

Documentación R2-D2

DIPLOMADO: INTELIGENCIA ARTIFICIAL CON ENFOQUE EN
ROBOTS DE SERVICIO

OSCAR YAÑEZ GOMEZ

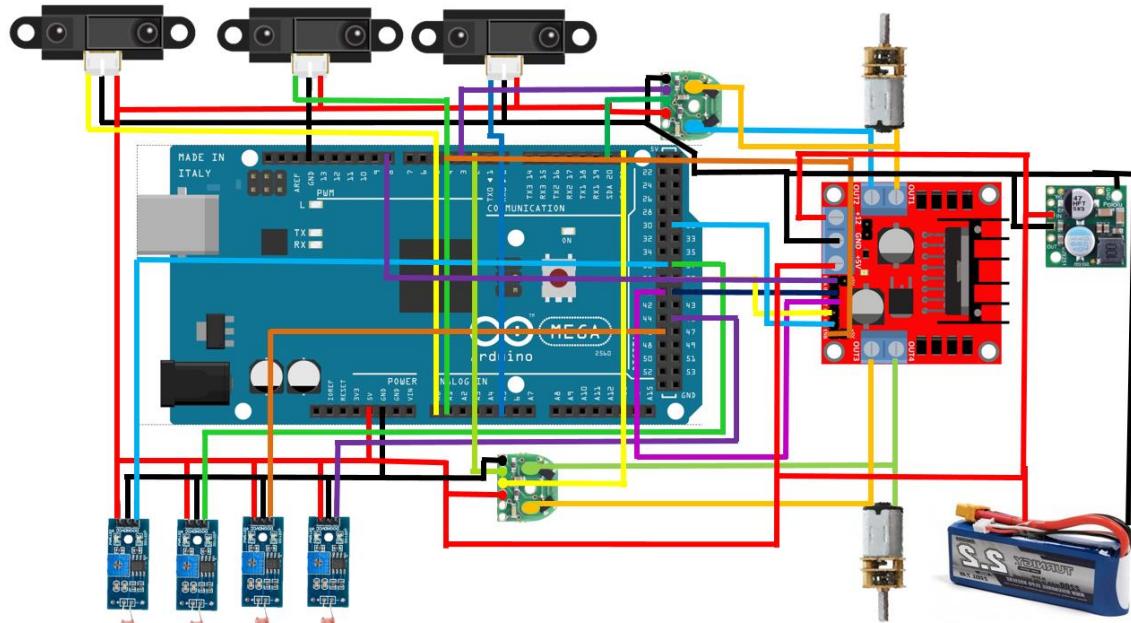
EMMANUEL SANCHEZ ESTRADA



INDICE

- 1.- CONEXIONES ELECTRÓNICAS
- 2.- CONEXIÓN AL R2D2
- 3.- CONEXIÓN CON ROS
- 4.- CARTA ASM DE LA MAQUINA DE ESTADOS PARA SEGUIR 2 COLORES Y EVADIR OBSTÁCULOS
- 5.- NODOS QUE HAY QUE CORRER PARA CORRER EL NODO QUE SIGUE COLORES Y EVADE OBSTÁCULOS
- 6.- CODIGO DE ARDUINO
- 7.- PROGRAMA QUE ABRE LA IMAGEN EN LA COMPUTADORA Y HACER EL FILTRADO DEL COLOR
- 8.- PROGRAMA QUE PUBLICA LA IMAGEN DE LA CÁMARA
- 9.- PROGRAMA PARA SEGUIR 2 COLORES Y EVADIR OBSTÁCULOS

1.- CONEXIONES ELECTRÓNICAS



Nombre del componente	Conexión a los pines del Arduino Mega
Fotorresistencia 1	36
Fotorresistencia 2	37
Fotorresistencia 3	45
Fotorresistencia 4	46
Sharp 1	A0
Sharp 2	A1
Sharp 3	A5
Dirección A1	40

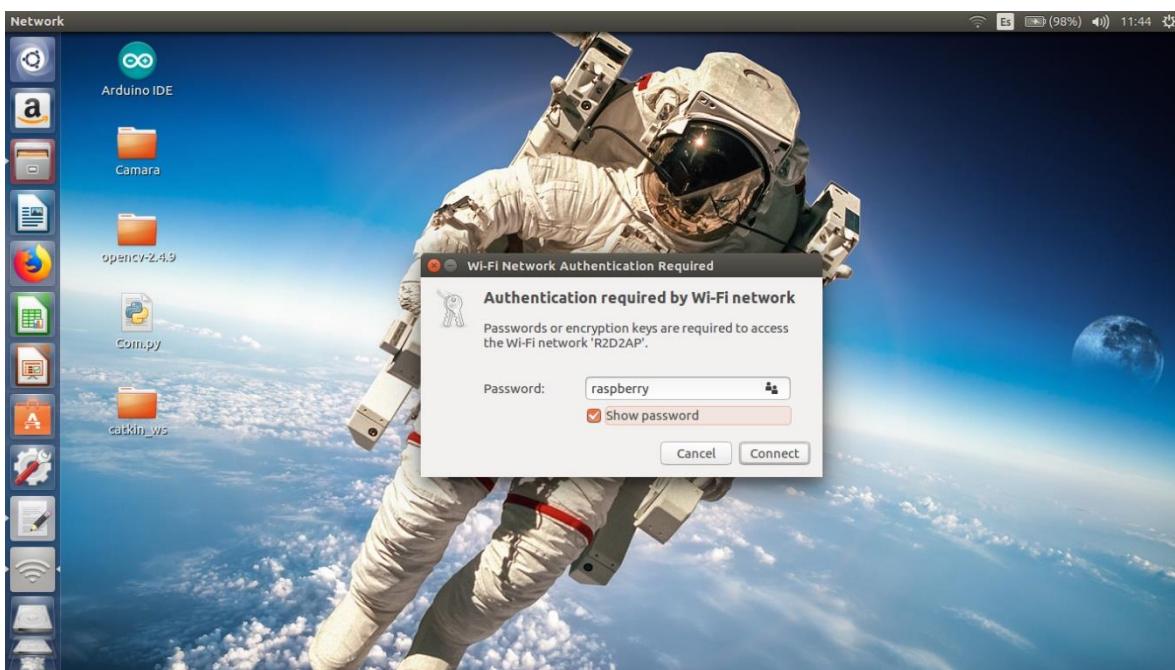
Dirección A2	41
Enable A	4
Dirección B1	31
Dirección B2	38
Enable B	9
Encoder Out A1	2
Encoder Out B1	20
Encoder Out A2	3
Encoder Out B2	21

2.- CONEXIÓN AL R2D2

1.- Para conectarnos al robot por wifi debes buscar en las redes una con el nombre R2D2AP tal como se muestra en la imagen



2.- Nos pedirá una contraseña la cual es raspberry



3.- Una vez establecida la conexión nos aparecerá un cuadro que dice conexión establecida tal y como se muestra en la imagen.



4.- Para poder hacer el intercambio de información debemos modificar el etc/host de nuestro equipo y del R2-D2 para esto debemos conocer la ip de nuestra computadora la cual podemos verificar en una terminal con el comando ifconfig como se muestra en la imagen a continuación

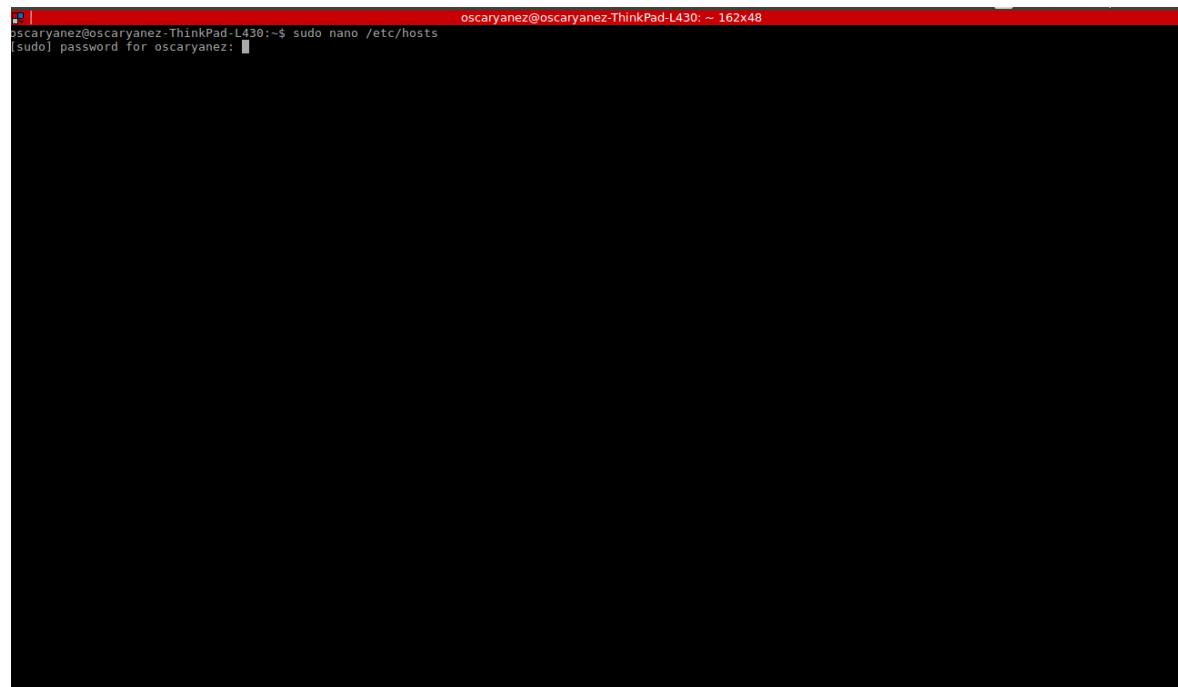
```
oscaryanez@oscaryanez-ThinkPad-L430:~$ ifconfig
enp1s0  Link encap:Ethernet HWaddr 3c:97:0e:0e:b1:17
        UP BROADCAST MULTICAST MTU:1500 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

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:61180 errors:0 dropped:0 overruns:0 frame:0
        TX packets:61180 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:3849811 (3.8 MB) TX bytes:3849811 (3.8 MB)

wlp6s0  Link encap:Ethernet HWaddr 7c:e9:d3:f6:f4:37
        inet addr:172.24.1.72 Bcast:172.24.1.255 Mask:255.255.255.0
        inet6 addr: fe80::3df4:ebcc:8e84:3566/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:19755634 errors:0 dropped:0 overruns:0 frame:0
        TX packets:7917901 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:29739652818 (29.7 GB) TX bytes:684485990 (684.4 MB)

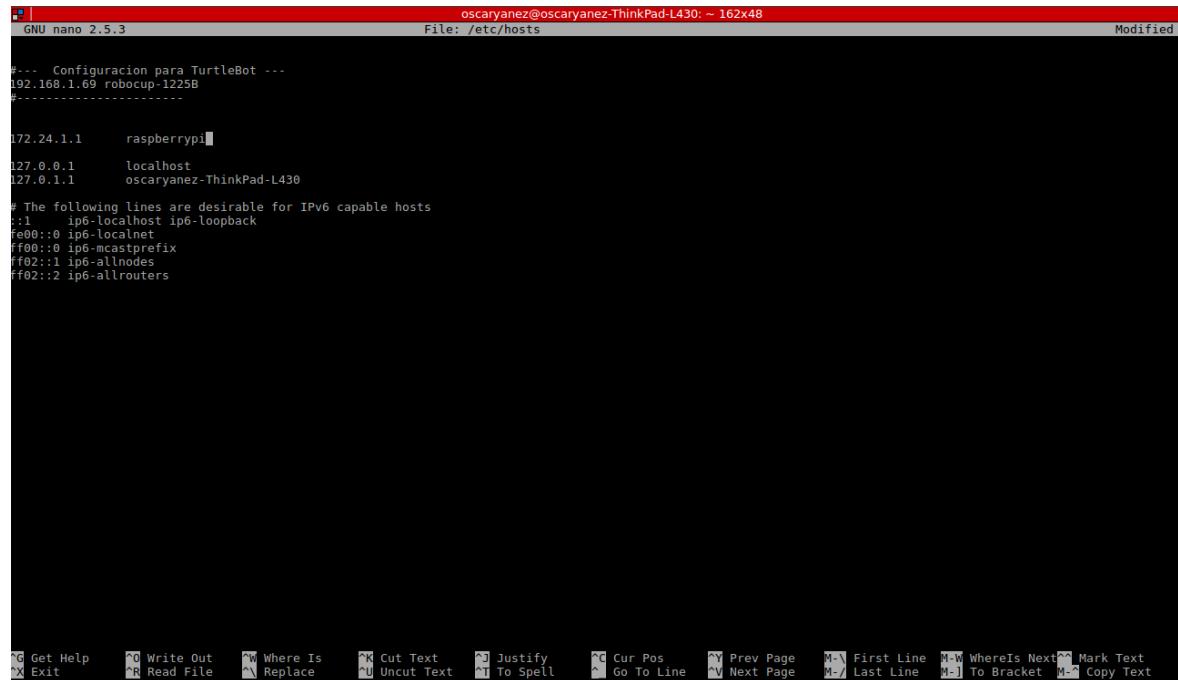
oscaryanez@oscaryanez-ThinkPad-L430:~$
```

5.- Para agregar la ip del R2D2 a nuestro equipo para que este pueda reconocerlo, en una terminal debemos ejecutar el comando: sudo nano /etc/hosts e introducir nuestra contraseña.



```
oscaryanez@oscaryanez-ThinkPad-L430:~ 162x48
oscaryanez@oscaryanez-ThinkPad-L430:~$ sudo nano /etc/hosts
[sudo] password for oscaryanez: [REDACTED]
```

6.- Nos abrirá la siguiente ventana en la cual debemos agregar la ip del arturito con su nombre respectivo la ip es 172.24.1.1 y en nombre es raspberrypi



```
GNU nano 2.5.3
oscaryanez@oscaryanez-ThinkPad-L430:~ 162x48
File: /etc/hosts
Modified

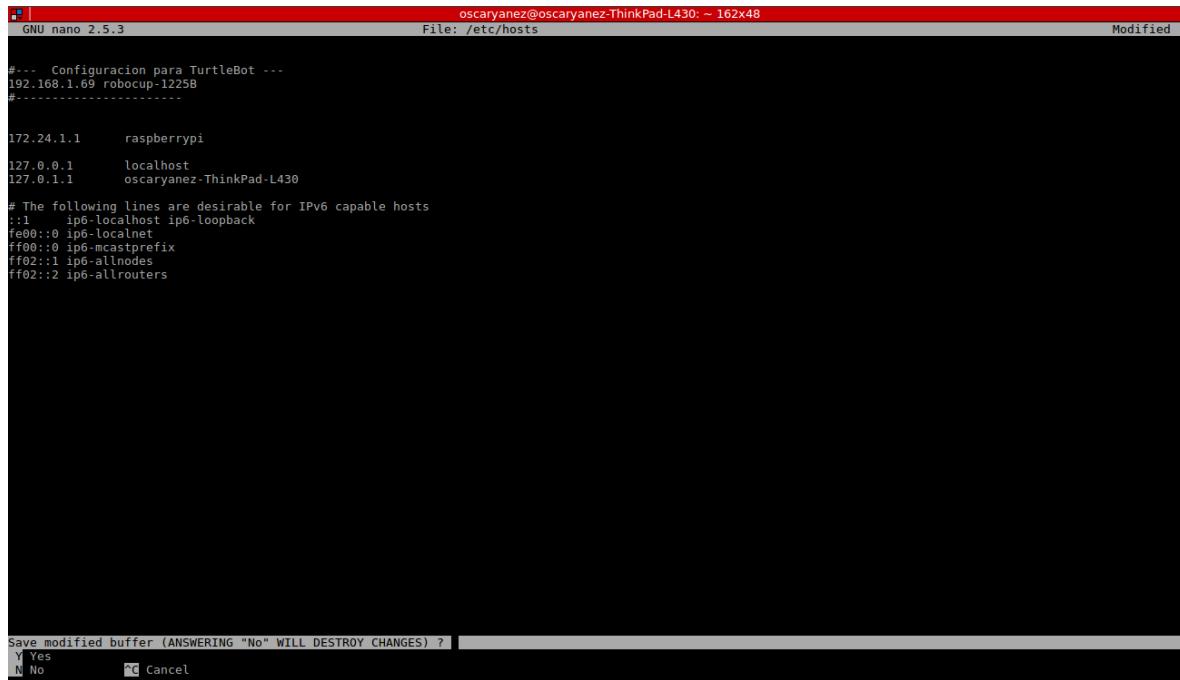
#--- Configuracion para TurtleBot ---
192.168.1.69 robocup-1225B
#-----

172.24.1.1      raspberrypi
127.0.0.1      localhost
127.0.1.1      oscaryanez-ThinkPad-L430

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

[TAB] Get Help [TAB] Write Out [TAB] Where Is [TAB] Cut Text [TAB] Justify [TAB] Cur Pos [TAB] Prev Page [TAB] First Line [TAB] WhereIs Next [TAB] Mark Text
[X] Exit [TAB] Read File [TAB] Replace [TAB] Uncut Text [TAB] To Spell [TAB] Go To Line [TAB] Next Page [TAB] Last Line [TAB] To Bracket [TAB] Copy Text
```

7.- Salimos con Ctrl+X y le decimos que guarde los cambios.



```
GNU nano 2.5.3          oscaryanez@oscaryanez-ThinkPad-L430: ~ 162x48          Modified
File: /etc/hosts

#--- Configuracion para TurtleBot ---
192.168.1.69 robocup-1225B
#-----

172.24.1.1      raspberrypi
127.0.0.1      localhost
127.0.1.1      oscaryanez-ThinkPad-L430

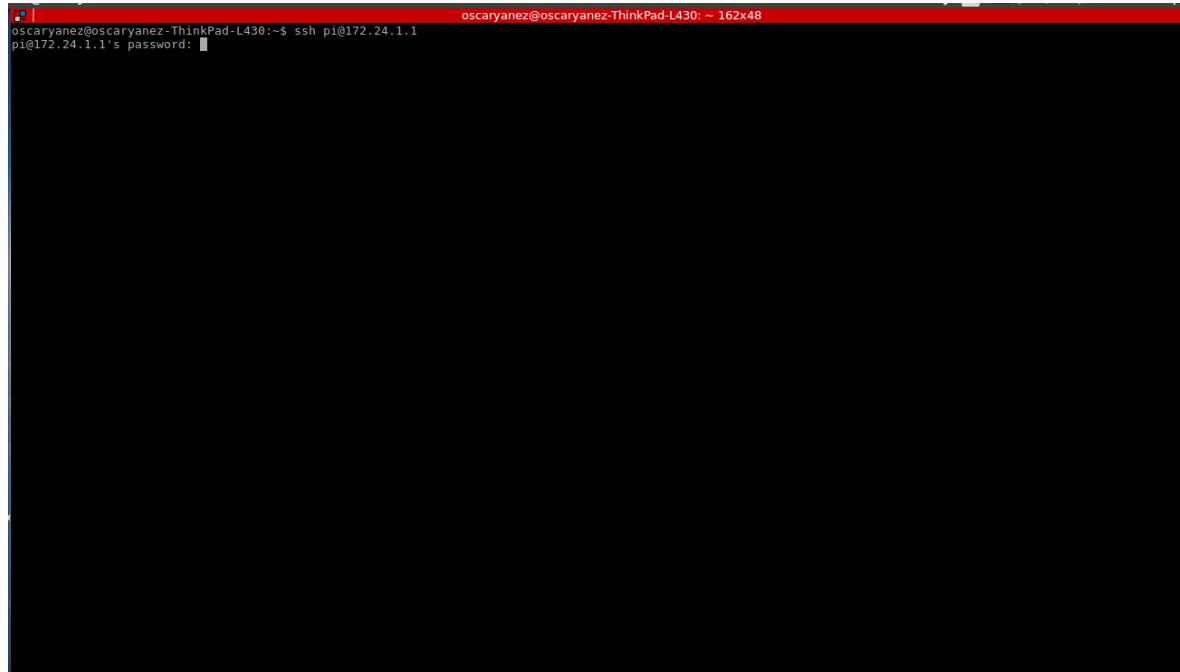
# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
```

Y Yes
N No
^C Cancel

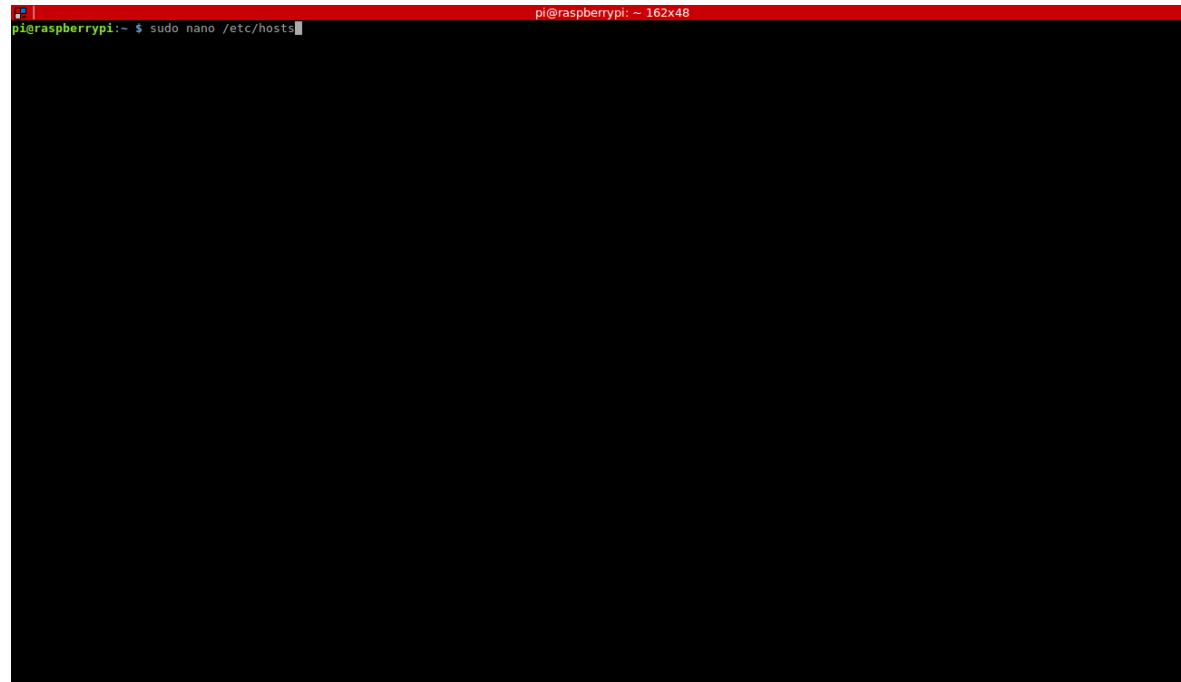
8.- De igual forma en la raspberry debemos agregar la ip de nuestro equipo obtenida en el paso 4 y el nombre de nuestro equipo, para acceder al R2-D2 abrimos una nueva terminal e introducimos el comando:

ssh pi@172.24.1.1 nos pedirá un password el cual es: raspberry

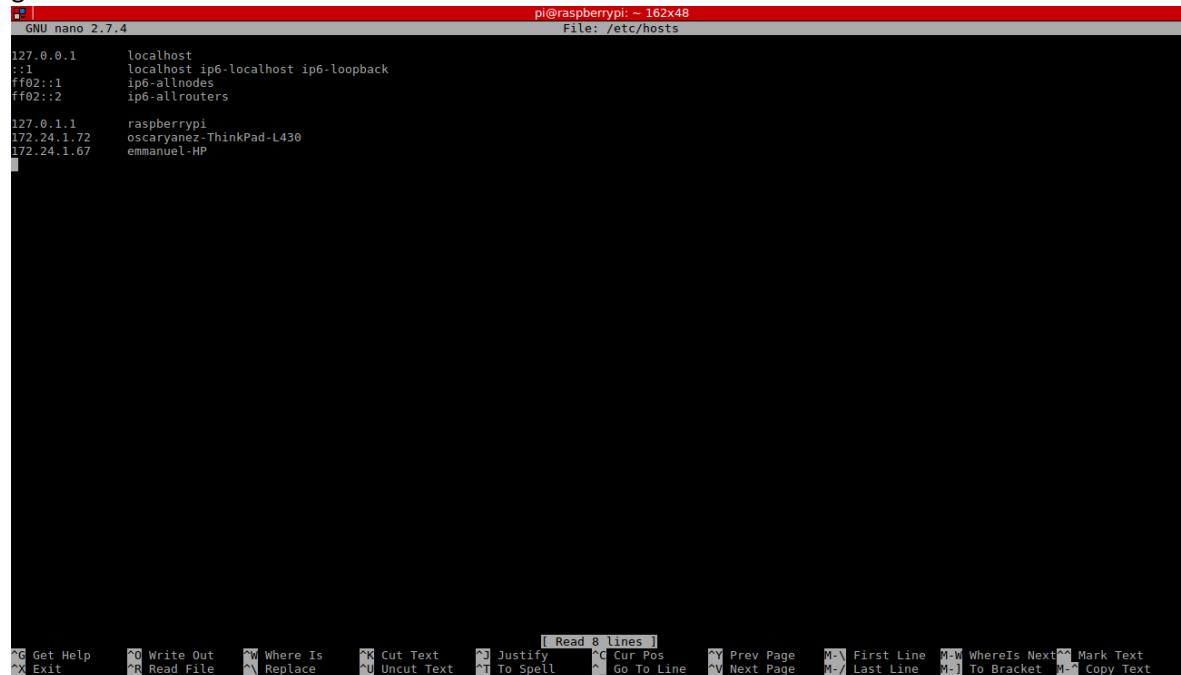


```
oscaryanez@oscaryanez-ThinkPad-L430:~$ ssh pi@172.24.1.1
pi@172.24.1.1's password: [REDACTED]
```

9.- Una vez en que nos conectemos a la raspberry vía ssh ejecutamos el comando: sudo nano /etc/hosts e introducimos la contraseña: raspberry.



10.- Agregamos la ip y el nombre de nuestro equipo salimos con Ctrl+X y le decimos que guarde los cambios.



```
pi@raspberrypi: ~ 162x48
File: /etc/hosts
GNU nano 2.7.4
127.0.0.1      localhost
::1            localhost ip6-localhost ip6-loopback
ff02::1        ip6-allnodes
ff02::2        ip6-allrouters
127.0.1.1      raspberrypi
172.24.1.72    oscaryanez-ThinkPad-L430
172.24.1.67    emmanuel-HP
```

[Read 8 lines]

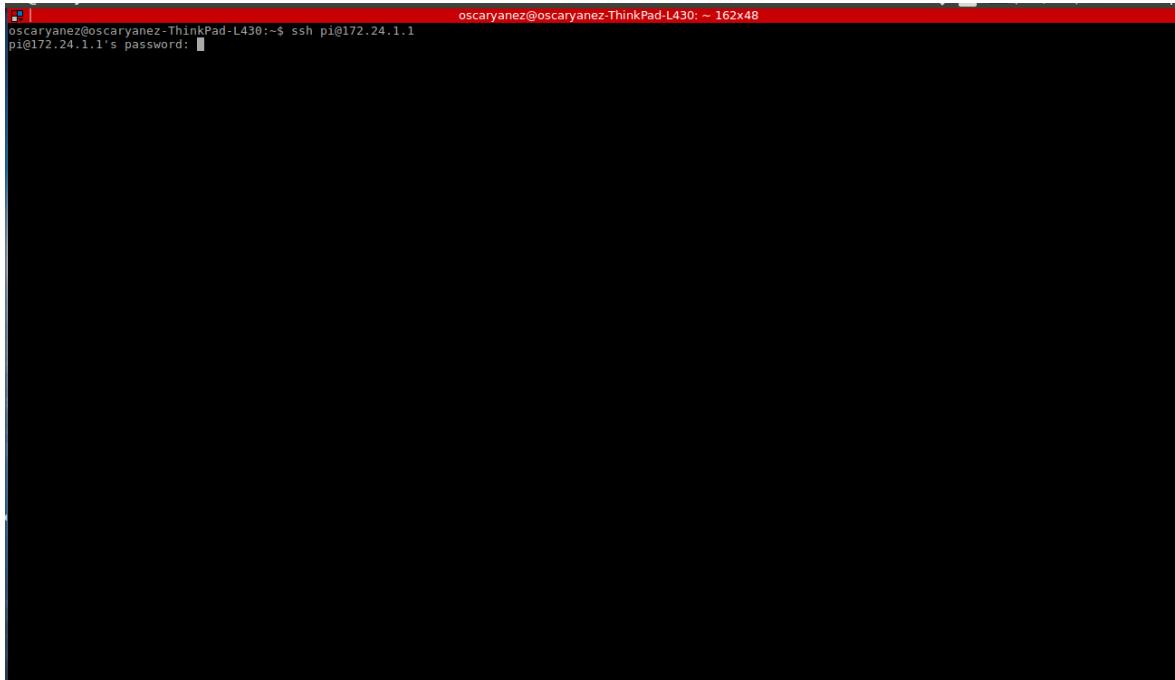
Get Help Write Out Where Is Cut Text Justify Cur Pos Prev Page Next Page First Line Last Line WhereIs Next To Bracket Mark Text

Exit Read File Replace Uncut Text To Spell Go To Line M-] Next Page M-] Last Line M-] To Bracket M-] Copy Text

3.- CONEXIÓN CON ROS

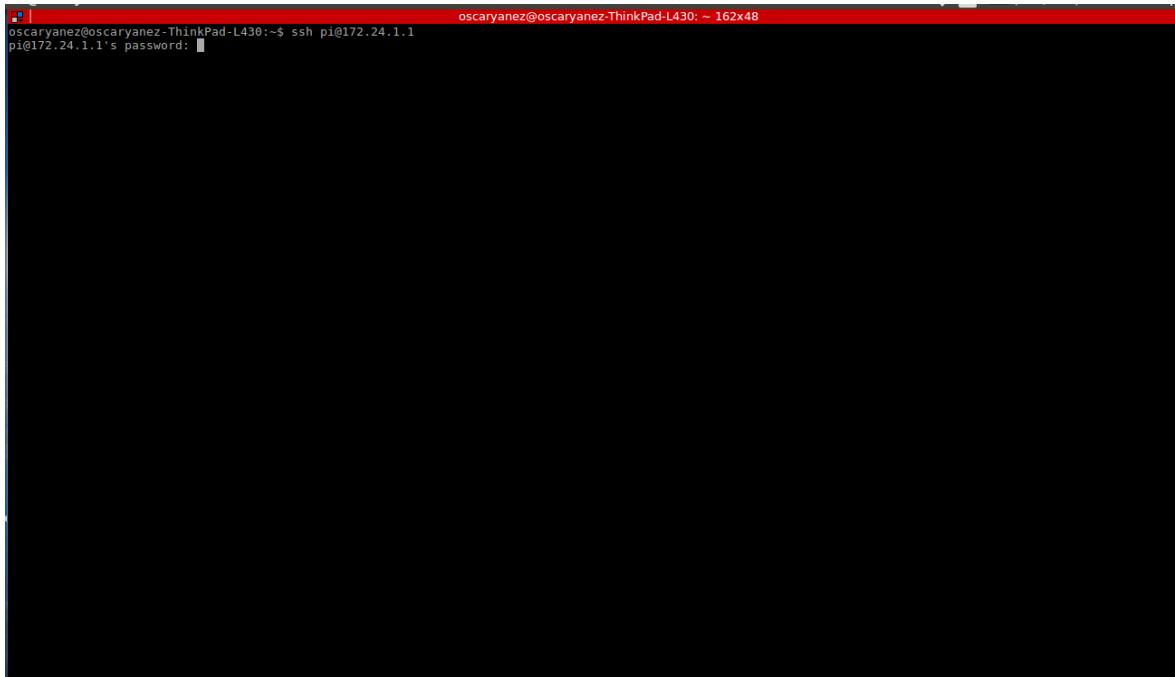
1.- Para acceder al R2-D2 abrimos una nueva terminal e introducimos el comando:

```
ssh pi@172.24.1.1
```



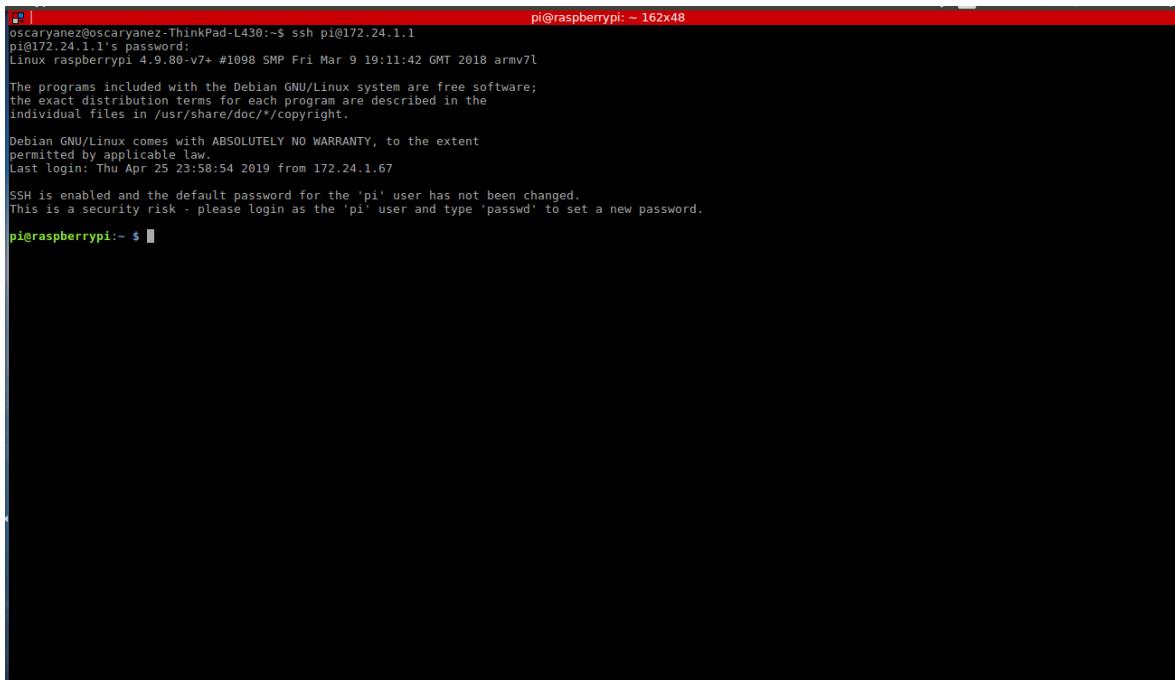
A screenshot of a terminal window titled 'oscaryanez@oscaryanez-ThinkPad-L430: ~ 162x48'. The window shows the command 'ssh pi@172.24.1.1' being run, followed by a password prompt 'pi@172.24.1.1's password: The password is partially obscured by a black redaction box.

2.- Nos pedirá una password el cual es: raspberry



A screenshot of a terminal window titled 'oscaryanez@oscaryanez-ThinkPad-L430: ~ 162x48'. The window shows the command 'ssh pi@172.24.1.1' being run, followed by a password prompt 'pi@172.24.1.1's password: The password 'raspberry' is being typed into the terminal, with the last few characters obscured by a black redaction box.

3.- Así se abrirá la consola de la raspberry que se encuentra en el R2-D2



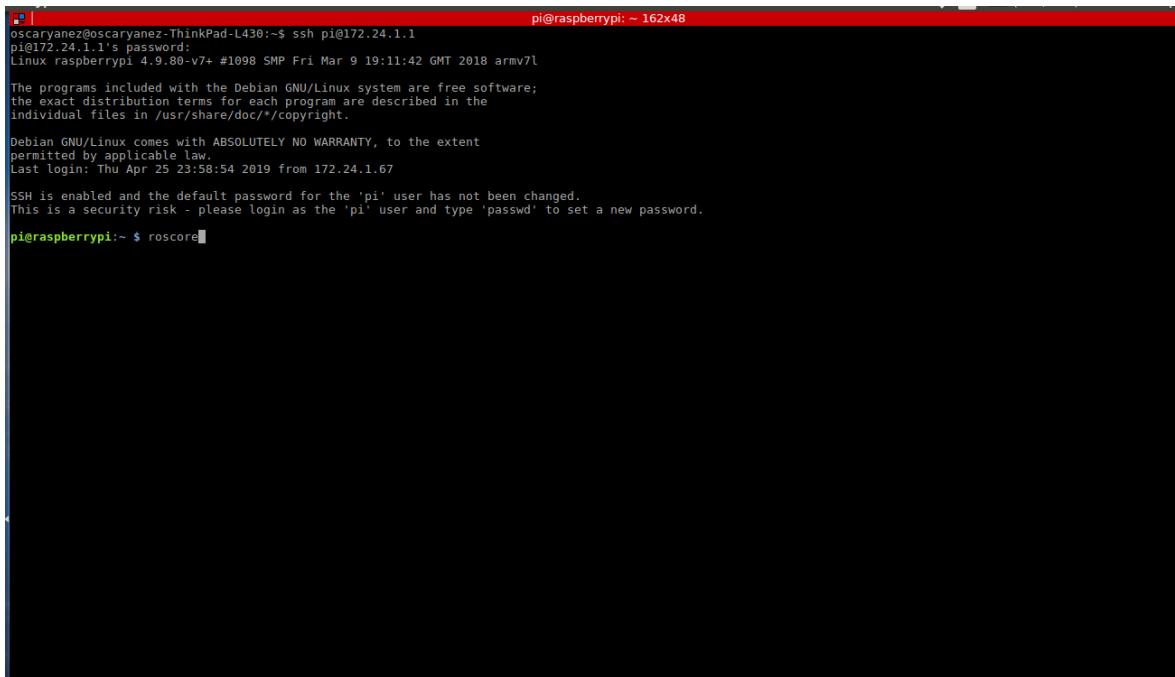
```
pi@raspberrypi: ~ 162x48
oscaryanez@oscaryanez-ThinkPad-L430:~$ ssh pi@172.24.1.1
pi@172.24.1.1's password:
Linux raspberrypi 4.9.80-v7+ #1098 SMP Fri Mar 9 19:11:42 GMT 2018 armv7l
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Apr 25 23:58:54 2019 from 172.24.1.67

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~ $
```

4.-Posteriormente ejecutamos ros con el comando: roscore



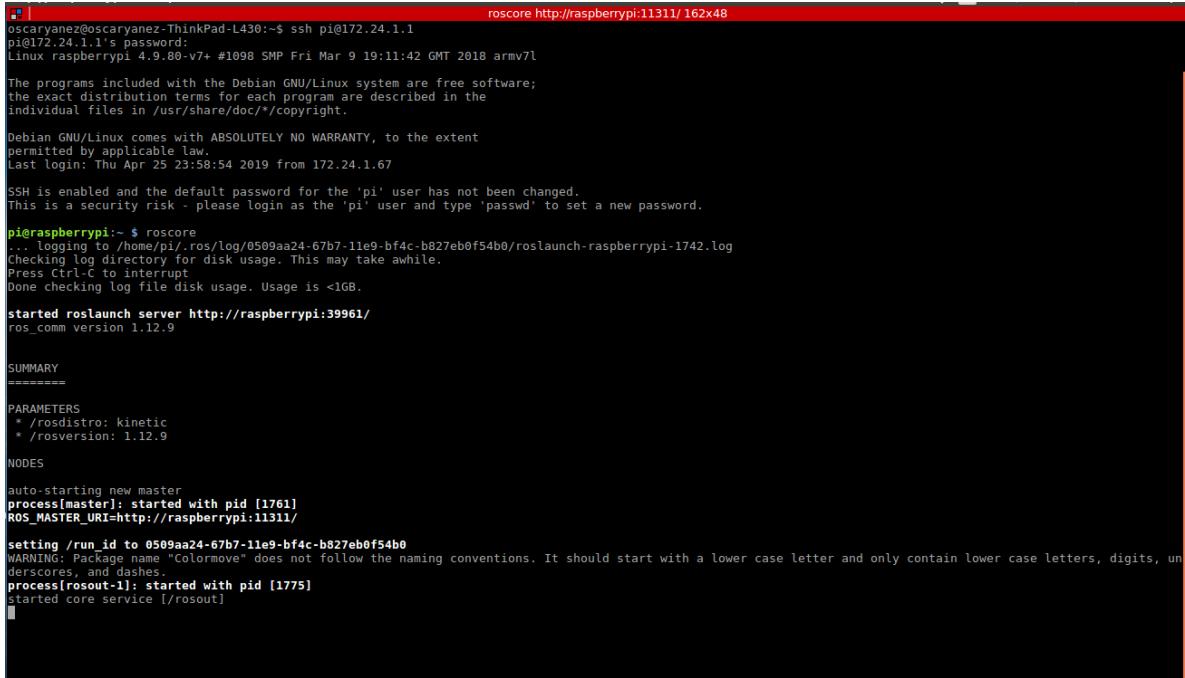
```
pi@raspberrypi: ~ 162x48
oscaryanez@oscaryanez-ThinkPad-L430:~$ ssh pi@172.24.1.1
pi@172.24.1.1's password:
Linux raspberrypi 4.9.80-v7+ #1098 SMP Fri Mar 9 19:11:42 GMT 2018 armv7l
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Apr 25 23:58:54 2019 from 172.24.1.67

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~ $ roscore
```

5.- En la consola veremos como se inicia el servicio de roscore



```
roscore http://raspberrypi:11311/ 162x48
oscaryanez@oscaryanez-ThinkPad-L430:~$ ssh pi@172.24.1.1
pi@172.24.1.1's password:
Linux raspberrypi 4.9.80-v7+ #1098 SMP Fri Mar 9 19:11:42 GMT 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Apr 25 23:58:54 2019 from 172.24.1.67

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~ $ roscore
... logging to /home/pi/.ros/log/0509aa24-67b7-11e9-bf4c-b827eb0f54b0/roslaunch-raspberrypi-1742.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://raspberrypi:39961/
ros_comm version 1.12.9

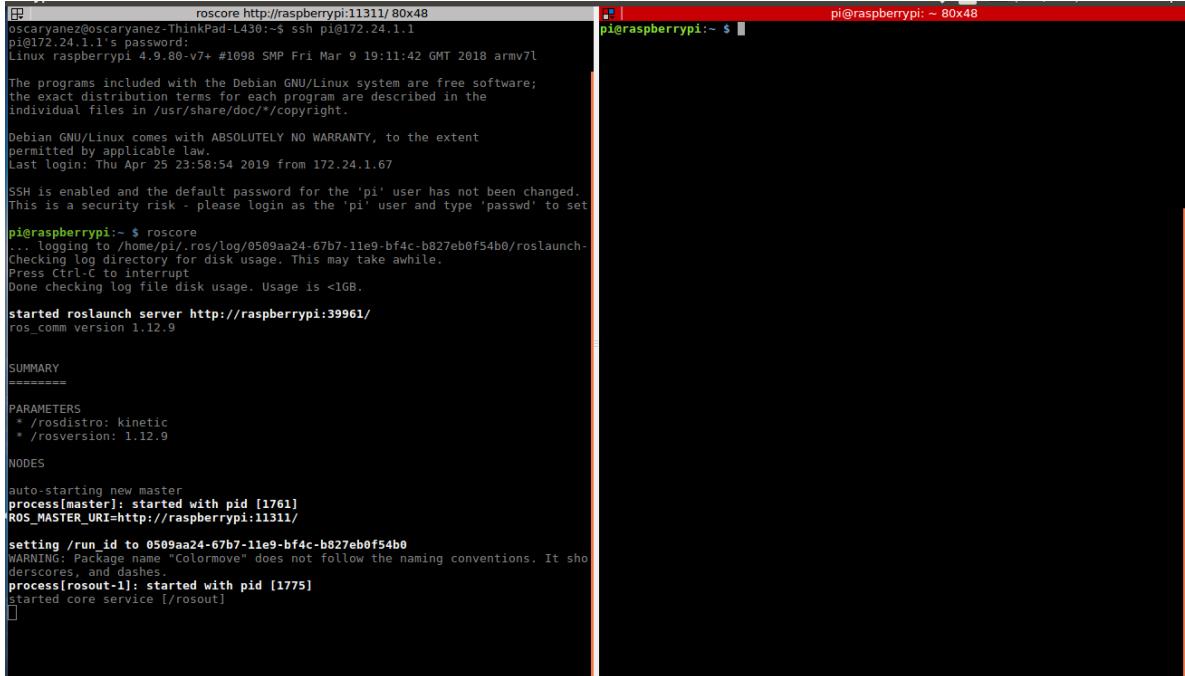
SUMMARY
=====

PARAMETERS
  * /rosdistro: kinetic
  * /rosversion: 1.12.9

NODES
auto-starting new master
process[master]: started with pid [1761]
ROS_MASTER_URI=http://raspberrypi:11311/
setting /run_id to 0509aa24-67b7-11e9-bf4c-b827eb0f54b0
WARNING: Package name "Colormove" does not follow the naming conventions. It should start with a lower case letter and only contain lower case letters, digits, underscores, and dashes.
process[rosout-1]: started with pid [1775]
started core service [/rosout]

```

6.- Posteriormente además de la terminal donde estamos ejecutando el roscore abriremos una nueva terminal de la raspberry pi repitiendo los pasos del 1 al 3.



```
roscore http://raspberrypi:11311/ 80x48
oscaryanez@oscaryanez-ThinkPad-L430:~$ ssh pi@172.24.1.1
pi@172.24.1.1's password:
Linux raspberrypi 4.9.80-v7+ #1098 SMP Fri Mar 9 19:11:42 GMT 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Apr 25 23:58:54 2019 from 172.24.1.67

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~ $ roscore
... logging to /home/pi/.ros/log/0509aa24-67b7-11e9-bf4c-b827eb0f54b0/roslaunch-raspberrypi-1742.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://raspberrypi:39961/
ros_comm version 1.12.9

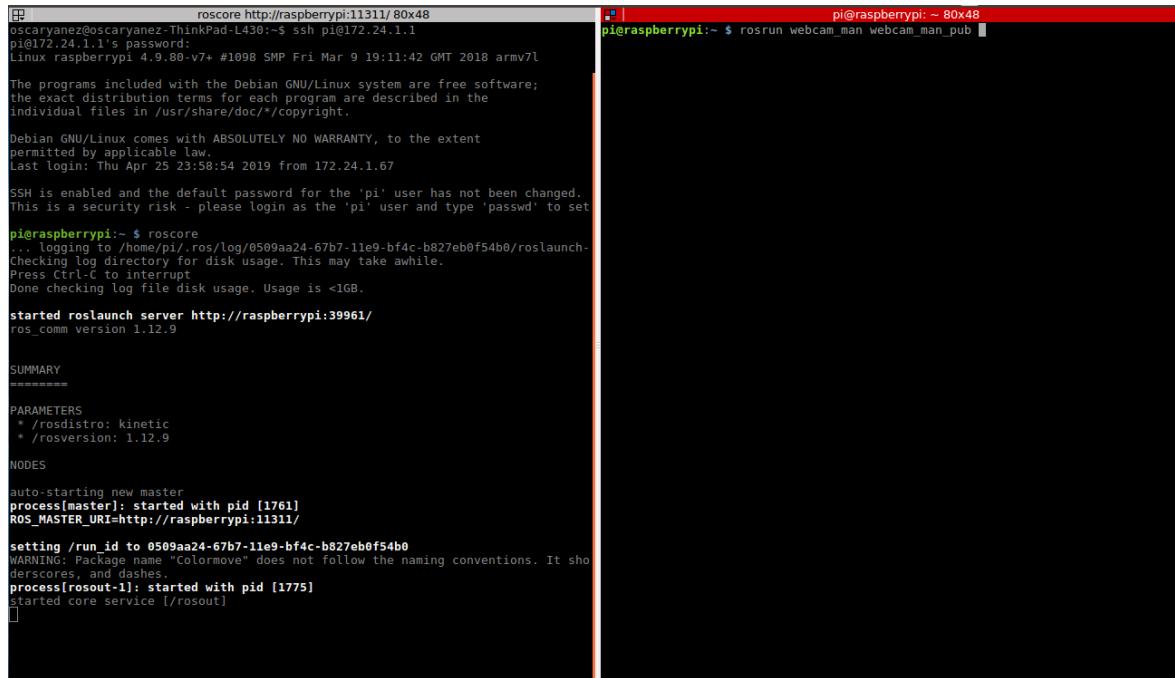
SUMMARY
=====

PARAMETERS
  * /rosdistro: kinetic
  * /rosversion: 1.12.9

NODES
auto-starting new master
process[master]: started with pid [1761]
ROS_MASTER_URI=http://raspberrypi:11311/
setting /run_id to 0509aa24-67b7-11e9-bf4c-b827eb0f54b0
WARNING: Package name "Colormove" does not follow the naming conventions. It should start with a lower case letter and only contain lower case letters, digits, underscores, and dashes.
process[rosout-1]: started with pid [1775]
started core service [/rosout]

```

7.- Para ejecutar el nodo que abre la cámara del R2-D2 en la terminal ejecutamos el comando: `rosrun webcam_man webcam_man_pub`



```
oscaryanez@oscaryanez-ThinkPad-L430:~$ ssh pi@172.24.1.1
pi@172.24.1.1's password:
Linux raspberrypi 4.9.80-v7+ #1098 SMP Fri Mar 9 19:11:42 GMT 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Apr 25 23:58:54 2019 from 172.24.1.67

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
pi@raspberrypi:~ $ roscore
... logging to /home/pi/.ros/log/0509aa24-67b7-11e9-bf4c-b827eb0f54b0/roslaunch-
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://raspberrypi:39961/
ros_comm version 1.12.9

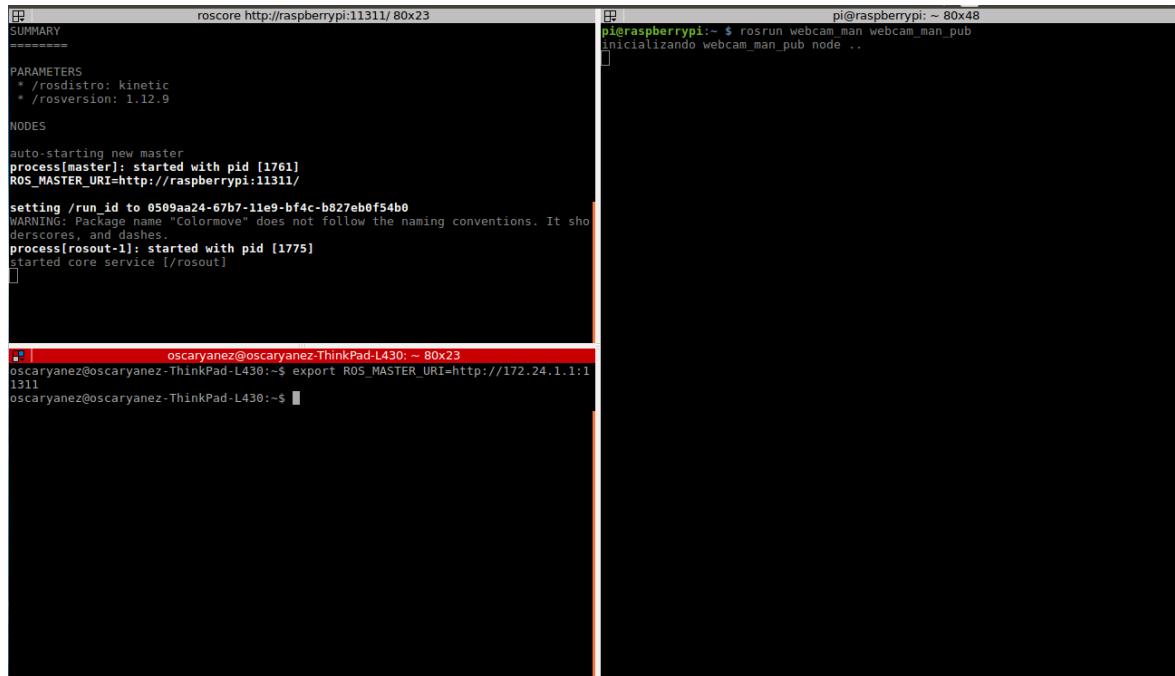
SUMMARY
=====
PARAMETERS
  * /rosdistro: kinetic
  * /rosversion: 1.12.9

NODES
auto-starting new master
process[master]: started with pid [1761]
ROS_MASTER_URI=http://raspberrypi:11311/
setting /run_id to 0509aa24-67b7-11e9-bf4c-b827eb0f54b0
WARNING: Package name "Colormove" does not follow the naming conventions. It sh
derscores, and dashes.
process[rosout-1]: started with pid [1775]
started core service [/rosout]
]

pi@raspberrypi:~ $ rosrun webcam_man webcam_man_pub
pi@raspberrypi:~ 80x48
```

8.- Para visualizar la imagen de la cámara en nuestra computadora debemos abrir una nueva terminal de nuestro equipo y ejecutar el comando:

```
export ROS_MASTER_URI=http://172.24.1.1:11311
```



```
oscaryanez@oscaryanez-ThinkPad-L430:~ 80x23
oscaryanez@oscaryanez-ThinkPad-L430:~ $ export ROS_MASTER_URI=http://172.24.1.1:11311
oscaryanez@oscaryanez-ThinkPad-L430:~ $ 
```

```
oscaryanez@oscaryanez-ThinkPad-L430:~ 80x23
oscaryanez@oscaryanez-ThinkPad-L430:~ $ ssh pi@172.24.1.1
pi@172.24.1.1's password:
Linux raspberrypi 4.9.80-v7+ #1098 SMP Fri Mar 9 19:11:42 GMT 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Apr 25 23:58:54 2019 from 172.24.1.67

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
pi@raspberrypi:~ $ roscore
... logging to /home/pi/.ros/log/0509aa24-67b7-11e9-bf4c-b827eb0f54b0/roslaunch-
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

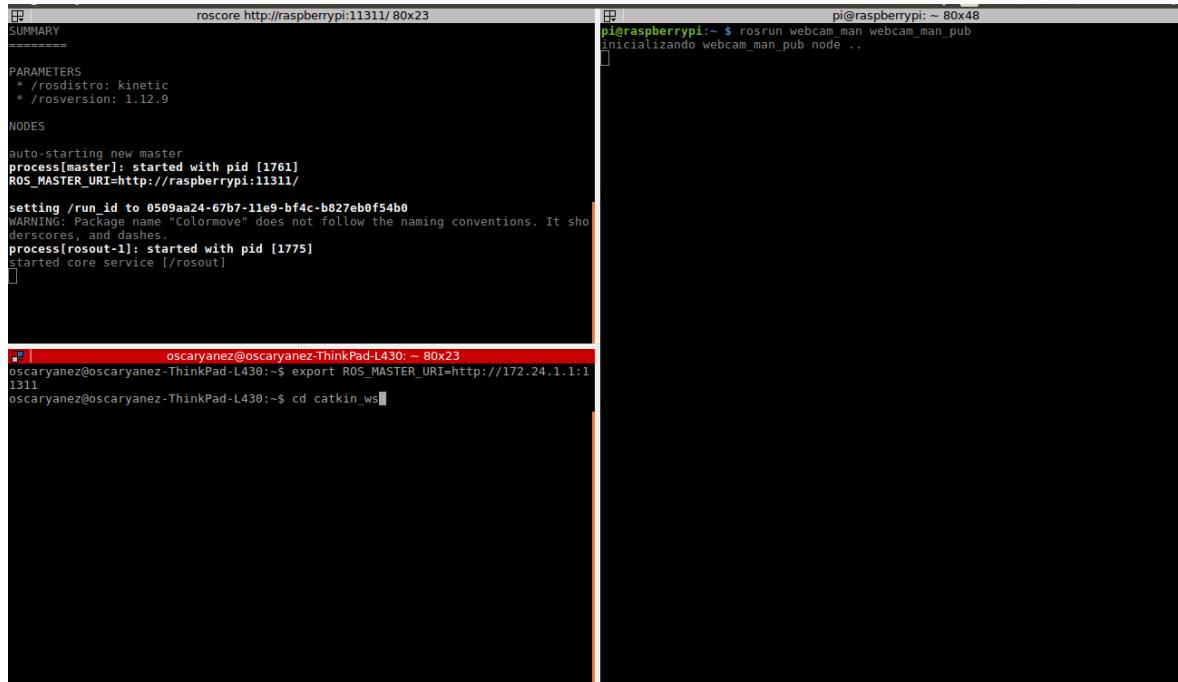
started roslaunch server http://raspberrypi:39961/
ros_comm version 1.12.9

SUMMARY
=====
PARAMETERS
  * /rosdistro: kinetic
  * /rosversion: 1.12.9

NODES
auto-starting new master
process[master]: started with pid [1761]
ROS_MASTER_URI=http://raspberrypi:11311/
setting /run_id to 0509aa24-67b7-11e9-bf4c-b827eb0f54b0
WARNING: Package name "Colormove" does not follow the naming conventions. It sh
derscores, and dashes.
process[rosout-1]: started with pid [1775]
started core service [/rosout]
]

pi@raspberrypi:~ $ rosrun webcam_man webcam_man_pub
pi@raspberrypi:~ 80x48
```

9.- Posteriormente nos movemos al workspace con el comando: cd catkin_ws

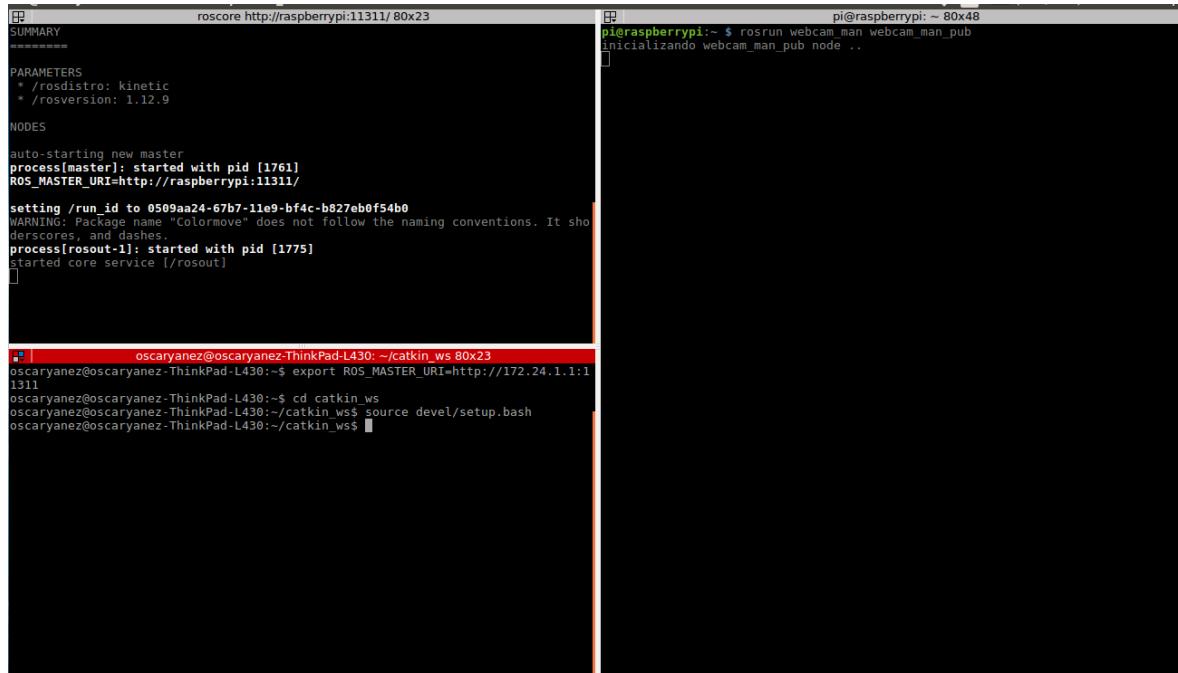


```
roscore http://raspberrypi:11311/ 80x23
SUMMARY
=====
PARAMETERS
  * /rosdistro: kinetic
  * /rosversion: 1.12.9
NODES
auto-starting new master
process[master]: started with pid [1761]
ROS_MASTER_URI=http://raspberrypi:11311/
setting /run_id to 0509aa24-67b7-11e9-bf4c-b827eb0f54b0
WARNING: Package name "Colormove" does not follow the naming conventions. It should
have underscores, and dashes.
process[rosout-1]: started with pid [1775]
started core service [/rosout]

pi@raspberrypi:~ 80x48
pi@raspberrypi:~ $ rosrun webcam man webcam_man_pub
inicializando webcam_man_pub node ...
]

oscaryanez@oscaryanez-ThinkPad-L430:~ 80x23
oscaryanez@oscaryanez-ThinkPad-L430:~$ export ROS_MASTER_URI=http://172.24.1.1:11311
oscaryanez@oscaryanez-ThinkPad-L430:~$ cd catkin_ws
```

10.- Posteriormente ejecutamos el comando: source devel/setup.bash



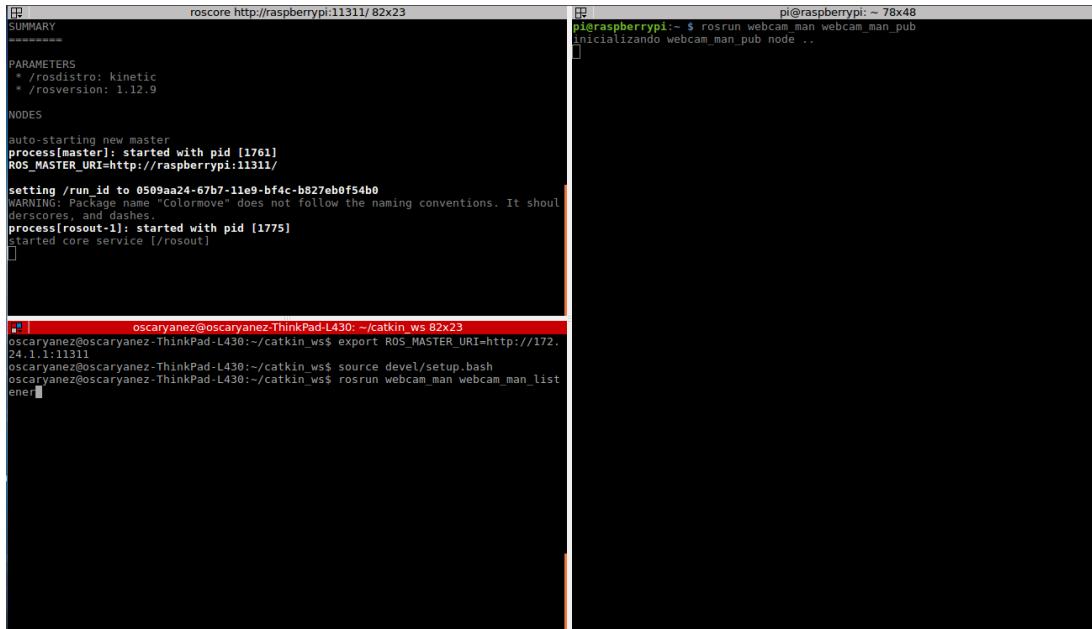
```
roscore http://raspberrypi:11311/ 80x23
SUMMARY
=====
PARAMETERS
  * /rosdistro: kinetic
  * /rosversion: 1.12.9
NODES
auto-starting new master
process[master]: started with pid [1761]
ROS_MASTER_URI=http://raspberrypi:11311/
setting /run_id to 0509aa24-67b7-11e9-bf4c-b827eb0f54b0
WARNING: Package name "Colormove" does not follow the naming conventions. It should
have underscores, and dashes.
process[rosout-1]: started with pid [1775]
started core service [/rosout]

pi@raspberrypi:~ 80x48
pi@raspberrypi:~ $ rosrun webcam man webcam_man_pub
inicializando webcam_man_pub node ...
]

oscaryanez@oscaryanez-ThinkPad-L430:~ 80x23
oscaryanez@oscaryanez-ThinkPad-L430:~$ export ROS_MASTER_URI=http://172.24.1.1:11311
oscaryanez@oscaryanez-ThinkPad-L430:~$ cd catkin_ws
oscaryanez@oscaryanez-ThinkPad-L430:~/catkin_ws$ source devel/setup.bash
oscaryanez@oscaryanez-ThinkPad-L430:~/catkin_ws$
```

11.- Para ejecutar el nodo que abre la imagen de cámara del R2-D2 en nuestra computadora ejecutamos el nodo que recibe la imagen con el comando:

```
rosrun webcam_man webcam_man_listener
```

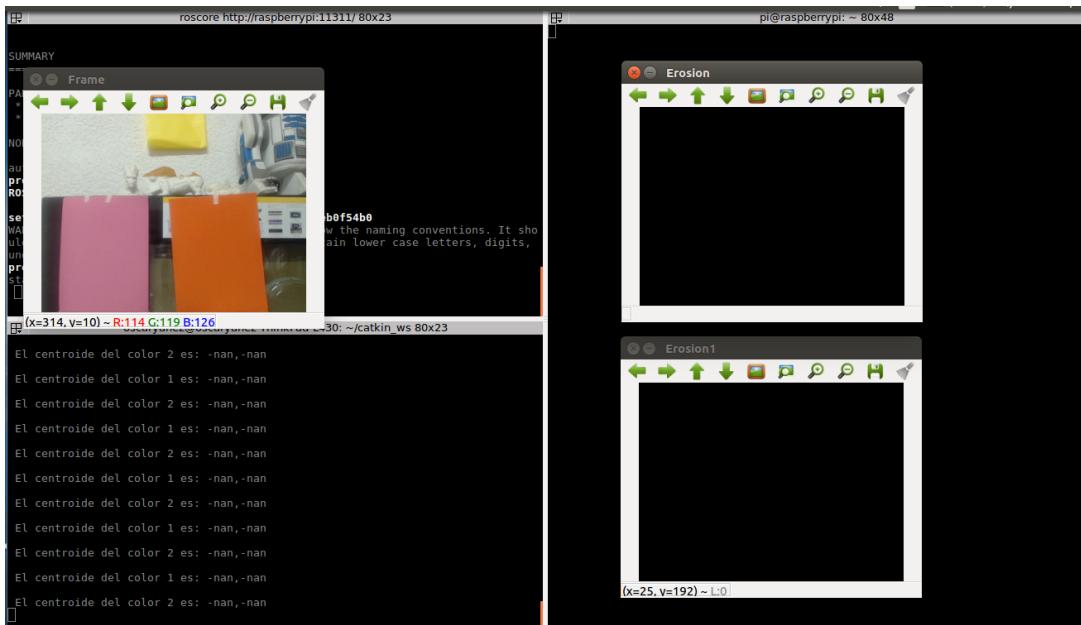


```
roscore http://raspberrypi:11311/ 82x23
SUMMARY
=====
PARAMETERS
  /rosdistro: kinetic
  * /rosversion: 1.12.9
NODES
auto-starting new master
process[master]: started with pid [1761]
ROS_MASTER_URI=http://raspberrypi:11311/
setting /run_id to 0509a24-67b7-11e9-bf4c-b827eb0f54b0
[WARNING] [roscore]: 'Colormove' does not follow the naming conventions. It should
use underscores, and dashes.
process[rosout-1]: started with pid [1775]
  started core service [/rosout]
```

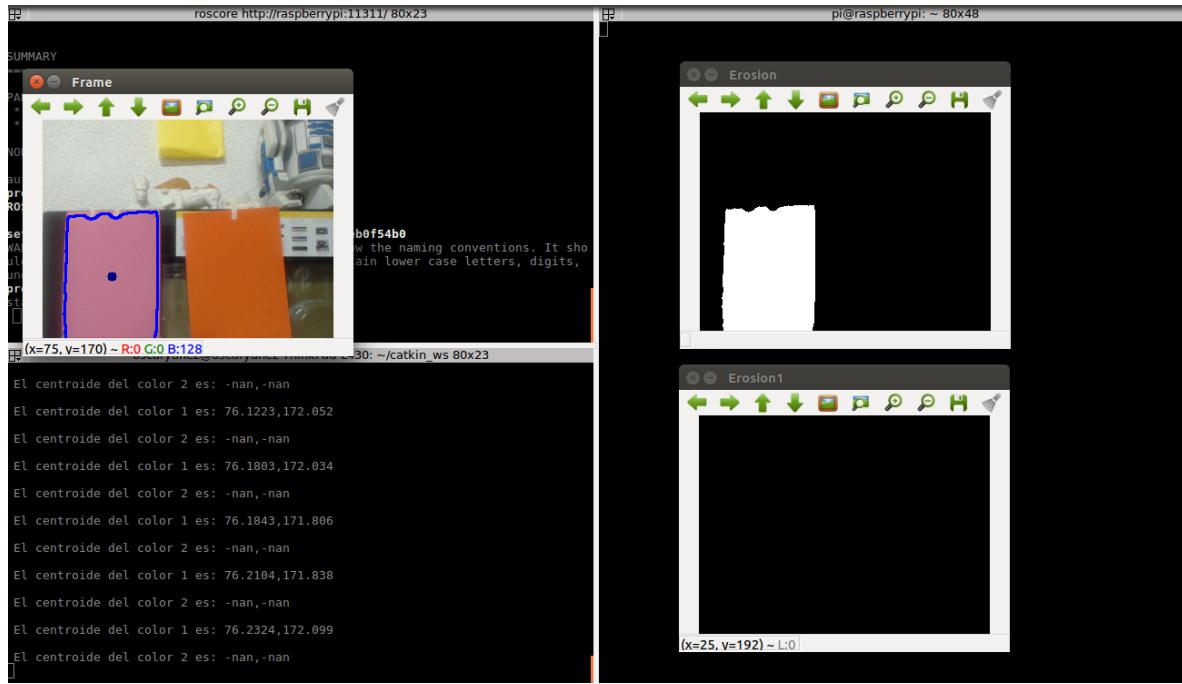


```
pi@raspberrypi:~ 78x48
pi@raspberrypi:~ $ rosrun webcam man webcam_man_pub
inicializando webcam_man_pub node ..
```

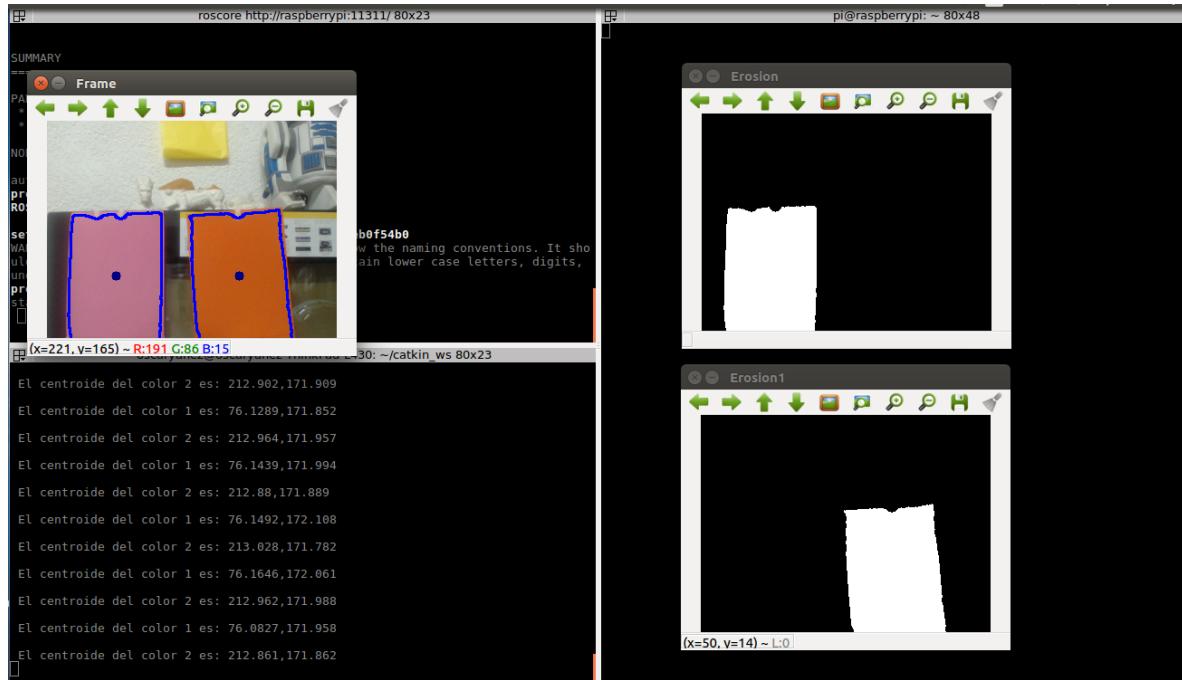
12.- Se abrirá la imagen original en una ventana con el nombre Frame y otras 2 ventanas donde se verán independiente los 2 colores filtrados, mientras tanto en la terminal nos aparecerá un mensaje el cual nos indicará los centroides.



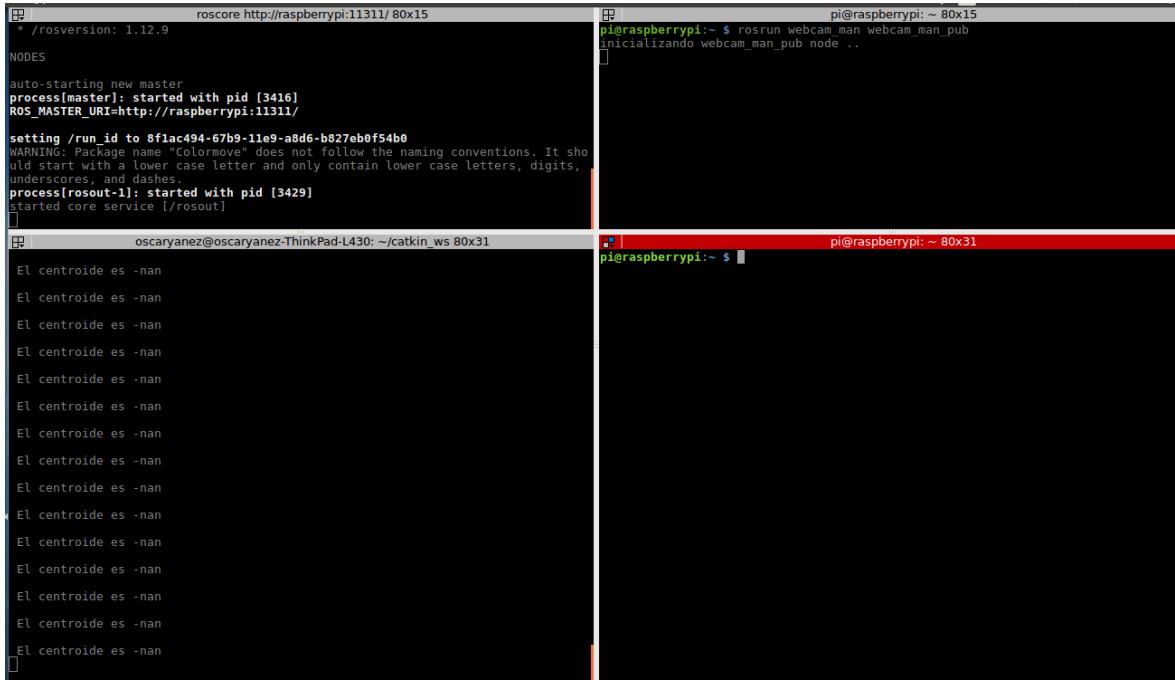
13.- Para filtrar el primer color debemos dirigirnos al objeto de determinado color y darle clic izquierdo.



14.- Para filtrar el segundo color debemos dirigirnos al objeto de determinado color y darle clic derecho.

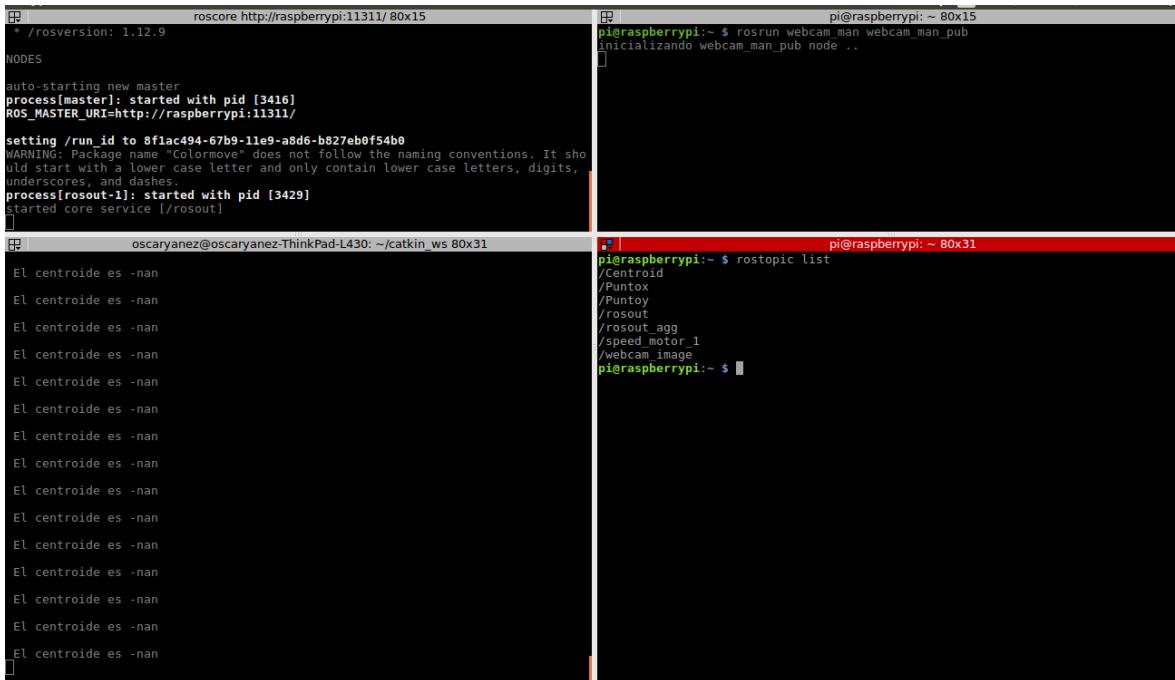


15.- Para hacer la conexión con el Arduino Mega y recibir los tópicos de los sensores y publicar para que se muevan los actuadores abrimos una nueva terminal.



The image shows two terminal windows side-by-side. The left terminal window shows the output of the command `roscore http://raspberrypi:11311`. It displays the ROS master starting up, including the master's IP address (192.168.1.113), the port (11311), and the ROS_MASTER_URI. It also shows the creation of a new master process and the start of the core service. The right terminal window shows the command `rosrun webcam man webcam_man_pub` being run. The output indicates that the `webcam_man_pub` node is being initialized.

16.- Verificamos los tópicos antes de la conexión con el Arduino con el comando:
`rostopic list`



The image shows two terminal windows side-by-side. The left terminal window shows the output of the command `rostopic list`. It lists several topics: `/Centroid`, `/Puntox`, `/Puntoy`, `/rosout`, `/rosout_agg`, `/speed_motor_1`, and `/webcam_image`. The right terminal window shows the command `rostopic list` being run again, with the same output as the left window.

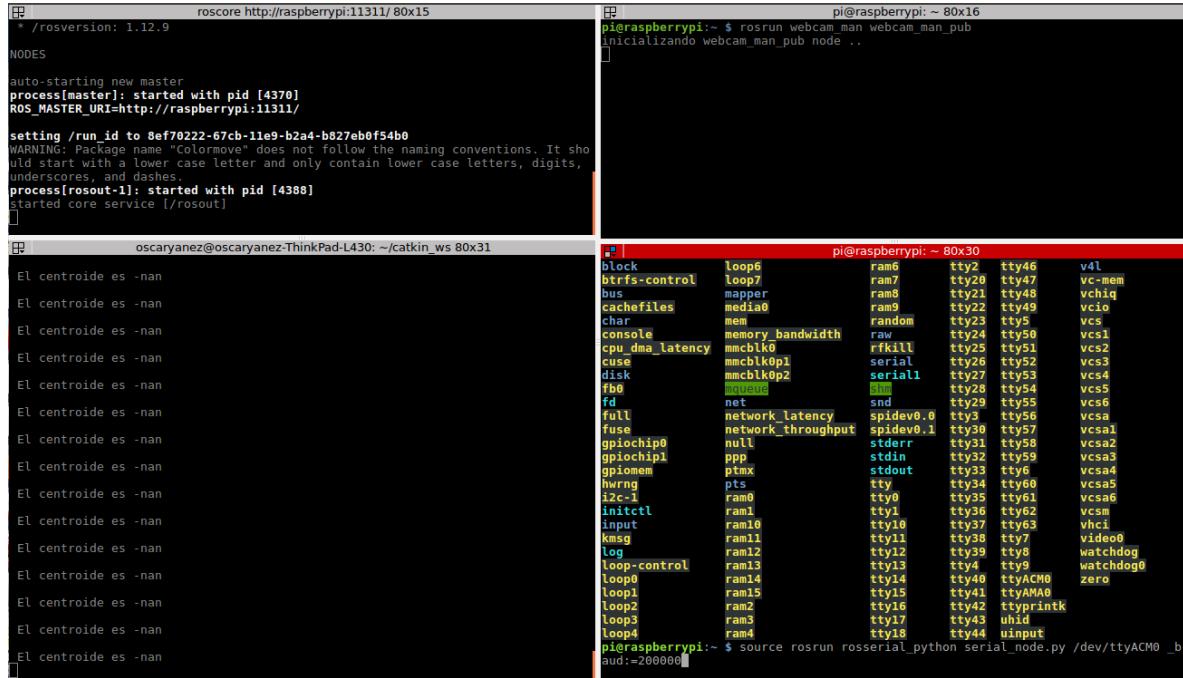
17.- Para encontrar el puerto donde está conectado el Arduino a la raspberry ejecutamos el comando: ls /dev

18.- Generalmente el Arduino lo podremos encontrar como ttyACM y un numero que en este caso es 0 siendo el dispositivo ttyACM0

19.- Para iniciar la comunicación de comunicación serial con Arduino ejecutamos el siguiente comando:

```
source rosrun rosserial_python serial_node.py /dev/ttyACM0 _baud:=200000
```

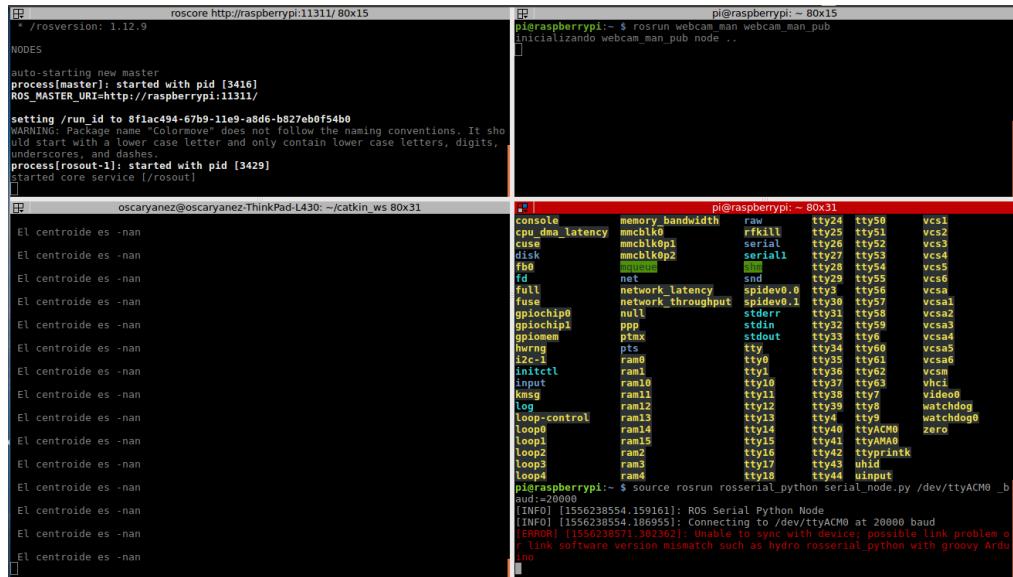
donde /dev/ttyAM0 es el dispositivo encontrado en la instrucción anterior (18) y _baud:=200000 es el numero de bits por segundo el cual debe coincidir con el programa de Arduino.



```
roscore http://raspberrypi:11311/ 80x15
* /rosversion: 1.12.9
NODES
auto-starting new master
process[master]: started with pid [4370]
ROS_MASTER_URI=http://raspberrypi:11311/
setting /run_id to 8ef70222-67cb-11e9-b2a4-b827eb0f54b0
WARNING: Package name "Colormove" does not follow the naming conventions. It should start with a lower case letter and only contain lower case letters, digits, underscores, and dashes.
process[rosout-1]: started with pid [4388]
started core service [/rosout]

oscaryanez@oscaryanez:~$ source rosrun rosserial_python serial_node.py /dev/ttyACM0 _baud:=200000
pi@raspberrypi: ~ 80x16
pi@raspberrypi:~ $ rosrun webcam man webcam_man_pub
inicializando webcam_man_pub node ...
```

20.- Si llegara a salir el siguiente error ir a la sección de errores si no pasar a la siguiente instrucción (21)



```
roscore http://raspberrypi:11311/ 80x15
* /rosversion: 1.12.9
NODES
auto-starting new master
process[master]: started with pid [3416]
ROS_MASTER_URI=http://raspberrypi:11311/
setting /run_id to 8fac494-67b9-11e9-a8d6-b827eb0f54b0
WARNING: Package name "Colormove" does not follow the naming conventions. It should start with a lower case letter and only contain lower case letters, digits, underscores, and dashes.
process[rosout-1]: started with pid [3429]
started core service [/rosout]

oscaryanez@oscaryanez:~$ source rosrun rosserial_python serial_node.py /dev/ttyACM0 _baud:=200000
pi@raspberrypi: ~ 80x15
pi@raspberrypi:~ $ rosrun webcam man webcam_man_pub
inicializando webcam_man_pub node ...
```

21.- Si no se presentó ningún error se iniciará la comunicación como se muestra a continuación.

22.- Para acceder a los tópicos accedemos a una nueva terminal de la raspberry repitiendo los pasos del 1 al 3.

23.- Verificamos que se encuentren los topicos del Arduino con el comando: rostopic list

Si la conexión con el Arduino esta bien encontraremos mas topicos que en el paso 17 los cuales son los topicos del Arduino.

24.- Para verificar el valor del Sharp 1 ejecutamos el comando:

```
rostopic echo /sharpSensor_1
```

25.- Una vez levantado el echo del Sharp 1 observaremos los valores de este en la consola como se muestra a continuación.

```

pi@raspberrypi:~ 80x15
* /rosversion: 1.12.9
NODES
auto-starting new master
process[master]: started with pid [4370]
ROS_MASTER_URI=http://raspberrypi:11311/
setting /run id to 8ef70222-67cb-11e9-b2a4-b827eb0f54b0
WARNING: Package name "Colormove" does not follow the naming conventions. It should start with a lower case letter and only contain lower case letters, digits, underscores, and dashes.
process[rosout-1]: started with pid [4388]
started core service [/rosout]

pi@raspberrypi:~ 80x16
pi@raspberrypi:~ $ rosrun webcam man webcam_man_pub
inicializando webcam_man_pub node ..

pi@raspberrypi:~ 80x14
pi@raspberrypi:~ $ rostopic echo /sharpSensor_1
data: 10
data: 4
data: 3
data: 7
data: 20
data: 15

pi@raspberrypi:~ 80x14
pi@raspberrypi:~ $ rostopic echo /sharpSensor_2
data: 10
data: 4
data: 3
data: 7
data: 20
data: 15

```

26.- Para verificar el valor del Sharp 2 ejecutamos el comando:

rostopic echo /sharpSensor_2

```

pi@raspberrypi:~ 80x15
* /rosversion: 1.12.9
NODES
auto-starting new master
process[master]: started with pid [4370]
ROS_MASTER_URI=http://raspberrypi:11311/
setting /run id to 8ef70222-67cb-11e9-b2a4-b827eb0f54b0
WARNING: Package name "Colormove" does not follow the naming conventions. It should start with a lower case letter and only contain lower case letters, digits, underscores, and dashes.
process[rosout-1]: started with pid [4388]
started core service [/rosout]

pi@raspberrypi:~ 80x16
pi@raspberrypi:~ $ rosrun webcam man webcam_man_pub
inicializando webcam_man_pub node ..

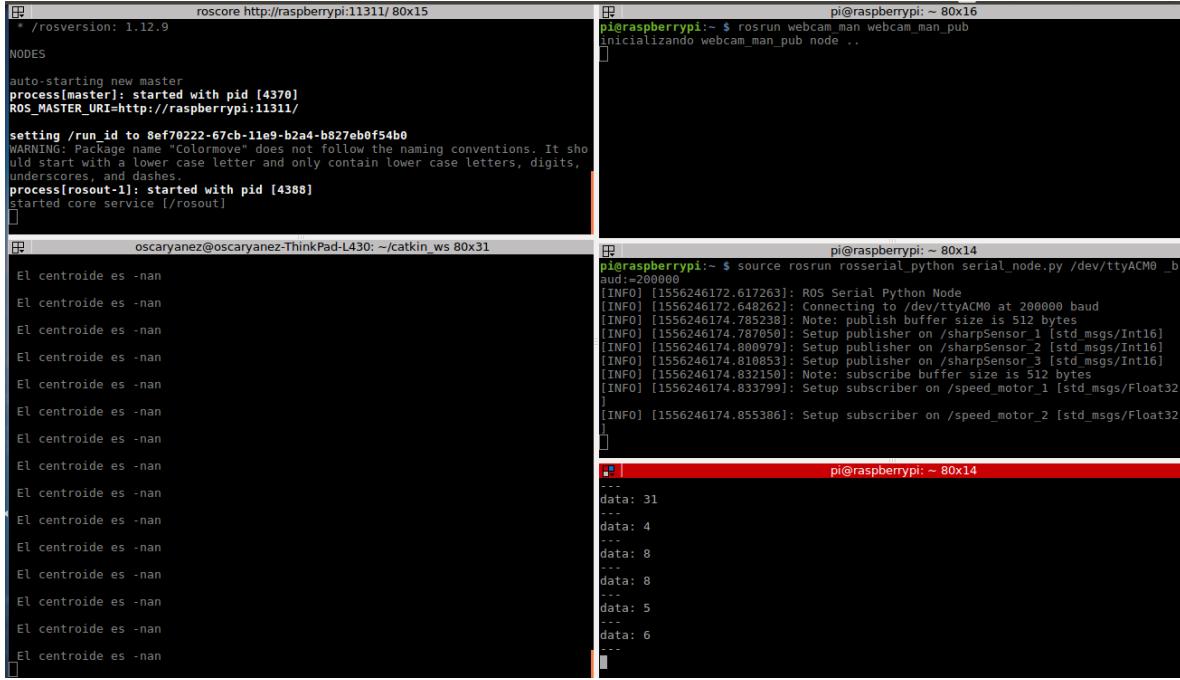
pi@raspberrypi:~ 80x14
pi@raspberrypi:~ $ source rosrun rosserial_python serial_node.py /dev/ttyACM0 _b
aud:=200000
[INFO] [1556246172.617263]: ROS Serial Python Node
[INFO] [1556246172.648262]: Connecting to /dev/ttyACM0 at 200000 baud
[INFO] [1556246174.785238]: Note: publish buffer size is 512 bytes
[INFO] [1556246174.787050]: Setup publisher on /sharpSensor_1 [std_msgs/Int16]
[INFO] [1556246174.800979]: Setup publisher on /sharpSensor_2 [std_msgs/Int16]
[INFO] [1556246174.810853]: Setup publisher on /sharpSensor_3 [std_msgs/Int16]
[INFO] [1556246174.832150]: Note: subscribe buffer size is 512 bytes
[INFO] [1556246174.833799]: Setup subscriber on /speed_motor_1 [std_msgs/Float32]
[INFO] [1556246174.855386]: Setup subscriber on /speed_motor_2 [std_msgs/Float32]

pi@raspberrypi:~ 80x14
pi@raspberrypi:~ $ rostopic list
/Centroid
/Puntox
/Puntoy
/diagnostics
/rosout
/rosout_agg
/sharpSensor_1
/sharpSensor_2
/sharpSensor_3
/speed_motor_1
/speed_motor_2
/webcam_image

pi@raspberrypi:~ $ rostopic echo /sharpSensor_2

```

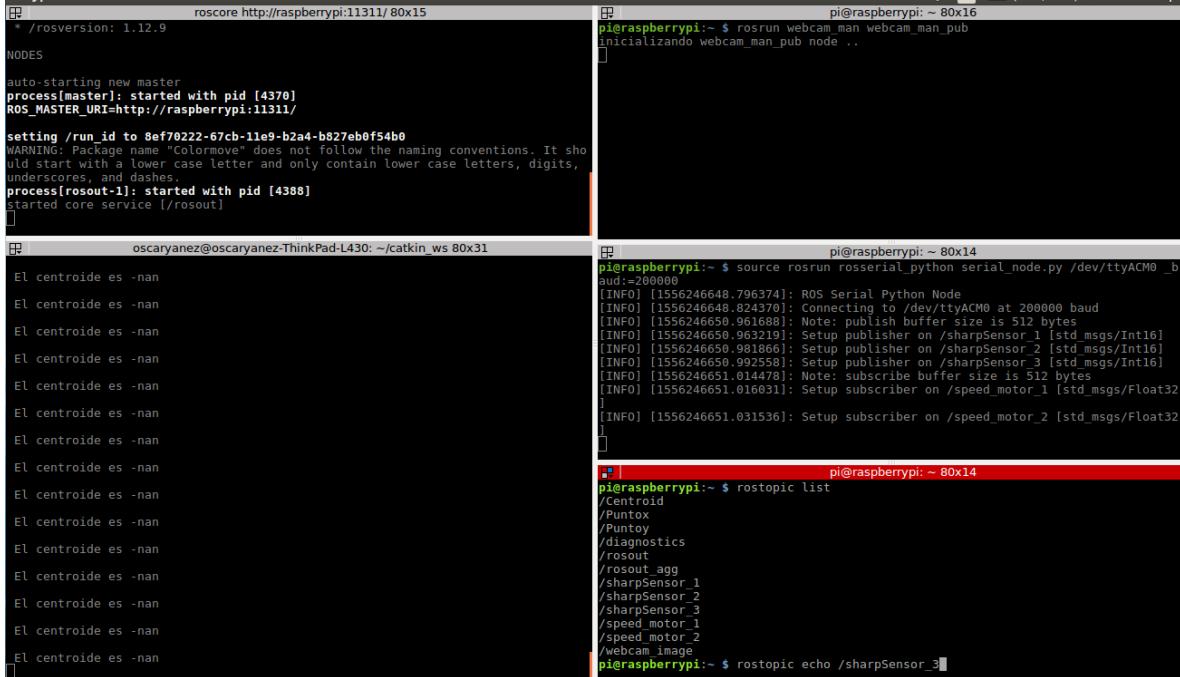
27.- Una vez levantado el echo del Sharp 2 observaremos los valores de este en la consola como se muestra a continuación.



The terminal window displays two panes. The left pane shows the output of the command `roscore http://raspberrypi:11311/ 80x15`, which starts a ROS master node and lists several nodes including `process[master]`, `process[rosout-1]`, and `process[rosout]`. The right pane shows the output of the command `rosrun webcam man webcam_man_pub`, which initializes a node and then displays a continuous stream of data from a serial port. The data consists of integer values (31, 4, 8, 8, 5, 6, ...) representing sensor readings.

28.- Para verificar el valor del Sharp 3 ejecutamos el comando:

```
rostopic echo /sharpSensor_3
```



The terminal window displays two panes. The left pane shows the output of the command `roscore http://raspberrypi:11311/ 80x15`, similar to the previous screenshot. The right pane shows the output of the command `rostopic list`, which lists all the topics currently published on the ROS bus. The list includes `/Centroid`, `/Puntox`, `/Puntoy`, `/Diagnostics`, `/rosout`, `/rosout_agg`, `/sharpSensor_1`, `/sharpSensor_2`, `/sharpSensor_3`, `/speed_motor_1`, `/speed_motor_2`, and `/webcam_image`. The command `rostopic echo /sharpSensor_3` is also shown at the bottom.

29.- Una vez levantado el echo del Sharp 2 observaremos los valores de este en la consola como se muestra a continuación.

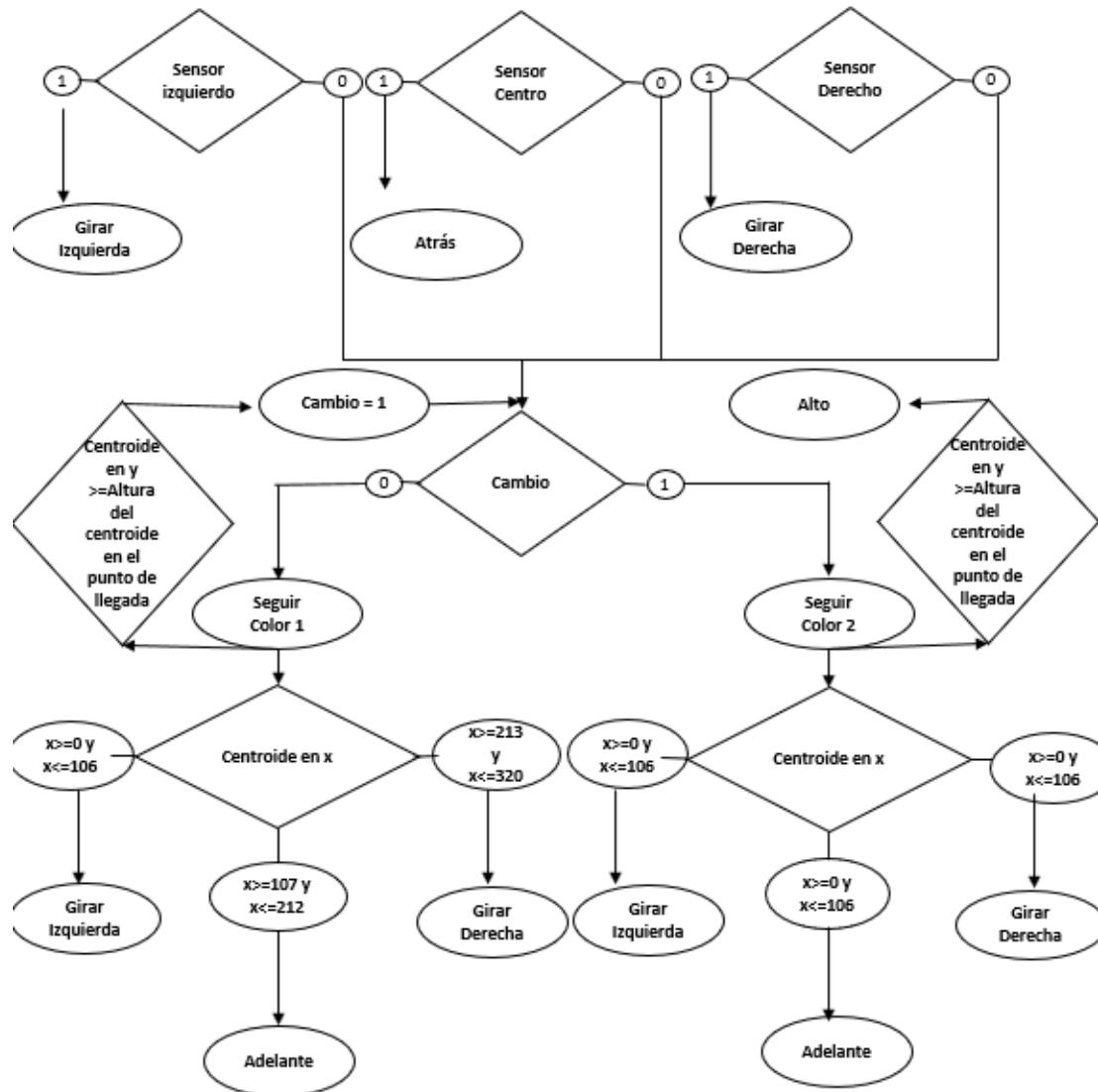
30.- Para probar los motores ejecutamos el siguiente comando:

```
rostopic pub /speed_motor_1 std_msgs/Float32 "data: 1"
```

Donde el 1 de /speed_motor_1 es el número de motor que puede ser 1 o 2 y el data 1 es un flotante que va de 0 a 1 dependiendo el pwm del motor

31.- Para correr el comportamiento de seguir un color y evadir obstáculos ejecutar el comando: `rosrun Colormove Colormove_node`

4.- CARTA ASM DE LA MAQUINA DE ESTADOS PARA SEGUIR 2 COLORES Y EVADIR OBSTÁCULOS



5.- NODOS QUE HAY QUE CORRER PARA CORRER EL NODO QUE SIGUE COLORES Y EVADE OBSTÁCULOS

Para ejecutar el rosrun Colormove Colormove_node debemos tener inicializado en una ventana de la raspberry ros, el nodo que publica la imagen, el nodo de serial Python que hace la comunicación con el Arduino y en nuestra computadora con el export_ROSMASTER el nodo de listener que hace el filtrado.



6.- CODIGO DE ARDUINO

```
#include <ros.h> //incluimos libreria para utilizar ROS.
#include <std_msgs/Float32.h> //incluimos libreria para un número
flotante de 32bits.
#include <std_msgs/Int64.h> //incluimos libreria para un número entero
de 64bits.
#include <std_msgs/Int16.h> //incluimos libreria para un número entero
de 16 bits.
#include <std_msgs/Int16MultiArray.h> //incluimos libreria para arreglos
de números de 12bits.

//Conexiones de motor 1
#define D00 40
#define D01 41
//Conexiones de motor 2
#define D10 39
#define D11 38
//Conexiones de enable
#define E1 4
#define E2 9

#define BAUD 200000 //se define el baudaje entre arduino y ROS, tienen
que coinsidir.

ros::NodeHandle nh;

std_msgs::Int16 sharpSensor1; //mensaje del sensor1 en enteros de 16bits
std_msgs::Int16 sharpSensor2; //mensaje del sensor2 en enteros de 16bits
std_msgs::Int16 sharpSensor3; //mensaje del sensor3 en enteros de 16bits

ros::Publisher sharpSensorPub1("/sharpSensor_1",
&sharpSensor1); //Etiquetamos el sensor1 con el nombre de sharpSensor_1 y
asi lo identificara ROS
ros::Publisher sharpSensorPub2("/sharpSensor_2",
&sharpSensor2); //Etiquetamos el sensor1 con el nombre de sharpSensor_2 y
asi lo identificara ROS
ros::Publisher sharpSensorPub3("/sharpSensor_3",
&sharpSensor3); //Etiquetamos el sensor1 con el nombre de sharpSensor_3 y
asi lo identificara ROS
```

```
void speedMotor1Callback(const std_msgs::Float32& mess){//Llamada a la
funcion por medio de ROS que realiza el movimiento de un motor
if(mess.data > 0){
    digitalWrite(D00, HIGH); //40
    digitalWrite(D01, LOW); //41
}
else{
    digitalWrite(D00, LOW);
    digitalWrite(D01, HIGH);
}
}
```

```
void speedMotor2Callback(const std_msgs::Float32& mess){//Llamada a la
funcion por medio de ROS que realiza el movimiento de un motor
if(mess.data > 0){
    digitalWrite(D10, HIGH); //39
    digitalWrite(D11, LOW); //38
}
else{
    digitalWrite(D10, LOW);
    digitalWrite(D11, HIGH);
}
}
```

```
ros::Subscriber<std_msgs::Float32> subSpeedMotor1("/speed_motor_1",
speedMotor1Callback);//Se subscribe a las velocidades del motor1
ros::Subscriber<std_msgs::Float32> subSpeedMotor2("/speed_motor_2",
speedMotor2Callback);//Se subscribe a las velocidades del motor2
```

```
void setup() {
    nh.getHardware()->setBaud(BAUD);
    nh.initNode();

    //Mensajes que envia
    nh.advertise(sharpSensorPub1);
    nh.advertise(sharpSensorPub2);
    nh.advertise(sharpSensorPub3);
    nh.subscribe(subSpeedMotor1);
```

```
nh.subscribe(subSpeedMotor2);

//Declaramos los pines como salida
pinMode(D00, OUTPUT);
pinMode(D01, OUTPUT);
pinMode(E1, OUTPUT);
pinMode(D10, OUTPUT);
pinMode(D11, OUTPUT);
pinMode(E2, OUTPUT);

}

void loop() {

sharpSensor1.data = analogRead(A0); //Sensor 1 conectado al pin A0
sharpSensor2.data = analogRead(A1); //Sensor 2 conectado al pin A1
sharpSensor3.data = analogRead(A5); //Sensor 3 conectado al pin A5

sharpSensorPub1.publish(&sharpSensor1); //publica el sensor sharp1
sharpSensorPub2.publish(&sharpSensor2); //publica el sensor sharp2
sharpSensorPub3.publish(&sharpSensor3); //publica el sensor sharp3

nh.spinOnce();
delay(20);
}
```

7.- PROGRAMA QUE ABRE LA IMAGEN EN LA COMPUTADORA Y HACER EL FILTRADO DEL COLOR

```
#include <ros/ros.h>
#include <std_msgs/Int16MultiArray.h>
#include <std_msgs/Int16.h>
#include <opencv2/opencv.hpp>
#include <sensor_msgs/Image.h>
#include <cv_bridge/cv_bridge.h>
#include <std_msgs/Float32.h>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
#include <vector>
#include <stdlib.h>
#include <stdio.h>

using namespace std;
int S,H,V;
int SL=256;//Valor mínimo
int SU=0;//Valor máximo
int HL=256;
int HU=0;
int VL=256;
int VU=0;
int contour_index;
ros::Publisher pubpunto; //Arreglo que contiene los 2 puntos del primer centroide
ros::Publisher pubpunto1; //Arreglo que contiene los 2 puntos del segundo centroide

ros::Publisher pubpuntox; //Arreglo que contiene el punto x
ros::Publisher pubpuntox1; //Arreglo que contiene el punto x1

ros::Publisher pubpuntoy; //Arreglo que contiene el punto y
ros::Publisher pubpuntoy1; //Arreglo que contiene el punto y1

ros::Publisher pubSpeedLeft;

cv::Mat mask;
cv::Mat img_rgb,todo;
cv::Mat img_hsv;
```

```

cv::Mat contours;
cv::Mat gray_image;
std::vector<vector <cv::Point> > contornos;
std::vector<cv::Vec4i> hierarchy;
cv::Mat imageErode;

// 
int S1,H1,V1;
int SL1=256;//Valor mínimo
int SU1=0;//Valor máximo
int HL1=256;
int HU1=0;
int VL1=256;
int VU1=0;
int contour_index1;

cv::Mat mask1;
cv::Mat img_rgb1,todo1;
cv::Mat img_hsv1;
cv::Mat contours1;
cv::Mat gray_image1;
std::vector<vector <cv::Point> > contornos1;
std::vector<cv::Vec4i> hierarchy1;
cv::Mat imageErode1;

void onMouse( int event, int x, int y, int, void* param ){

    cv::Mat* hsv = (cv::Mat*)param;

    if(event == CV_EVENT_LBUTTONDOWN){
        printf("%d %d %d\n",
        (int)(*hsv).at<cv::Vec3b>(y, x)[0],//color azul
        (int)(*hsv).at<cv::Vec3b>(y, x)[1],//color verde
        (int)(*hsv).at<cv::Vec3b>(y, x)[2]);//color rojo

        H=(int)(*hsv).at<cv::Vec3b>(y, x)[0];
        S=(int)(*hsv).at<cv::Vec3b>(y, x)[1];
        V=(int)(*hsv).at<cv::Vec3b>(y, x)[2];

        if(H > HU) {

```

```

        HU=H;
    }
    if(H < HL){
        HL=H;
    }

    if(S > SU) {
        SU=S;
    }
    if(S < SL){
        SL=S;
    }

    if(V > VU){
        VU=V;
    }
    if(V < VL){
        VL=V;
    }
}

////

if(event == CV_EVENT_RBUTTONDOWN){
    printf("%d %d %d\n",
        (int)(*hsv).at<cv::Vec3b>(y, x)[0],//color azul
        (int)(*hsv).at<cv::Vec3b>(y, x)[1],//color verde
        (int)(*hsv).at<cv::Vec3b>(y, x)[2]);//color rojo

    H1=(int)(*hsv).at<cv::Vec3b>(y, x)[0];
    S1=(int)(*hsv).at<cv::Vec3b>(y, x)[1];
    V1=(int)(*hsv).at<cv::Vec3b>(y, x)[2];

    if(H1 > HU1) {
        HU1=H1;
    }
    if(H1 < HL1){
        HL1=H1;
    }

    if(S1 > SU1) {

```

```

        SU1=S1;
    }
    if(S1 < SL1){
        SL1=S1;
    }

    if(V1 > VU1){
        VU1=V1;
    }
    if(V1 < VL1){
        VL1=V1;
    }

}
}

void callbackImage(const sensor_msgs::ImageConstPtr &msg)
{
    cv_bridge::CvImagePtr cv_ptr;
    cv_ptr= cv_bridge::toCvCopy(msg,sensor_msgs::image_encodings::BGR8);
    cv::Mat frame = cv_ptr->image;
    //cv::Mat grayImage;
    //cv::cvtColor(frame, grayImage, cv::COLOR_BGR2GRAY);
    cv::resize(frame,frame, cv::Size(320, 240));
    cvtColor(frame,img_hsv,CV_BGR2HSV);
    // cv::imshow("hsv",img_hsv);
    cv::imshow("Frame",frame);
    cv::setMouseCallback("Frame", onMouse, &img_hsv); //



    cv::inRange(img_hsv,cv::Scalar(HL,SL,VL),cv::Scalar(HU,SU,VU),mask);
    cv::inRange(img_hsv,cv::Scalar(HL1,SL1,VL1),cv::Scalar(HU1,SU1,VU1),mask1);
    // cv::imshow("Mask",mask);

    int dilatationSize = 4;
    cv::Mat element1 =
    getStructuringElement(cv::MORPH_ELLIPSE, cv::Size(2*dilatationSize+1,2*dilatationSize+1),cv::Point(dilatationSize,dilatationSize));
    cv::Mat imageDilatation;
    cv::Mat imageDilatation1;
    dilate(mask,imageDilatation,element1);
}

```

```

dilate(mask1,imageDilatation1,element1);

cv::namedWindow("Erosion");
cv::namedWindow("Erosion1");
// imshow("Erosion",imageDilatation);

int erosionSize = 6;
cv::Mat element =
getStructuringElement(cv::MORPH_ELLIPSE, cv::Size(2*erosionSize+1,2*erosionSize+1
),cv::Point(erosionSize,erosionSize));
erode(imageDilatation,imageErode,element);
erode(imageDilatation1,imageErode1,element);

//cv::namedWindow("Erosion");
imshow("Erosion",imageErode);
imshow("Erosion1",imageErode1);

cv::Canny(imageErode,contours,100,20);//le pasamos la imagen en HSV y en la imagen
contours aplicamos los contornos
cv::Canny(imageErode1,contours1,100,20);

cv::findContours(contours,contornos,hierarchy, cv::RETR_TREE, cv::CHAIN_APPROX
_SIMPLE, cv::Point(0,0));

cv::findContours(contours1,contornos1,hierarchy, cv::RETR_TREE, cv::CHAIN_APPROX
_SIMPLE, cv::Point(0,0));

cv::drawContours(frame,contornos,contour_index, cv::Scalar(255,0,0),2,5,hierarchy,0, cv::
Point(0,0)); //dibuja en la imagen original

cv::drawContours(frame,contornos1,contour_index1, cv::Scalar(255,0,0),2,5,hierarchy1,0,c
v::Point(0,0));

float sumx=0, sumy=0;
float num_pixel = 0;
for(int x=0; x<imageErode.cols; x++) {
    for(int y=0; y<imageErode.rows; y++) {
        int val = imageErode.at<uchar>(y,x);
        if( val >= 50) {

```

```

        sumx += x;
        sumy += y;
        num_pixel++;
    }
}

float sumx1=0, sumy1=0;
float num_pixel1 = 0;
for(int x1=0; x1<imageErode1.cols; x1++) {
    for(int y1=0; y1<imageErode1.rows; y1++) {
        int val1 = imageErode1.at<uchar>(y1,x1);
        if( val1 >= 50) {
            sumx1 += x1;
            sumy1 += y1;
            num_pixel1++;
        }
    }
}
cv::Point p(sumx/num_pixel, sumy/num_pixel);
cv::Point p3(sumx1/num_pixel1, sumy1/num_pixel1);

cv::Moments m = moments(imageErode, false);
cv::Moments m1 = moments(imageErode1, false);

cv::Point p1(m.m10/m.m00, m.m01/m.m00);
cv::Point p2(m1.m10/m1.m00, m1.m01/m1.m00);

cv::circle(frame, p, 5, cv::Scalar(128,0,0), -1);
cv::circle(frame, p3, 5, cv::Scalar(128,0,0), -1);

///////////
std_msgs::Int16MultiArray centroid_msg;
centroid_msg.data.resize(2);
centroid_msg.data[0]=sumx/num_pixel;
centroid_msg.data[1]=sumy/num_pixel;
pubpunto.publish(centroid_msg);

std_msgs::Int16 puntox_msg;
puntox_msg.data=sumx/num_pixel;

```

```

pubpuntox.publish(puntox_msg);

std_msgs::Int16 puntoy_msg;
puntoy_msg.data=sumy/num_pixel;
pubpuntoy.publish(puntoy_msg);

////////

std_msgs::Int16MultiArray centroid1_msg;
centroid1_msg.data.resize(2);
centroid1_msg.data[0]=sumx1/num_pixel1;
centroid1_msg.data[1]=sumy1/num_pixel1;
pubpunto1.publish(centroid1_msg);

std_msgs::Int16 puntox1_msg;
puntox1_msg.data=sumx1/num_pixel1;
pubpuntox1.publish(puntox1_msg);

std_msgs::Int16 puntoy1_msg;
puntoy1_msg.data=sumy1/num_pixel1;
pubpuntoy1.publish(puntoy1_msg);

cout<<"\n El centroide es ";
cout<<sumx/num_pixel;
cout<<endl;
//cv::imshow("Cany",contours);
cv::imshow("Frame",frame);

cv::waitKey(1);

}

int main(int argc, char **argv)
{
cout<<"inicializando webcam_man_listener node .. "<<endl;
ros::init(argc, argv, "webcam_man_listener");

```

```
ros::NodeHandle n;
ros::NodeHandle nh;

ros::Subscriber subImage = n.subscribe("/webcam_image",1,callbackImage);
pubpunto = n.advertise<std_msgs::Int16MultiArray>("/Centroid",1000);
pubpunto1 = n.advertise<std_msgs::Int16MultiArray>("/Centroid1",1000);
pubpuntox = n.advertise<std_msgs::Int16>("/Puntox",1000);
pubpuntoy = n.advertise<std_msgs::Int16>("/Puntoy",1000);
pubpuntox1 = n.advertise<std_msgs::Int16>("/Puntox1",1000);
pubpuntoy1 = n.advertise<std_msgs::Int16>("/Puntoy1",1000);

pubSpeedLeft=nh.advertise<std_msgs::Float32>("/speed_motor_1",1);

ros::Rate loop (50);

while(ros::ok())
{
    ros::spinOnce();
    loop.sleep();
}

return 0;
}
```

8.- PROGRAMA QUE PUBLICA LA IMAGEN DE LA CÁMARA

```
#include <ros/ros.h>
#include <opencv2/opencv.hpp>
#include <sensor_msgs/Image.h>
#include <cv_bridge/cv_bridge.h>
#include <webcam_man/getImage.h>
#include <image_transport/image_transport.h>
#include <std_msgs/Float32.h>

using namespace std;
ros::Publisher pubSpeedLeft;

cv::Mat frame;

bool callbackImage(webcam_man::getImage::Request &req,
webcam_man::getImage::Response &res)
{
    sensor_msgs::ImagePtr msg = cv_bridge::CvImage(std_msgs::Header(), "bgr8",
frame).toImageMsg();
    res.imageSrv = *msg;

    return true;
}

int main(int argc, char **argv)
{
    cout<<"Starting webcam_man_server by Luis Näva..."<<endl;
    ros::init(argc, argv, "webcam_man_server");
    ros::NodeHandle nh;

    ros::ServiceServer servImage = nh.advertiseService("/webcam_image",
callbackImage);
    //ros::Publisher pubImage = nh.advertise<sensor_msgs::Image>("/webcam_image",
1);
    image_transport::ImageTransport it(nh);
    image_transport::Publisher pubImage = it.advertise("/webcam_image",1);
    ros::Rate loop(50);

    cv::VideoCapture capture;
    capture.open(0);
```

```
pubSpeedLeft=nh.advertise<std_msgs::Float32>("/speed_motor_1",1);

while( ros::ok())
{
    capture.read(frame);
    cv::resize(frame,frame, cv::Size(320, 240));
    sensor_msgs::ImagePtr msg = cv_bridge::CvImage(std_msgs::Header(), "bgr8",
frame).toImageMsg();
    pubImage.publish(msg);
    // cv::imshow("Hola",frame);

    ros::spinOnce();
    loop.sleep();
    //cv::waitKey(1);
}

return 0;
}
```

9.- PROGRAMA PARA SEGUIR 2 COLORES Y EVADIR OBSTÁCULOS

```
#include <ros/ros.h>
#include <std_msgs/Int16MultiArray.h>
#include <std_msgs/Float32.h>
#include <std_msgs/Int16.h>

using namespace std;
int valorx;//Punto x del centroide del color 1
int valory;//Punto y del centroide del color 1
int valorx1;// Punto x del centroide del color 2
int valory1;// Punto y del centroide del color 2
int sharp1;// Valor del sensor de distancia izquierdo
int sharp2;// Valor del sensor de distancia derecho
int sharp3;// Valor del sensor de distancia del centro
int cambio=0;//bandera que cambia el color a seguir

ros::Publisher pubSpeedLeft; //Publica la velocidad al motor Izquierdo
ros::Publisher pubSpeedRight; // Publica la velocidad del motor Decho

void adelante() //Funcion para que el robot se mueva adelante
{
    std_msgs::Float32 speedleftmsg;
    speedleftmsg.data=0.5;
    pubSpeedLeft.publish(speedleftmsg);

    std_msgs::Float32 speedrightmsg;
    speedrightmsg.data=1;
    pubSpeedRight.publish(speedrightmsg);
}

void atras()//Funcion para que el robot se mueve atras
{
    std_msgs::Float32 speedleftmsg;
    speedleftmsg.data=-0.5;
    pubSpeedLeft.publish(speedleftmsg);
```

```
    std_msgs::Float32 speedrightmsg;
    speedrightmsg.data=-1;
    pubSpeedRight.publish(speedrightmsg);
}

void paro()// Funcion que para el robot
{
    std_msgs::Float32 speedleftmsg;
    speedleftmsg.data=0;
    pubSpeedLeft.publish(speedleftmsg);

    std_msgs::Float32 speedrightmsg;
    speedrightmsg.data=0;
    pubSpeedRight.publish(speedrightmsg);
}

void derecha()// Funcion para que el robot valla a la derecha
{
    std_msgs::Float32 speedleftmsg;
    speedleftmsg.data=-0.125;
    pubSpeedLeft.publish(speedleftmsg);

    std_msgs::Float32 speedrightmsg;
    speedrightmsg.data=0.25;
    pubSpeedRight.publish(speedrightmsg);

}

void izquierda()//Funcion para que el robot valla a la izquierda
{
    std_msgs::Float32 speedleftmsg;
    speedleftmsg.data=0.125;
    pubSpeedLeft.publish(speedleftmsg);

    std_msgs::Float32 speedrightmsg;
    speedrightmsg.data=-0.25;
    pubSpeedRight.publish(speedrightmsg);

}
```

```
void sharp1Callback(const std_msgs::Int16::ConstPtr& msg)//obtenemos el valor del sensor de distancia izquierdo y lo imprimimos
```

```
{  
    sharp1=msg->data;  
    cout<<sharp1<<endl;  
}
```

```
void sharp2Callback(const std_msgs::Int16::ConstPtr& msg)//obtenemos el valor del censor de distancia derecho y lo imprimimos
```

```
{  
    sharp2=msg->data;  
    cout<<sharp2<<endl;  
}
```

```
void sharp3Callback(const std_msgs::Int16::ConstPtr& msg)//obtenemos el valor del sensor de distancia central y lo imprimimos
```

```
{  
    sharp3=msg->data;  
    cout<<sharp3<<endl;  
}
```

```
void centroidCallback(const std_msgs::Int16MultiArray::ConstPtr& msg)//obtenemos el valor del centroide 1 y vamos al objeto 1
```

```
{  
    valorx=msg->data[0];  
    valory=msg->data[1];  
    cout<<"El centroide del color 1 en x es: "<<valorx<<" en y es "  
<<valory<<"\n"<<endl;//imprime el centroide del color 1
```

```
if(sharp3>=400 && sharp1<=200 && sharp2<=200) //si el obstaculo esta enfrente el robot ira hacia atras
```

```
{  
    atras();  
}
```

```
else if( sharp3<=400 && sharp1>=200 && sharp2<=200)//si el obstaculo esta a la izquierda el robot gira a la izquierda
```

```
{  
    derecha();
```

```

    }

    else if (sharp3<=400 && sharp1<=200 && sharp2>=200 &&cambio==0)//si el
    obstaculo esta a la derecha gira a la izquierda
    {
        izquierda();
    }

    else if(sharp3<=400 && sharp1<=200 && sharp2<=200 &&cambio==0) //si
    no hay obstaculo
    {
        if(valorx==0)//si no ves el objeto gira a la derecha
        {
            derecha();
        }

        else if(valorx>=0 && valorx<=106)//si el objeto esta a la izquierda gira
        a la izquierda para centrarte
        {
            izquierda();
        }

        else if(valorx>=107 && valorx<=212)//si el centroide esta en medio ve
        adelante
        {
            adelante();
        }

        else if(valorx>=213 && valorx<=320)//si el objeto esta a la derecha
        gira a la derecha para centrarte
        {
            derecha();
        }

    }

    if(valory>=175)//Esta y se tiene que calibrar dependiendo la altura a la que se
    encuentra el color
    {
        cambio=1;//Dispara la bandera para que siga el siguiente color
    }

```

```

        cout<<"LLegó al primer color" << endl;
    }
}

void centroid1Callback(const std_msgs::Int16MultiArray::ConstPtr& msg)//obtenemos
el valor del centroide 2
{
    valorx1=msg->data[0];
    valory1=msg->data[1];
    cout<<"El centroide del color 2 en x es: " << valorx1 << " en y es: "
<< valory1 << "\n" << endl;
    if(cambio==1)
    {
        if(sharp3>=400 && sharp1<=200 && sharp2<=200)
        {
            atras();
        }

        else if( sharp3<=400 && sharp1>=200 && sharp2<=200)
        {
            derecha();
        }

        else if (sharp3<=400 && sharp1<=200 && sharp2>=200)
        {
            izquierda();
        }

        else if(sharp3<=400 && sharp1<=200 && sharp2<=200 && cambio==1)
        {
            if(valorx1==0)
            {

                derecha();

            }

            else if(valorx1>0 && valorx1<=106)
            {

```

```

        izquierda();
    }

    else if(valorx1>=107 && valorx1<=212)
    {
        adelante();
    }

    else if(valorx1>=213 && valorx1<=320)
    {
        derecha();
    }
}

if(valory1>=110)//Esta y se tiene que calibrar dependiendo la altura a la que se
encuentra el color
{
    cambio=2;//Dispara la bandera para que siga el siguiente color
    cout<<"LLego al segundo color"<<endl;
}
}

int main(int argc,char **argv)
{
    cout<<"Comportamiento R2-D2 Seguidor de color"<<endl;
    ros::init(argc,argv,"Colormove");
    ros::NodeHandle nh;

    ros::Subscriber subSharp1=nh.subscribe("/sharpSensor_1",1000,sharp1Callback);
    ros::Subscriber subSharp2=nh.subscribe("/sharpSensor_2",1000,sharp2Callback);
    ros::Subscriber subSharp3=nh.subscribe("/sharpSensor_3",1000,sharp3Callback);
    ros::Subscriber subcentroid=nh.subscribe("/Centroid",1000,centroidCallback);
    ros::Subscriber subcentroid1=nh.subscribe("/Centroid1",1000,centroid1Callback);
    pubSpeedLeft=nh.advertise<std_msgs::Float32>("/speed_motor_1",1);
    pubSpeedRight=nh.advertise<std_msgs::Float32>("/speed_motor_2",1);
}

```

```
while(ros::ok())
{
    ros::spinOnce();
}

return 0;
}
```