

Open Source Lab:IoT mini Lab

The [Internet of Things](#) is a new concept to us. But if we think about it, Internet access is nothing new. We come across many “things” in day-to-day life. They help make our life easier each day. For example, take an ordinary light bulb. It consumes energy and produces light, which helps us see every day.

Technology and improvements have stripped down resource consumption to the bare minimum. They optimize the output, and now we have an era where the mobile and telecommunications industry are booming. The speed of the Internet is unimaginable compared to the past. From that, we have the idea of making things “smarter” by connecting them to the Internet, analyzing petabytes of historical and real-time data, and automating their operation. This results in a smarter way of living. The Internet of Things affects almost all major areas of the industry: agriculture, health care, home automation, and many more.

It is now easy to control a light on an Arduino without an Ethernet shield, but just over HTTP. The idea is to let you control a single bulb or series of bulbs in your home from the tap of an application on your device.

Ingredients for your homemade light switch

1. [Arduino UNO](#) with USB port
2. [Arduino IDE](#)
3. An Internet connection
4. “Root” access to the development machine
5. [Node.js](#)
6. [Johnny-Five](#) and narf

Arduino UNO is what we will use as the micro-controller for the switch. In this guide, the Arduino board will control a light. To keep things simple, we will use Pin13 of the Arduino for the light source. This is a LED light in the Arduino itself. An Ethernet or WiFi shield is not used.

Setting up the Arduino

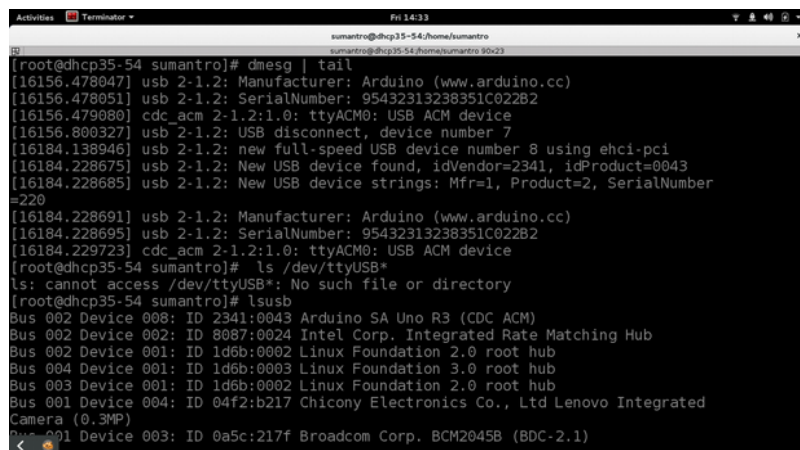
To get started, you will need the Arduino integrated development environment, or IDE. If you are using Fedora, you can install the official Arduino IDE with a single command in a terminal.

```
$ sudo dnf install arduino
```

Once installed, make sure that you plug in your Arduino and check if your system detects it. After inserting the USB into your system, enter the terminal and look for where the system is registering the Arduino. You will need this information later on to execute the code you will make with the Arduino IDE.

The following command should tell you of its place.

```
$ dmesg | tail
```



```
[root@dhcp35-54 sumantro]# dmesg | tail
[16156.478047] usb 2-1.2: Manufacturer: Arduino (www.arduino.cc)
[16156.478051] usb 2-1.2: SerialNumber: 95432313238351C02282
[16156.479080] cdc_acm 2-1.2:1.0: ttyACM0: USB ACM device
[16156.800327] usb 2-1.2: USB disconnect, device number 7
[16184.138946] usb 2-1.2: new full-speed USB device number 8 using ehci-pci
[16184.228675] usb 2-1.2: New USB device found, idVendor=2341, idProduct=0043
[16184.228685] usb 2-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber
=220
[16184.228691] usb 2-1.2: Manufacturer: Arduino (www.arduino.cc)
[16184.228695] usb 2-1.2: SerialNumber: 95432313238351C02282
[16184.229723] cdc_acm 2-1.2:1.0: ttyACM0: USB ACM device
[root@dhcp35-54 sumantro]# ls /dev/ttyUSB*
ls: cannot access /dev/ttyUSB*: No such file or directory
[root@dhcp35-54 sumantro]# lsusb
Bus 002 Device 008: ID 2341:0043 Arduino SA Uno R3 (CDC ACM)
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 004: ID 04f2:b217 Chicony Electronics Co., Ltd Lenovo Integrated
Camera (0.3MP)
Bus 001 Device 003: ID 0a5c:217f Broadcom Corp. BCM2045B (BDC-2.1)
```

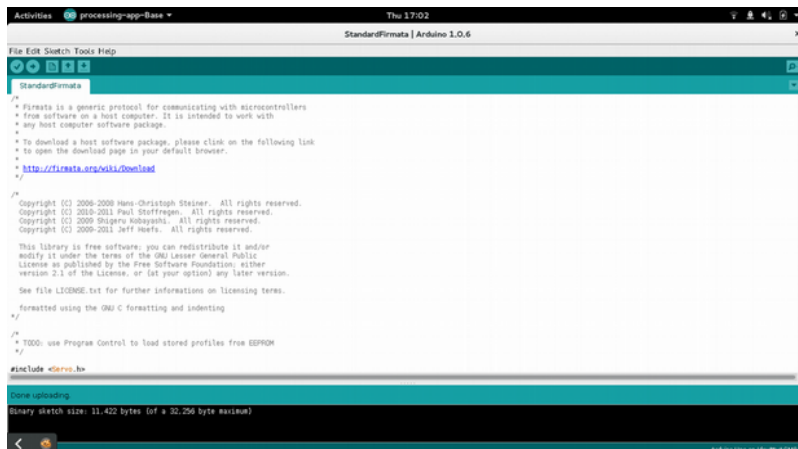
Look for a bus device and Arduino in the same line. In my screenshot, the line I needed was Bus 002 Device: ID Arduino SA Uno R3. Once you find the board, we can move ahead to setting up the communication protocol.

Setting up Johnny-Five

Johnny-Five is the JavaScript Robotics & IoT platform. Released by Bocoup in 2012, Johnny-Five is maintained by a community of passionate software developers and hardware engineers. Over 75 developers have made contributions towards building a robust, extensible, and composable ecosystem. In this set-up, we will also be using **Firmata**. The Firmata library implements the Firmata protocol for communicating with software on the host computer. This allows you to write custom firmware without having to create your own protocol and objects for the programming environment that you use.

To install and set up Johnny-Five, open the Arduino IDE we installed in the previous step (if you're using GNOME, it should be in the Applications menu). In the IDE, go to *File > Examples > Firmata > StandardFirmata*. We're going to upload the StandardFirmata to the Arduino board for us to use when creating the switch. StandardFirmata is available in all versions of Firmata greater than v2.5.0.

Once you find this in the Arduino IDE, hit the "Upload" button to push the firmware to the Arduino board. If the upload was successful, the board is prepared for us to use, and you can now close the Arduino IDE.



Set up project workspace

You will need to create and set up a project workspace for creating the Arduino application. For our project, we will be using Node.js as the language for creating the switch. There are several ways to create this kind of application, but to help get you started, I created an HTML page and the JavaScript file you can use for your own set up.

You can find my demo code available on GitHub. For this project, you will want a copy of the index.html and LED_Server.js files. You can copy and paste the two files into the project workspace you created earlier.

Setting up node.js

Now that we have our workspace and files needed for running the project, we will need to set up a Node.js server to run the application. To begin with running the “light switch server”, you will need to install Node.js and NPM, the package manager for Node.js applications.

Enter the following commands to install the necessary dependencies.

```
$ sudo dnf install npm nodejs
$ npm install narf johnny-five
```

Once all the dependencies are installed, you will now be able to start your light switch server. Making sure you are in the project workspace folder, enter the following command to start the Node.js server.

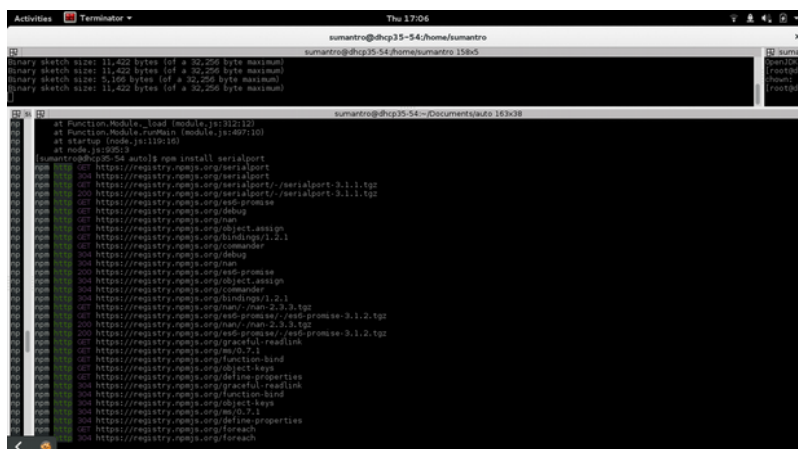
```
$ node LED_Server.js
```

Once you execute the command, your terminal window should look like the following.

A terminal window titled 'Terminator' showing the execution of 'node LED_Server.js'. The output includes a 'Board closing' message, a 'Connected' status, and a list of methods: startServer, configure, getConnectedClients, setDebug, and setServer. The terminal also shows some system messages like 'Binary sketch size: 11,422 bytes'.

If you receive an error about a serial port not found, you may need another dependency (depending on your environment). To resolve this, run the following command to install serialport via npm.

```
$ npm install serialport
```

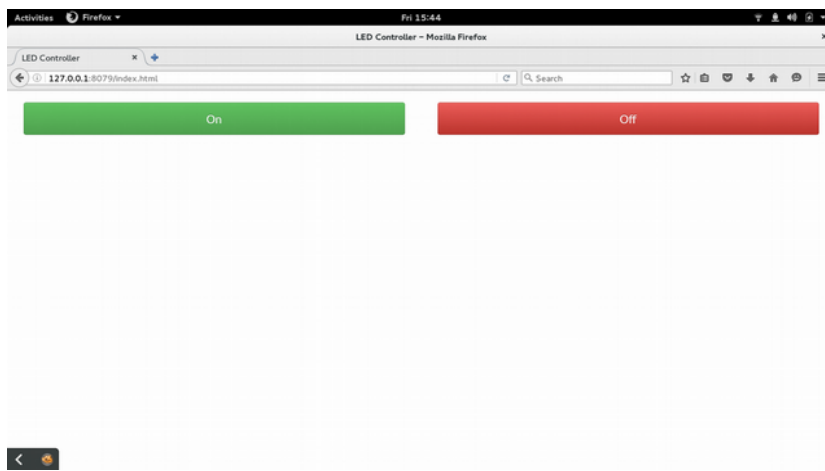
A terminal window titled 'Terminator' showing the installation of 'serialport' via npm. The output includes a list of methods: startServer, configure, getConnectedClients, setDebug, and setServer. The terminal also shows some system messages like 'Binary sketch size: 11,422 bytes'.

The application will now be running. To test if it's working, open up your favorite browser and point it at <http://127.0.0.1:8079/index.html>. This is the address of a local page on your system where you can view the virtual power switch we created.

Controlling the Arduino light

Now, you can control the power for Pin13 on your Arduino board from this webpage. In this proof of concept, you will only be able to control the LED light on the Arduino. However, in a more realistic example, perhaps you leave for a vacation and can't remember if you turned off the lamp next to your bed while you were packing. This solution would allow you power off the lamp from anywhere in the world at any time.

Here is how the web page looks in action from the above example.



In root folder(/root same folder where we install johnny five and other SW),create index.html and LED_Server.js fi
paste code in it.

Make sure that you have connected uno to pc and it is detected.
Code for index.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
  <head>
```

```
    <meta charset="utf-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Simple LED :FAMT@Open Source Lab</title>
```

```
  <style>
```

```
    body {
```

```

width: 100%;

float: left;

margin: 0;

padding: 0;
}

button.btn-success {
    color: #ffffff;
    text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25);
    background-color: #5bb75b;
    background-image: -moz-linear-gradient(top, #62c462, #51a351);
    background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#62c462),
to(#51a351));

    background-image: -webkit-linear-gradient(top, #62c462, #51a351);
    background-image: -o-linear-gradient(top, #62c462, #51a351);
    background-image: linear-gradient(to bottom, #62c462, #51a351);
    background-repeat: repeat-x;
    border-color: #51a351 #51a351 #387038;
    border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25);
    filter:
progid:DXImageTransform.Microsoft.gradient(startColorstr='#ff62c462', endColorstr='#ff51a351',
GradientType=0);

    filter: progid:DXImageTransform.Microsoft.gradient(enabled=false);
}

button.btn-danger {
    color: #ffffff;
    text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25);
    background-color: #da4f49;
    background-image: -moz-linear-gradient(top, #ee5f5b, #bd362f);
    background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#ee5f5b),
to(#bd362f));

```

```

background-image: -webkit-linear-gradient(top, #ee5f5b, #bd362f);
background-image: -o-linear-gradient(top, #ee5f5b, #bd362f);
background-image: linear-gradient(to bottom, #ee5f5b, #bd362f);
background-repeat: repeat-x;
border-color: #bd362f #bd362f #802420;
border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25);
filter:
progid:DXImageTransform.Microsoft.gradient(startColorstr='#ffee5f5b', endColorstr='#ffbd362f',
GradientType=0);

filter: progid:DXImageTransform.Microsoft.gradient(enabled=false);
}
.btn {
display: inline-block;
padding: 0.5em 0.7em;
width: 46%;
float: left;
margin: 1em 2%;
font-size: 20px;
line-height: 30px;
color: #333333;
text-align: center;
text-shadow: 0 1px 1px rgba(255, 255, 255, 0.75);
vertical-align: middle;
cursor: pointer;
background-color: #f5f5f5;
background-image: -moz-linear-gradient(top, #ffffff, #e6e6e6);
background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#ffffff),
to(#e6e6e6));

background-image: -webkit-linear-gradient(top, #ffffff, #e6e6e6);
background-image: -o-linear-gradient(top, #ffffff, #e6e6e6);

```

```

background-image: linear-gradient(to bottom, #ffffff, #e6e6e6);
background-repeat: repeat-x;
border: 1px solid #bbbbbb;
border-color: #e6e6e6 #e6e6e6 #bfbfbf;
border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25);
border-bottom-color: #a2a2a2;
-webkit-border-radius: 4px;
-moz-border-radius: 4px;
border-radius: 4px;
filter:
progid:DXImageTransform.Microsoft.gradient(startColorstr='#ffffff', endColorstr='#ffe6e6e6',
GradientType=0);

filter: progid:DXImageTransform.Microsoft.gradient(enabled=false);
-webkit-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px
rgba(0, 0, 0, 0.05);
-moz-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px
rgba(0, 0, 0, 0.05);
box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0,
0, 0.05);

box-sizing: border-box;
}
</style>

```

```
<script>
```

```
function httpGet( theUrl ){
```

```
var xmlhttp = null;
```

```
xmlhttp = new XMLHttpRequest();
```



```
xmlHttp.open( 'GET', theUrl, false );  
xmlHttp.send( null );  
  
return xmlHttp.responseText;  
}  
</script>
```

</head>

<body>

<!--<button class="btn btn-success" onclick="httpGet('http://localhost:8080/?serverfunction=ledSwitch&value=1')" >On</button>-->

<!--<button class="btn btn-danger" onclick="httpGet('http://localhost:8080/?serverfunction=ledSwitch&value=0')" >Off</button>-->

<button class="btn btn-success" onclick="httpGet('http://172.16.5.154:8080/?serverfunction=ledSwitch&value=1')" >LED1 On</button>

<button class="btn btn-danger" onclick="httpGet('http://172.16.5.154:8080/?serverfunction=ledSwitch&value=0')" >LED1 Off</button>

<button class="btn btn-success" onclick="httpGet('http://172.16.5.154:8080/?serverfunction=ledSwitch&value=2')" >LED2 On</button>

<button class="btn btn-danger" onclick="httpGet('http://172.16.5.154:8080/?serverfunction=ledSwitch&value=20')" >LED2 Off</button>

<button class="btn btn-success" onclick="httpGet('http://172.16.5.154:8080/?serverfunction=ledSwitch&value=3')" >Speaker On</button>

```
<button class="btn btn-danger" onclick="httpGet('http://172.16.5.154:8080/?
serverfunction=ledSwitch&value=30')" >Speaker Off</button>
```

```
</body>
</html>
```

Now the code for LED_Server.js is

```
var five = require( 'johnny-five' ),
    board,
    narf = require( 'narf' );
```

```
board = new five.Board();
```

```
/*
  Executes a command and fires event when done that
  will return the command output
*/
```

```
// The board's pins will not be accessible until
// the board has reported that it is ready
board.on("ready", function() {
  var val = 0;

  // Set pin 8 to OUTPUT mode
  this.pinMode( 8, 1 );
  this.pinMode( 12, 1 );
  this.pinMode( 7, 1 );
```

```

// Mode Table
// INPUT:  0
// OUTPUT: 1
// ANALOG: 2
// PWM:    3
// SERVO:  4

this.digitalWrite( 8, 0 );
this.digitalWrite( 12, 0 );
this.digitalWrite( 7, 0 );

    /* Api functions */

    var self = this;

    var APIFunctions = {

        GET : {

            ledSwitch : function ( data, callback ){

                data.url.value = parseInt( data.url.value, 0 );

                if( data.url.value === 1 || data.url.value === 0){

                    self.digitalWrite( 8, data.url.value );

                }

                if(data.url.value === 2 ){

                    self.digitalWrite( 12, 1 );

                }

                if(data.url.value === 20 ){

                    self.digitalWrite( 12, 0 );

```

```
        }  
        if(data.url.value === 3 ){  
            self.digitalWrite( 7, 1 );  
        }  
        if(data.url.value === 30 ){  
            self.digitalWrite( 7, 0 );  
        }
```

```
        callback( data.url.value );  
    }  
},  
POST : {}
```

```
};
```

```
console.log( narf );
```

```
var hs = new narf.HttpServer( { port : 8080 } ).start();
```

```
hs.on( 'port', function( port ){
```

```
    hs.addAPI( { functions : APIFunctions } );  
} );
```

```
});
```

```
narf.setDebug( false );
```

```
narf.pageServer( {
```

```
    port : 8079,
```

```
    path : __dirname + '/'
```

```
  } );
```

save both files

Complete the breadboard connections I.e LED1,Speaker,LED2 on their respective pins.

Now run command on terminal

```
#node LED_Server.js
```

Now open a browser with <http://172.16.5.154:8079/index.html>
It should show buttons

If it shows errors of mime ,fix it as follows

```
[root@localhost ~]# pwd
```

```
/root
```

```
[root@localhost ~]# vi /node_modules/pageserver/lib/pageserver.js
```

use mime-type instead of mime. So, Install mime-types first:

```
npm install mime-types
```

then make change in your code:

```
var mime=require('mime-types');
```

Make the change in pageserver.js file I.e by adding mime-types
Save changes and run node LED_Server