



Test de Manejo

ROS y más ROS

4 de Abril de 2019

Hora de las noticias

La Granja del futuro



¿Por qué el interés en esta noticia?

- Actualidad Robots
- Vehículos autónomos
- Otras Áreas



Iron Ox

- El objetivo a medio plazo es que todo el proceso, desde el momento en que se planta la semilla hasta la cosecha, esté enteramente automatizado.



Imágen: Iron Ox están experimentando con el cultivo de lechugas y otros vegetales de hoja verde.

Iron Ox

- Han combinado robots con brazos articulados y plataformas móviles capaces de levantar cada remesa de cultivo.



En que afecta a Duckietown

- Un 90 % menos de agua consumida
- Un 30% más de producción por hectárea
- Cosechas durante todo el año con una calidad homogénea
- Reducción de la polución



Reflexión y discusión

- ¿Qué pasa con las personas que trabajan en las granjas?
- ¿Cual es la importancia a largo plazo?
- ¿Es realmente viable esta “granja”?





Test de Manejo

ROS y más ROS

4 de Abril de 2019



Fernando Marón
Systems Development Lead



Darío Osses
Duckspeak Translator

Capacitación de hoy

1. Recapitulación capacitación anterior
2. Conexión inalámbrica duckiebot
3. Implementación
4. Test de Manejo



Recapitulación

Redes y conexiones

SSH

FTP

SSH

- Protocolo de seguridad
- Permite conectar 2 computadores por red
- Permite iniciar procesos a través de otro computador

FTP

SSH

- Protocolo de seguridad
- Permite conectar 2 computadores por red
- Permite iniciar procesos a través de otro computador

FTP

- Protocolo de intercambio de archivos
- Permite traspasar archivos de un computador a otro
- Con ayuda de software, permite editar los archivos de un computador usando otro.

Conexión PC a PC

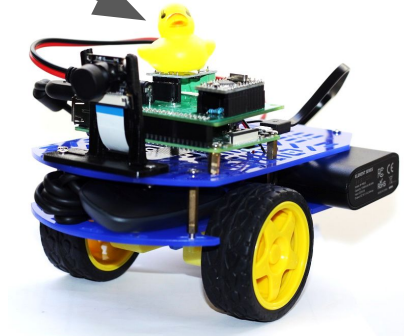
duckietop1



duckietop

IP1: 10.42.0.101

duckiebot



duckiebot

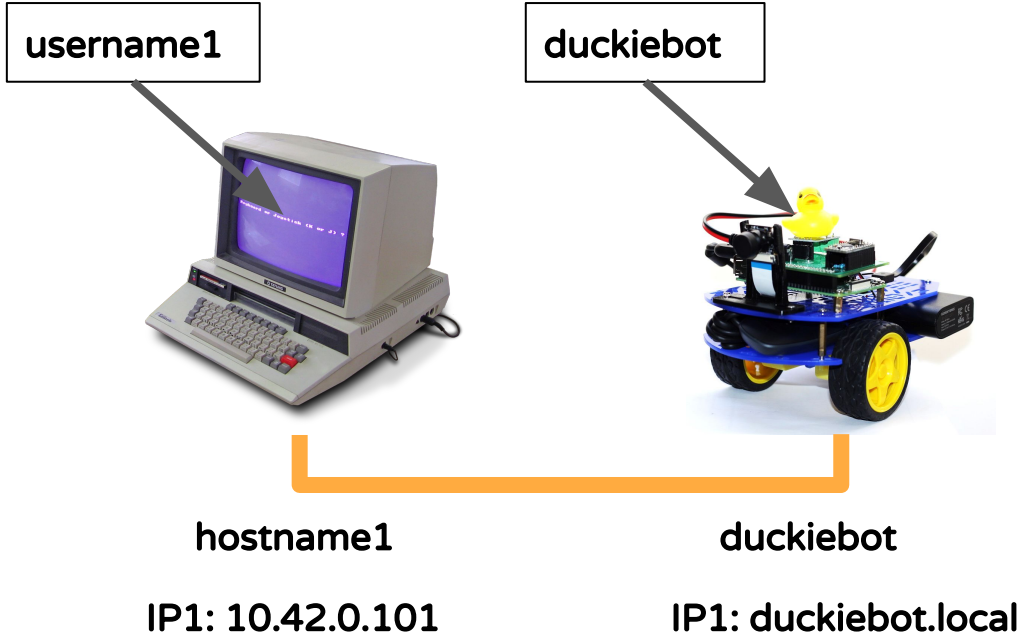
IP1: duckiebot.local
(10.42.0.X)

duckietop debe darle conexión
a internet a **duckiebot** (DHCP)

De este modo conformamos
una red



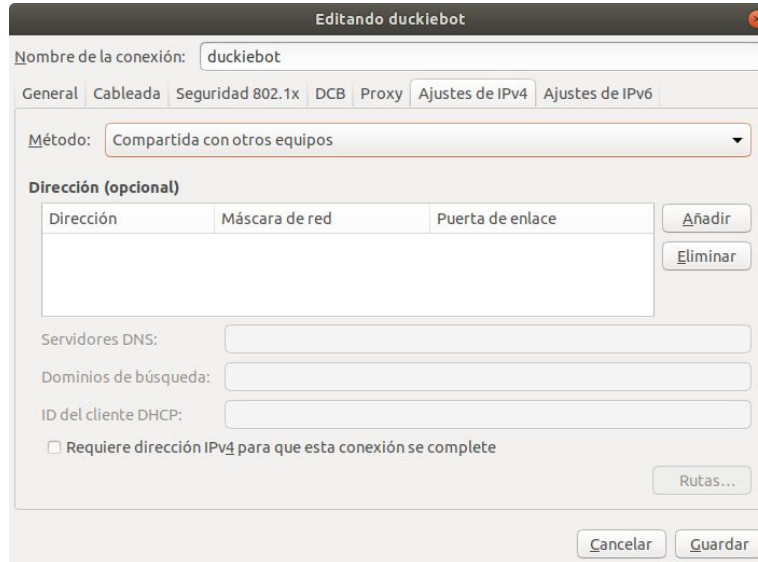
Asegurar que están conectados



```
$ ping duckiebot.local
```

Primera conexión a Duckiebot

- Configurar conexión Ethernet
 - Ubuntu 18: \$ nm-connection-editor



Editando duckiebot

Nombre de la conexión: duckiebot

General | Cableada | Seguridad 802.1x | DCB | Proxy | Ajustes de IPv4 | Ajustes de IPv6

Método: Compartida con otros equipos

Dirección (opcional)

Dirección	Máscara de red	Puerta de enlace
<div></div>		

Añadir
Eliminar

Servidores DNS:

Domínios de búsqueda:

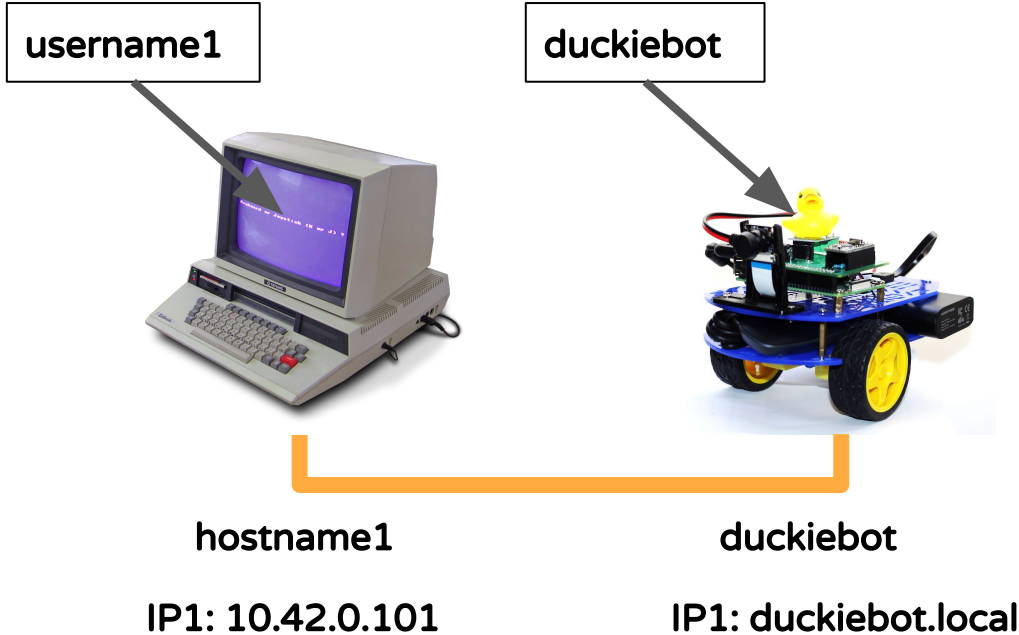
ID del cliente DHCP:

☐ Requiere dirección IPv4 para que esta conexión se complete

Rutas...

Cancelar Guardar

Conexión SSH a duckiebot



```
$ ssh  
duckiebot@duckiebot.local
```

```
pwd: quackquack
```

Byobu

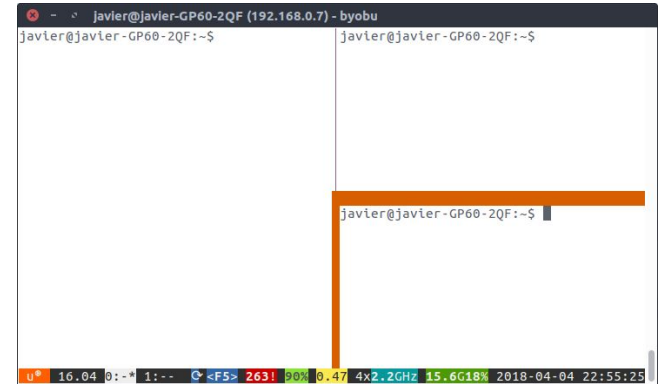
- Terminal²
- Emula múltiples terminales en una sola
- Ideal para trabajar con duckiebots

<F2> : Crear nueva pestaña

<F3> : Mover a pestaña anterior

<F4> : Mover a pestaña siguiente

<Ctrl + D > o “exit” : Cerrar pestaña

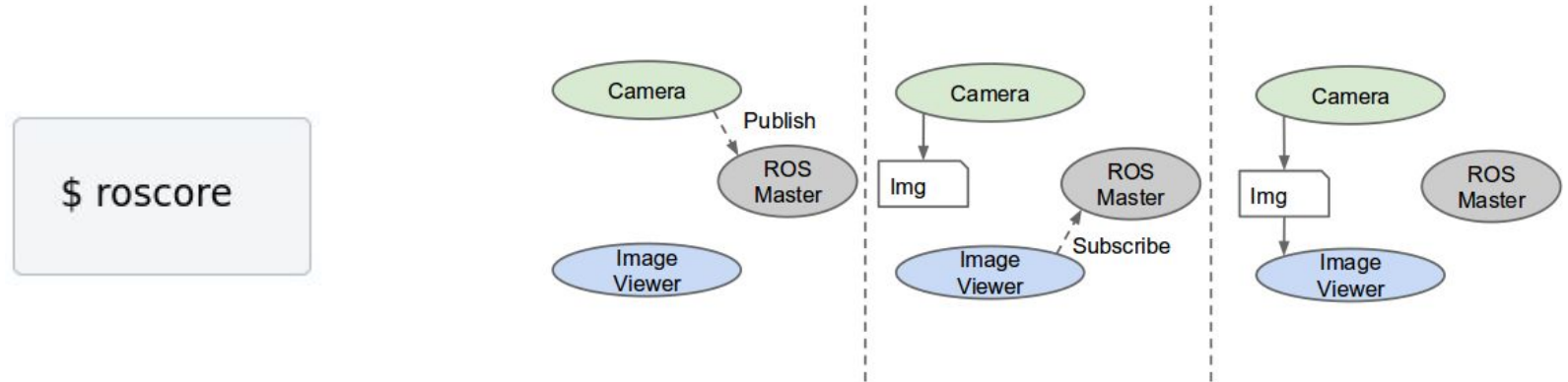


Recapitulación

ROS

Arquitectura : ROS Master

Proporciona el registro de nombres y la búsqueda de nodos. Sin el Master, los nodos no serían capaces de encontrar e intercambiar mensajes, o invocar servicios.



roslaunch

ros_cap

duckie_core.launch

veh:=duckiebot

comando

package

nombre del archivo
launch

argumentos

```
roslaunch ros_cap duckie_core.launch  
veh:=duckiebot
```

wheels_driver_node

joy_node

camera_node



Checklist

1. roscore
2. Abrir nueva terminal con byobu (F2)
3. roslaunch ros_cap duckie_core.launch veh:=duckiebot
4. Abrir ventana nueva (F2)
5. Verificar que este todo corriendo con: rostopic list
6. Verificar que los motores funcionan:
7. rostopic pub /duckiebot/wheels_driver_node/car_cmd duckietown_msgs/Twist2DStamped “header”: *[tab]*

Uso de hotspot

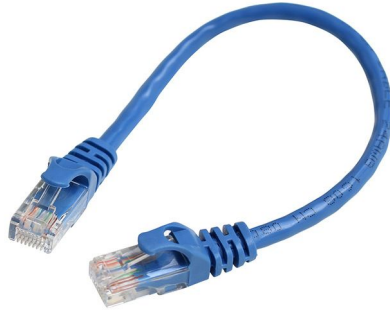
Conexión ethernet con duckiebot

username1

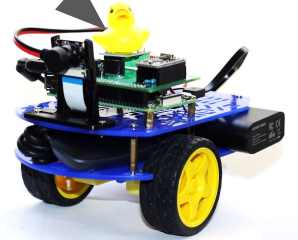


hostname1

IP1: 10.42.0.1



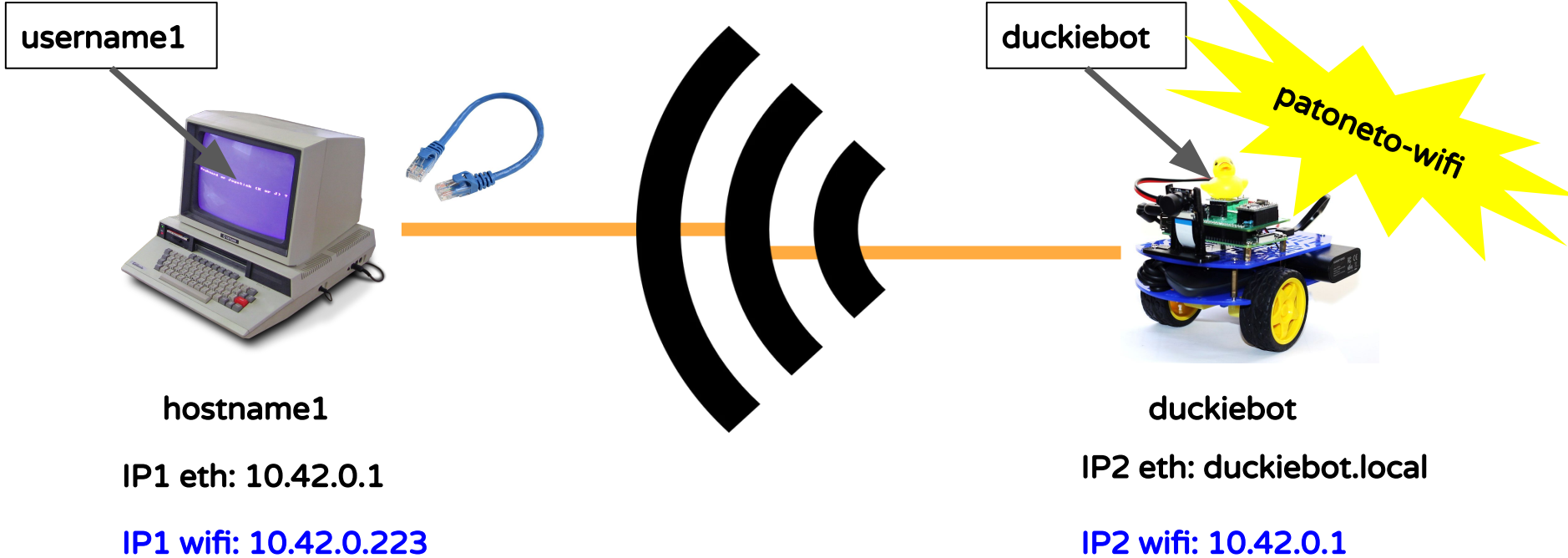
duckiebot



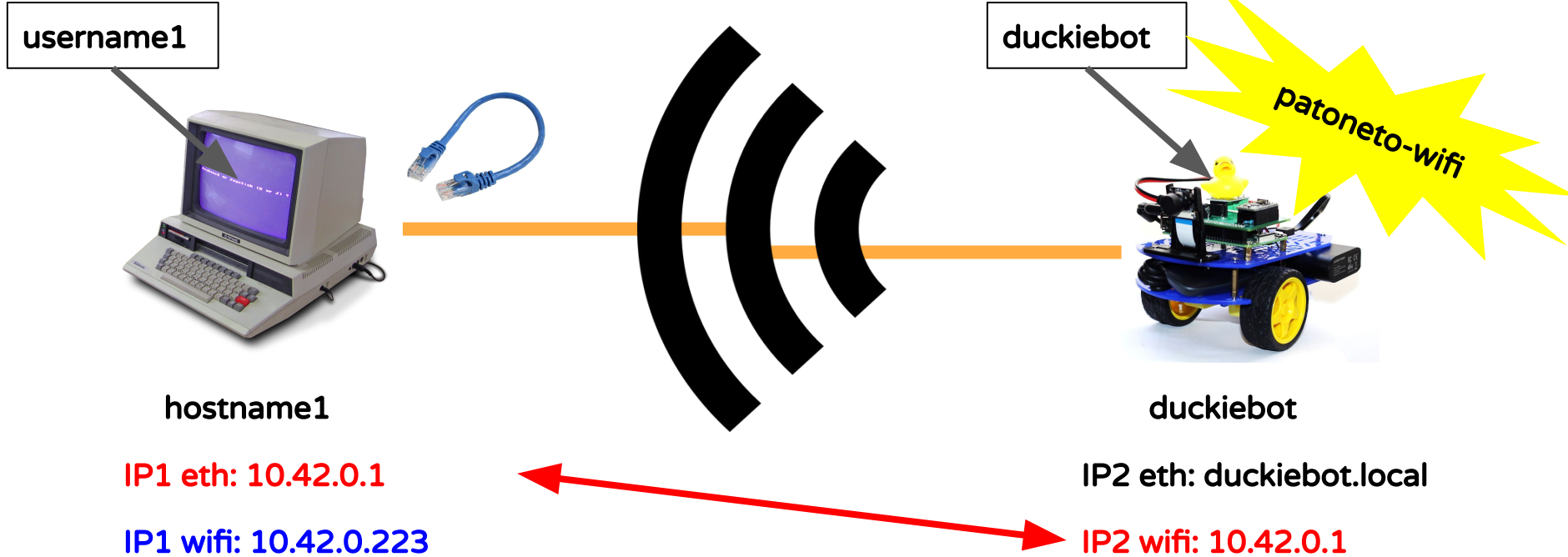
duckiebot

IP2: duckiebot.local

Conexión Wifi con duckiebot



Conexión Wifi con duckiebot



Conexión Wifi con duckiebot

username1



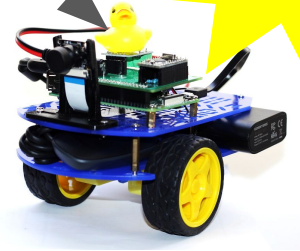
hostname1

~~IP1 eth0: 10.42.0.1~~

IP1 wifi: 10.42.0.223



duckiebot



duckiebot

~~IP2 eth0: duckiebot.local~~

IP2 wifi: 10.42.0.1



Duckietown
Engineering
Chile

Cómo habilitar el hotspot

1. Conectados por ssh, ir a la carpeta duckietown

a. `$ cd ~/duckietown`

2. Cambiar el nombre del wifi de su duckiebot editando el archivo `vehicle_name.sh` utilizando:

`$ nano vehicle_name.sh` o `$ pluma vehicle_name.sh`

`$ source vehicle_name.sh`

1. Ejecutar el script para habilitar el hotspot

b. `$./hotspot.sh on`



Cómo conectarse al Hotspot

1. La terminal se quedará “pegada”, por el conflicto con la IP **10.42.0.1**
2. Desconectar el cable Ethernet
3. Conectarse al wifi ***patoneto-wifi***
 - a. La clave es ***quackquack***
 - b. (mantenganla en secreto)
4. Conectarse por ssh nuevamente
 - a. `$ ssh duckiebot@duckiebot.local`
5. ...
6. Profit!

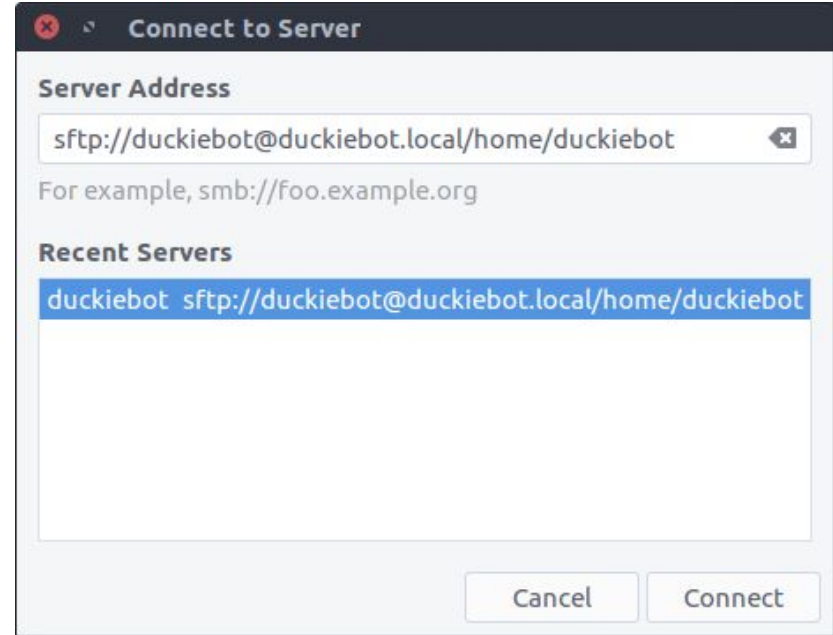


Cómo conectarse al Hotspot

1. La terminal se quedará “pegada”, por el conflicto con la IP **10.42.0.1**
2. Desconectar el cable Ethernet
3. Conectarse al wifi ***patoneto-wifi***
 - a. La clave es ***quackquack***
 - b. (mantenganla en secreto)
4. Conectarse por ssh nuevamente
 - a. `$ ssh duckiebot@duckiebot.local`
5. ...

Conexión FTP

- Abrir navegador de archivos(Nautilus)
- ir a **Files->Connect to Server...**
- **/duckietown/catkin_ws/src/ros_catkin_ws/src**



sftp://duckiebot@duckiebot.local/home/duckiebot



Actividad

Implementación funcionalidades duckiebot

Test de manejo

Exámen al final de la capacitación para obtener su licencia de conducir

Su duckiebot debe:

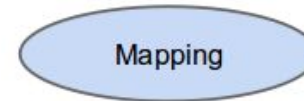
1. Avanzar
2. Retroceder
3. Girar
4. Tener un freno de emergencia



Arquitectura: Nodos

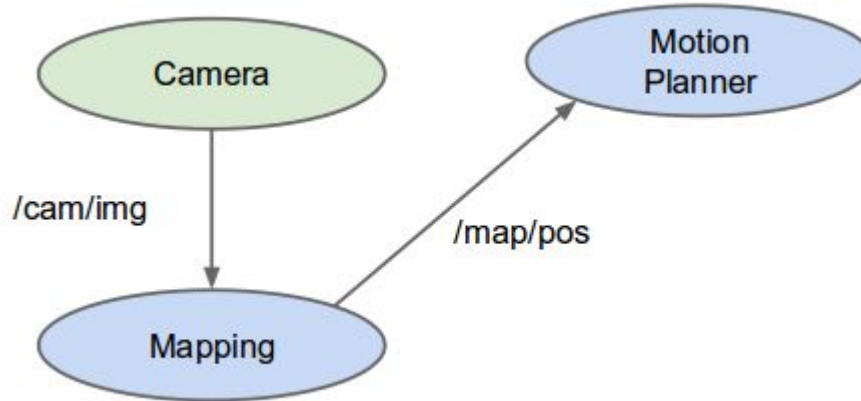
Arquitectura: Nodos

- Equivalente a un módulo e independiente en ejecución
- En general, se programan en C++ o Python
- Puede:
 - interactuar con el **hardware** (driver)
 - mandar y recibir **mensajes**



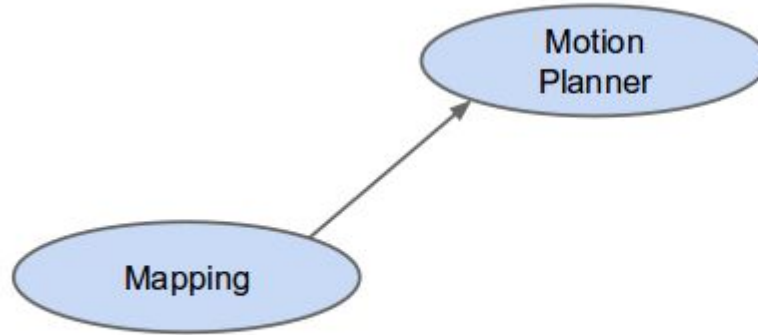
Arquitectura : Tópicos

- Flujos de datos con publicación/suscripción semántica.
- Ellos son los únicos que pueden ser identificados por su nombre.



Arquitectura : Mensajes

- Un mensaje es simplemente una estructura de datos, que comprende los tipos de campos.
- Lenguaje de representación de datos agnóstico. C++ puede hablar con Python.

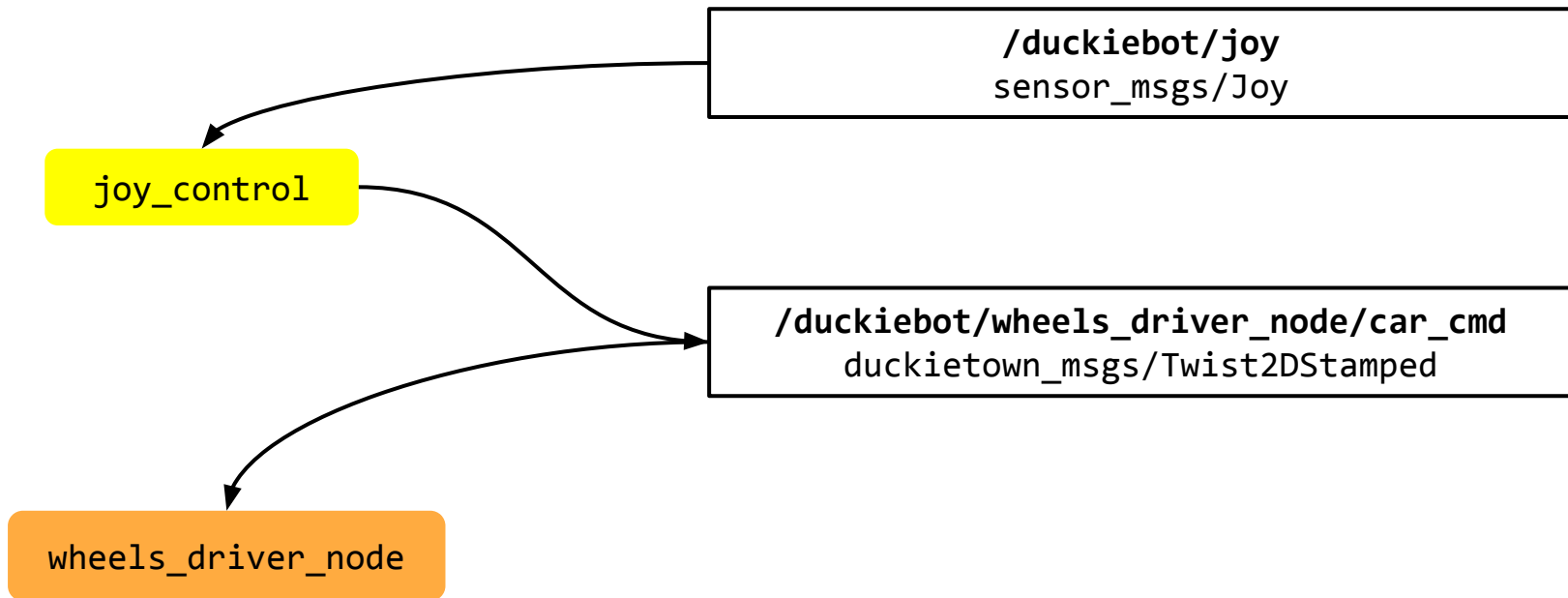


File: pos.msg

```
string robotName  
uint32 posX  
uint32 posY  
uint32 goalX  
uint32 goalY
```


Comunicación Inter-Nodos

- Los nodos pueden publicar y suscribirse a tópicos
- Los tópicos se identifican por su nombre absoluto. Ej: “/chatter”
- El master se encarga de que los mensajes enviados a un tópico lleguen a sus suscriptores



Ejemplo : Publicación

```
#!/usr/bin/env python

import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```



Ejemplo : Subscripción

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```



Template

File: /home/dario/duckietown/catkin...s/src/ros_cap/src/template.jpg

Page 1 of 1

```
#!/usr/bin/env python

import rospy #importar ros para python
from std_msgs.msg import String, Int32 # importar mensajes de ROS tipo String y tipo
Int32
from geometry_msgs.msg import Twist # importar mensajes de ROS tipo geometry / Twist

class Template(object):
    def __init__(self, args):
        super(Template, self).__init__()
        self.args = args

    #def publicar(self):

    #def callback(self,msg):

def main():
    rospy.init_node('test') #creacion y registro del nodo!

    obj = Template('args') # Crea un objeto del tipo Template, cuya definicion se
    encuentra arriba

    #objeto.publicar() #llama al metodo publicar del objeto obj de tipo Template

    #rospy.spin() #funcion de ROS que evita que el programa termine - se debe
    usar en Subscribers

if __name__ == '__main__':
    main()
```



std_msgs/Joy

std_msgs/Header header

float32[] axes

int32[] buttons

duckietown_msgs/Twist2DStamped

Header header

float32 v

float32 omega

Importante

Importar:

```
from sensor_msgs.msg import Joy
```

```
from duckietown_msgs.msg import Twist2DStamped
```

Probar archivo

```
/duckietown/catkin_ws/src/ros_cap/src
```

```
chmod +x template(copy).py
```

```
roslaunch ros_cap template(copy).py
```




```
from sensor_msgs.msg import Joy

from duckietown_msgs.msg import Twist2DStamped

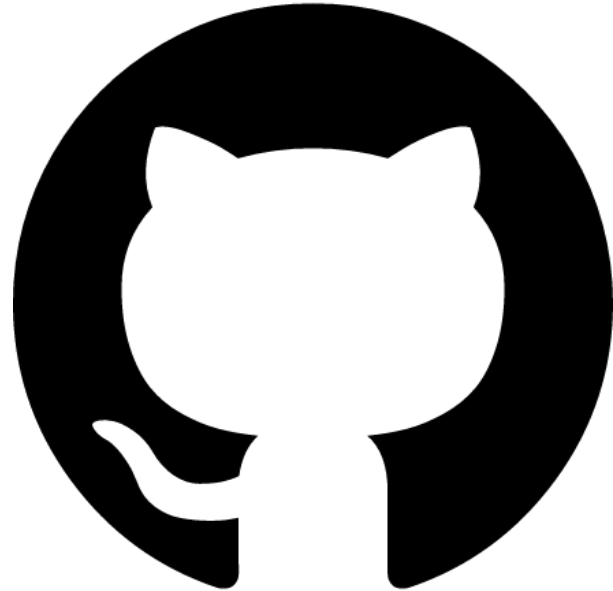
self.publisher = rospy.Publisher("topico", tipo de mensaje, queue_size = "x")

self.subscriber = rospy.Subscriber("topico", tipo de mensaje, self.callback)

self.publisher.publish(self.twist)
```

Tarea

Crear cuenta de Github





Test de Manejo

ROS y más ROS