How to add a WiFi USB dongle

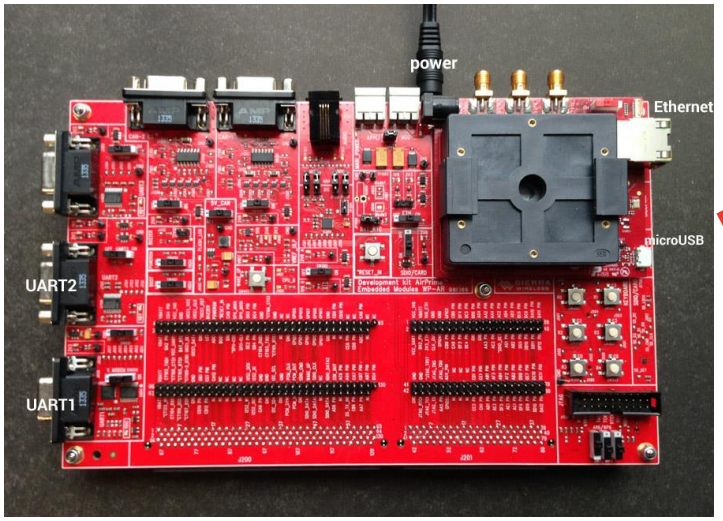
EMEA Application Engineering

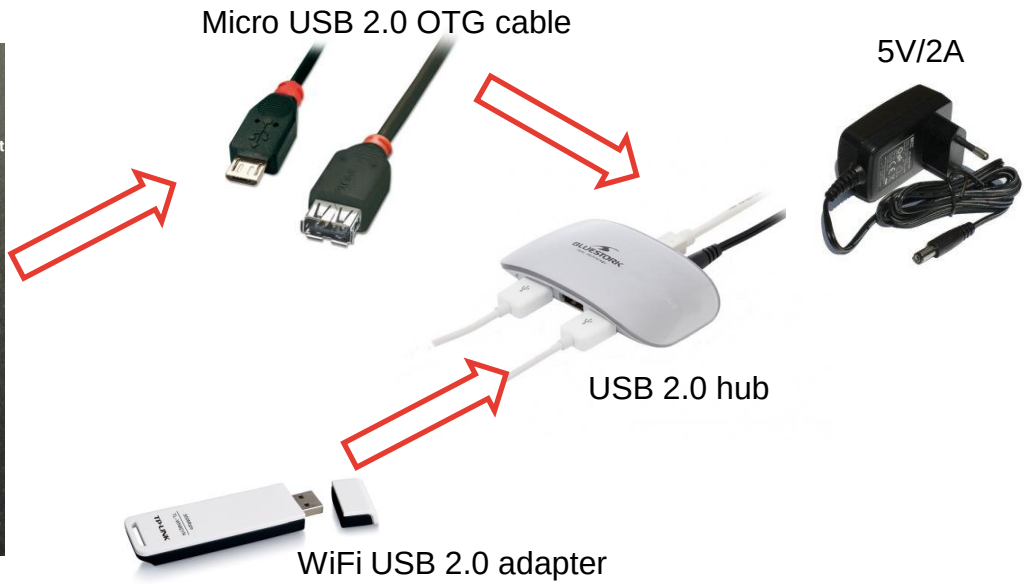
Introduction

- The goal of this tutorial is to explain how to use Yocto in order to add the support of a WiFi USB dongle connected on the USB 2.0 OTG port of a module running the Legato Platform (e.g. WP710x, AR755x, AR8652, WP8548, WP750x).
- On some modules, it may be needed to add external components in order to activate the USB OTG mode (check the PTS).
- As the WiFi driver is dependent of the chipset used in the USB dongle, this tutorial will focus only on one model: **Realtek RTL8192CU**.
  http://www.realtek.com.tw/products/productsView.aspx?Langid=1&PNid=21&PFid=48&Level=5&Conn=4&ProdID=277
- This WiFi 802.11b/g/n chipset is widely used in several devices: https://wikidevi.com/wiki/Rtl8192cu
- Tested devices:
  - TP-LINK TL-WN821N v4
  - TP-LINK TL-WN822N v3

Hardware setup

- Part list:
  - Sierra Wireless Development Kit for WP-AR series
  - Socket-up board with WP or AR module (e.g. WP710x or AR755x)
  - USB 2.0 OTG cable (one is provided with the DevKit)
  - USB 2.0 USB hub, powered by an external supply (5V/2A recommended)
  - WiFi USB 2.0 adapter device
- Hardware connections:





Get the driver from the chipset maker

By default, the mainline Linux kernel offers drivers for various devices, including the Realtek RTL8192CU chipset.

But unfortunately, the "**`rtl8192cu`**" module provided by the mainline kernel is bogus ☹

So, it is recommended to use the Linux driver provided by the manufacturer (Realtek):
http://www.realtek.com.tw/downloads/downloadsView.aspx?Langid=1&PNid=21&PFid=48&Level=5&Conn=4&ProdID=277&DownTypeID=3&GetDown=false&Downloads=true#2772

This allow to build a new kernel module named "**`8192cu`**" (not "`rtl8192cu`").

The latest driver version (at the time of this tutorial) is **4.0.2_9000**.

The filename is: **`RTL8188C_8192C_USB_linux_v4.0.2_9000.20130911.zip`**

Extract the zip file.

The Linux driver itself is located in the "`driver`" folder:
`rtl8188C_8192C_usb_linux_v4.0.2_9000.20130911.tar.gz`

Install Yocto build env on host PC

- System Requirements:

  - Powerful 64-bit CPU (Quadcore HyperThreaded) + lot of RAM (>6GB)
  - Linux distro is Ubuntu 12.04.5 LTS (64-bit)

- Documentations:
  http://www.legato.io/legato-docs/14_10/getstarted_yocto_linux.html
  http://share.sierrawireless.local/syseng/wiki/Legato_Docs/Legato Yocto Linux v001.pdf

- Go on DevZone - Legato Distribution:
  http://developer.sierrawireless.com/Resources/Resources/Legato/Distribution.aspx

- Get the Legato Yocto Linux "Sources package":
  `Legato-Dist-Source-mdm9x15-14.10.tar.bz2`

- Get the "toolchain for Linux PC x86 64bits":
  `Legato-Toolchain-mdm9x15-14.10-x86_64.sh`

- Extract the "Sources package" in a folder of your choice:
  `$ tar jxvf Legato-Dist-Source-mdm9x15-14.10.tar.bz2`

- Install the toolchain (do a "chmod 755" before, if needed):
  `$ ./Legato-Toolchain-mdm9x15-14.10-x86_64.sh`

Check Yocto build env on host PC

- Build Yocto a first time:
  ```
  $ cd LegatoYocto1410
  $ make
  ```
  → Warning: it will take a very long time (couple of hours) to build Yocto the first time.

- Check that Yocto images have been well generated:
  ```
  $ ls -al build_bin/tmp/deploy/images/swi-mdm9x15
  ```
  → You should see the "`kernel`" and "`rootfs`" files.

Create the Yocto recipe for RTL8192CU driver (1/4)

- As this recipe is platform dependent, it needs to be created in the folder "`meta-swi/meta-swi-mdm9x15`":
```
$ cd meta-swi/meta-swi-mdm9x15
$ mkdir -p recipes-connectivity/realtek/files
```
- Copy the Linux driver that has been downloaded previously:
```
$ cp <path_to_the_extracted_zip_file>/driver/
rtl8188C_8192C_usb_linux_v4.0.2_9000.20130911.tar.gz ./recipes-
connectivity/realtek/files/.
```
- In order to compile the driver for the SIERRA_MDM9x15 target, it is needed to patch the `Makefile` to provide arch settings and the paths to the cross toolchain and the target kernel source code.
- Create a patch file call "`rtl8192cu_sierra_mdm9x15.patch`":
```
$ vi recipes-
connectivity/realtek/files/rtl8192cu_sierra_mdm9x15.patch
```

Create the Yocto recipe for RTL8192CU driver (2/4)

- Copy/Paste the following lines and save the file:

```
--- a/Makefile  2014-11-26 18:44:01.459030231 +0100
+++ b/Makefile  2014-11-26 18:45:47.859032433 +0100
@@ -38,7 +38,7 @@
 CONFIG_INTEL_WIDI = n
 CONFIG_WAKE_ON_WLAN = n

-CONFIG_PLATFORM_I386_PC = y
+CONFIG_PLATFORM_I386_PC = n
 CONFIG_PLATFORM_TI_AM3517 = n
 CONFIG_PLATFORM_ANDROID_X86 = n
 CONFIG_PLATFORM_JB_X86 = n
@@ -70,6 +70,7 @@
 CONFIG_PLATFORM_MSTAR_A3 = n
 CONFIG_PLATFORM_ARM_SUNxI = n
 CONFIG_PLATFORM_ARM_SUN6I = n
+CONFIG_PLATFORM_ARM_SIERRA_MDM9X15 = y

 CONFIG_DRVEXT_MODULE = n

@@ -524,6 +525,15 @@
 #KSRC:= ../lichee/linux-3.3/
 endif

+ifeq ($(CONFIG_PLATFORM_ARM_SIERRA_MDM9X15), y)
+EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN -DCONFIG_ARM -DCONFIG_SIERRA -DCONFIG_IOCTL_CFG80211
+ARCH := arm
+CROSS_COMPILE := /opt/swi/y16-ext/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-
+KSRC := /home/your_name/data/LegatoYocto1410/build_bin/tmp/sysroots/swi-mdm9x15/usr/src/kernel
+KVER := 3.4.91
+endif
+
+
 ifneq ($(USER_MODULE_NAME),)
 MODULE_NAME := $(USER_MODULE_NAME)
 endif
```

Create the Yocto recipe for RTL8192CU driver (3/4)

- In order to improve the driver performances, some debug traces can be removed by creating the following patch:
  `$ vi recipes-connectivity/realtek/files/r8192cu_remove_debug.patch`

- Copy/Paste the following lines and save the file:

```
diff --git a/include/autoconf.h b/include/autoconf.h
index 12294df..f62231b 100755
--- a/include/autoconf.h
+++ b/include/autoconf.h
@@ -296,9 +296,9 @@
 //#define CONFIG_DEBUG_RTL871X

 #define DBG    0
-#define CONFIG_DEBUG_RTL819X
+//#define CONFIG_DEBUG_RTL819X

-#define CONFIG_PROC_DEBUG      1
+//#define CONFIG_PROC_DEBUG    1

 //#define DBG_IO
 //#define DBG_DELAY_OS
```

Create the Yocto recipe for RTL8192CU driver (4/4)

- Now create the recipe file:
  `$ vi recipes-connectivity/realtek/rtl8192cu_4.0.2_9000.bb`

- Copy/Paste the following lines and save the file:

```
SUMMARY = "Driver for Realtek RTL8192CU USB wireless devices"
HOMEPAGE = "http://www.realtek.com.tw/"
LICENSE = "GPLv2"
PR = "r1"
LIC_FILES_CHKSUM = "file://os_dep/linux/os_intfs.c;endline=19;md5=72c75de415f1e8a42587d170459677e2"

SRC_URI = " \
    file://rtl8188C_8192C_usb_linux_v4.0.2_9000.20130911.tar.gz \
    file://r8192cu_remove_debug.patch \
    file://rtl8192cu_sierra_mdm9x15.patch \
    "

S = "${WORKDIR}/rtl8188C_8192C_usb_linux_v4.0.2_9000.20130911/"

inherit module siteinfo

do_install() {
        install -d ${D}/lib/modules/${KERNEL_VERSION}/kernel/drivers/net/wireless
        install -m 0644 ${S}/8192cu.ko
${D}${base_libdir}/modules/${KERNEL_VERSION}/kernel/drivers/net/wireless
}
```

Add cfg80211 kernel module in the Linux recipe

- In order to support WiFi on the target, it is mandatory to use the Linux 802.11 configuration API which is a kernel module called "**cfg80211**". http://wireless.kernel.org/en/developers/Documentation/cfg80211
- To add this module in the kernel recipe:
  - Create a config file:
    ```
    $ echo "CONFIG_CFG80211=m" > recipes-kernel/linux/files/Enable-CFG80211-as-kernel-module.cfg
    ```
  - Edit the kernel recipe file to add this config file:
    ```
    $ vi recipes-kernel/linux/linux-yocto_3.4.bbappend
    ```
  - Add the following line (after "`SRC_URI = "git://${KSRC_linux_yocto_3_4};…`":
    ```
    SRC_URI += "file://Enable-CFG80211-as-kernel-module.cfg"
    ```
  - Save the file

Add the new packages to the image recipe

- Edit the mdm9x15 image include file:
  ```
  $ vi recipes-core/images/mdm9x15-image.inc
  ```
- Add the following lines:
  ```
  IMAGE_INSTALL += "rtl8192cu"
  IMAGE_INSTALL += "linux-firmware-rtl8192cu"
  IMAGE_INSTALL += "kernel-module-cfg80211"
  ```

Install additional WiFi tools

- In order to manage the WiFi connection, it is needed to install additional packages:
  - **Wireless tools**:
    Provides a set of commands (`iwconfig`, `iwlist`, …) useful to configure and manage the WiFi connection.
    http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html
  - **WPA Supplicant**:
    Needed to handle WPA/WPA2/Entreprise authentications (in addition to Open and WEP).
    http://w1.fi/wpa_supplicant/
- Add these two packages into the target image recipe:
  `$ vi recipes-core/images/mdm9x15-image-minimal.bb`
  After the line "`require mdm9x15-image.inc`", put these two lines:

```
IMAGE_INSTALL += "wireless-tools"
IMAGE_INSTALL += "wpa-supplicant"
```

Build and install Yocto images

- Go in the root of the Yocto folder (e.g. `LegatoYocto1410`) and run:
  `$ make`
- If everything went well then you will find the kernel module file `8192cu.ko` in the following folder:
  `build_bin/tmp/work/swi_mdm9x15-poky-linux-gnueabi/rtl8192cu/4.0.2-r1/rtl8188C_8192C_usb_linux_v4.0.2_9000.20130911/`
- Install the newly generated "`kernel`" and "`roofs`" images.
  Example using `adb` and `fastboot` commands:

```
$ cd build_bin/tmp/deploy/images/swi-mdm9x15
$ adb reboot-bootloader
$ fastboot erase kernel
$ fastboot erase rootfs
$ fastboot flash kernel kernel
$ fastboot flash rootfs rootfs
$ fastboot reboot
```

- **Warning**: the resulting **rootfs** yaffs2 image size will be > 50MiB.
  If your **rootfs** flash partition is too small then increase it using the `AT!APPPARTCHG` command (not needed when ubifs will be supported)

Setup the WiFi network interface

- Start the module and log as root
- Connect the powered USB hub to the USB OTG port of the module
- Add the loadable modules to the running kernel:
```
# modprobe cfg80211
# modprobe 8192cu
```
- Connect the WiFi USB dongle to on of the USB hub ports
- Wake-up the wlan0 network interface:
```
# ifconfig wlan0 up
```
- Scan the WiFi network to list the available access points:
```
# iwlist wlan0 scan
```

Setup the WiFi connection

- Select one SSID from this list and create a matching wpa_supplicant configuration file.
- Config example for Sierra Wireless “SWI-MOBILE” access point using WPA2 (WPA-PSK authentication and CCMP/AES encryption):
  `# vi swi_mobile.conf`

```
network={
    ssid="SWI-MOBILE"
    proto=RSN
    key_mgmt=WPA-PSK
    pairwise=CCMP TKIP
    group=CCMP TKIP
    psk=“XXXXXXXX"
}
```

  replace “XXXXXXXX” by the famous password ☺

- Connect to the access point:
  `# wpa_supplicant -B -i wlan0 -c /home/root/swi_mobile.conf`

Test the WiFi connection (1/2)

- The easiest way is to disconnect all other network interfaces (e.g. Ethernet, USB, RmNet, …) in order to keep only the wlan0 interface.
- Check the connection to the access point:
  `# iwconfig wlan0`

```
wlan0     IEEE 802.11abgn  ESSID:"SWI-MOBILE"
          Mode:Managed  Frequency:2.437 GHz  Access Point: 00:1A:1E:FC:42:C3
          Bit Rate=54 Mb/s   Tx-Power=12 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on
          Link Quality=65/70  Signal level=-45 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

- Request an IP address:
  `# udhcpc -i wlan0`

```
udhcpc (v1.22.1) started
Sending discover...
Sending select for 192.168.40.155...
Lease of 192.168.40.155 obtained, lease time 28800
/etc/udhcpc.d/50default: Adding DNS 8.8.8.8
/etc/udhcpc.d/50default: Adding DNS 4.2.2.2
```

Test the WiFi connection (2/2)

- Check that wlan0 interface got a valid IP address:
  `# ifconfig wlan0`

```
wlan0     Link encap:Ethernet  HWaddr 64:66:B3:22:4F:C9
          inet addr:192.168.40.155  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:36 errors:0 dropped:2 overruns:0 frame:0
          TX packets:64 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3642 (3.5 KiB)  TX bytes:15908 (15.5 KiB)
```

- Ping the Internet…
  `# ping www.google.com`

```
PING www.google.com (64.233.167.99): 56 data bytes
64 bytes from 64.233.167.104: seq=2 ttl=45 time=9.401 ms
64 bytes from 64.233.167.104: seq=3 ttl=45 time=8.942 ms
64 bytes from 64.233.167.104: seq=4 ttl=45 time=8.729 ms
64 bytes from 64.233.167.104: seq=5 ttl=45 time=9.705 ms
64 bytes from 64.233.167.104: seq=6 ttl=45 time=9.706 ms
64 bytes from 64.233.167.104: seq=7 ttl=45 time=8.577 ms
--- www.google.com ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 8.577/9.177/9.706 ms
```

- Enjoy!

Automatic WiFi setup at startup (1/2)

- It can be convenient to have an automatic WiFi connection at startup (similar to what it is already done for the Ethernet interface).
- Edit (or create) a file to tell which loadable modules are automatically loaded and add the lines "`cfg80211`" and "`8192cu`":
  `# vi /etc/modules`

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

cfg80211
8192cu
```

- Edit (or create) wpa_supplicant configuration file to match your WiFi connection settings:
  `# vi /etc/wpa_supplicant.conf`

```
network={
    ssid="SWI-MOBILE"
    proto=RSN
    key_mgmt=WPA-PSK
    pairwise=CCMP TKIP
    group=CCMP TKIP
    psk="XXXXXXXX"
}
```

Automatic WiFi setup at startup (2/2)

- Edit the configuration file for network interfaces to add "`auto wlan0`":
  `# vi /etc/network/interfaces`

```
# Wireless interfaces
auto wlan0
iface wlan0 inet dhcp
        wireless_mode managed
        wireless_essid any
        wpa-driver wext
        wpa-conf /etc/wpa_supplicant.conf
```

- Done!
  At next startup the WiFi connection will be automatically activated.

More to do …

- Add packages in order to setup a WiFi hotspot (`hostapd`, …)
- Add other WiFi chipsets (e.g. Atheros/Qualcomm, Broadcom, CSR, …)

Thank You

SIERRA WIRELESS®