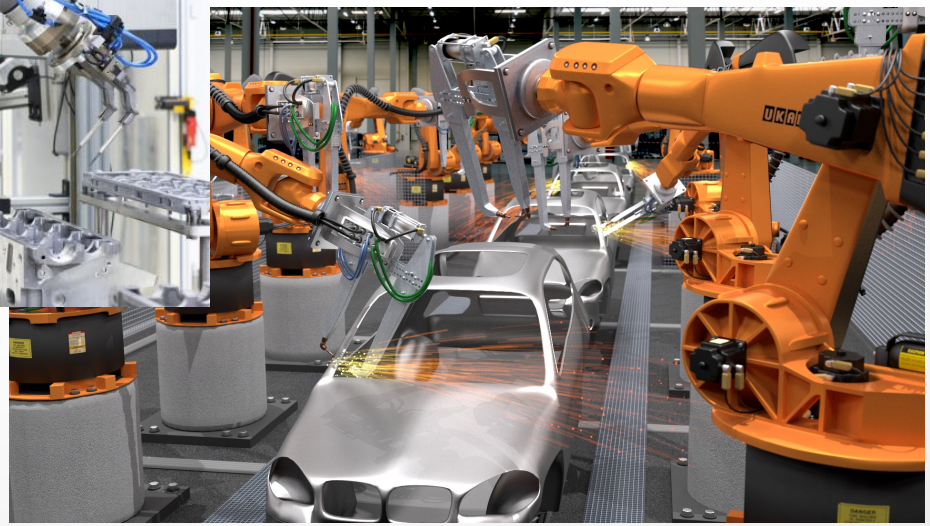

UBIDOTS

PLATAFORMA IOT DE REFERENCIA

presentado por:

Francisco Javier Ferrández Pastor

UBIDOTS

INTRODUCCIÓN

El primer paso para trabajar con esta plataforma es el de crear una cuenta.

Para ello se deben seguir los tutoriales e información del fabricante

Se recomienda este vídeo

https://youtu.be/V20wIZ0ajJE

UBIDOTS

Connect the Raspberry Pi with Ubidots (http://help.ubidots.com/connect-your-devices/connect-the-raspberry-pi-with-ubidots) (*fuente UBIDOTS*)
Learn how to setup the Raspberry Pi using Wi-Fi or Ethernet and how send data to Ubidots.

Raspberry Pi has become a widely used device, not only for prototyping and educational purposes, but also for production projects within companies.
Besides its small size, low cost and fully operational Linux OS, the greatest difference from your Destkop PC lies in its ability to interact with other peripherals through GPIO pins (General Purpose Input/Ourput Pins).
This allows you to code pretty robust hardware applications without having to be an expert in embedded electronics.
At the end of this guide you'll be able to read and send data values from your Pi to Ubidots.

UBIDOTS

EJEMPLO

Requests: HTTP

Requests is a popular Python library that simplifies making HTTP requests from any python script which can be run in your computer's terminal or any embedded Linux device such is the Raspberry Pi.
To install the library run the command below in the Raspberry Pi Terminal:

```
$ pip install requests
```

UBIDOTS

EJEMPLO

Send one value to Ubidots
Let's create a python script using your favorite text-editor. We'll use "**nano**" in this case. This script sends a random value to Ubidots but it can be easily adapted to send values from digital inputs in your Raspberry Pi, or even analog readings when using special shields.
Create Python script

```
$ nano ubi_test.py
```

Copy and paste the below code in your terminal; assigning your Ubidots TOKENwhere indicated in the code. For more information, please see the Ubidots REST API Reference.

PLATAFORMAS IOT

EJEMPLO

```
import time
import requests
import math
import random

TOKEN = "..."  # Put your TOKEN here
DEVICE_LABEL = "machine"  # Put your device label here
VARIABLE_LABEL_1 = "temperature"  # Put your first variable label here
VARIABLE_LABEL_2 = "humidity"  # Put your second variable label here
VARIABLE_LABEL_3 = "position"  # Put your second variable label here
```

PLATAFORMAS IOT

EJEMPLO

```
def build_payload(variable_1, variable_2, variable_3):
    # Creates two random values for sending data
    value_1 = random.randint(-10, 50)
    value_2 = random.randint(0, 85)

    # Creates a random gps coordinates
    lat = random.randrange(34, 36, 1) + \
        random.randrange(1, 1000, 1) / 1000.0
    lng = random.randrange(-83, -87, -1) + \
        random.randrange(1, 1000, 1) / 1000.0
    payload = {variable_1: value_1,
               variable_2: value_2,
               variable_3: {"value": 1, "context": {"lat": lat, "lng": lng}}}

    return payload
```

PLATAFORMAS IOT

EJEMPLO

```
def post_request(payload):
    # Creates the headers for the HTTP requests
    url = "http://things.ubidots.com"
    url = "{}/api/v1.6/devices/{}".format(url, DEVICE_LABEL)
    headers = {"X-Auth-Token": TOKEN, "Content-Type": "application/json"}
    # Makes the HTTP requests
    status = 400
    attempts = 0
    while status >= 400 and attempts <= 5:
        req = requests.post(url=url, headers=headers, json=payload)
        status = req.status_code
        attempts += 1
        time.sleep(1)
    # Processes results
    if status >= 400:
        print("[ERROR] Could not send data after 5 attempts, please check \
            your token credentials and internet connection")
        return False
    print("[INFO] request made properly, your device is updated")
    return True
```

PLATAFORMAS IOT

EJEMPLO

```
def main():
    payload = build_payload(
        VARIABLE_LABEL_1, VARIABLE_LABEL_2, VARIABLE_LABEL_3)

    print("[INFO] Attemping to send data")
    post_request(payload)
    print("[INFO] finished")


if __name__ == '__main__':
    while (True):
        main()
        time.sleep(1)
```

UBIDOTS

EJEMPLO CONEXIÓN CON MQTT-UBIDOTS

Connect to Ubidots MQTT broker with TLS Security

A Python example on how to use our MQTT broker with TLS

In our Ubidots security guide, we explained why encryption is important for IoT and how we support two main methods for data encryption: SSL for HTTP communications, and TLS for MQTT.

MQTT is a lightweight publish/subscribe messaging transport optimized for IoT that supports **TLS encryption. TLS** (Transport Layer Security) provides a secure communication channel between a client and a server. Just like SSL, TLS is a cryptographic protocol that uses a handshake mechanism to create a secure connection between the client and the server. After the handshake is completed, an encrypted communication between client and server is established and no attacker could understand the content of the communication.

In this guide, we'll see how to publish MQTT messages using TLS encryption.

1- Retrieve our SSL certificate

Unlike unencrypted MQTT traffic, a TLS-powered client will need a certificate stored on its side. So let's start by storing that certificate locally by creating a new text file and entering the text below:

FORMAS IOT

EJEMPLO

-----**BEGIN** CERTIFICATE-----
MIIDSjCCAjKgAwIBAgIQRK+wgNajJ7qJMDmGLvhAazANBgkqhkiG9w0BAQUFADA/
MSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT
DkRTVCBSb290IENBIFgzMB4XDTAwMDkzMDIxMTIxOVoXDTIxMDkzMDE0MDExNVow
PzEkMCIGA1UEChMbRGlnaXRhbCBTaWduYXR1cmUgVHJ1c3QgQ28uMRcwFQYDVQQD
Ew5EU1QgUm9vdCBDQSBYMzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AN+v6ZdQCINXtMxiZfaQguzH0yxrMMpb7NnDfcdAwRgUi+DoM3ZJKuM/IUmTrE4O
rz5Iy2Xu/NMhD2XSKtkyj4zl93ewEnu1lcCJo6m67XMuegwGMoOifooUMM0RoOEq
OLI5CjH9UL2AZd+3UWODyOKIYepLYYHsUmu5ouJLGiifSKOeDNoJjj4XLh7dIN9b
xiqKqy69cK3FCxolkHRyxXtqqzTWMln/5WgTe1QLyNau7Fqckh49ZLOMxt+/yUFw
7BZy1SbsOFU5Q9D8/RhcQPGX69Wam40dutolucbY38EVAjqr2m7xPi71XAicPNaD
aeQQmxkqtilX4+U9m5/wAl0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNV
HQ8BAf8EBAMCAQYwHQYDVR0OBBYEFMSnsaR7LHH62+FLkHX/xBVghYkQMA0GCSqG
SIb3DQEBBQUAA4IBAQCjGiybFwBcqR7uKGY3Or+Dxz9LwwmglSBd49lZRNI+DT69
ikugdB/OEIKcdBodfpga3csTS7MgROSR6cz8faXbauX+5v3gTt23ADq1cEmv8uXr
AvHRAosZy5Q6XkjEGB5YGV8eAIrwDPGxrancWYaLbumR9YbK+rlmM6pZW87ipxZz
R8srzJmwN0jP41ZL9c8PDHIyh8bwRLtTcm1D9SZlmlJnt1ir/md2cXjbDaJWFBM5
JDGFoqgCWjBH4d1QB7wCCZAA62RjYJsWvIjJEubSfZGL+T0yjWW06XyxV3bqxbYo
Ob8VZRzI9neWagqNdwvYkQsEjgfbKbYK7p2CNTUQ
-----**END** CERTIFICATE-----
-----**BEGIN** CERTIFICATE-----
MIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw
TzELMAkGA1UEBhMCVVMxKTAnBgNVBAoTIEludGVybmV0IFNIY3VyaXR5IFJlc2Vh
cmNolEdyb3VwMRUwEwYDVQQDEwxJU1JHIFJvb3QgWDEwHhcNMTUwNjA0MTEwNDM4
WhcNMzUwNjA0MTEwNDM4WjBPMQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJu
ZXQgU2VjdXJpdHkgUmVzZWFyY2ggR3JvdXAxFTATBgNVBAMTDElTUkcgUm9vdCBY
MTCCAilwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAK3oJHP0FDfzm54rVygc
h77ct984kIxuPOZXoHj3dcKi/vVqbvYATyjb3miGbESTtrFj/RQSa78f0uoxmyF+
0TM8ukj13Xnfs7j/EvEhmkvBioZxaUpmZmyPfjxwv60pIgbz5MDmgK7iS4+3mX6U
A5/TR5d8mUgjU+g4rk8Kb4Mu0UlXjIB0ttov0DiNewNwIRt18jA8+o+u3dpjq+sW
T8KOEUt+zwvo/7V3LvSye0rgTBIlDHCNAymg4VMk7BPZ7hm/ELNKjD+Jo2FR3qyH
B5T0Y3HsLuJvW5iB4YlcNHlsdu87kGJ55tukmi8mxdAQ4Q7e2RCOFvu396j3x+UC
B5iPNgiV5+I3Ig02dZ77DnKxHZu8A/lJBdiB3QW0KtZB6awBdpUKD9jf1b0SHzUv
KBds0pjBqAlkd25HN7rOrFleaJ1/ctaJxQZBKT5ZPt0m9STJEadao0xAH0ahmbWn
OlFuhjuefXKnEgV4We0+UXgVCwOPjdAvBbI+e0ocS3MFEvzG6uBQE3xDk3SzynTn
jh8BCNAw1FtxNrQHusEwMFxIt4I7mKZ9YIqioymCzLq9gwQbooMDQaHWBfEbwrbw
qHyGO0aoSCqI3Haadr8faqU9GY/rOPNk3sgrDQoo//fb4hVC1CLQJ13hef4Y53CI
rU7m2Ys6xt0nUW7/vGT1M0NPAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAPBgNV
HRMBAf8EBTADAQH/MB0GA1UdDgQWBBR5tFnme7bl5AFzgAiIyBpY9umbbjANBgkq
hkiG9w0BAQsFAAOCAgEAVR9YqbyyqFDQDLHYGmkgJykIrGF1XIpu+ILIaS/V9IZL
ubhzEFnTIZd+50xx+7LSYK05qAvqFyFWhfFQDlnrzuBZ6brJFe+GnY+EgPbk6ZGQ
3BebYhtF8GaV0nxvwuo77x/Py9auJ/GpsMiu/X1+mvoiBOv/2X/qkSsisRcOj/KK
NFtY2PwByVS5uCbMiogziUwthDyC3+6WVwW6LLv3xLfHTjuCvjHIInNzktHCgKQ5
ORAzI4JMPJ+GslWYHb4phowim57iaztXOoJwTdwJx4nLCgdNbOhdjsnvzqvHu7Ur
TkXWStAmzOVyyghqpZXjFaH3pO3JLF+I+/+sKAIuvtd7u+Nxe5AW0wdeRlN8NwdC
jNPElpzVmbUq4JUagEiuTDkHzsxHpFKVK7q4+63SM1N95R1NbdWhscdCb+ZAJzVc
oyi3B43njTOQ5yOf+1CceWxG1bQVs5ZufpsMljq4Ui0/1Ivh+wjChP4kqKOJ2qxq
4RgqsahDYVvTH9w7jXbyLeiNdd8XM2w9U/t7y0Ff/9yi0GE44Za4rF2LN9d11TPA
mRGunUHBcnWEvgJBQl9nJEiU0Zsnvgc/ubhPgXRR4Xq37Z0j4r7g1SgEEzwxA57d
emyPxgcYxn/eR44/KJ4EBs+lVDR3veyJm+kXQ99b21/+jh5Xos1AnX5iltreGCc=
-----END CERTIFICATE-----

UBIDOTS

EJEMPLO

Then save it and give it any name, in this case "industrial.pem":

2- Install MQTT Paho

Paho is a popular MQTT client for Python, which has worked pretty well for us. You can install it using pip:

```
$ pip install paho-mqtt==1.3
```

For troubleshooting, or just to learn more, check out paho-mqtt page.

3- Create a Python script

Open a fresh Python file and copy the code below. Please note that:

The MQTT broker port is 8883, instead of 1883.

TLSv1.1 and TLSv1.2 are supported (in Python: *ssl.PROTOCOL_TLSv1_1* and *ssl.PROTOCOL_TLSv1_2*)

This is a publish example, but it should work for subscribe as well. Just specify the TLS configuration before the "connect" line.

You should replace "industrial.cer" with the path where you stored the above file with the certificate,

You should replace the TOKEN with your own Ubidots account token

UBIDOTS

EJEMPLO

```
import paho.mqtt.client as mqttClient
import time
import json
import ssl
def on_connect(client, userdata, flags, rc):
    if rc == 0:

        print("Connected to broker")

        global Connected                #Use global variable
        Connected = True                #Signal connection

    else:

        print("Connection failed")

def on_publish(client, userdata, result):

print "Published!"
Connected = False
broker_address= "industrial.api.ubidots.com"
port = 8883
user = "YOUR-UBIDOTS-TOKEN"
password = ""
topic = "/v1.6/devices/mqtt"
```

UBIDOTS

EJEMPLO

```
client = mqttClient.Client()
client.username_pw_set(user, password=password)
client.on_connect = on_connect
client.on_publish = on_publish
client.tls_insecure_set(False)
client.tls_set(ca_certs="industrial.pem", certfile=None, keyfile=None, cert_reqs=ssl.CERT_REQUIRED, tls_version=ssl.PROTOC
client.connect(broker_address, port=port)
client.loop_start()

while Connected != True:    #Wait for connection
    print "Connecting..."
    time.sleep(1)

try:
    while True:
        payload = json.dumps({"tls_publish_test":20})
        print payload
        client.publish(topic, payload)
        time.sleep(10)

except KeyboardInterrupt:

    client.disconnect()
    client.loop_stop()
```

UBIDOTS

EJEMPLO

This script will send the value "20" every 10 seconds, to a device called "mqtt" inside your Ubidots account. When running the code:

```
$ python publish_singlevalue_tls.py
```

...you should get something like this:

```
Connecting...
Connected to broker
{"tls_publish_test": 20}
Published!
```