Ethernet Access and Remote Control with Intel Galileo

Prepared by

Derrick Anang

For

EECE456 Embedded Systems Design Lab
Instructor: Dr. Charles Kim

Electrical and Computer Engineering
Howard University

Class Web: www.mwftr.com/emblab.html

INTEL® GALILEO

* **Intel® Galileo Gen2 Board**

A microcontroller board based on the Intel® Quark SoC X1000 application processor (a 32-bit Intel® Pentium brand system on a chip). The 32-bit processor can run at up to 400MHz and has a 512KB SRAM built in. This board functions as a fully featured, cost-effective development platform which complements with the Arduino software to provide an advanced compute functionality. It basically serves as an interface between the Arduino Software (IDE) and the Grove system components.

* It comes with an on-board 10/100 Mb/s Ethernet connector port.
* This on-board Ethernet port is accessible via Linux and Arduino IDE using the Ethernet Library.

GETTING STARTED

HARDWARE COMPONENTS

* Intel® Galileo Gen 2 Board
* Micro-SD Card
* USB to 6-pin FTDI Serial Cable
* Ethernet Cable
* Grove Starter Kit
* An Ethernet Switch or Wi-Fi Router with a free Ethernet port (connects Galileo board to the Local Access Network (LAN) being used)
* Micro B to Type A USB Cable
* 12 VDC, 1.5A Power Supply

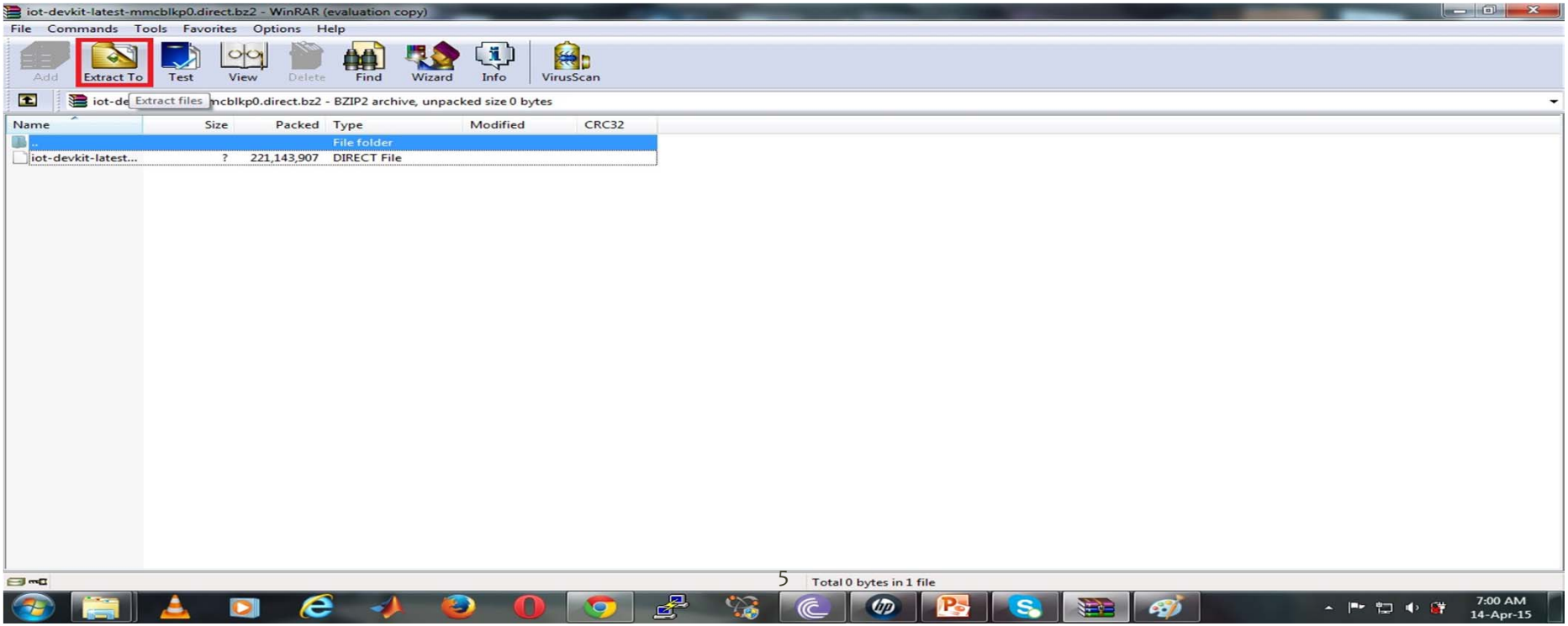
SOFTWARE COMPONENTS

* Arduino Integrated Development Environment (IDE) for Galileo
* Intel XDK IoT Edition (Download link: http://software.intel.com/en-us/html5/xdk-iot)
* Terminal Emulator e.g. PuTTy (http://www.putty.org/) or Bonjour Browser for Mac OS, link:
  (http://www.tildesoft.com/files/BonjourBrowser.dmg)
* FTDI Driver Installer for Windows, link:
  (http://www.ftdichip.com/Drivers/D2XX.htm)
* Yocto Boot Image (http://iotdk.intel.com/images/iot-devkit-latest-mmcblkp0.direct.bz2)
* WinRAR or 7-Zip Extractor (http://www.7-zip.org/)
* Win32 Disk Imager (http://sourceforge.net/projects/win32diskimager/)
* Bonjour
  (http://support.apple.com/downloads/DL999/en_US/BonjourPSSetup.exe)

CONNECTING GALILEO BOARD TO COMPUTER

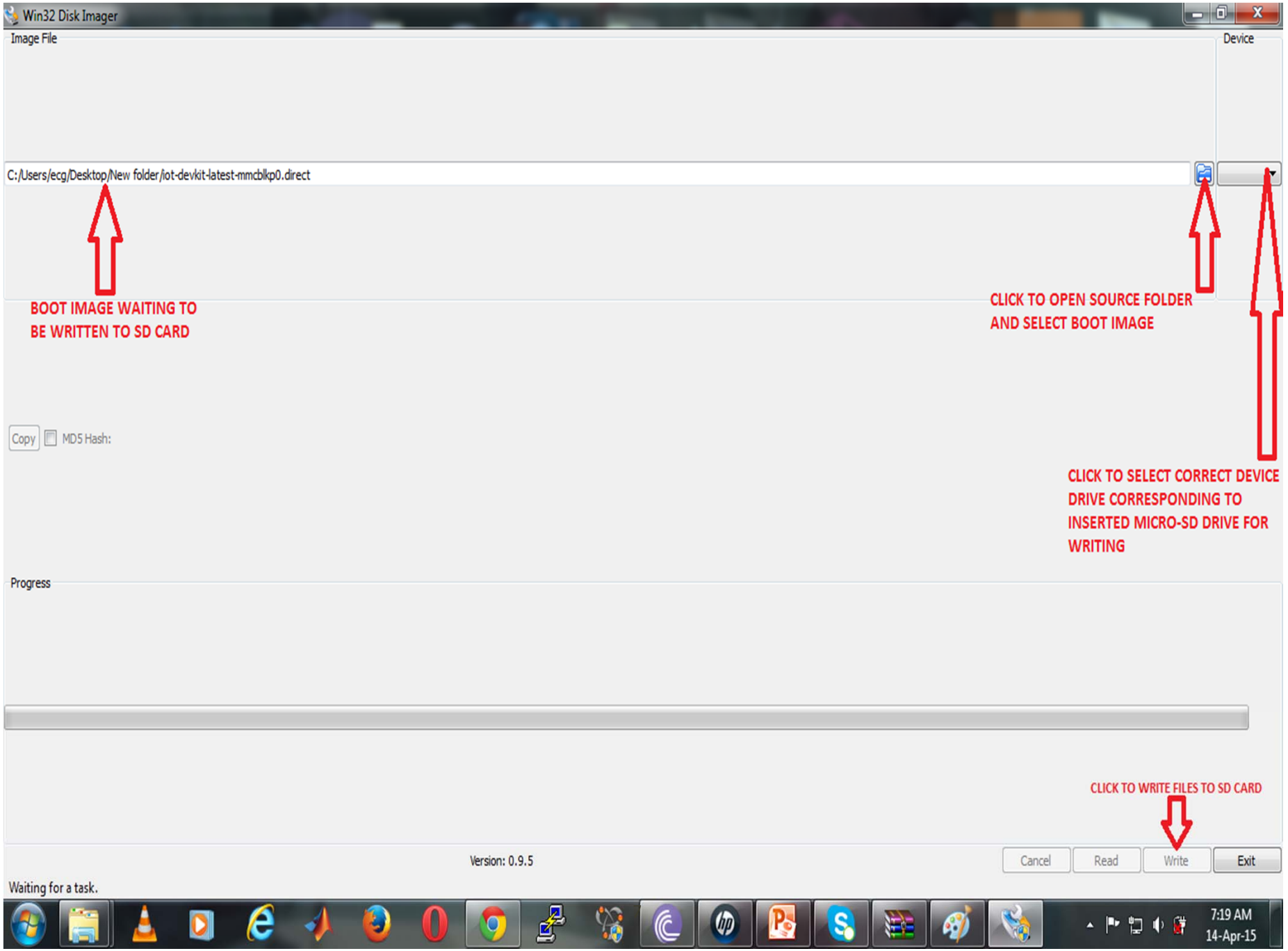
WINDOWS

* Download the compressed Yocto Boot Image from the link given under software components.
* Using WinRAR or 7-Zip Extractor, extract the components of the boot image to a source folder. In the image below, WinRAR Extractor is used. The extract option (in the red box) is used to extract files to a source folder.



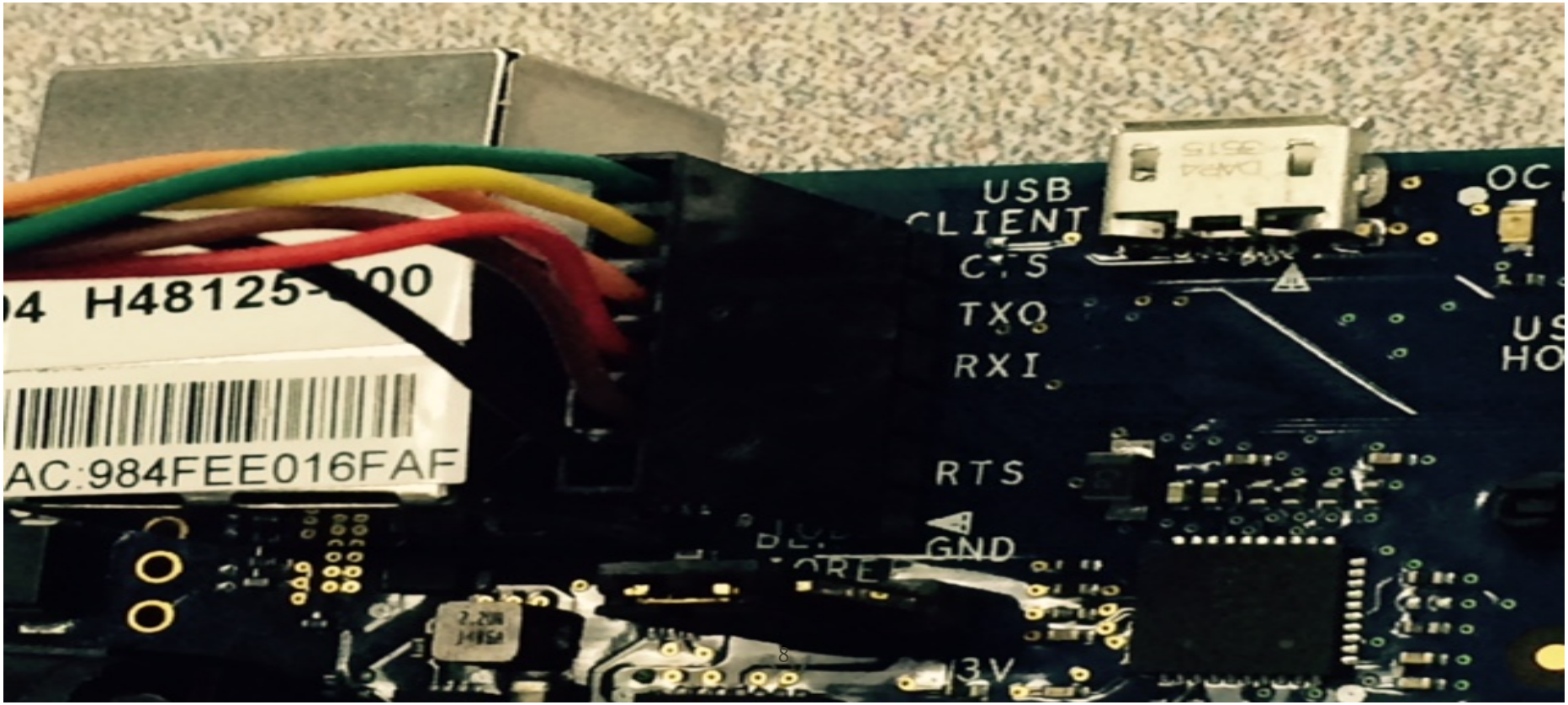
CONNECTING GALILEO BOARD TO COMPUTER

* Insert the micro-SD Card into the appropriate card slot of your Windows System.
* Download the Win32 Disk Imager utility and install it as an Administrator. After a successful installation, run the Win32 Disk Imager as an Administrator.
* Ensure that you have selected the correct Device Drive corresponding to your inserted micro-SD card for writing.
* Under “Image File”, open source folder location where boot image is stored and select the extracted boot image.
* After, click on “Write” to write boot image to micro-SD card.
* Wait thoroughly for write process to be complete before exiting.


Win32 Disk Imager
Image File
Device
C:/Users/ecg/Desktop/New folder/iot-devkit-latest-mmcblkp0.direct
BOOT IMAGE WAITING TO
BE WRITTEN TO SD CARD
CLICK TO OPEN SOURCE FOLDER
AND SELECT BOOT IMAGE
Copy
MD5 Hash:
CLICK TO SELECT CORRECT DEVICE
DRIVE CORRESPONDING TO
INSERTED MICRO-SD DRIVE FOR
WRITING
Progress
CLICK TO WRITE FILES TO SD CARD
Version: 0.9.5
Cancel
Read
Write
Exit
Waiting for a task.
7:19 AM
14-Apr-15

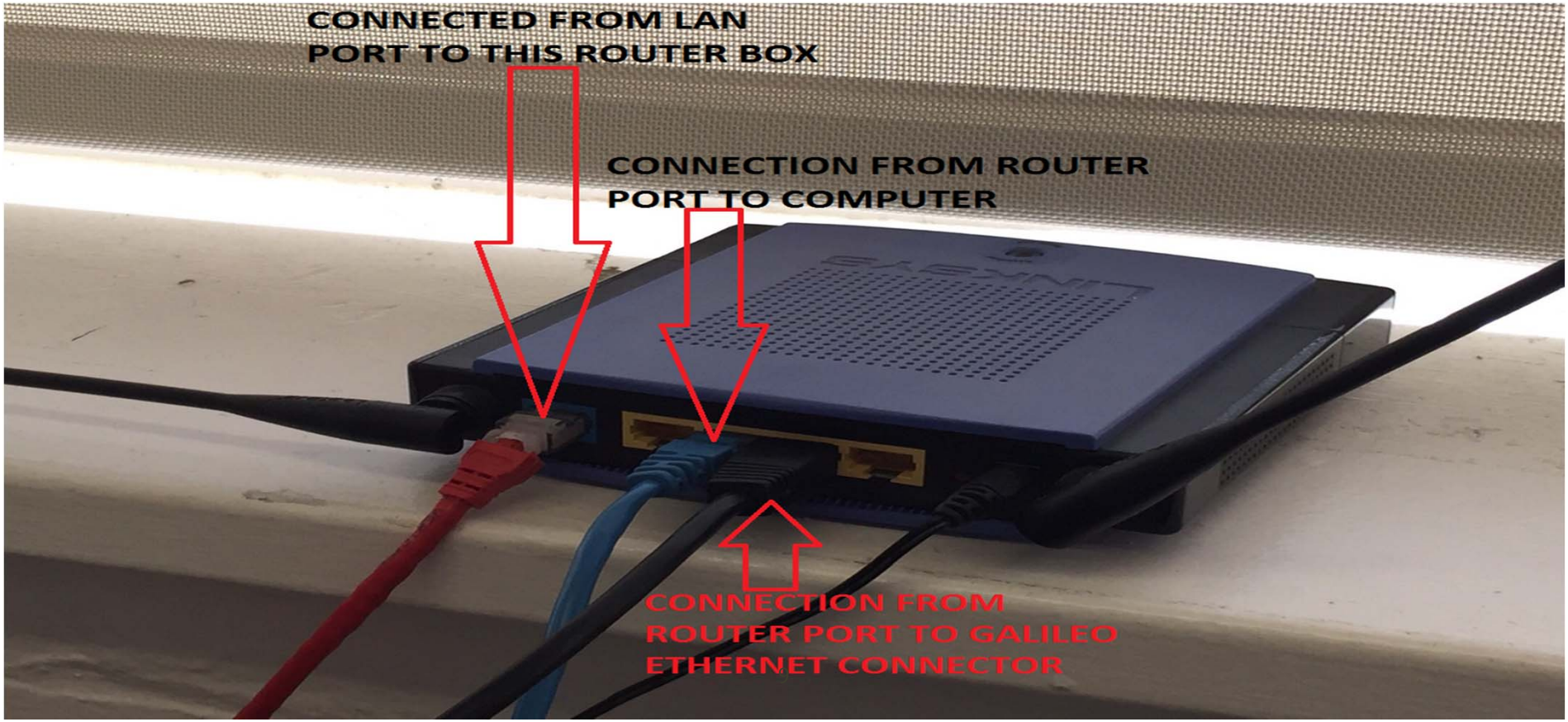
CONNECTING GALILEO BOARD TO COMPUTER

* Make sure the Galileo Board is unplugged.
* Place the prepared micro-SD Card in the micro-SD card slot on the Galileo Board.
* Connect the correspondent FTDI Serial Cable in the appropriate port on the board the connect the other side to the USB port on your computer. Note: the lines on the FTDI cable are color coded. Black is for Ground so make sure it corresponds to the Ground Pin on the FTDI port on the board.



CONNECTING GALILEO BOARD TO COMPUTER

* Connect the Ethernet Cable to the Ethernet connector port on your Galileo Board and connect the other side to your LAN such as a router port.
* Connect your computer to your LAN for example using another router port.

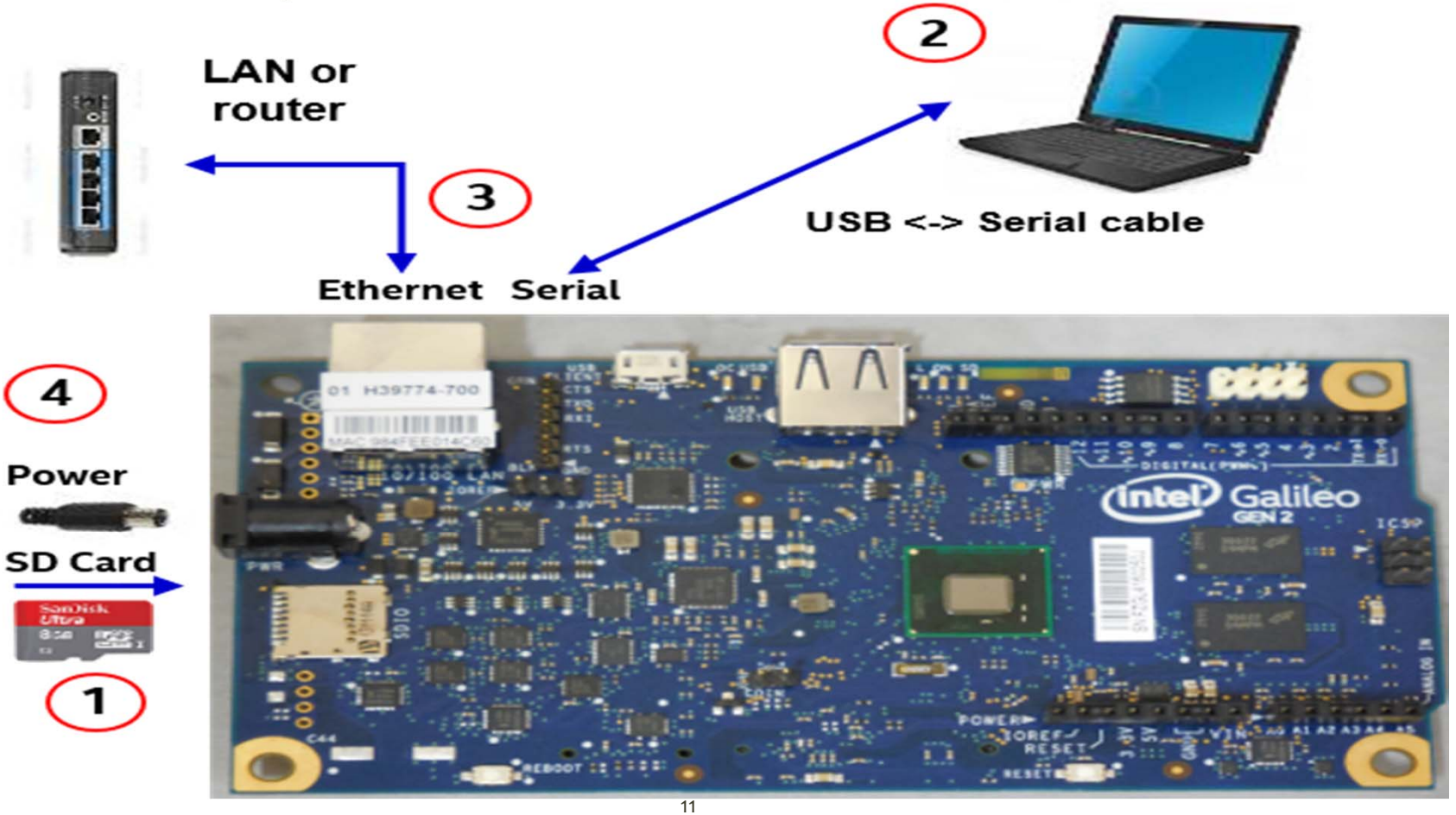


CONNECTING GALILEO BOARD TO COMPUTER

* Notice in the above image that both computer and Galileo share the same LAN subnet which comes from the red Ethernet cable to the router box
* Connect the power cable to the Galileo Board and connect to electrical power to start it up.
* The on-board LED labeled ‘SD’ should blink which indicates there is activity with the micro-SD card. Wait for LED to stop blinking.

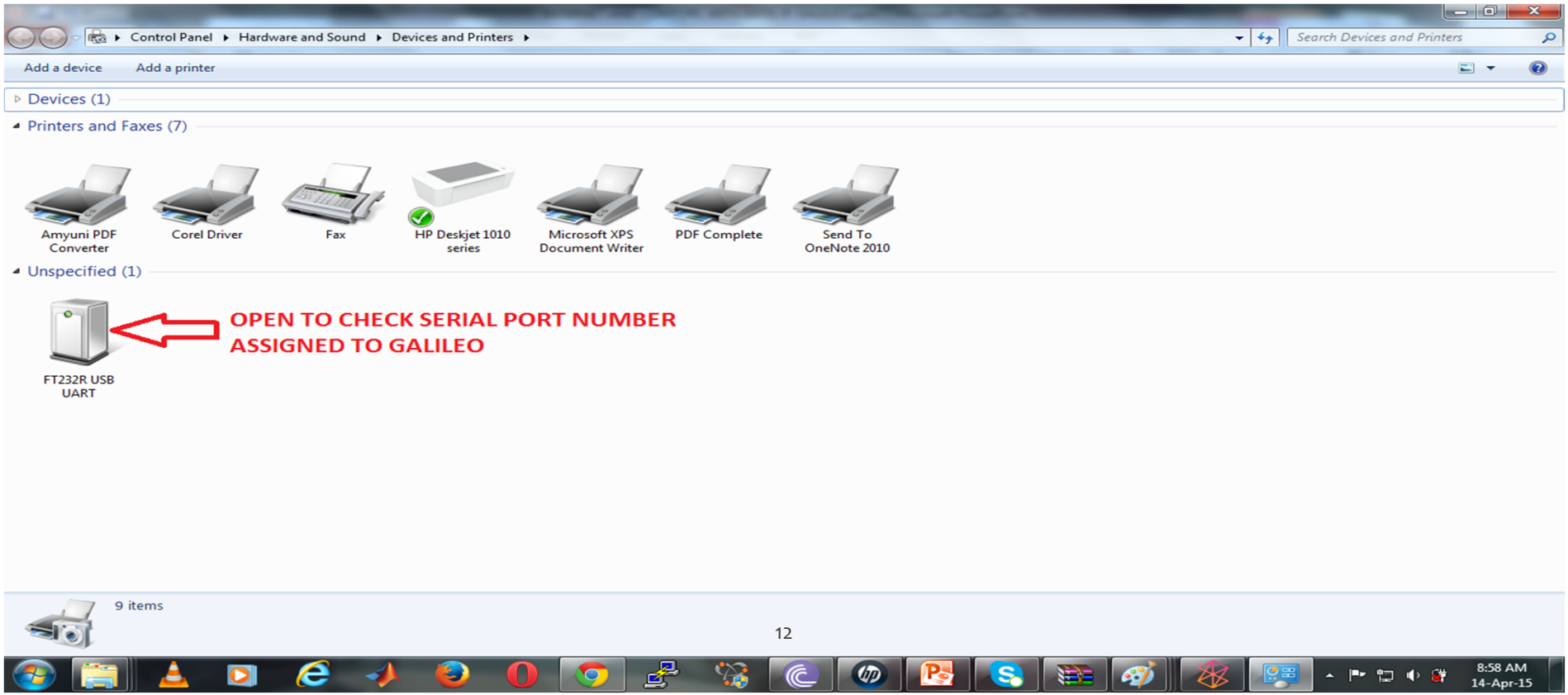
CONNECTING GALILEO BOARD TO COMPUTER

* Full setup should look like this:



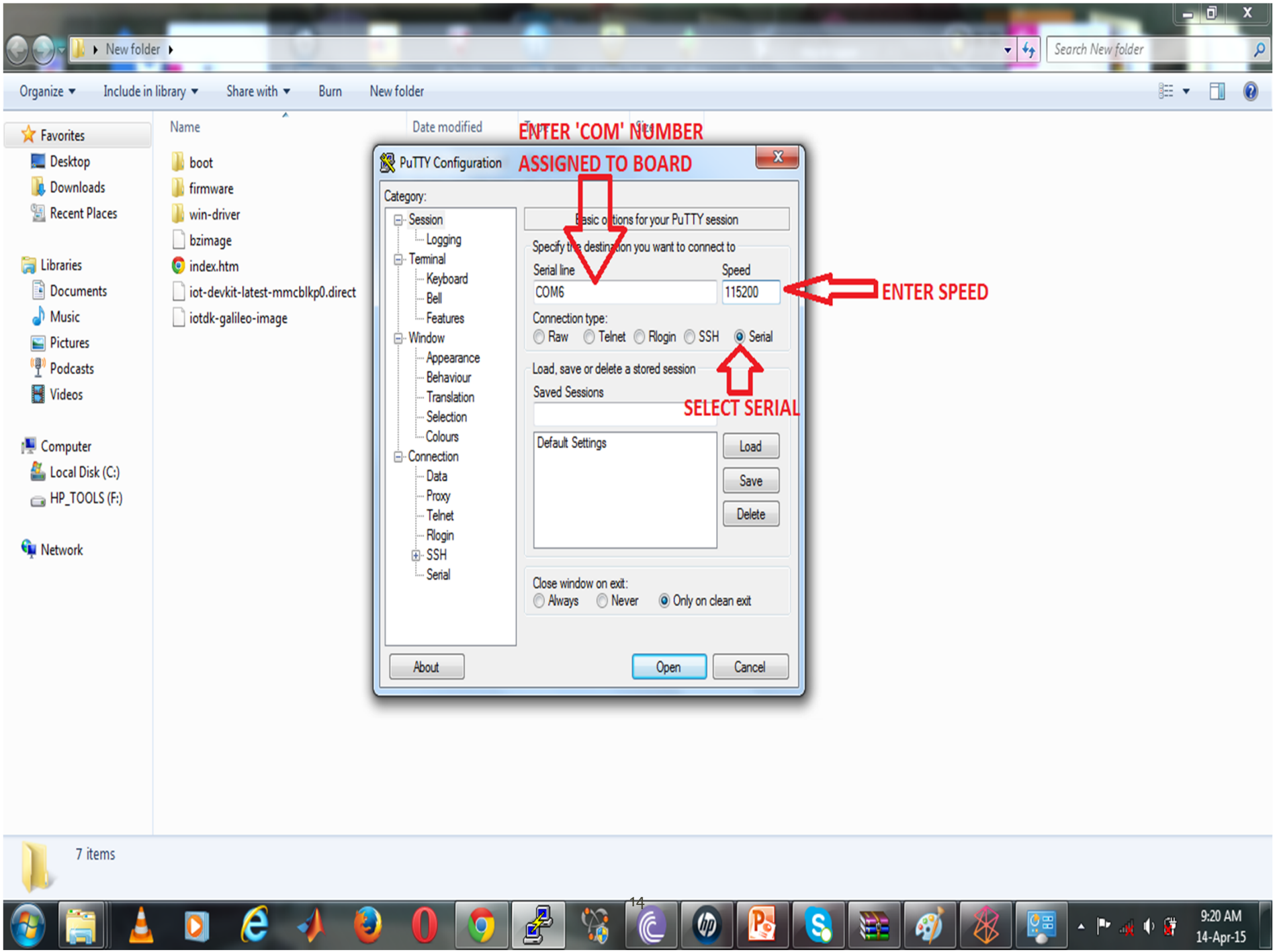
DISCOVERING THE BOARD'S IP ADDRESS

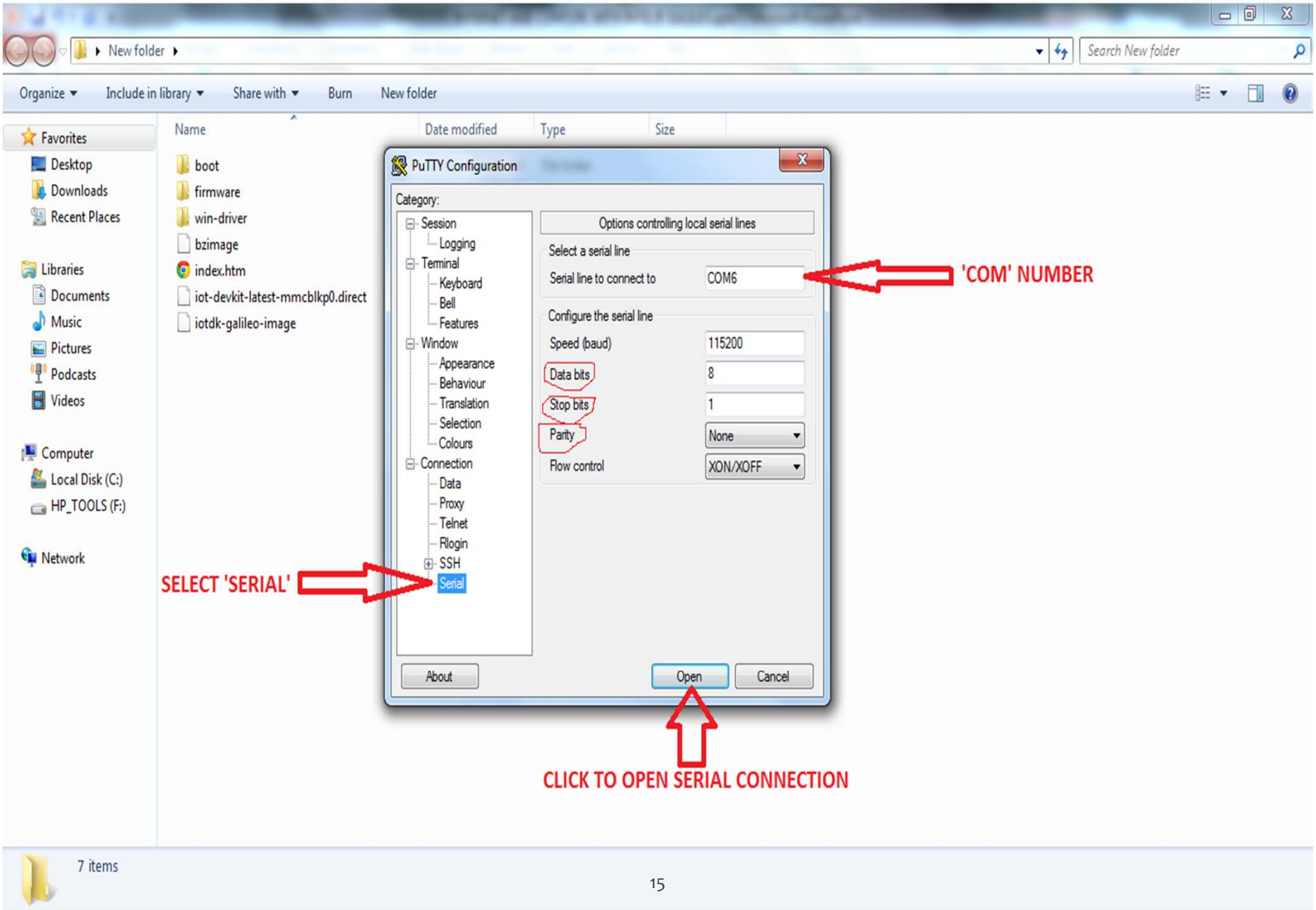
- First, you need to find which serial port your Windows system recognizes as connected to the Galileo board.
- Open 'Start' then click on 'Devices and Printers'. A USB Serial entry displayed by a CPU-like icon appears, double click on it to see the 'COM' number assigned to the board.



DISCOVERING THE BOARD'S IP ADDRESS

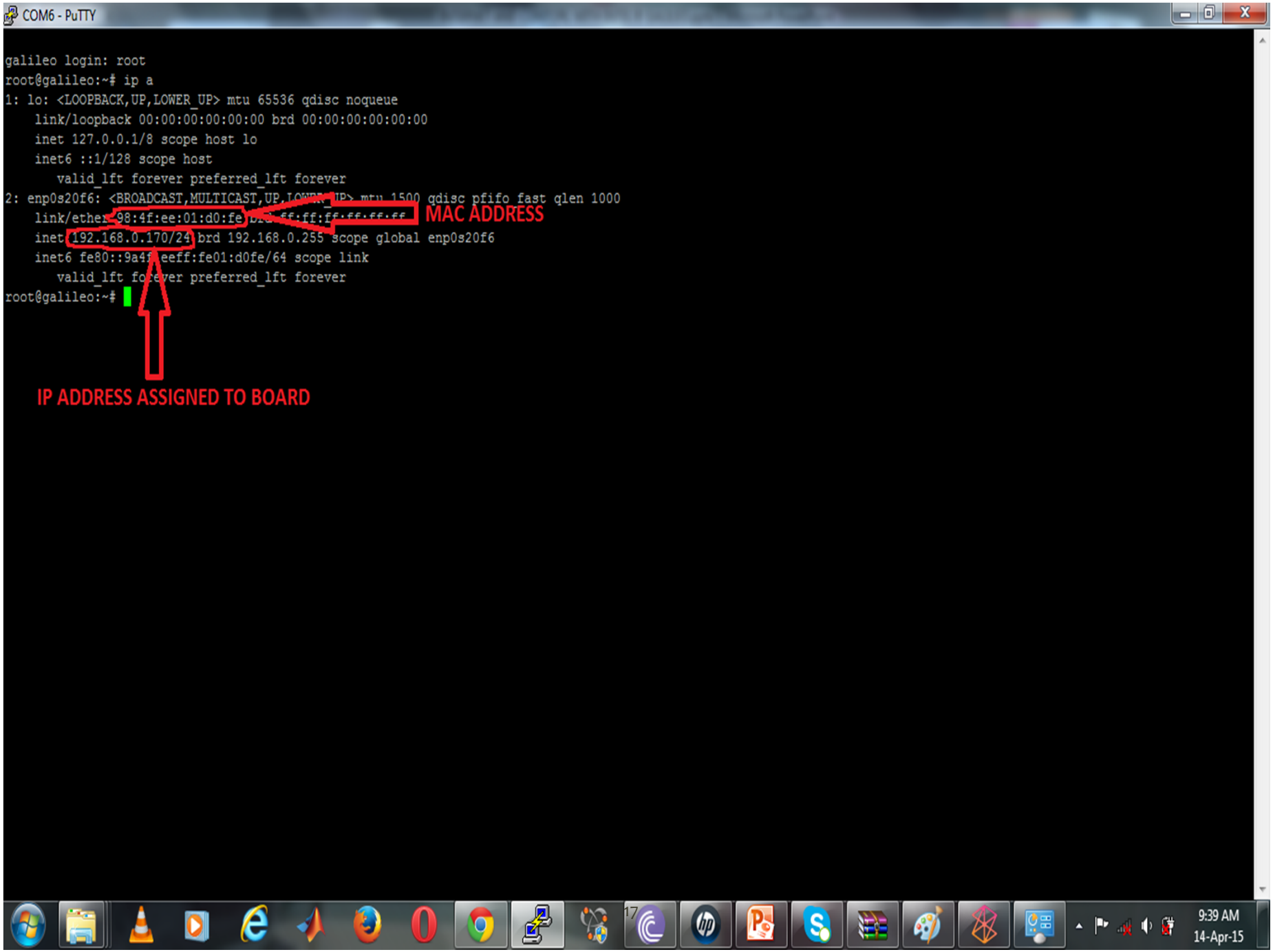
- Run 'PuTTy' software to open a serial connection with the board.
- Set the following configurations:
  - **Under 'Category', choose 'Session' and look for the various options under 'Connection Type'**
  - **Choose 'Serial'**
  - **Under 'Serial Line', type in the 'COM' number assigned to board e.g. COM6**
  - **Under 'Speed' type 115200**
  - **Under 'Category' again, go down and choose 'Serial' which can be found under 'SSH' sub-category.**
  - **Confirm the 'COM' number under 'Serial line to connect to'. Speed = 115200, Data bits = 8 , Stop Bits = 1, Parity = 1**
  - **Then click on 'Open'.**


New folder
Search New folder
Organize
Include in library
Share with
Burn
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Podcasts
Videos
Computer
Local Disk (C:)
HP_TOOLS (F:)
Network
Name
Date modified
boot
firmware
win-driver
bzimage
index.htm
iot-devkit-latest-mmcblkp0.direct
iotdk-galileo-image
ENTER 'COM' NUMBER
ASSIGNED TO BOARD
PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Serial line
Speed
COM6
115200
ENTER SPEED
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
SELECT SERIAL
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel
7 items
9:20 AM
14-Apr-15


New folder
Search New folder
Organize
Include in library
Share with
Burn
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Podcasts
Videos
Computer
Local Disk (C:)
HP_TOOLS (F:)
Network
Name
Date modified
Type
Size
boot
firmware
win-driver
bzimage
index.htm
iot-devkit-latest-mmcblkp0.direct
iotdk-galileo-image
PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Options controlling local serial lines
Select a serial line
Serial line to connect to
COM6
'COM' NUMBER
Configure the serial line
Speed (baud)
115200
Data bits
8
Stop bits
1
Parity
None
Flow control
XON/XOFF
SELECT 'SERIAL'
About
Open
Cancel
CLICK TO OPEN SERIAL CONNECTION
7 items

DISCOVERING THE BOARD'S IP ADDRESS

* If nothing appears when you click 'Open', hit the enter button on your keyboard.
* Type in 'root' as the username. The password is empty so leave it.
* Using the command 'ip a', identify the IP address being used by the board. It should be listed as 'inet' inside the second listed entry.


COM6 - PuTTY
galileo login: root
root@galileo:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
2: enp0s20f6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
link/ether 98:4f:ee:01:d0:fe
inet 192.168.0.170/24 brd 192.168.0.255 scope global enp0s20f6
inet6 fe80::9a4f:eeff:fe01:d0fe/64 scope link
valid_lft forever preferred_lft forever
root@galileo:~#
MAC ADDRESS
IP ADDRESS ASSIGNED TO BOARD
9:39 AM
14-Apr-15

ETHERNET

* Unplug the power source from the Galileo board.
* While the board is off, remove the micro-SD card from the micro-SD socket.
* Also, unplug the Serial FTDI cable from Galileo Board while leaving the Ethernet cable plugged in.
* Connect the Micro B to Type A USB cable to the Galileo Board and then to a USB port on your computer.
* Plug in the power source to turn the Galileo board back on.
* Run the Arduino IDE software. Ensure the 'Port' and 'Board' selections are correct.
* Open an Ethernet sketch such as the one below.

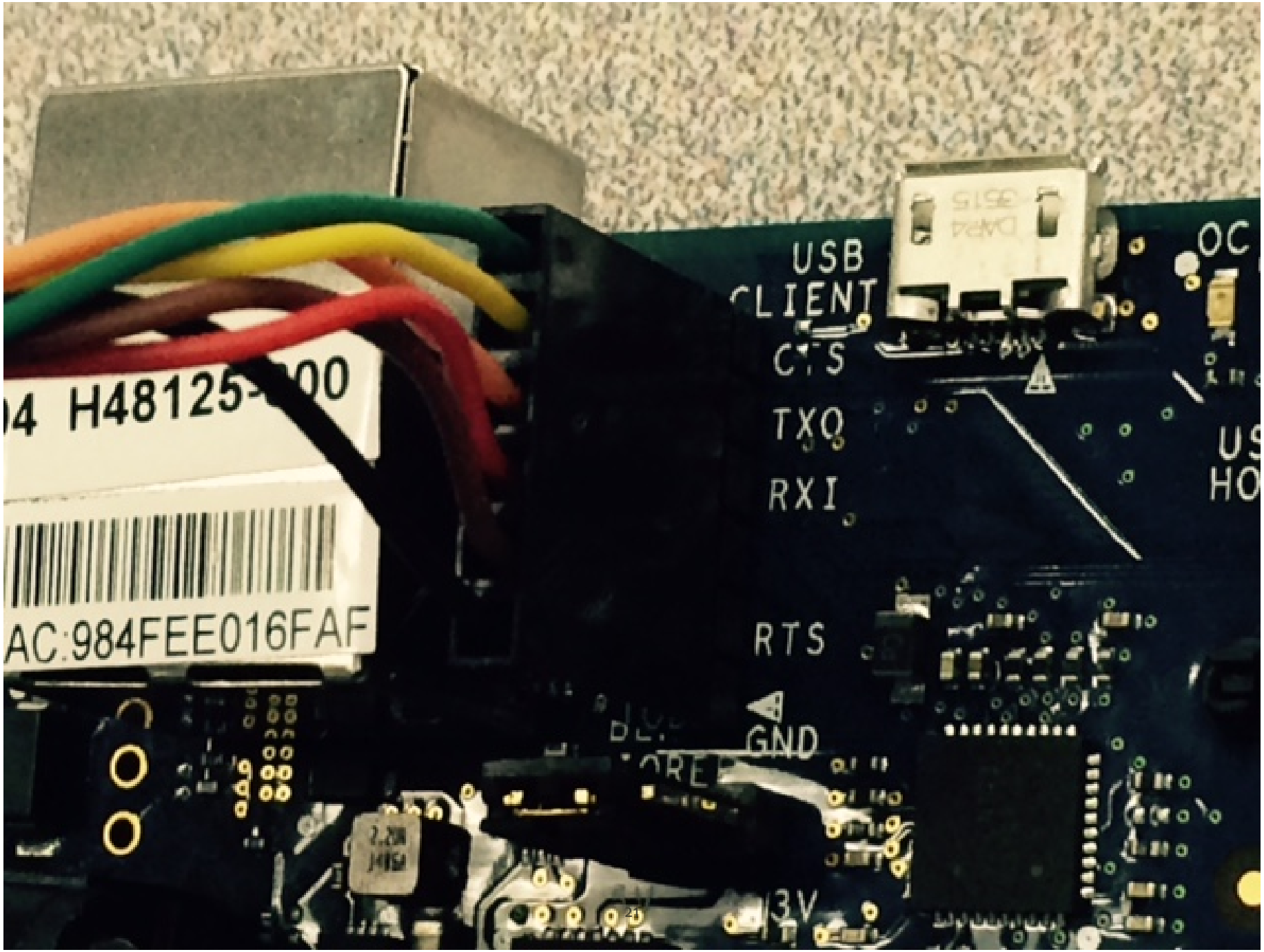
```
#include <SPI.h>
#include <Ethernet.h>
// MAC address for the Galileo (there's a sticker on the Ethernet connector)
byte mac[] = { 0x??, 0x??, 0x??, 0x??, 0x??, 0x?? };
//the IP address for the Galileo: (will be used if there's no DHCP server on your network)
byte ip[] = { ???, ???, ???, ??? };

void setup() {
        delay(5000);
        Serial.println("Attempting to configure Ethernet using DHCP");
        if (Ethernet.begin(mac) == 0) {
                Serial.begin(9600);
                Serial.println("Failed to configure Ethernet using DHCP");
                Serial.println("Attempting to configure Ethernet using Static IP");
                Ethernet.begin(mac, ip);
                Serial.println(" Please check ifconfig");
        } else
                Serial.println("Sounds good");

        system("ifup eth0"); // load Ethernet interface!
}
void loop () {}
```

ENTERING IP AND MAC ADDRESSES

* In above sketch, it can be seen that the columns for entering the IP and MAC addresses are represented with yellow question marks.
* Let us first begin with the MAC address. The MAC address for your Galileo board is printed on a sticker that's on the Ethernet connector on the board. It's a sequence of 12 digits. It should read like this 'MAC: ############'.
* In the above code, the MAC address is specified as a series of hex formatted numbers. Just place the numbers after each "0x" in the code with two numbers from your MAC address.


USB
CLIENT
CTS
TXO
RXI
RTS
GND
3V
AC:984FEE016FAF

* In the above image, the MAC address can be seen on the sticker placed on the Ethernet connector.
* MAC address is 984FEE016FAF.
* Therefore, in the sketch code it will be represented like this:

  byte mac[] = { 0x98, 0x4F, 0xEE, 0x01, 0x6F, 0xAF};

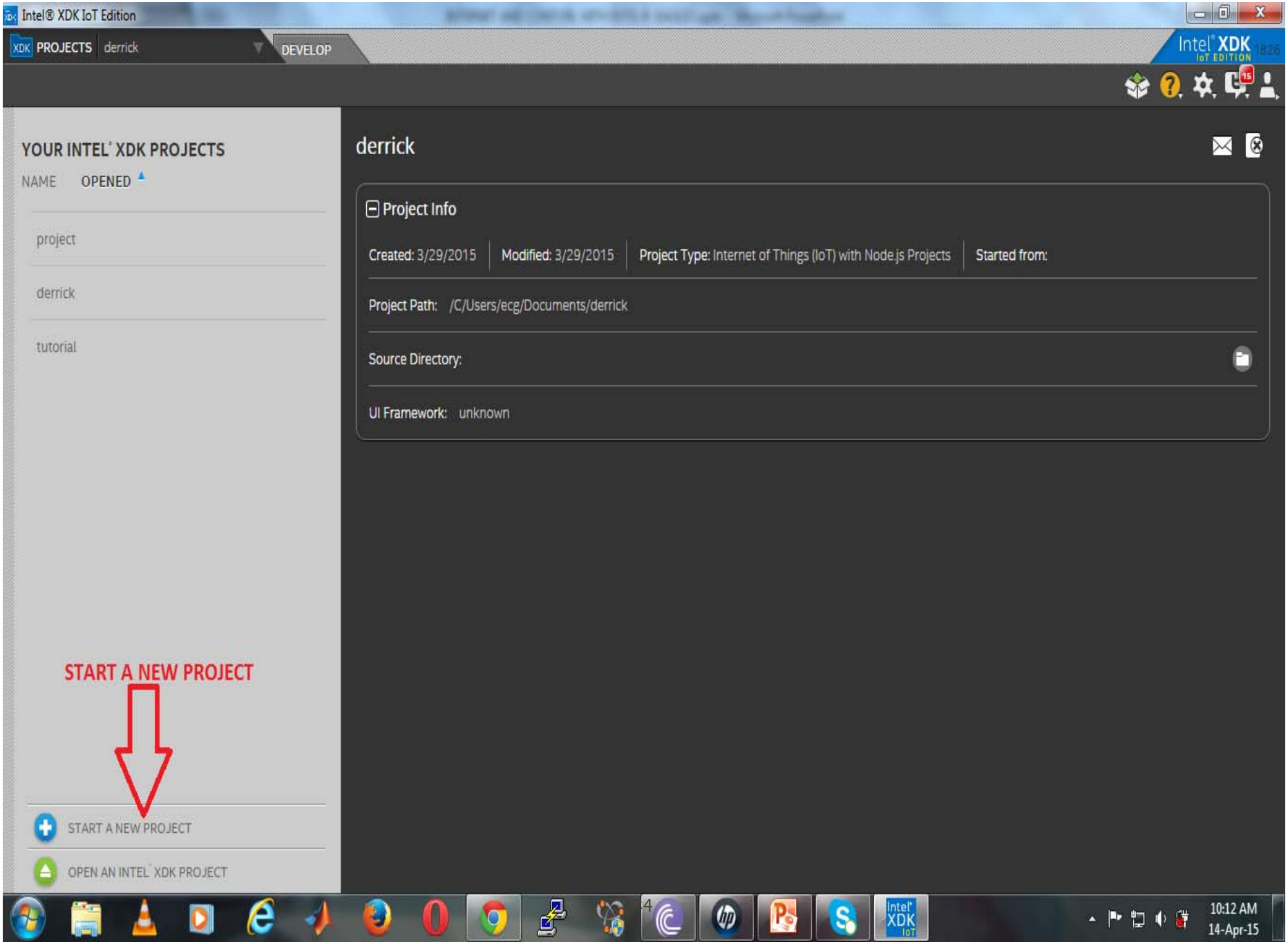
* After the MAC address is configured, you have to configure the IP address using the address you found earlier being used by the board.
* With that address, the IP address input should look like this:

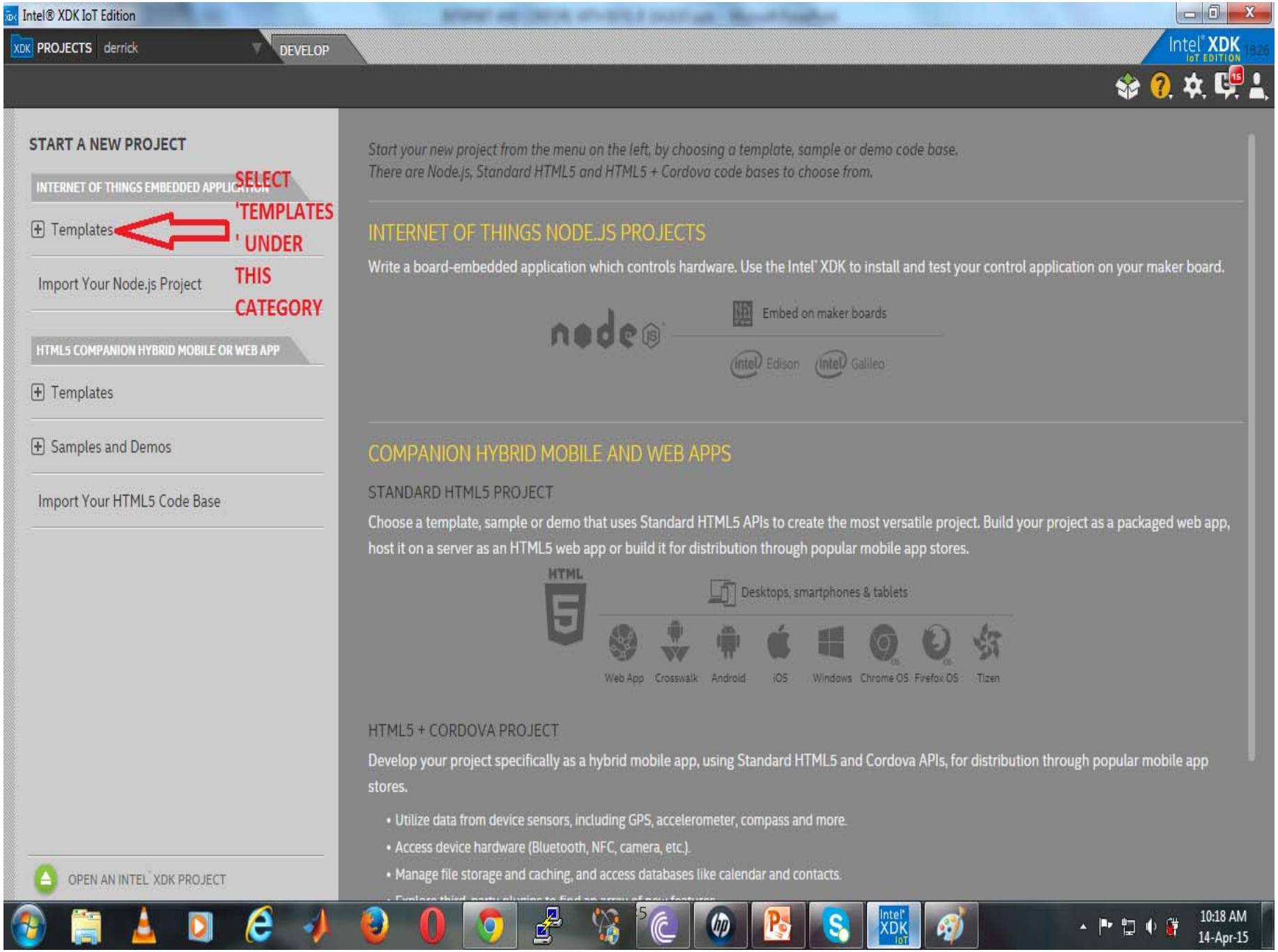
  byte ip [] = { 192, 168, 70, 138}; or

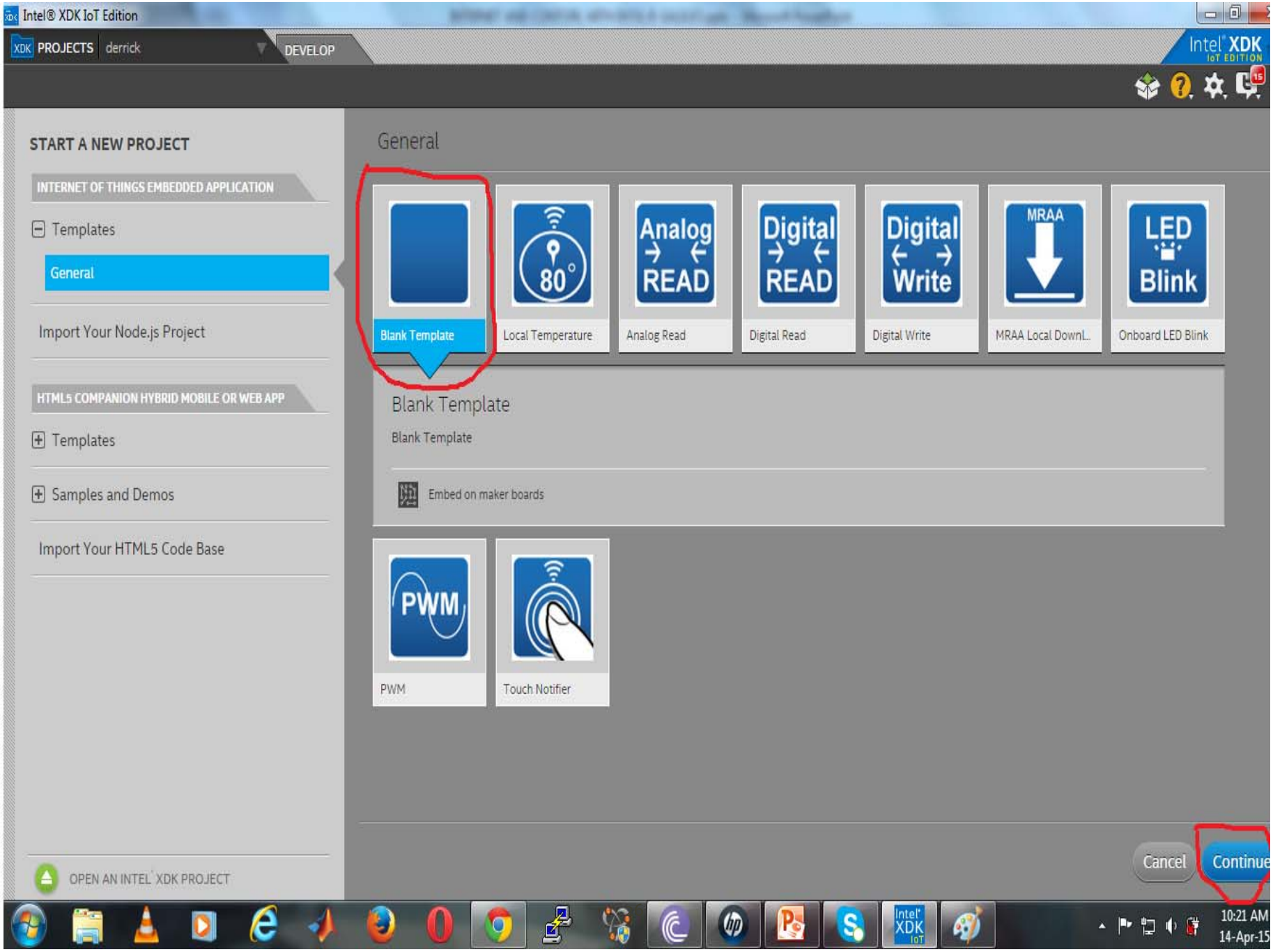
  IPAddress ip(192, 168, 70, 138);
* Your sketch code is then ready to upload.
* Compile and upload the code to the board.
* Using a web browser on your computer, open the assigned IP address using the raw figures without all the 'www...com' business.
* Browser should display whatever content you have in the sketch code.

CONNECTING USING INTEL XDK IoT EDITION

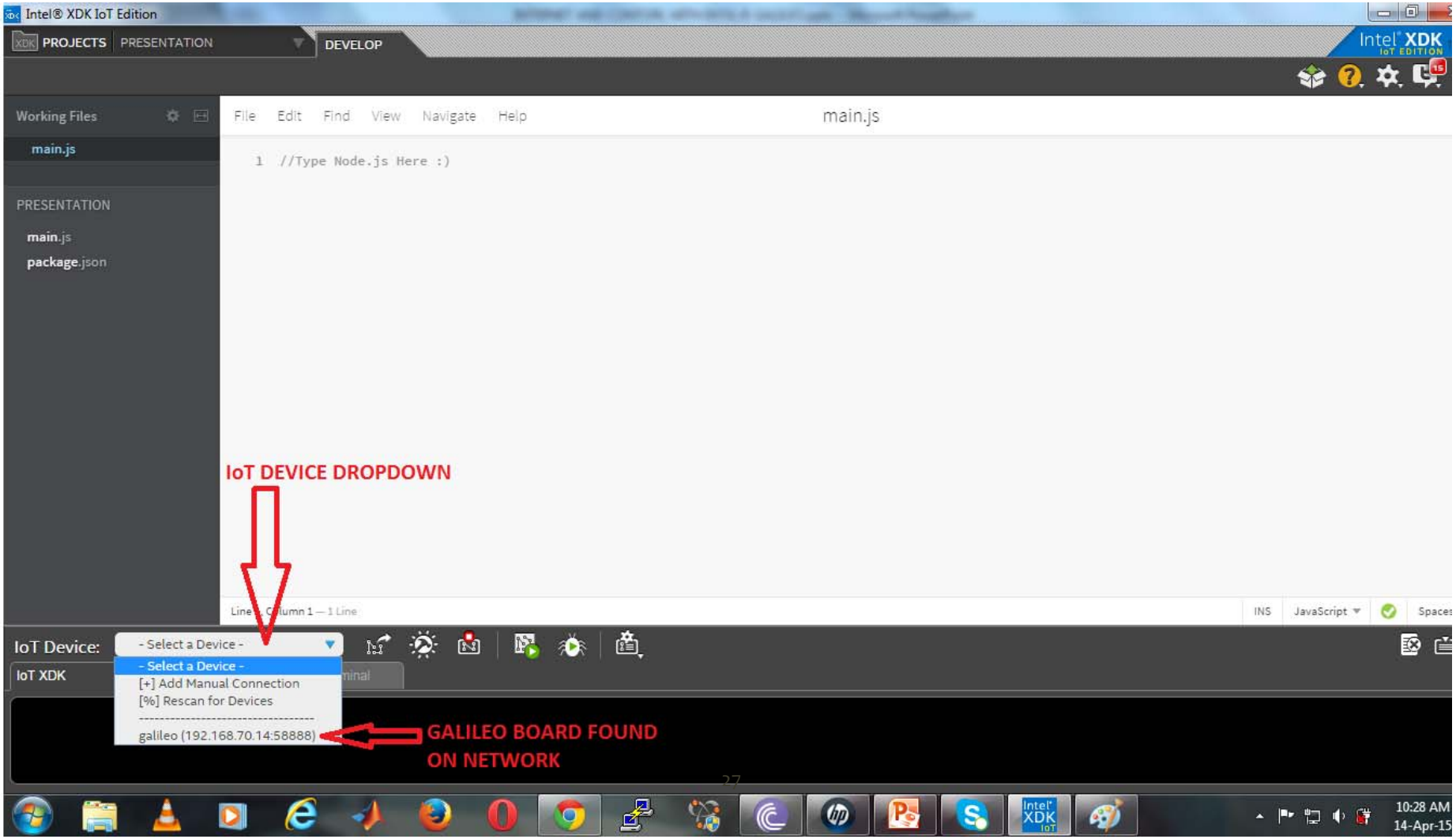
* Unplug power source from Galileo board.
* While off, disconnect the Micro B to Type A USB cable from the board.
* Insert the micro-SD card with the boot image back into micro-SD port.
* Plug in the power source to turn the Galileo board back on.
* Run the 'Intel XDK IoT Edition' software.
* Click on 'Start A New Project'.
* Under 'Internet of Things Embedded Application', click on 'Templates' then choose 'Blank Template'. After, click 'Continue'.
* Choose directory to save your project then assign a project name.
* A pop-up window appears offering to give you a tour. Simply choose 'No, thanks'.


Intel® XDK IoT Edition
PROJECTS
derrick
DEVELOP
Intel XDK IoT EDITION
YOUR INTEL XDK PROJECTS
NAME
OPENED
project
derrick
tutorial
START A NEW PROJECT
START A NEW PROJECT
OPEN AN INTEL XDK PROJECT
derrick
Project Info
Created: 3/29/2015
Modified: 3/29/2015
Project Type: Internet of Things (IoT) with Node.js Projects
Started from:
Project Path: /C/Users/ecg/Documents/derrick
Source Directory:
UI Framework: unknown
10:12 AM
14-Apr-15

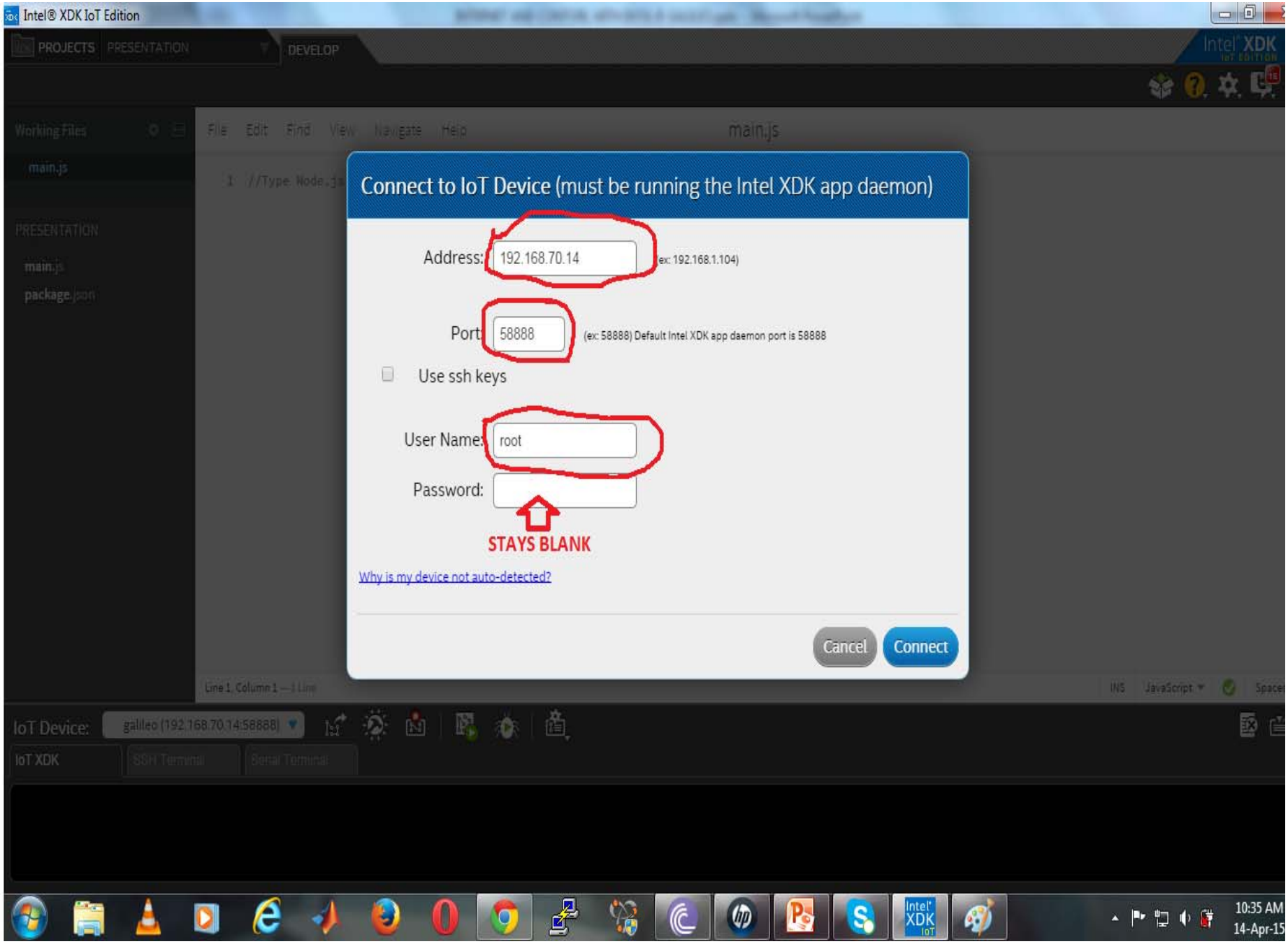

Intel® XDK IoT Edition
PROJECTS
derrick
DEVELOP
Intel XDK IoT EDITION
START A NEW PROJECT
INTERNET OF THINGS EMBEDDED APPLICATION
Templates
SELECT 'TEMPLATES' UNDER THIS CATEGORY
Import Your Node.js Project
HTML5 COMPANION HYBRID MOBILE OR WEB APP
Templates
Samples and Demos
Import Your HTML5 Code Base
OPEN AN INTEL XDK PROJECT
Start your new project from the menu on the left, by choosing a template, sample or demo code base.
There are Node.js, Standard HTML5 and HTML5 + Cordova code bases to choose from.
INTERNET OF THINGS NODE.JS PROJECTS
Write a board-embedded application which controls hardware. Use the Intel XDK to install and test your control application on your maker board.
node
Embed on maker boards
Edison
Galileo
COMPANION HYBRID MOBILE AND WEB APPS
STANDARD HTML5 PROJECT
Choose a template, sample or demo that uses Standard HTML5 APIs to create the most versatile project. Build your project as a packaged web app, host it on a server as an HTML5 web app or build it for distribution through popular mobile app stores.
HTML
Desktops, smartphones & tablets
Web App
Crosswalk
Android
iOS
Windows
Chrome OS
Firefox OS
Tizen
HTML5 + CORDOVA PROJECT
Develop your project specifically as a hybrid mobile app, using Standard HTML5 and Cordova APIs, for distribution through popular mobile app stores.
• Utilize data from device sensors, including GPS, accelerometer, compass and more.
• Access device hardware (Bluetooth, NFC, camera, etc.).
• Manage file storage and caching, and access databases like calendar and contacts.
10:18 AM
14-Apr-15


Intel® XDK IoT Edition
PROJECTS
derrick
DEVELOP
Intel XDK IoT EDITION
START A NEW PROJECT
INTERNET OF THINGS EMBEDDED APPLICATION
Templates
General
Import Your Node.js Project
HTML5 COMPANION HYBRID MOBILE OR WEB APP
Templates
Samples and Demos
Import Your HTML5 Code Base
OPEN AN INTEL XDK PROJECT
General
Blank Template
Local Temperature
Analog Read
Digital Read
Digital Write
MRAA Local Downl..
Onboard LED Blink
Blank Template
Blank Template
Embed on maker boards
PWM
Touch Notifier
Cancel
Continue
10:21 AM
14-Apr-15

* A window like the one below should appear.
* After the board is finished booting, it will be running the Intel XDK app daemon.
* Click on the IoT device dropdown. If you don't see an IP address and a port after a separator, click on 'Rescan for devices' and the IDE will try to find the device on the network.

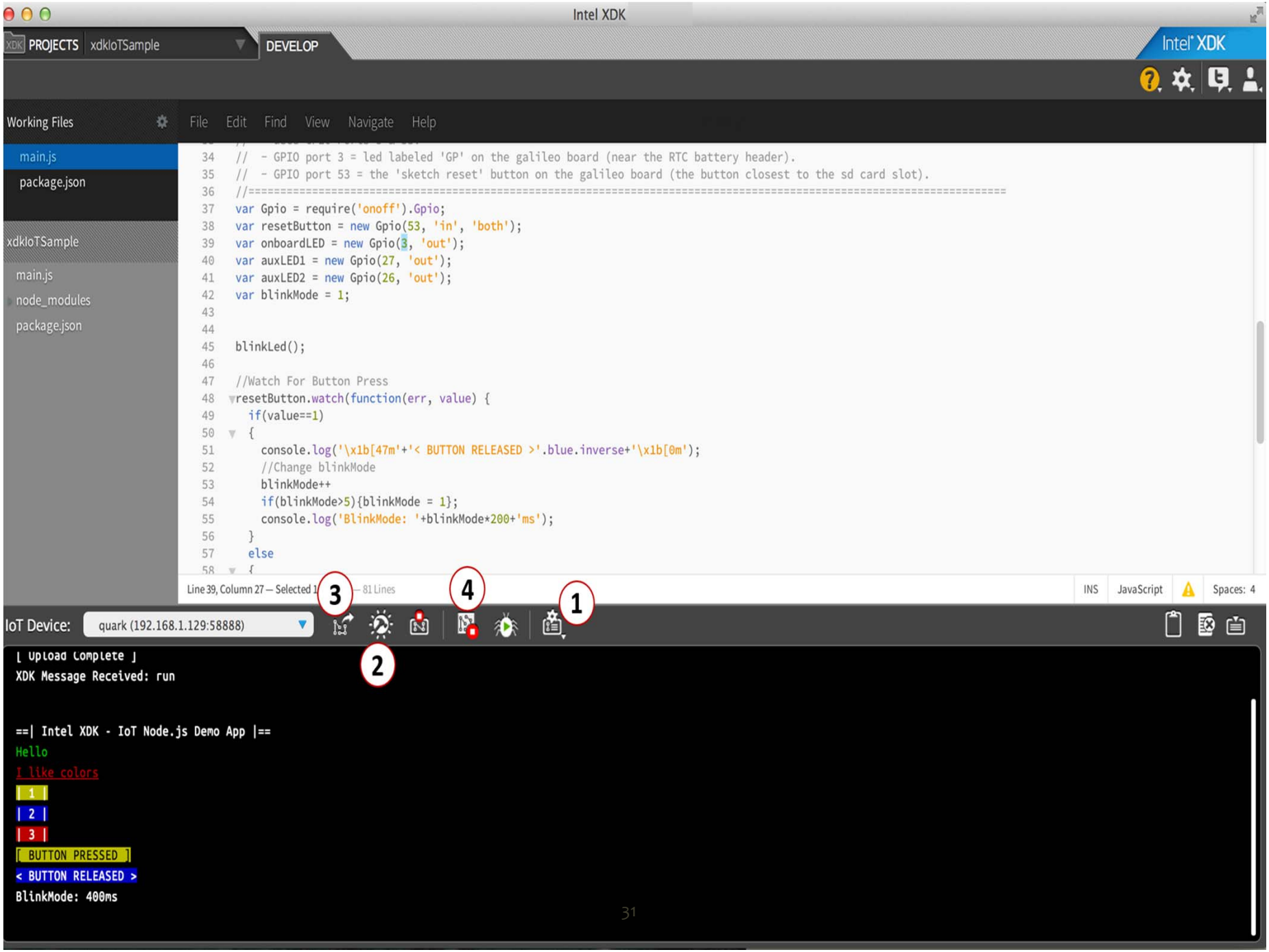


* If the device is still not listed check to make sure that the Galileo board is on the same subnet network.
* Also, download and install Bonjour from the link given under software components. Follow the instructions to automatically detect the compatible devices connected on the LAN.
* If the IDE identifies the device, select it and connect IDE with device. The default port number is 58888. Type in ‘root’ as username. Leave the password column blank then connect with the device.
* You should be alerted when the IDE is done connecting with the device. The board is then ready to communicate with the IDE


Intel® XDK IoT Edition
PROJECTS
PRESENTATION
DEVELOP
Intel XDK
Working Files
main.js
File Edit Find View Navigate Help
main.js
PRESENTATION
main.js
package.json
Connect to IoT Device (must be running the Intel XDK app daemon)
Address: 192.168.70.14
ex: 192.168.1.104)
Port: 58888
(ex: 58888) Default Intel XDK app daemon port is 58888
Use ssh keys
User Name: root
Password:
STAYS BLANK
Why is my device not auto-detected?
Cancel
Connect
IoT Device:
galileo (192.168.70.14:58888)
IoT XDK
SSH Terminal
Serial Terminal
10:35 AM
14-Apr-15

* When device is connected, look for ‘Manage your daemon or IoT device’ icon indicated with the number ‘1’ in below image. Click on ‘Sync PC time with clock on target device’.

* Also under ‘Manage your daemon/IoT device’, activate the following checkboxes below Build Settings.
  - Clean node_modules before building.
  - Run npm install directly on IoT device (requires internet connection).
* Now under the main code area with the inscription ‘Type Node.js here’, copy and paste below project.
* Once the code is uploaded, go back to the IoT device dropdown panel.
* Click on the ‘Install/Build’ icon (Number ‘2’ in image below) and wait for it to finish executing.
* Ignore the ‘require is not defined’, ‘console is not defined’ and ‘setTimeout is not defined’ JSHint problems.
* Now to the left is the ‘Upload’ (Number ‘3’ in image below) icon. Click on it to upload the project to the device.
* After it is uploaded, click on the ‘Run’ (Number ‘4’ in image below) icon to run the uploaded project.
* The onboard LED should blink and also voltage measurements from Analog Pin A1 should display in the activity space under the IoT device dropdown panel.
* The IDE is then communicating with the board.


Intel XDK
PROJECTS
xdkIoTSample
DEVELOP
Intel XDK
Working Files
File
Edit
Find
View
Navigate
Help
main.js
package.json
xdkIoTSample
main.js
node_modules
package.json
34 // - GPIO port 3 = led labeled 'GP' on the galileo board (near the RTC battery header).
35 // - GPIO port 53 = the 'sketch reset' button on the galileo board (the button closest to the sd card slot).
36 //==========
37 var Gpio = require('onoff').Gpio;
38 var resetButton = new Gpio(53, 'in', 'both');
39 var onboardLED = new Gpio(3, 'out');
40 var auxLED1 = new Gpio(27, 'out');
41 var auxLED2 = new Gpio(26, 'out');
42 var blinkMode = 1;
43
44
45 blinkLed();
46
47 //Watch For Button Press
48 resetButton.watch(function(err, value) {
49 if(value==1)
50 {
51 console.log('\x1b[47m'+'< BUTTON RELEASED >'.blue.inverse+'\x1b[0m');
52 //Change blinkMode
53 blinkMode++
54 if(blinkMode>5){blinkMode = 1};
55 console.log('BlinkMode: '+blinkMode*200+'ms');
56 }
57 else
58 {
Line 39, Column 27 — Selected 1
81 Lines
INS
JavaScript
Spaces: 4
IoT Device:
quark (192.168.1.129:58888)
1
2
3
4
[Upload Complete]
XDK Message Received: run
==| Intel XDK - IoT Node.js Demo App |==
Hello
I like colors
[1]
[2]
[3]
[BUTTON PRESSED]
< BUTTON RELEASED >
BlinkMode: 400ms

```
1. // Require the MRAA library
2. var mraa = require('mraa');
3. // Print the MRAA library version to the IDE console
4. console.log('MRAA Library Version: ' + mraa.getVersion());
5. // The Galileo Gen 2 onboard led is mapped to pin #13
6. var onboardLed = new mraa.Gpio(13);
7. // Set the GPIO direction to output (I want to turn on and off the LED)
8. onboardLed.dir(mraa.DIR_OUT);
9. // Hold the LED state
10. var ledState = false;
11. // Analog input pin #1 (A1)
12. var analogPin1 = new mraa.Aio(1);
13. // Call the periodict activity function
14. periodicActivity();
15.
16. function periodicActivity()
17. {
18. // Invert the LED state
19. ledState = !ledState;
20. // Turn on the LED by writing a '1' (high) or
21. // Turn off the LED by writing a '0' based on ledState
22. onboardLed.write(ledState?1:0);
23. // Read the value from the analog pin (up to 4096)
24. var analogValue = analogPin1.read();
25. // Convert the retrieved value to the appropriate voltage value
26. var measuredVoltage = analogValue / 4096 * 5;
27. // Write the measured voltage to the IDE console
28. console.log("Measured voltage (in Volts): " + measuredVoltage);
29. // Call periodicActivity again after 1000 milliseconds (1 second)
30. setTimeout(periodicActivity,1000);
31. }
```