

Sistema para adquisición de datos para el control y supervisión de cultivos domésticos por protocolo IPV6

Nombre del Estudiante: **Wilson López**

1. Introducción

La problemática ambiental que enfrenta el mundo, debido al crecimiento exponencial de las ciudades, donde el área forestal ha sido reducida casi por completo varios países del mundo han creado políticas locales y nacionales con el fin de concientizar a la sociedad acerca de la importancia del agro y la forestación, para enfrentar los efectos del cambio climático. En el caso particular de Colombia el ministerio de agricultura tiene leyes para el desarrollo científico e investigativo en el campo de la agricultura y desarrollo rural [Ministerio de agricultura, 2016] ,Se puede presentar como caso particular el desarrollo de techos verdes en el acuerdo 418 de 2009, la Secretaría de Ambiente ha desarrollado la campaña “Una piel natural para Bogotá” esta dicta capacitaciones y asesorías de forma gratuita para quienes deseen implementar estas tecnologías en el distrito. [Ministerio de medio ambiente, 2016].

Un techo verde es un sistema constructivo que permite mantener de manera sostenible un paisaje vegetal sobre la cubierta de un inmueble mediante la adecuada integración entre el inmueble intervenido, la vegetación escogida, el medio de crecimiento diseñado y los factores climáticos y ambientales. [Dunnett et al., 2004]



Figura 1: Terraza piso 3 Secretaria Distrital de Ambiente

La importancia que tiene hoy en día tomar conciencia sobre los techos verdes y los cultivos domésticos y conectividad a Internet que tiene la mayoría de los dispositivos que nos rodean debido a la conciencia que se esta generando hoy en día de techos verdes y cultivos domestico y la introducción de Internet de las cosas (IoT) en este campo([Del Barrio, 1998]). el proyecto de curso busca tener control sobre estos cultivos por medio de un sistemas de adquisición de datos que tenga conectividad Ipv4 y IPV6 ([Deering, 1998]) que se pueda conectar a un servidor web donde se puedan visualizar las variables de los sensores

2. Descripción del proyecto

El proyecto está enmarcado en la temática de Internet de las Cosas (IoT) y Wireless Sensor Networks (WSNs), para el control y supervisión de los cultivos domésticos. El objetivo principal del proyecto es adquirir datos de diferentes sensores(humedad y temperatura) para controlar las variables medidas. Permitiendo

la supervisión de las condiciones de los cultivos propios y de los demás usuarios del sistema, de tal forma que se puedan hacer comparaciones, de esta forma poder hacer comparaciones de diferentes cultivos y posteriormente poder realizar un intercambio de experiencias y cultivos.

El proyecto se divide en los siguientes componentes:

- **adquisición de datos:** los datos se simularan por medio de un números aleatorios dentro de la Raspberry
- **actuadores:** por medio de una raspberry pi se controlaran un actuador desde la raspberry y desde la pagina que conecta a la raspberry.
- **comunicación IPv6:** En el caso de la comunicación IPv6 se trabajara en la red interna de la universidad dado que IPv6 no esta soportado por nubes privadas como es el caso de amazon. La raspberry pi soportan IPv6, se plantea apuntar a la direccional IPv6 de alguno de los dispositivos para adquirir datos por medio de una pagina HTML donde se ven los valores de los sensores y se pueda controlar el actuador
- **visualización de los datos en la nube:** se plantea subir los datos a el servidor de Ubidots para que los usuarios puedan ver los datos desde lugares externos a la Universidad.
- **seguridad:** Como se trata de un ejercicio educativo y que la aplicación no se centra en la seguridad de los datos el sistema contara con una seguridad de los datos baja.



Figura 2: cultivo domestico

3. Objetivos

3.1. General

- Diseñar y realizar un prototipo de un sistema para adquisición de datos para el control y supervisión de cultivos domésticos por protocolo IPV6

3.2. Especificos

- diseñar un dispositivo para la adquisición y control de variables (temperatura y humedad)
- diseñar he implementar comunicación por IPv6 entre los dispositivos.
- Implementar sistema para visualizar los datos adquiridos en la nube

4. Desarrollo del proyecto

4.1. Preparación de la Raspberry

El primer paso es instalar el sistema operativo NOOBS en la Raspberry. Para la instalación se utiliza una microSD de 8Gb mínimo se descarga la imagen de la pagina de raspberrypi, a través de Internet y se instala dicha imagen en la microSD. (<https://www.raspberrypi.org/downloads/noobs/>). al finalizar la instalación arranca el SO de raspberrypi

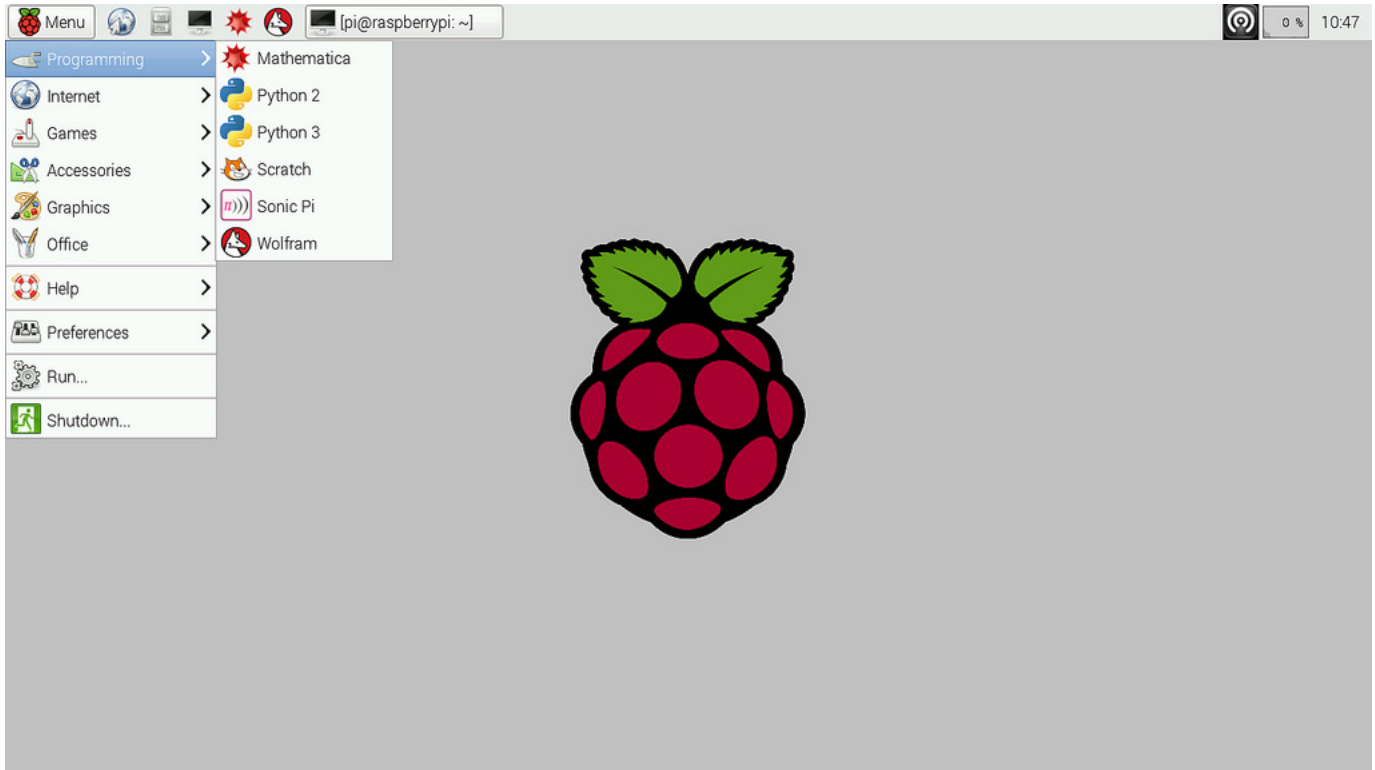


Figura 3: Sistema Operativo de raspberry raspbian

Procedemos a instalar los paquetes y programas necesarios para adquirir los datos y mostrarlos por medio de una interfaz HTML por medio de python, para esto abrimos una consola en las raspberry y ejecutamos los siguientes comandos:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install python
sudo apt-get install python-dev
sudo apt-get install libjpeg-dev
sudo apt-get install libfontconfig-dev
sudo apt-get install python-setuptools
sudo apt-get install python-rpi.gpio
sudo easy_install pip
sudo pip install ubidots
```

Ahora procedemos a descargar e instalar el gestor de paquetes de Python PIP:

```
sudo apt-get install python-pip
pip install tornado
pip install --upgrade pip
```

Con los paquetes instalados podemos iniciar el trabajo, sin embargo se recomienda instalar el paquete *xrdp* con el comando `sudo apt-get install xrdp` este controlará la raspberry por medio de un escritorio remoto:

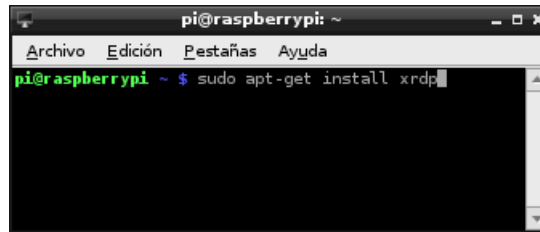


Figura 4: Instalación paquete *xrdp* (tomado de <http://rc-code.blogspot.com.co/2014/02/raspberry-pi-y-windows-conexion-por.html>)

Ahora podemos entrar desde otro equipo que se encuentre en la red a ver y controlar el monitor de la raspberry para acceder por defecto, el usuario es pi y la contraseña es raspberry:

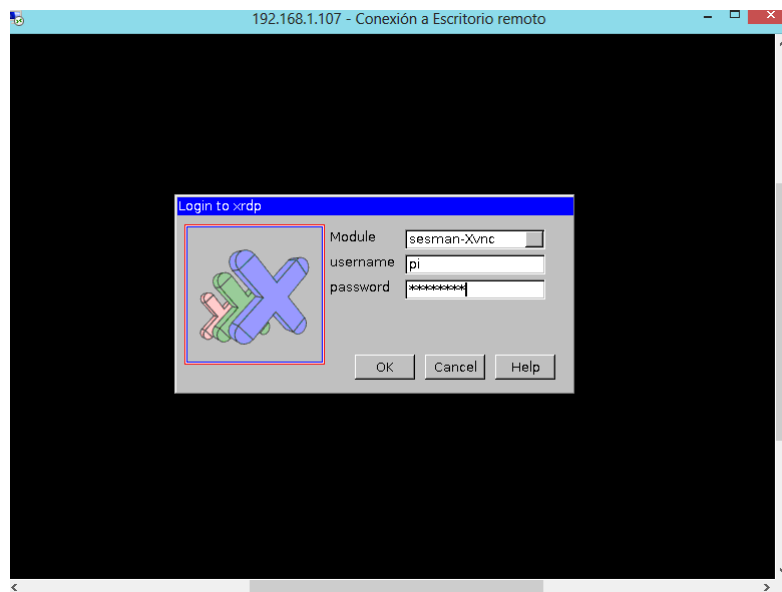


Figura 5: Escritorio Remoto *xrdp* (tomado de <http://4.bp.blogspot.com/-tkTJc9k92no/UwZlYu6IEYI/AAAAAAAAAUU/qtcO16PWVCc/s1600/1.png>)

4.2. Ubidots

Ubidots es un servicio en la nube que permite almacenar y analizar información de sensores en tiempo real. La creación de cuenta es gratuita y permite manejar diferentes variables al mismo tiempo. Comenzamos creando una cuenta en ubidots, cuando esta creada entramos a My Profile:

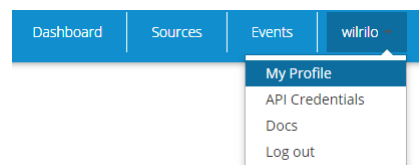


Figura 6: Estructura Ubidots

En la Parte Derecha se encuentra un menu llamado ".API Keys" dando clic en este menú, tenemos el API Key y Tokens para autentificar nuestra cuenta en la raspberry proceso que se explicara mas adelante

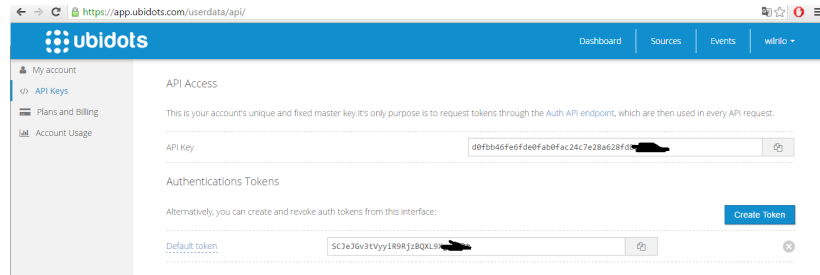


Figura 7: adquirir API y Token de Ubidots

Ahora procedemos a crear las variables y tomar las ID's de las variables necesarias para subir los datos a la nube. en la parte derecha superior vamos al menú *Sources*, donde agregaremos los diferentes tipos de variables en **Add Data Soures**, le damos en el signo mas y le damos un nombre a la fuente que acabamos de crear.

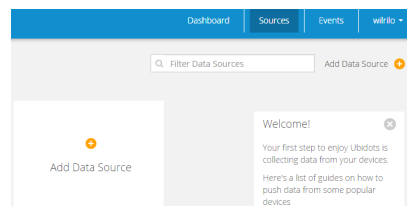


Figura 8: Agregar Variables en Ubidots

Al dar clic en la fuente que acabamos de crear encontramos un menú que nos permite crear variables en *Add Variable*, para este caso particular creamos una variable por defecto.

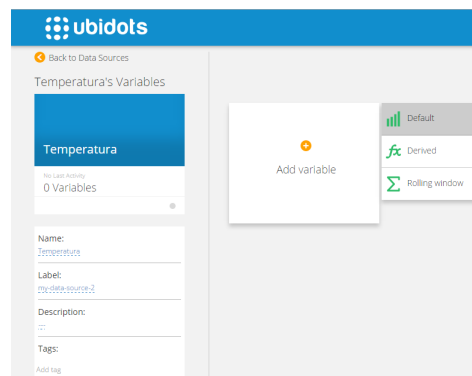


Figura 9: Creación de variables Ubidots

Al crear variables en la parte superior derecha de cada variable se encuentra un signo de *i* latina damos clic sobre este símbolo nos mostrara el ID de dicha variable la cual sera utilizada en el programa de la raspberry para subir los datos a la nube:



Figura 10: ID de las variables en Ubidots

Finalmente para el proyecto que se pretende desarrollar creamos dos fuentes de datos Humedad y temperatura cada uno con las variables

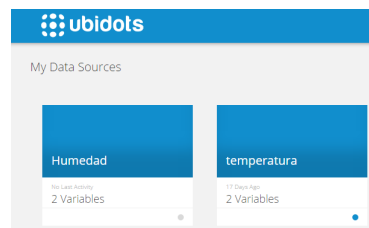


Figura 11: Variables creadas para el proyecto en Ubidots

4.3. Programa en python

Para comenzar a trabajar en python es necesario crear la siguiente estructura de archivos.

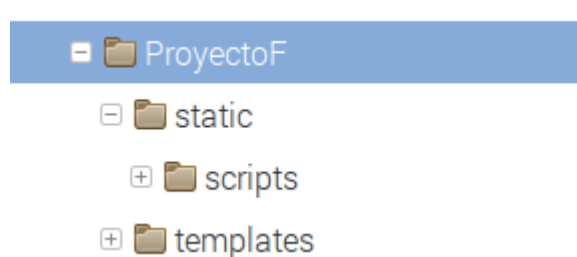


Figura 12: Estructura de archivos para trabajar en python

En la carpeta `proyectoF_Pi` vamos a crear un archivo llamado `programa.py` en el que estará todo el programa de python, en la carpeta `templates` se almacena un archivo llamado `index.html` donde está el código de la página HTML que nos muestra a cualquier usuario que se encuentre en la red los datos de que están adquiriendo y finalmente en la carpeta `scripts` que está dentro de la carpeta `static` se encuentra el archivo `raspberrypi.js` el cual configura la página HTML la dirección IP y puerto donde se va a visualizar:

4.3.1. Código archivo `programa.py`:

En el archivo se importan las librerías de `tornado` con el que se crea el servidor HTML, librerías para tener `timer` en la aplicación, para generar números aleatorios, usar el `GPIO` de la `raspberrypi` y para subir las variables a Ubidots

```
import tornado.web
import tornado.websocket
import tornado.httpserver
import tornado.ioloop
import tornado.options
import json
import time
import threading
import random
from uuid import uuid4
from ubidots import ApiClient
#libreria para usar el puerto GPIO
import RPi.GPIO as GPIO
```

Se crean la API y los ID's de Ubidots para poder subir y visualizar las variables en la nube, los codigos correspondiente al API y los ID's se generan en la pagina de Ubidots proceso que se explica mas adelante.

```
# crear api
api = ApiClient(token= 'SCJeJGv3tVyyiR9RjzBQXL9XgzCCxt')
#creamos las api y los id
humedad = api.get_variable("5763b2ab7625421ca1a7d82a")
temperatura = api.get_variable("5763ac2376254249a1fa9eba")
```

ahora configuramos los pines para entrada y salida del control de la válvula:

```
#pines por el numero canal de las etiquetas.
GPIO.setmode(GPIO.BCM)

#Configurando el pin de salida
GPIO.setup(11, GPIO.OUT)
GPIO.output(11, False)

#Configurando el pin de entrada
swichtPin = 10

GPIO.setup(swichtPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

Creamos la interrupción por hardware de la valvula:

```
def pinkCall(channel):
pulsador = Raspberry()
inputValue = GPIO.input(11)

if(inputValue == True):
GPIO.output(11, False)

pulsador.notifyCallbacks(0, 'La valvula fue apagado por Hardware')

if(inputValue == False):
GPIO.output(11, True)

pulsador.notifyCallbacks(1, 'La valvula fue encendido por Hardware')

print('Interrupcion por hardware')
```

```
GPIO.add_event_detect(switchPin, GPIO.RISING, callback=pinkCall, bouncetime=500)
```

Ahora definimos la clase Raspberry que se encarga de generar datos de variables aleatoriamente, enviarlas a Ubidots, y enviar el estado de la válvula

```
class Raspberry(object):
    callbacks = []

    def obVariables(self):
        hume = random.randint(20,50)
        humedad.save_value({"value":hume})
        time.sleep(1)
        tempe = random.randint(7,16)
        temperatura.save_value({"value":tempe})
        print('humedad= ', hume)
        print('temperatura= ', tempe)

    def register(self, callback):
        self.callbacks.append(callback)

    def unregister(self, callback):
        self.callbacks.remove(callback)

    def ledON(self):
        #Encendemos el Led conectado en el pin 11
        GPIO.output(11, True)
        self.notifyCallbacks(1, "Valvula Encendido")

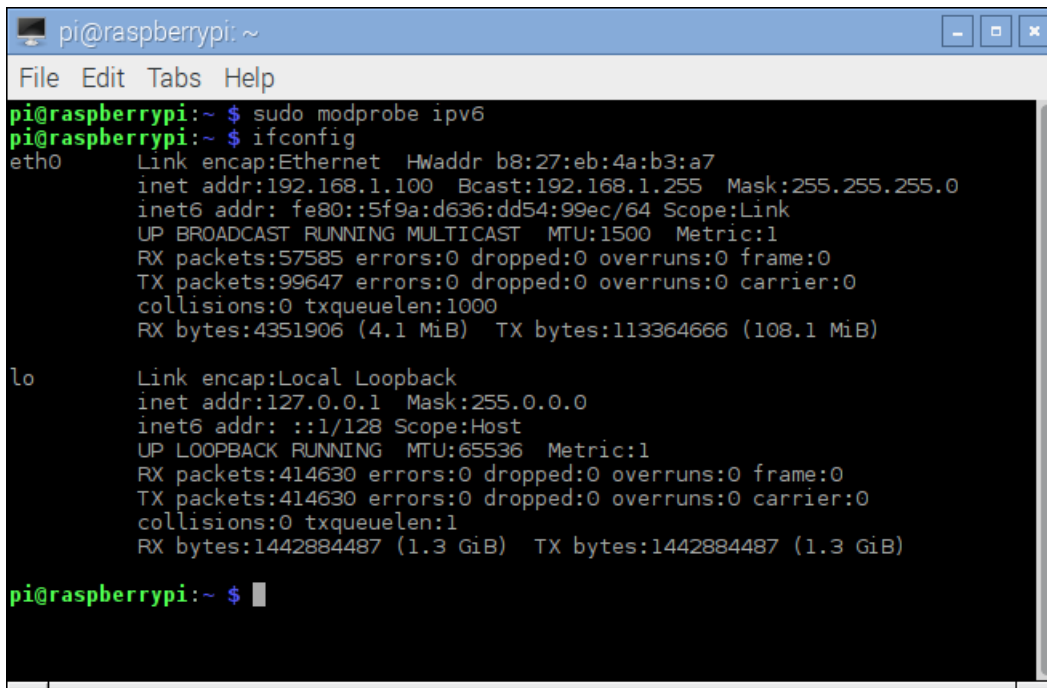
    def ledOFF(self):
        #Apagamos el Led conectado en el pin 11
        GPIO.output(11, False)
        self.notifyCallbacks(0, "Valvula Apagado")

    def notifyCallbacks(self, ledStd, estado):
        for callback in self.callbacks:
            callback(ledStd, estado)
```

Las demás partes del código se encargan de inicializar el servidor HTML para visualizar las variables en la red local por medio de IPv6 el código completo se puede descargar de github.

4.3.2. configuración IPv6

Por defecto en la raspberry el protocolo IPv6 viene desactivado para activarlo en consola ejecutamos el comando `sudo modprobe ipv6`, para verificar nuestra dirección ipv6 ejecutamos el comando `ifconfig` con lo que obtenemos:



```

pi@raspberrypi:~ $ sudo modprobe ipv6
pi@raspberrypi:~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:4a:b3:a7
          inet addr:192.168.1.100 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::5f9a:d636:dd54:99ec/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500 Metric:1
          RX packets:57585 errors:0 dropped:0 overruns:0 frame:0
          TX packets:99647 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4351906 (4.1 MiB)  TX bytes:113364666 (108.1 MiB)

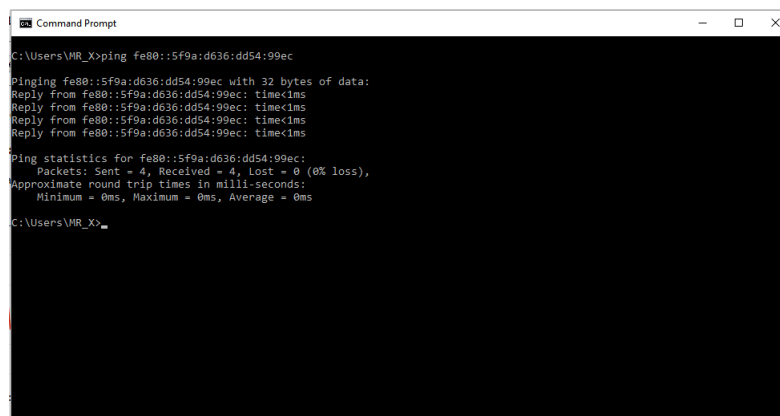
lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536 Metric:1
          RX packets:414630 errors:0 dropped:0 overruns:0 frame:0
          TX packets:414630 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:1442884487 (1.3 GiB)  TX bytes:1442884487 (1.3 GiB)

pi@raspberrypi:~ $

```

Figura 13: Configuración IPv6 raspberry pi

Para este caso en específico la IPv6 es: `fe80::5f9a:d636:dd54:99ec` para comprobar podemos hacer un ping desde una pc que se encuentre en la red:



```

C:\Users\VMR_X>ping fe80::5f9a:d636:dd54:99ec

Pinging fe80::5f9a:d636:dd54:99ec with 32 bytes of data:
Reply from fe80::5f9a:d636:dd54:99ec: time<1ms
Reply from fe80::5f9a:d636:dd54:99ec: time<1ms
Reply from fe80::5f9a:d636:dd54:99ec: time<1ms
Reply from fe80::5f9a:d636:dd54:99ec: time<1ms

Ping statistics for fe80::5f9a:d636:dd54:99ec:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\VMR_X>

```

Figura 14: ping a la raspberry por protocolo IPv6

Como ya se tiene la dirección IPv6 de la raspberry procedemos a configurarla en el servidor HTML que se va a crear, nos dirigimos a la carpeta `scripts` que creamos previamente en la carpeta `static`, en ella abrimos un archivo llamado `raspberry.js` el cual llamará las variables generadas en python a la página HTML, para ver el código completo de este archivo se puede ingresar al proyecto en [hithub](#), en este archivo vamos a introducir la dirección IPv6 de la raspberry modificando la primera línea de código del archivo:

```
var ipDir = '///fe80::5f9a:d636:dd54:99ec:5000';
```

4.3.3. página HTML

En la carpeta `templates` creamos un archivo llamado `index.html` el cual tiene la página html que van a visualizar los usuarios de la red:

```

<!doctype html>
<html lang="es">
<head>
<meta charset="UTF-8">
<title>Sistema para adquisicion de datos para el control y supervision de cultivos domesticos por protocolo
  IPV6</title>
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js" type="text/javascript"></script>
<script src="{{ static_url('scripts/raspberry.js') }}" type="application/javascript"></script>
</head>
<body>
<div>
<h1>Sistema para adquisicion de datos para el control y supervision de cultivos domesticos por protocolo
  IPV6</h1>
<p><span>Estado de la valvula: <span style="color: red;" id="estado">{{ estado }}</span></span></p>
</div>
<hr />
<input type="hidden" id="session" value="{{ session }}" />
<div id="led-on">
<p><input type="submit" value="Encender la valvula" id="add-button" /></p>
</div>
<div id="led-off" style="display: none;">
<p><input type="submit" value="Apagar la valvula" id="remove-button" /></p>
</div>
<hr />
<div>
<h2>Medicion de variables</h2>
<h3><span>Medidas de mi Sistema</span></h3>
<table border="2px">
<tr>
<td><p><span>Temperatura</span></p></td>
<td><p><span>Humedad</span></p></td>
</tr>
<tr>
<td><h2><iframe width="430" height="280" frameborder="0"
  src="https://app.ubidots.com/ubi/getchart/k_u4fOlasl9g0IsPvi6uHzqqgkg"></iframe></h2>
</td>
<td><h2><iframe width="430" height="280" frameborder="0"
  src="https://app.ubidots.com/ubi/getchart/uct647GdFRNHlosEX3_7LQ4Wx88"></iframe></h2>
</td>
</tr>
</table>
<h3><span>Medidas de los demas</span></h3>
<table border="2px">
<tr>
<td><p><span>Temperatura</span></p></td>
<td><p><span>Humedad</span></p></td>
</tr>
<tr>
<td><h2><h2><iframe width="430" height="280" frameborder="0"
  src="https://app.ubidots.com/ubi/getchart/S862eWcAo1rArKC6IXQePIxw9fA"></iframe></h2>
</td>
<td><h2><iframe width="430" height="280" frameborder="0"
  src="https://app.ubidots.com/ubi/getchart/ltNyvhkOJGUaQtj_bMICPUKLI4E"></iframe></h2>
</td>
</tr>
</table>
</div>

```

```
</body>  
</html>
```

Para obtener las frames de las variables de Ubidots nos vamos a la pagina de en **Dashboard** se vizualizan las variables que se han creado en la parte superior derecha de cualquiera de las variables, se hace clic en la flecha para obtener los datos de la variable:

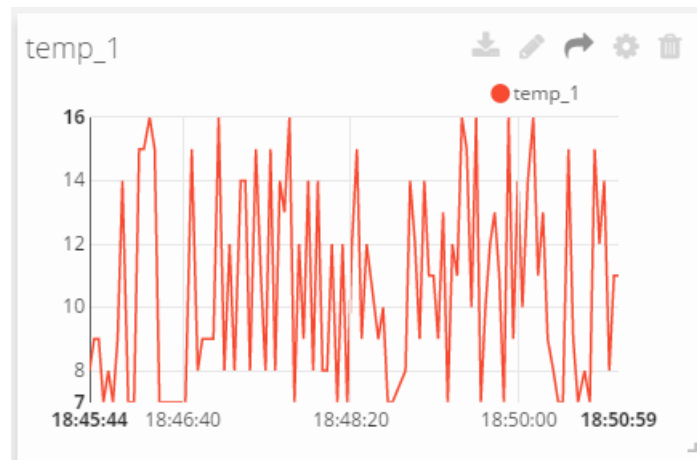



Figura 15: compartir gráficos variables Ubidots para html

Finalmente copiamos el texto que aparece en el campo *Add the following snippet to your HTML*, el cuadro que sale para compartir las gráficas también nos muestra la dirección UML de la gráfica de temperatura

Share Widget ×

Public link:

https://app.ubidots.com/ubi/getchart/page/k_u4f0lasl9g0IsPvi6uHzqqgkg 

Embed the widget in your website

Add the following snippet to your HTML:

```
<iframe width="430" height="280" frameborder="0" src="https://app.ubidots.com/ubi,
```




Figura 16: compartir gráficos variables Ubidots para html

Finalmente la pagina para la visualización de los datos es la siguiente:

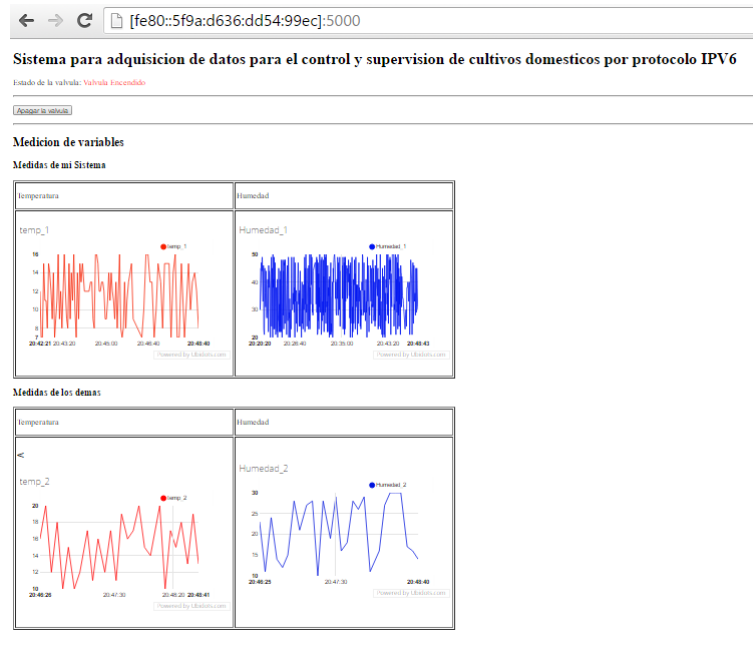


Figura 17: pagina html

4.4. Ejecución del programa

Para la ejecución del proyecto abrimos una terminal en la raspberry, nos ubicamos en la carpeta donde desarrollamos el proyecto en este caso `proyectoFpi` y escribimos el comando `python programa.py`:

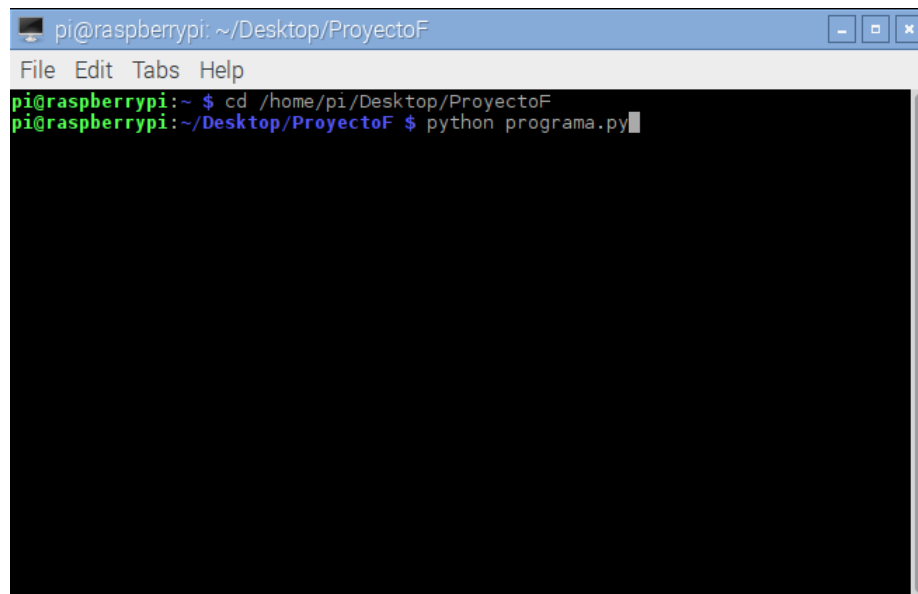
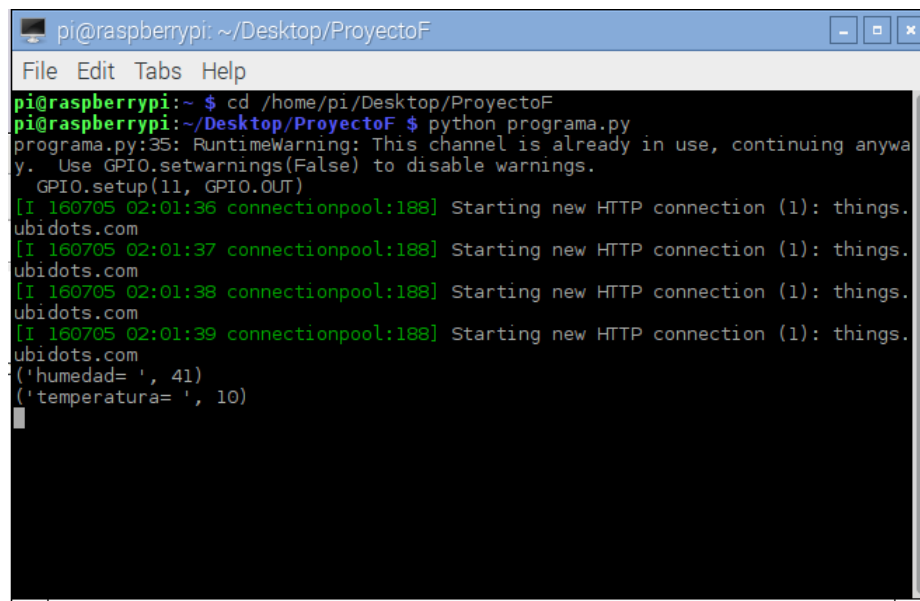


Figura 18: Comandos para ejecutar el programa

Al darle enter la consola comienza a mostrar las diferentes conexiones con la pagina de Ubidots para subir los datos:



```
pi@raspberrypi: ~/Desktop/ProyectoF
File Edit Tabs Help
pi@raspberrypi:~ $ cd /home/pi/Desktop/ProyectoF
pi@raspberrypi:~/Desktop/ProyectoF $ python programa.py
programa.py:35: RuntimeWarning: This channel is already in use, continuing anyway. Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(11, GPIO.OUT)
[I 160705 02:01:36 connectionpool:188] Starting new HTTP connection (1): things.ubidots.com
[I 160705 02:01:37 connectionpool:188] Starting new HTTP connection (1): things.ubidots.com
[I 160705 02:01:38 connectionpool:188] Starting new HTTP connection (1): things.ubidots.com
[I 160705 02:01:39 connectionpool:188] Starting new HTTP connection (1): things.ubidots.com
('humedad= ', 41)
('temperatura= ', 10)
```

Figura 19: Programa ejecutándose en consola

Si entramos desde un dispositivo que se encuentre en la red a la dirección IPv6 de la raspberry al puerto 5000 observamos la pagina html que actualiza constantemente los datos que envía la raspberry:



Figura 20: pagina html

En la pagina se visualizan las 4 variables dos de temperatura y dos de humedad también se ve el estado de la válvula se puede encender y apagar desde la pagina o con un botón físico conectado a la raspberry. Para verificar la conectividad entre dos dispositivos se creo un proyecto que se pudiese ejecutar en un computador con Ubuntu, el proyecto para PC se puede descargar de github

4.5. Analisis de los datos

En esta seccion se realizara un analisis estadistico de los datos por medio de R, como primer paso se instalan las siguientes librerías y paquetes

```
library(methods)
library(jsonlite)
```

Con estos dos paquetes Instalados podemos Descargar de la nube los datos generados por la Raspberry con los siguiente comandos:

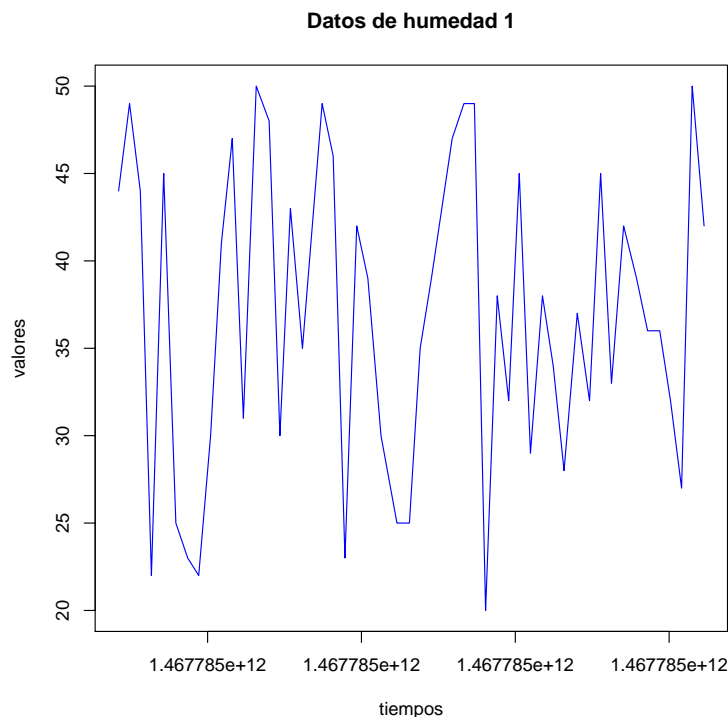
```
ubiURL<-"http://things.ubidots.com/api/v1.6/variables/5763b2ab7625421ca1a7d82a/"
ubiURL<-paste(ubiURL,"values/?token=SCJeJGv3tVyyiR9RjzBQXL9XgzCCxt",sep = "")
```

En este caso hay que remplazar ID_de_la_variable por el ID de cada variable y el tokendeUbidots por le token de la cuenta donde se encuentra la variable, la obtención de estos dos parámetros se explica en la sección llamada Ubidots. Ahora guardamos y separamos las diferentes variables que se descargan con jsonlite.

```
ubidots <- fromJSON(ubiURL)
valores <- ubidots$results$value
tiempos <- ubidots$results$time
marcas <- ubidots$results$created
```

Ahora procedemos a realizar una gráfica de los datos obtenidos, para este graficamos la Humedad 1 :

```
plot(tiempos,valores,'l',col = "blue",main = "Datos de humedad 1")
```



También podemos realizar un estudio estadístico de los valores de la Humedad 1

```
#media
mean(valores)

## [1] 36.84

#mediana
median(valores)

## [1] 37.5

#cuartiles
quantile(valores)

##      0%    25%    50%    75%   100%
## 20.00 30.00 37.50 44.75 50.00

#rango
range(valores)

## [1] 20 50

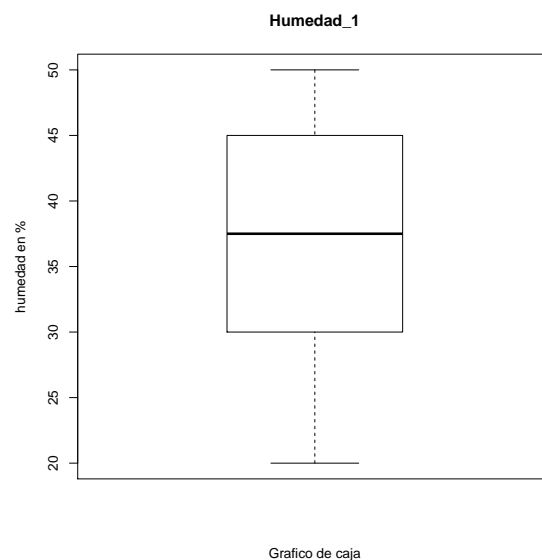
#varianza
var(valores)

## [1] 78.50449

#Desviacion estandar
sd(valores)

## [1] 8.860276
```

```
boxplot(valores,main="Humedad_1",sub="Grafico de caja",ylab="humedad en %")
```



Como parte Final de este trabajo se presentan las gráficas de las 4 variables que se manejaron se descargan y guardan todas las variables:

```

#humedad 1
ubiURL<-"http://things.ubidots.com/api/v1.6/variables/5763b2c87625421f2f017764/"
ubiURL<-paste(ubiURL,"values/?token=SCJeJGv3tVyyiR9RjzBQXL9XgzCCxt",sep = "")
ubidots1 <- fromJSON(ubiURL)
hum1 <- ubidots$results$value
thume1 <- ubidots$results$time

#humedad 2
ubiURL<-"http://things.ubidots.com/api/v1.6/variables/5763b2c87625421f2f017764/"
ubiURL<-paste(ubiURL,"values/?token=SCJeJGv3tVyyiR9RjzBQXL9XgzCCxt",sep = "")
ubidots2 <- fromJSON(ubiURL)
hum2 <- ubidots$results$value
thume2 <- ubidots$results$time

#temperatura 1
ubiURL<-"http://things.ubidots.com/api/v1.6/variables/5763ac2376254249a1fa9eba/"
ubiURL<-paste(ubiURL,"values/?token=SCJeJGv3tVyyiR9RjzBQXL9XgzCCxt",sep = "")
ubidots1 <- fromJSON(ubiURL)
tem1 <- ubidots$results$value
ttem1 <- ubidots$results$time

#temperatura 2
ubiURL<-"http://things.ubidots.com/api/v1.6/variables/5763b35276254225fcaaa266/"
ubiURL<-paste(ubiURL,"values/?token=SCJeJGv3tVyyiR9RjzBQXL9XgzCCxt",sep = "")
ubidots2 <- fromJSON(ubiURL)
tem2 <- ubidots$results$value
ttem2 <- ubidots$results$time

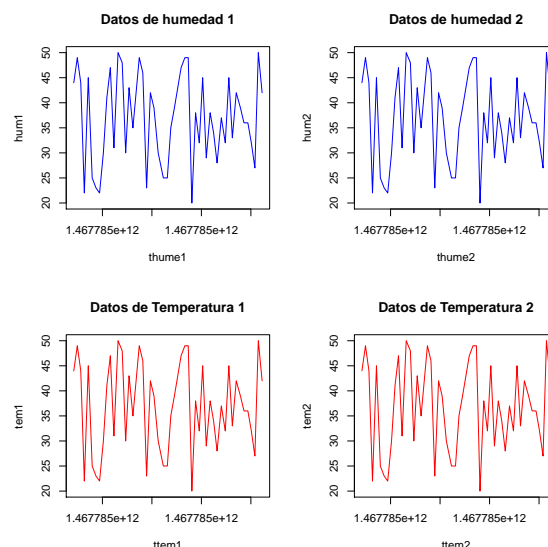
```

Ahora se muestran las cuatro variables en una gráfica:

```

par(mfrow=c(2,2))
plot(thume1,hum1,'l',col = "blue",main = "Datos de humedad 1")
plot(thume2,hum2,'l',col = "blue",main = "Datos de humedad 2")
plot(ttem1,tem1,'l',col = "red",main = "Datos de Temperatura 1")
plot(ttem2,tem2,'l',col = "red",main = "Datos de Temperatura 2")

```



Accesos WEB

1. características sensor de humedad soporte rita IPv6
2. imagen arduino
3. imagen sensor de temperatura
4. imagen raspberry pi
5. ministerioagricultura
6. <http://ambientebogota.gov.co/techos-verdes-y-jardines-verticalessthash.tSdqSq1s.dpuf> [fecha de consulta: 22 de agosto de 2015]
7. <http://ambientebogota.gov.co/web/una-piel-natural-para-bogota//consulta-la-guia-tecnica-de-techos-> [fecha de consulta: 22 de agosto de 2015]
8. <http://www.eltiempo.com/colombia/cali/proyecto-vive-digital-llega-a-escuelas-de-narino/14681459> [fecha de consulta: 22 de agosto de 2015]
9. <http://weblog.aklmedia.nl/tag/raspberry-pi/>

Referencias

- [Deering, 1998] Deering, S. E. (1998). Internet protocol, version 6 (ipv6) specification.
- [Del Barrio, 1998] Del Barrio, E. P. (1998). Analysis of the green roofs cooling potential in buildings. *Energy and buildings*, 27(2):179–193.
- [Dunnett et al., 2004] Dunnett, N., Kingsbury, N., et al. (2004). *Planting green roofs and living walls*, volume 254. Timber Press Portland, OR.
- [Ministerio de agricultura, 2016] Ministerio de agricultura (10 de bril de 2016). <https://www.minagricultura.gov.co/>.
- [Ministerio de medio ambiente, 2016] Ministerio de medio ambiente (8 de bril de 2016). <http://ambientebogota.gov.co/techos-verdes-y-jardines-verticalessthash.tSdqSq1s.dpuf>.