

# Virtual development environment

* Base contiki SDK <http://sourceforge.net/projects/contiki/files/Instant%20Contiki/> version 2.7
* Getting started instructions <http://www.contiki-os.org/start.html>

# Hardware

* Buy <http://www.ti.com/tool/cc2538dk>

[Contiki](http://www.contiki-os.org/index.html)

* [Get started](http://www.contiki-os.org/start.html)
* [Hardware](http://www.contiki-os.org/hardware.html)
* [Support](http://www.contiki-os.org/support.html)
* [Download](http://www.contiki-os.org/download.html)
* [Community](http://www.contiki-os.org/community.html)
* [License](http://www.contiki-os.org/license.html)
* [Blog](http://contiki-os.blogspot.com/)
* [Contact](http://www.contiki-os.org/contact.html)

[](https://github.com/contiki-os/contiki)

# Get Started with Contiki

* [Step 1: Grab Instant Contiki](http://www.contiki-os.org/start.html#install-instant-contiki)
* [Step 2: Start Cooja](http://www.contiki-os.org/start.html#start-cooja)
* [Step 3: Run Contiki in simulation](http://www.contiki-os.org/start.html#simulation)
* [Step 4: Run Contiki on hardware](http://www.contiki-os.org/start.html#hardware)
* [Step 5: Now what?](http://www.contiki-os.org/start.html#next)

# Step 1: Grab Instant Contiki

Contiki is a complex piece of software, but don't despair! Instant Contiki and Cooja makes Contiki easy to install and get started with.

### **About Instant Contiki**

Instant Contiki is an entire Contiki development environment in a single download. It is an Ubuntu Linux virtual machine that runs in VMWare player and has Contiki and all the development tools, compilers, and simulators used in Contiki development installed.

Instant Contiki is so convenient that even hardcore Contiki developers use it.

We begin by downloading Instant Contiki, installing VMWare Player, and booting up Instant Contiki.

## **Download Instant Contiki**

Download Instant Contiki. Get a coffee: it is a large file, just over 1 gigabyte. When downloaded, unzip the file, place the unzipped directory on the desktop.

[Download Instant Contiki »](http://sourceforge.net/projects/contiki/files/Instant%20Contiki/)

## **Install VMWare Player**

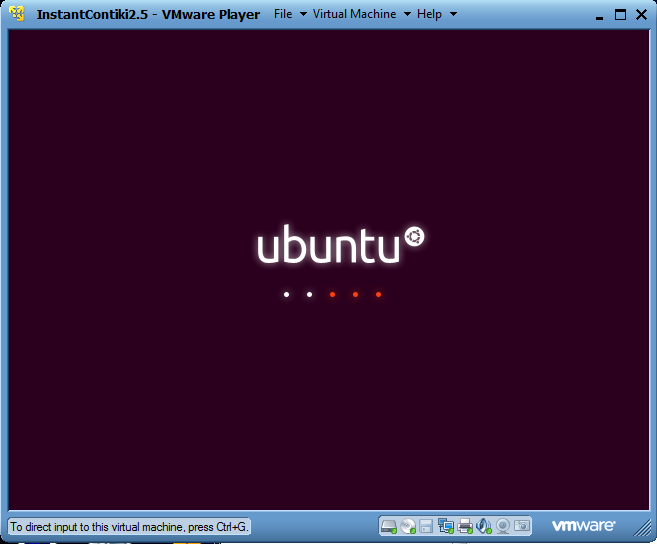
Download and install VMWare Player. It is free to download, but requires a registration. It might require a reboot of your computer, which is unfortunate but needed to get networking working.

[Download VMWare Player »](http://www.vmware.com/go/downloadplayer/)

## **Start Instant Contiki**

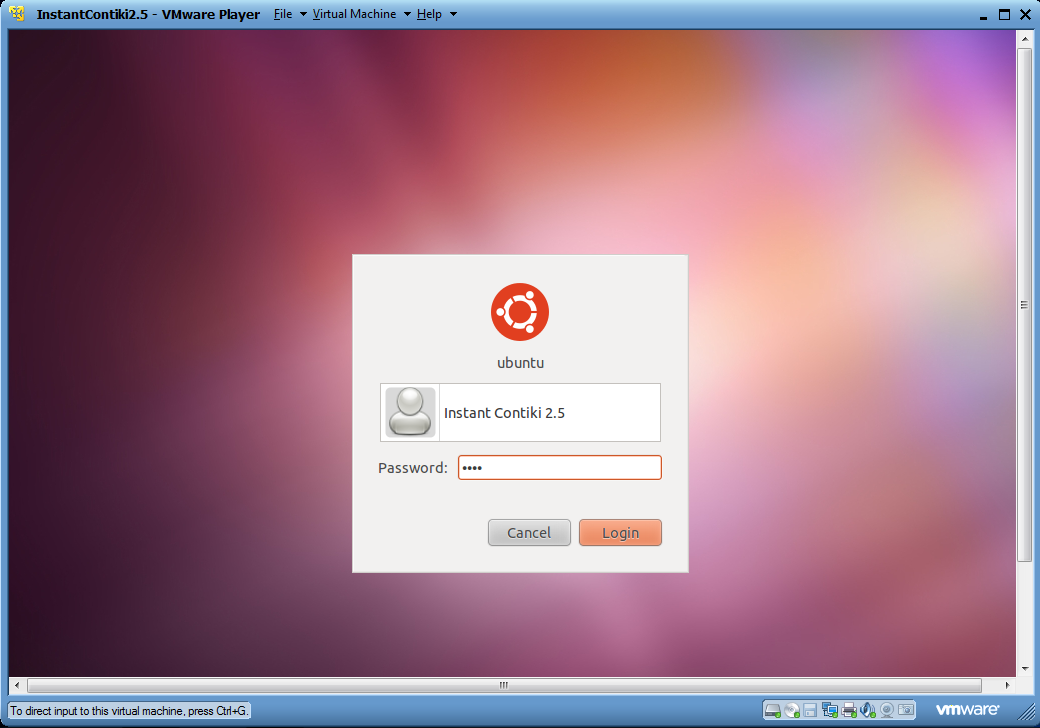
### **Boot Ubuntu**

Start Instant Contiki by running InstantContiki2.6.vmx. Wait for the virtual Ubuntu Linux boot up.



### **Log in**

Log into Instant Contiki. The password is **user**.



Congratulations! Now that we have Instant Contiki up and running, we will start Cooja.

# Step 2: Start Cooja

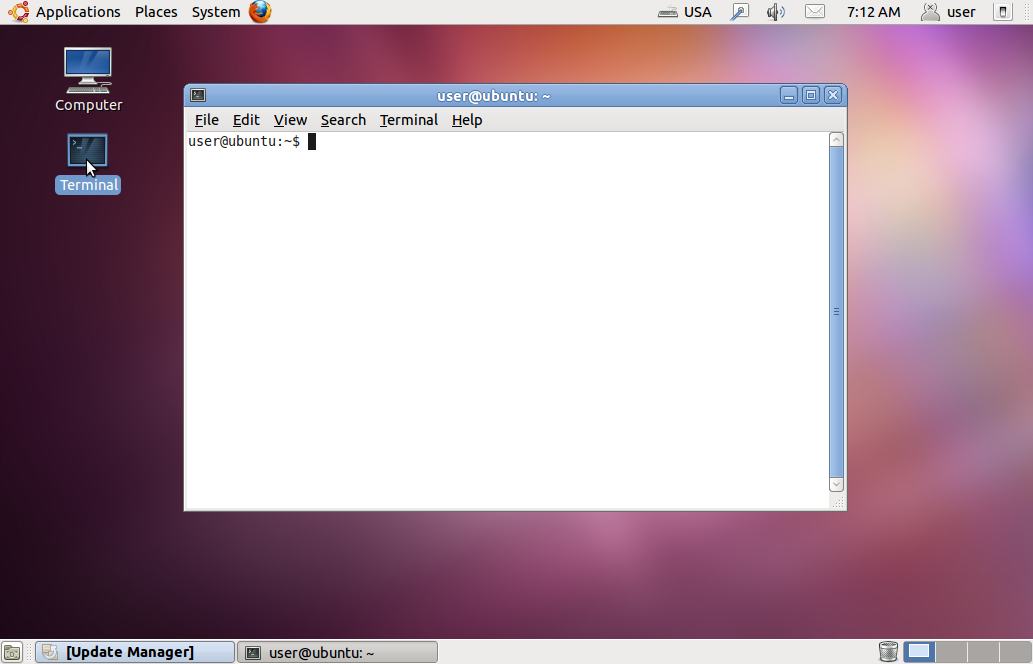
### **About Cooja**

Cooja is the Contiki network simulator. Cooja allows large and small networks of Contiki motes to be simulated. Motes can be emulated at the hardware level, which is slower but allows precise inspection of the system behavior, or at a less detailed level, which is faster and allows simulation of larger networks.

We will now compile and start Cooja, the Contiki network simulator.

## **Open a terminal window**

To start Cooja, first open a terminal window.



## **Start Cooja**

In the terminal window, go to the Cooja directory:

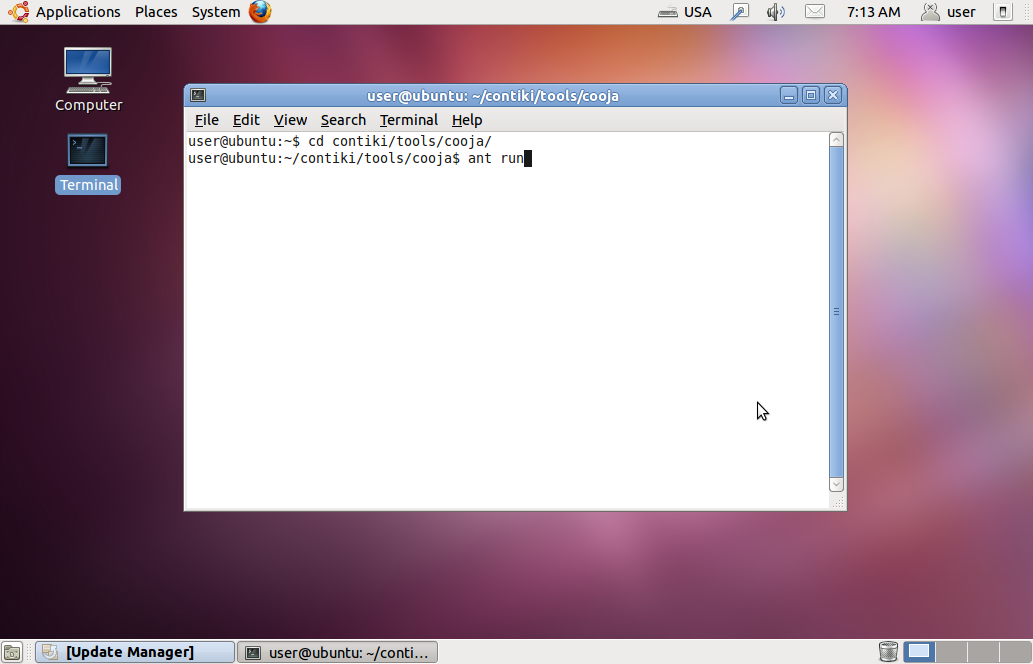
cd contiki/tools/cooja

Start Cooja with the command:

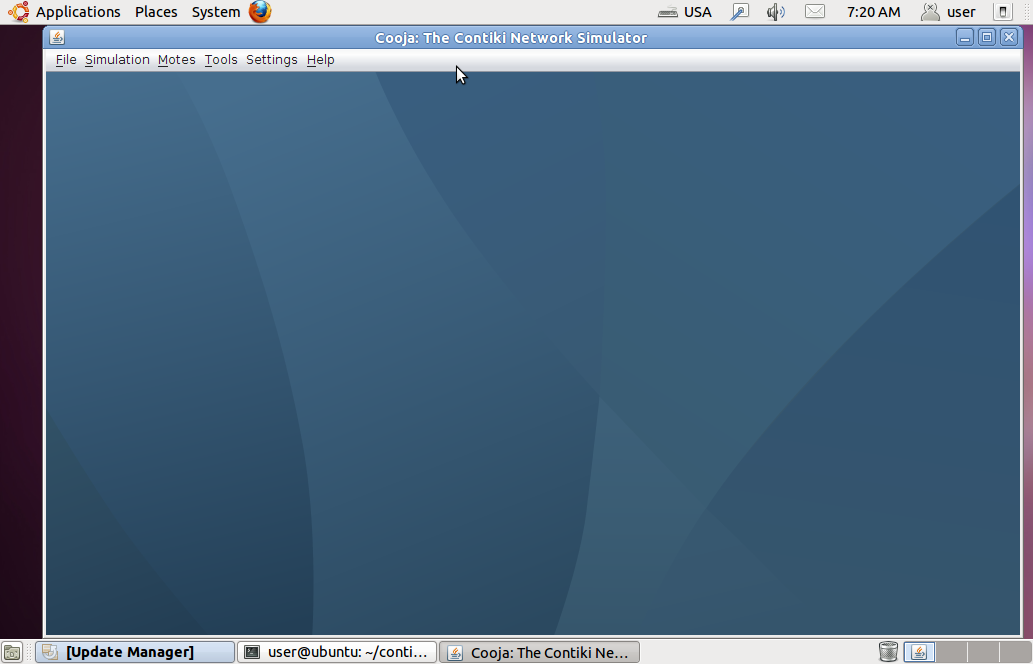
ant run

## **Wait for Cooja to start**

When Cooja first starts, it will first compile itself, which may take some time.



When Cooja is compiled, it will start with a blue empty window.



Now that Cooja is up and running, we can try it out with an example simulation.

# Step 3: Run Contiki in simulation

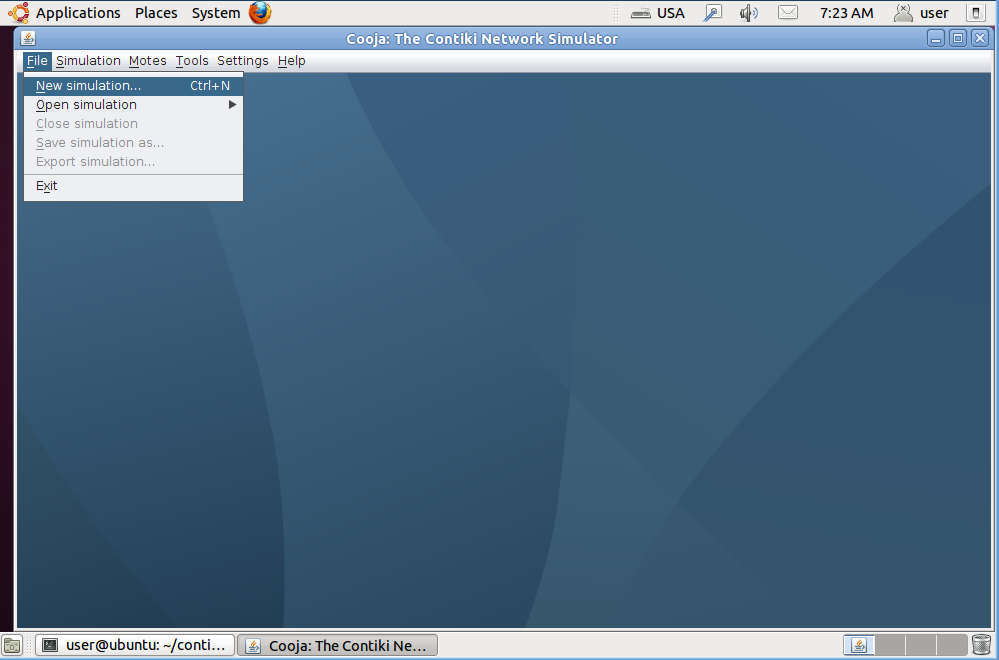
### **About Cooja simulations**

Cooja is a highly useful tool for Contiki development as it allows developers to test their code and systems long before running it on the target hardware. Developers regularly set up new simulations both to debug their software and to verify the behavior of their systems.

## **Create a new simulation**

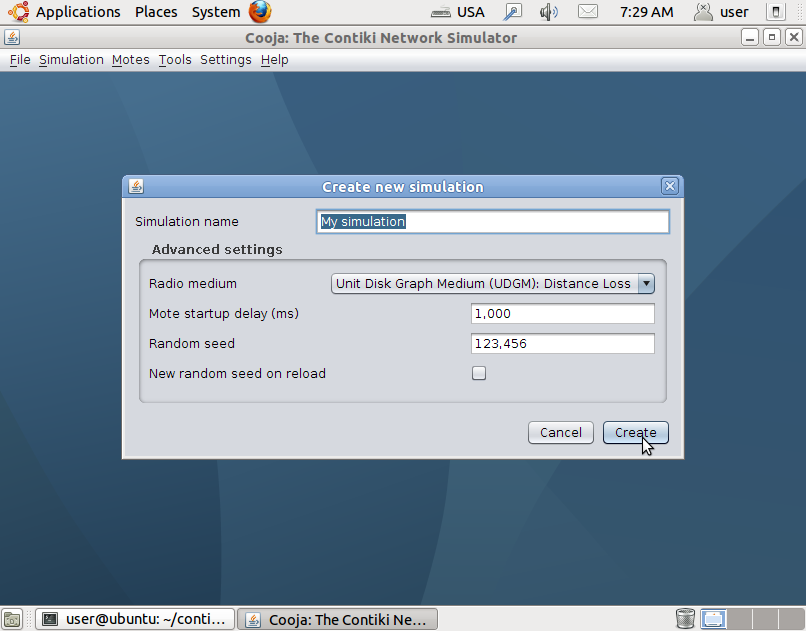
### **Create new simulation**

Click the **File** menu and click **New simulation...**.



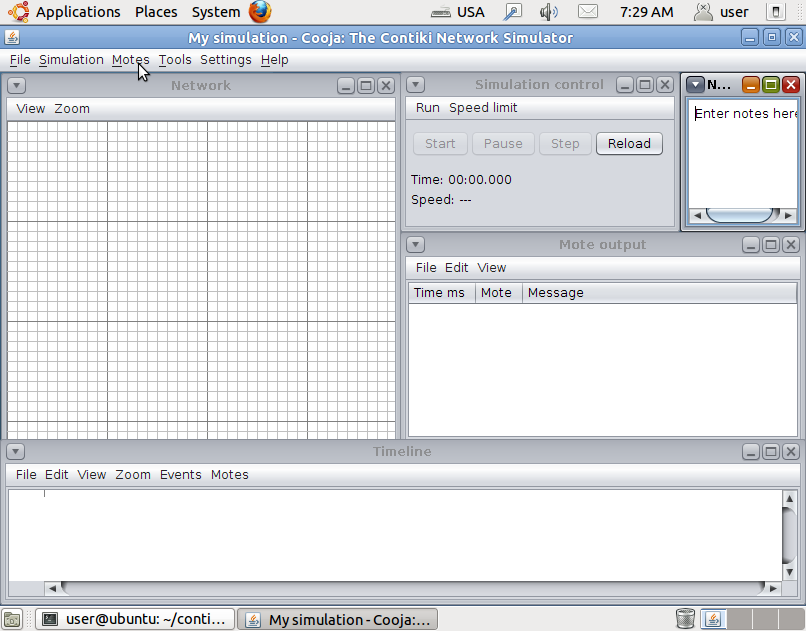
### **Set simulation options**

Cooja now opens up the **Create new simulation** dialog. In this dialog, we may choose to give our simulation a new name, but for this example, we'll just stick with My simulation. Click the **Create** button.



### **Simulation windows**

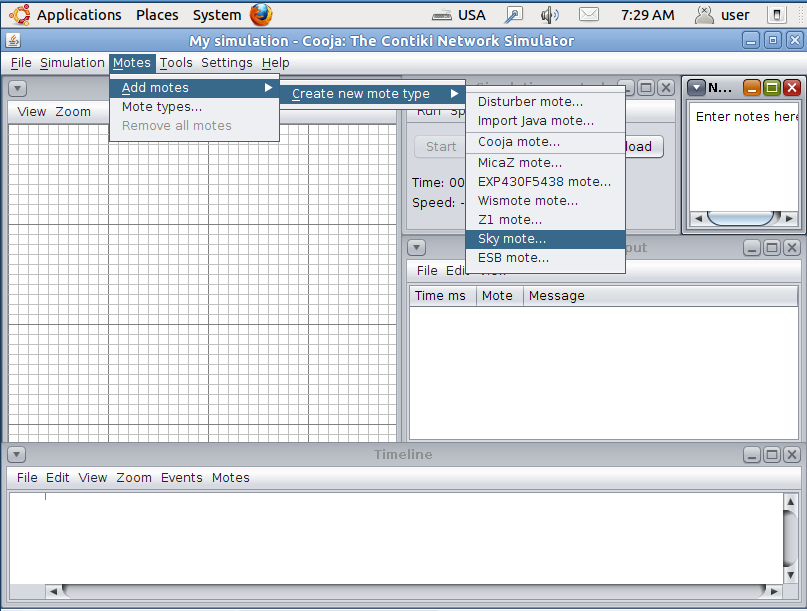
Cooja brings up the new simulation. The **Network** window, at the top left of the screen, shows all the motes in the simulated network - it is empty now, since we have no motes in our simulation. The **Timeline** window, at the bottom of the screen, shows all communication events in the simulation over time - very handy for understanding what goes on in the network. The **Mote output** window, on the right side of the screen, shows all serial port printouts from all the motes. The **Notes** window on the top right is where we can put notes for our simulation. And the **Simulation control** window is where we start, pause, and reload our simulation.



## **Add motes to the simulation**

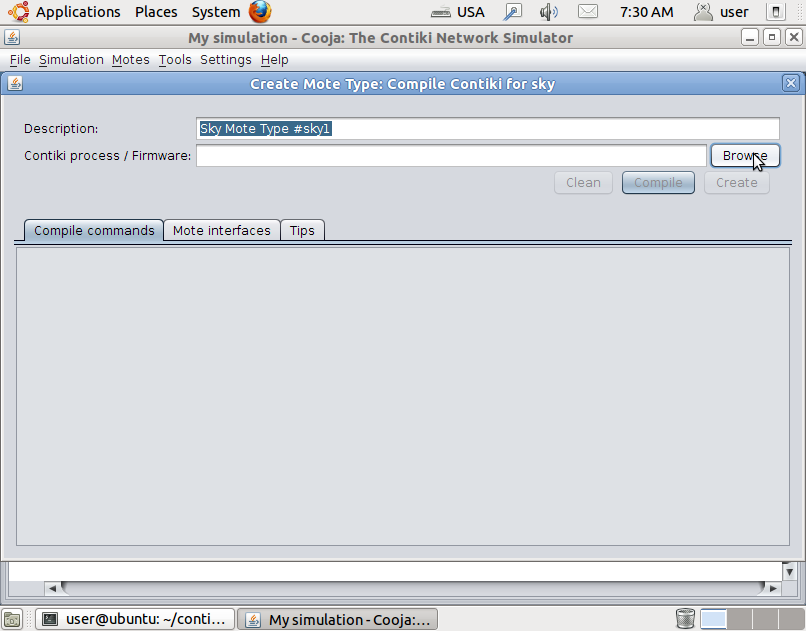
### **Add motes**

Before we can simulate our network, we must add one or more motes. We do this via the **Motes** menu, where we click on**Add motes...**. Since this is the first mote we add, we must first create a mote type to add. Click **Create new mote type...** and select one of the available mote types. For this example, we click **Sky mote...** to create an emulated Tmote Sky mote type.



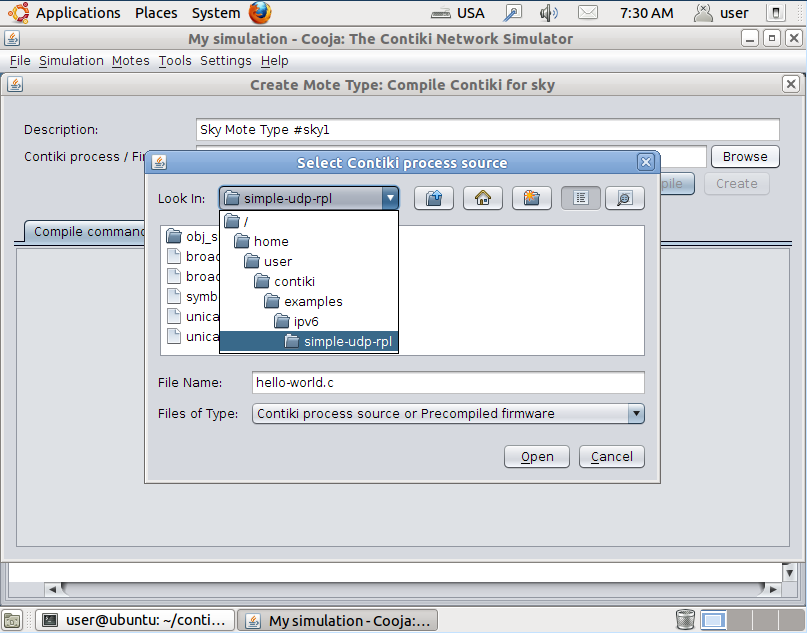
### **Create a new mote type**

Cooja opens the **Create Mote Type** dialog, in which we can choose a name for our mote type as well as the Contiki application that our mote type will run. For this example, we stick with the suggested name, and instead click on the **Browse...** button on the right hand side to choose our Contiki application.



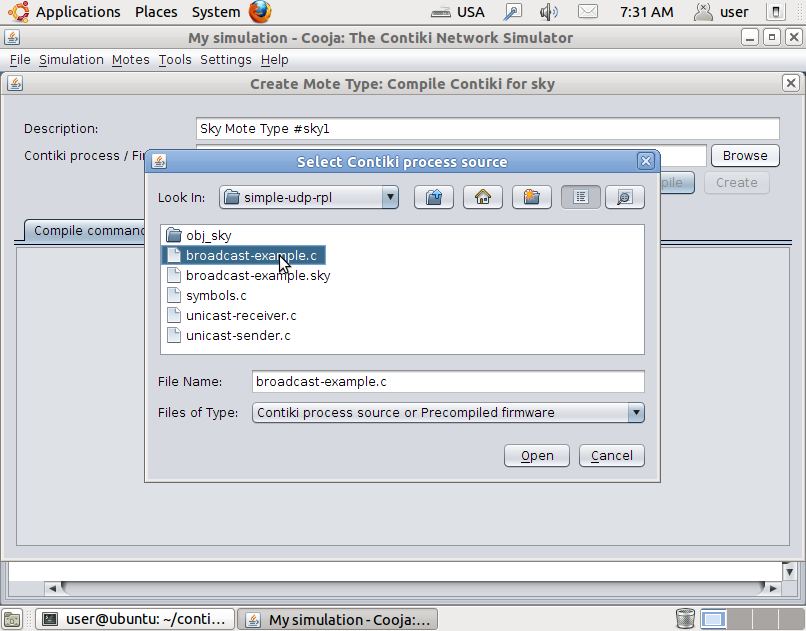
### **Find example Contiki application**

We go to the directory **/home/user/contiki/examples/ipv6/simple-udp-rpl**. This directory holds a number of Contiki applications that provide examples for how to do simple UDP communication over IPv6.



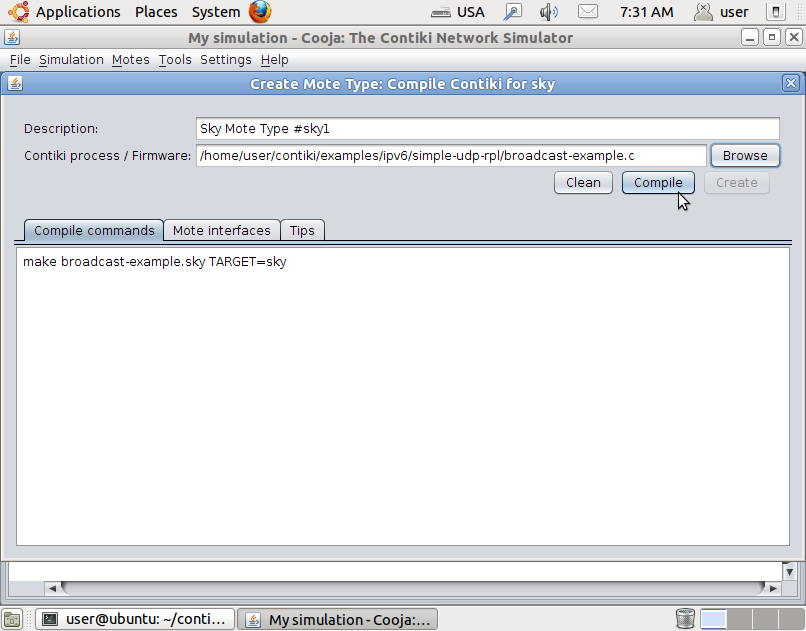
### **Specify application C source file**

Choose the file **broadcast-example.c**. This file contains a simple Contiki application that randomly broadcasts a UDP packet to its neighbors. Click the **Open** button to choose the file.



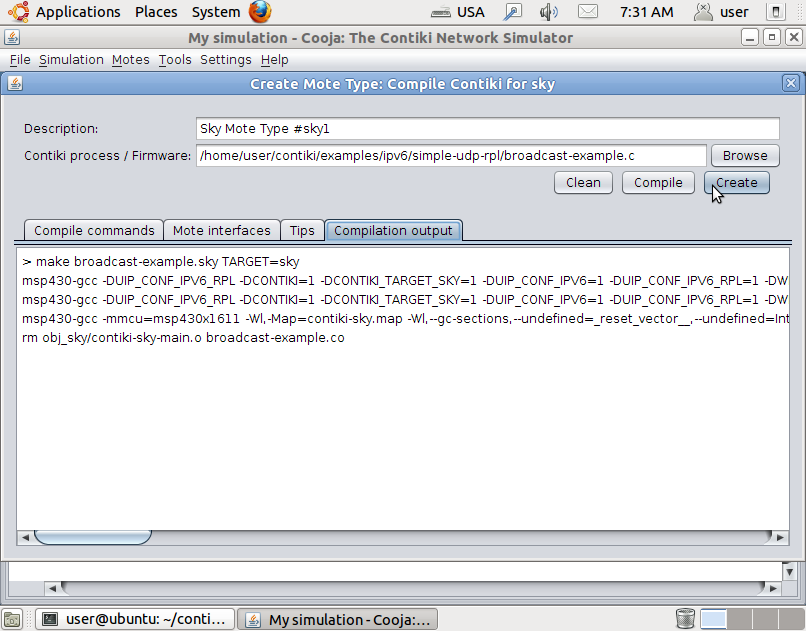
### **Compile Contiki and the application**

Now Cooja will verify that the selected Contiki application compiles for the platform that we have selected. Click the **Compile** button. This will take some time the first time around, expect it to take a minute at least. The compilation output will show up in the white panel at the bottom of the window.



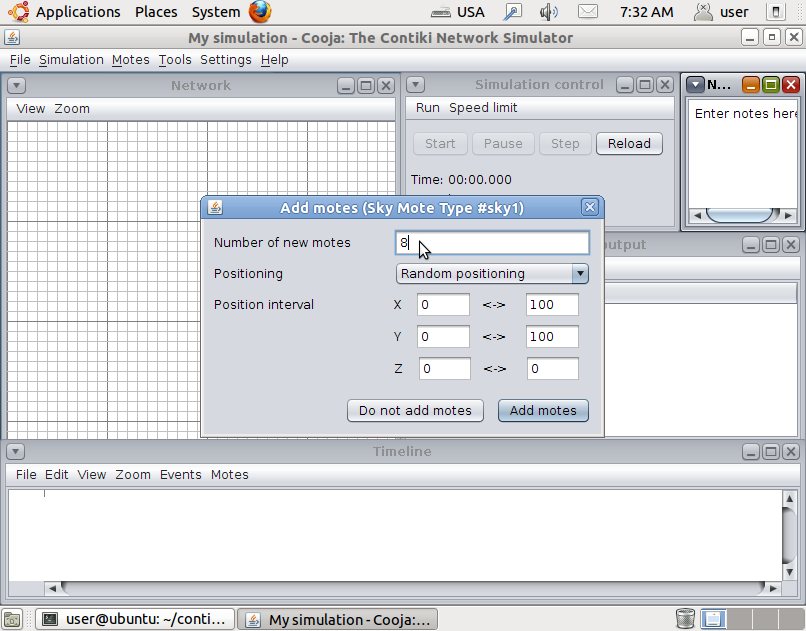
### **Create the mote type**

Click the **Create** button to create the mote type. The window will close.



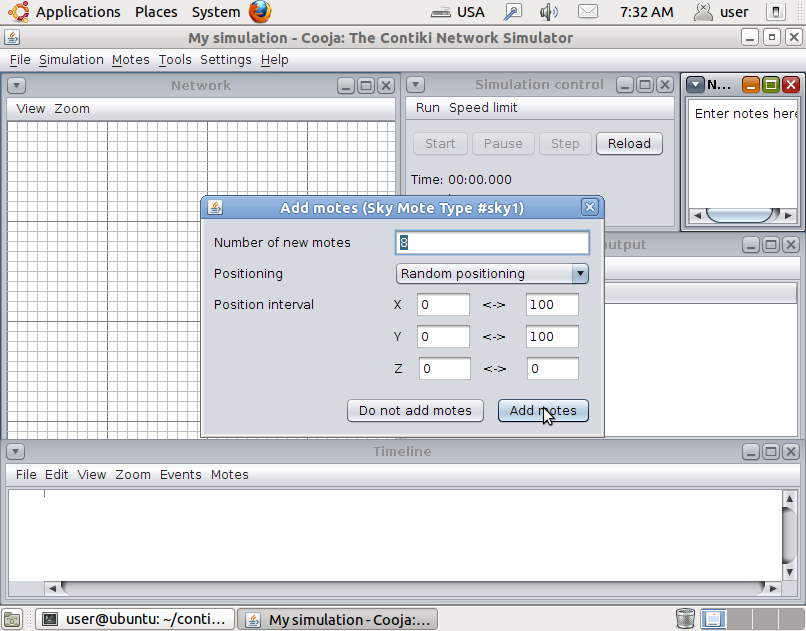
### **Add motes to simulation**

Cooja will now ask us if we want to add motes from the newly created mote type to the simulation. We change the number of motes to add in the **Number of motes** field to 8.



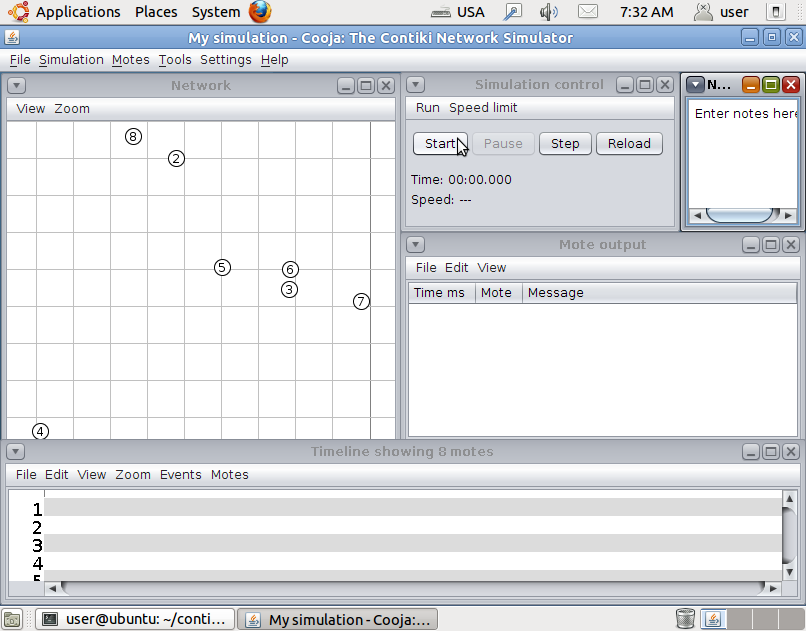
### **Add the motes**

We click the **Add motes** button to add the motes to the simulation.



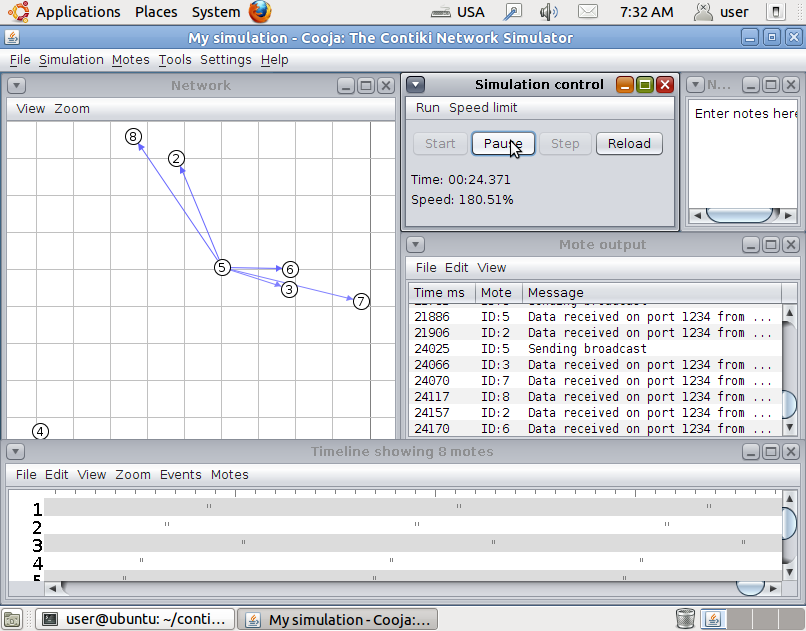
### **Start the simulation**

We can now see the 8 motes we added to the simulation in the **Network** window. Click the **Start** button to start the simulation.



### **Pause the simulation**

We see printouts from the simulated motes appearing in the **Mote output** window. The **Network** window shows communication going on in the network. The **Timeline** window shows communication and radio events over time - the small gray lines are ContikiMAC periodically waking the radio up. We can click the **Pause** button to pause the simulation.



## **Done!**

Congratulations! We have now created your first Cooja simulation with Contiki nodes that send periodic IPv6/UDP packets with ContikiMAC sleepy router functionaliy.

# Step 4: Run Contiki on hardware

### **About Contiki on hardware**

Contiki has a build system that is intended to make it easy to run Contiki directly on hardware. The build system is designed to be the same across different hardware platforms, so that the build commands are familiar when switching hardware. The build system consists of a set of makefiles. The base makefile iscontiki/Makefile.include, platform makefiles incontiki/platform/\*/Makefile.platformand contiki/cpu/\*/Makefile.cpu.

## **Connect the hardware**

In this example, we assume you have a [Zolertia Z1](http://www.contiki-os.org/hardware.html) mote connected to a USB port of your PC. We first need to enable the Z1 in Instant Contiki, so that Instant Contiki is able to talk to it. This is done through the Virtual Machine menu in VMWare Player, followed by theRemovable Devices menu item.

## **Open a terminal, go to the code directory**

The Contiki build system is run in a terminal window. Open a terminal window and go to the Hello World example directory:

cd contiki/examples/hello-world

## **Compile Contiki and the application**

We can now compile the Hello World application for our hardware platform. This also compiles the entire Contiki system, so it will take some time the first time around.

make TARGET=z1 hello-world

If you plan to compile more than once for the chosen platform, you can ask Contiki to remember your choice of hardware with the special savetarget maketarget. Do this now:

make TARGET=z1 savetarget

## **Upload Contiki to the hardware**

Now we are ready to upload the compiled code to the hardware. We do this with the special %.upload maketarget like this:

make hello-world.upload

**Note:** If you see output along the lines of the following, that means that the Z1 mote has not been connected to Instant Contiki:

make z1-reset z1-upload

make[1]: Entering directory `/home/user/contiki/examples/hello-world'

make -k -j 20 z1-reset-sequence

make[2]: Entering directory `/home/user/contiki/examples/hello-world'

Done

make[2]: Leaving directory `/home/user/contiki/examples/hello-world'

make -j 20 z1-upload-sequence

make[2]: Entering directory `/home/user/contiki/examples/hello-world'

Done

make[2]: Leaving directory `/home/user/contiki/examples/hello-world'

make[1]: Leaving directory `/home/user/contiki/examples/hello-world'

rm hello-world.ihex

Check the connection to the PC, check the connection in VMWare Player, and try again.

Note that we could have done the compilation and uploading in one step: the make hello-world.upload target automatically compiles the application and Contiki before uploading it to the hardware.

## **See the Hello World output**

Now that the code is burned onto the hardware device, we can check the serial port to see the output of the program. To see the serial port output of our connected mote, we run the following command:

make login

This will not show any output (except potentially a few garbage characters which sometimes seem to be inserted by VMWare) because the Hello World program has already finished. To see the output, press the reset button on the mote. Something like the following should show up:

Rime started with address 1.1

MAC 01:01:00:00:00:00:00:00 Contiki-2.6 started. Node id is set to 257.

CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26

Starting 'Hello world process'

Hello, world

## **Done!**

Congratulations! If you have completed all the steps in this tutorial, you have successfully been running Contiki both in simulation and on hardware. If everything worked, you might try to run the code we ran in simulation on hardware: upload the **broadcast-example**program in the **contiki/examples/ipv6/simple-udp-rpl/** directory to two hardware devices and look at them sending low-power wireless IPv6/UDP messages to each other.

# Step 5: Now what?

## **Hardware**

To see what your hardware options for Contiki are, or get hold of a hardware platform to run Contiki on, go to the hardware page:

[Hardware »](http://www.contiki-os.org/hardware.html)

## **Source code**

To dive into the Contiki source code, you can download the latest release or grab the current development version with git:

[Download »](http://www.contiki-os.org/download.html)

## **Community**

To engage with the Contiki developer community, go to the Contiki community page:

[Community »](http://www.contiki-os.org/community.html)

## **License**

To read the legal details of the Contiki open source license, go to the Contiki license page:

[Open source »](http://www.contiki-os.org/license.html)

## **Resources**

Read more about further Contiki resources:

[Resources »](http://www.contiki-os.org/support.html)

Hosted by [Thingsquare](http://thingsquare.com/" \t "_blank)

##### Description

The CC2538 development kit provides a complete hardware performance test platform and generic software development environment for the ARM Cortex M3-based, IEEE 802.15.4 compliant CC2538 system-on-chip from Texas Instruments. The kit includes two CC2538 RF evaluation modules (CC2538EM), two general purpose development boards (SmartRF06EB) for software and hardware prototyping, one CC2531 USB dongle for packet sniffing, cables, and documentation to get you up and running with the CC2538 quickly and easily.

The CC2538 evaluation modules can be used as reference modules for prototyping and for verifying the performance of the CC2538 RF IC. In combination with SmartRF06EB and SmartRF Studio, you have a complete set of tools for measuring RF parameters and more generally evaluate the RF performance of the chip. SmartRF Studio can generate register values and set up the radio to send and receive packets, set up a continuous wave signal and read the received signal strength (RSSI).

The SmartRF06EB has integrated the XDS100v3 debug probe, so no additional hardware tools are required for debugging of software running on the devices. The debug probe is compatible with Code Composer Studio and IAR Embedded Workbench. Other JTAG probes that support ARM Cortex M3 devices, like JLink from Segger and I-jet from IAR, can also be used to download and debug software on the CC2538.

The CC2538EM boards come pre-programmed with a Packet Error Rate test, which can be used for practical range testing of the radio. Texas Instruments' ZigBee stack (Z-Stack) is of course also available for the kit.

##### Features

* 2 x SmartRF06 Evaluation Boards
* 2 x CC2538 Evaluation Modules
* 1 x CC2531 USB Dongle (for packet sniffing)
* Cables
* Documentation

# Virtual development environment

* Base contiki SDK <http://sourceforge.net/projects/contiki/files/Instant%20Contiki/> version 2.7
* Getting started instructions <http://www.contiki-os.org/start.html>

# Hardware

* Buy <http://www.ti.com/tool/cc2538dk>

# Current component suite

<https://github.com/nallott/picosec-code/tree/master/bson-cpp-master>

<https://github.com/nallott/picosec-code/tree/master/contiki>

<https://github.com/nallott/picosec-code/tree/master/jsoncpp-master>

<https://github.com/nallott/picosec-code/tree/master/jsonrpc-cpp-0.4>

<https://github.com/nallott/picosec-code/tree/master/libjson-rpc-cpp-master>

<https://github.com/nallott/picosec-code/tree/master/libwebsockets>

<https://github.com/nallott/picosec-code/tree/master/mDNSResponder-544>

<https://github.com/nallott/picosec-code/tree/master/mdns>

<https://github.com/nallott/picosec-code/tree/master/tinydtls-0.4.0>

<https://github.com/nallott/picosec-code/tree/master/ubiOS>

<https://github.com/nallott/picosec-code/tree/master/uip-cygwin>