Lab 2: Managing Sensors and Actuators with Raspberry Pi & Arduino using Wyliodrin Studio (2)

物聯網技術與應用(英) IoT/M2M Technologies and Applications

國立交通大學資訊工程系
Department of Computer Science
National Chiao Tung University

October 5, 2018

Outline

- Automatic Lamp & Temperature Monitoring Application
  - Assembling Schematics *(Checkpoint 1)*
  - Coding with Blocks *(Checkpoint 2)*
  - Using Wyliodrin Dashboard *(Checkpoint 3)*
  - Task *(Checkpoint 4)*
- HTTP Application
  - Sending Sensor Values to Server *(Checkpoint 5)*
  - Controlling Actuators via Web *(Checkpoint 6)*

Attention!

Please, start your virtual machine, and connect your Raspberry Pi to the power source now!

AUTOMATIC LAMP & TEMPERATURE MONITORING APPLICATION

Goal

Simulating a smart-home environment with automatic lamp and temperature monitoring.

- Being able to know the room's temperature through the Wyliodrin Dashboard.
- When the room is getting dark, the lamp should be automatically turned on.
- When the room is having enough light, the lamp should be automatically turned off.

List of Components

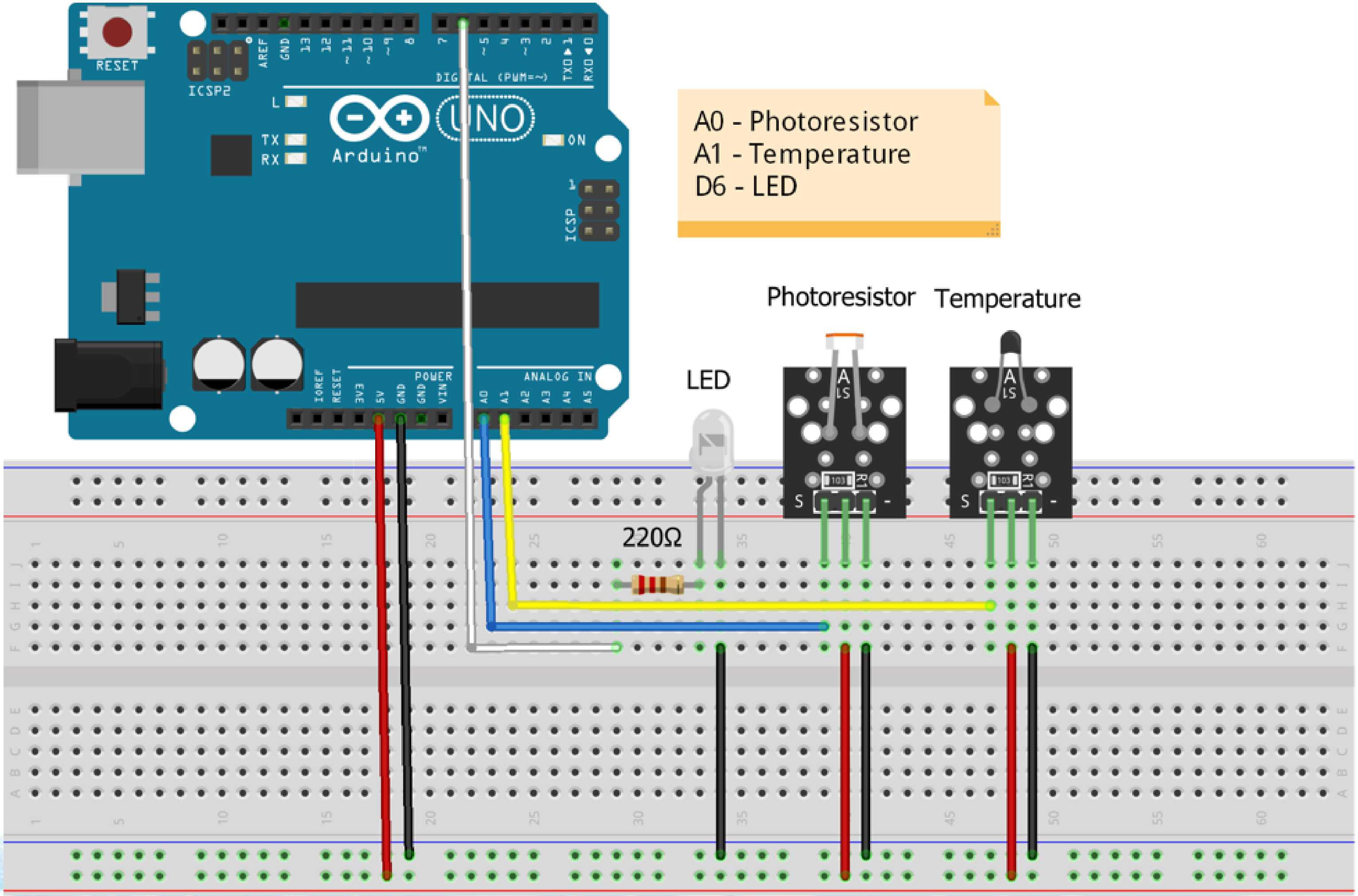
- 1 breadboard.
- 1 Raspberry Pi 3.
- 1 Arduino Uno
- 1 resistor 220Ω.
- 1 LED any color.
- 1 Arduino Photoresistor sensor module.
- 1 Arduino Temperature sensor module.
- Cables.

Schematics

Connect all the modules and components according to the schema.

Then **Connect Arduino to Raspberry with Arduino USB-cable**

Ask TA if you are not sure how to connect the components.

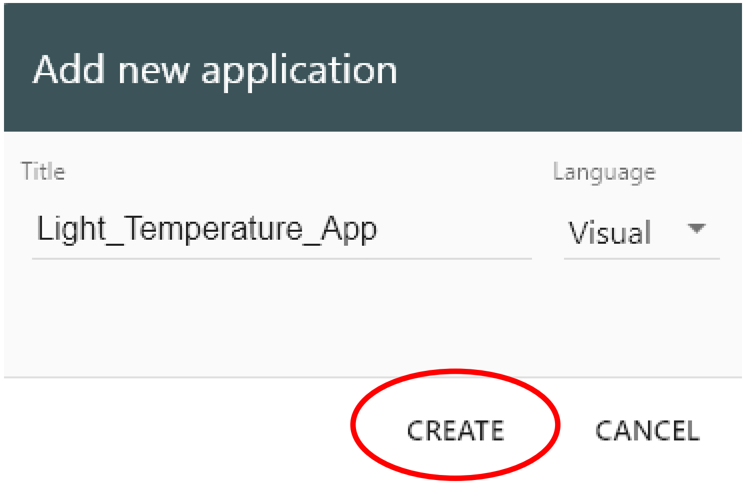


A0 - Photoresistor
A1 - Temperature
D6 - LED

CHECKPOINT 1!

Implementation (1)

1. Create new application in Wyliodrin.
2. Type “Light_Temperature_App” for Title and choose “Visual” as Language.
3. Click the “Create” button.



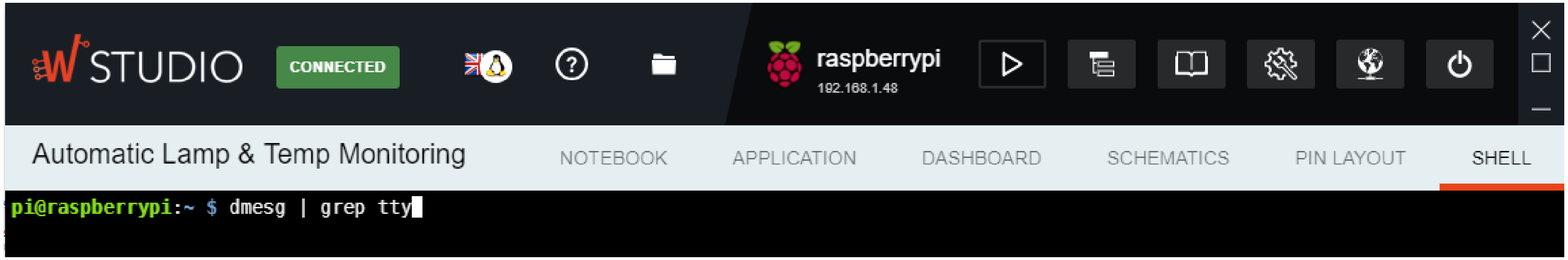
Add new application

Title: Light_Temperature_App

Language: Visual

Implementation (2)

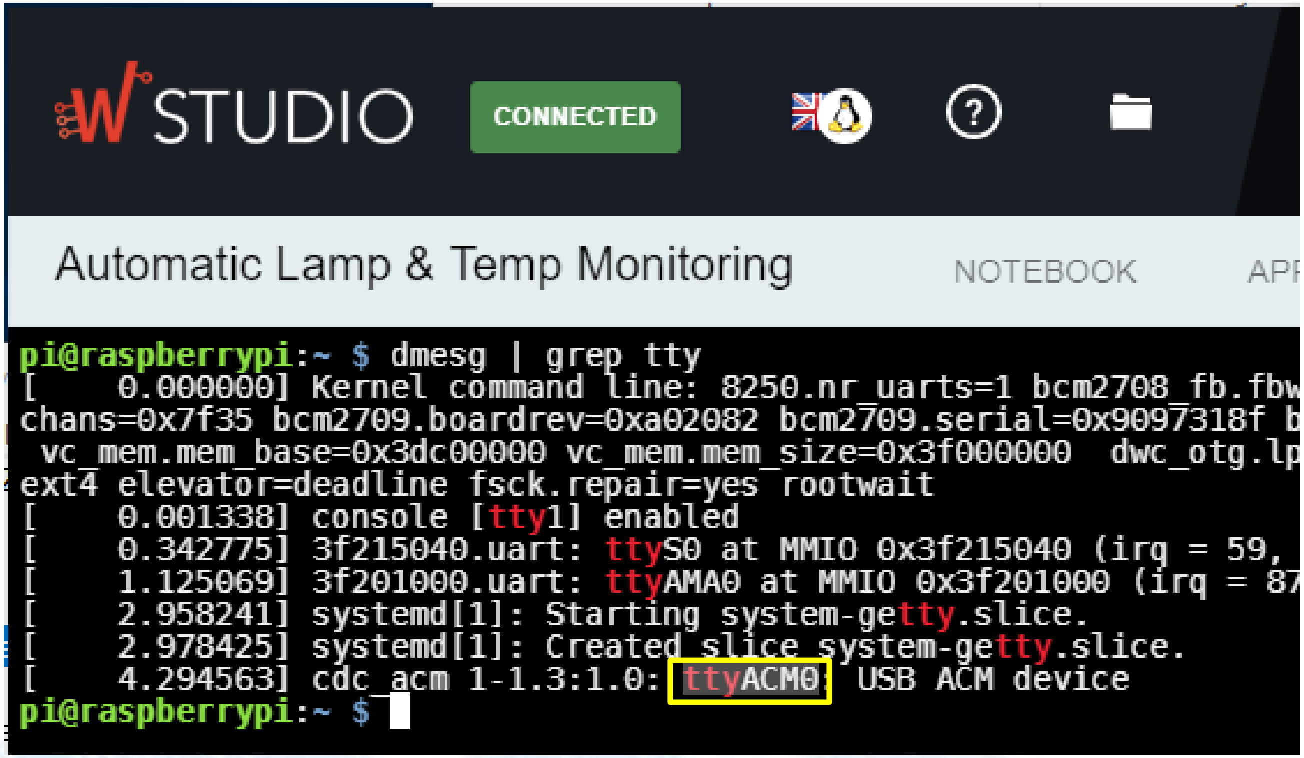
4. Click on the “Shell” tab.
5. Type “dmesg | grep tty” and press ENTER.



```
pi@raspberrypi:~ $ dmesg | grep tty
```

Implementation (3)

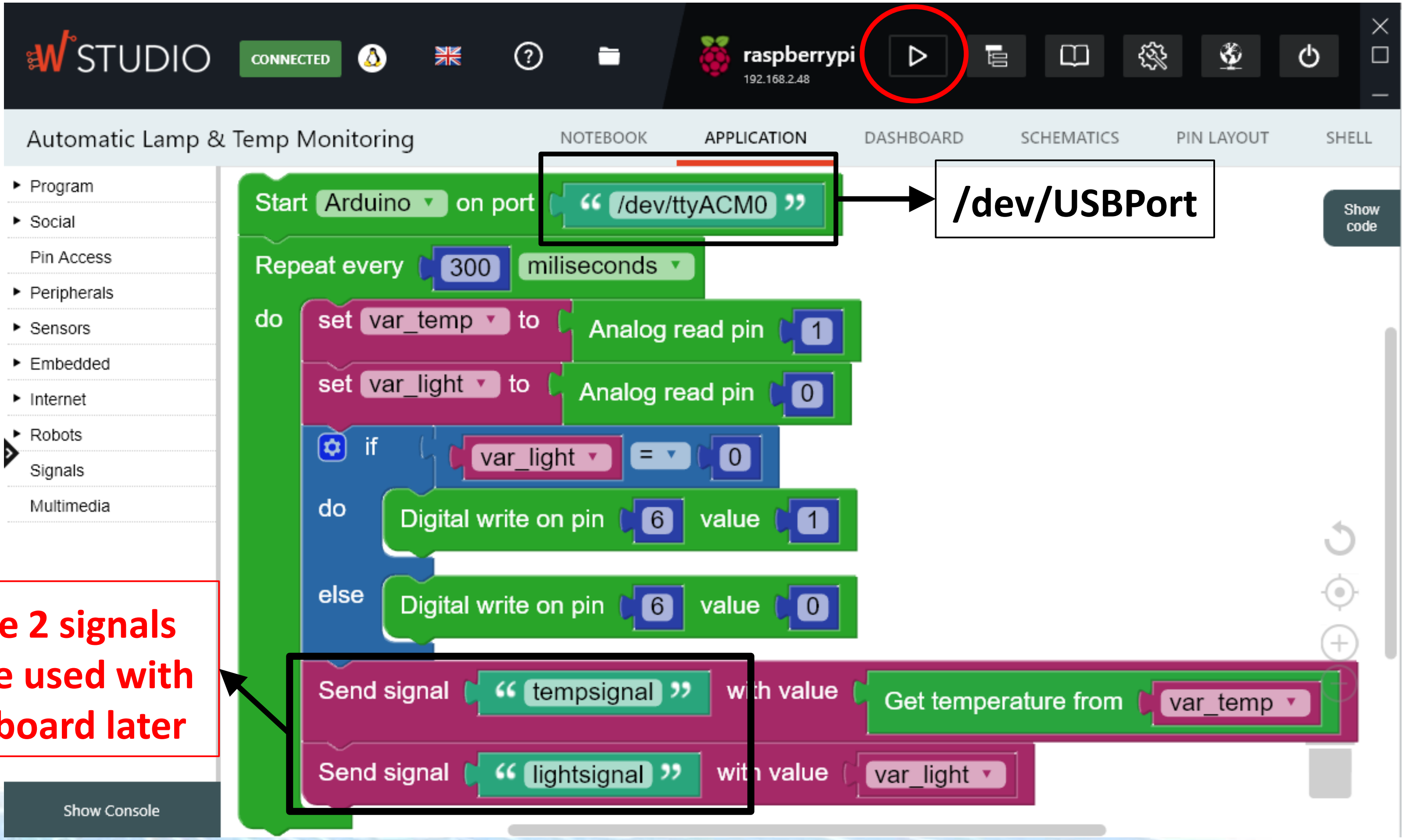
6. Look at the last line in the output of executing “dmesg | grep tty” command.
7. Please notice and remember the highlighted string in your screen. It is the **USBPort** of your connected Arduino with Raspberry Pi via USB. You may have different port. You will use this **USBPort** in the next steps.



```
pi@raspberrypi:~ $ dmesg | grep tty
[    0.000000] Kernel command line: 8250.nr_uarts=1 bcm2708_fb.fbw
chans=0x7f35 bcm2709.boardrev=0xa02082 bcm2709.serial=0x9097318f b
 vc_mem.mem_base=0x3dc00000 vc_mem.mem_size=0x3f000000  dwc_otg.lp
ext4 elevator=deadline fsck.repair=yes rootwait
[    0.001338] console [tty1] enabled
[    0.342775] 3f215040.uart: ttyS0 at MMIO 0x3f215040 (irq = 59,
[    1.125069] 3f201000.uart: ttyAMA0 at MMIO 0x3f201000 (irq = 87
[    2.958241] systemd[1]: Starting system-getty.slice.
[    2.978425] systemd[1]: Created slice system-getty.slice.
[    4.294563] cdc_acm 1-1.3:1.0: ttyACM0: USB ACM device
pi@raspberrypi:~ $
```

Implementation (4)

8. Click on “Application” tab and follow the code blocks as shown below.
9. Run your application.



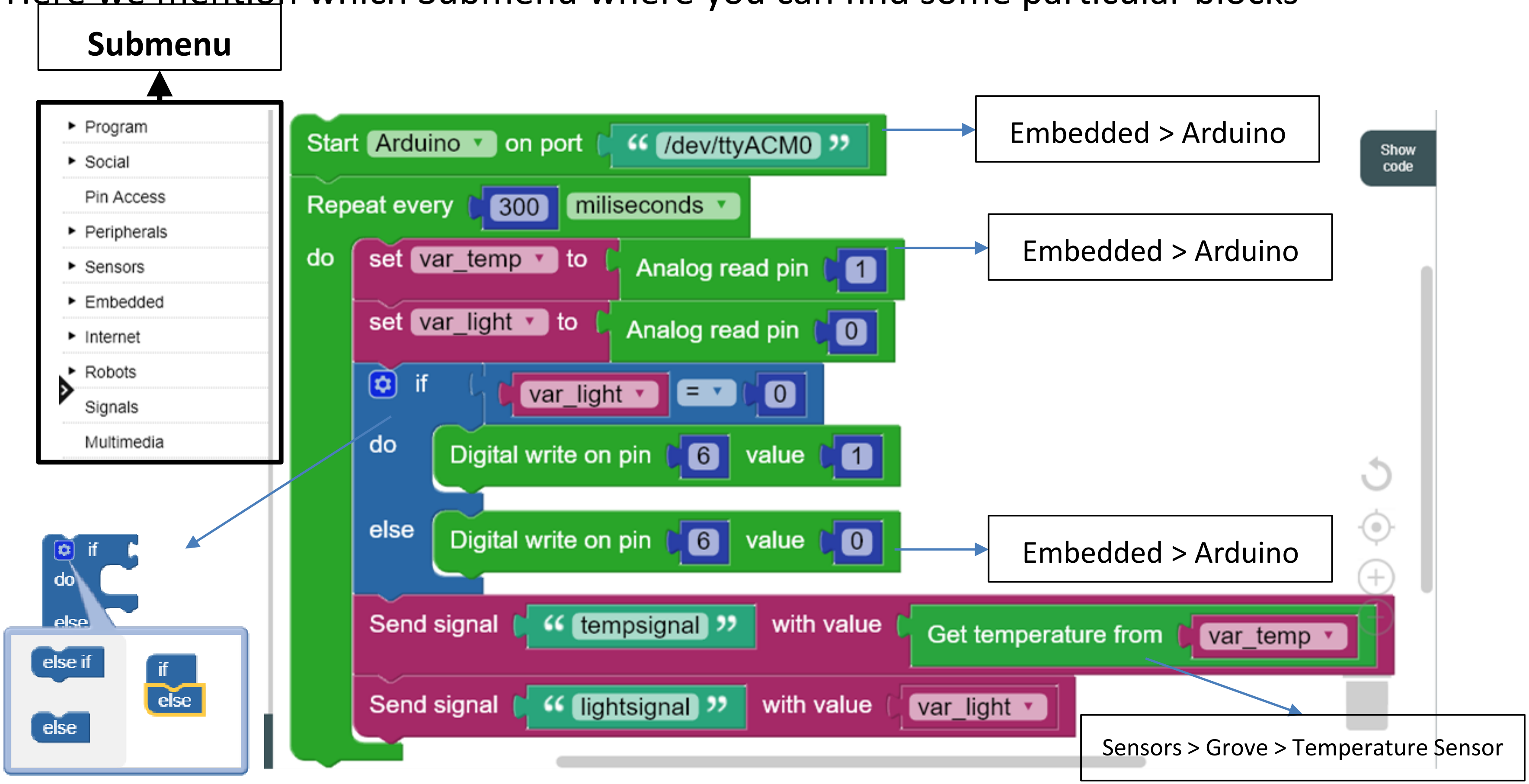
```
Start Arduino on port “/dev/ttyACM0”
Repeat every 300 miliseconds
do  set var_temp to Analog read pin 1
    set var_light to Analog read pin 0
    if var_light = 0
    do   Digital write on pin 6 value 1
    else Digital write on pin 6 value 0
    Send signal “tempsignal” with value Get temperature from var_temp
    Send signal “lightsignal” with value var_light
```

/dev/USBPort

These 2 signals will be used with Dashboard later

Information

Here we mention which Submenu where you can find some particular blocks



Submenu

Embedded > Arduino

Embedded > Arduino

Embedded > Arduino

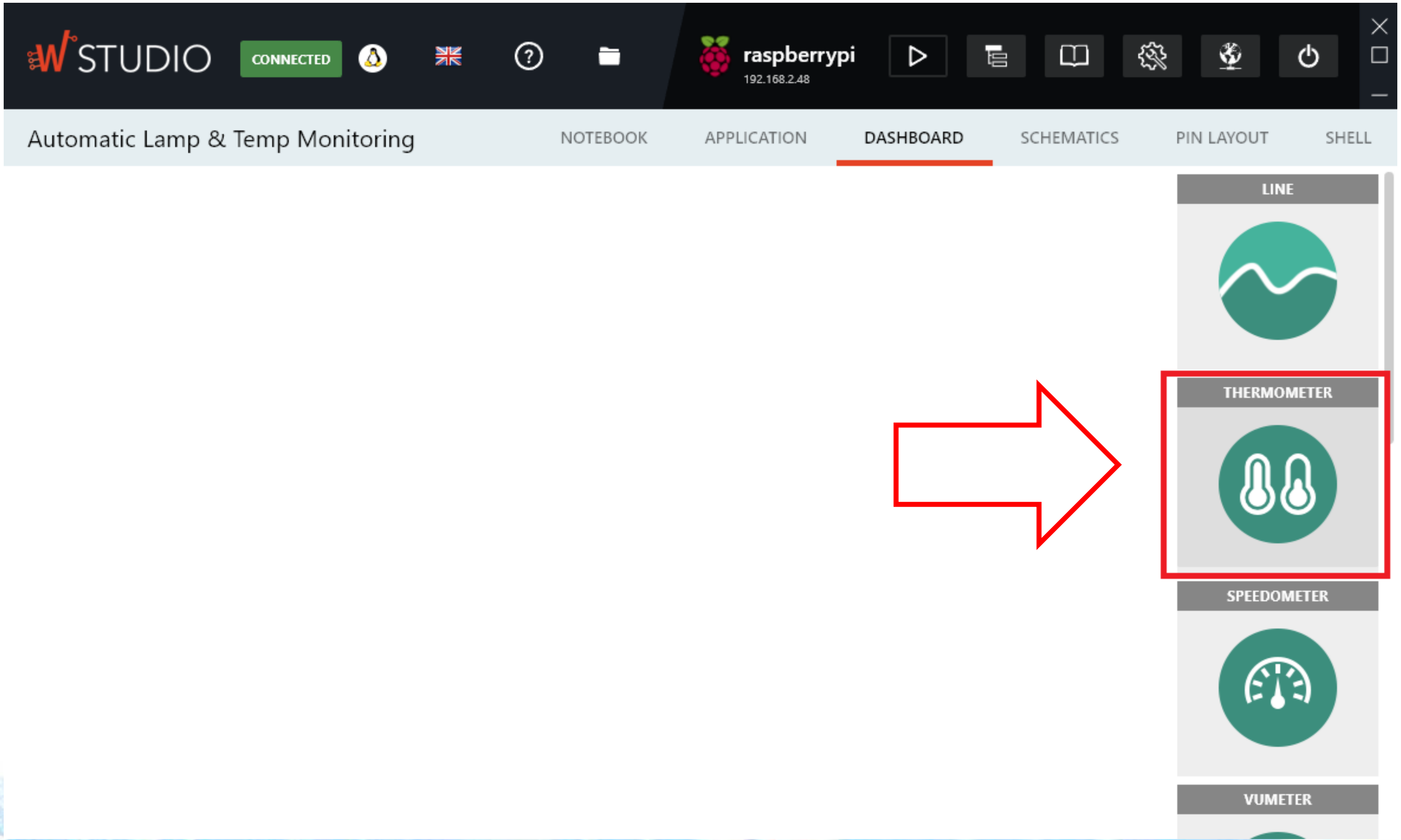
Sensors > Grove > Temperature Sensor

```
Start Arduino on port "/dev/ttyACM0"
Repeat every 300 miliseconds
do  set var_temp to Analog read pin 1
    set var_light to Analog read pin 0
    if var_light = 0
    do   Digital write on pin 6 value 1
    else Digital write on pin 6 value 0
    Send signal "tempsignal" with value Get temperature from var_temp
    Send signal "lightsignal" with value var_light
```

CHECKPOINT 2!

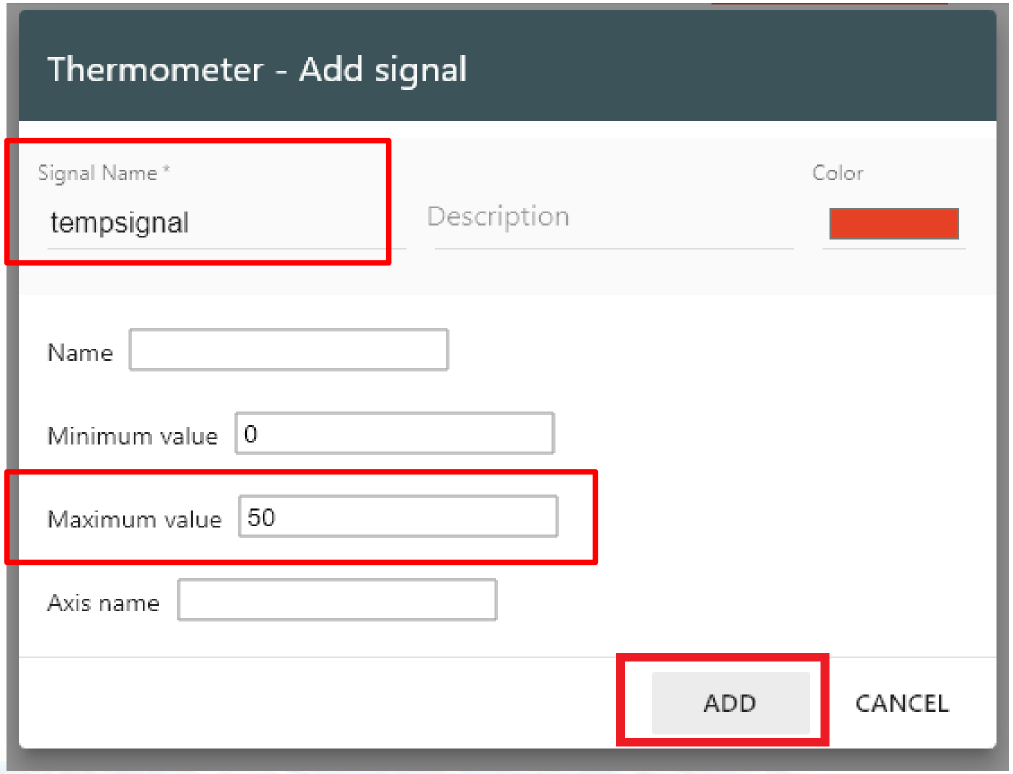
Signals and Dashboard (1)

1. Click on “Dashboard” tab.
2. On the right panel, click on “Thermometer”.



Signals and Dashboard (2)

3. Type “tempsignal” on Signal Name.
4. Change “Maximum value” to “50”.
5. Click on “Add” button.



Thermometer - Add signal

Signal Name * tempsignal

Description

Color 🟥

Name

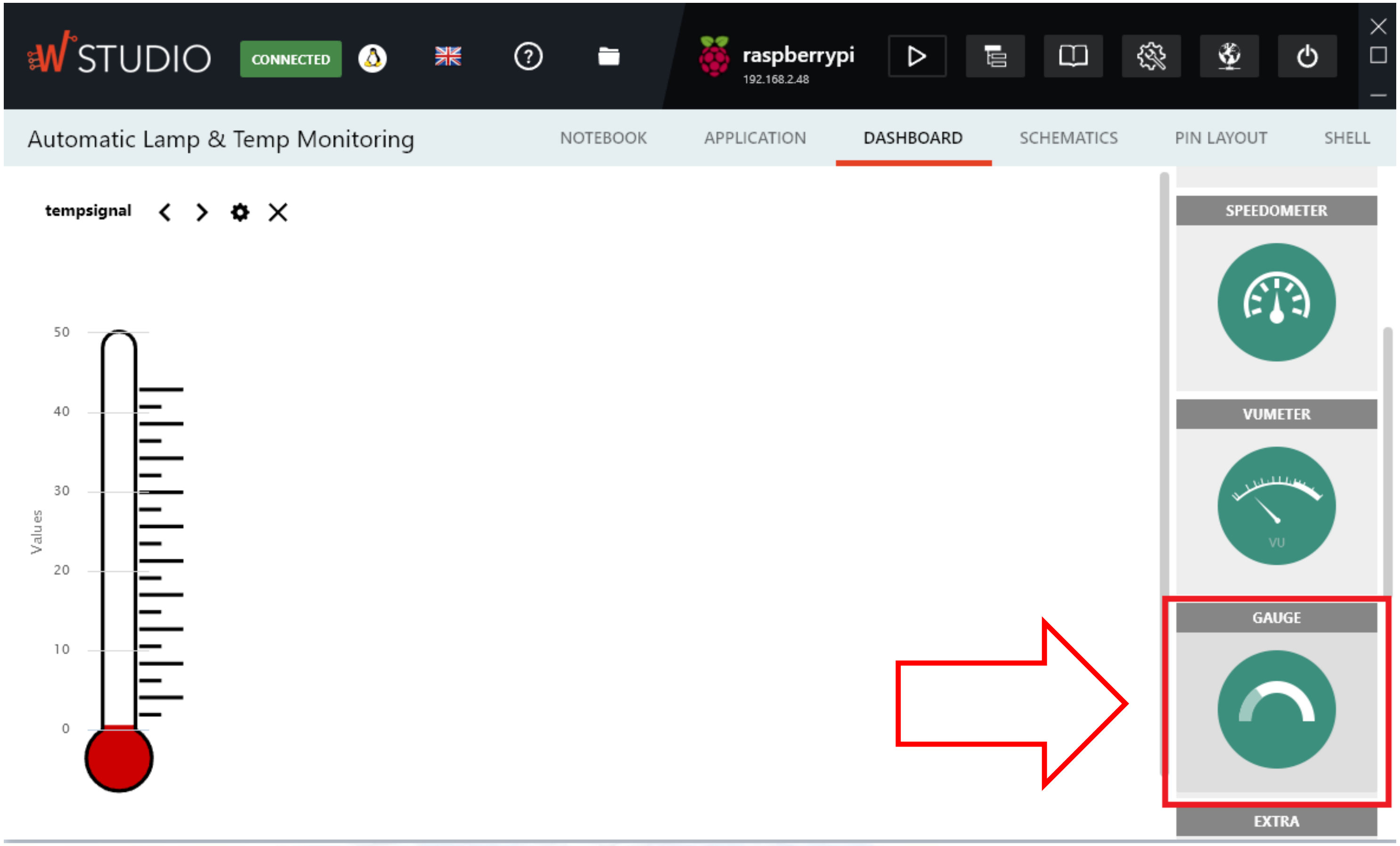
Minimum value 0

Maximum value 50

Axis name

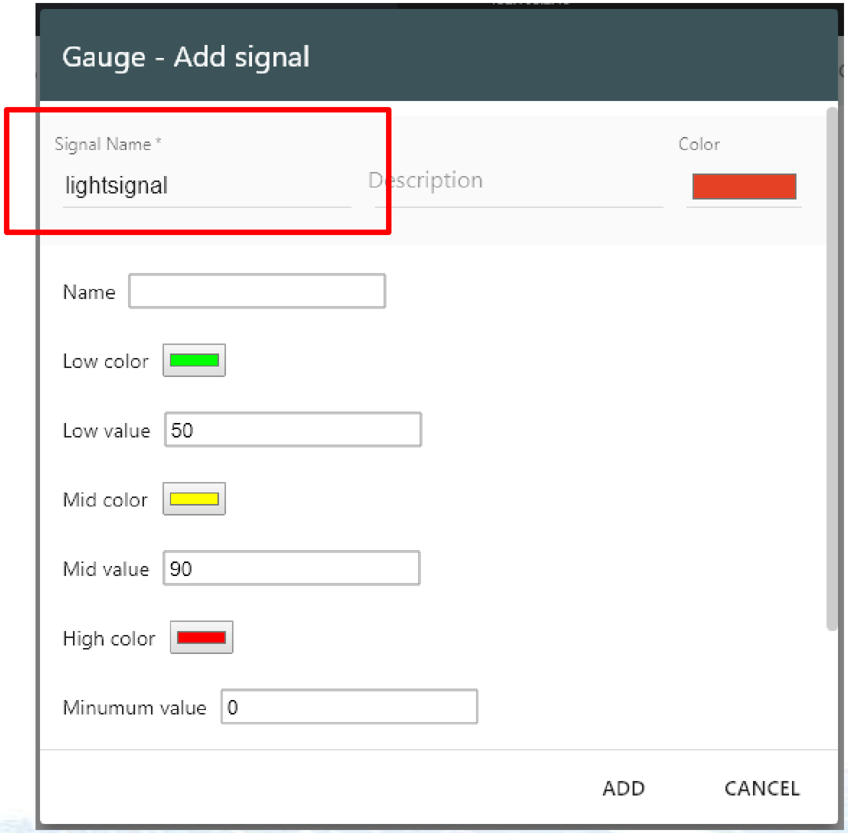
Signals and Dashboard (3)

6. On the right panel, click on “Gauge”.



Signals and Dashboard (4)

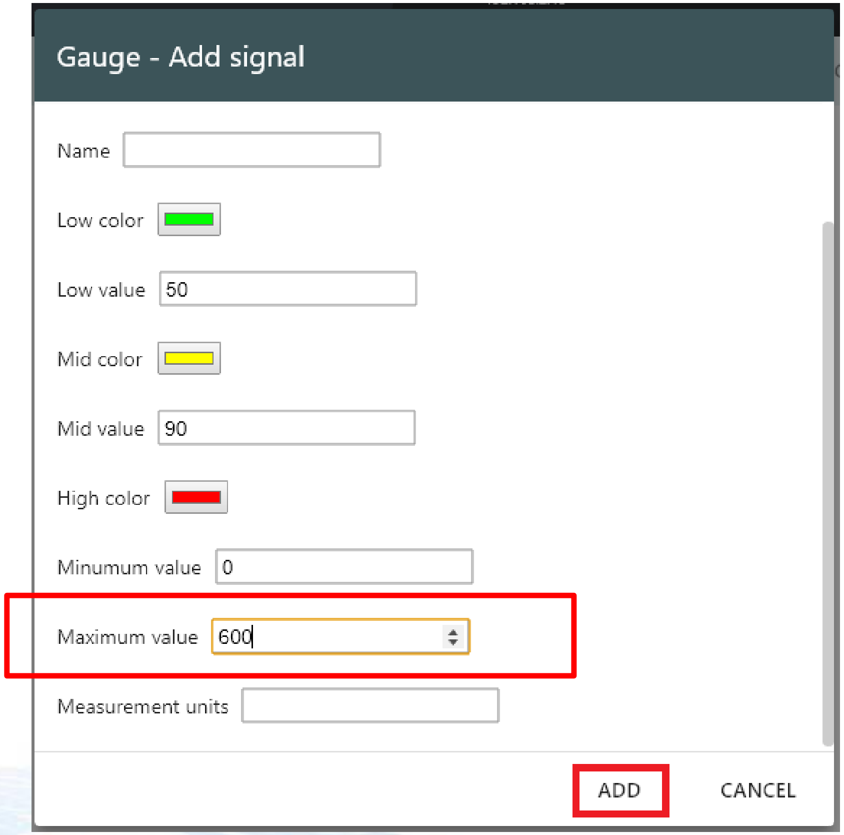
7. Type “lightsignal” on Signal Name.
8. Change “Maximum value” to “600”.
9. Click on “Add” button.



Gauge - Add signal

Signal Name *		Color
lightsignal	Description	🟥

Name	
Low color	🟩
Low value	50
Mid color	🟨
Mid value	90
High color	🟥
Minumum value	0

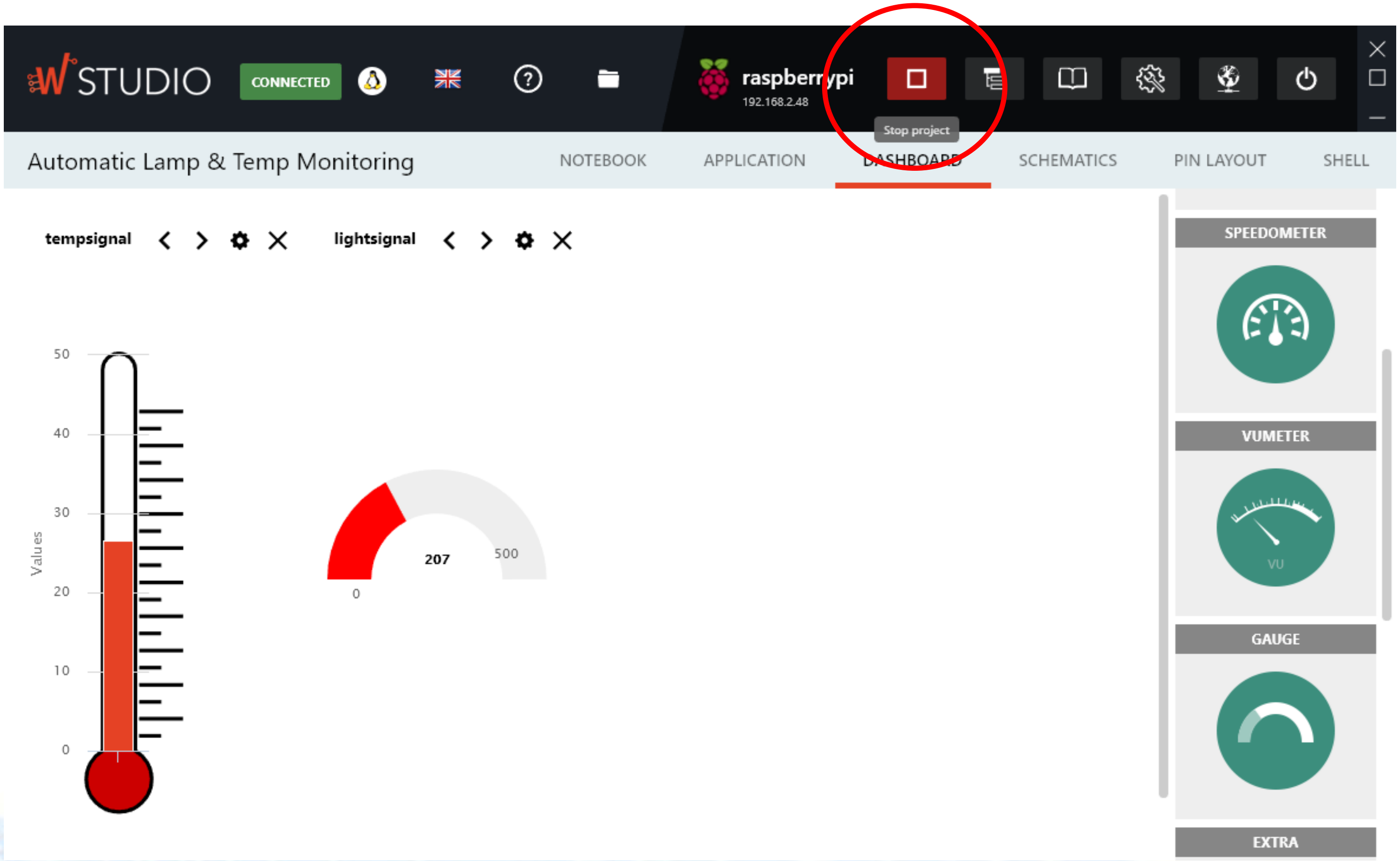


Gauge - Add signal

Name	
Low color	🟩
Low value	50
Mid color	🟨
Mid value	90
High color	🟥
Minumum value	0
Maximum value	600
Measurement units	

Signals and Dashboard (5)

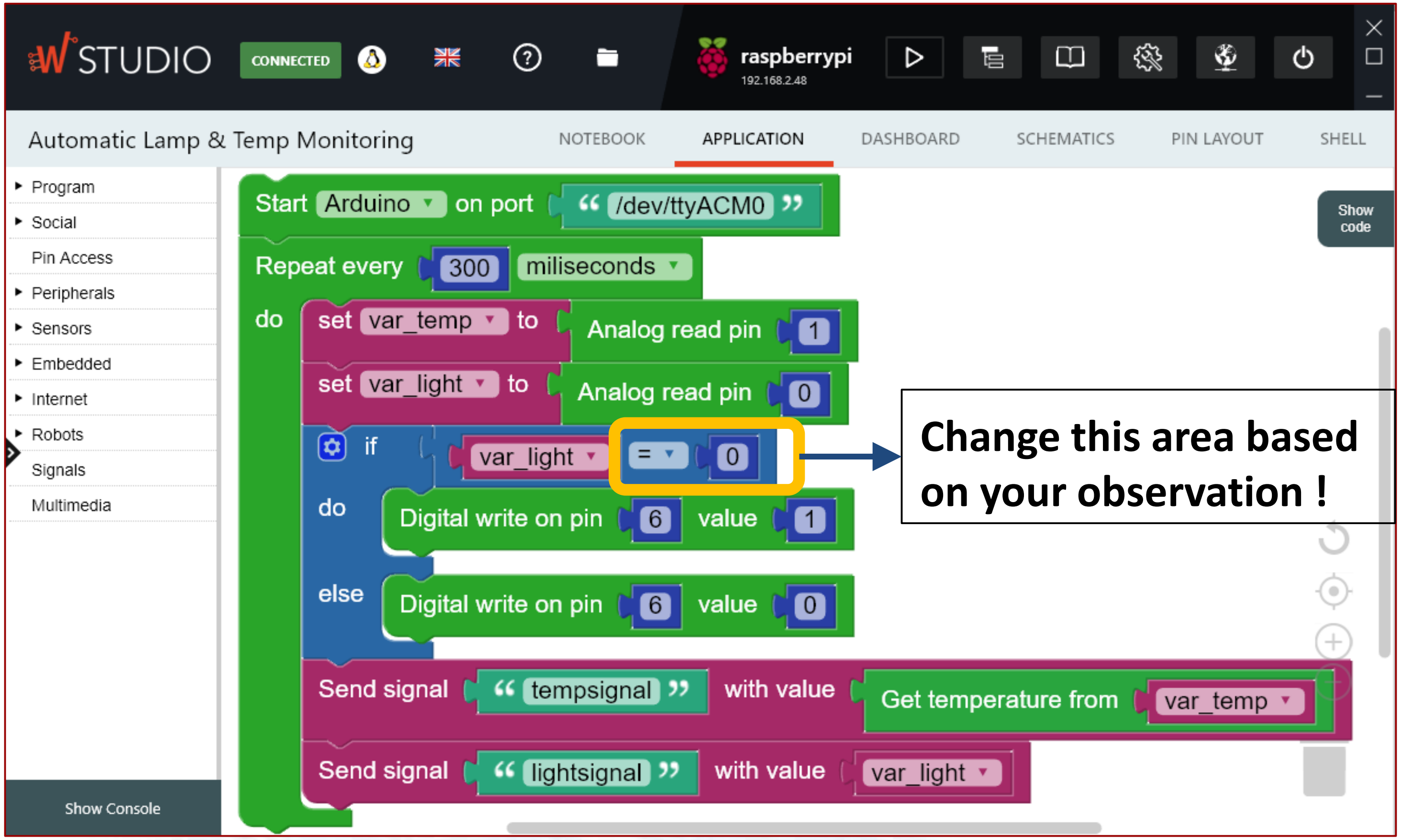
10. Run the project.
11. Now you can monitor light intensity and temperature from Dashboard.



CHECKPOINT 3!

Task

Please change the condition and the value in the highlighted area to trigger the LED's on status based on your observation in the Dashboard. Example: > 100.



```
Start Arduino on port "/dev/ttyACM0"
Repeat every 300 miliseconds
do  set var_temp to Analog read pin 1
    set var_light to Analog read pin 0
    if var_light = 0
    do    Digital write on pin 6 value 1
    else  Digital write on pin 6 value 0
    Send signal "tempsignal" with value Get temperature from var_temp
    Send signal "lightsignal" with value var_light
```

Change this area based on your observation !

CHECKPOINT 4!

HTTP APPLICATION

SENDING SENSOR VALUES TO A SERVER

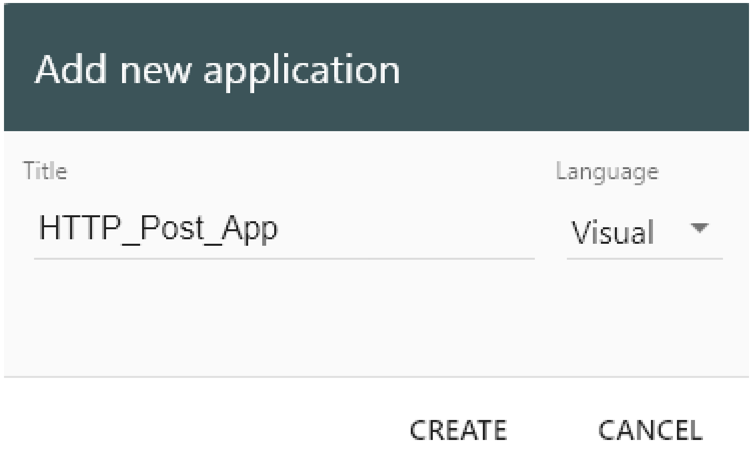
Goal

- Sending values of sensors to a server via HTTP Post.

We will reuse the schematics from our previous application (Light Intensity and Temperature Alarm Application)

Implementation (1)

1. Create a new “Visual” application called “HTTP_Post_App”.



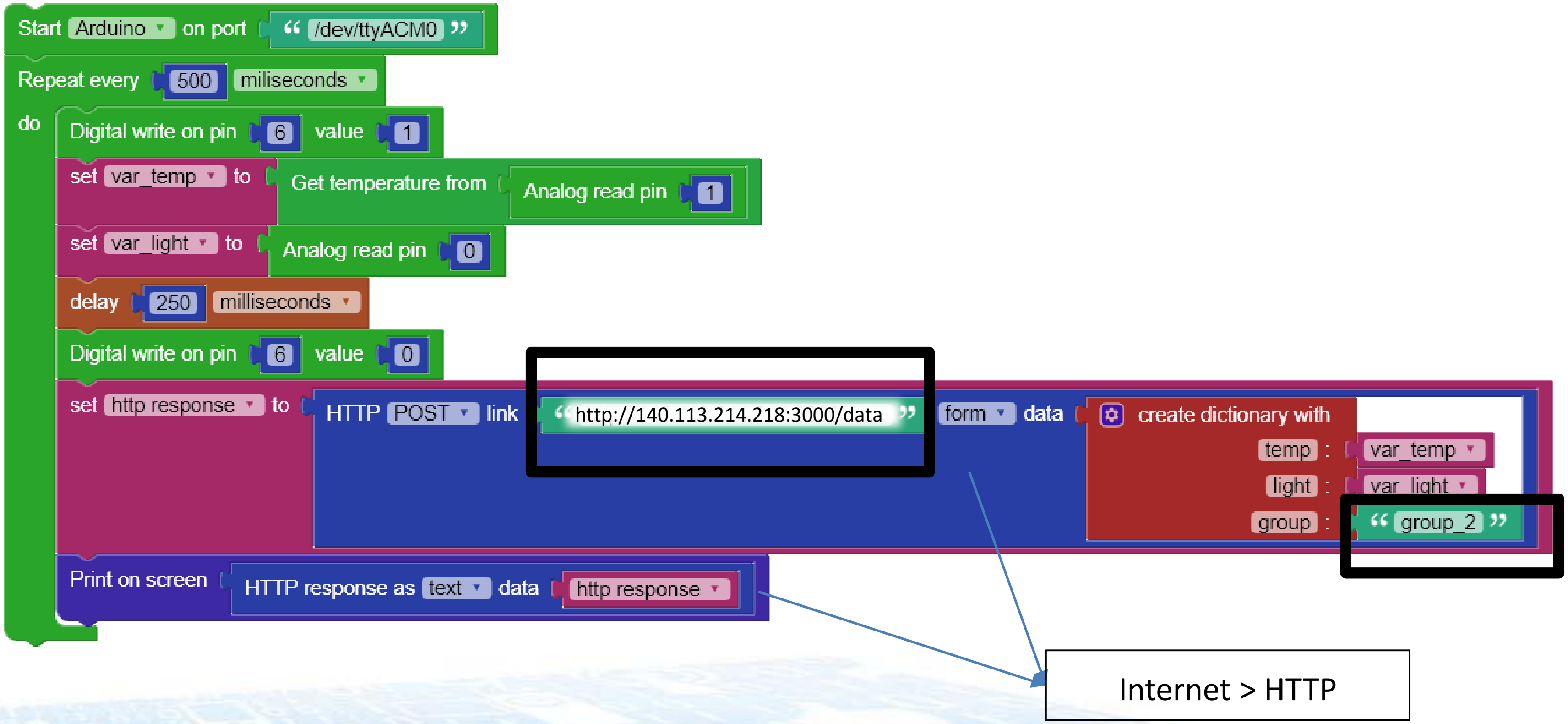
Add new application

Title: HTTP_Post_App

Language: Visual

Implementation (2)

2. Follow the code blocks as shown below.
3. Please type “**YOUR_GROUP_ID**” in the highlighted value of “group”.
4. Please type “**http://140.113.214.218:3000/data**” for “link”.
5. Run the application.

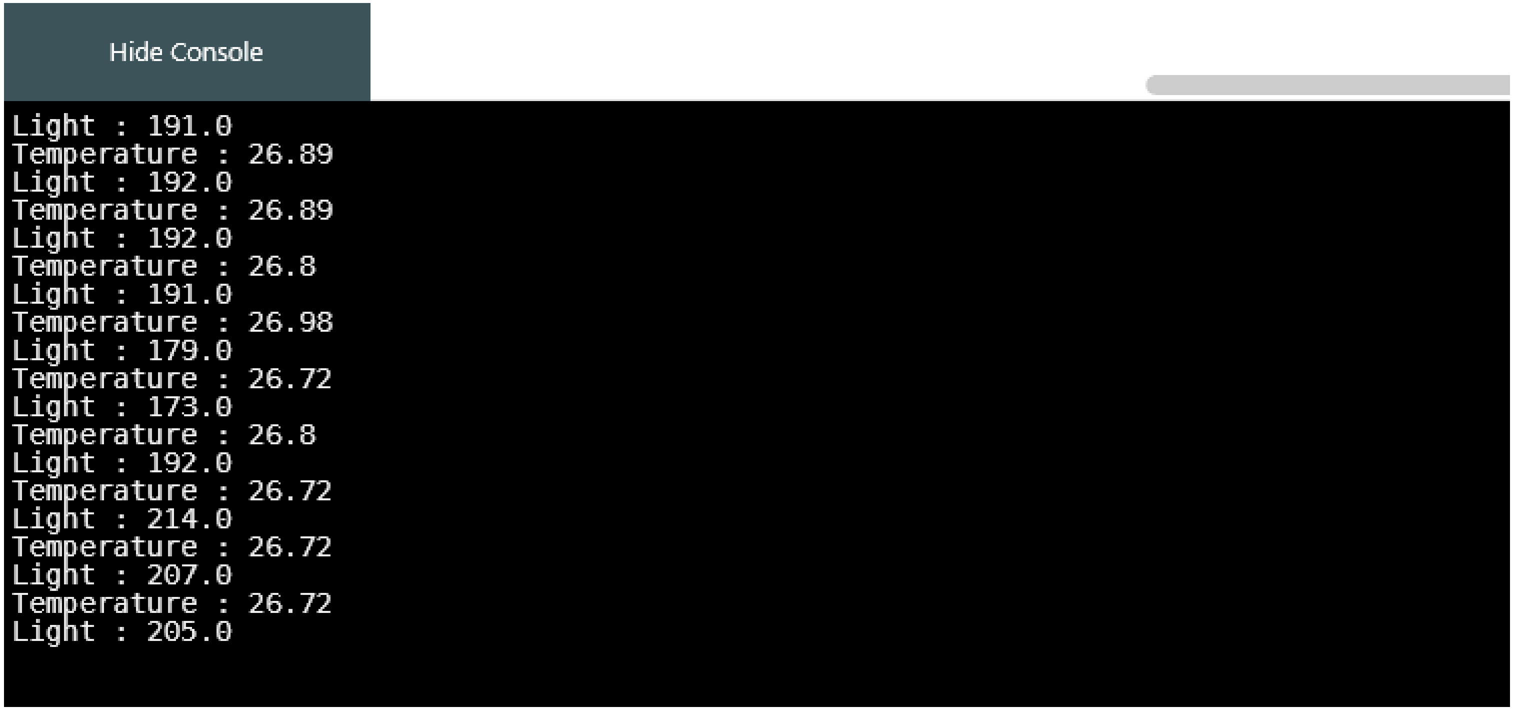


```
Start Arduino on port “/dev/ttyACM0”
Repeat every 500 miliseconds
do
    Digital write on pin 6 value 1
    set var_temp to Get temperature from Analog read pin 1
    set var_light to Analog read pin 0
    delay 250 milliseconds
    Digital write on pin 6 value 0
    set http response to HTTP POST link “http://140.113.214.218:3000/data” form data create dictionary with
        temp : var_temp
        light : var_light
        group : “group_2”
    Print on screen HTTP response as text data http response
```

Internet > HTTP

Implementation (3)

6. After you run the application, check the server's response in the console.



```
Light : 191.0
Temperature : 26.89
Light : 192.0
Temperature : 26.89
Light : 192.0
Temperature : 26.8
Light : 191.0
Temperature : 26.98
Light : 179.0
Temperature : 26.72
Light : 173.0
Temperature : 26.8
Light : 192.0
Temperature : 26.72
Light : 214.0
Temperature : 26.72
Light : 207.0
Temperature : 26.72
Light : 205.0
```

CHECKPOINT 5!

HTTP APPLICATION

CONTROLLING ACTUATORS VIA WEB

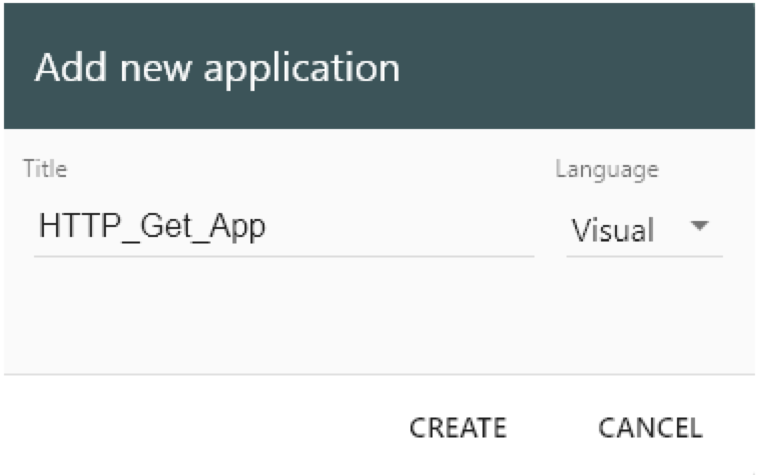
Goal

- Turning On/Off LEDs via Web using HTTP Get.

We will reuse the schematics from our previous application (Automatic Lamp & Temperature Monitoring Application)

Implementation (1)

1. Create a new “Visual” application called “HTTP_Get_App”.



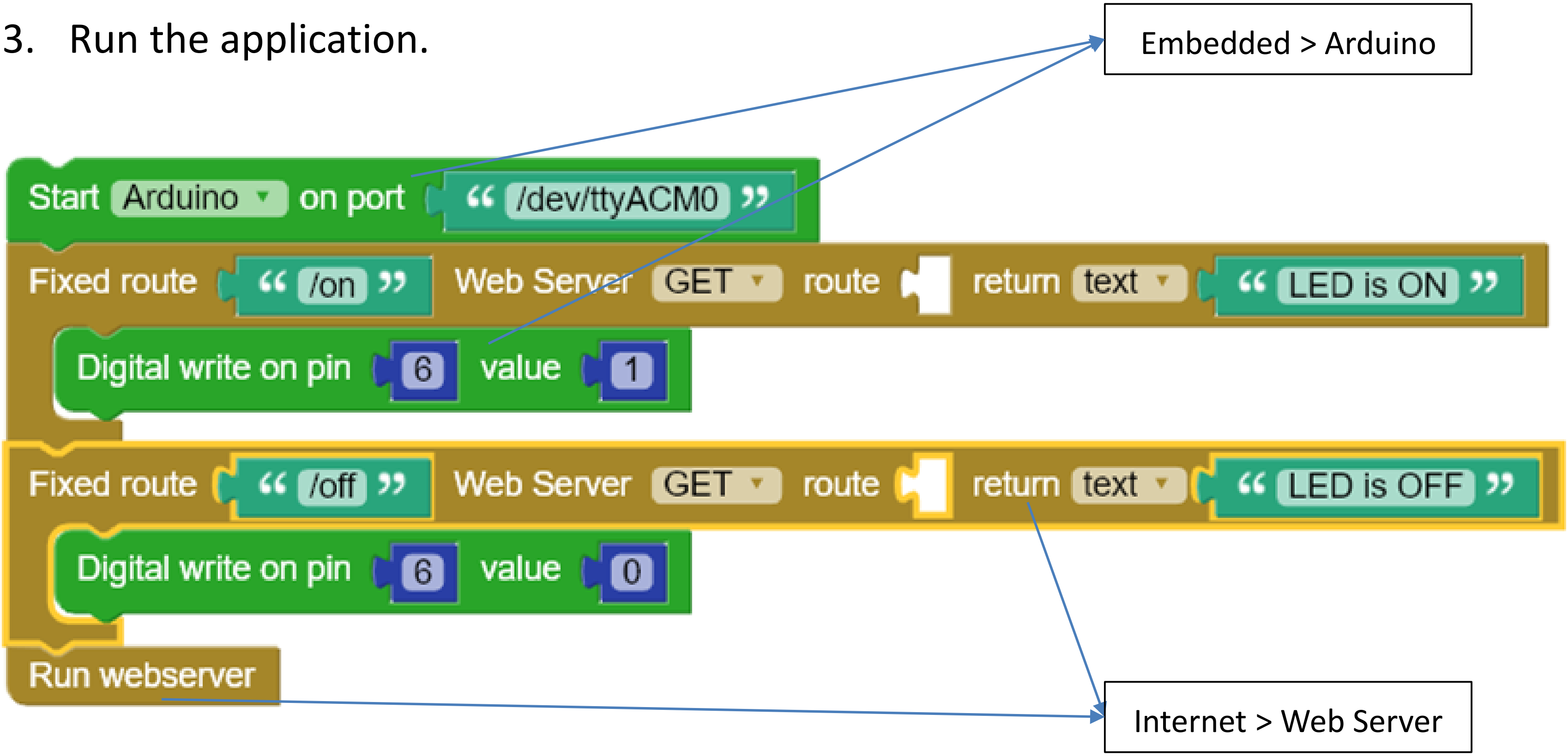
Add new application

Title: HTTP_Get_App

Language: Visual

Implementation (2)

2. Follow the code blocks as shown below.
3. Run the application.



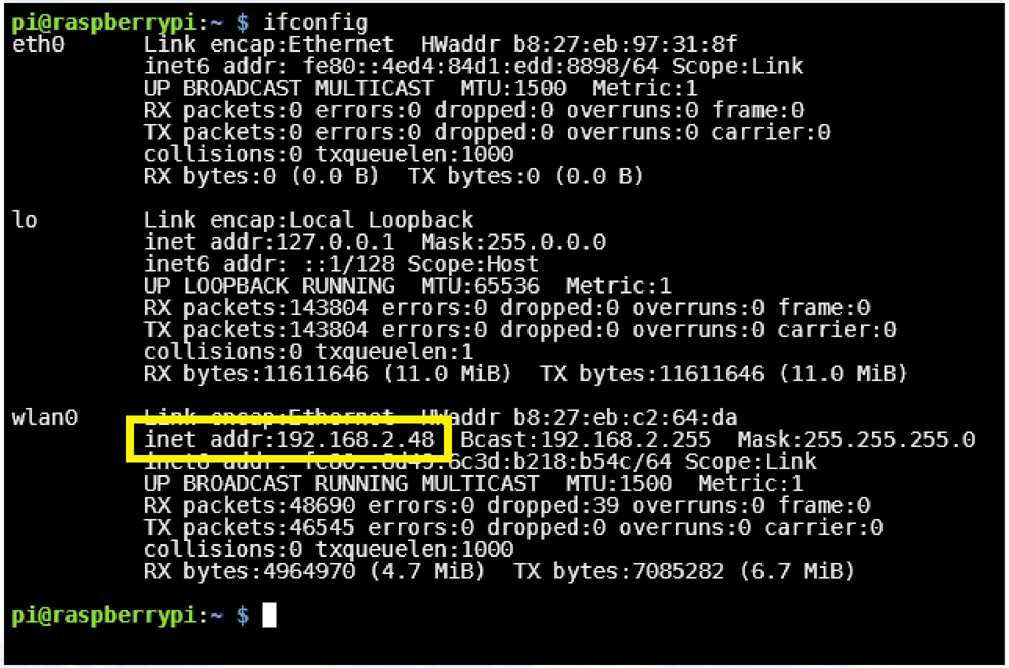
```
Start Arduino ▾ on port " /dev/ttyACM0 "
Fixed route " /on " Web Server GET ▾ route    return text ▾ " LED is ON "
    Digital write on pin 6 value 1
Fixed route " /off " Web Server GET ▾ route    return text ▾ " LED is OFF "
    Digital write on pin 6 value 0
Run webserver
```

Embedded > Arduino

Internet > Web Server

Implementation (3)

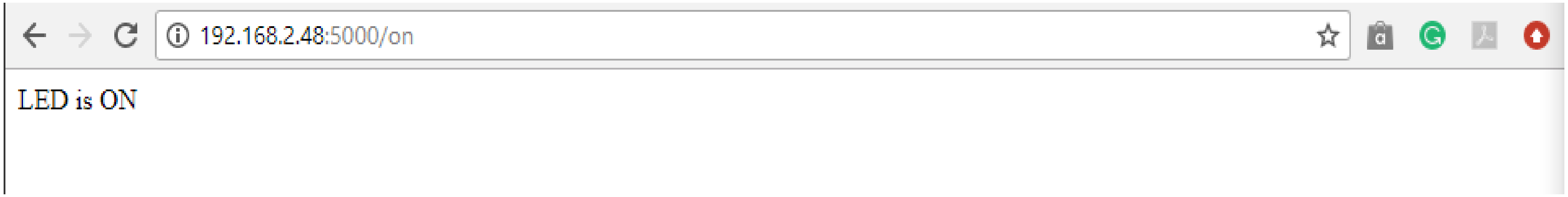
4. Open a Web browser in your PC or VM and target to the following URLs:
   http://YOUR.RASPBERRY.IP.ADDRESS:5000/on
   http://YOUR.RASPBERRY.IP.ADDRESS:5000/off
5. If you forgot your Raspberry Pi's IP address, you can get it by typing "ifconfig" in the Wyliodrin "Shell".



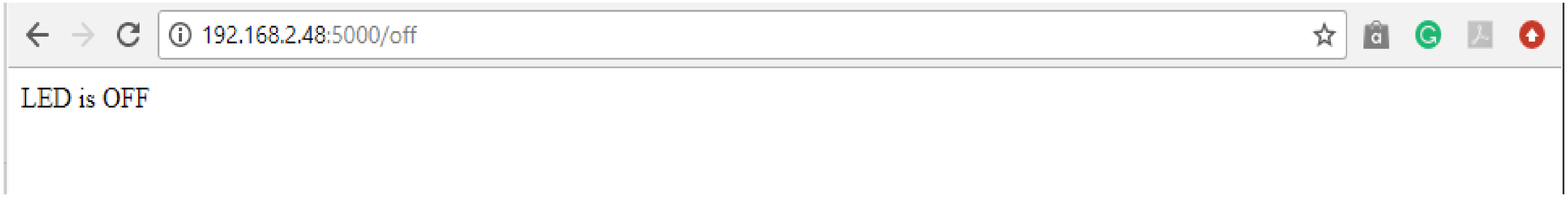
```
pi@raspberrypi:~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:97:31:8f
          inet6 addr: fe80::4ed4:84d1:edd:8898/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:143804 errors:0 dropped:0 overruns:0 frame:0
          TX packets:143804 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:11611646 (11.0 MiB)  TX bytes:11611646 (11.0 MiB)

wlan0     Link encap:Ethernet  HWaddr b8:27:eb:c2:64:da
          inet addr:192.168.2.48  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::8d49:6c3d:b218:b54c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:48690 errors:0 dropped:39 overruns:0 frame:0
          TX packets:46545 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4964970 (4.7 MiB)  TX bytes:7085282 (6.7 MiB)

pi@raspberrypi:~ $ 
```

Implementation (4)

6. The browser will show the corresponding message (as shown in the example below).
7. Please verify that the status of your LEDs in your breadboard is correct.


192.168.2.48:5000/on

LED is ON


192.168.2.48:5000/off

LED is OFF

CHECKPOINT 6!