Lab 7: AR.Drone Control

EE-379 Embedded Systems and Applications

Electrical Engineering Department, University at Buffalo

Last update: Cristinel Ababei, April 2013

1. Objective

The objective of this (supplemental) lab is to use the MCB1700 board, a WRT54GL wireless router, and a Wii NunChuck to control a Parrot AR.Drone. The overall set-up of this project is shown in the figure below.

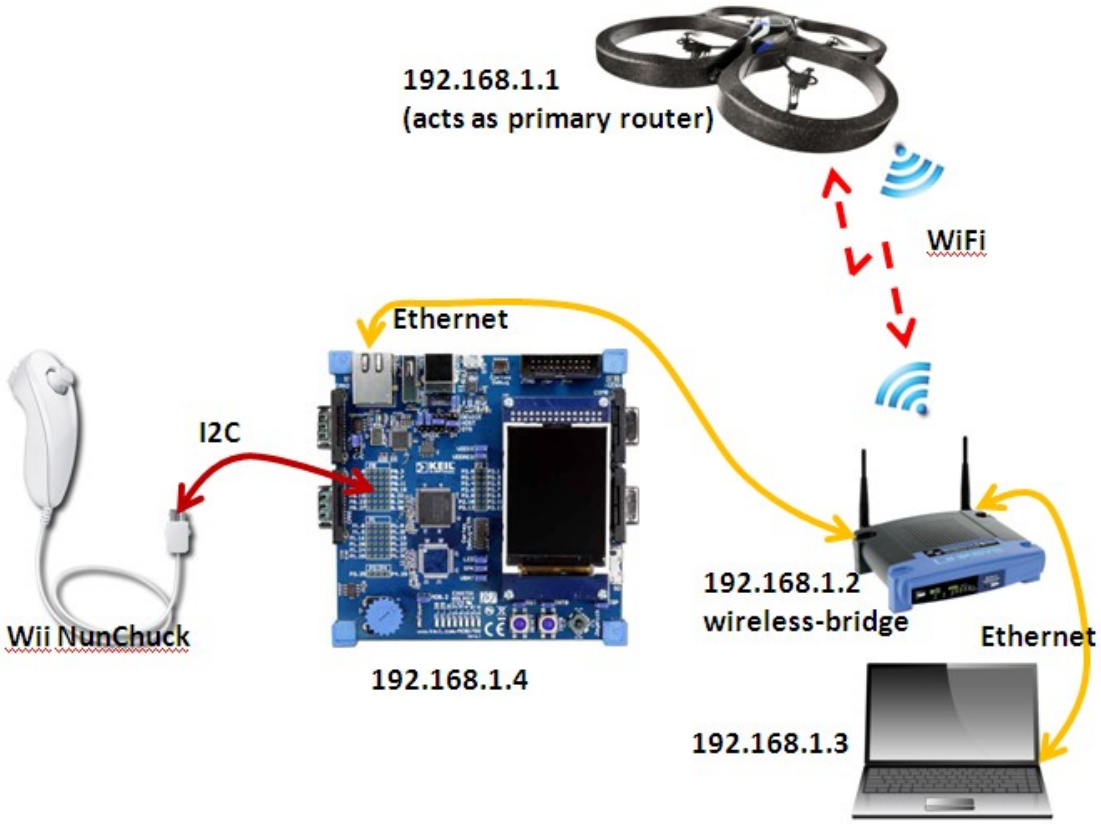

Figure 1 Overview of the project developed in this lab.

2. AR.Drone

AR.Drone is a sophisticated Linux-based WiFi-enabled flying embedded system. It's a neat toy and much more!

The Parrot AR.Drone was revealed at the International CES 2010 in Las Vegas. At the same time, Parrot also demonstrated the iOS applications used to control it. Along with AR.Freeflight, the application designed for free operation of the drone, Parrot also released AR.Race, allowing users to participate in solo games, or interact with other drones in combat simulations.

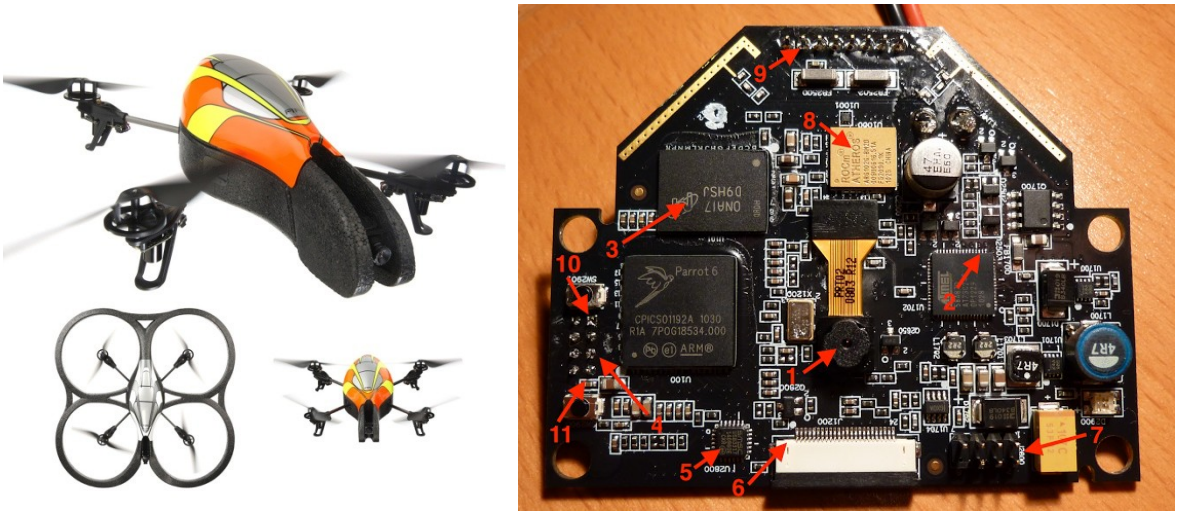

Figure 2 AR.Drone and its main board.

The AR.drone can be piloted by an iPod app, making it usable with any WiFi-enabled iDevice. Android-based apps have emerged too. The drone has two video cameras, one front facing and the other down facing, plus an accelerometer, two gyrometers, and an ultrasonic altimeter. It is all controlled by an ARM-based processor running Linux. Both the iOS-based app and the Linux-based embedded system are hackable. Parrot provides an SDK and encourages developers to use it as an application platform. Cool things can be developed with it – imagine virtual reality.

I controlled my AR.drone with:
1) my iPod,
2) laptop-based application via the laptop's wireless card,
3) same laptop-based application via a Wireless-G router, WRT54GL, which is connected to the laptop via an Ethernet cable; the router is programmed to work as a "wireless bridge"; also, the laptop's own wireless card is shut-off in this case, and
4) MCB1700 board (has an LPC1768 microcontroller) connected to the WRT54GL router (again set-up as a wireless bridge) and a Wii NunChuck.

In this supplemental lab, we'll study the number 4) aforementioned scenario.

Some technical specifications of the AR.Drone include (of the first model):
- CPU: 468MHz ARM9 embedded microcontroller
- RAM: 128MB
- Interfaces: USB and Wi-Fi 802.11b/g
- Front camera: VGA sensor with 93º lens
- Vertical camera: 64º lens, recording up to 60fps

A closer look at the motherboard is shown in Fig.1, which points out some of the main elements:
1: Camera, I2C bus 0 address 5d
2: I2C connectors for I2C Bus 0 arrive here
3: DRAM cache
4: Connector to navigation board, at least one serial port

5: USB driver SMSC USB3317
6: Connector to front camera, probably not an I2C device
7: External connection port with TTYS0 serial port and USB port
8: ROCm Atheros AR6102G-BM2D wireless
9: Power supply for the engines
10: Ground
11: VCC +5V

Read more details about AR.Drone here:
--Parrot AR.Drone official website; http://ardrone2.parrot.com/usa/
--AR.Drone open API platform; https://projects.ardrone.org/
--AR.Drone Developer Guide SDK 1.7; http://www.ardrone-flyers.com/wiki/Version_SDK-1.7
--AR.Drone mainboard overview (Fig.1 right taken from here); http://awesome-drone.blogspot.com/
--Hacking the AR-DRONE Parrot; http://www.libcrack.so/2012/10/13/hacking-the-ar-drone-parrot/
--Chip Overclock's dissection (6 parts, so far);
http://coverclock.blogspot.com/2011/02/deconstructing-ardrone.html
http://coverclock.blogspot.com/2011/02/deconstructing-ardrone-part-2.html
http://coverclock.blogspot.com/search/label/AR.drone
--Shwetak N. Patel, Univ. of Washington; http://abstract.cs.washington.edu/~shwetak/classes/ee472/
--Buy it for about $300 at Amazon; http://www.amazon.com/Parrot-AR-Drone-Quadricopter-Controlled-Android/dp/B007HZLLOK/ref=sr_1_1?ie=UTF8&qid=1367113226&sr=8-1&keywords=ar+drone

3. Linksys Wireless-G WRT54GL router

This is already a somewhat old router but it's still very popular and with a lot of online support. It can be utilized as an access-point to extend your wireless LAN but also as a wireless-bridge to further extend your network. A nice thing is that it can be utilized as a "wireless card" for a laptop/PC which does not have one; not to mention its Linux compatibility.

The Linksys Wireless-G Broadband Router is really three devices in one box. First, there's the Wireless Access Point, which lets you connect both screaming fast Wireless-G (802.11g at 54Mbps) and Wireless-B (802.11b at 11Mbps) devices to the network. There's also a built-in 4-port full-duplex 10/100 Switch to connect your wired-Ethernet devices together. Connect four PCs directly, or attach more hubs and switches to create as big a network as you need. Finally, the Router function ties it all together and lets your whole network share a high-speed cable or DSL Internet connection.

Figure 3 Wireless-G WRT54GL router.

We'll this router to connect to and control the AR.Drone. To do that, we need to set it to work as a **"wireless bridge"**. But before that, we must update its firmware with Tomato.

The drone itself acts as a router, that's why it can't connect to another router like a wrt54gl. With factory software is can't function as a client of another router, but routers can be reconfigured to relay the drones signal and extend the range. The drone emits a standard wifi signal for clients to connect to, so you have to configure your router to act as the client. With the wrt54gl routers, one can use Tomato's firmware:
http://www.polarcloud.com/tomato
Tomato is a simple replacement firmware for Linksys' WRT54G/GL/GS, Buffalo WHR-G54S/WHR-HP-G54 and other Broadcom-based routers. It features an easy to use GUI, a new bandwidth usage monitor, more advanced QOS and access restrictions, enables new wireless features such as WDS and wireless client modes, raises the limits on maximum connections for P2P, allows you to run your custom scripts or telnet/ssh in and do all sorts of things like re-program the SES/AOSS button, adds wireless site survey to see your wifi neighbors, and more.

So, download the Tomato's firmware and save it on your computer (you will need only WRT54G_WRT54GL.bin).
Then, open the original Linksys GUI in your browser. The default URL is http://192.168.1.1/.
Leave user name as blank and type "admin" as password.
Select Upgrade Firmware and use Tomato's WRT54G_WRT54GL.bin.

After, a few minutes the upgrade will be done.
Next, we do a so called **Hard Reset** (a.k.a. 30/30/30 reset) of the router. The following procedure will clear out the NVRAM and set dd-wrt back to default values:

- With the unit powered on, press and hold the reset button on back of unit for 30 seconds
- Without releasing the reset button, unplug the unit and hold reset for another 30 seconds
- Plug the unit back while STILL holding the reset button a final 30 seconds

Note: *This procedure should be done BEFORE and AFTER every firmware upgrade/downgrade.*
Open the new Tomato GUI in your browser. The default URL is http://192.168.1.1/.
User "root" as user and "admin" as password.
Then, Navigate to Basic–>Network. There are a few settings to toggle in this section.
Within the Network sub-menu do the following settings:

- First, toggle the WAN / Internet to Disabled.
- Second, change the values in the LAN section to the following:
  - Router IP Address: 192.168.1.2 (presumes that the Primary Router IP is 192.168.1.1 – that will be the AR.Drone)
  - Subnet Mask: 255.255.255.0
  - Default Gateway: 192.168.1.1 (the IP of the Primary Router, i.e., the AR.Drone)
  - Static DNS: 191.168.1.1
  - DCHP Server: Unchecked

In the Wireless section of the Network sub-menu, configure the following settings:

- Enable Wireless: Checked.
- Wireless Mode: Wireless Ethernet Bridge
- Wireless Network Mode: Mixed

- SSID: ardrone_291961. This is the name SSID of the network whose Primary Router is the AR.Drone. In my case it's ardrone_291961; you can check what yours is by looking up for wireless networks with your laptop, after you power-up the drone.
- Channel: The channel of the Primary Router, i.e., 6 – 2.437.
- Security: Disabled

After the above changes, your Tomato's GUI should look like in the figure below:

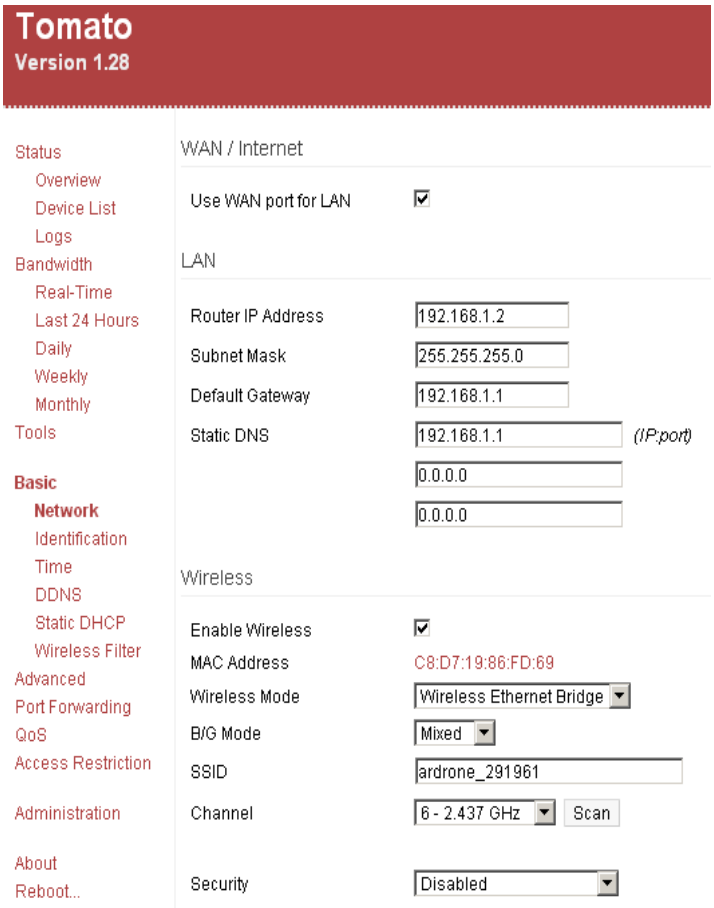

Figure 4 Setting up the router as wireless-bridge.

Note: *For some reason, each time I need to log-in and set-up the router, I need to do the 30/30/30 hard-reset described above, which is becoming annoying. An alternative to Tomato is DD-WRT firmware (http://www.dd-wrt.com/site/index). I have not tried it yet though.*

Read more details about WRT54GL wireless router and more here:
--How To Turn An Old Router Into A Wireless Bridge; http://www.makeuseof.com/tag/how-to-turn-an-old-router-into-a-wireless-bridge/
--How To Extend Your Wi-Fi Network With Simple Access Points; http://www.howtogeek.com/104469/how-to-extend-your-wi-fi-network-with-simple-access-points/
--How To Extend Your Wireless Network with Tomato-Powered Routers; http://www.howtogeek.com/104007/how-to-extend-your-wireless-network-with-tomato-powered-routers/
--Turn Your Old Router into a Range-Boosting Wi-Fi Repeater; http://lifehacker.com/5563196/turn-your-old-router-into-a-range+boosting-wi+fi-repeater

--Buy a used WRT54GL for about $25 here; http://www.amazon.com/Cisco-Linksys-WRT54GL-Wireless-G-Broadband-Router/dp/B000BTL0OA

4. Wii NunChuck

The Wii NunChuk is an input device with a joystick, two buttons, and a three-axis accelerometer as illustrated in the figure below. The three axes X, Y, and Z correspond to the data produced by the accelerometer, joystick. X is right/left, Y is forward/backwards, and Z is up/down (accelerometer only).

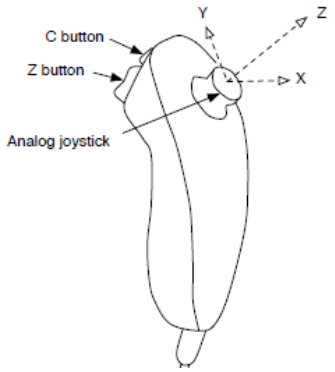

Figure 5 Wii NunChuck.

The NunChuck communicates via I2C. We'll hook the NunChuck to the I2C1 bus of the LPC1768 microcontroller on the MCB1700 board. We'll initialize the Nunchuk to a known state and then regularly "poll" its state.

We've already discussed this in lab#4. Please re-visit lab#4 as we'll utilize much of it to put together the project herein.

5. uIP – A Light Weight TCP/IP Stack

The AR.Drone can be controlled via UDP packets. We'll do that by utilizing a popular TCP/IP stack: micro IP (a.k.a., uIP).

The uIP was an open source TCP/IP stack capable of being used with tiny 8- and 16-bit microcontrollers. It was initially developed by Adam Dunkels (http://dunkels.com/adam/) of the "Networked Embedded Systems" group at the Swedish Institute of Computer Science, licensed under a BSD style license, and further developed by a wide group of developers. uIP can be very useful in embedded systems because it requires very small amounts of code and RAM. It has been ported to several platforms, including DSP platforms.

The uIP implementation is designed to have only the absolute minimal set of features needed for a full TCP/IP stack. It can only handle a single network interface and contains only a rudimentary UDP implementation, but focuses on the IP, ICMP and TCP protocols. uIP is written in the C programming language.

Download and read more details about UIP here:
--uIP downloads; https://github.com/adamdunkels/uip/tags
--uIP (micro IP); http://en.wikipedia.org/wiki/UIP_(micro_IP)
-- The uIP TCP/IP stack; http://www.gaisler.com/doc/net/uip-0.9/doc/html/main.html
--Some documentation; http://ucapps.de/midibox_ng/doxygen/group__uip.html

6. Example 1: Simple Controls of AR.Drone

The files necessary for this example are located in downloadable archive for this lab. This is actually the whole uVision project directory. Just copy it, then clean and re-build the project. Download to the board.

Make all the necessary connections as illustrated in Fig.1.

- Power-up the AR.Drone and place on the floor.
- Connect MCB1700 board to the WRT54GL wireless router using an Ethernet cable.
- Connect the NunChuck as done in lab#4 using the NunChuck adaptor, additional wires, and power supply provided by your instructor. Use the button "C" for take-off and button "Z" to land. Use the joystick to move the drone. Read the source code, which has comments explaining these controls.

Observe operation and comment.

7. TODOs

Lab assignments:
--Change the source code to implement more complex controls.
--Read also the data provided by the AR.Drone and do something about it, e.g., display text and/or video data on the LCD screen of the board, etc.
--Propose something creative and implement it!