Internet of Things 101: Building IoT Prototype with Raspberry Pi

Feb 9 and 11, 2016 at Forward 4 Conf
Tomomi (@girlie_mac)

Bhavana (@bhavanalllo)
#ForwardJS
While you are waiting...

Connect Wi-Fi

Wi-Fi Network: NewCircle Student

Password: opensource99

* This requires you to sign in from a captive portal
https://github.com/pubnub/workshop-raspberrypi
You should have:

- Raspberry Pi 2
- Micro SD card (preloaded with Raspbian)
- Mini Wi-Fi adapter
- Micro USB power supply
- HDMI Cable
- Wires
- Breadboard
- LED
- Resistors
- PIR sensor
- DHT22 sensor
What You Will Learn Today

1. How to start up Raspberry Pi & start Raspbian OS
2. How to connect the Pi remotely from your laptop
3. How to send & receive data with PubNub using Python
4. How to wire a LED & resistor to Pi using breadboard
5. How to program Pi to blink the LED
6. The First IoT : Remote-controlled LED from web interface
7. Projects: Using sensors (Work on your own)
1. Powering the Raspberry Pi
Setting up Your Pi

1. SD CARD
2. WI-FI ADAPTER
3. TO KEYBOARD
4. TO MOUSE
5. TO MONITOR
6. USB TO POWER SOURCE
Starting Raspbian

Username: pi
Password: raspberry
pi@raspberrypi  ~$ startx

(Don’t worry about the error dialog. Just dismiss it!)
Wi-Fi Configuration

Menu > Preference > WiFi Configuration

Wi-Fi Network: NewCircle Student   Password (PSK): opensource99
2. Remote

Connect the Pi

https://github.com/pubnub/workshop-raspberrypi/blob/master/remote-vnc.md
Remote Connect Pi

Getting your Pi’s IP address

```
pi@raspberrypi ~$ hostname -I
```

🌟 You’ll need the IP address when you connect the Pi from your computer!!!
Remote Connect Pi

You can choose from the followings:

- SSH to your Pi from terminal, and keep working on the terminal
- SSH with a client, e.g. Cyberduck, and use your usual IDE
- Use VNC (Virtual Network Computing), and work on the virtual GUI
SSH into your Rasp Pi

SSH to Pi from your laptop

(Terminal on Mac/Linux, PuTTY on Windows):

me@MyMac ~$ ssh pi@10.96.70.1

Use your Pi's IP!

Your Pi's username

If SSH-ing fails, try:

$ sudo raspi-config on your Pi
Remote Access w/ VNC

Remote-access to Raspberry Pi’s graphical interface.

1. On your laptop:
   - Install VNC Viewer (Client)
   - \text{http://www.realvnc.com/download/viewer/}

2. On your Pi (Either by SSH, or directly):
   - Install VNC server
Remote Access w/ VNC

Install **Tight VNC Server**

```
pi@raspberrypi
  ~$ sudo apt-get install tightvncserver
```

Run the server

```
pi@raspberrypi  ~$ tightvncserver
```
Remote Access w/ VNC

Run the client

Pi's IP address

usually :1 (but it can be different. Check the VNC server!)
SSH w/ Cyberduck

- No virtual GUI, but faster than using VNC
- You can use your fave IDE to edit files
- Mac users can keep using Mac keyboard shortcuts

Download the client at: https://cyberduck.io/
SSH w/ Cyberduck
3. Get Started w/ PubNub Python SDK
Update System First

Update the System’s package list

~$ sudo apt-get update

Upgrade the installed packages to the latest versions

~$ sudo apt-get upgrade
Get Started w/ Python

Install python and pip

```bash
~$ sudo apt-get install python-dev
~$ sudo apt-get install python-pip
```
Get Started w/ PubNub

Install pubnub libs

~$ sudo pip install pubnub
Hello World w/ PubNub

Hello World w/ PubNub

Import & init (hello.py)

```python
import sys
from pubnub import Pubnub

pubnub = Pubnub(publish_key='pub-c-123...',
    subscribe_key='sub-c-456...')
```
Hello World w/ PubNub

Publish (Sending data)

channel = 'hello-pi'
data = { 'username': 'SpongeBob',
       'message': 'Hello world from Pi!' }

def callback(m):
    print(m)

pubnub.publish(channel, data, callback=callback,
    error=callback)
Hello World w/ PubNub

Run your program

~$ sudo python hello.py
Hello World w/ PubNub

Subscribing data you are publishing

http://pubnub.github.io/workshop-raspberrypi/web/hello.html
3. Using the Debug Console
Debug Console

http://pubnub.com/console/

1. channel: hello-pi
2. pub key: demo
3. sub key: demo
4. Blinking LED

https://github.com/pubnub/workshop-raspberrypi/tree/master/projects-python/led
Blinking LED

- Raspberry Pi 2
- 1 LED (1.9 - 3.2V)
- 1 Resistor (200Ω)
- 1 Breadboard
- 2 M-to-F jumper wires, 2 colors
OMG Physics!

1.9V 3.2V

- + - +

Cathode (-) Anode (+)
OMG Physics!

\[ R = \frac{V_s - V_f}{I} \]

- Source voltage (V)
- Forward voltage (V) (LED voltage drop)
- Resistance (\( \Omega \))
- Current thru the LED (A)
OMG Physics!

\[ R = \frac{3.3v - 1.9v}{0.02 \text{ A}} = 70 \, \Omega \]

- **source voltage (V)**
- **forward voltage (V)** (Red LED voltage drop)
- **resistance (\(\Omega\))**
- **current thru the LED (A)**
4-band Resistor Color Code

4 7 \times 10^2 \pm 5\%

multiplier
tolerance

47 \times 100 = 4.7 \text{ kOhms}

Learn more at: https://learn.adafruit.com/multimeters/resistance
5-band Resistor Color Code

Learn more at: https://learn.adafruit.com/multimeters/resistance
Breadboard

An electronics breadboard is a fundamental tool to build circuits. It is solderless, and a great tool for prototyping.

We are using this kind today.

You may find this type of breadboard when googling circuits. They have power rails that go vertical!

Connected!
Conductive metal strips go horizontally

Mini
not connected!

400-pin
Turning LED on

3.3V (Raspberry Pi)
Turning LED on

3.3V (Pin 1)

GND

Anode (longer leg)

Cathode
# Raspberry Pi 2 Pins

## 3.3V

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.3V Power</td>
</tr>
<tr>
<td>2</td>
<td>3.3V Power</td>
</tr>
<tr>
<td>3</td>
<td>Ground</td>
</tr>
<tr>
<td>4</td>
<td>GPIO10</td>
</tr>
<tr>
<td>5</td>
<td>GPIO11</td>
</tr>
<tr>
<td>6</td>
<td>GPIO12</td>
</tr>
<tr>
<td>7</td>
<td>GPIO13</td>
</tr>
<tr>
<td>8</td>
<td>GPIO14</td>
</tr>
<tr>
<td>9</td>
<td>GPIO15</td>
</tr>
<tr>
<td>10</td>
<td>GPIO16</td>
</tr>
<tr>
<td>11</td>
<td>GPIO17</td>
</tr>
<tr>
<td>12</td>
<td>GPIO18</td>
</tr>
<tr>
<td>13</td>
<td>GPIO19</td>
</tr>
<tr>
<td>14</td>
<td>GPIO20</td>
</tr>
<tr>
<td>15</td>
<td>GPIO21</td>
</tr>
<tr>
<td>16</td>
<td>GPIO22</td>
</tr>
<tr>
<td>17</td>
<td>GPIO23</td>
</tr>
<tr>
<td>18</td>
<td>GPIO24</td>
</tr>
<tr>
<td>19</td>
<td>GPIO25</td>
</tr>
<tr>
<td>20</td>
<td>GPIO26</td>
</tr>
<tr>
<td>21</td>
<td>GPIO27</td>
</tr>
<tr>
<td>22</td>
<td>GPIO28</td>
</tr>
<tr>
<td>23</td>
<td>GPIO29</td>
</tr>
<tr>
<td>24</td>
<td>GPIO30</td>
</tr>
<tr>
<td>25</td>
<td>GPIO31</td>
</tr>
<tr>
<td>26</td>
<td>GPIO32</td>
</tr>
<tr>
<td>27</td>
<td>GPIO33</td>
</tr>
<tr>
<td>28</td>
<td>GPIO34</td>
</tr>
<tr>
<td>29</td>
<td>GPIO35</td>
</tr>
<tr>
<td>30</td>
<td>GPIO36</td>
</tr>
<tr>
<td>31</td>
<td>GPIO37</td>
</tr>
<tr>
<td>32</td>
<td>GPIO38</td>
</tr>
<tr>
<td>33</td>
<td>GPIO39</td>
</tr>
<tr>
<td>34</td>
<td>GPIO40</td>
</tr>
</tbody>
</table>

## GPIO (general purpose input output)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GPIO0</td>
</tr>
<tr>
<td>2</td>
<td>GPIO1</td>
</tr>
<tr>
<td>3</td>
<td>GPIO2</td>
</tr>
<tr>
<td>4</td>
<td>GPIO3</td>
</tr>
<tr>
<td>5</td>
<td>GPIO4</td>
</tr>
<tr>
<td>6</td>
<td>GPIO5</td>
</tr>
<tr>
<td>7</td>
<td>GPIO6</td>
</tr>
<tr>
<td>8</td>
<td>GPIO7</td>
</tr>
<tr>
<td>9</td>
<td>GPIO8</td>
</tr>
<tr>
<td>10</td>
<td>GPIO9</td>
</tr>
<tr>
<td>11</td>
<td>GPIO10</td>
</tr>
<tr>
<td>12</td>
<td>GPIO11</td>
</tr>
<tr>
<td>13</td>
<td>GPIO12</td>
</tr>
<tr>
<td>14</td>
<td>GPIO13</td>
</tr>
<tr>
<td>15</td>
<td>GPIO14</td>
</tr>
<tr>
<td>16</td>
<td>GPIO15</td>
</tr>
<tr>
<td>17</td>
<td>GPIO16</td>
</tr>
<tr>
<td>18</td>
<td>GPIO17</td>
</tr>
<tr>
<td>19</td>
<td>GPIO18</td>
</tr>
<tr>
<td>20</td>
<td>GPIO19</td>
</tr>
<tr>
<td>21</td>
<td>GPIO20</td>
</tr>
<tr>
<td>22</td>
<td>GPIO21</td>
</tr>
<tr>
<td>23</td>
<td>GPIO22</td>
</tr>
<tr>
<td>24</td>
<td>GPIO23</td>
</tr>
<tr>
<td>25</td>
<td>GPIO24</td>
</tr>
<tr>
<td>26</td>
<td>GPIO25</td>
</tr>
<tr>
<td>27</td>
<td>GPIO26</td>
</tr>
<tr>
<td>28</td>
<td>GPIO27</td>
</tr>
<tr>
<td>29</td>
<td>GPIO28</td>
</tr>
<tr>
<td>30</td>
<td>GPIO29</td>
</tr>
<tr>
<td>31</td>
<td>GPIO30</td>
</tr>
<tr>
<td>32</td>
<td>GPIO31</td>
</tr>
<tr>
<td>33</td>
<td>GPIO32</td>
</tr>
<tr>
<td>34</td>
<td>GPIO33</td>
</tr>
<tr>
<td>35</td>
<td>GPIO34</td>
</tr>
<tr>
<td>36</td>
<td>GPIO35</td>
</tr>
<tr>
<td>37</td>
<td>GPIO36</td>
</tr>
<tr>
<td>38</td>
<td>GPIO37</td>
</tr>
<tr>
<td>39</td>
<td>GPIO38</td>
</tr>
<tr>
<td>40</td>
<td>GPIO39</td>
</tr>
</tbody>
</table>
Programming LED

GPIO-4 (Pin 7)
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
LED = 4
GPIO.setup(LED, GPIO.OUT)

while True:
    GPIO.output(LED, True)
    time.sleep(0.5)
    GPIO.output(LED, False)
    time.sleep(0.5)

---

import RPi.GPIO libs
set pin type. use BCM, not pin number
GPIO 4 pin (Pin 7)
set LED pin as output
toggle light pin signal to low/high to make it blink.
5. Introduction to IoT
Making it IoT:
Remote-Controlled LED

https://github.com/pubnub/workshop-raspberrypi/tree/master/projects-python/remote-led
Making it IoT: Remote-Controlled LED

P → PubNub → S

publish data
subscribe data

Yo Pi, blink LED!
Blink, blink

http://
Making it IoT: Remote-Controlled LED

Subscribing data from a web client

```python
pubnub = Pubnub(publish_key='demo', subscribe_key='demo')
channel = 'disco'

def _callback(m, channel):
    if m['led'] == 1:
        for i in range(6):
            GPIO.output(LED_PIN, True)
            time.sleep(0.5)
            GPIO.output(LED_PIN, False)
            time.sleep(0.5)

pubnub.subscribe(channels=channel, callback=_callback, error=_error)
```

button.addEventListener('click', publish);

When the button is clicked on browser, it publishes data, {'led': 1}
Making it IoT:
Remote-Controlled LED

IoT & PubNub
Case Study: Insteon

http://www.insteon.com
6. Go conquer IoT
Projects

- Pyroelectric IR Motion sensor
- Combinations of sensors and LED
- DHT22 Temperature & Humidity sensor
PIR Motion Sensor

It detects motions by measuring changes in IR radiation when an object moves around it.


http://pubnub.github.io/workshop-raspberrypi/web/motion.html
PIR Motion Sensor

Check VCC & GND on your sensor. They may be opposite from this diagram!!!
PIR Motion Sensor w/ LED

Combination of the PIR motion sensor with a LED as a visual indicator

https://github.com/pubnub/workshop-raspberry-pi/tree/master/projects-python/motion-led

http://pubnub.github.io/workshop-raspberry-pi/web/motion.html
PIR Motion Sensor w/ LED

Note: The circuit change only. The code remains the same.
Data Visualization with Temperature Sensor

It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin.


http://pubnub.github.io/workshop-raspberrypi/web/temperature.html
DHT22 Sensor

GPIO 4

4.7 k ~ 10 k Ohm
DHT22 Sensor

Download & Install Adafruit DHT library:

~$ git clone https://github.com/adafruit/Adafruit_Python_DHT.git

~$ cd Adafruit_Python_DHT

~$ sudo python setup.py install
Realtime Data Graphs & Charts

https://github.com/pubnub/eon-chart
You’ve got some extra time for one more project?

Come to front to pick up:

- HC-SR04 Ultrasonic sensor
- resistors (1k Ohm & 2.2k Ohm)
- extra wires
Ultrasonic RangeFinder

The HC-SR04 ultrasonic sensor uses sonar signals to determine distance to an object.
Ultrasonic RangeFinder


http://pubnub.github.io/workshop-raspberrypi/web/range.html
Ultrasonic RangeFinder

1 K Ohm

2.2 K Ohm
Thank you :-)

Creative Commons Attributions

- LED circuit: Wikimedia
- PIR Sensor: Wikimedia / Oomlout
- Ultrasonic: Wikimedia / Georg Wiora (Dr. Schorsch)
- GPIO Pins: RaspberryPi-Spy.co.uk

Also, great public domain images from Pixabay!
Please give us feedback at:

https://goo.gl/0WeZQb