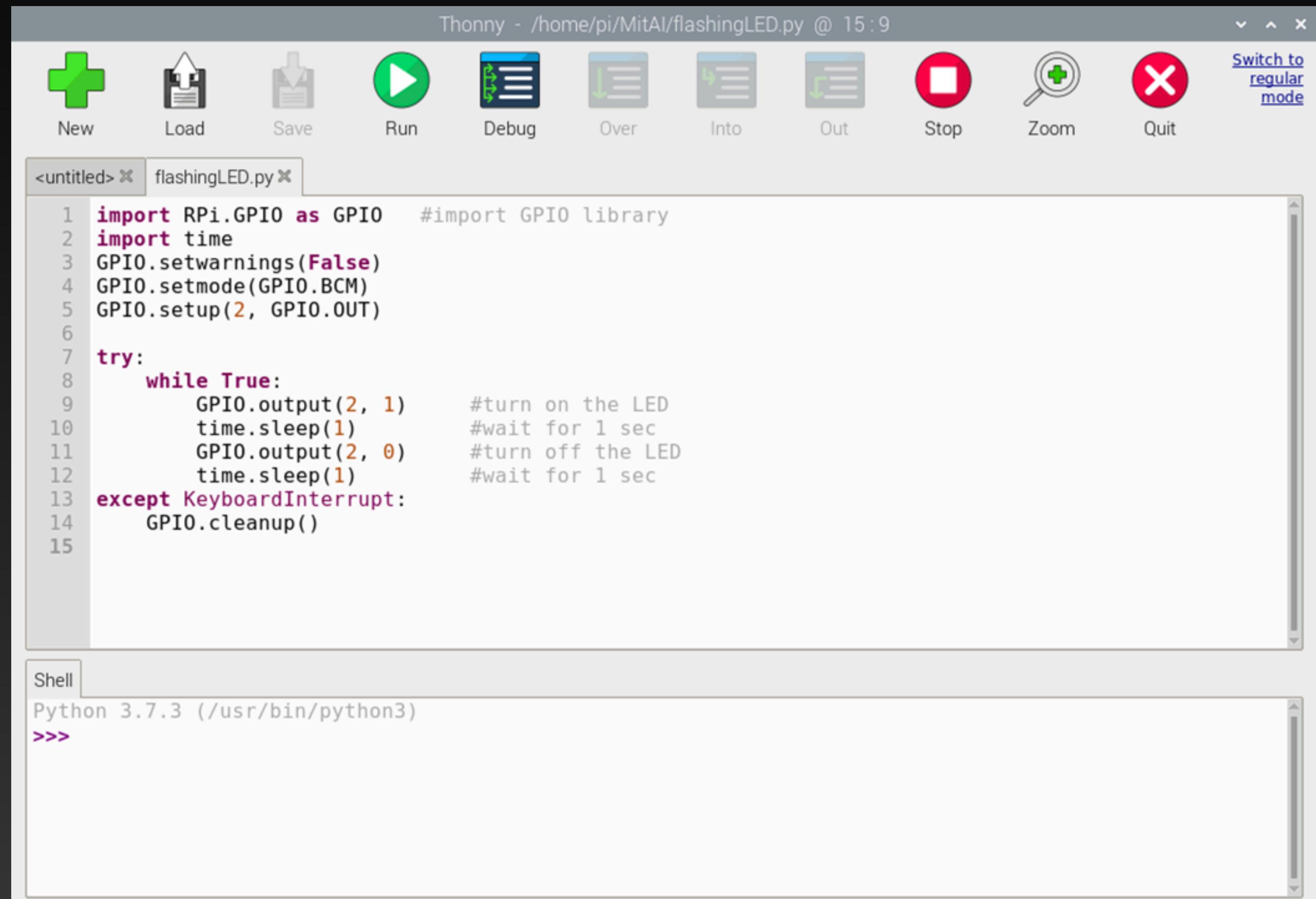
```
Python 3.7.3 (/usr/bin/python3)
>>>
```

Physical Computing from Python - GPIO Library

Channel(I/O port pin) Config Output configuration

At the end of the program, we should return all the used resources to the operating system. This is done by including the following statement at the end of our program:

`GPIO.cleanup()`



The screenshot shows the Thonny IDE interface with a Python script named `flashingLED.py`. The script code is as follows:

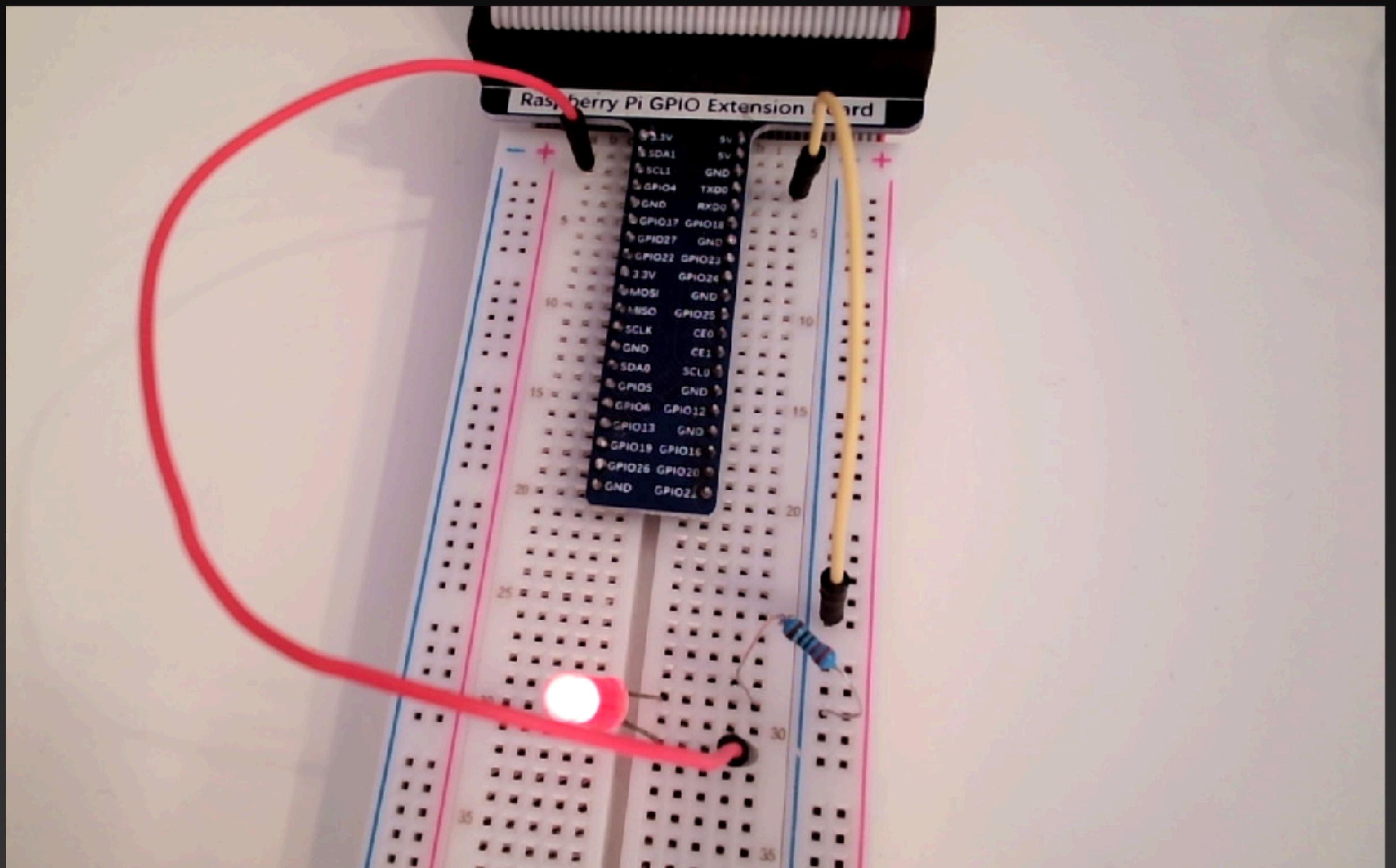
```
1 import RPi.GPIO as GPIO      #import GPIO library
2 import time
3 GPIO.setwarnings(False)
4 GPIO.setmode(GPIO.BCM)
5 GPIO.setup(2, GPIO.OUT)

6
7 try:
8     while True:
9         GPIO.output(2, 1)      #turn on the LED
10        time.sleep(1)        #wait for 1 sec
11        GPIO.output(2, 0)      #turn off the LED
12        time.sleep(1)        #wait for 1 sec
13 except KeyboardInterrupt:
14     GPIO.cleanup()
```

The Thonny interface includes a toolbar with icons for New, Load, Save, Run, Debug, Over, Into, Out, Stop, Zoom, and Quit. A status bar at the top right indicates the file path `/home/pi/MitAI/flashingLED.py` and the line number `@ 15 : 9`. A "Switch to regular mode" button is also visible.

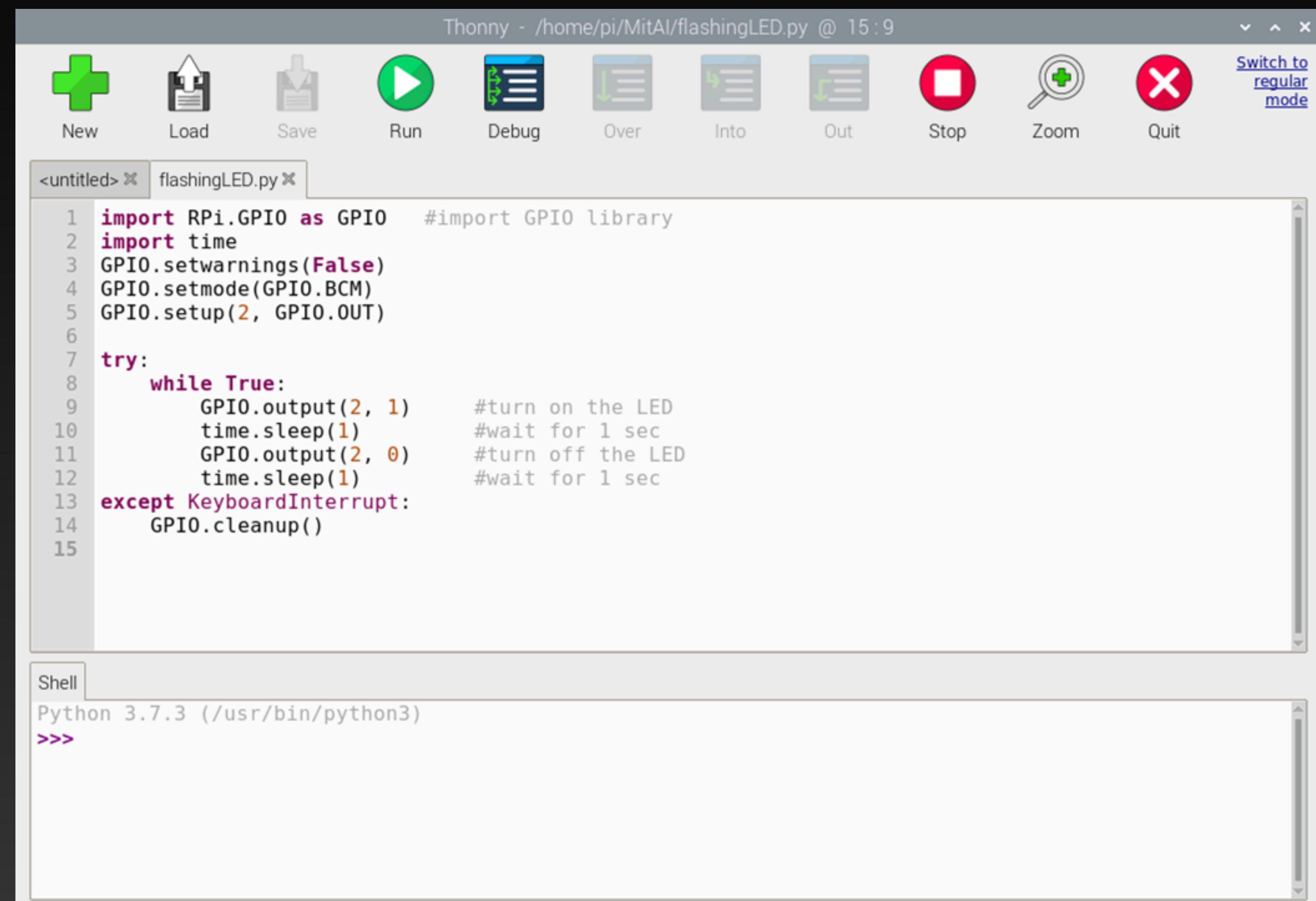
Flashing LED - Accessing the External World

Connect the wires use GPIO2
as output.



Flashing LED - Accessing the External World

Code Flashing LED program
in Thonny.



The screenshot shows the Thonny IDE interface. The title bar reads "Thonny - /home/pi/MitAI/flashingLED.py @ 15:9". The toolbar includes icons for New, Load, Save, Run, Debug, Over, Into, Out, Stop, Zoom, and Quit. The main code editor window has tabs for "untitled" and "flashingLED.py". The code in "flashingLED.py" is:

```
1 import RPi.GPIO as GPIO      #import GPIO library
2 import time
3 GPIO.setwarnings(False)
4 GPIO.setmode(GPIO.BCM)
5 GPIO.setup(2, GPIO.OUT)

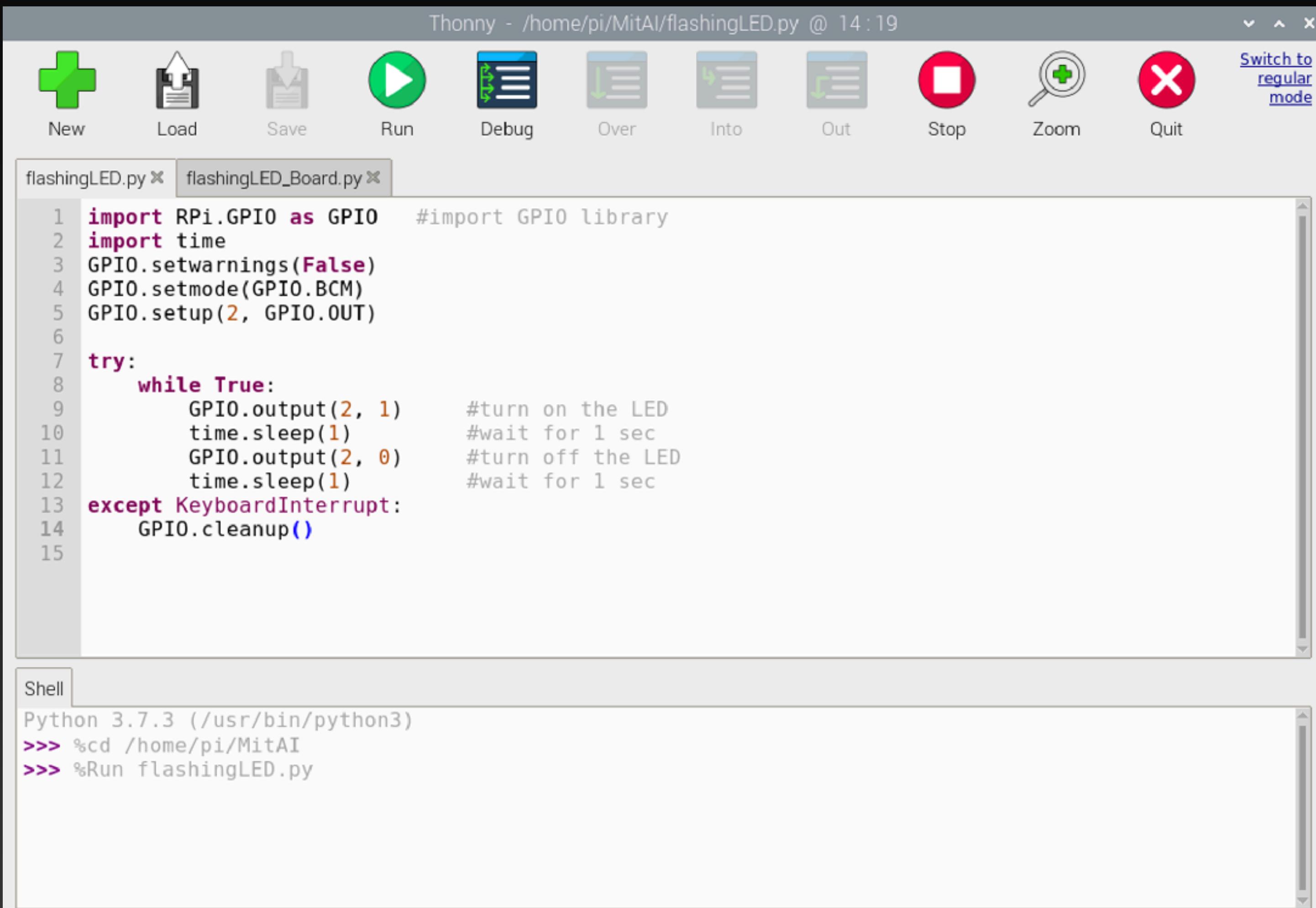
6
7 try:
8     while True:
9         GPIO.output(2, 1)      #turn on the LED
10        time.sleep(1)        #wait for 1 sec
11        GPIO.output(2, 0)      #turn off the LED
12        time.sleep(1)        #wait for 1 sec
13 except KeyboardInterrupt:
14     GPIO.cleanup()
```

Below the code editor is a "Shell" window showing the Python environment:

```
Shell
Python 3.7.3 (/usr/bin/python3)
>>>
```

Flashing LED - Accessing the External World

Run the code.



The screenshot shows the Thonny IDE interface. The title bar reads "Thonny - /home/pi/MitAI/flashingLED.py @ 14:19". The toolbar contains icons for New, Load, Save, Run, Debug, Over, Into, Out, Stop, Zoom, and Quit. A "Switch to regular mode" link is also present. The code editor window displays two files: "flashingLED.py" (selected) and "flashingLED_Board.py". The code in "flashingLED.py" is:

```
1 import RPi.GPIO as GPIO      #import GPIO library
2 import time
3 GPIO.setwarnings(False)
4 GPIO.setmode(GPIO.BCM)
5 GPIO.setup(2, GPIO.OUT)

6
7 try:
8     while True:
9         GPIO.output(2, 1)      #turn on the LED
10        time.sleep(1)        #wait for 1 sec
11        GPIO.output(2, 0)      #turn off the LED
12        time.sleep(1)        #wait for 1 sec
13 except KeyboardInterrupt:
14     GPIO.cleanup()
```

The shell window at the bottom shows the command line:

```
Shell
Python 3.7.3 (/usr/bin/python3)
>>> %cd /home/pi/MitAI
>>> %Run flashingLED.py
```

Flashing LED - Accessing the External World

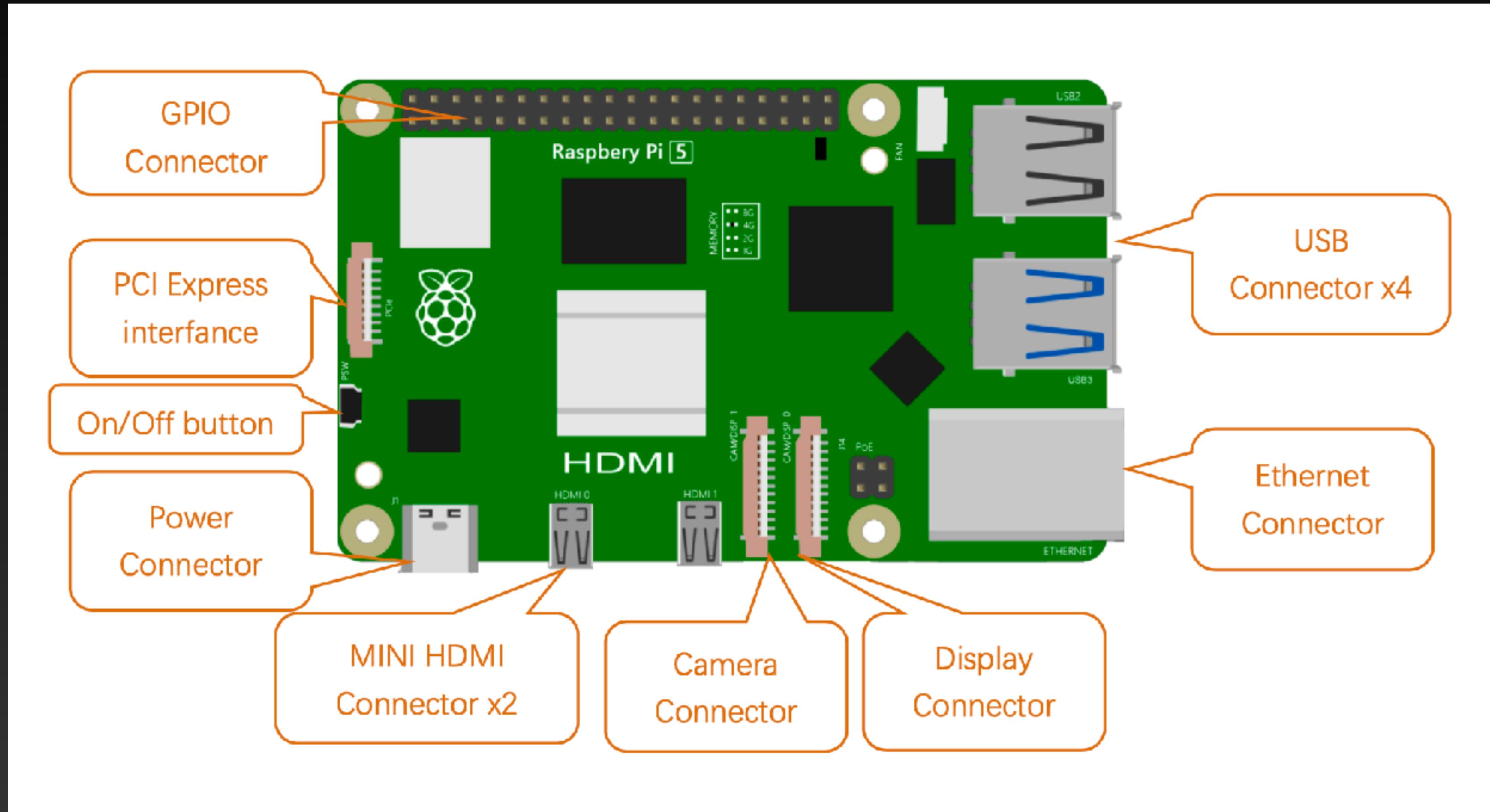
What did you see?

IoT with MIT App Inventor

Back at 8:12PM

Please update your name with studentID
In Zoom meeting

GPIO control in python on Raspberry Pi

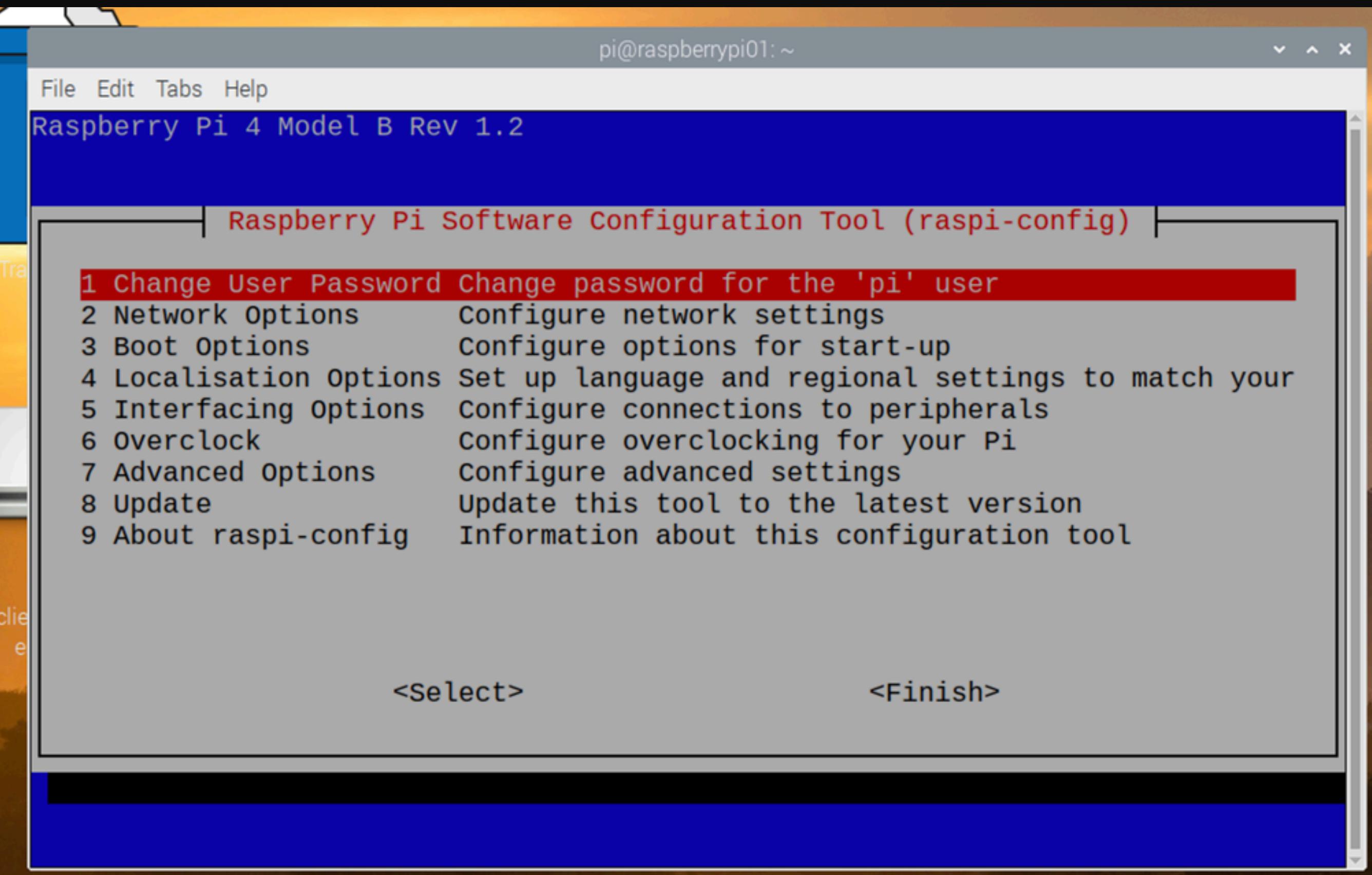


Remote Access -Setup

- \$ sudo raspi-config
- config P2 SSH and P3 VNC to YES under Interface Options
- Reboot
- SSH access. Windows, download Putty. [Https://www.putty.org/](https://www.putty.org/)
- MAC, SSH pi@IP
- Windows, download <https://www.realvnc.com/en/connect/download/viewer/>
- MAC, app store, windows remote desktop

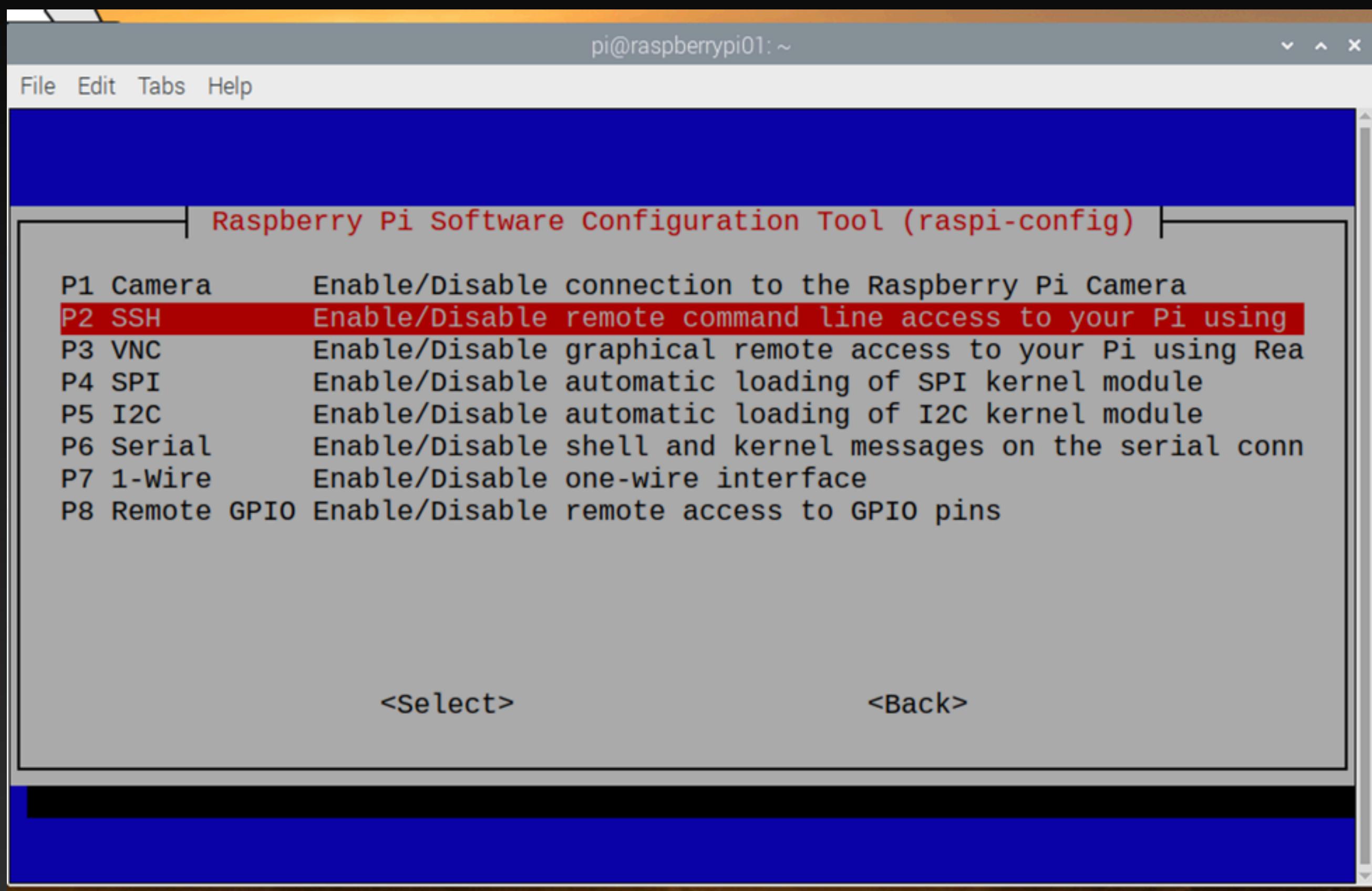
Remote Access -Setup

- \$ sudo raspi-config
- Go to Interfacing Options



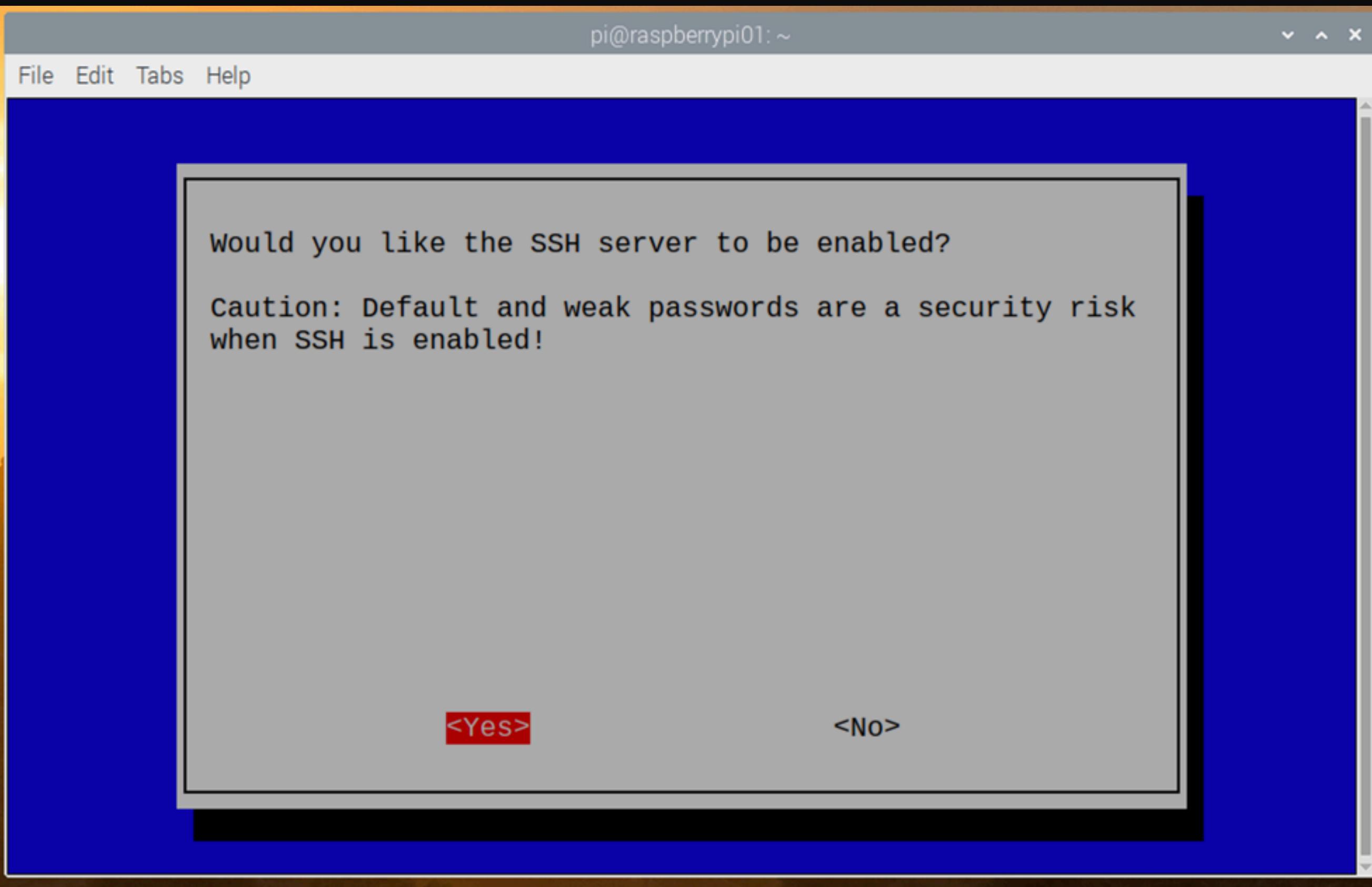
Remote Access -Setup

- Choose P2 SSH, hit enter



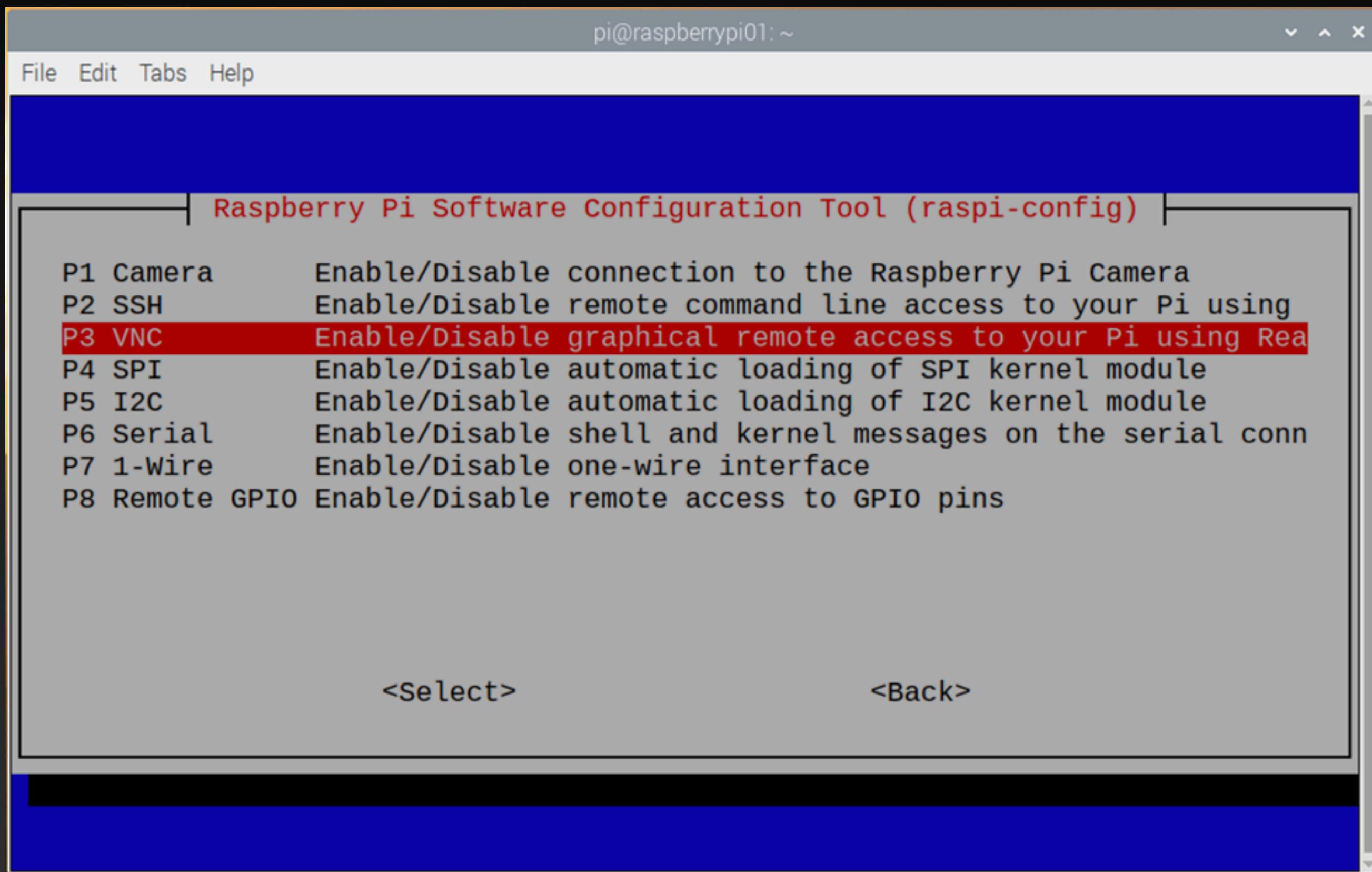
Remote Access -Setup

- Select Yes, hit enter



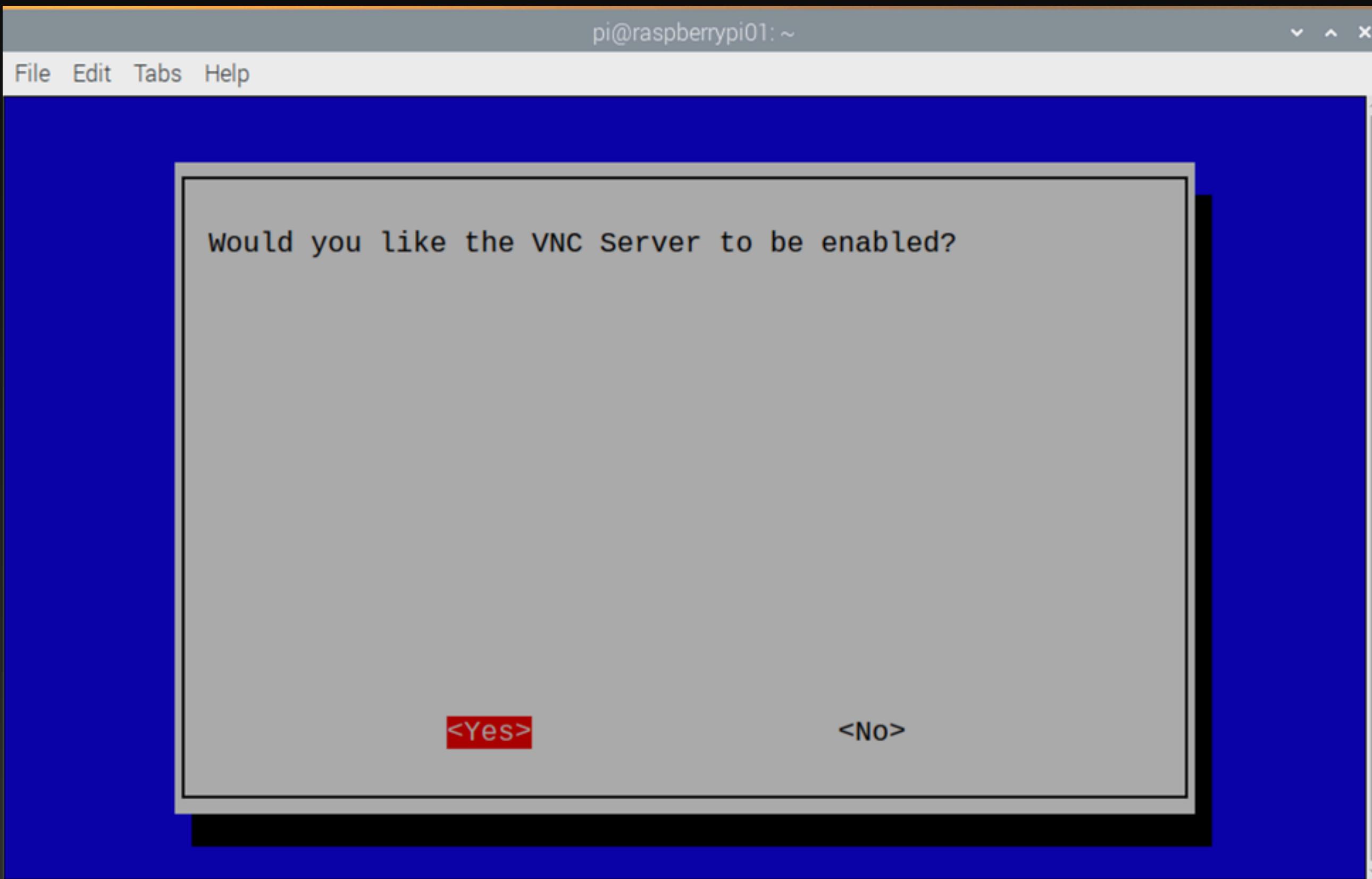
Remote Access -Setup

- Choose P3 VNC, hit enter



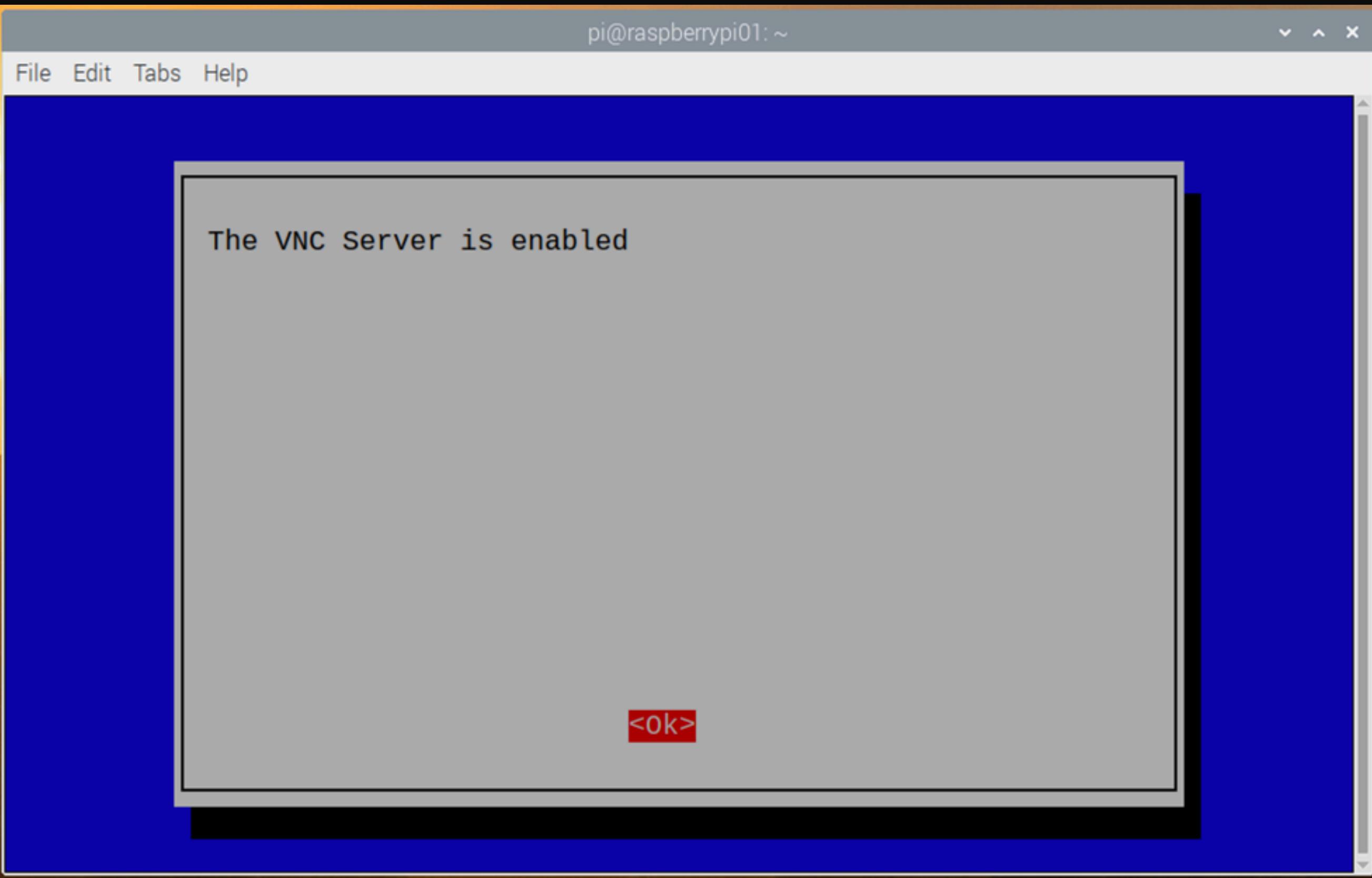
Remote Access -Setup

- Select Yes, hit enter



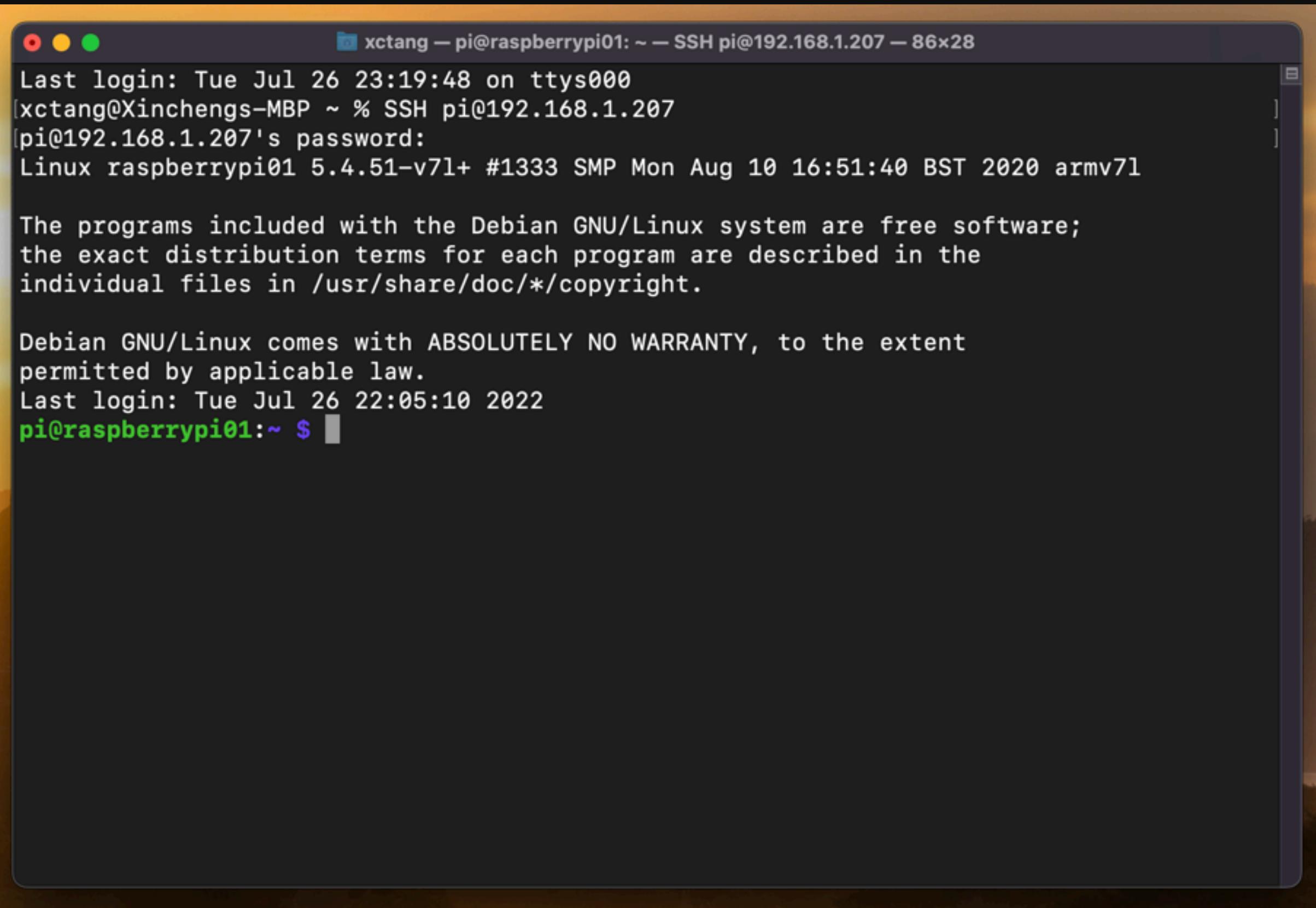
Remote Access -Setup

- You will see screen which confirm VNC is enabled



Remote Access-SSH

- SSH on MAC



A screenshot of a macOS terminal window titled "xctang — pi@raspberrypi01: ~ — SSH pi@192.168.1.207 — 86x28". The window shows the following text:

```
Last login: Tue Jul 26 23:19:48 on ttys000
[xctang@Xinchengs-MBP ~ % SSH pi@192.168.1.207
[pi@192.168.1.207's password:
Linux raspberrypi01 5.4.51-v7l+ #1333 SMP Mon Aug 10 16:51:40 BST 2020 armv7l

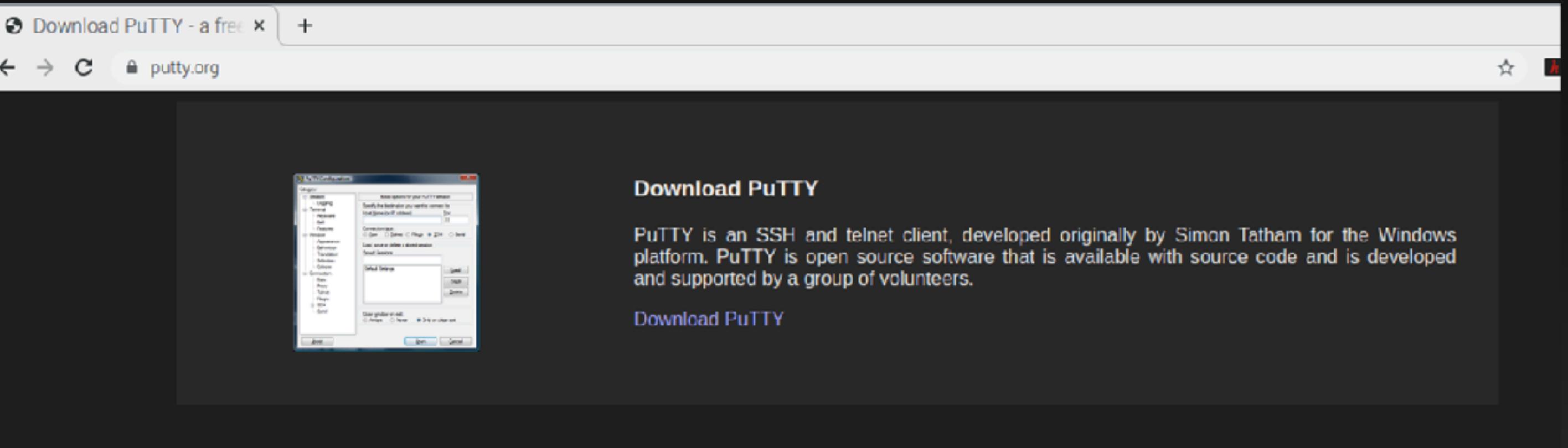
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jul 26 22:05:10 2022
pi@raspberrypi01:~ $
```

Remote Access-SSH

- SSH on Window Machine
- download Putty

[Https://www.putty.org/](https://www.putty.org/)



The screenshot shows a web browser window with the URL [putty.org](https://www.putty.org/) in the address bar. The page displays information about the PuTTY software, including its configuration interface and a download link.

Download PuTTY

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers.

[Download PuTTY](#)

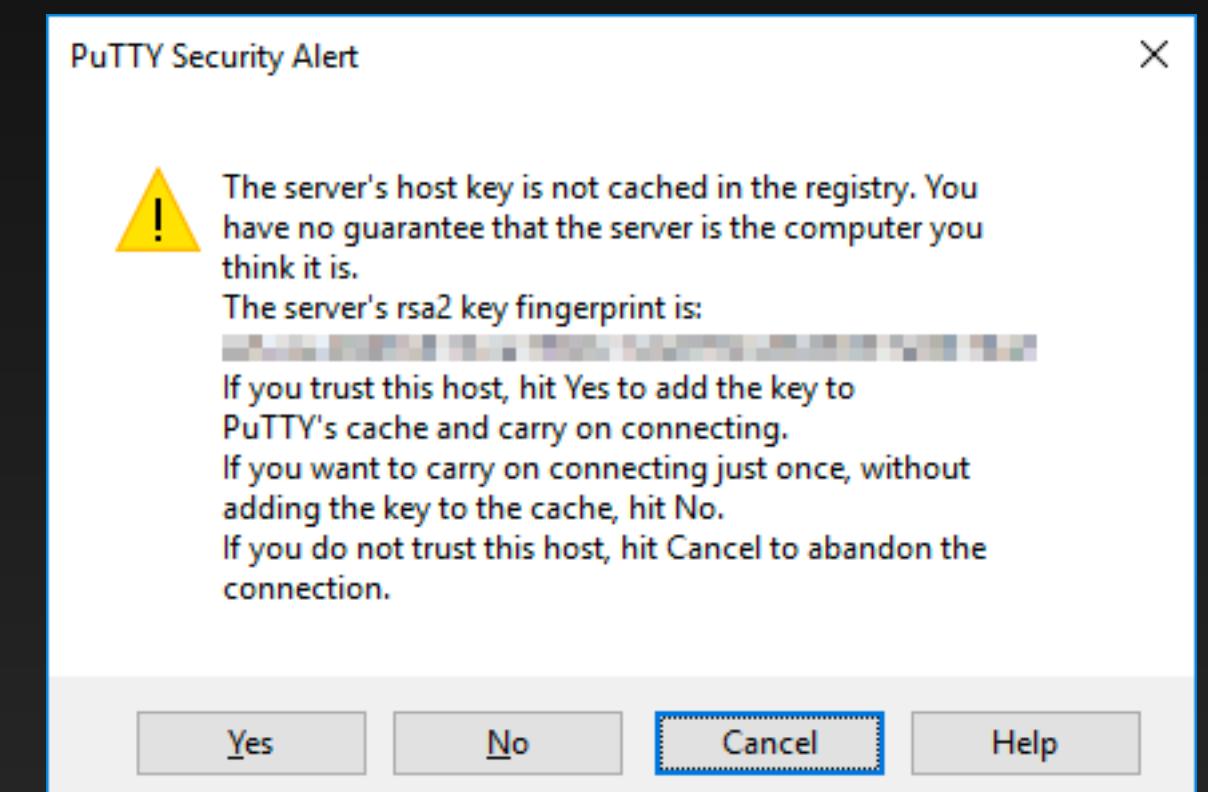
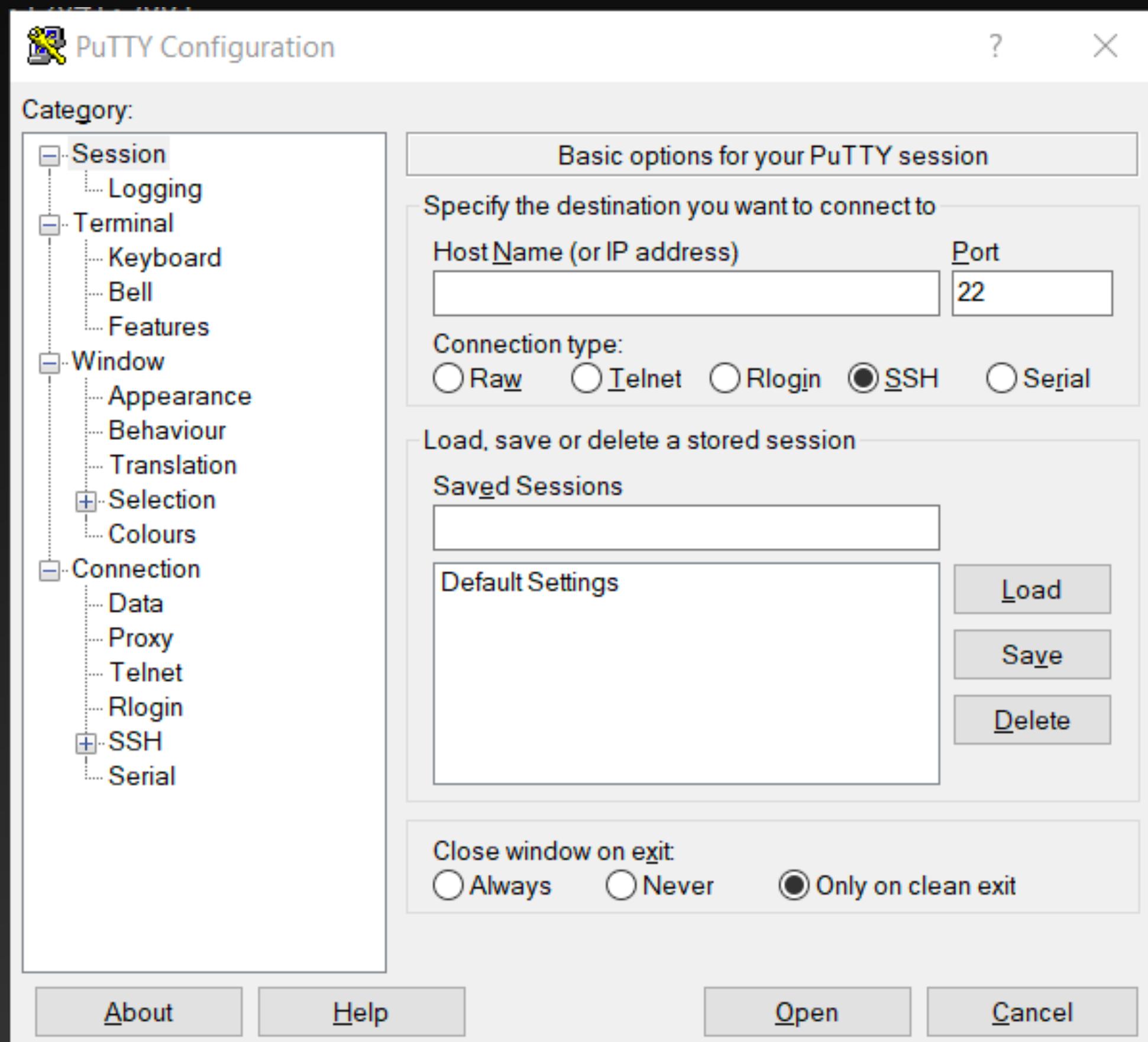
Below suggestions are independent of PuTTY. They are not endorsements by the PuTTY project.

Bitvise SSH Client

Bitvise SSH Client is an SSH and SFTP client for Windows. It is developed and supported professionally by Bitvise. The SSH Client is robust, easy to install, easy to use, and supports all features supported by PuTTY, as well as the

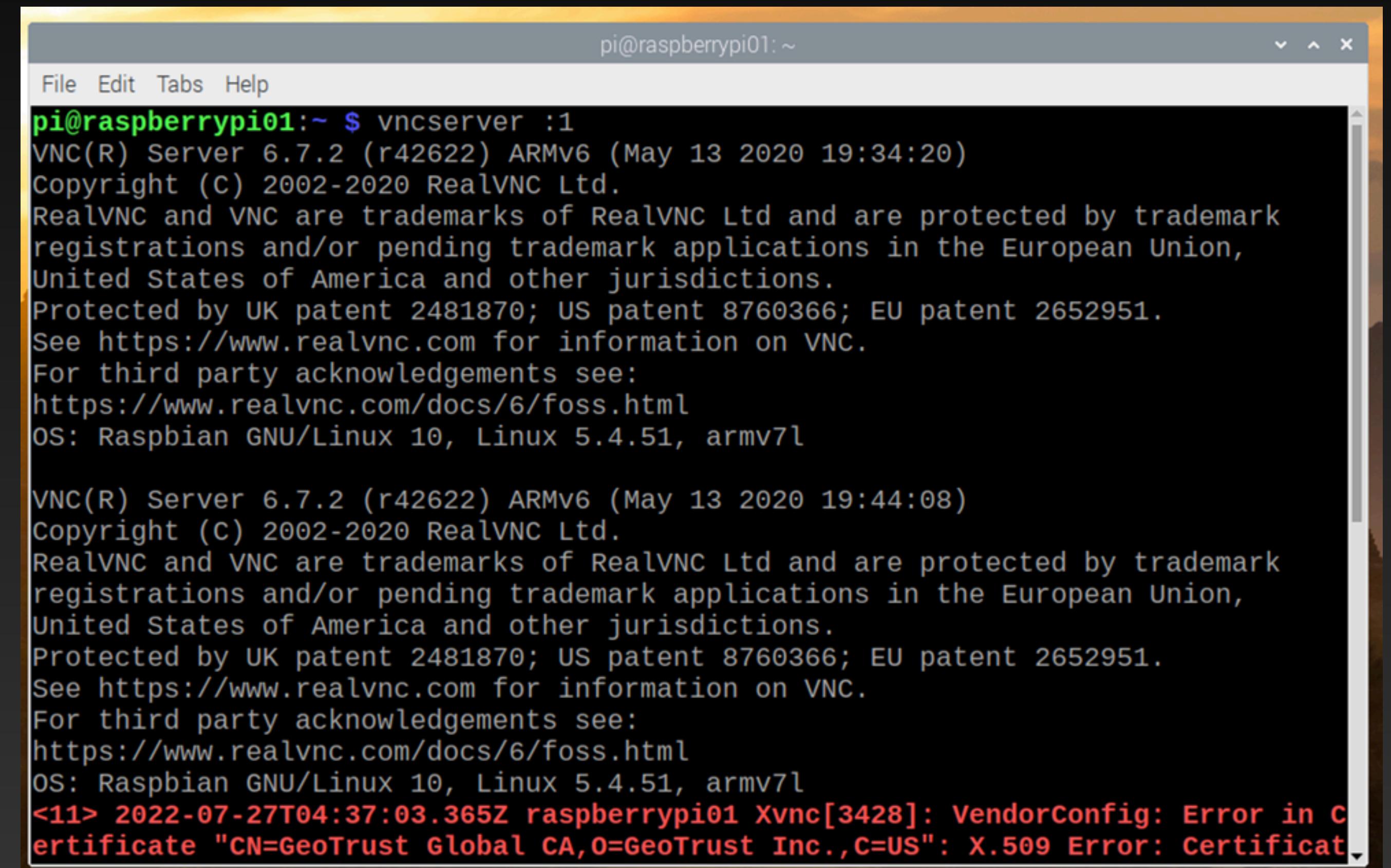
Remote Access-SSH

- Putty config screen
- Enter your IP, click open
- You will see security alert, Click Yes to accept
- You will prompt to enter your Pi username and password



Remote Access-VNC

- VNC consists of two part, VNC server and the VNC Viewer.
- VNC server run on RaspberryPi and VNC viewer run on your Window PC

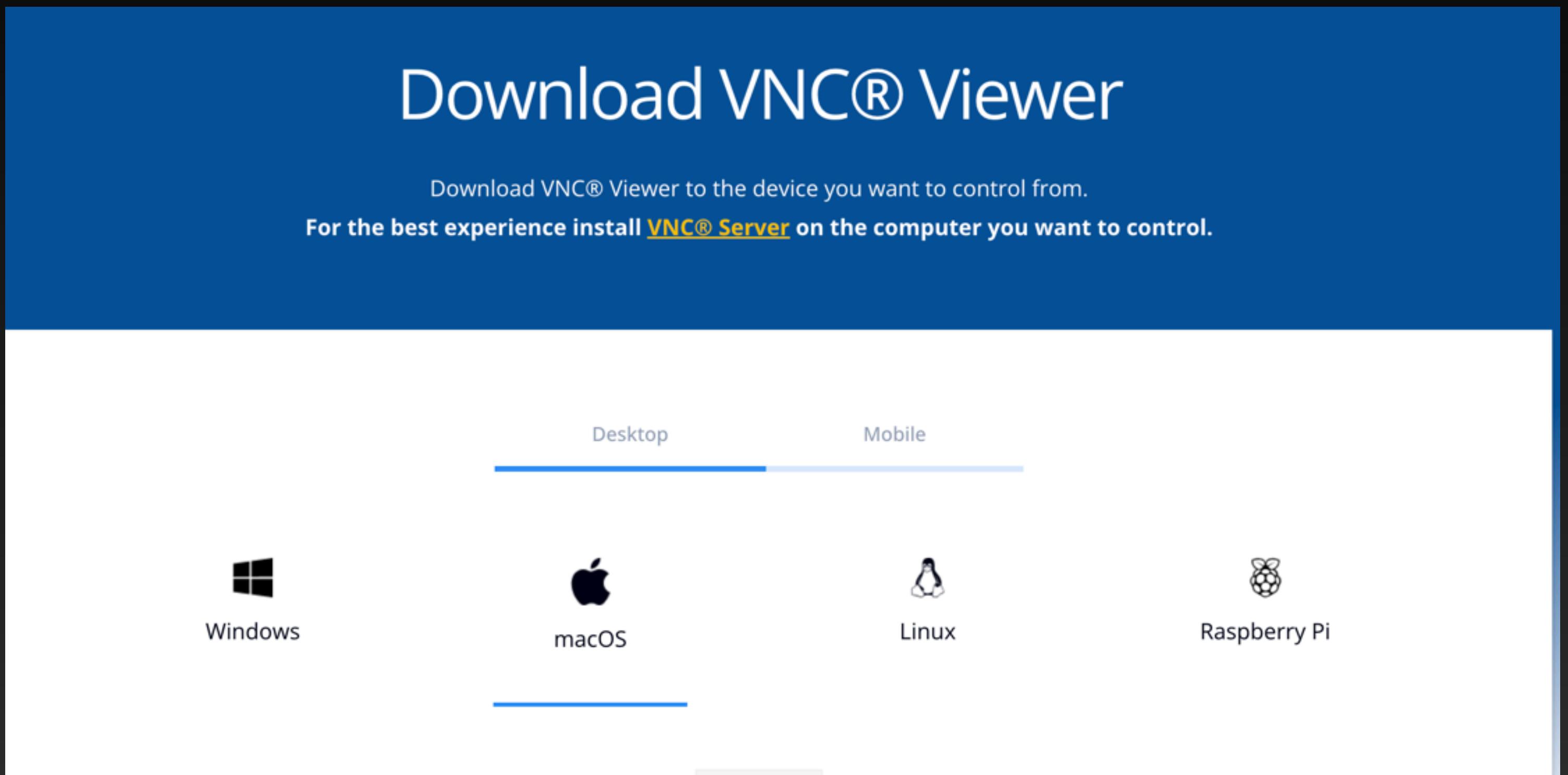


```
pi@raspberrypi01: ~
File Edit Tabs Help
pi@raspberrypi01:~ $ vncserver :1
VNC(R) Server 6.7.2 (r42622) ARMv6 (May 13 2020 19:34:20)
copyright (C) 2002-2020 RealVNC Ltd.
RealVNC and VNC are trademarks of RealVNC Ltd and are protected by trademark
registrations and/or pending trademark applications in the European Union,
United States of America and other jurisdictions.
Protected by UK patent 2481870; US patent 8760366; EU patent 2652951.
See https://www.realvnc.com for information on VNC.
For third party acknowledgements see:
https://www.realvnc.com/docs/6/foss.html
OS: Raspbian GNU/Linux 10, Linux 5.4.51, armv7l

VNC(R) Server 6.7.2 (r42622) ARMv6 (May 13 2020 19:44:08)
Copyright (C) 2002-2020 RealVNC Ltd.
RealVNC and VNC are trademarks of RealVNC Ltd and are protected by trademark
registrations and/or pending trademark applications in the European Union,
United States of America and other jurisdictions.
Protected by UK patent 2481870; US patent 8760366; EU patent 2652951.
See https://www.realvnc.com for information on VNC.
For third party acknowledgements see:
https://www.realvnc.com/docs/6/foss.html
OS: Raspbian GNU/Linux 10, Linux 5.4.51, armv7l
<11> 2022-07-27T04:37:03.365Z raspberrypi01 Xvnc[3428]: VendorConfig: Error in C
ertificate "CN=GeoTrust Global CA,O=GeoTrust Inc.,C=US": X.509 Error: Certificat
```

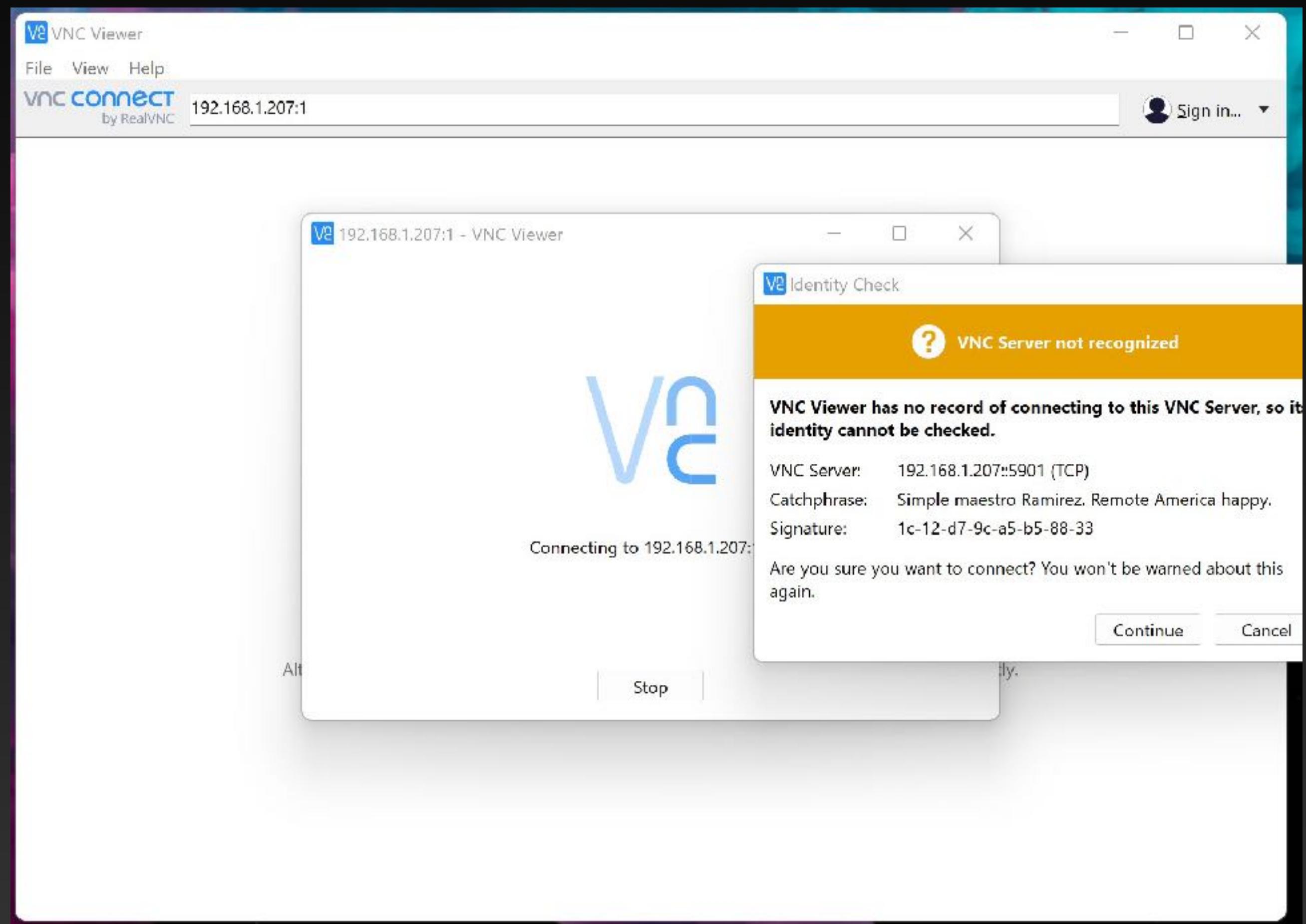
Remote Access-VNC

- [https://www.realvnc.com/
en/connect/download/
viewer/](https://www.realvnc.com/en/connect/download/viewer/)



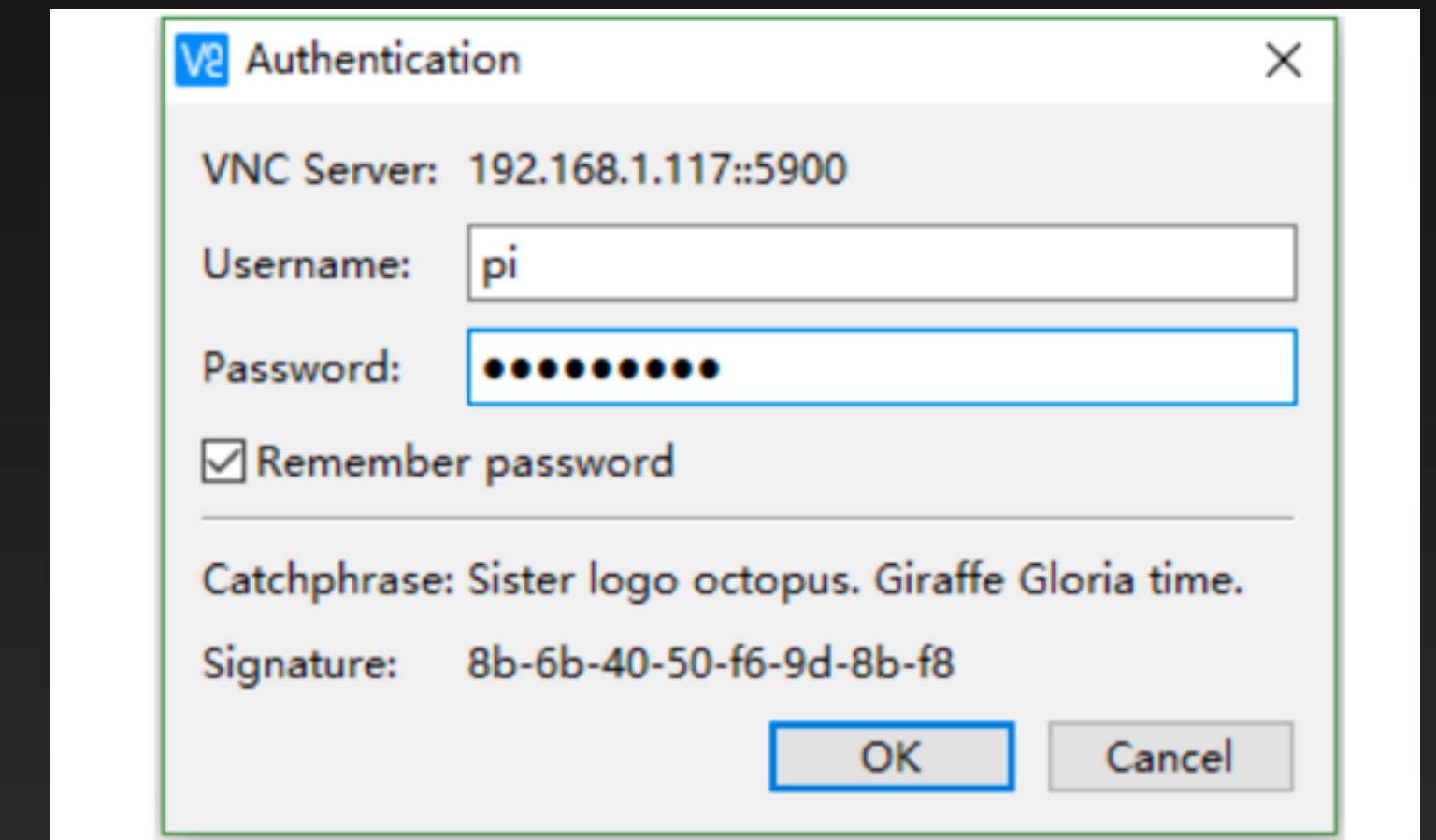
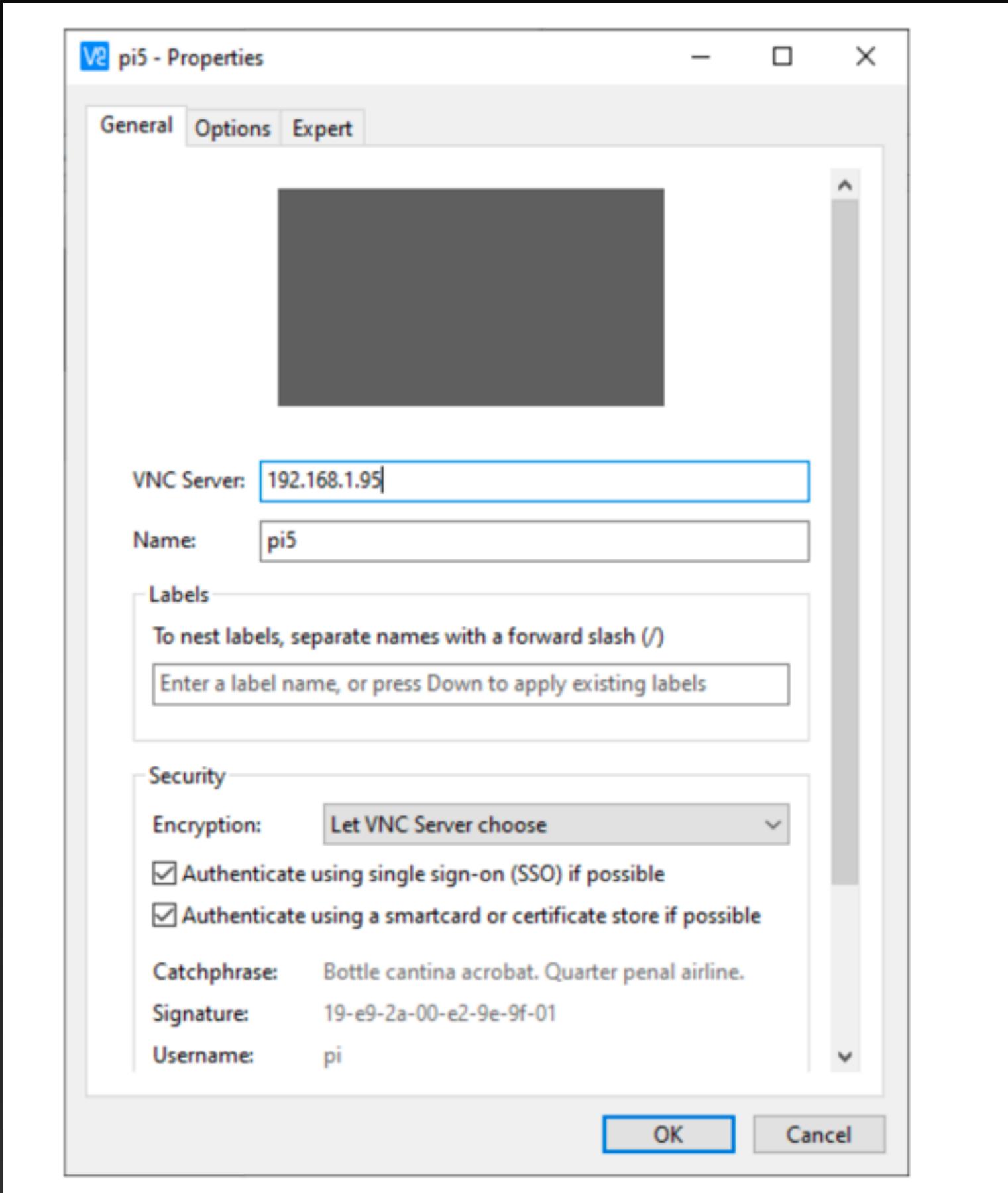
Remote Access-VNC

- Put your Pi IP address in the installed VNC viewer



Remote Access-VNC

- Put your Pi IP address in the installed VNC viewer



Remote Access-VNC

- Enter your username and password

