



行動寬頻尖端技術跨校教學聯盟



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 Wylidrin 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.

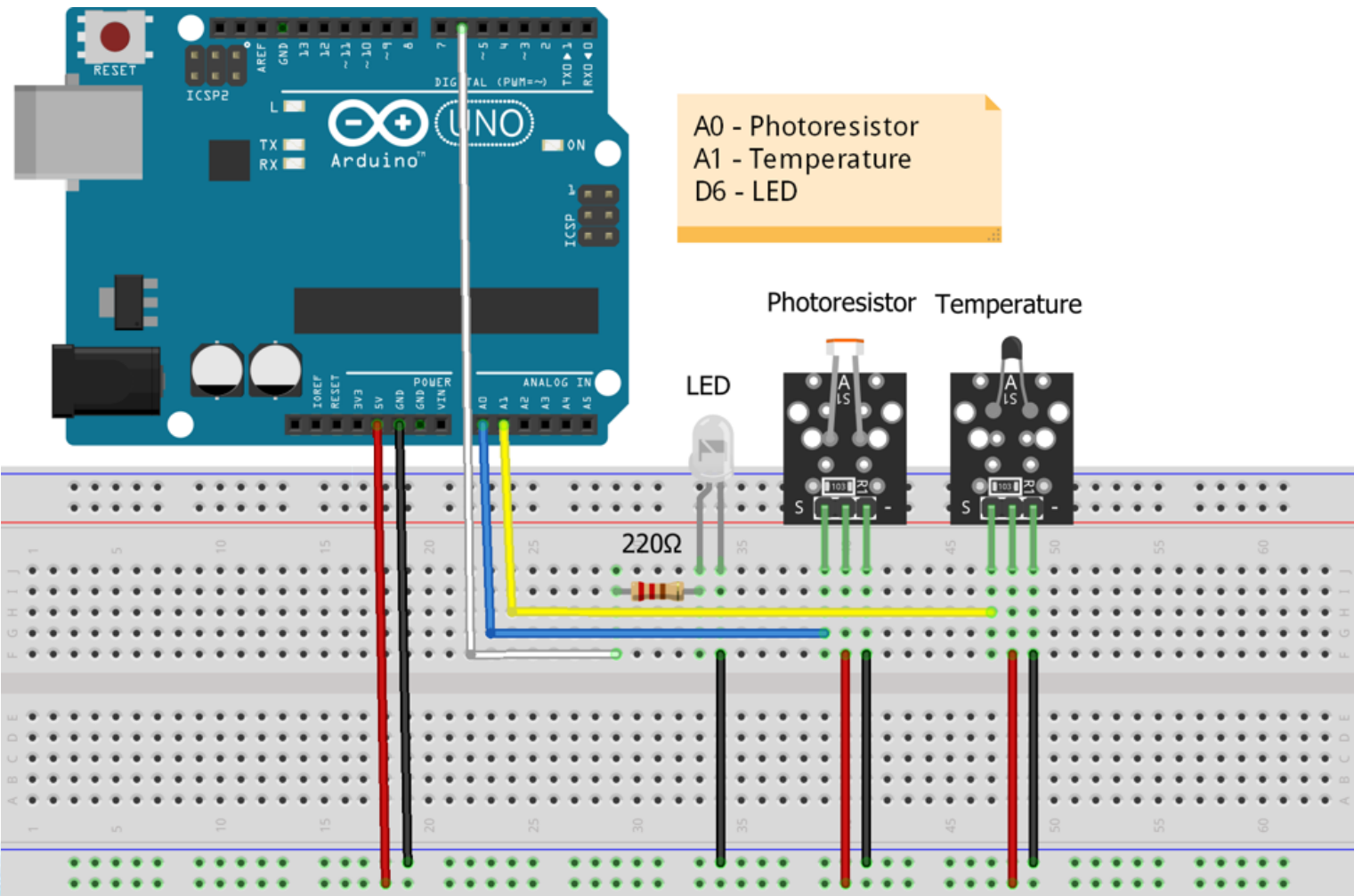
- Being able to know the room's temperature through the Wylidrin Dashboard.
- When the room is getting dark, the lamp should be automatically turned on.
- When the room is having enough light, the lamp should be automatically turned off.

List of Components

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

Schematics

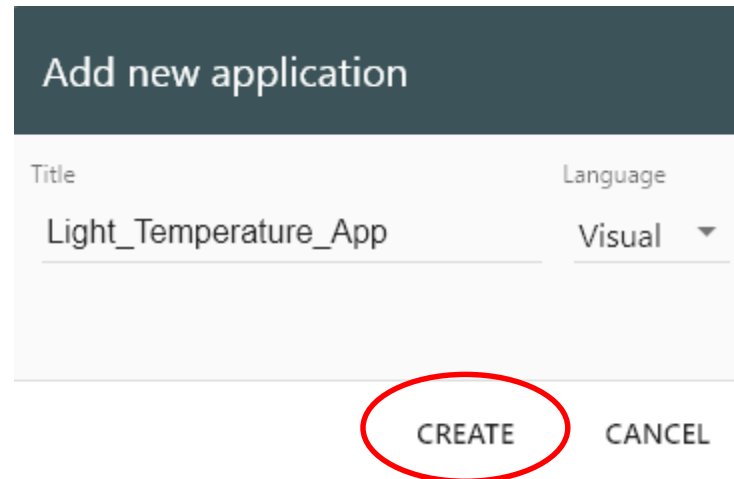
Connect all the modules and components according to the schema.
Then **Connect Arduino to Raspberry with Arduino USB-cable**
Ask TA if you are not sure how to connect the components.



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.

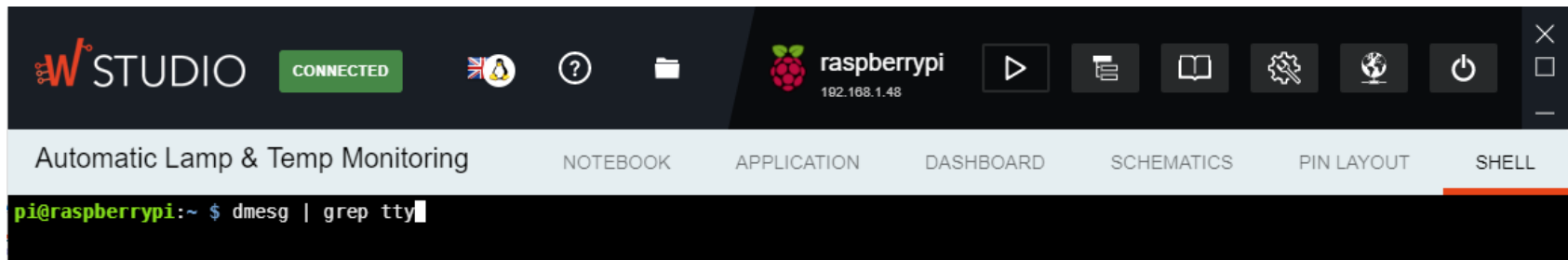
A screenshot of a web form titled "Add new application". The form has two input fields: "Title" and "Language". The "Title" field contains the text "Light_Temperature_App". The "Language" field is a dropdown menu with "Visual" selected. Below the form, there are two buttons: "CREATE" and "CANCEL". The "CREATE" button is circled in red.

Title	Language
Light_Temperature_App	Visual ▼

CREATE CANCEL

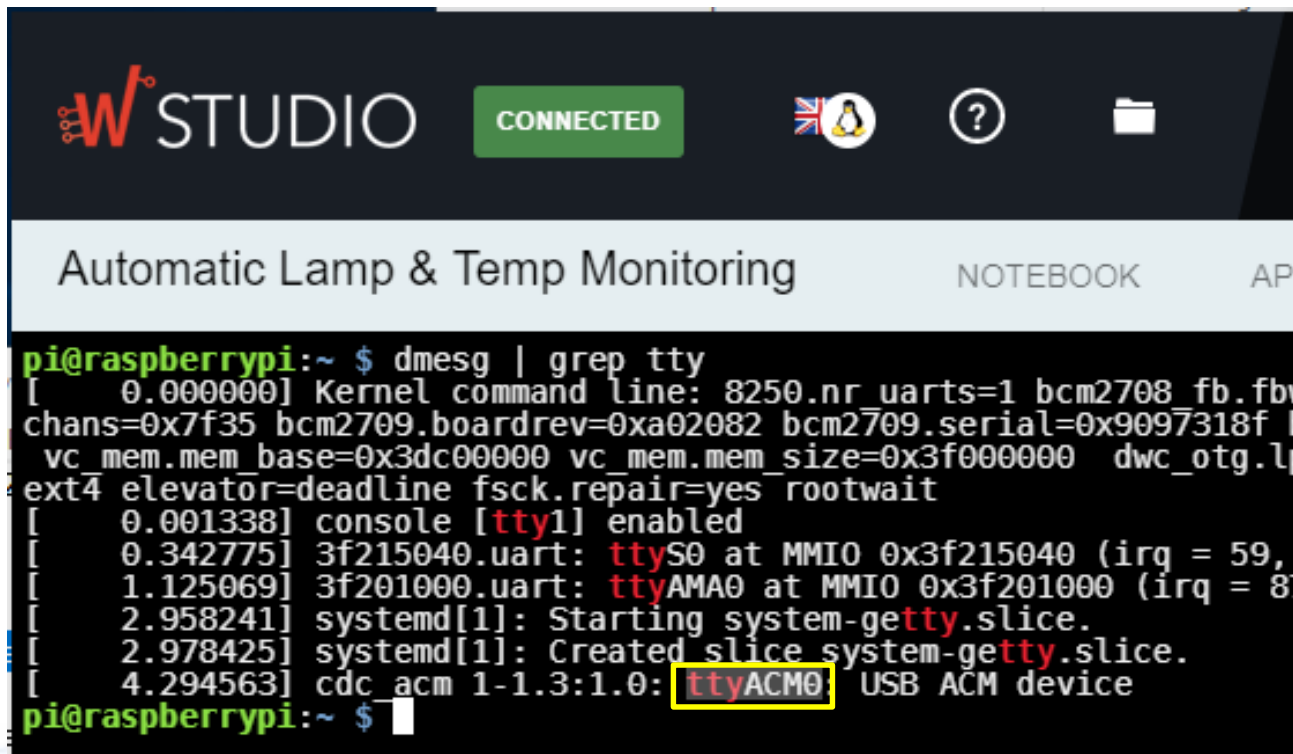
Implementation (2)

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



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.



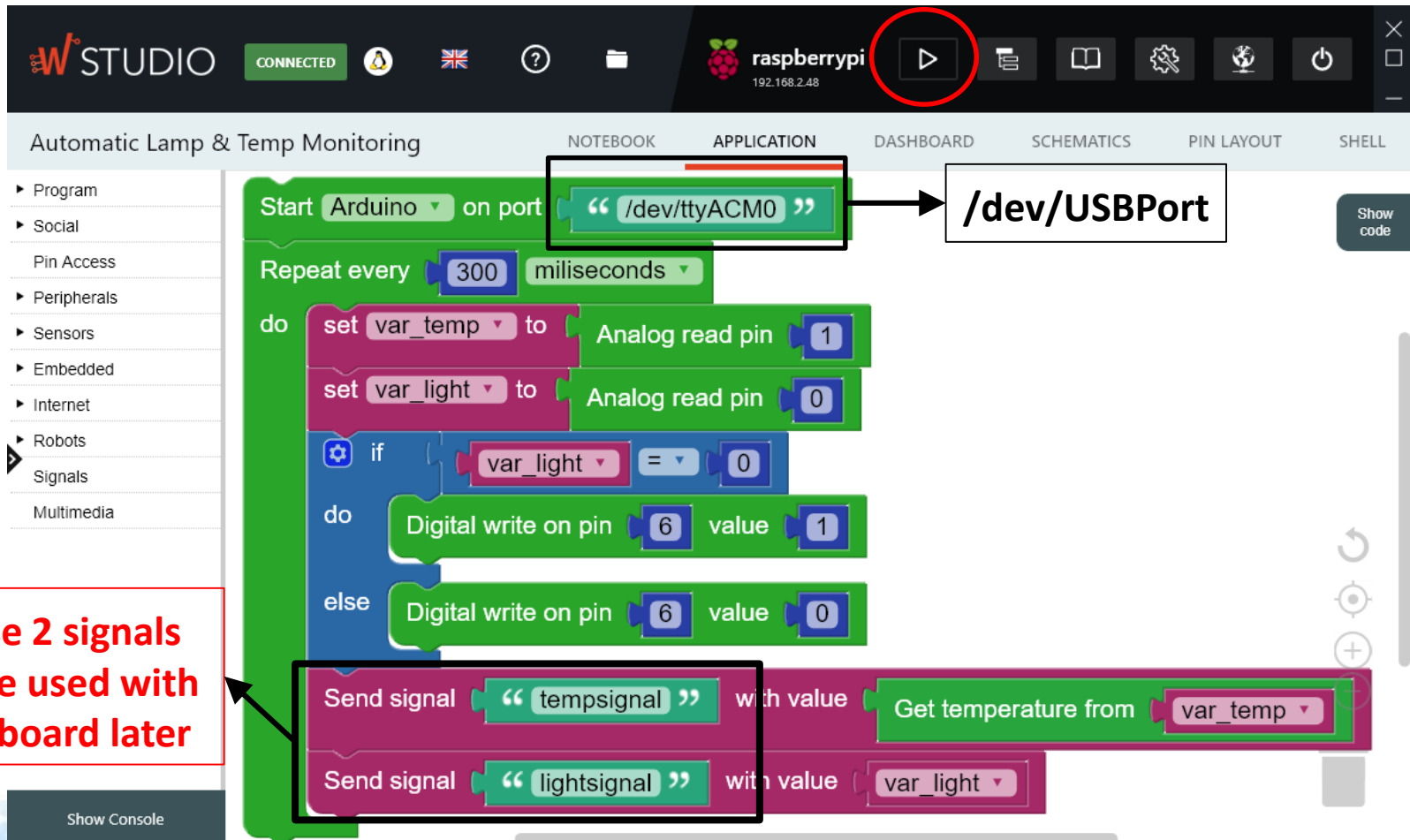
```
WSTUDIO CONNECTED [UK Flag] [Penguin Icon] [Question Mark] [Folder Icon]

Automatic Lamp & Temp Monitoring NOTEBOOK APP

pi@raspberrypi:~ $ dmesg | grep tty
[ 0.000000] Kernel command line: 8250.nr_uarts=1 bcm2708_fb.fbws=
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)

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



The screenshot shows the W Studio interface with the "APPLICATION" tab selected. The code is written in a block-based language. The code starts with "Start Arduino on port" set to "/dev/ttyACM0". This is followed by a "Repeat every 300 milliseconds" loop. Inside the loop, there are two "set" blocks: "set var_temp to Analog read pin 1" and "set var_light to Analog read pin 0". An "if" block follows, checking if "var_light" is equal to 0. If true, it executes "Digital write on pin 6 value 1". If false, it executes "Digital write on pin 6 value 0". After the if block, there are two "Send signal" blocks: "Send signal 'temp signal' with value Get temperature from var_temp" and "Send signal 'light signal' with value var_light".

Annotations in the image:

- A red circle highlights the play button in the top right corner of the W Studio interface.
- A box around the "/dev/ttyACM0" port string has an arrow pointing to a box labeled "/dev/USBPort".
- A red box with the text "These 2 signals will be used with Dashboard later" has an arrow pointing to the "Send signal" blocks.

Information

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

Submenu

- ▶ Program
- ▶ Social
- ▶ Pin Access
- ▶ Peripherals
- ▶ Sensors
- ▶ Embedded
- ▶ Internet
- ▶ Robots
- ▶ Signals
- ▶ Multimedia

if
do
else

else if
if
else

Start **Arduino** on port `“ /dev/ttyACM0 ”` → Embedded > Arduino

Repeat every **300** milliseconds

do

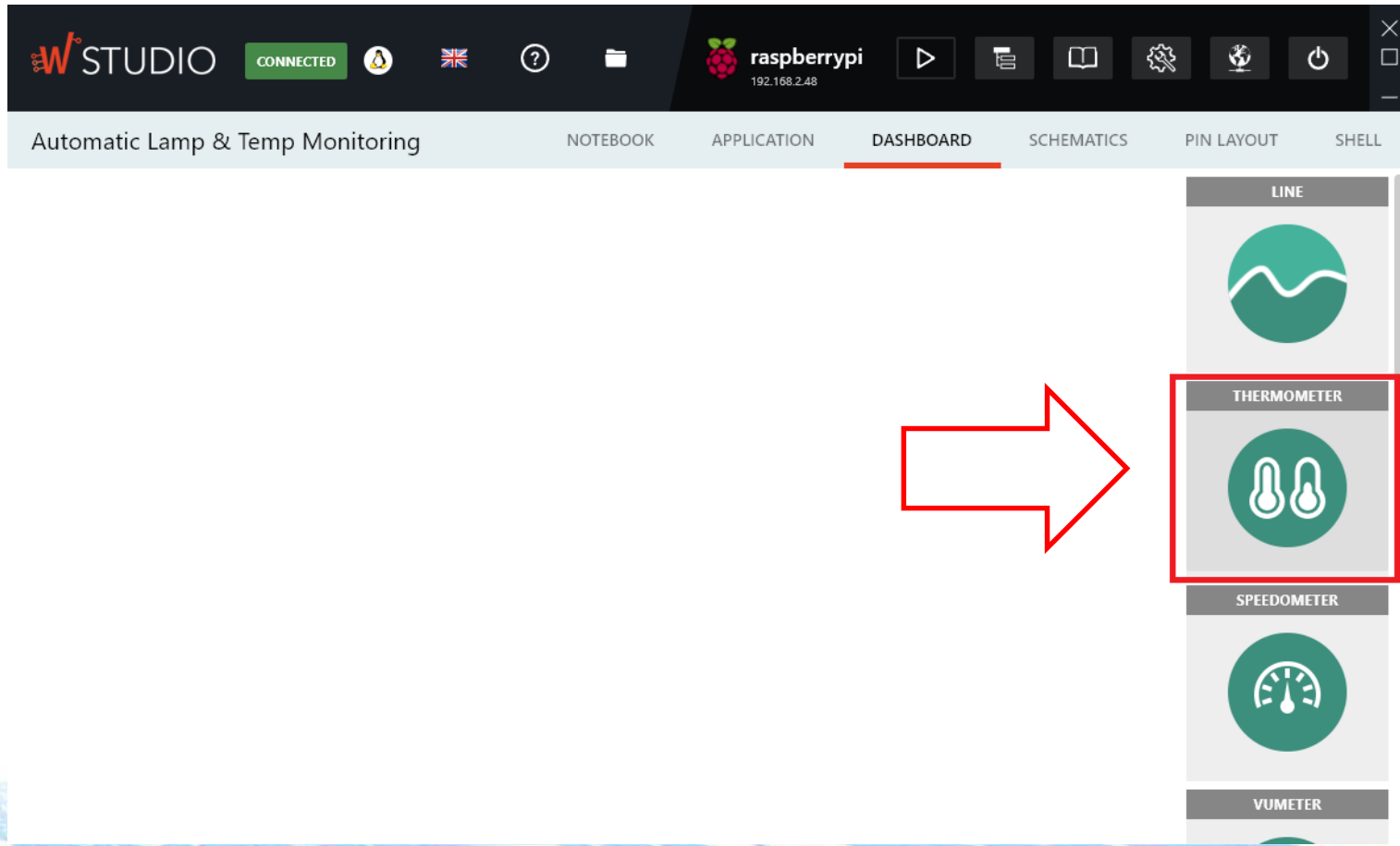
- set **var_temp** to **Analog read pin** **1** → Embedded > Arduino
- set **var_light** to **Analog read pin** **0**
- if **var_light** **=** **0**
 - do **Digital write on pin** **6** value **1**
 - else **Digital write on pin** **6** value **0** → Embedded > Arduino
- Send signal **“ tempsignal ”** with value **Get temperature from** **var_temp** → Sensors > Grove > Temperature Sensor
- Send signal **“ lightsignal ”** with value **var_light**

Show code

CHECKPOINT 2!

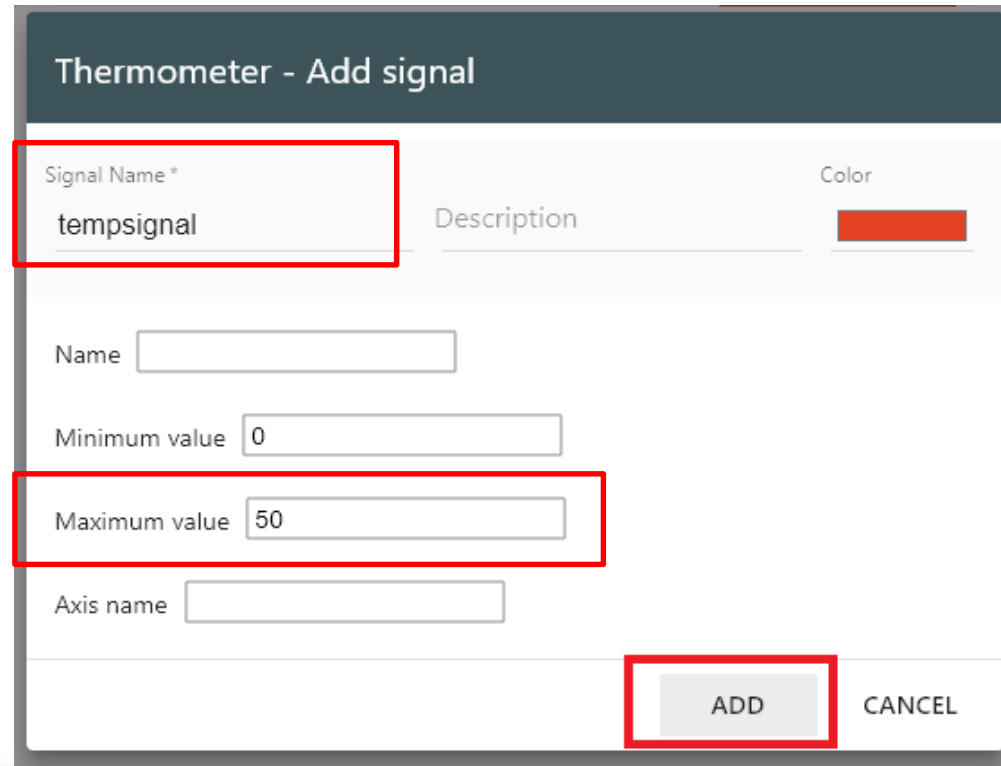
Signals and Dashboard (1)

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.

A screenshot of a web form titled "Thermometer - Add signal". The form has a dark grey header bar with the title. Below the header, there are several input fields. The "Signal Name *" field contains the text "tempsignal" and is highlighted with a red rectangle. To its right is a "Color" field with a red color swatch. Below these is a "Description" field. Further down are fields for "Name", "Minimum value" (containing "0"), "Maximum value" (containing "50" and highlighted with a red rectangle), and "Axis name". At the bottom right, there are two buttons: "ADD" (highlighted with a red rectangle) and "CANCEL".

Thermometer - Add signal

Signal Name *
tempsignal

Color
[Red swatch]

Description

Name
[Empty field]

Minimum value
0

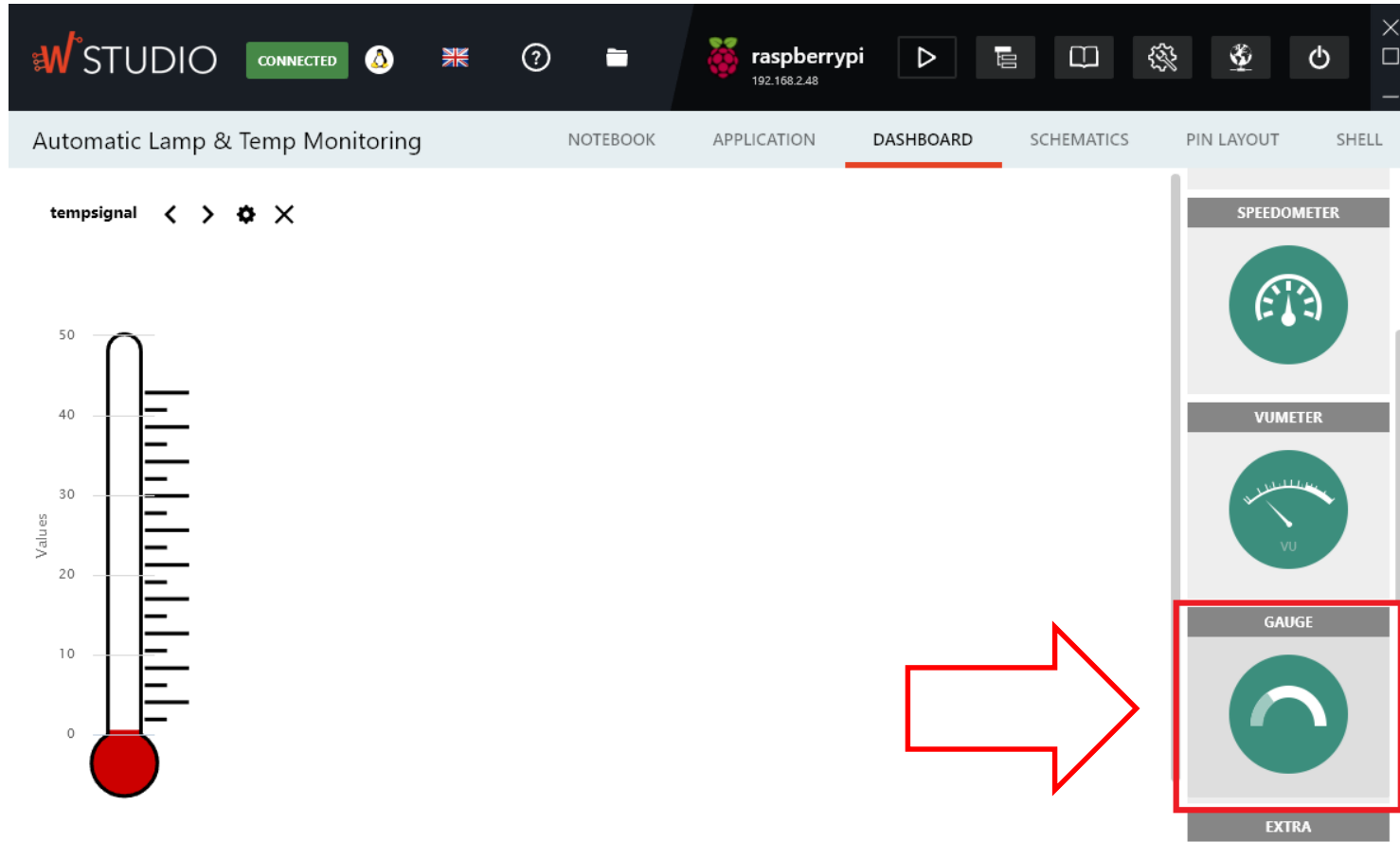
Maximum value
50

Axis name
[Empty field]

ADD CANCEL

Signals and Dashboard (3)

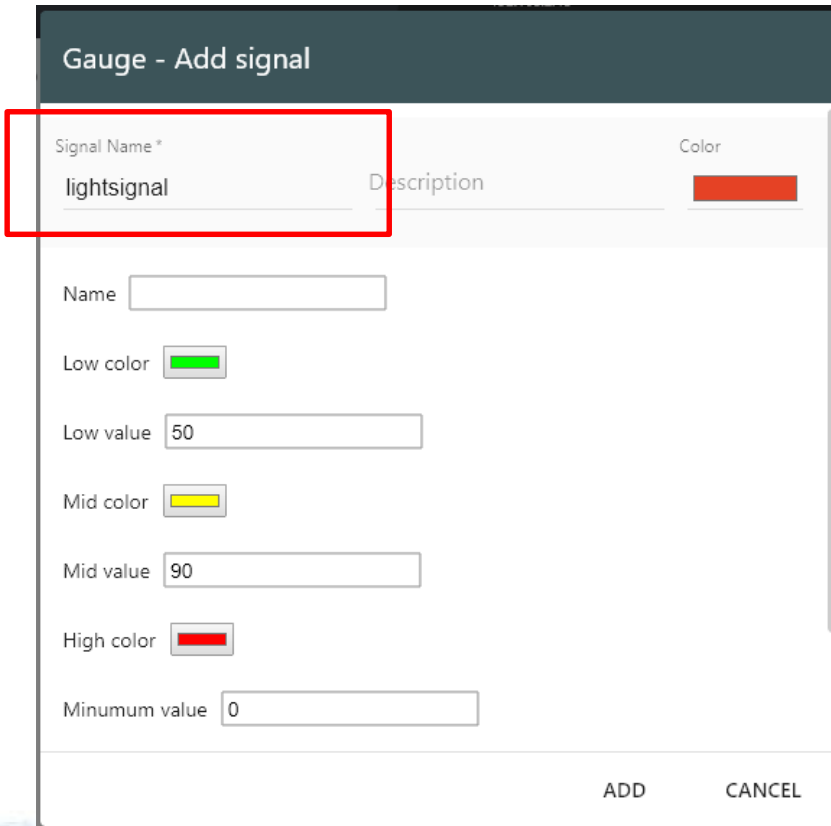
6. On the right panel, click on “Gauge”.



The screenshot displays the W Studio interface. The top bar shows 'STUDIO' with a 'CONNECTED' status and a Raspberry Pi icon labeled 'raspberrypi 192.168.2.48'. Below this, a navigation bar includes 'Automatic Lamp & Temp Monitoring', 'NOTEBOOK', 'APPLICATION', 'DASHBOARD' (highlighted), 'SCHEMATICS', 'PIN LAYOUT', and 'SHELL'. The main area is divided into two panels. The left panel, titled 'temp signal', features a vertical thermometer-style gauge with a red bulb at the bottom and a scale from 0 to 50. The right panel contains a vertical stack of widget options: 'SPEEDOMETER', 'VUMETER', 'GAUGE', and 'EXTRA'. The 'GAUGE' option is highlighted with a red rectangular border, and a large red arrow points from the left panel towards it.

Signals and Dashboard (4)

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

A screenshot of the "Gauge - Add signal" form. The "Signal Name" field is highlighted with a red box and contains the text "lightsignal". Below it, there are fields for "Name", "Low color", "Low value", "Mid color", "Mid value", "High color", and "Minimum value". The "ADD" and "CANCEL" buttons are at the bottom right.

Gauge - Add signal

Signal Name * lightsignal

Description

Color

Name

Low color

Low value 50

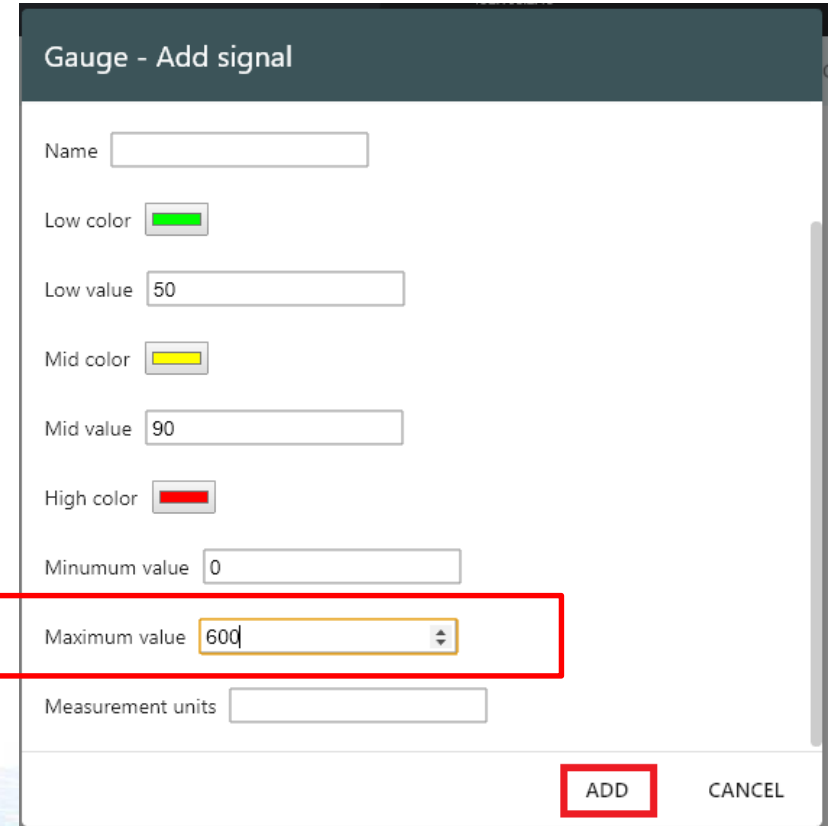
Mid color

Mid value 90

High color

Minimum value 0

ADD CANCEL

A screenshot of the "Gauge - Add signal" form. The "Maximum value" field is highlighted with a red box and contains the value "600". Below it, there is a "Measurement units" field. The "ADD" and "CANCEL" buttons are at the bottom right.

Gauge - Add signal

Name

Low color

Low value 50

Mid color

Mid value 90

High color

Minimum value 0

Maximum value 600

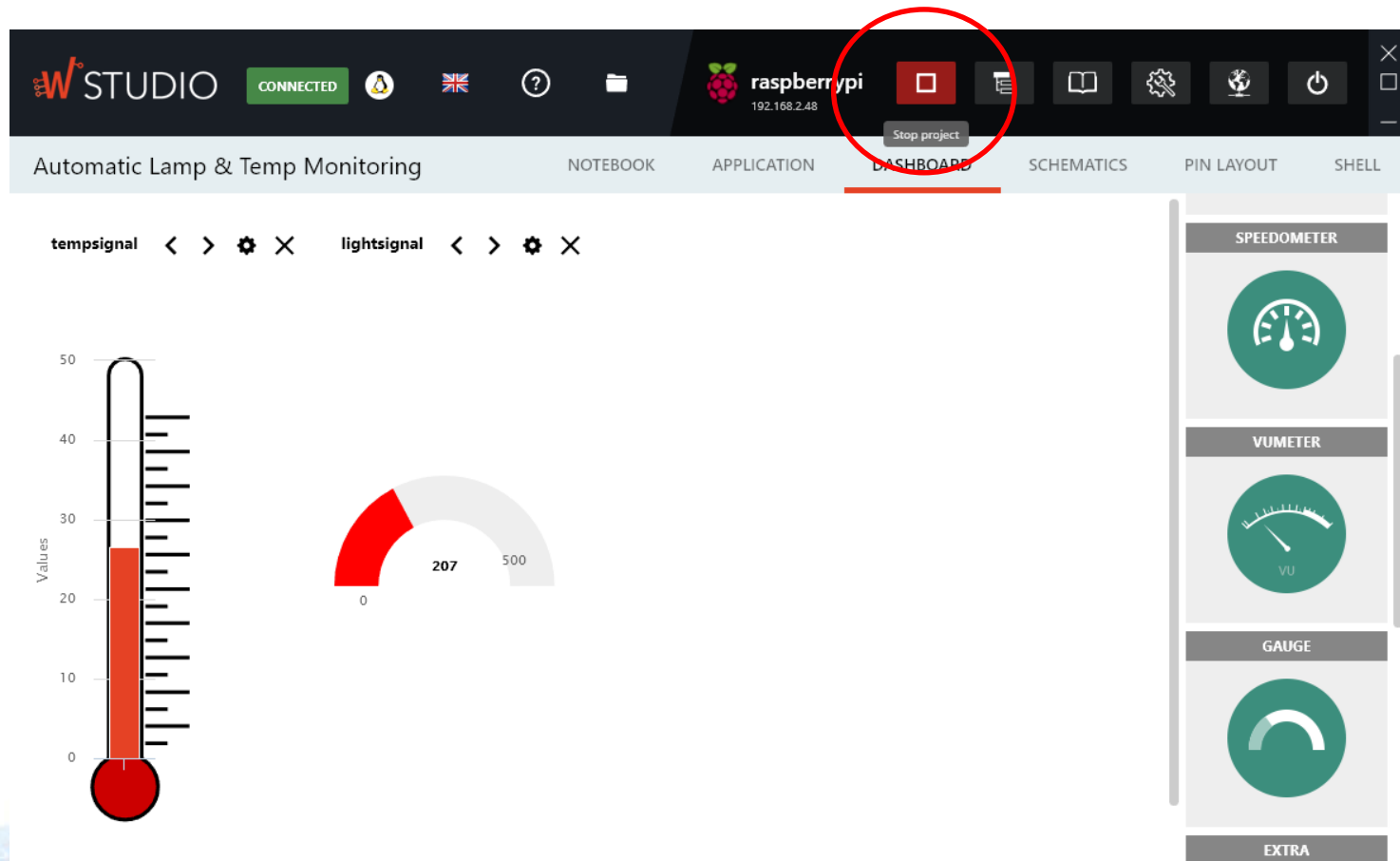
Measurement units

ADD CANCEL

Signals and Dashboard (5)

10. Run the project.

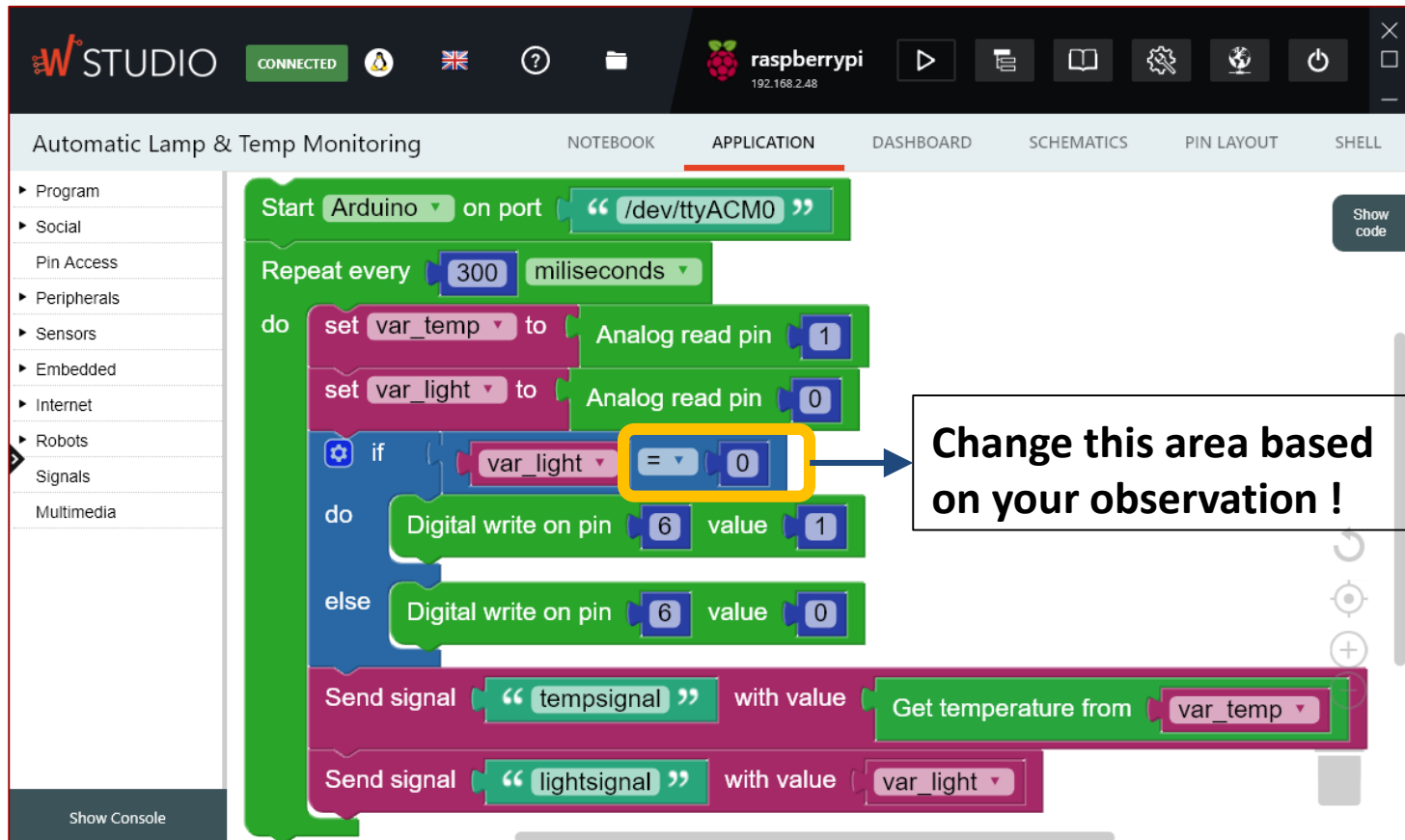
11. Now you can monitor light intensity and temperature from Dashboard.



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.



The screenshot shows the WStudio IDE interface for a Raspberry Pi. The project is titled "Automatic Lamp & Temp Monitoring". The code is written in a block-based language. The following code blocks are visible:

- Start Arduino on port "/dev/ttyACM0"
- Repeat every 300 milliseconds
- 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 "temp signal" with value Get temperature from var_temp
 - Send signal "light signal" with value var_light

A yellow box highlights the condition "var_light = 0" in the "if" statement. A blue arrow points from this box to a text box that says "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”.

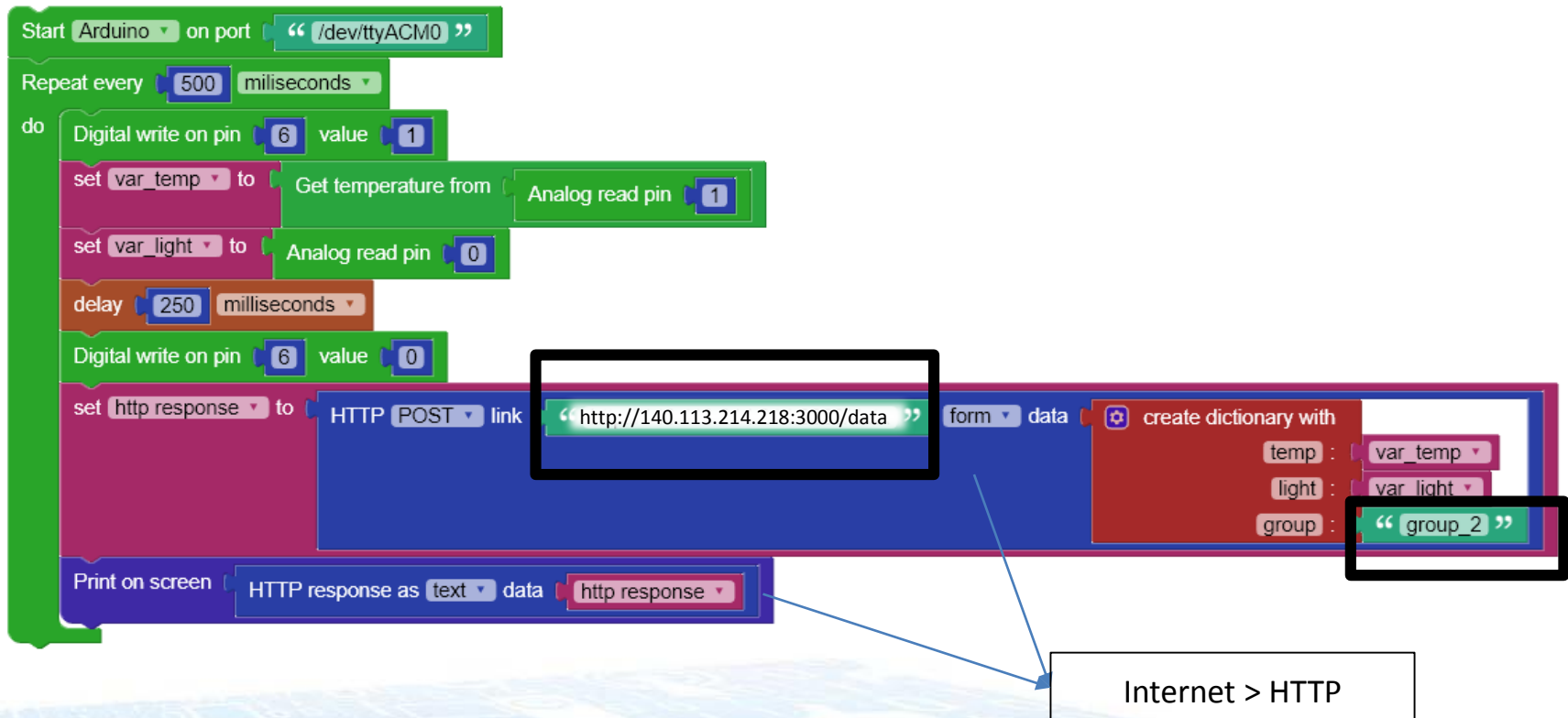
Add new application

Title	Language
<input type="text" value="HTTP_Post_App"/>	<input type="text" value="Visual"/> ▼

CREATECANCEL

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.



The image displays a Scratch script for an Arduino-based IoT application. The script is as follows:

- Start** Arduino on port `"/dev/ttyACM0"`
- Repeat every** 500 milliseconds
- 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

Annotations in the image include:

- A black box around the link `"http://140.113.214.218:3000/data"` with an arrow pointing to a box labeled "Internet > HTTP".
- A black box around the group value `"group_2"` with an arrow pointing to the same "Internet > HTTP" box.

Implementation (3)

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

Hide 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”.

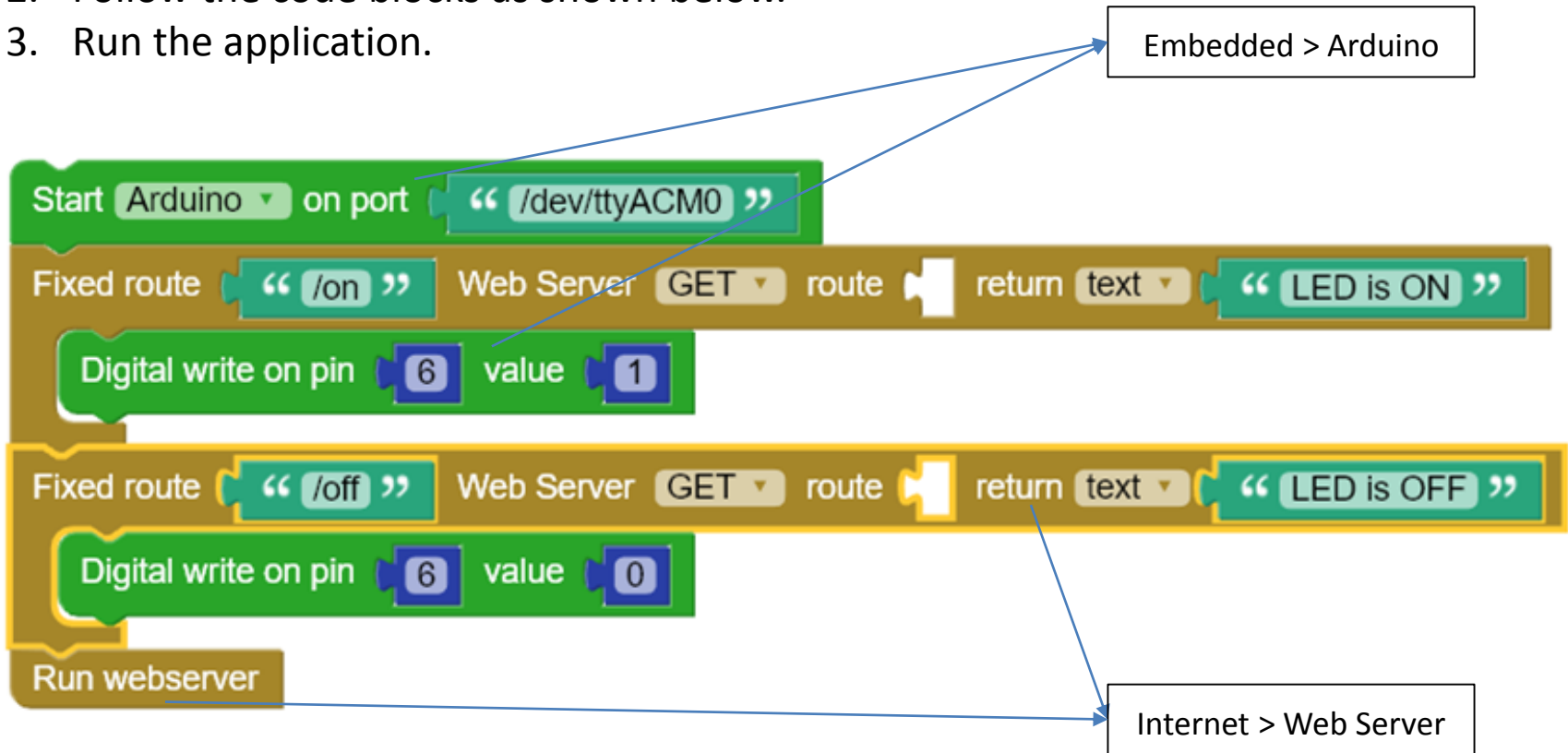
Add new application

Title	Language
<input type="text" value="HTTP_Get_App"/>	<input type="text" value="Visual"/> ▼

CREATECANCEL

Implementation (2)

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



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::6d45:6c3d: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.



CHECKPOINT 6!