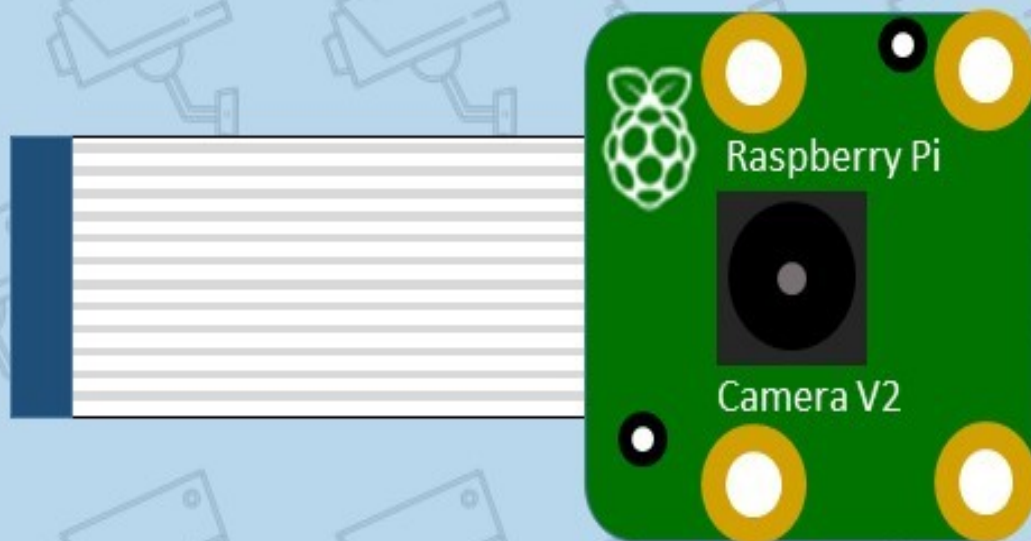


Surveillance Camera with PiCamera and Node-RED



Surveillance Camera with Pi Camera and Node-RED

In this guide you're going to build a low-cost surveillance camera with the Raspberry Pi Camera module and Node-RED. You'll build a web application with Node-RED that streams your video onto the Node-RED Dashboard. This is an easy way to add a surveillance camera to your home automation system.

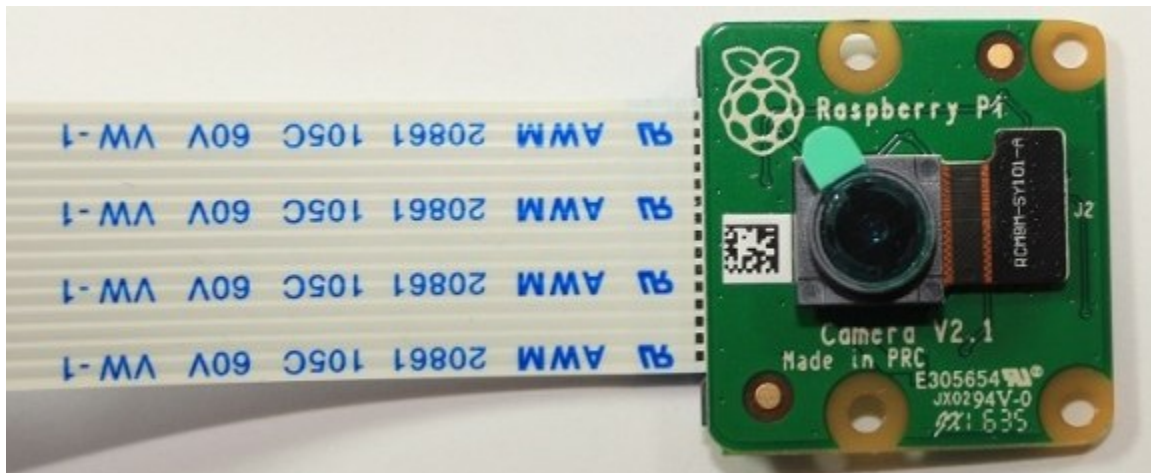
Parts Required

To build your Surveillance Camera with Node-Red you need the following parts (click the parts to compare their price at different stores):

- [Raspberry Pi](#)
- [Raspberry Pi Camera Module V2](#)

Introducing the Raspberry Pi Camera Module

This tutorial was built using the Raspberry Pi Camera Module V2.1, but it also works with other versions of the Raspberry Pi Camera.



[The Raspberry Pi Camera V2 module](#) is an add-on for the Raspberry Pi that you can get for \$20 to \$30. It is really tiny, so it is great to place it anywhere and serve all sorts of different projects. It features an 8 megapixel Sony IMX219 image sensor with fixed focus lens, it is capable of 3280×2464 pixel static images and supports 1080p30, 720p60, and 640×480p90 video. All of this to say it is a pretty good camera for its small size and tiny price.

The camera comes with a 15 cm ribbon used to connect the camera to the Raspberry Pi CSI port. If you need a longer ribbon, you can easily find long ribbon cables on online stores.

Installing the Raspberry Pi Camera

If your Raspberry Pi is running Raspbian Lite, you need to install Pi Camera for Python 3 on your Raspberry Pi. On the terminal, enter the following command to install Pi Camera on Python 3.

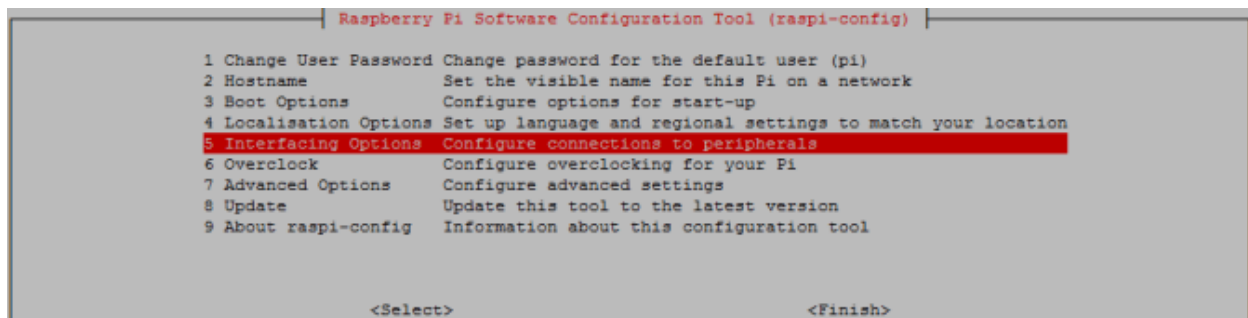
```
pi@raspberrypi:~$ sudo apt install python3-picamera
```

Enabling the Camera

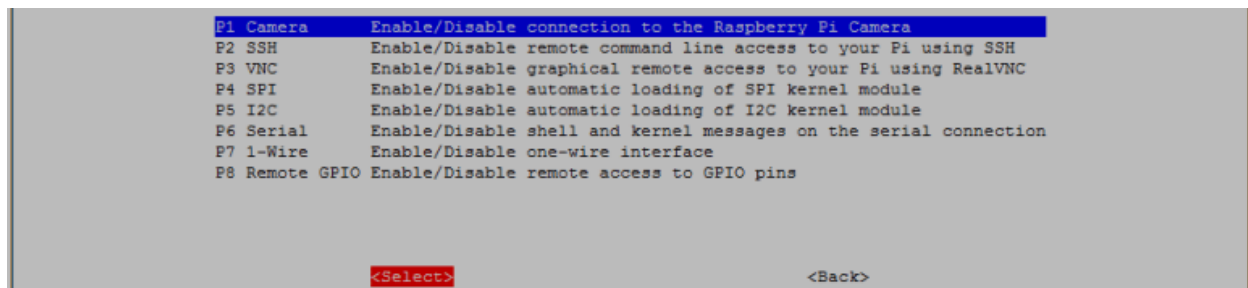
To use the Raspberry Pi Camera module, you need to enable the camera software. Enter the following command:

```
pi@raspberrypi:~$ sudo raspi-config
```

You should see the Raspberry Pi software configuration tool. Select the **Interfacing Options**:

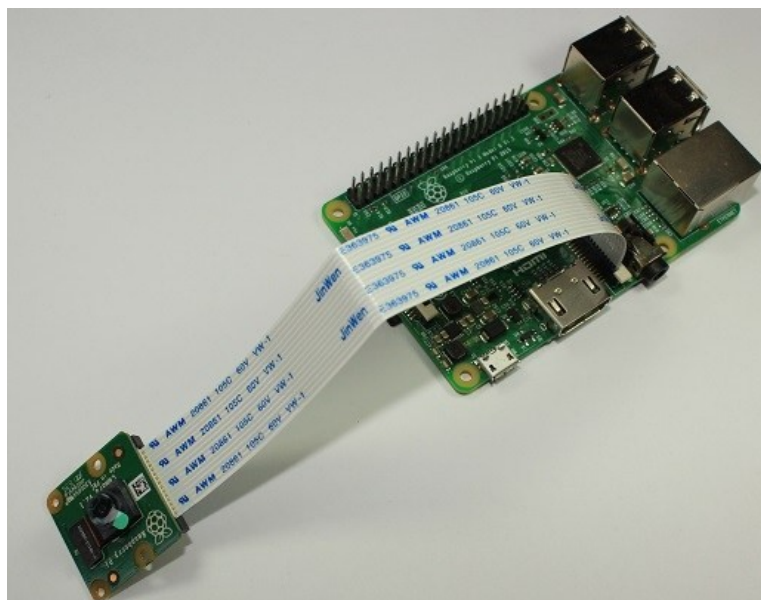


Enable the camera and reboot your Pi for the changes to make effect.



Connecting the Camera

Connecting the Raspberry Pi Camera module is easy. With the Pi shutdown, connect the camera to the Pi CSI port as shown in the following figure. Make sure the camera is connected in the right orientation with the ribbon blue letters facing up.



Writing the Video Streaming Python Script

To stream your video in Node-RED, first you need to create a Python script that creates a video streaming web server. Your video will be streamed on that server and then, you'll embed the video on the Node-RED Dashboard.

Create a new file called **rpi_camera_surveillance_system.py**:

```
pi@raspberrypi:~ $ nano rpi_camera_surveillance_system.py
```

Copy the following code to your newly created file:

DOWNLOAD SOURCE CODE

https://github.com/RuiSantosdotme/Home-Automation-Course/blob/master/code/rpi_camera_surveillance_system.py

```
# Web streaming example
# Source code from the official PiCamera package
# http://picamera.readthedocs.io/en/latest/recipes2.html#web-streaming

import io
import picamera
import logging
import socketserver
from threading import Condition
from http import server

PAGE="""\
<html>
<head>
<title>Raspberry Pi - Surveillance Camera</title>
</head>
<body>
<center><h1>Raspberry Pi - Surveillance Camera</h1></center>
```

```

<center></center>
</body>
</html>
"""

class StreamingOutput(object):
    def __init__(self):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = Condition()

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            # New frame, copy the existing buffer's content and notify
            all

            # clients it's available
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
            return self.buffer.write(buf)

class StreamingHandler(server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/':
            self.send_response(301)
            self.send_header('Location', '/index.html')
            self.end_headers()
        elif self.path == '/index.html':
            content = PAGE.encode('utf-8')
            self.send_response(200)
            self.send_header('Content-Type', 'text/html')
            self.send_header('Content-Length', len(content))
            self.end_headers()
            self.wfile.write(content)

```

```

        elif self.path == '/stream.mjpg':
            self.send_response(200)
            self.send_header('Age', 0)
            self.send_header('Cache-Control', 'no-cache, private')
            self.send_header('Pragma', 'no-cache')
            self.send_header('Content-Type', 'multipart/x-mixed-
replace; boundary=FRAME')
            self.end_headers()
            try:
                while True:
                    with output.condition:
                        output.condition.wait()
                        frame = output.frame
                        self.wfile.write(b'--FRAME\r\n')
                        self.send_header('Content-Type', 'image/jpeg')
                        self.send_header('Content-Length', len(frame))
                        self.end_headers()
                        self.wfile.write(frame)
                        self.wfile.write(b'\r\n')
            except Exception as e:
                logging.warning(
                    'Removed streaming client %s: %s',
                    self.client_address, str(e))
        else:
            self.send_error(404)
            self.end_headers()

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
    allow_reuse_address = True
    daemon_threads = True

with picamera.PiCamera(resolution='640x480', framerate=24) as camera:
    output = StreamingOutput()
    #Uncomment the next line to change
    #your Pi's Camera rotation (in degrees)
    #camera.rotation = 90

```



```
camera.start_recording(output, format='mjpeg')
try:
    address = ('', 8000)
    server = StreamingServer(address, StreamingHandler)
    server.serve_forever()
finally:
    camera.stop_recording()
```

To save your file press **CTRL+X**, type **Y** and hit **Enter**.

Testing the Video Streaming

After writing and saving the Python script, run it using Python 3 with the following command:

```
pi@raspberrypi:~ $ python3 rpi_camera_surveillance_system.py
```

Once the script is running, you can access your video streaming web server at: **http://<Your_Pi_IP_Address>:8000**. Replace with your own Raspberry Pi IP address, in my case **http://192.168.1.112:8000**.

You can access the video streaming through any device that has a browser and is connected to the same network that your Pi.

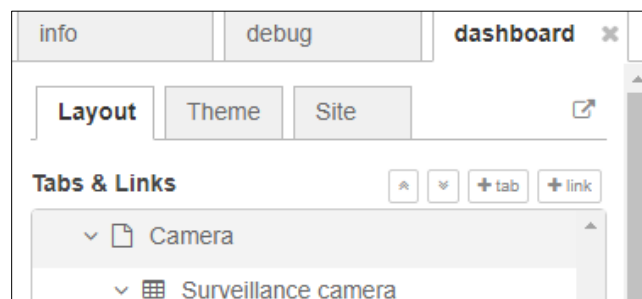


After testing the video streaming and ensuring it works properly, you can move to the Node-RED.

Creating the Node-RED Dashboard Layout

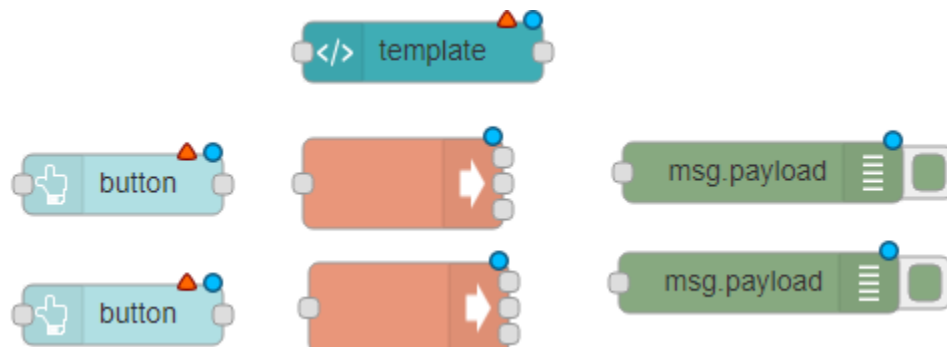
After having Node-RED running, go to your Node-RED dashboard at **http://<Your_RPi_IP_address>:1880**.

In the top right corner of the Node-RED window, select the **dashboard** tab. Under the **dashboard** tab select the **layout** tab. Create a new tab called **Camera**. Under that tab, create a new group called **Surveillance camera**. This is where you'll place your dashboard widgets.



Creating the Node-RED Flow

Add the following nodes to the flow: 1 **template**, 2 **buttons**, 2 **exec**, and 2 **debug** nodes, as shown in the figure below.



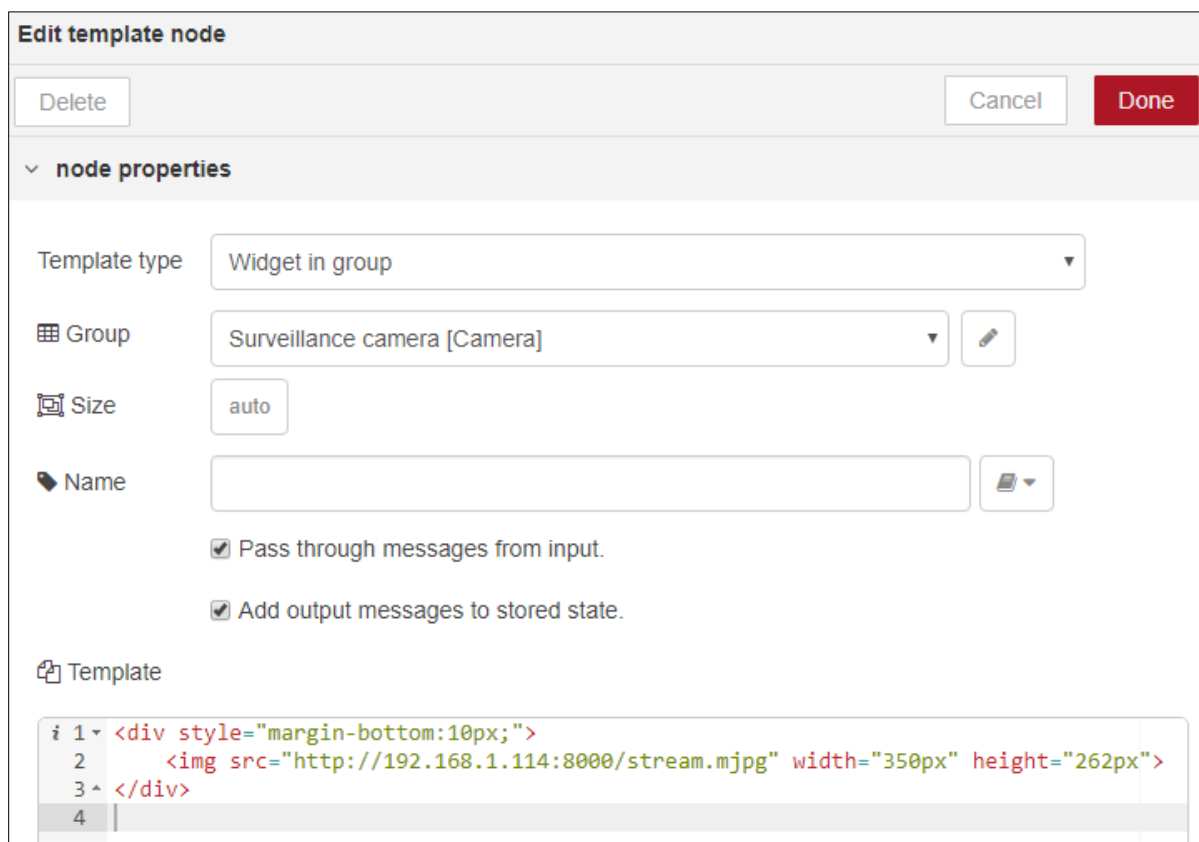
Briefly, here's what this flow aims to achieve:

- When you click one of the buttons, the exec node runs the **rpi_camera_surveillance_system.py** file you've created earlier, so that your camera starts streaming.
- The template node will embed the video that is being streamed on the streaming web server on the Node-RED Dashboard
- The other button will trigger the other exec node that will stop the video streaming by stopping the **rpi_camera_surveillance_system.py** file.

Now, you just need to edit the nodes to do those tasks.

Template node

Double-click on the template node, and a new window should open. Edit the node with these properties:



Edit template node

Delete Cancel Done

▼ node properties

Template type: Widget in group

Group: Surveillance camera [Camera]

Size: auto

Name:

☒ Pass through messages from input.

☒ Add output messages to stored state.

Template

```
1 <div style="margin-bottom:10px;">
2   
3 </div>
4
```

Copy the following HTML to the template section.

```
<div style="margin-bottom:10px;">
  
</div>
```

Don't forget to edit the code with your own IP address where the video streaming web server is running. In my case, the template code looks as follows:

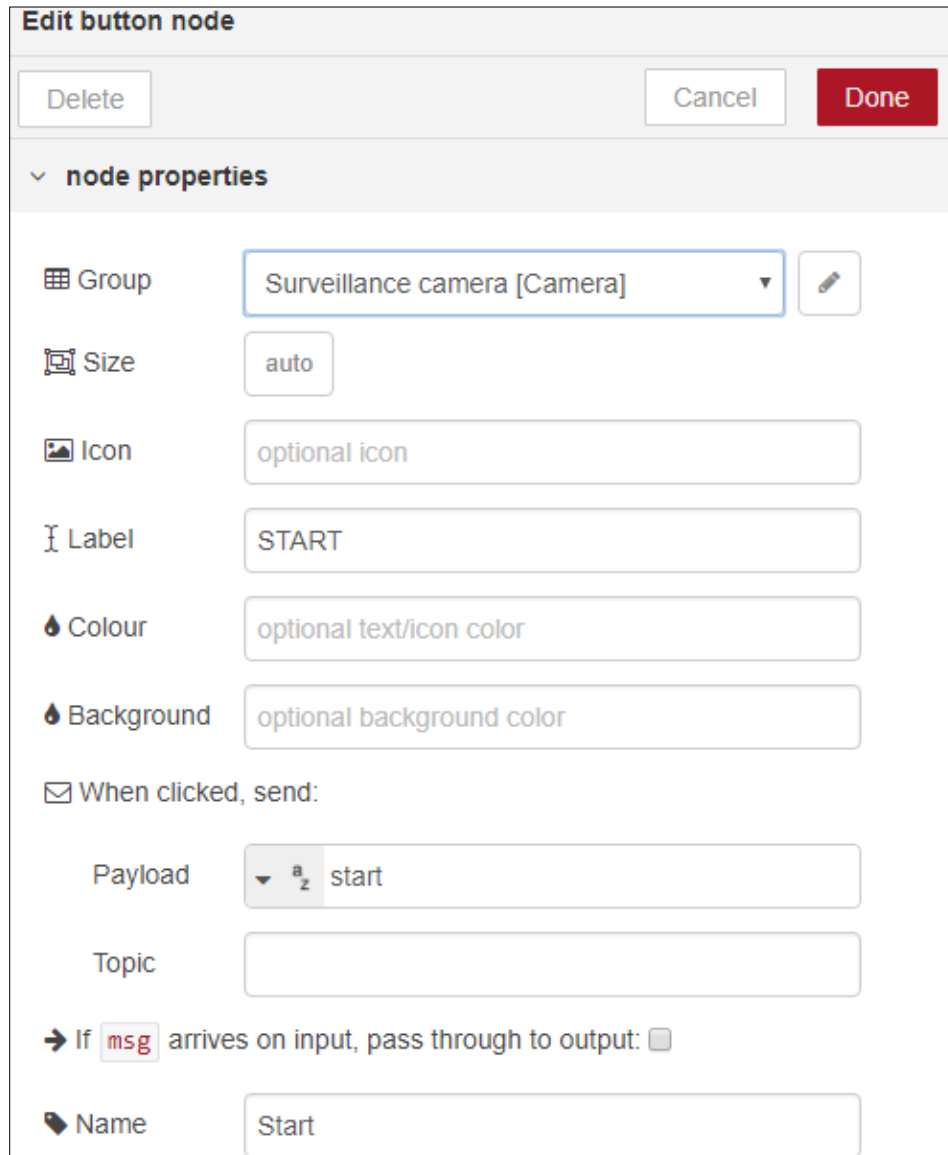
```
<div style="margin-bottom:10px;">
  
</div>
```

This template simply embeds the video on the streaming web server onto the Node-RED Dashboard. In this template, you can also edit the **width="350px"** and **height="262px"** of the video by changing the numbers highlighted in red.

Note: we're embedding the video using the HTML tag ``. This tag is used to embed images, but it also works with the .mjpg video format, because it's simply a sequence of several images.

Button nodes

Double click the first button. This will be the button that will trigger the video streaming. Let's call it **"START"** and edit its properties as shown below.



The screenshot shows the 'Edit button node' dialog box with the following configuration:

- Buttons:** Delete, Cancel, Done
- node properties:**
 - Group:** Surveillance camera [Camera] (dropdown menu)
 - Size:** auto
 - Icon:** optional icon
 - Label:** START
 - Colour:** optional text/icon color
 - Background:** optional background color
 - When clicked, send:**
 - Payload:** start (dropdown menu)
 - Topic:**
 - If msg arrives on input, pass through to output:** ☐
 - Name:** Start

When you click this button, it will send a payload message with the text "start".

You also need a button to stop the video streaming. So, you need a **STOP** button. Edit the other button with the following properties.

Edit button node

Delete

Cancel

Done

▼ node properties

Group

Surveillance camera [Camera]

Size

auto

Icon

optional icon

Label

STOP

Colour

optional text/icon color

Background

optional background color

☒ When clicked, send:

Payload

▼ a_z stop

Topic

→ If **msg** arrives on input, pass through to output:

☐

Name

Stop

When you click on this button it will send a payload message with the text "stop".

[LATEST PROJECTS](#) – [DOWNLOAD OTHER RNT PRODUCTS](#) – [ASK QUESTIONS](#)

13

Exec nodes

Edit the first **exec** node. This will start the video streaming when the **START** button is pressed.

Edit exec node

Delete Cancel Done

▼ node properties

Command

+ Append ☐ msg.payload

↔ Output ▼

☐ Use old style output (compatibility mode)

⌚ Timeout seconds

🏷 Name

This node simply runs the **rpi_camera_surveillance_system.py** file using Python 3 when triggered (when you click on the START button).

Next, edit the other **exec** node that will stop the **rpi_camera_surveillance_system.py** file to stop the video streaming when the **STOP** button is pressed.

Edit exec node

Delete
Cancel
Done

node properties

Command

+ Append
☐ msg.payload

Output

☐ Use old style output (compatibility mode)

Timeout
 seconds

Name

Debug nodes

The debug nodes are useful for debugging purposes. Double-click the debug nodes and you can leave the default settings, as shown below.

Edit debug node

Delete
Cancel
Done

node properties

Output

to

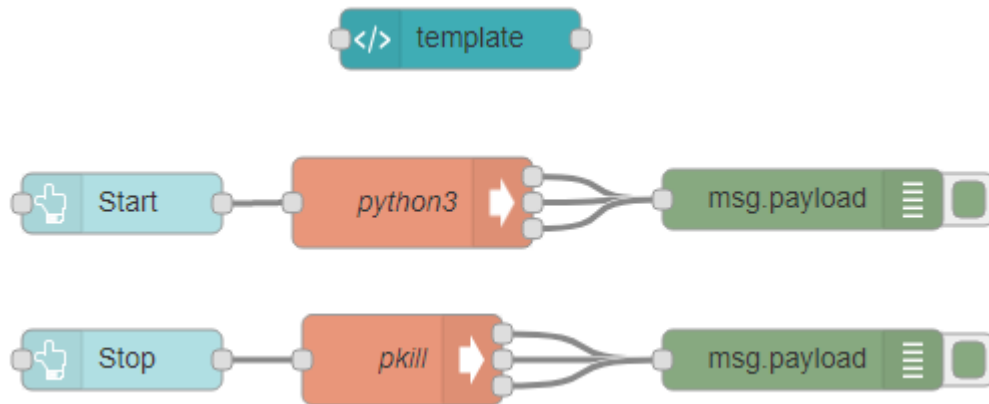
Name

[LATEST PROJECTS](#) – [DOWNLOAD OTHER RNT PRODUCTS](#) – [ASK QUESTIONS](#)

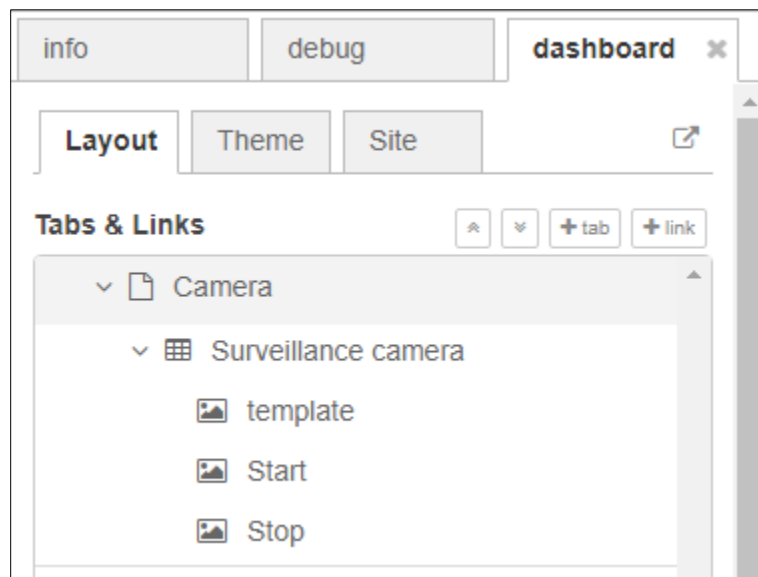
15

Wiring the Flow

Finally, wire all the nodes as shown below.



And here's how you dashboard layout should look like with all the nodes ready.



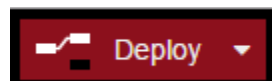
You can also go to the **site** tab and edit the size of your template widget, which is the widget where the video is – take a look at the figure below. We've increased the horizontal size so the video fits better. You may change those values to better suit your needs.

The screenshot shows the Node-RED Dashboard configuration panel. At the top, there are tabs for 'info', 'debug', and 'dashboard' (which is active). Below the tabs are three sub-tabs: 'Layout', 'Theme', and 'Site' (which is active). The 'Site' tab contains the following settings:

- Title:** A text input field containing 'Node-RED Dashboard'.
- Options:** Two dropdown menus. The first is 'Show the title bar' and the second is 'No swipe between tabs'.
- Date Format:** A text input field containing 'DD/MM/YYYY'.
- Sizes:** A table with two columns: 'Horizontal' and 'Vertical'. The rows are '1x1 Widget Size', 'Widget Spacing', 'Group Padding', and 'Group Spacing'.

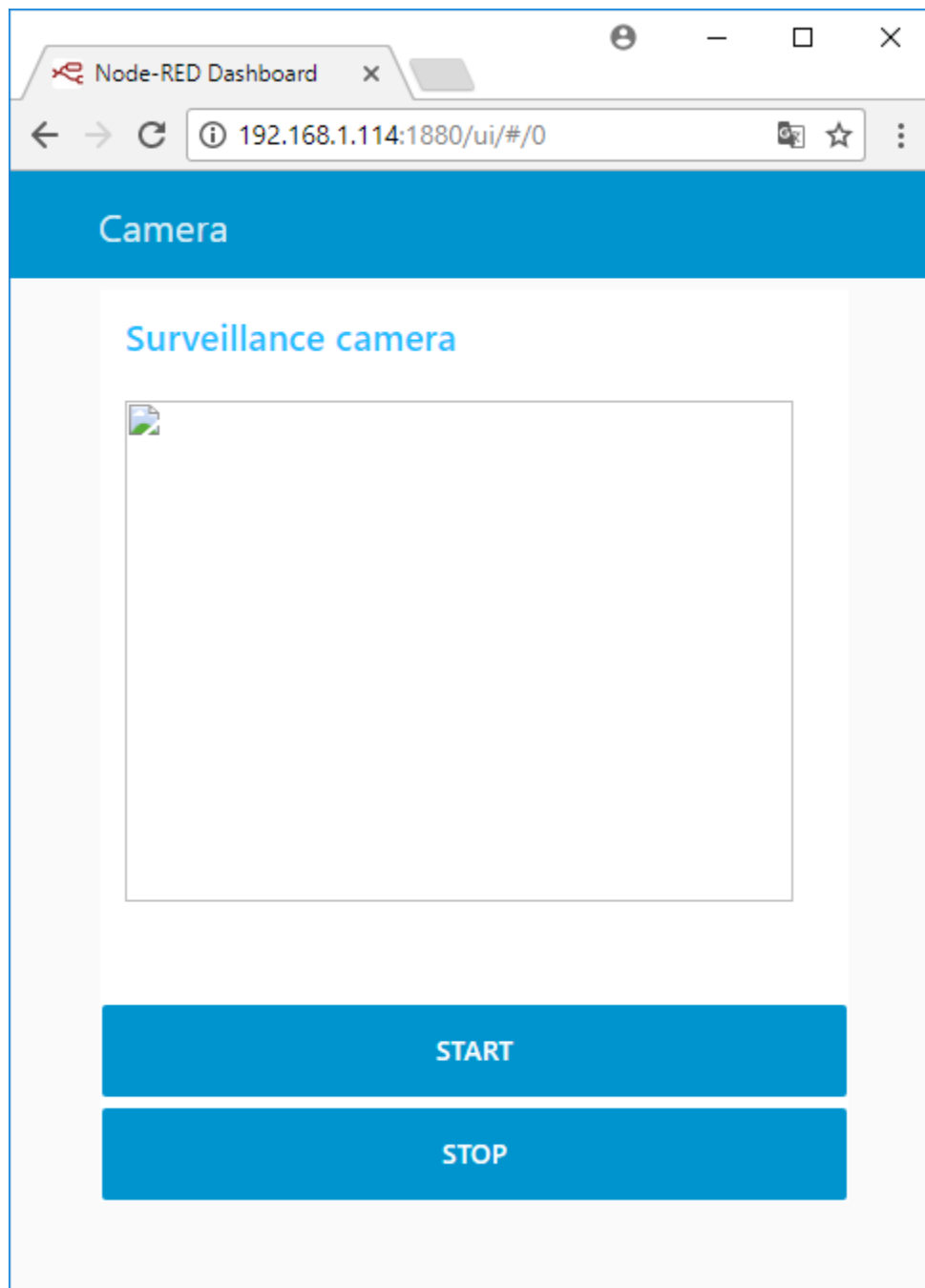
Sizes	Horizontal	Vertical
1x1 Widget Size	60	48
Widget Spacing	6	6
Group Padding	0	0
Group Spacing	6	6

Click the **Deploy** button to save the changes and run the application.

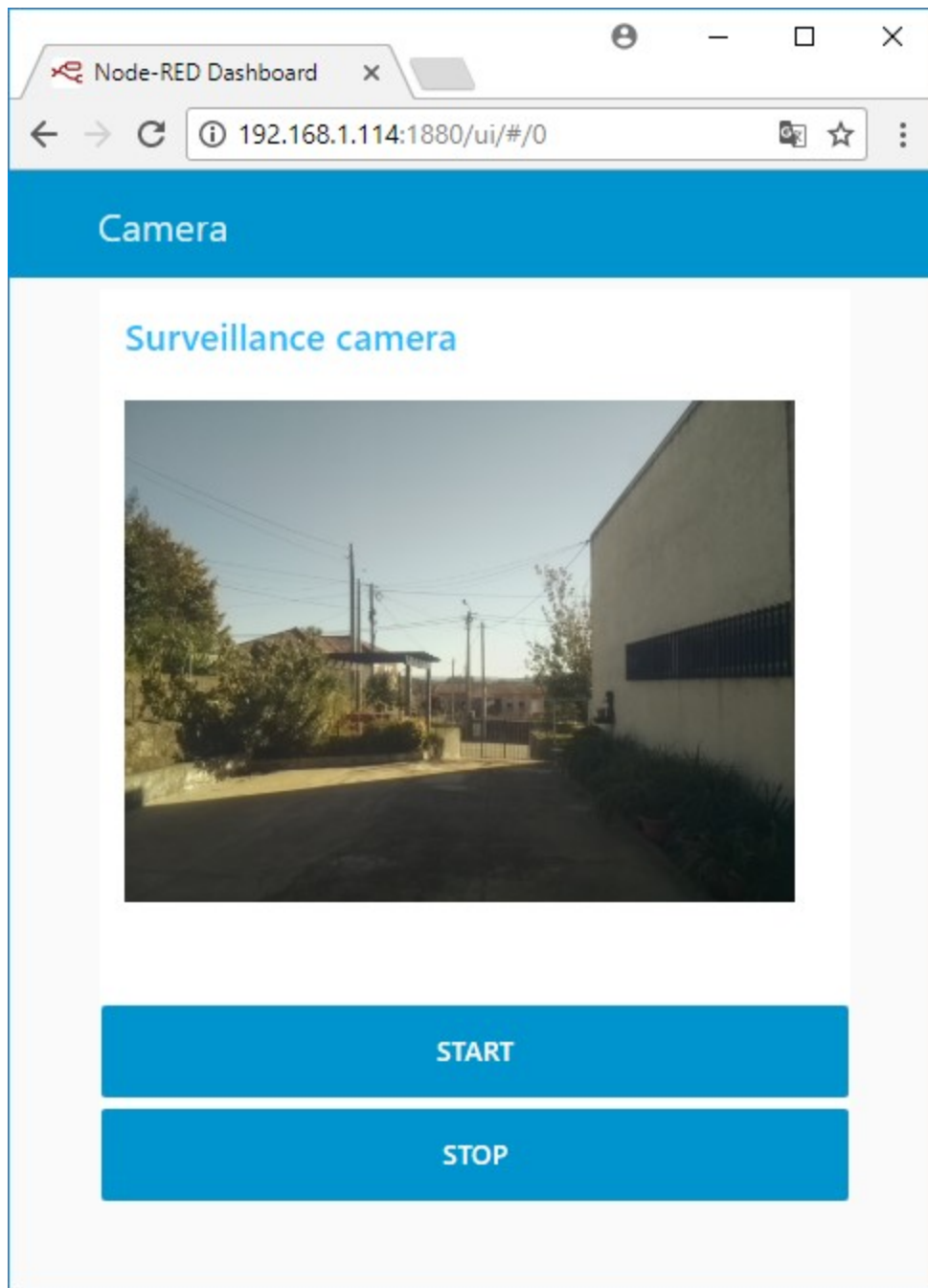


Testing your Node-RED Application

Congratulations! Your surveillance system Node-RED application is ready. Go to **[http:// <Your_RPi_IP_address>:1880/ui](http://<Your_RPi_IP_address>:1880/ui)** and see how your Node-RED application looks. It should look like something as in the following figure.



Now tap the **START** button to start the video streaming and **refresh** the Node-RED Dashboard page to watch the camera streaming.



To stop the video streaming, you simply need to tap the **STOP** button.

Taking It Further

You can build a fully home surveillance system with several Raspberry Pi cameras with this system. Here's what I suggest:

1. We recommend using several [Raspberry Pi Zero W boards](#) with their own [Raspberry Pi camera](#) attached
2. Set the **rpi_camera_surveillance_system.py** script to run automatically when the Raspberry Pi Zero W boards boot.
3. Take note of each of the Raspberry Pi IP addresses so that you know the URL where each video streaming is running.
4. Use one Raspberry Pi as the main server where you'll host the Node-RED application.
5. Add several template nodes to your Node-RED flow, and edit each of them to stream Pi Camera running on the other Pi boards, by using their IP addresses.