

# V2X Laboratory Lesson

## Corso di Tecnologie di Infrastrutture di Reti

**Carlo Augusto Grazia**

*Assistant Professor*

Department of Engineering *Enzo Ferrari*  
University of Modena and Reggio Emilia



Modena, 16th May 2019

# How to create an 802.11p Network

## Arduino-Yun based Laboratory

# Laboratory: Arduino-Yun based 802.11p Network



Atheros AR9331 processor, running Linux and the OpenWrt wireless stack



## If you are interested

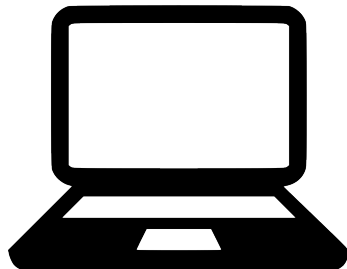
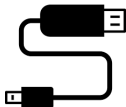
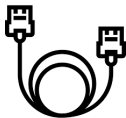
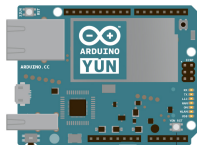
I've created a "tutorial for dummies" to configure a Yun with the right Kernel and be able to use 11p and other features (otherwise hard to deploy on a Yun)

# Requirements

What we need for start:

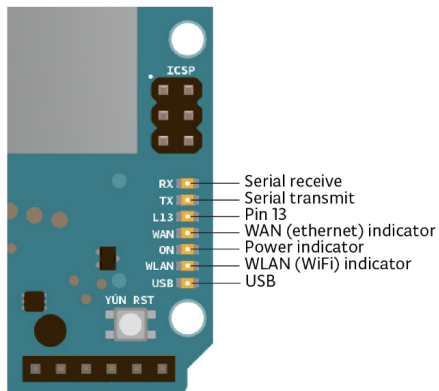
- If you don't want to "use your hands", you can simply follow as a standard lesson
  - Requirements: your own brain
- If you want to participate (forming groups of 2-3 students per Yun), each group needs:
  - One laptop
  - One Ethernet socket
  - One USB port
  - OS: Linux/OSX
  - Windows is allowed (putty instead of ssh) but you'll lose my support
- We only need to connect via ssh to the Yun and we then can start

# Setup



Yun <-> Eth ————— Eth <-> PC  
Yun <-> MicroUSB ————— USB <-> PC

# Powering the Yun



Once connected to the USB port the “power indicator” LED will switch on



# Connecting to the Yun



The number in the Open Source Heart is YOUR ID, we will call it **X** during the lesson.

Remember your **X**, we will use it a lot.

10 Yun for you:  $\mathbf{X} \in [0, 9]$

# Connecting to the Yun

## Critical Part

Create the wired Ethernet connection

## Yun side

Nothing to do, the Yun once booted creates its own wired interface with the IPv4 192.168.X.1

## PC side

Modify your Ethernet NIC assigning a manual IPv4 192.168.X.2 with NetMask 255.255.255.0

All the OS have the “Network Setting” possibility

# Connecting to the Yun: Lab Canali PC

In Superadmin role, click on network icon

- Impostazioni di rete
- Modifica opzioni scheda
- Ethernet n
- Proprieta'
- Protocollo [...] TCP/IPv4
- Proprieta'

Modify IP and NetMask fields

manual IPv4 192.168.X.2 with NetMask 255.255.255.0

# Connecting to the Yun: Verifying

Open a Terminal (console, bash, shell, ...)

```
$ ping 192.168.X.1
```

What do you see?

Only if you have the reply we can move forward

# Connecting to the Yun: Finalizing

```
$ ssh root@192.168.X.1
```

Password: arduino      ...(not always asked)

You are now inside!

# Connecting to the Yun (Lab Canali): Finalizing

Open PUTTY application

```
ip: 192.168.X.1  
username: root  
port: 22
```

Password: arduino      ...(not always asked)

You are now inside!

# On the Yun: Some commands

## ip & iw

- **ip**: command that replaces ifconfig on new kernels: can manage links, routing table, assign ip, enable/disable interfaces
- **iw**: command that manages the wireless interface

Try some commands by yourself

```
$ ip link  
[output]  
$  
$ iw list  
[output]
```

# On the Yun: Some commands

## ip & iw

- **ip**: command that replaces ifconfig on new kernels: can manage links, routing table, assign ip, enable/disable interfaces
- **iw**: command that manages the wireless interface

Try some commands by yourself

```
$ iw dev wlan0 info
```

```
[output]
```

```
$
```

```
$ iwinfo wlan0 frequency
```

```
[output]
```



# On the Yun: Some commands

iw

Tells us if the OCB mode is available

```
$ iw list
```

```
...
```

Supported interface modes:

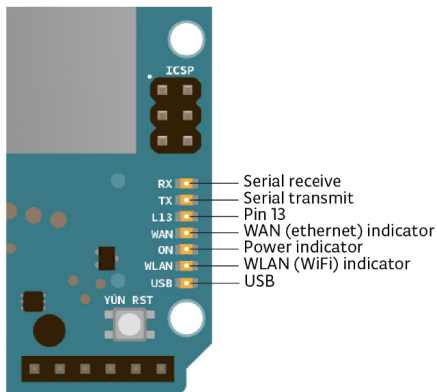
- \* IBSS
- \* managed
- \* AP
- \* AP/VLAN
- \* monitor
- \* mesh point
- \* P2P-client
- \* P2P-GO
- \* outside context of a BSS

```
...
```

# On the Yun: wlan0

ip

A little game to play with the WLAN LED



```
$ ip link set dev wlan0 up
```

```
% WLAN LED up %
```

```
$ ip link set dev wlan0 up
```

```
% WLAN LED down %
```

# On the Yun: 802.11p Creation

`ip & iw`

**ip** to enable/disable wlan0 and **iw** to configure it

Try the commands by yourself

```
$ ip link set dev wlan0 down  
$ iw dev wlan0 set type ocb  
$ ip link set dev wlan0 up  
$ iw dev wlan0 ocb join 2462 10mhz
```

## On the Yun: Actually use 802.11p

`ip`  
**ip** to assign the address and route the traffic

Try the commands by yourself

```
$ ip addr add 192.168.100.1X/24 dev wlan0  
$ route add default gw 192.168.100.1X wlan0
```

Please, remember to change **X** with your Yun number

## On the Yun: Ping an 802.11p colleague

ping

**ping** command does not need any description at all

Try to ping another device **Y**

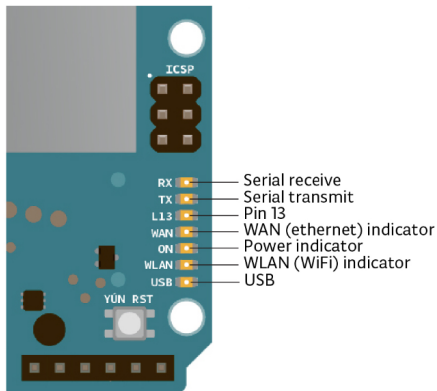
```
$ ping 192.168.100.1Y
```

Please, remember to change **Y** with someone-else Yun number

# On the Yun: Ping an 802.11p colleague

ping

**ping** command does not need any description at all



```
$ ping 192.168.100.1Y
```

While pinging, the WLAN LED flashes (also the receiver one!)

## On the Yun: Detect an 802.11p ping

tcpdump

**tcpdump** is a powerful command-line packet analyzer

Try to capture ping packets (better for the receivers)

```
$ tcpdump -n -i wlan0
```

# On the Yun: 802.11p bandwidth

iperf

**iperf** is the perfect tool to create TCP/UDP client/server applications

Start with 1 volunteer **X** that would be the server and one volunteer **Y** that would be the client

**SERVER: X**

```
$ iperf -s
```

**CLIENT: Y**

```
$ iperf -c 192.168.100.1X -t 3
```



## On the Yun: 802.11p shared bandwidth

Continuing with 1 volunteer **X** that would be the server and more volunteers **Y**, **Z**, ... that would be the clients

**SERVER: X**

```
$ iperf -s
```

**CLIENT: Y**

```
$ iperf -c 192.168.100.1X -t 10
```

**CLIENT: Z**

```
$ iperf -c 192.168.100.1X -t 10
```

Please, sync in order to actually share the spectrum

# On the Yun: 802.11p message passing

ncat

**ncat** is a feature-packed networking utility which reads and writes data across networks from the command line

Pairs of 1 volunteer **X** that would be the server and one volunteer **Y** that would be the client

**SERVER: X**

```
$ ncat -l -k 8080
```

**CLIENT: Y**

```
$ ncat 192.168.100.1X 8080
```

You can also simply try

```
$ echo "message" | ncat SERVER_IP 8080
```

## On the Yun: 802.11p **broadcast** message passing

EVERYONE: enable broadcast

```
$ echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

HOW TO RECEIVE MESS

```
$ ncat -k -l -u -p 8080 -sh-exec "cat > /proc/$$/fd/1"
```

SEND IN BROADCAST

```
$ echo "message" | ncat -u 192.168.100.255 8080
```