

WIMEA-ICT RSS2 SENSOR NODE APPLICATION BASED ON CONTIKI EMBEDDED OPERATING SYSTEM

TECHNICAL MANUAL

Version 1.0

Prepared by

Mary Nsabagwa (mnsabagwa@cit.ac.ug)

Sandra Alinitwe (sandraalinitwe@gmail.com)

Ian Tukesiga (tukian04@gmail.com)

Under the

WIMEA-ICT Project <http://wimea-ict.net/>

Table of Contents

Table of Contents	ii
1. Introduction	1
2. Getting started with Contiki Operating System	1
2.1 Accessing Contiki Repository	1
3. The Contiki Directory Structure	2
4. Building your first contiki application	3
5. Installing AVR toolchain	6
6. Compiling and downloading the Application Firmware	8
7. Accessing RSS2 Mote using serial Client	12
8. Writing and adding a custom drivers to Contiki operating system	14
References	15

Table of figures

Figure 1. Cloning WIMEA-ICT contiki repository.....	1
Figure 2. Instant Contiki log in form.....	2
Figure 3. Directory Structure of Contiki Operating System.	2
Figure 4. Basic contiki application.....	4
Figure 5. Contents of a sample Makefile	5
Figure 6. Contents of project-conf.h file	6
Figure 7. Installing avr- toolchain.....	7
Figure 8. Confirming that avr-dude is successfully installed.....	7
Figure 9. Compiling the application	8
Figure 10. Listing details of the files	9
Figure 11. Checking for serial Port being used.....	9
Figure 12. Downloading the application on the RSS2 node.....	11
Figure 13 Avrdudes window , used in windows OS	12
Figure 14. Installing putty from the terminal.....	13
Figure 15. Putty configuration for serial line and speed.....	13
Figure 16 Serial Interface for the application	14

1. Introduction

This technical manual is a guide on developing applications for an RSS2 wireless sensor node based on contiki operating system[1]. The manual gives a step-by-step guide on writing a contiki application for the RSS2 mote and demonstrates how such firmware is downloaded to the mote. RSS2 Wireless sensor node provides interfaces for connecting a wide range of sensors and has been used in implementation applications for Automatic Weather Stations (AWSs).

Contiki is an open source operating system for the Internet of things used to connect tiny low-cost and low-power microcontrollers. It supports multitasking and a built-in Internet Protocol Suite (TCP/IP stack).

2. Getting started with Contiki Operating System

Windows operating system users may install VMWare player, a free desktop application that supports running many operating system on a single host operating system. VMWare for windows can be downloaded here [2]. Some software is required including git and avr-toolkit

2.1 Accessing Contiki Repository

Open the terminal after booting your Linux machine

- i. At the terminal, clone the contiki operating system repository, using the command
git clone <https://github.com/wimea-ict/contiki>

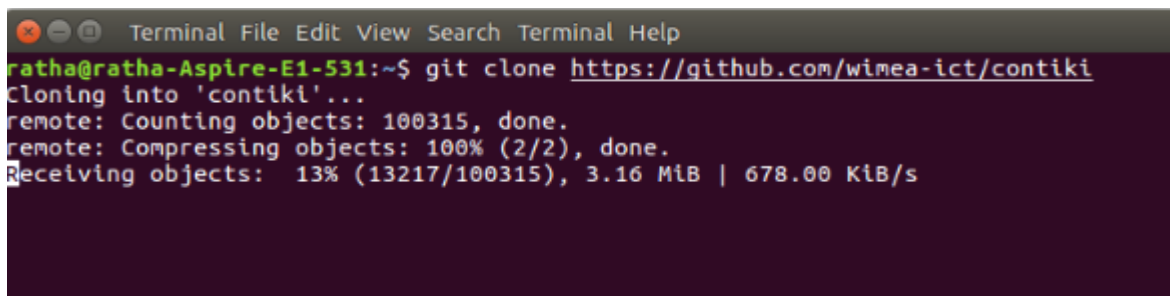
A screenshot of a terminal window with a dark background. The window title bar shows 'Terminal File Edit View Search Terminal Help'. The prompt is 'ratha@ratha-Aspire-E1-531:~\$'. The command entered is 'git clone https://github.com/wimea-ict/contiki'. The output shows 'Cloning into 'contiki'...', 'remote: Counting objects: 100315, done.', 'remote: Compressing objects: 100% (2/2), done.', and 'Receiving objects: 13% (13217/100315), 3.16 MiB | 678.00 KiB/s'.

Figure 1. Cloning WIMEA-ICT contiki repository

In order to simulate contiki, download instant contiki from <https://sourceforge.net/projects/contiki/files/InstantContiki> and follow login using password: user (Figure 2)

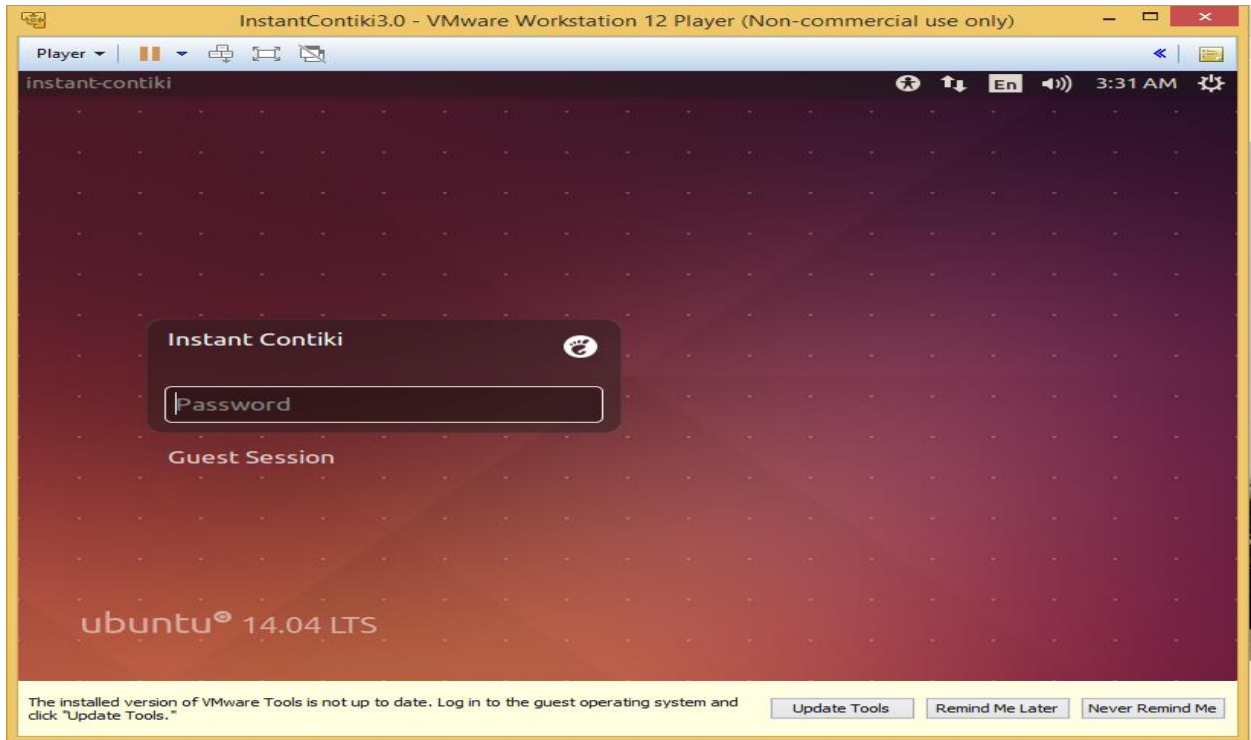


Figure 2. Instant Contiki log in form

3. The Contiki Directory Structure

The Contiki directory structure is illustrated in Figure 4

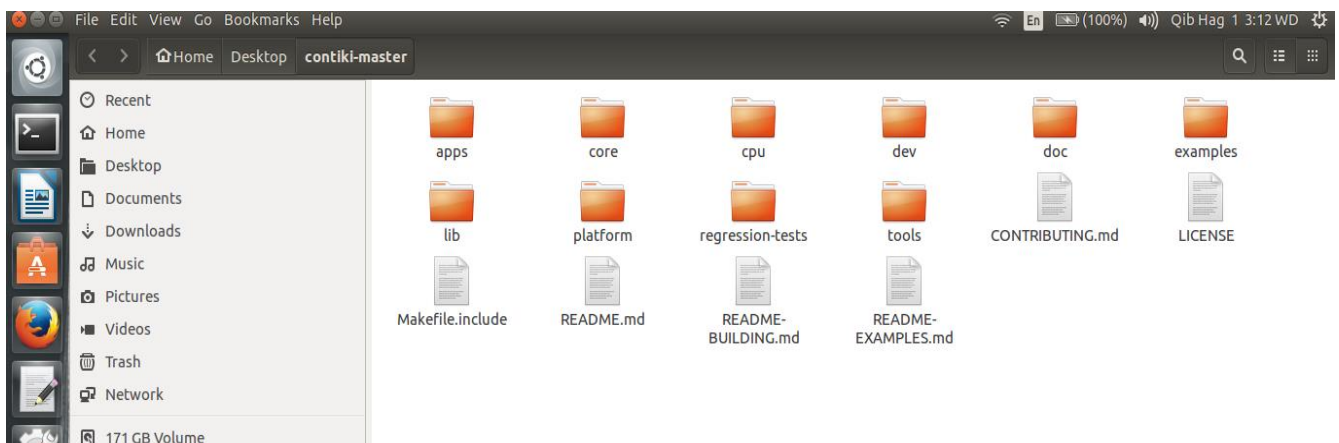


Figure 3. Directory Structure of Contiki Operating System.

Below are the directories in the root directory of contiki:-

- i. examples: Contains examples of Contiki's applications This directory contains a list of ready to compile contiki applications that the user can start with apps: applications ready to use
- ii. cpu: contains specific MCU files, the /cpu/msp430/ folder contains the drivers of the MCU used by the Z1 mote.
- iii. Platform: contains specific board files and drivers. Contiki applications for different hardware platforms can easily be compiled by supplying different parameters to the make command without editing the make file.
- iv. Core: contains Contiki's file system containing folders and files such as /dev, header files for devices such as sensors, LEDs, files and libraries for networking such as CC2420 transceiver and Contiki file system
- v. Doc: contains a list of text files that correspond to the directories in the contiki structure
- vi. dev: contains files for external devices that are supported by the Contiki Operating System
- vii. Lib: contains libraries that are used to develop different contiki applications
- viii. Makefile: this contains instructions for the compilation tools.
- ix. The makefile.include- it contains all the C files of the core contiki system and it is located in the root of the contiki source tree. It includes Makefile.\$(TARGET) and all applications.

4. Building your first contiki application

- i. Change directory to contiki/examples folder create a directory named my_contiki_application
- ii. Copy HelloWorld.c, Makefile, and projectconf files from /contiki/examples/helloworld into your newly created directory
- iii. Rename HelloWorld.c to my_contiki_application.c and open the file for editing. Figure 4 shows the structure of the program

```

#include "contiki.h"

#include <stdio.h> /* For printf() */
PROCESS(my_application_process, "my_application");
/*-----
AUTOSTART_PROCESSES(&my_application_process);
/*-----
PROCESS_THREAD(my_application_process, ev, data)
{
    PROCESS_BEGIN();

    printf("This is my first application\n");

    PROCESS_END();
}
/*-----

```

Figure 4. Basic contiki application

Explanation of the above code.

#include "contiki.h"

This contains all the declarations of the contiki operating systems.

#include <stdio.h>

for printf function

PROCESS(my_application_process, "my_application");

Every process in contiki starts with a process macro and it takes two arguments that is the variable name of the process structure and the string representation of the process name as the second parameter.

AUTOSTART_PROCESSES(&my_application_process);

Automatically starts the process given in the arguments. That is, it will start my_application process.

PROCESS_THREAD(my_application_process, ev, data){

MACRO used to define of a protothread of a process which is called whenever an event occurs

in the system. It contains the `PROCESS_BEGIN` and `PROCESS_END` macros and other user defined C statement(s)

`PROCESS_BEGIN();`

This macro defines the beginning of a process.

`printf("This is my first application\n");`

A simple c statement to be executed in `my_application.c`

`PROCESS_END();`

This macro defines the end of a process.

Note: if we compile this code and run it in the terminal, the string in the brackets of the `printf` function will be printed in the terminal and if this code is compiled in the Cooja Network Simulator, the string will be printed in the mote output window

iv. Editing the Makefile

```
CONTIKI_PROJECT = my_contiki_application
all: $(CONTIKI_PROJECT)

CONTIKI = ../..
include $(CONTIKI)/Makefile.include
```

Figure 5. Contents of a sample Makefile

`CONTIKI` variable defines the location of Contiki.

`CONTIKI_PROJECT` is the name of the application.

`APPS` variable includes the powertrace application from the `app/` directory.

When compiled, it creates supporting files like `.map`, `.avr-rss2`, and `.hex`.

v) Editing `project_conf.h` file


```

#ifndef PROJECT_CONF_H_
#define PROJECT_CONF_H_

#define NETSTACK_CONF_RDC nullrdc_driver
#define NETSTACK_CONF_MAC nullmac_driver

#define RS232_BAUDRATE USART_BAUD_38400

// #define NETSTACK_CONF_MAC          csma_driver
// #define NETSTACK_CONF_RDC          contikimac_driver

#endif /* PROJECT_CONF_H_ */

```

Figure 6. Contents of *project-conf.h* file

5. Installing AVR toolchain

This is a collection of tools used to create applications for AVR microcontrollers

Avr toolchain contains the following packages.

- i. avrdude
- ii. binutils-avr
- iii. gdb-avr
- iv. avr-libc
- v. gcc-avr

All these can be got in one shot using command

sudo apt-get install gcc-avr binutils-avr gdb-avr avr-libc avrdude

```
Terminal File Edit View Search Terminal Help
ratha@ratha-Aspire-E1-531:~$ sudo apt-get install gcc-avr binutils-avr gdb-avr a
vr-libc avrdude
Reading package lists... Done
Building dependency tree
Reading state information... Done
avr-libc is already the newest version (1:1.8.0+Atmel3.5.0-1).
avrdude is already the newest version (6.2-5).
binutils-avr is already the newest version (2.25+Atmel3.5.0-2).
gcc-avr is already the newest version (1:4.9.2+Atmel3.5.0-1).
gdb-avr is already the newest version (7.7-2build1).
0 upgraded, 0 newly installed, 0 to remove and 315 not upgraded.
ratha@ratha-Aspire-E1-531:~$
```

Figure 7. Installing avr- toolchain

```
Terminal File Edit View Search Terminal Help
ratha@ratha-Aspire-E1-531:~$ avrdude
Usage: avrdude [options]
Options:
  -p <partno>          Required. Specify AVR device.
  -b <baudrate>        Override RS-232 baud rate.
  -B <bitclock>        Specify JTAG/STK500v2 bit clock period (us).
  -C <config-file>     Specify location of configuration file.
  -c <programmer>      Specify programmer type.
  -D                  Disable auto erase for flash memory
  -i <delay>           ISP Clock Delay [in microseconds]
  -P <port>            Specify connection port.
  -F                  Override invalid signature check.
  -e                  Perform a chip erase.
  -O                  Perform RC oscillator calibration (see AVR053).
  -U <memtype>[:r|w|v:<filename>[:format]]
                        Memory operation specification.
                        Multiple -U options are allowed, each request
                        is performed in the order specified.
  -n                  Do not write anything to the device.
  -V                  Do not verify.
  -u                  Disable safemode, default when running from a script.
  -s                  Silent safemode operation, will not ask you if
                        fuses should be changed back.
  -t                  Enter terminal mode.
  -E <exitspec>[,<exitspec>] List programmer exit specifications.
  -x <extended_param> Pass <extended_param> to programmer.
  -y                  Count # erase cycles in EEPROM.
  -Y <number>          Initialize erase cycle # in EEPROM.
  -v                  Verbose output. -v -v for more.
  -q                  Quell progress output. -q -q for less.
  -l logfile           Use logfile rather than stderr for diagnostics.
  -?                  Display this usage.

avrdude version 6.2, URL: <http://savannah.nongnu.org/projects/avrdude/>
ratha@ratha-Aspire-E1-531:~$
```

Figure 8. Confirming that avr-dude is successfully installed

In case after installing the avr toolchain, you get errors when you compile, uninstall the avr-toolchain with the command

`sudo apt-get remove package_names`

re-install using the following steps

```
# Ubuntu and 4.8.1 only! Add new avr-gcc 4.8.1 package
sudo add-apt-repository ppa:pmjdebruijn/gcc-avr-release
# reload repositories and check for the gcc-avr package
```

```

sudo apt-get update
sudo apt-cache search gcc-avr

# install avr-gcc 4.7.2
sudo apt-get install gcc-avr avr-libc

# check installed version of avr-gcc
avr-gcc -v

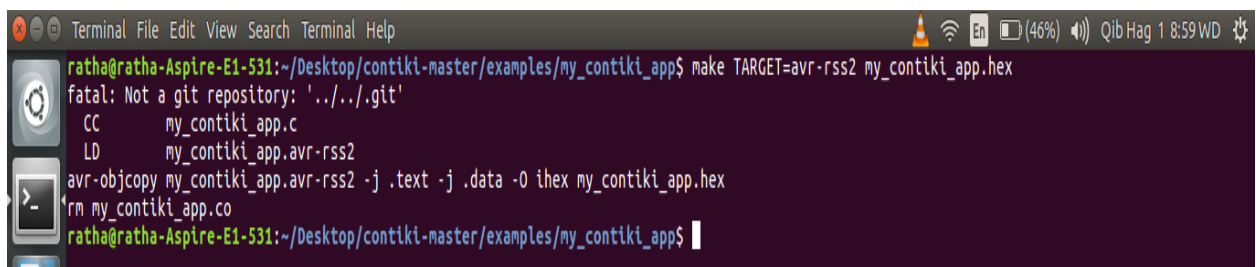
```

6. Compiling and downloading the Application Firmware

- i) Go to the directory of the C file you are to make the hex file for. For example if the directory name is “my-contiki-application”. In the first line of the Makefile, put the correct path to the directory where your contiki source file resides. As an example, if your source code is located in the path ”/home/apps/desktop/contiki” your first line of the Makefile should look like below
CONTIKI= /home/apps/desktop/contiki
- ii) To compile the application, use the command on the terminal:

make TARGET=avr-rss2 application_name.hex

The command translates the .C file into the hex file.



```

Terminal File Edit View Search Terminal Help
ratha@ratha-Aspire-E1-531:~/Desktop/contiki-master/examples/my_contiki_app$ make TARGET=avr-rss2 my_contiki_app.hex
fatal: Not a git repository: '../.git'
CC      my_contiki_app.c
LD      my_contiki_app.avr-rss2
avr-objcopy my_contiki_app.avr-rss2 -j .text -j .data -O ihex my_contiki_app.hex
rm my_contiki_app.co
ratha@ratha-Aspire-E1-531:~/Desktop/contiki-master/examples/my_contiki_app$

```

Figure 9. Compiling the application

Note: **avr-rss2** is the platform, which specifies the RSS2 node driver. The platform name is case sensitive.

Confirm that the generated .hex file is less than 256KB (Figure 10)

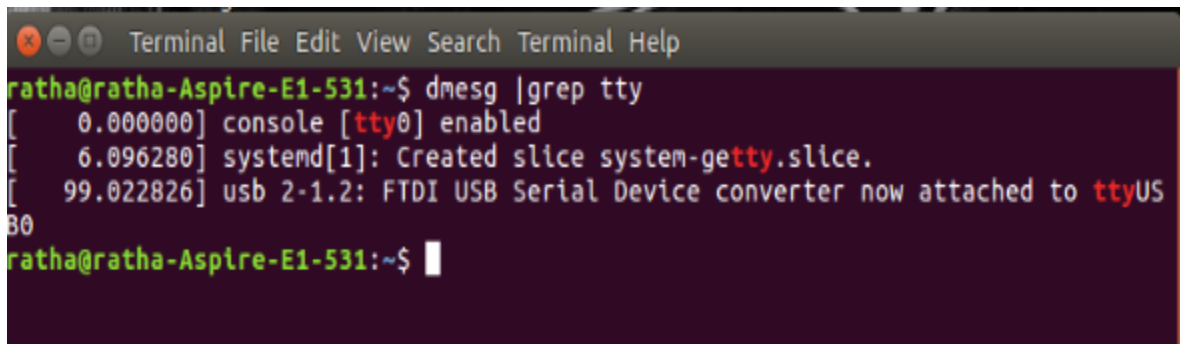


```
Terminal File Edit View Search Terminal Help
ratha@ratha-Aspire-E1-531:~/Desktop/contiki-master/examples/my_contiki_app$ make TARGET=avr-rss2 my_contiki_app.hex
fatal: Not a git repository: '../.git'
CC      my_contiki_app.c
LD      my_contiki_app.avr-rss2
avr-objcopy my_contiki_app.avr-rss2 -j .text -j .data -O ihex my_contiki_app.hex
rm my_contiki_app.co
ratha@ratha-Aspire-E1-531:~/Desktop/contiki-master/examples/my_contiki_app$ ls -l
total 2708
-rw-rw-r-- 1 ratha ratha 1896260 Hag 1 08:43 contiki-avr-rss2.a
-rw-rw-r-- 1 ratha ratha 219618 Hag 1 08:59 contiki-avr-rss2.map
-rw-rw-r-- 1 ratha ratha 110 Hag 1 08:38 Makefile
-rwxrwxr-x 1 ratha ratha 462368 Hag 1 08:59 my_contiki_app.avr-rss2
-rw-rw-r-- 1 ratha ratha 2323 Hag 1 08:59 my_contiki_app.c
-rw-rw-r-- 1 ratha ratha 164603 Hag 1 08:59 my_contiki_app.hex
drwxrwxr-x 2 ratha ratha 12288 Hag 1 08:43 obj_avr-rss2
-rw-rw-r-- 1 ratha ratha 1231 Ado 23 16:06 README.md
ratha@ratha-Aspire-E1-531:~/Desktop/contiki-master/examples/my_contiki_app$
```

Figure 10. Listing details of the files

iii) Connect the mote to the computer via the FTDI cable.

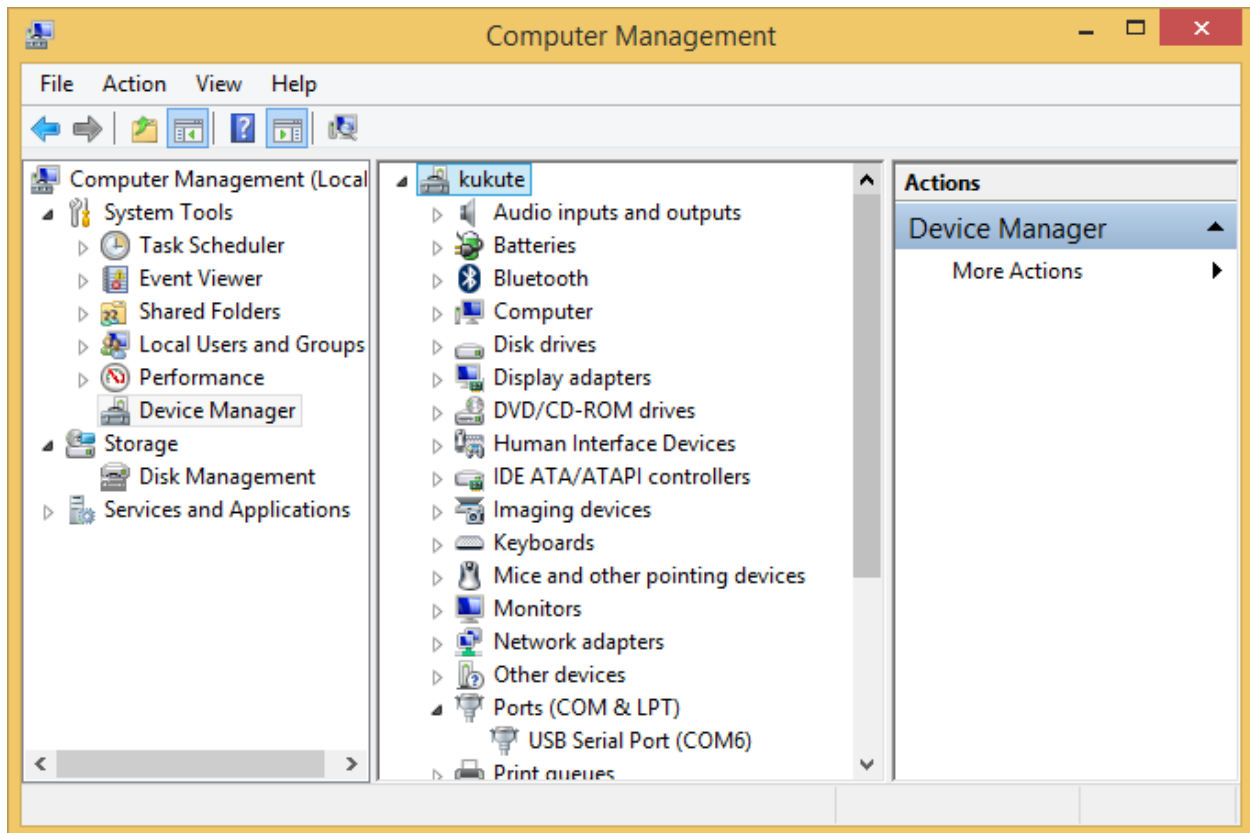
Run the command **dmesg | grep tty** to see the port on which the device has been attached



```
Terminal File Edit View Search Terminal Help
ratha@ratha-Aspire-E1-531:~$ dmesg |grep tty
[ 0.000000] console [tty0] enabled
[ 6.096280] systemd[1]: Created slice system-getty.slice.
[ 99.022826] usb 2-1.2: FTDI USB Serial Device converter now attached to ttyUS
30
ratha@ratha-Aspire-E1-531:~$
```

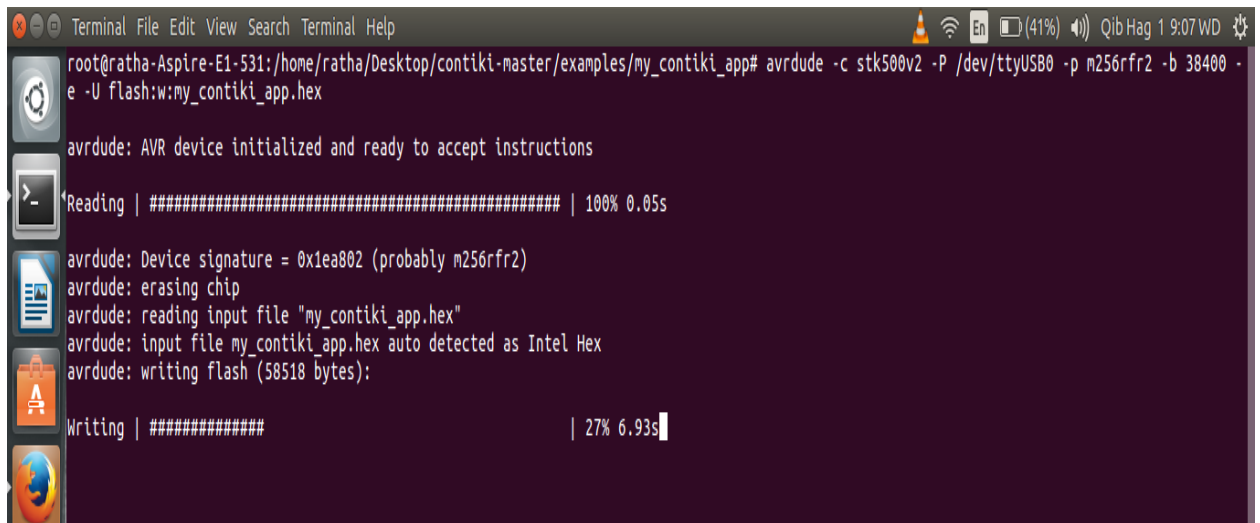
Figure 11. Checking for serial Port being used

When checking for the serial port in windows, right click my computer and select “Manage”. Click Device manager and on the right-hand side, check the port number under Ports (COM & LPT)



- iv) Download .hex file on the RSS2 mote using command below:-
- ```
avrdude -c stk500v2 -P /dev/ttyUSB0 -p m256rfr2 -b 38400 -e -U
flash:w:my_application.hex,
```
- (Press the reset button on the mote and after a few seconds press enter)*

*If interested in compiling wimea-ict-rss2 application, compile the application under  
/platform/avr-rss2/apps/wimea-ict-rss2*



```
Terminal File Edit View Search Terminal Help
root@ratha-Aspire-E1-531:/home/ratha/Desktop/contiki-master/examples/my_contiki_app# avrdude -c stk500v2 -P /dev/ttyUSB0 -p m256rfr2 -b 38400 -e -U flash:w:my_contiki_app.hex
avrdude: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.05s

avrdude: Device signature = 0x1ea802 (probably m256rfr2)
avrdude: erasing chip
avrdude: reading input file "my_contiki_app.hex"
avrdude: input file my_contiki_app.hex auto detected as Intel Hex
avrdude: writing flash (58518 bytes):

Writing | ##### | 27% 6.93s
```

*Figure 12. Downloading the application on the RSS2 node*

Windows users may download and install avrduess

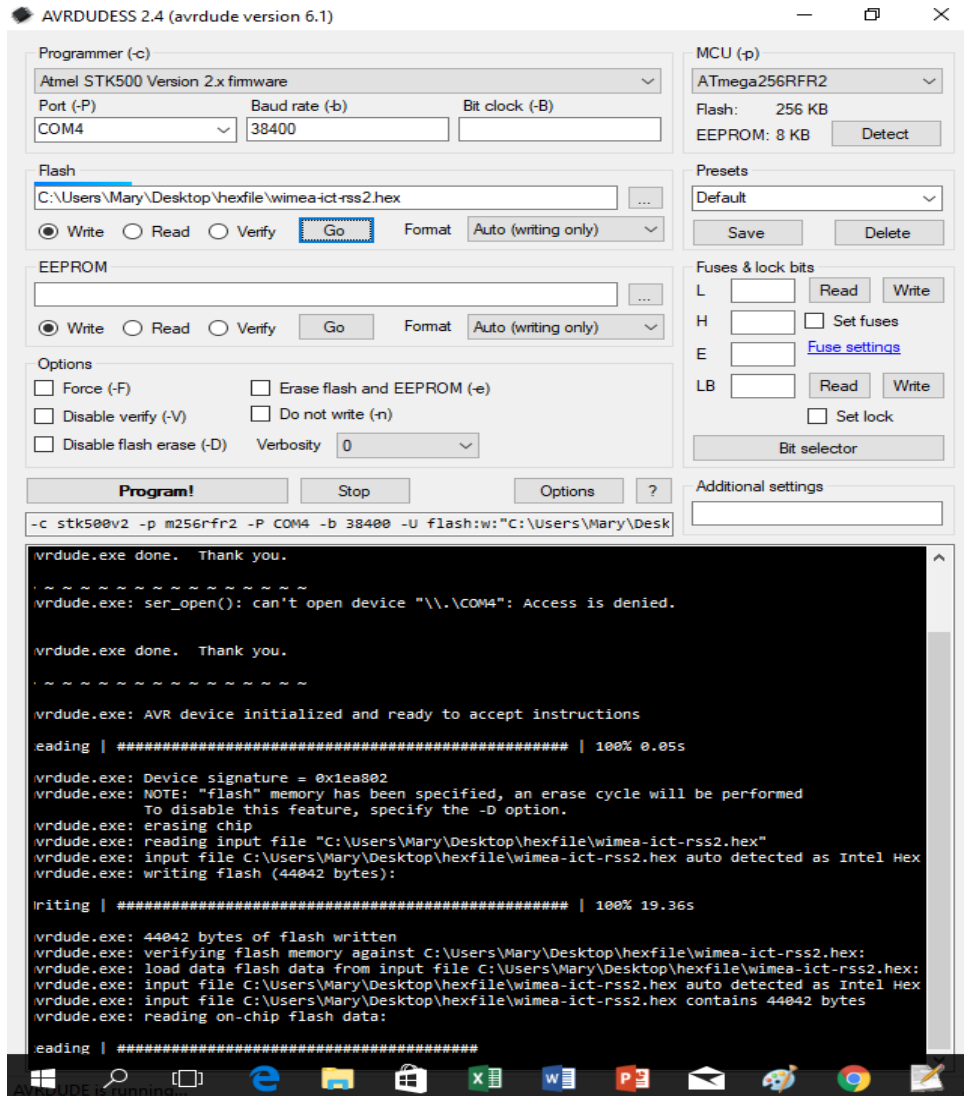


Figure 13 Avrdudes window , used in windows OS

## 7. Accessing RSS2 Mote using serial Client

Putty is a versatile tool for remote access using SSH and serial communication using a FTDI cable. Run command in Figure 14 to install putty on Linux (if it does not exist) and configure as in Figures 15

```
Terminal File Edit View Search Terminal Help
root@ratha-Aspire-E1-531:/home/ratha# sudo apt-get install putty
Reading package lists... Done
Building dependency tree
Reading state information... Done
putty is already the newest version (0.67-2).
The following packages were automatically installed and are no longer required:
 linux-headers-4.4.0-21 linux-headers-4.4.0-21-generic
 linux-image-4.4.0-21-generic linux-image-extra-4.4.0-21-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 326 not upgraded.
root@ratha-Aspire-E1-531:/home/ratha#
```

Figure 14. Installing putty from the terminal

step2: Configuring Putty

Open putty using command **putty** on the terminal

Enter the serial communication number for example /dev/ttyUSB0 and the baud rate of 38400 as the speed in the serial box.

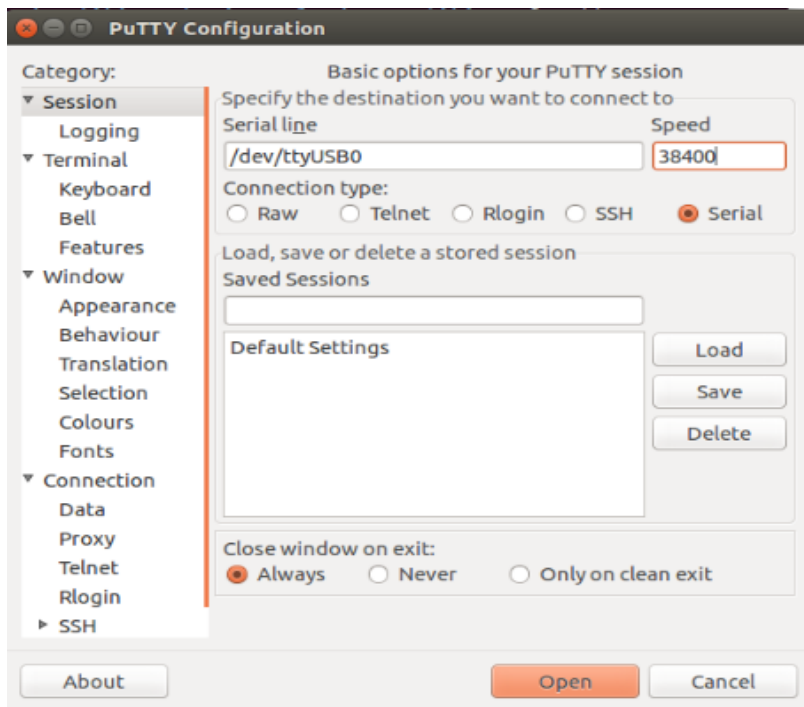
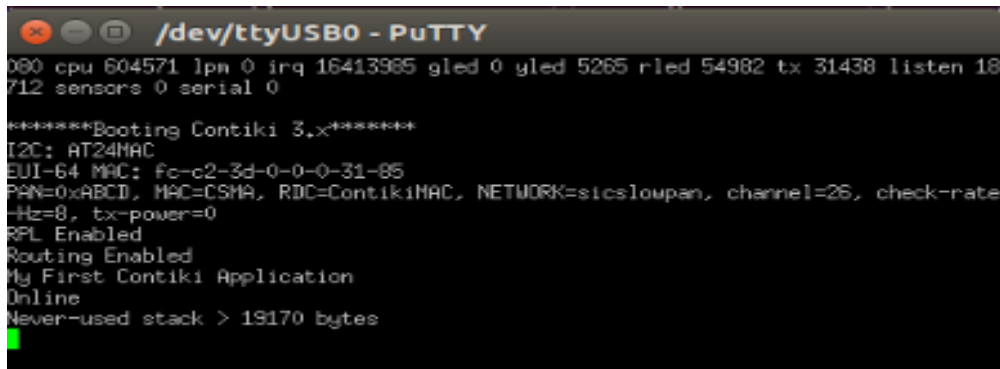


Figure 15. Putty configuration for serial line and speed

After pressing “Open” in Figure 15, Figure 16 shall appear



A screenshot of a PuTTY terminal window titled "/dev/ttyUSB0 - PuTTY". The terminal displays boot logs for a Contiki 3.x system. The logs include hardware statistics (cpu, lpm, irq, gled, rled, tx, listen, sensors, serial), boot messages ("\*\*\*\*\*Booting Contiki 3.x\*\*\*\*\*"), and system initialization details (I2C: AT24MAC, EUI-64 MAC, PAN, RDC, NETWORK, channel, check-rate, Hz, tx-power, RPL Enabled, Routing Enabled, My First Contiki Application, Online, Never-used stack).

```
/dev/ttyUSB0 - PuTTY
000 cpu 604571 lpm 0 irq 16413985 gled 0 rled 5265 tx 31438 listen 18
712 sensors 0 serial 0

*****Booting Contiki 3.x*****
I2C: AT24MAC
EUI-64 MAC: fc-c2-3d-0-0-0-31-85
PAN=0xABCD, MAC=CSMA, RDC=ContikiMAC, NETWORK=sicslowpan, channel=26, check-rate
-Hz=8, tx-power=0
RPL Enabled
Routing Enabled
My First Contiki Application
Online
Never-used stack > 19170 bytes
```

Figure 16 Serial Interface for the application

## 8. Writing and adding a custom drivers to Contiki operating system

Drivers contain functions that are used by the respective devices such as sensors and implementation details may be found in the respective datasheets of the drivers. Driver files can be found in the `contiki/platform/avr-rss2/dev` directory. `.c` and `.h` sample driver files are provided below:-

The C file contains some function implementations of the driver and the header file contains the prototypes of the functions in C file and are placed under `/dev` folder of the corresponding platform. The contents of these files may be like:-

My\_driver.h

```
void driver_function();
```

My\_driver.c

```
#include "dev/leds.h"

Static int status(int type){
Return 0;
}

static int value(int type){
case 1:
void driver_function(){
leds_on(LEDS_RED);
}
```

```

return 0;
}
Static int configure(int type,int value){

return 0;
}
SENSORS_SENSOR(My_driver, My_driver,value,configure,status);

```

Note that all driver implementation files contain functions value, configure and status.

Examples can be found under <https://github.com/wimea-ict/contiki/blob/master/platform/avr-rss2/dev/>.

In order to call or use the defined driver, append the created .c file to the src files in file contiki/platform/avr-rss2\$

CONTIKI\_TARGET\_SOURCEFILES += temp-sensor.c ds3231.c **My\_driver.c**

Next, open the application file, for instance contiki/platform/avr-rss2/apps/wimea-ict-rss2/wimea-ict-rss2.c and include the following

```

#include "dev/my_driver.h"

PROCESS_THREAD(process_name, ev, data)
{
 PROCESS_BEGIN();
 SENSORS_ACTIVATE(My_driver);
 printf("Value is ", My_driver.value(0));
 ..
 SENSORS_DEACTIVATE(My_driver);
}

```

Figure 17 Application that returns the result

Compile the program

## References

- [1] "Contiki." [Online]. Available: [github.com/wimea-ict/](https://github.com/wimea-ict/).

- [2] “VMWare” [Online]. Available: [www.vmware.com/](http://www.vmware.com/)
- [3] “Cygwin” [Online]. Available: [www.cygwin.com/](http://www.cygwin.com/)
- [4] “FTDI driver” [Online]. Available: [www.ftdichip.com/FTDrivers.html](http://www.ftdichip.com/FTDrivers.html)
- [5] “WinAVR” [Online]. Available: <https://sourceforge.net/projects/winavr/>
- [6] “VMWare”[Online].Available  
<https://www.vs.inf.ethz.ch/edu/SS2007/WSN/tut/software/avrdude-5.1p.zip>
- [7] “VMWare” [Online]. Available: [www. Putty.org/](http://www.putty.org/)