# 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 figures

# 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**

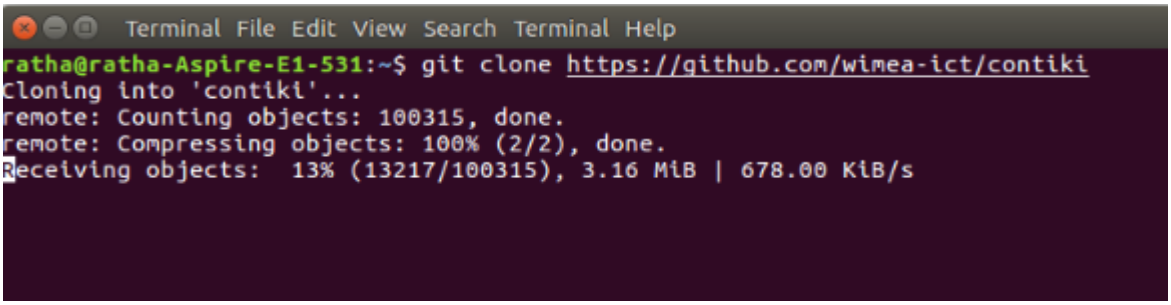



*Figure 1. Cloning WIMEA-ICT contiki repository*

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

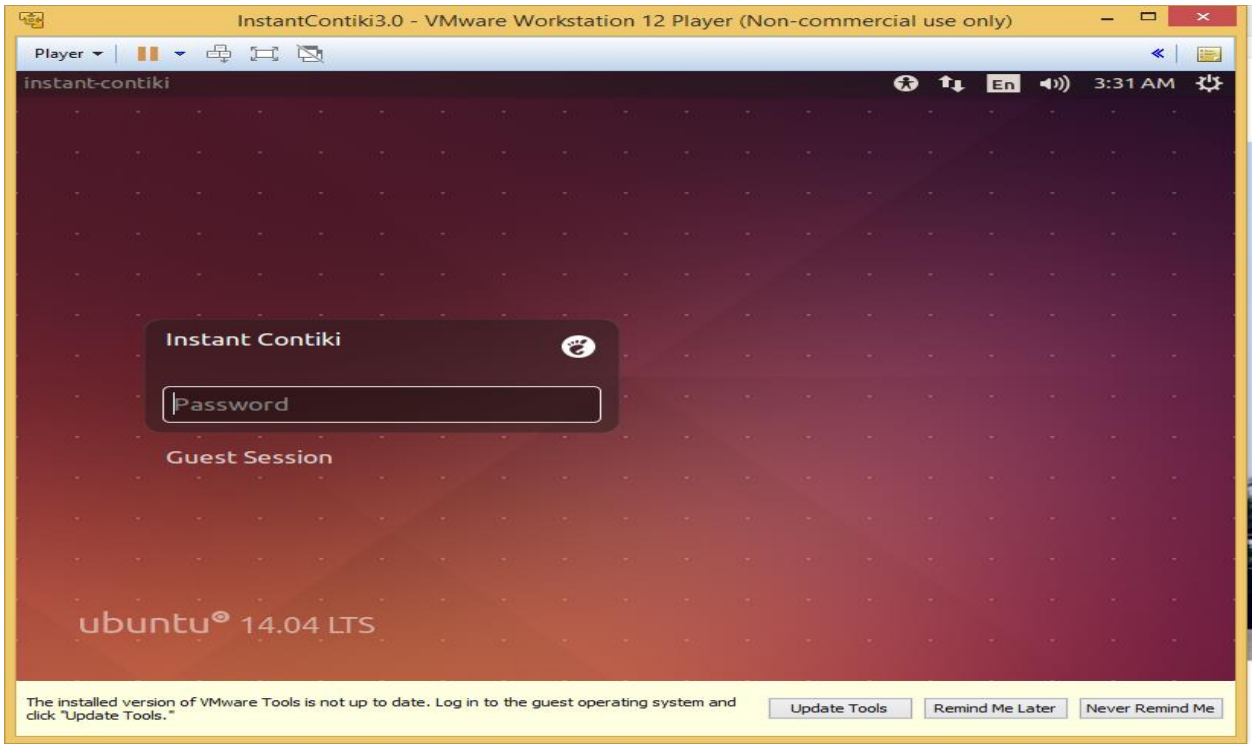



*Figure 2. Instant Contiki log in form*

# 3. The Contiki Directory Structure

The Contiki directory structure is illustrated in Figure 4

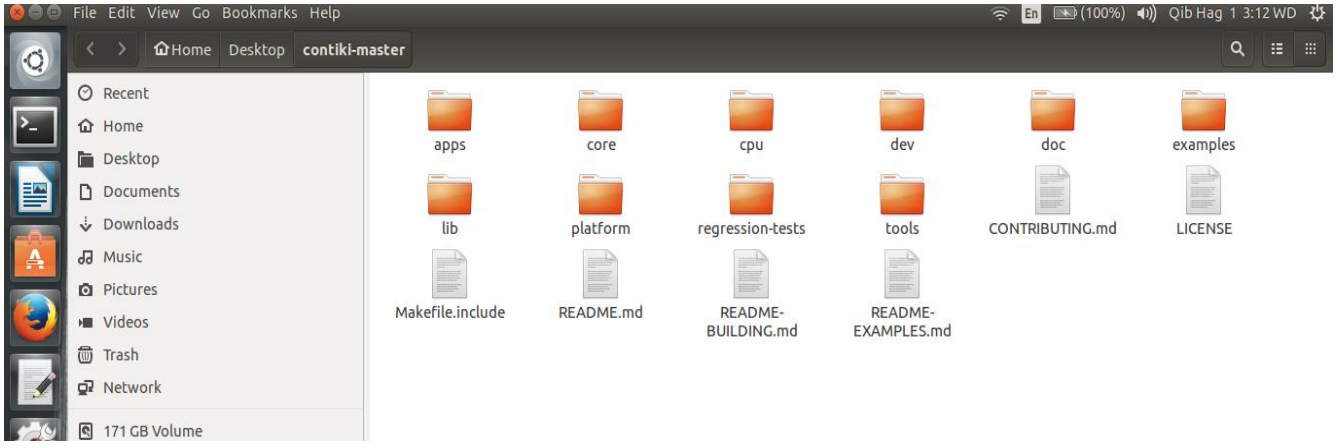



*Figure 3. Directory Structure of Contiki Operating System.*

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 priintf function will be printed in the terminal and if this code is compiled in the Cooja Network Simulator, the string will 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**

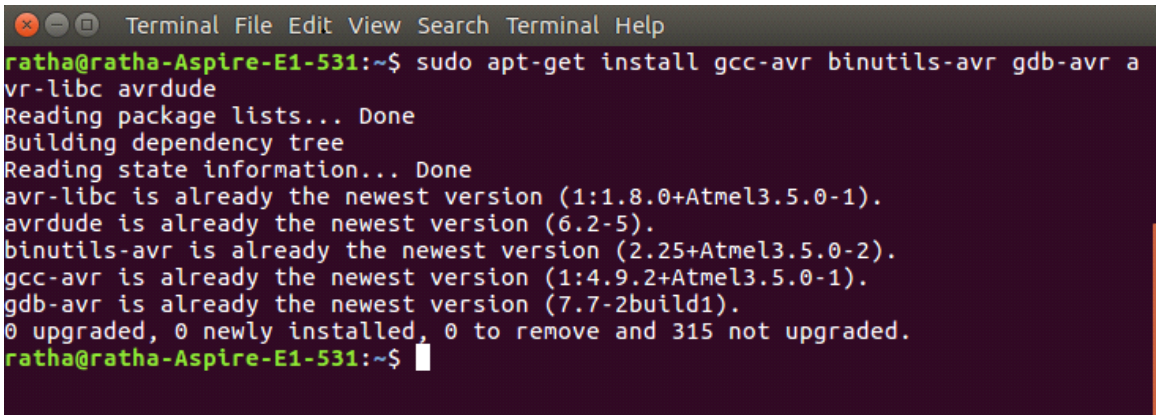



*Figure 7. Installing avr- toolchain*

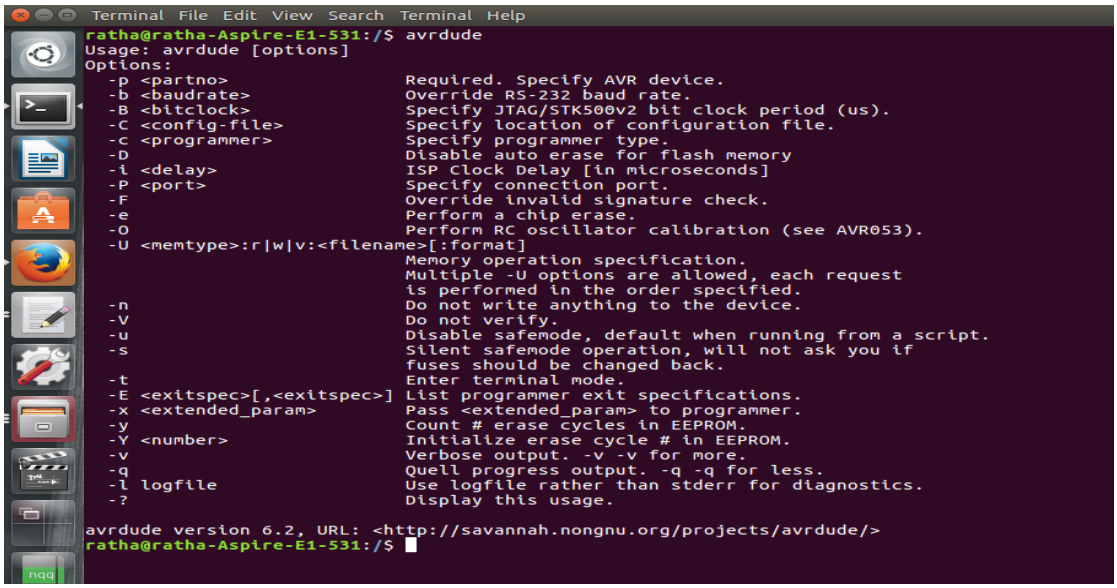



*Figure 8. Confirming that avr-dude is successfully installed*

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.


*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)


*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

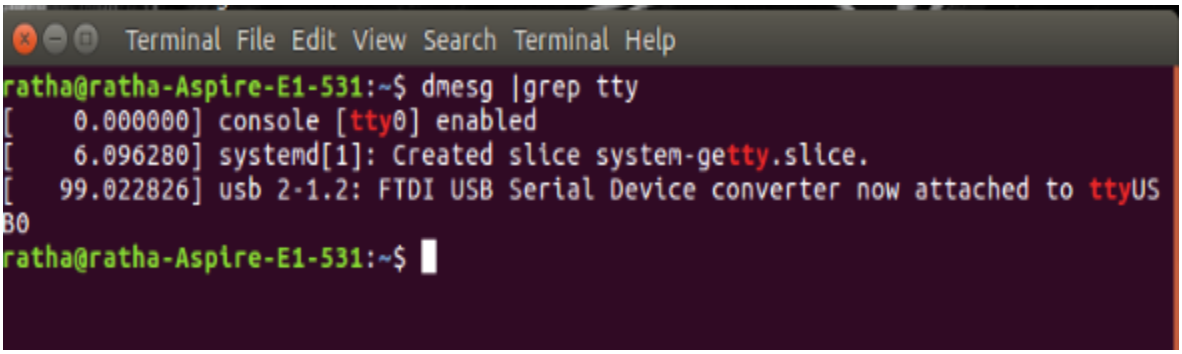
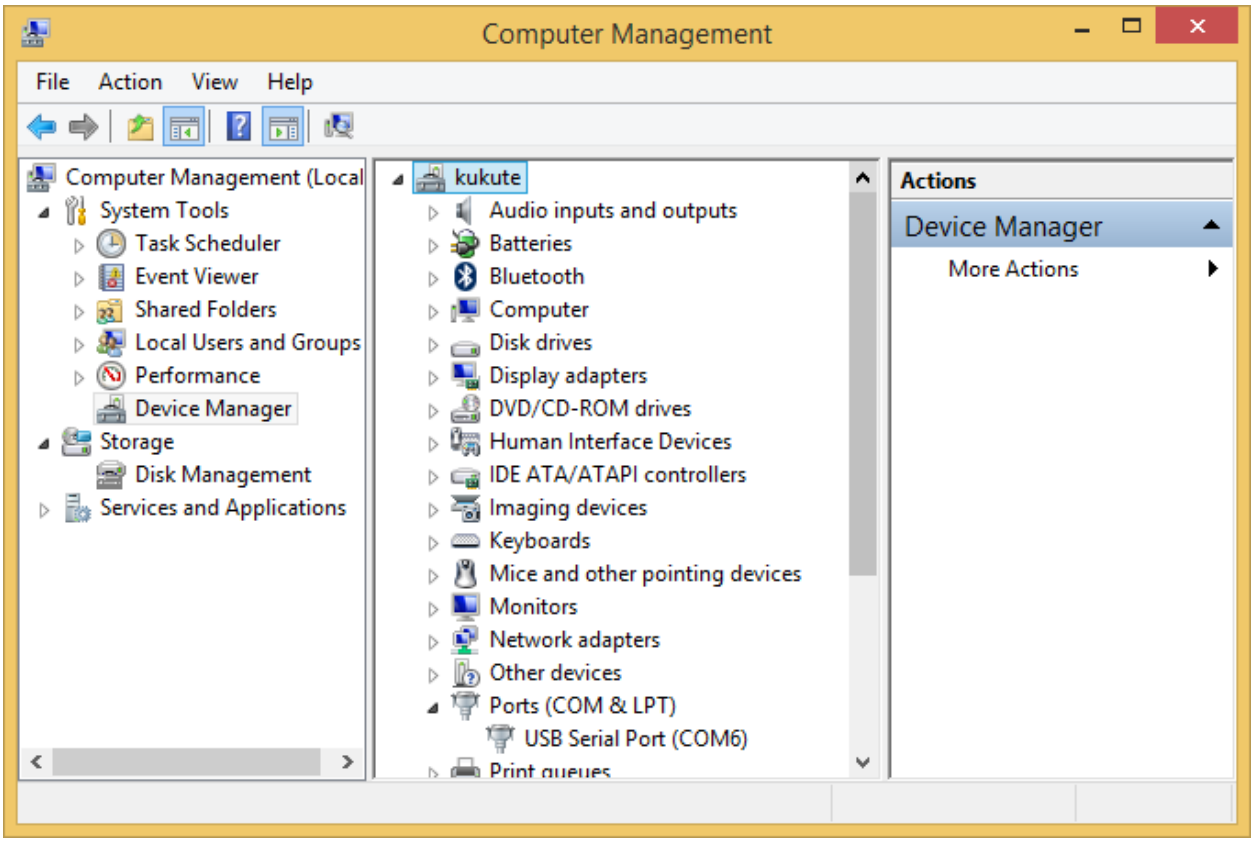



*Figure 11. Checking for serial Port being used*

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*

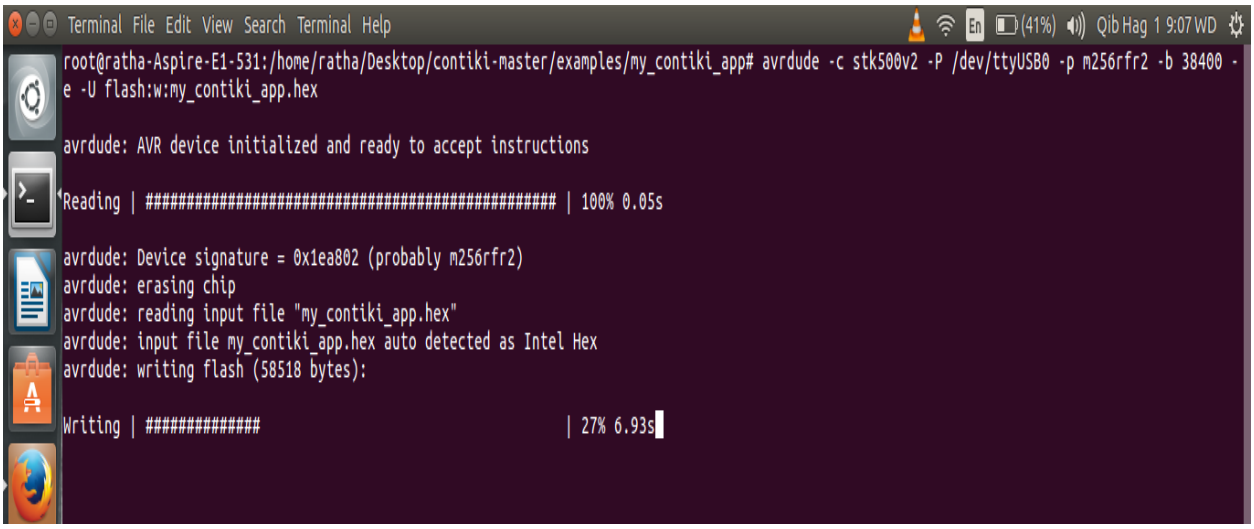



*Figure 12. Downloading the application on the RSS2 node*

Windows users may download and install avrdudess

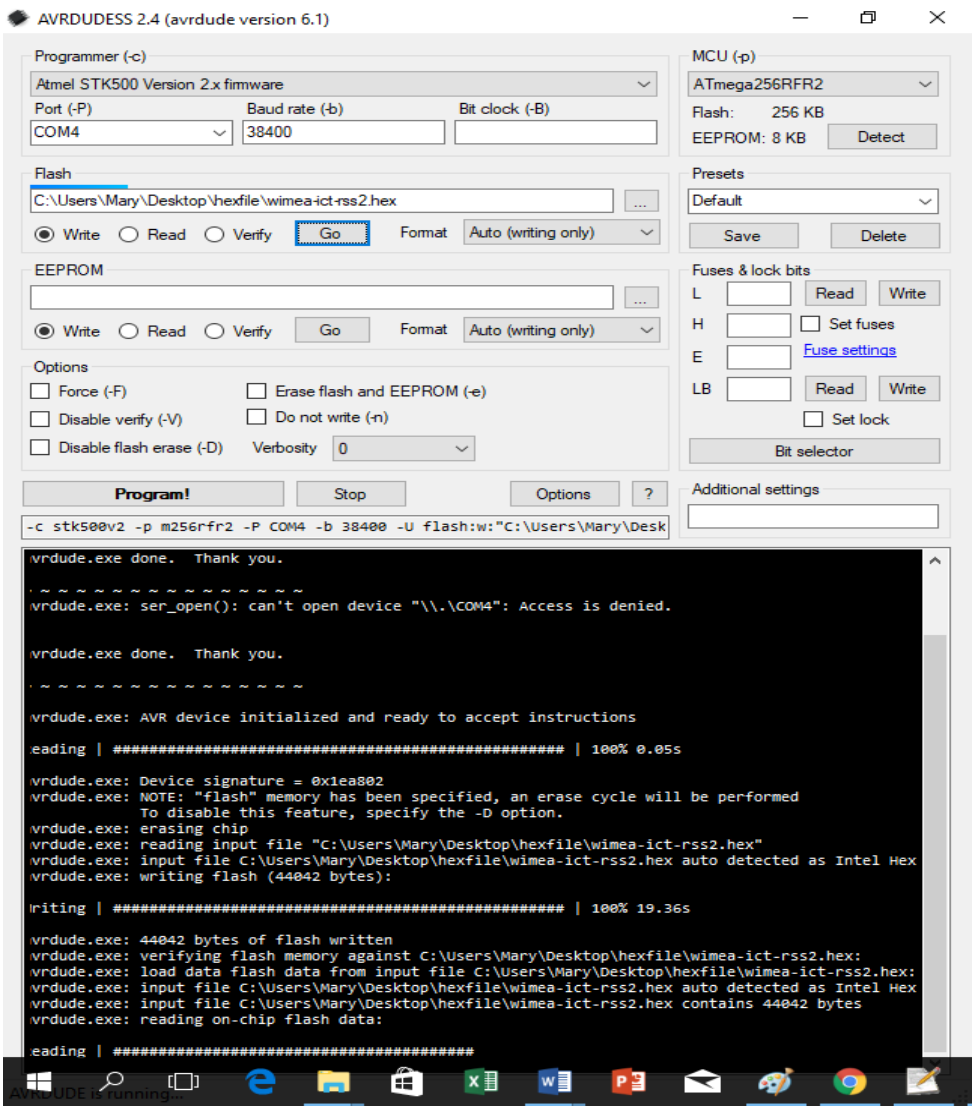



*Figure 13 Avrdudes window , used in windows OS*

# 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

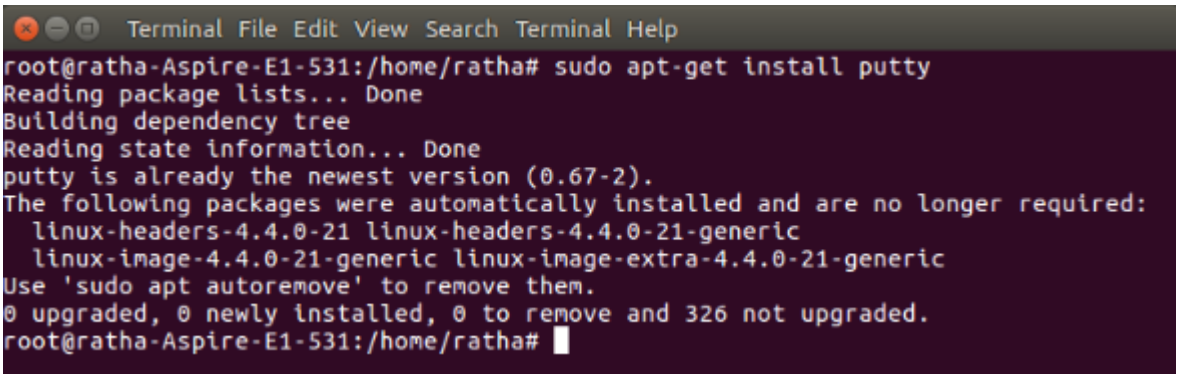



*Figure 14. Installing putty from the terminal*

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.

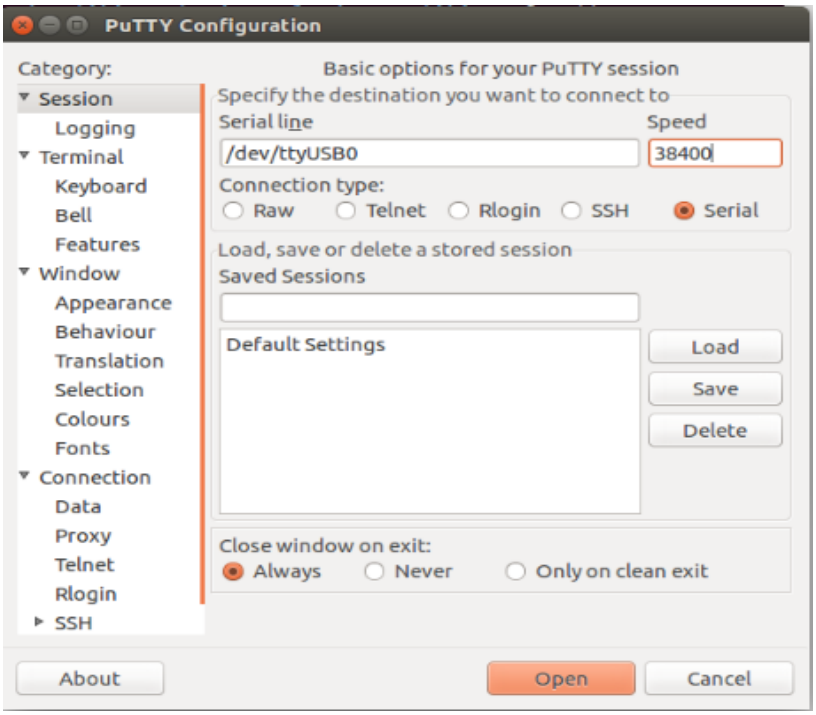



*Figure 15. Putty configuration for serial line and speed*

After pressing "Open" in Figure 15, Figure 16 shall appear


*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"
Staic 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/.

[2] "VMWare" [Online]. Available: www.vmware.com/

[3] "Cygwin" [Online]. Available: www.cygwin.com/

[4] "FTDI driver" [Online]. Available: 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/