



Lab 5: Smart Home Monitoring using OM2M, Node-RED, Wylodrin, and App Inventor 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

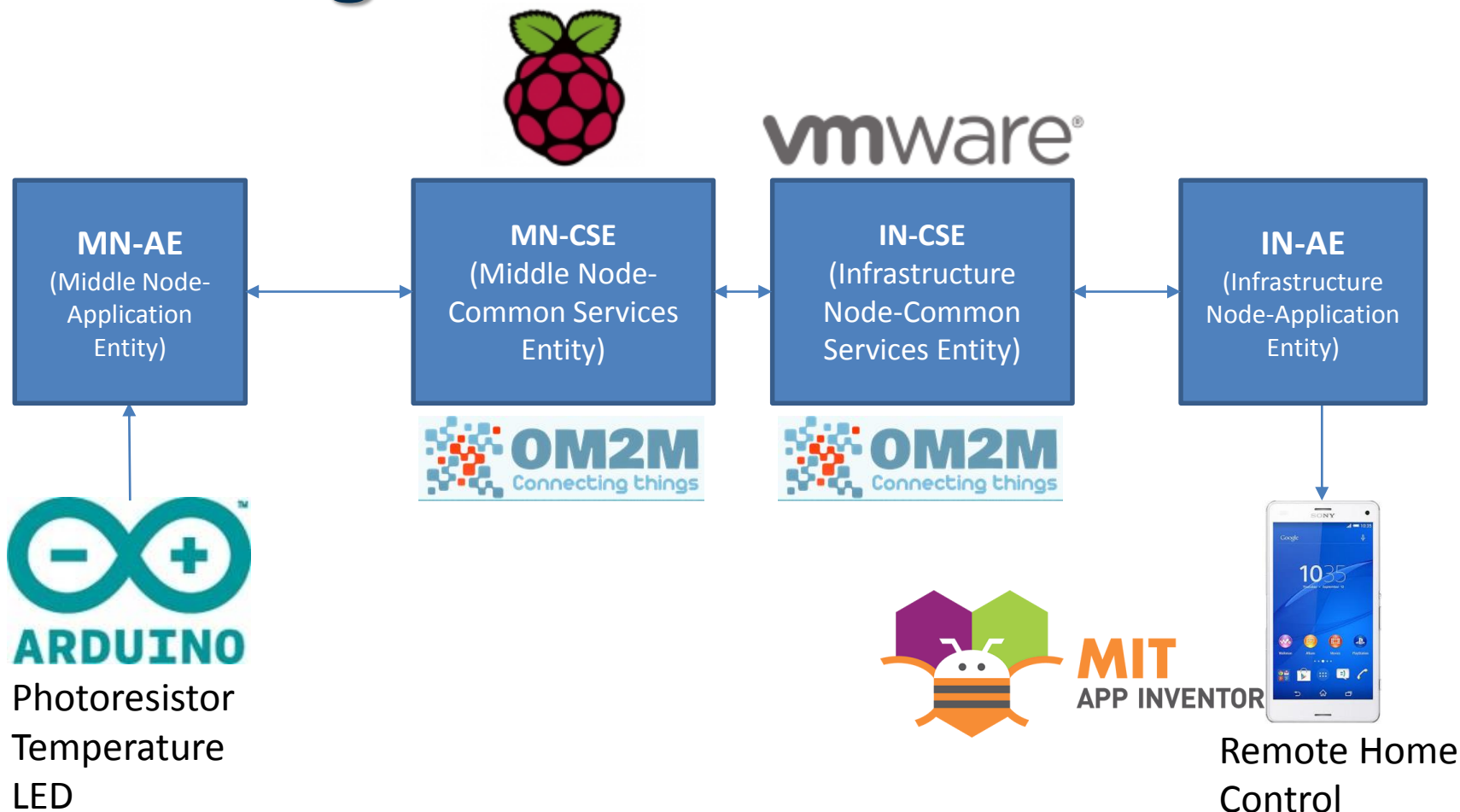
National Chiao Tung University

December 7, 2018

Outline

- High Level Architecture.
- Creating IN-AE and DATA Container to Monitor Smart Home Sensors with Node-Red. *(Checkpoint 1)*
- Modifying the Smart Home MN-AE Notification Handler Webservice using Node-Red. *(Checkpoint 2)*
- Creating a Subscription and Notification Handler for DATA Changes of Monitor Smart Home IN-AE using Node-Red. *(Checkpoint 3)*
- Using Point of Access (PoA) to Control Smart Home Actuators using Node-Red and Wylidrin. *(Checkpoint 4)*
- Overview of App Inventor 2 and Hello World Android Application. *(Checkpoint 5)*
- Creating an Android Application for Smart Home Remote Control App using App Inventor 2. *(Checkpoint 6)*

High Level Architecture



Attention!!

Start your virtual machine and connect your Raspberry Pi, Arduino, and circuit board as you did in Lab 4!

Start the following software in your VM, in this order:

IN-CSE

Wylidrin

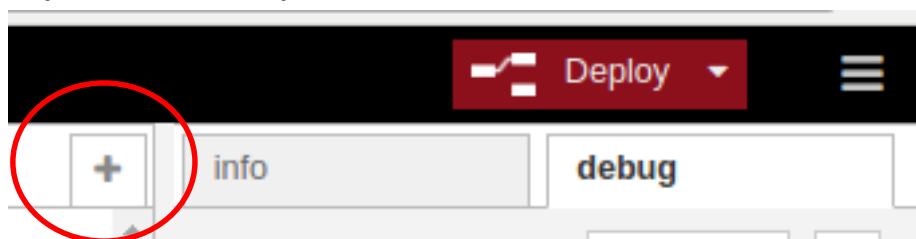
MN-CSE-PI

Node-Red

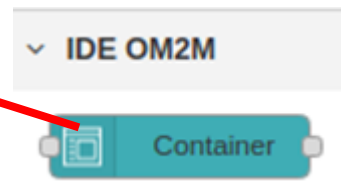
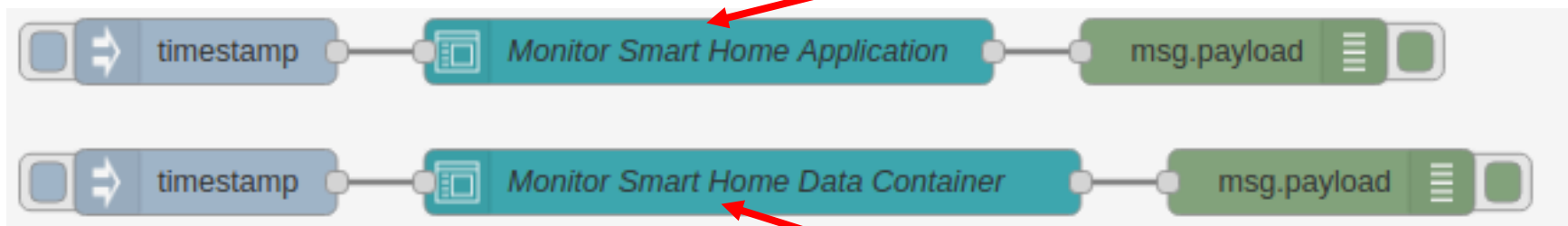
Trigger all the flows in Node-Red until reaching the state of Lab 4

Create IN-AE to monitor smart home sensors

Add a new Flow to your workspace.

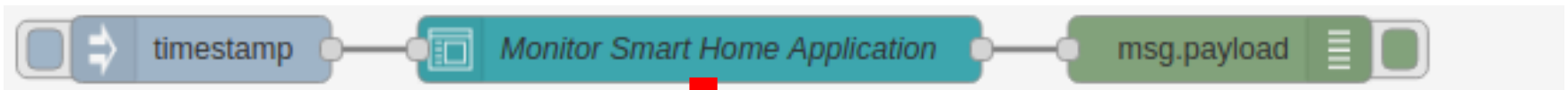


Create the following flows:



Create IN-AE to monitor smart home sensors

Add a new Platform to connect to your IN-CSE which is running in your VM.



Edit Application node

Delete Cancel

node properties

Platform Add new xN_CSE...

Application Add new AE...

Point of Access poa

Announce

Cancel Add

Platform IN-CSE

URLBase http://192.168.1.51:8080/~in-cse/in-name

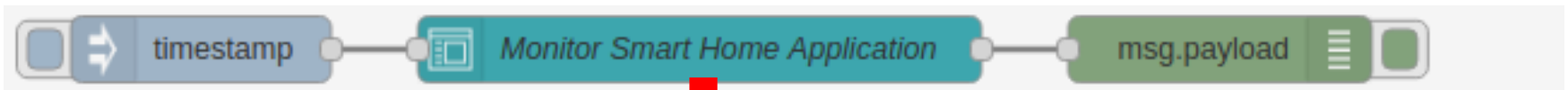
Username admin

Password

URLBase: http://**YOUR.VM.IP.ADDRESS**:8080/~in-cse/in-name
Username: admin
Password: admin

Create IN-AE to monitor smart home sensors

Set “MONITOR_SMART_HOME” as the AppID.



Edit Application node

Delete Cancel

node properties

Platform Add new xN_CSE...

Application Add new AE...

Point of Access poa

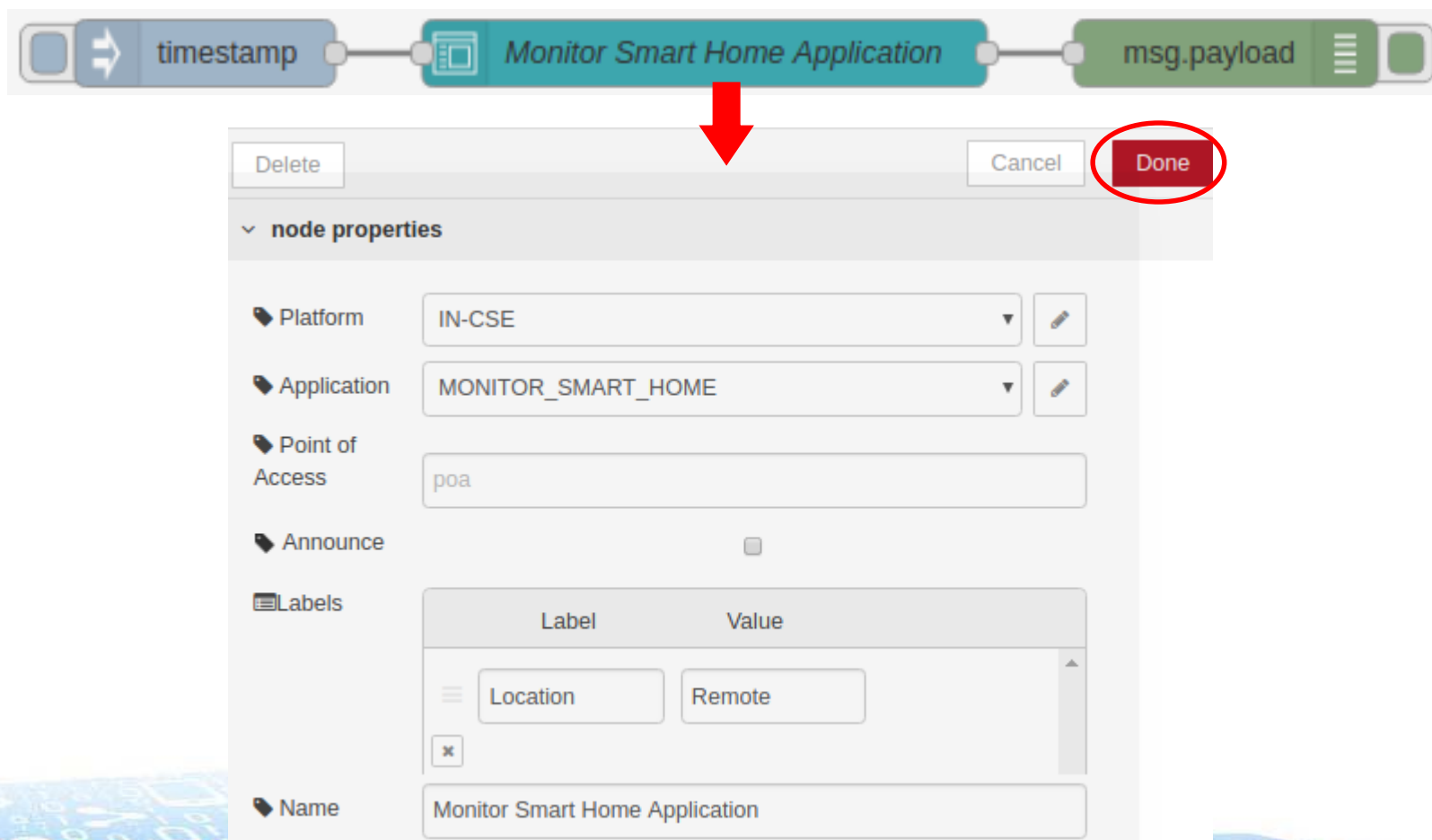
Announce

Cancel Add

AppID MONITOR_SMART_HOME

Create IN-AE to monitor smart home sensors

Complete the remaining attributes as shown in the picture below.



The screenshot displays the configuration interface for the 'Monitor Smart Home Application' node. The node is part of a flow that includes a 'timestamp' input and a 'msg.payload' output. The 'node properties' section is expanded, showing the following attributes:

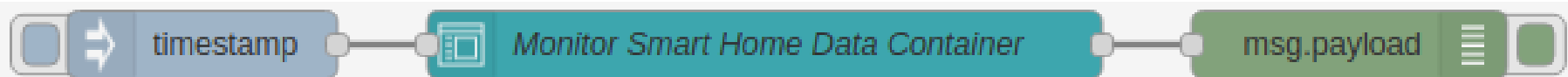
- Platform:** IN-CSE
- Application:** MONITOR_SMART_HOME
- Point of Access:** poa
- Announce:** ☐
- Labels:** A table with two columns: Label and Value.

Label	Value
Location	Remote
- Name:** Monitor Smart Home Application

The 'Done' button is highlighted with a red circle, indicating the completion of the configuration.

Create IN-AE to monitor smart home sensors

Create a DATA container.



Edit Container node

Delete

Cancel

Done

node properties

Platform

IN-CSE

Application

MONITOR_SMART_HOME

Container

DATA

Name

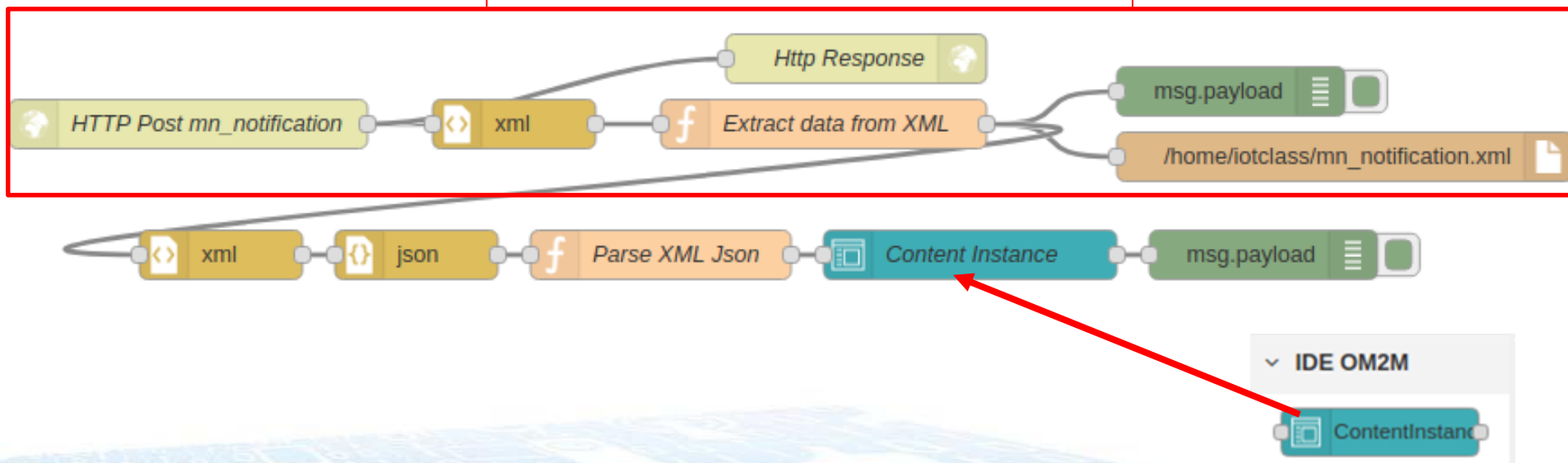
Monitor Smart Home Data Container

CHECKPOINT 1!

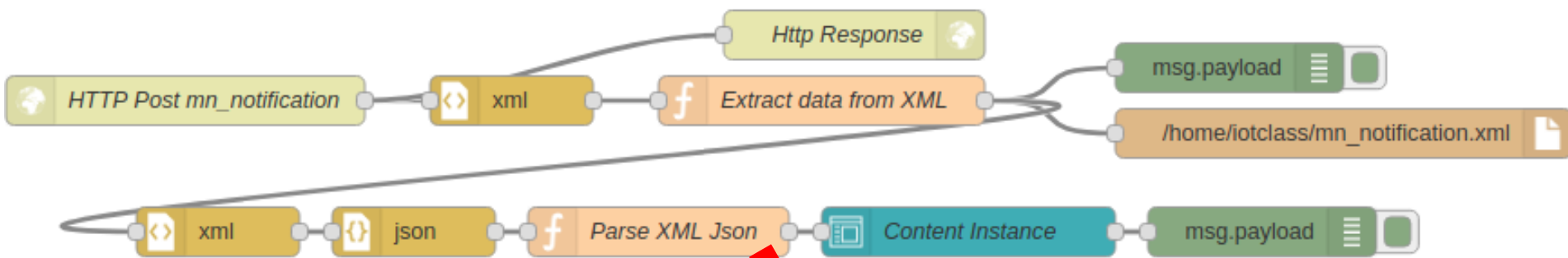
Modify HTTP Post mn_notification

We are going to combine parts of different flows we developed so far. The purpose is to extract only the value of photoresistor from the notification message, and store this value in the DATA Container of the MONITOR_SMART_HOME application.

Do not need to be modified!



Modify HTTP Post mn_notification



CODE (Copy & Paste in Node Red)

```
var data = JSON.parse(msg.payload);
var object1 = data.obj;
var str1 = object1.str;

var photores = str1[0];
var photores_$ = photores.$;
var photoresistor = photores_$.val;

var data = [{
  type:'str',
  label:'photoresistor',
  value: photoresistor
}];
msg.externalInput = JSON.stringify(data);
return msg;
```

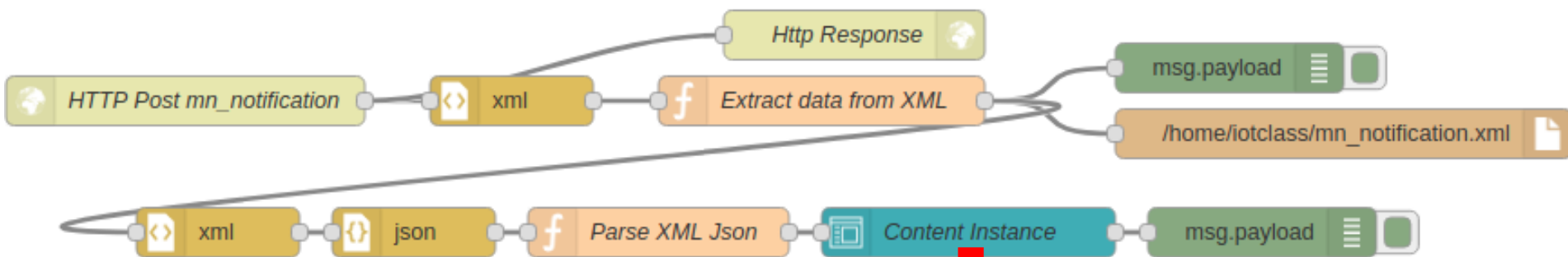
node properties

Name

Function

```
1 var data = JSON.parse(msg.payload);
2 var object1 = data.obj;
3 var str1 = object1.str;
4
5 var photores = str1[0];
6 var photores_$ = photores.$;
7 var photoresistor = photores_$.val;
8
9 var data = [{
10   type:'str',
11   label:'photoresistor',
12   value: photoresistor
13 }];
14 msg.externalInput = JSON.stringify(data);
15 return msg;
```

Modify HTTP Post mn_notification



Edit ContentInstance node

Delete

Cancel

Done

node properties

Platform

IN-CSE

Application

MONITOR_SMART_HOME

Container

DATA

Labels

Type

Name

Value

Modify HTTP Post mn_notification

Push the switch button in your breadboard. Verify the photoresistor value is stored into the DATA container of the MONITOR_SMART_HOME application.

<http://192.168.1.51:8080/~in-cse/cin-816072674>

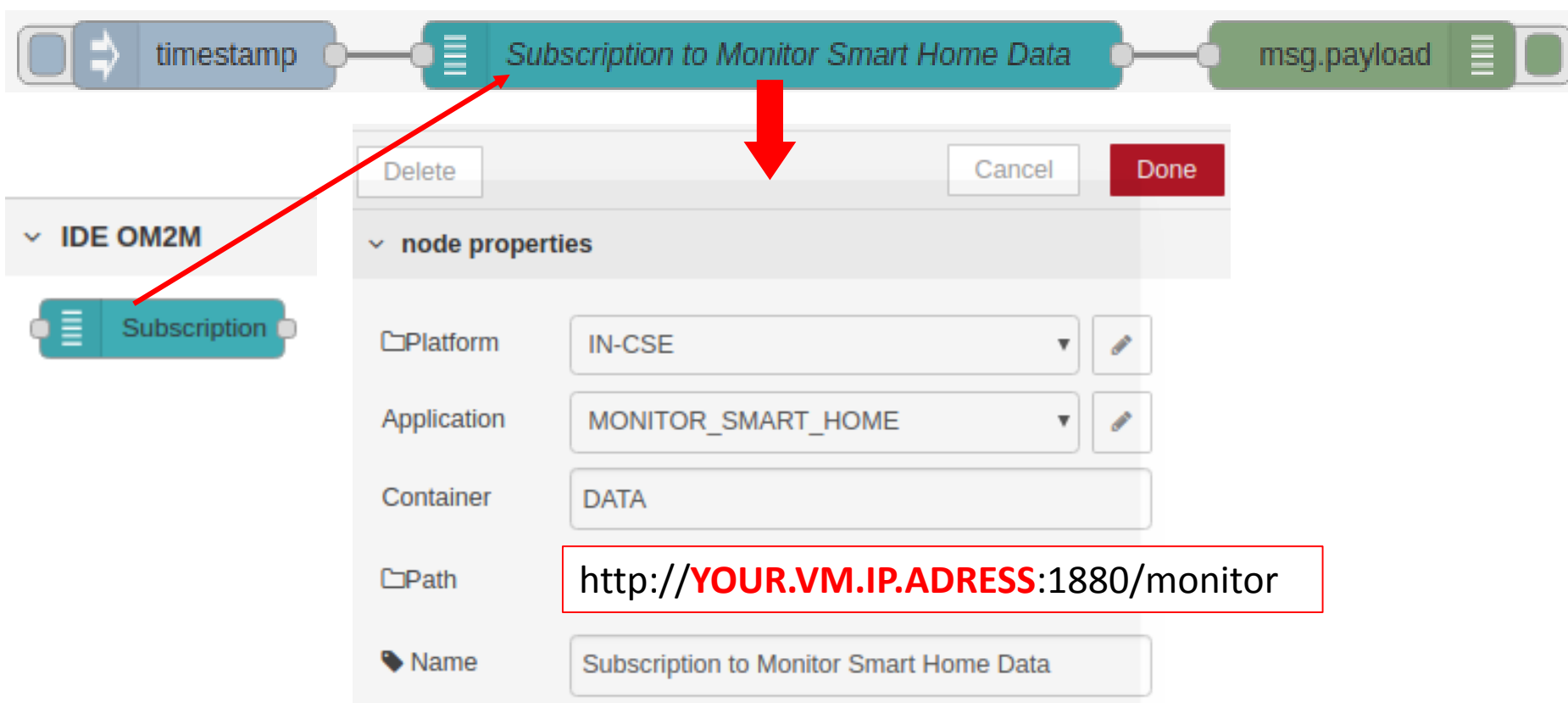
```
– in-name
  |
  |– acp_admin
  |– MONITOR_SMART_HOME
      |
      |– DATA
          |
          |– cin_816072674
  |– mn-name-pi
```

Attribute	Value				
rn	cin_816072674				
ty	4				
ri	/in-cse/cin-816072674				
pi	/in-cse/cnt-107923819				
ct	20180705T192218				
lt	20180705T192218				
st	0				
cnf	message				
cs	93				
con	<table><tr><th>Attribute</th><th>Value</th></tr><tr><td>photoresistor</td><td>75.0</td></tr></table>	Attribute	Value	photoresistor	75.0
Attribute	Value				
photoresistor	75.0				

CHECKPOINT 2!

Create a subscription to Monitor Smart Home DATA container

Create a Subscription to DATA container according to the picture below.



The screenshot displays the IDE OM2M interface. At the top, a sequence of components is shown: a 'timestamp' component, a 'Subscription to Monitor Smart Home Data' component, and a 'msg.payload' component. A red arrow points from the 'Subscription' component in the left sidebar to the 'Subscription to Monitor Smart Home Data' component in the main workspace. Another red arrow points from the 'Subscription to Monitor Smart Home Data' component to the 'node properties' panel below. The 'node properties' panel contains the following fields:

- Platform: IN-CSE
- Application: MONITOR_SMART_HOME
- Container: DATA
- Path: <http://YOUR.VM.IP.ADRESS:1880/monitor>
- Name: Subscription to Monitor Smart Home Data

Create a subscription to Monitor Smart Home DATA container

Trigger all the flows and verify the correct execution via the OM2M GUI.

<http://192.168.1.51:8080/~in-cse/sub-916>

– in-name

```

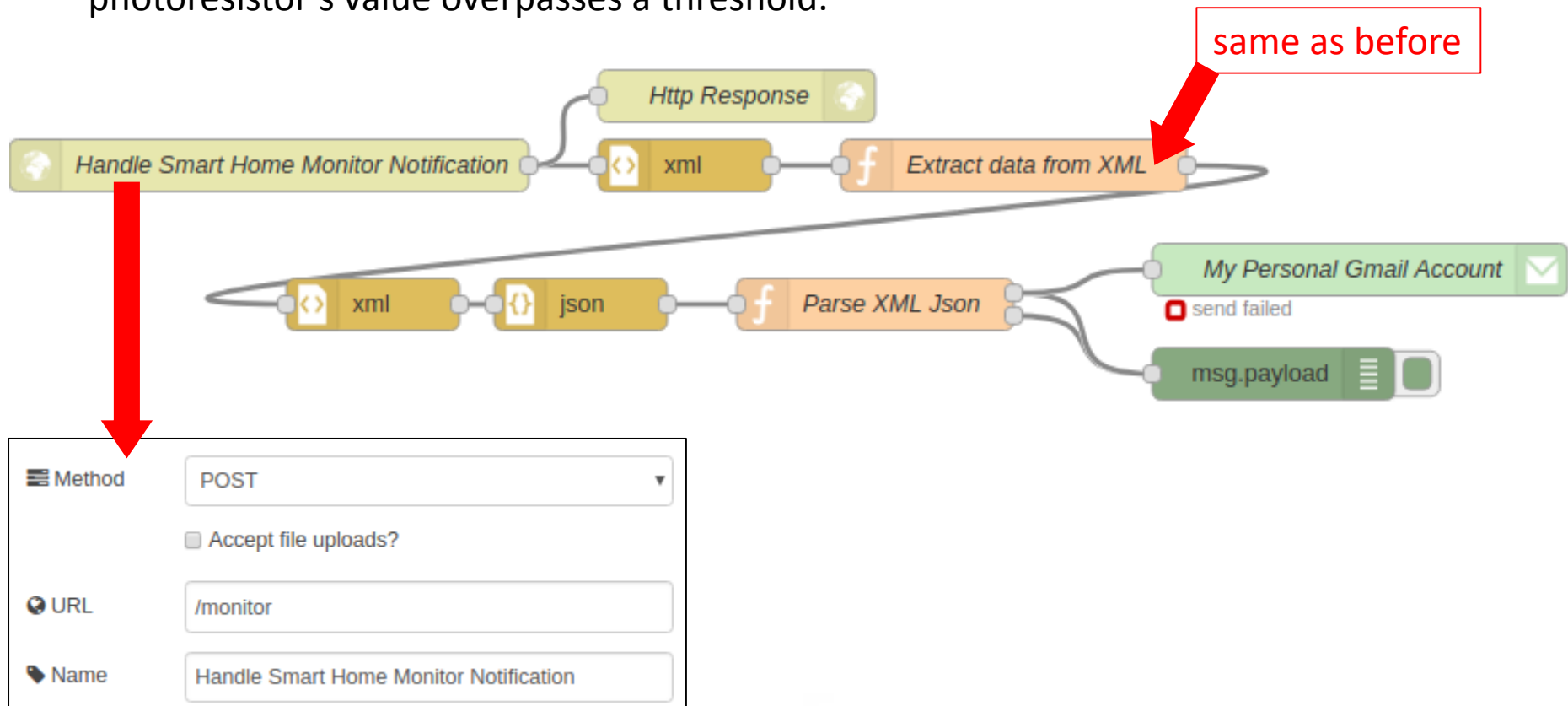
  |— acp_admin
  |— MONITOR_SMART_HOME
  |   |— DATA
  |       |— SUBSCRIPTION
  |— mn-name-pi
  
```



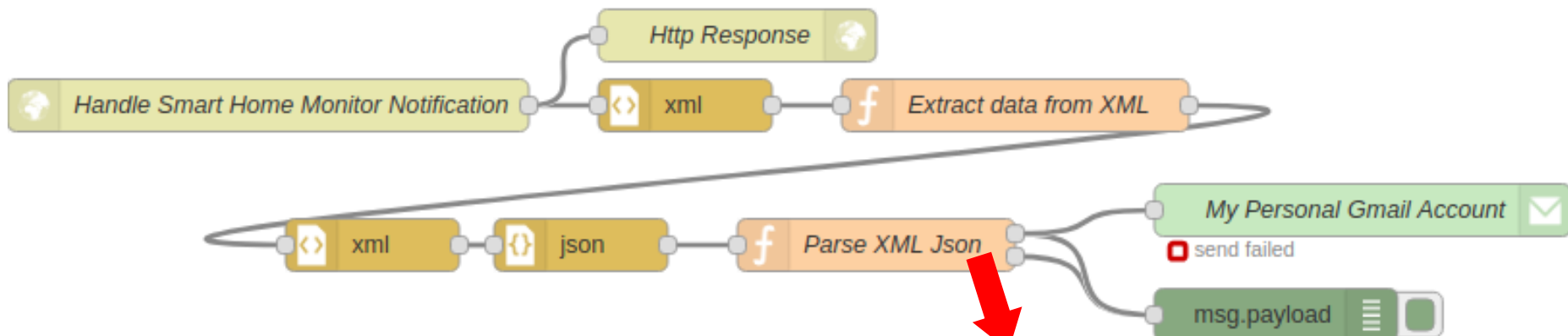
Attribute	Value
rn	SUBSCRIPTION
ty	23
ri	/in-cse/sub-916764550
pi	/in-cse/cnt-559137139
ct	20180705T165934
lt	20180705T165934
acpi	<div>AccessControlPolicyIDs</div> <div>/in-cse/acp-419143047</div>
nu	<ul style="list-style-type: none"> http://192.168.1.51:1880/monitor
nct	2

Create the “monitor” webservice

The purpose of this webservice is to send an notification email to a user if the photoresistor's value overpasses a threshold.



Create the “monitor” webservice



CODE (Copy & Paste in Node Red)

```

var data = JSON.parse(msg.payload);
var object1 = data.obj;
var str1 = object1.str;
var photores = str1[0];
var photores_$ = photores.$;
var photoresistor = photores_$.val;

msg.payload = "Photoresistor: " +
photoresistor;
msg.topic = "Emergency"

if (photoresistor >=200 ) {
    return [ msg, null ];
} else {
    return [ null, msg ];
}
  
```

Parse XML Json

Function

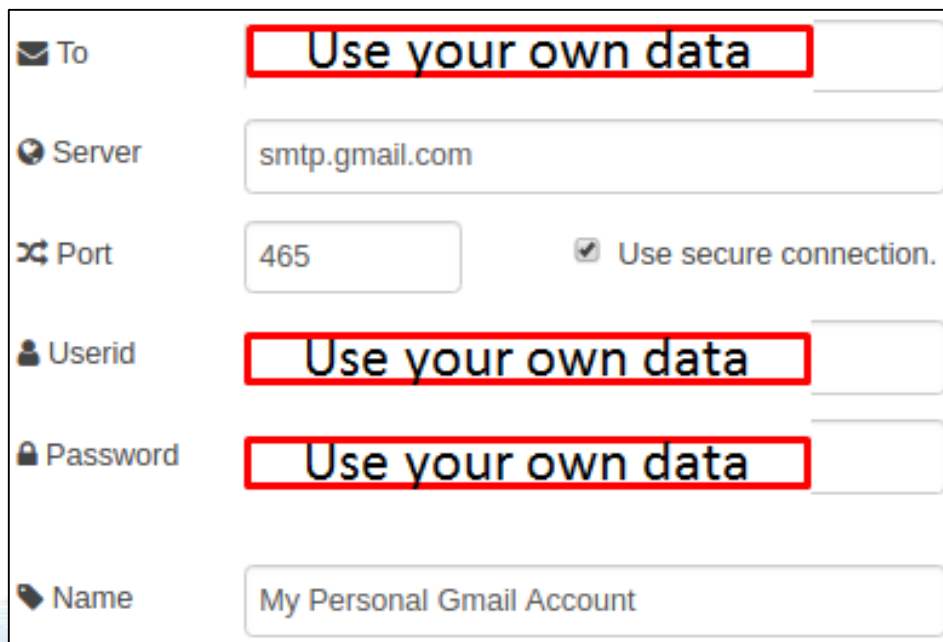
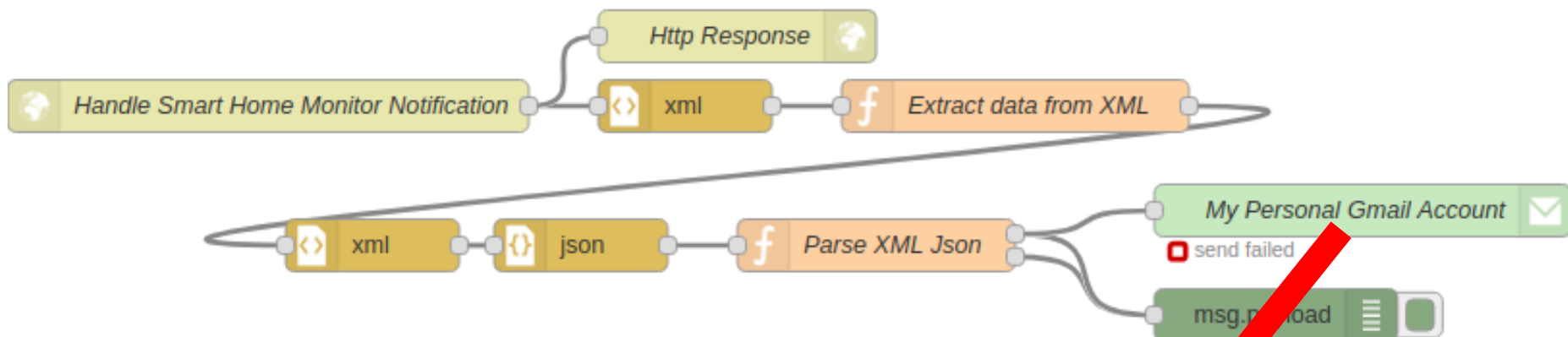
```

1 var data = JSON.parse(msg.payload);
2 var object1 = data.obj;
3 var str1 = object1.str;
4
5 var photores = str1[0];
6 var photores_$ = photores.$;
7 var photoresistor = photores_$.val;
8
9 msg.payload = "Photoresistor: " + photoresistor;
10 msg.topic = "Emergency"
11
12 if (photoresistor >=200 ) {
13     return [ msg, null ];
14 } else {
15     return [ null, msg ];
16 }
  
```

Outputs

2

Create the “monitor” webservice



A red arrow points from the "My Personal Gmail Account" node in the flowchart to this configuration form.

To	<input type="text" value="Use your own data"/>
Server	<input type="text" value="smtp.gmail.com"/>
Port	<input type="text" value="465"/> <input checked="" type="checkbox"/> Use secure connection.
Userid	<input type="text" value="Use your own data"/>
Password	<input type="text" value="Use your own data"/>
Name	<input type="text" value="My Personal Gmail Account"/>

Try the whole system

The email notification may fail due to Google's security settings, somehow you can see the debug window of Node-Red to make sure the email was up to be sent.

```
info debug
all nodes
7/5/2018, 8:54:18 PM node: 2378a2d2.94e8fe
msg.payload : string[542]
"<?xml version="1.0"
encoding="UTF-8"?><m2m:cin
xmlns:m2m="http://www.onem2m.org/xml
xmlns:hd="http://www.onem2m.org/xml/
rn="cin_160492984"><ty>4</ty>
<ri>/in-cse/cin-160492984</ri>
<pi>/in-cse/cnt-107923819</pi>
<ct>20180705T205418</ct>
<lt>20180705T205418</lt>
<st>0</st> <cnf>message</cnf>
<cs>94</cs> <con>&lt;obj>&lt;str
name="photoresistor">
val="312.0">/in="obix:
out="obix:Nil">
is="retrieve">/>&lt;/obj>
</con></m2m:cin>"
```

```
7/5/2018, 8:54:18 PM node: f9c323df.f8fef
Emergency : msg.payload : string[20]
"Photoresistor: 312.0"
7/5/2018, 8:54:19 PM node:
josedelabastida@gmail.com
msg : error
▶ "Error: Invalid login: 534-
5.7.14
<https://accounts.google.com/signin/
sarp=1&sc=1&plt=AKgnsbvLw534-
5.7.14 l9Mx-
xGJPfi3rUedZppqZV5_ZgyNUCYR1omIpzVSR
RgsEq3lpWNOMiPgSujG5f3sTmV8Pw534-
5.7.14
23Tz2cnvxgf2uyawBqp0aEnyMYU4wcwGEEs-
Yq8VUs78U8Jz-
ykapys0xo8rWR2J8tNY80w534-5.7.14
qG-
UZLZK_J0ja6RnxTUuzR8YnwHAawM_37Tm6w0
5.7.14
po9dlVMZc54xfSzmpRacXuPewcE10>
Please log in via your web browser
and 534-5.7.14 then try
again 534-5.7.14 login more
```

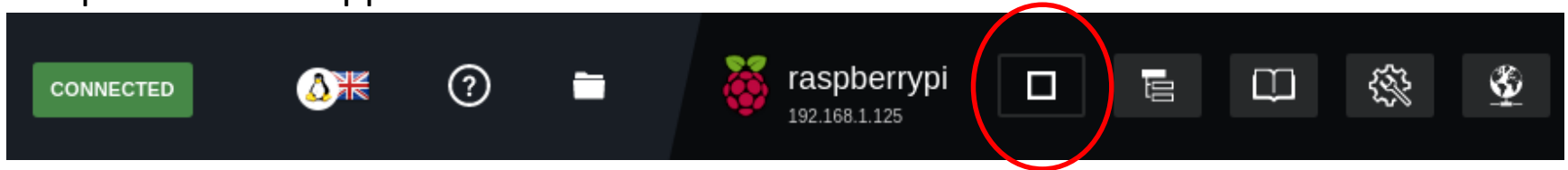
CHECKPOINT 3!

Attention!!

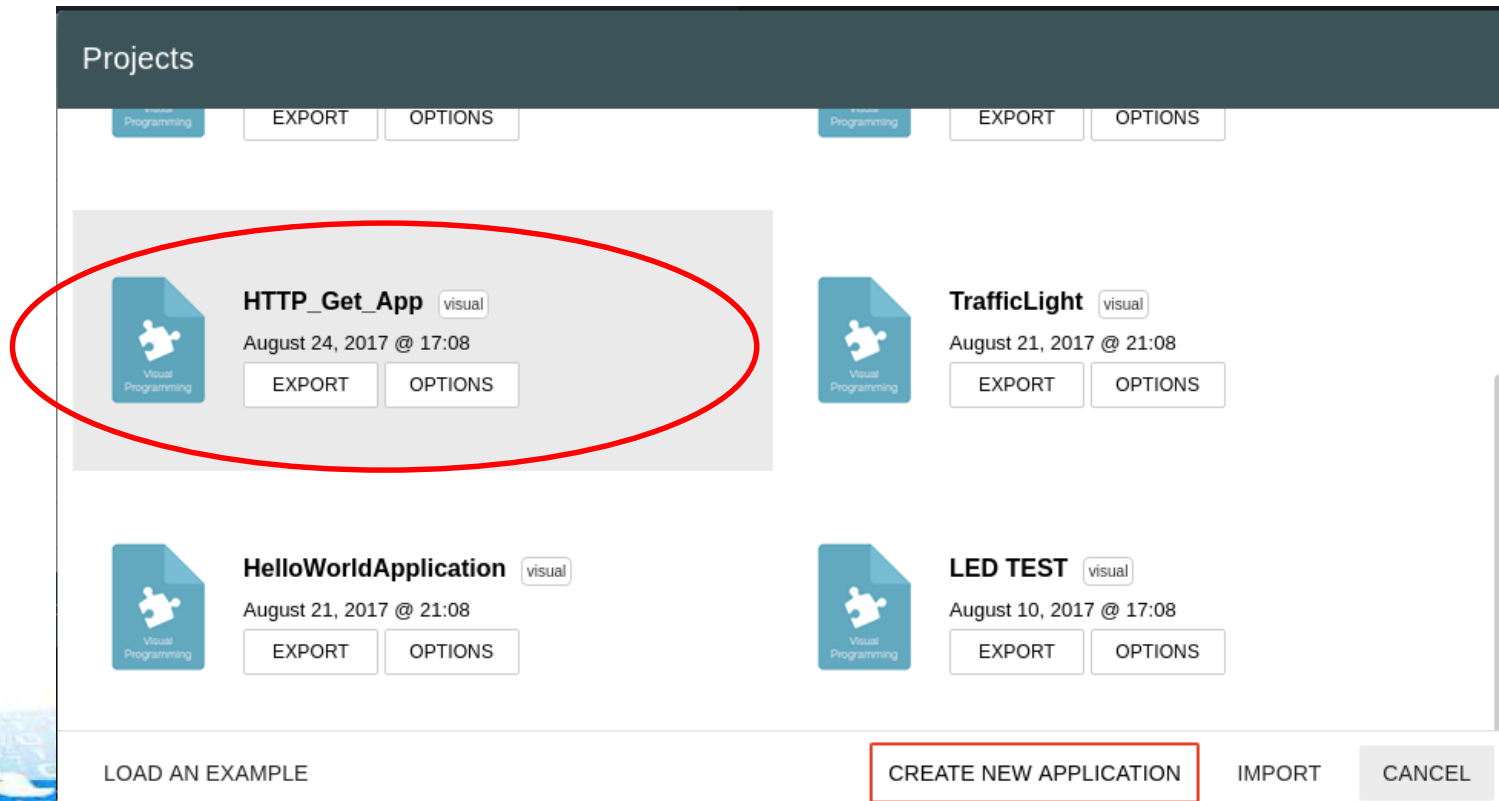
After demo checkpoint 3 to TA, please stop the current Wyliodrin App, and open the application "HTTP_Get_App" you created on Lab 2.

Open HTTP_Get_App

Stop the current app.

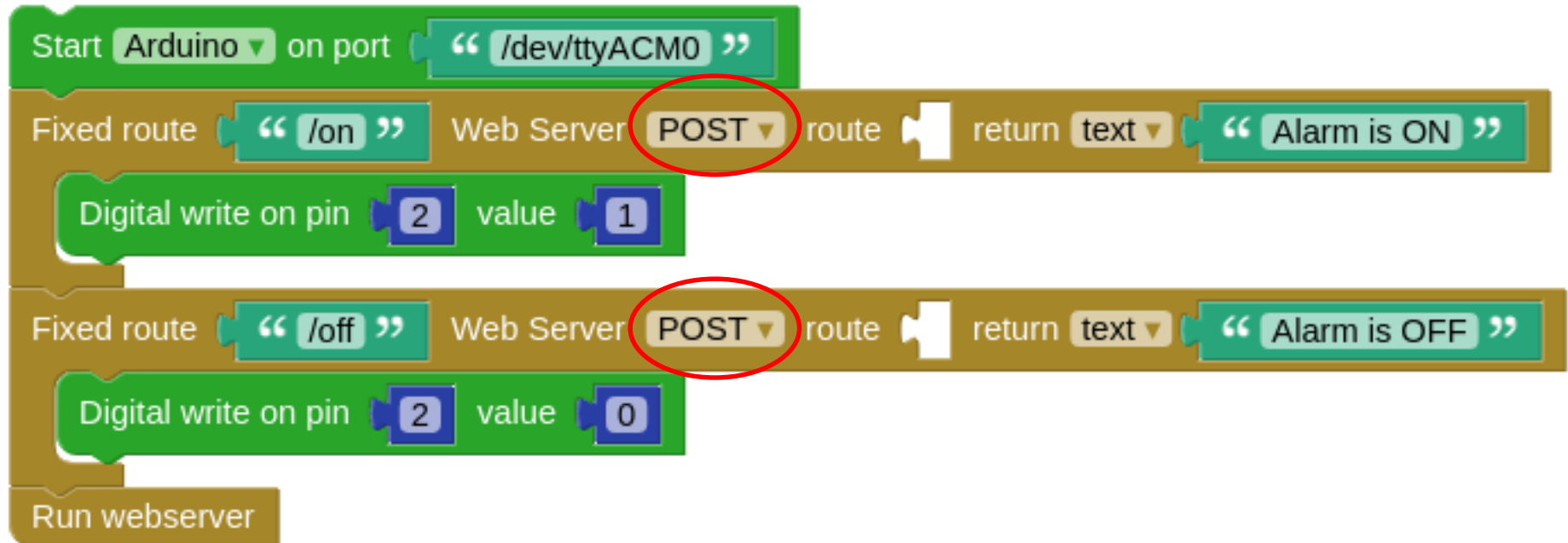


Open HTTP_Get_App



Modify and Run HTTP_Get_App

Change, Web Server from GET to POST.



The image shows a Scratch script for an Arduino web server. The script starts with a green flag click event block: "Start Arduino on port /dev/ttyACM0". This is followed by two "Fixed route" blocks for a "Web Server". The first route is for "/on" and is configured with a "POST" method (circled in red), returning the text "Alarm is ON" and triggering a "Digital write on pin 2" with a value of 1. The second route is for "/off" and is also configured with a "POST" method (circled in red), returning the text "Alarm is OFF" and triggering a "Digital write on pin 2" with a value of 0. The script concludes with a "Run webserver" block.

Run the application

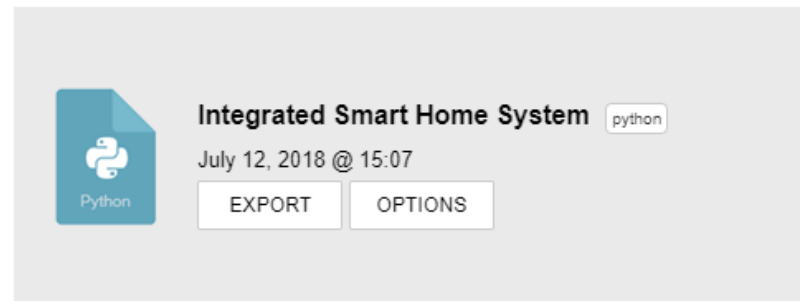


The image shows a Raspberry Pi terminal window. On the left, a green button says "CONNECTED". In the center, there are icons for a user, a flag, a question mark, and a folder. On the right, the text "raspberrypi" and the IP address "192.168.1.125" are displayed. A yellow circle highlights a play button icon in the bottom right corner of the terminal window.

Integrated System [Optional] (1)

To integrate the smart home and monitoring applications, we have written a single python script that achieves these tasks using threading. Threading is not provided in visual programming Wylidrin.

Import and load the given python project “Integrated Smart Home System” from Wylidrin.



Attention! : This task is optional and can be skipped. However it can substitute previous slide application “Modify and Run HTTP_Get_App”

Integrated System [Optional] (2)

- Change the ip_address in the url written inside **loopButton()** function in line 77 into your VM's ip_address

```
75 ▾ def loopButton():  
76 ▾     if (digitalRead (15)) == 0:  
77 ▾         http_response = requests.post('http://192.168.1.51:1880/postSmartHomeData',  
78 ▾         print((http_response.text))  
79 ▾         Timer(0.15, loopButton).start()  
80 ▾
```

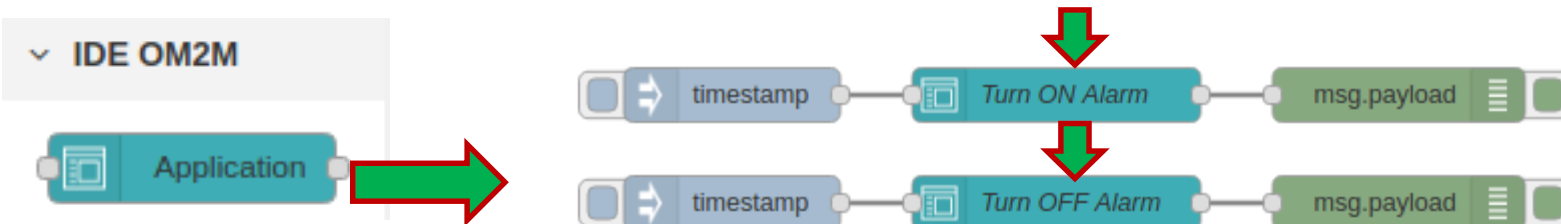
- Run the application



Attention! : This task is optional and can be skipped. However it can substitute previous slide application “Modify and Run HTTP_Get_App”

Create Two Applications in Node-Red

In the MN-CSE flow, create two additional MN-AE. Use the IDE-OM2M **Application** node:



node properties

Platform: MN-CSE-PI

Application: ALARM_ON

Point of Access: <http://YOUR.RASPBERRY.IP.ADDR:5000/on>

Announce: ☒

Label	Value
Location	pi

Name: Turn ON Alarm

node properties

Platform: MN-CSE-PI

Application: ALARM_OFF

Point of Access: <http://YOUR.RASPBERRY.IP.ADDR:5000/off>

Announce: ☒

Label	Value
Location	pi

Name: Turn OFF Alarm

Please deploy and trigger your applications.

Verify the two new applications are created in OM2M GUI

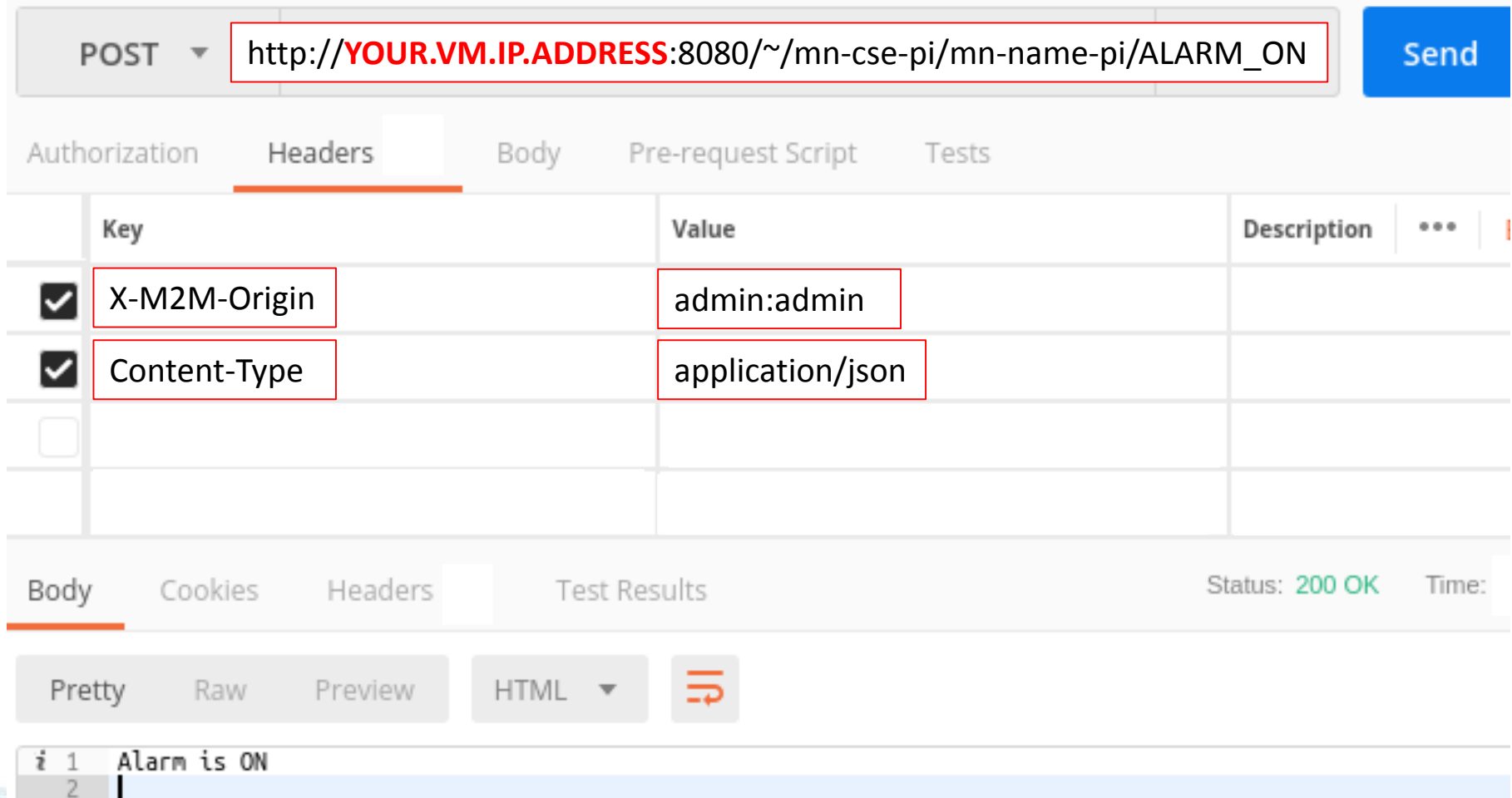
OM2M CSE Resource Tree

<http://localhost:8080/~mn-cse-pi/CAE917367037>

```
– mn-name-pi
  |
  |– acp_admin
  |– SMART_HOME
  |– ALARM_ON
  |– ALARM_OFF
  |
  |– in-name
```

Use Postman to test PoA (1)

Please verify that your LED is ON




POST `http://YOUR.VM.IP.ADDRESS:8080/~mn-cse-pi/mn-name-pi/ALARM_ON` Send

Authorization Headers Body Pre-request Script Tests

	Key	Value	Description	...
<input checked="" type="checkbox"/>	X-M2M-Origin	admin:admin		
<input checked="" type="checkbox"/>	Content-Type	application/json		
<input type="checkbox"/>				

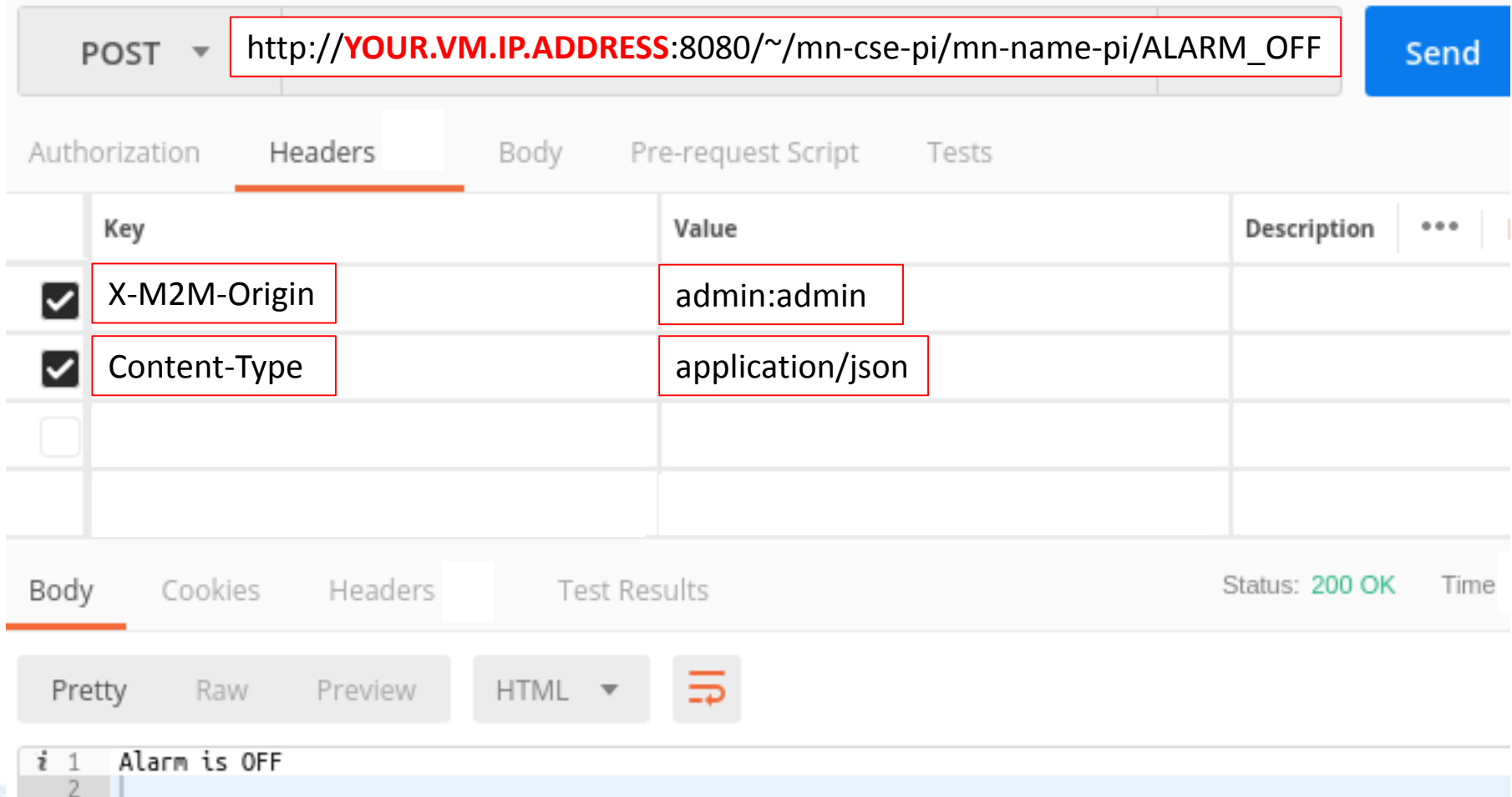
Body Cookies Headers Test Results Status: 200 OK Time:

Pretty Raw Preview HTML 

```
i 1 Alarm is ON
2 |
```

Use Postman to test PoA (2)

Please verify that your LED is OFF




POST `http://YOUR.VM.IP.ADDRESS:8080/~mn-cse-pi/mn-name-pi/ALARM_OFF` Send

Authorization Headers Body Pre-request Script Tests

	Key	Value	Description	...
<input checked="" type="checkbox"/>	X-M2M-Origin	admin:admin		
<input checked="" type="checkbox"/>	Content-Type	application/json		
<input type="checkbox"/>				

Body Cookies Headers Test Results Status: 200 OK Time

Pretty Raw Preview HTML 

i 1 Alarm is OFF
2

CHECKPOINT 4!

CREATING A REMOTE CONTROL APP IN APP INVENTOR 2

App Inventor 2 Overview (1)

- Cloud-based tool for developing Android Apps.
- Originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT).
- Apps can be built right in your web browser → Visual Programming.
- Url of the service: ai2.appinventor.mit.edu.
- Works with your google account.
- Focus on the Application not on Coding.



App Inventor 2 Overview (2)

- General Features:
 - A **designer**, in which program's *components* are specified. This includes visible components, such as buttons and images, which are placed on a simulated screen, and non-visible components, such as sensors and web connections.
 - A **blocks editor**, in which the program's logic is created.
 - A **compiler** based on the Kawa language framework.
 - **Real-time deployment and debugging** on a connected Android device.
 - **Real-time deployment and debugging** on a provided Emulator.
 - **Deployment via .apk** using QR code or direct download.

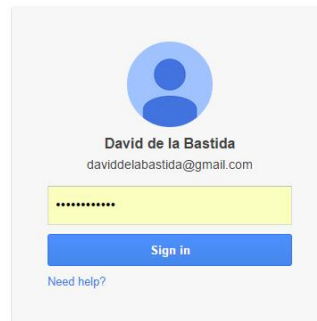
App Inventor 2 Login (1)

1. Type <http://ai2.appinventor.mit.edu>
2. Login using your google account.
3. Allow permission to MIT AppInventor Version 2 to access your google account (if required).



One account. All of Google.

Sign in with your Google Account

A screenshot of the Google sign-in interface. It features a blue circular profile picture placeholder, the name "David de la Bastida", and the email address "davidelabastida@gmail.com". Below this is a yellow password field with masked characters "*****". A blue "Sign in" button is positioned below the password field. At the bottom left of the form, there is a link that says "Need help?".

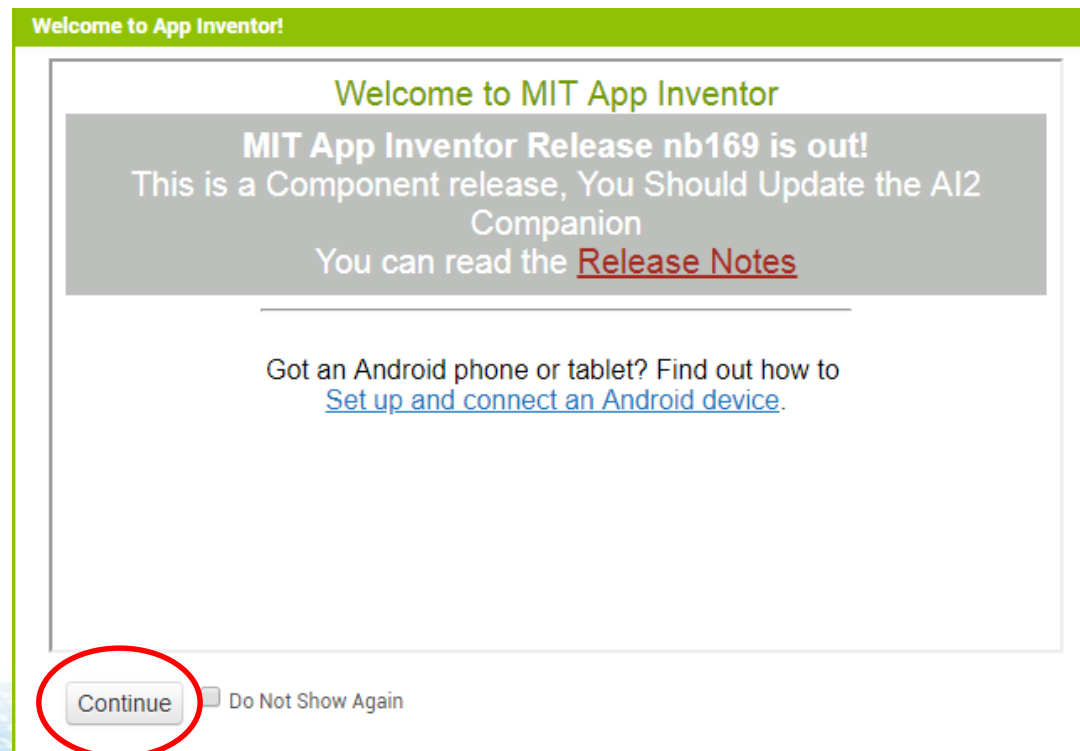
[Sign in with a different account](#)

One Google Account for everything Google



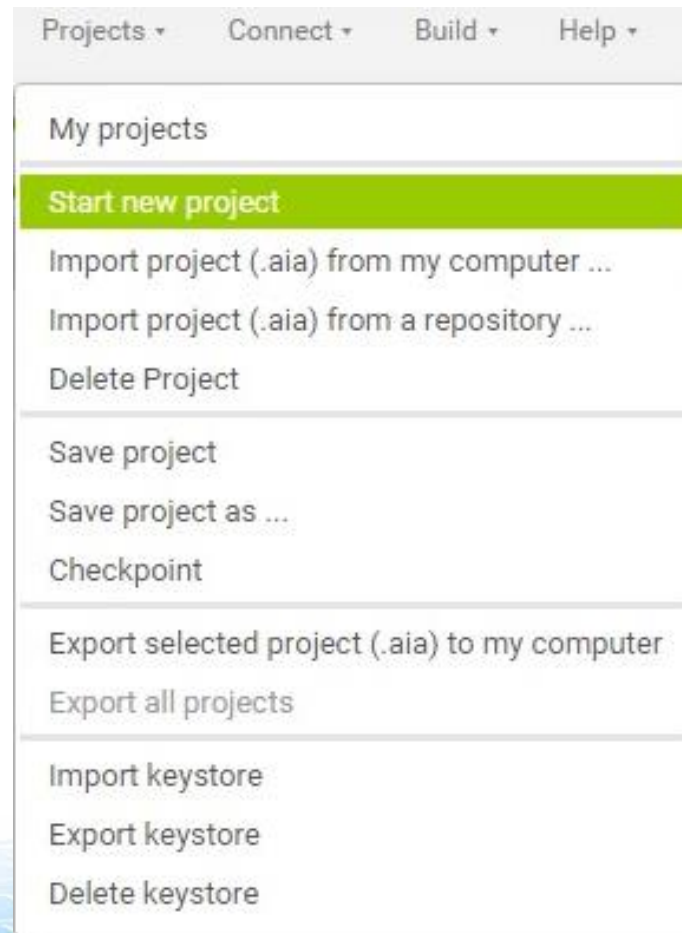
App Inventor 2 Login (2)

1. Type <http://ai2.appinventor.mit.edu>
2. Login using your google account.
3. Allow permission to MIT AppInventor Version 2 to access your google account (if required).
4. Click “continue”, uncheck “do not show again” (if required).



Hello World Application (1)

1. From menu “My projects” select “Start new project”.



Hello World Application (2)

2. Provide a name to your project. (In this example we use “HelloWorldApp”).

A screenshot of the "Create new App Inventor project" dialog box. The dialog has a green header bar with the title "Create new App Inventor project". Below the header, there is a label "Project name:" followed by a text input field containing the text "HelloWorldApp". At the bottom of the dialog, there are two buttons: "Cancel" on the left and "OK" on the right.

Create new App Inventor project

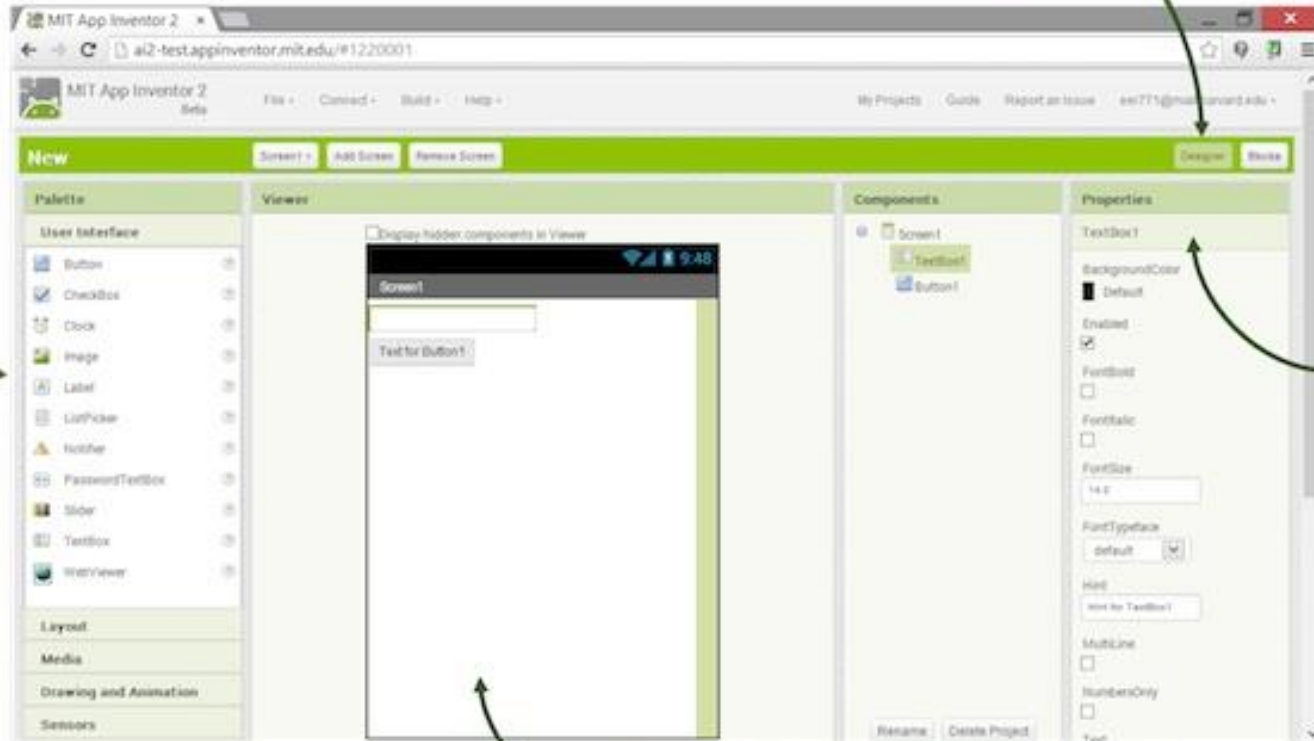
Project name: HelloWorldApp

Cancel OK

App Inventor 2 GUI (1)

Palette: Find your components and drag them to the Viewer to add them to your app.

Designer Button:
Click from any tab to go to the Designer tab.



Properties: Select a Component in the Components List to change its properties (color, size, behavior) here.

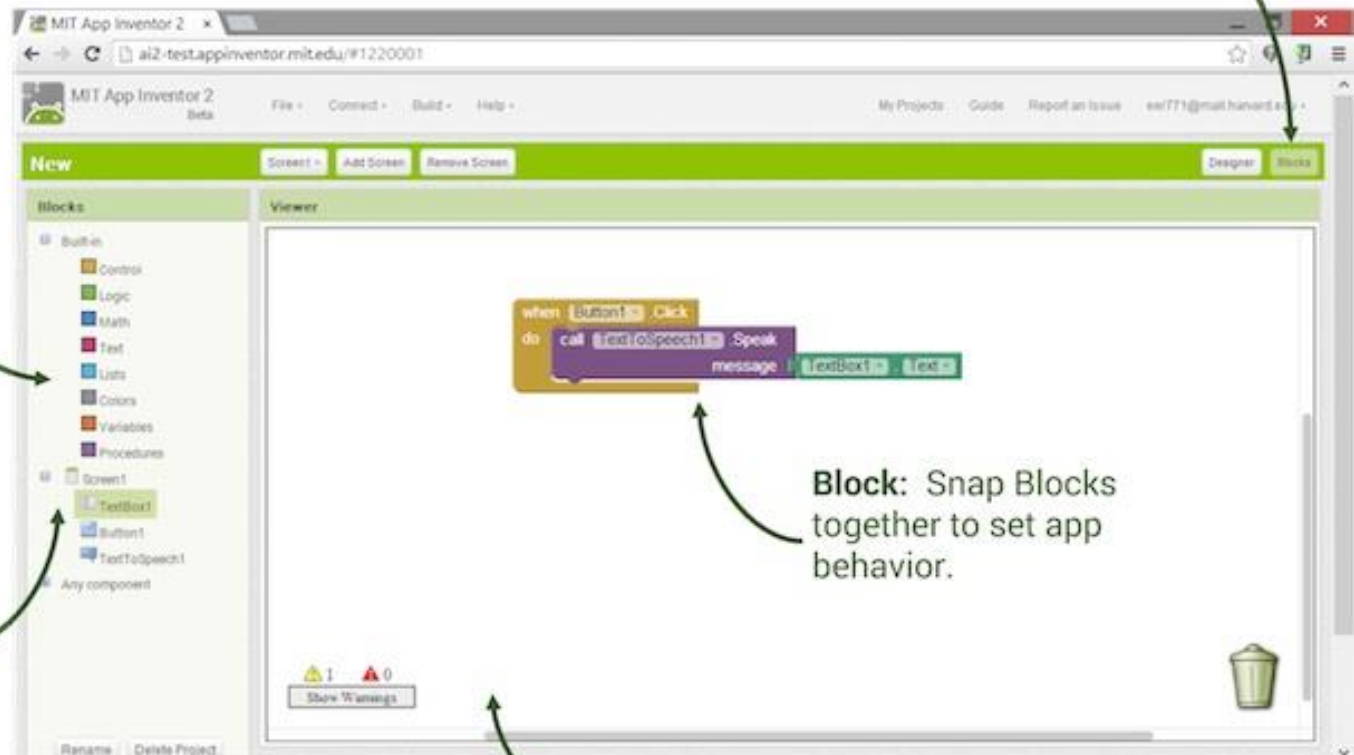
Viewer: Drag components from the Palette to the Viewer to see what your app will look like.

App Inventor 2 GUI (2)

Built-In Drawers: Find Blocks for general behaviors you may want to add to your app and drag them to the Blocks Viewer.

Blocks Button: Click from any tab to go to the Blocks tab.

Component-Specific Drawers: Find Blocks for behaviors for specific Components and drag them to the Blocks Viewer.

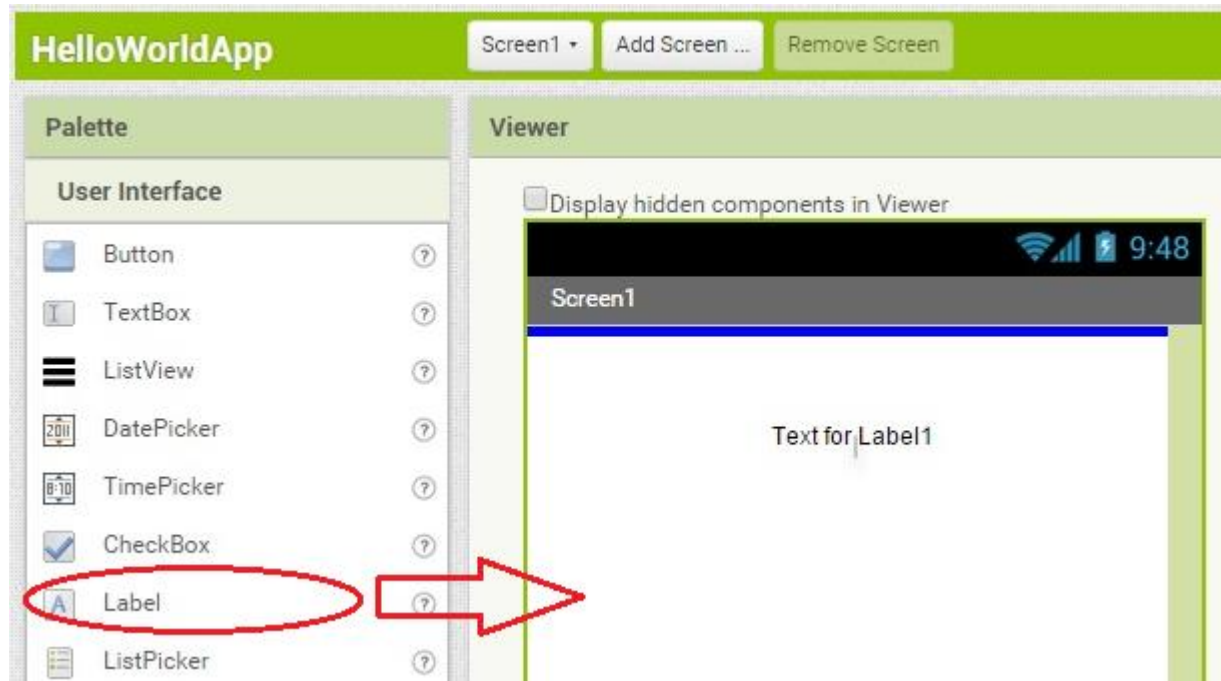


Block: Snap Blocks together to set app behavior.

Viewer: Drag Blocks from the Drawers to the Blocks Viewer to build relationships and behavior.

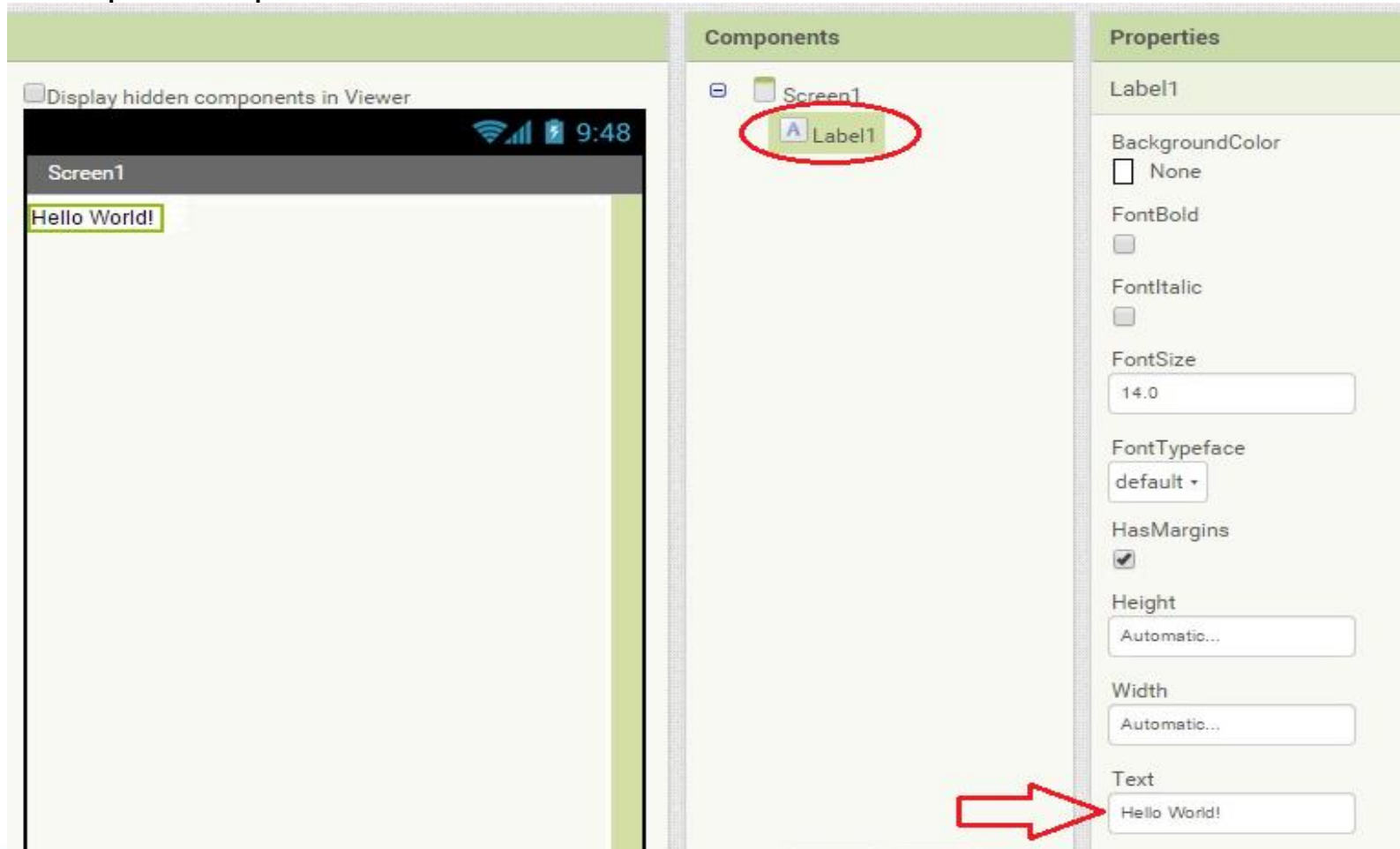
Hello World Application (3)

3. Drag and Drop a “Label” control from the “User Interface Panel” to the screen of your app.



Hello World Application (4)

4. Change the text of the label component. (Click “Label1” in the “Components” panel, then type “Hello World!” in the “Text” field in the “Properties” panel).



The screenshot displays the development environment for a Hello World application. It is divided into three main sections:

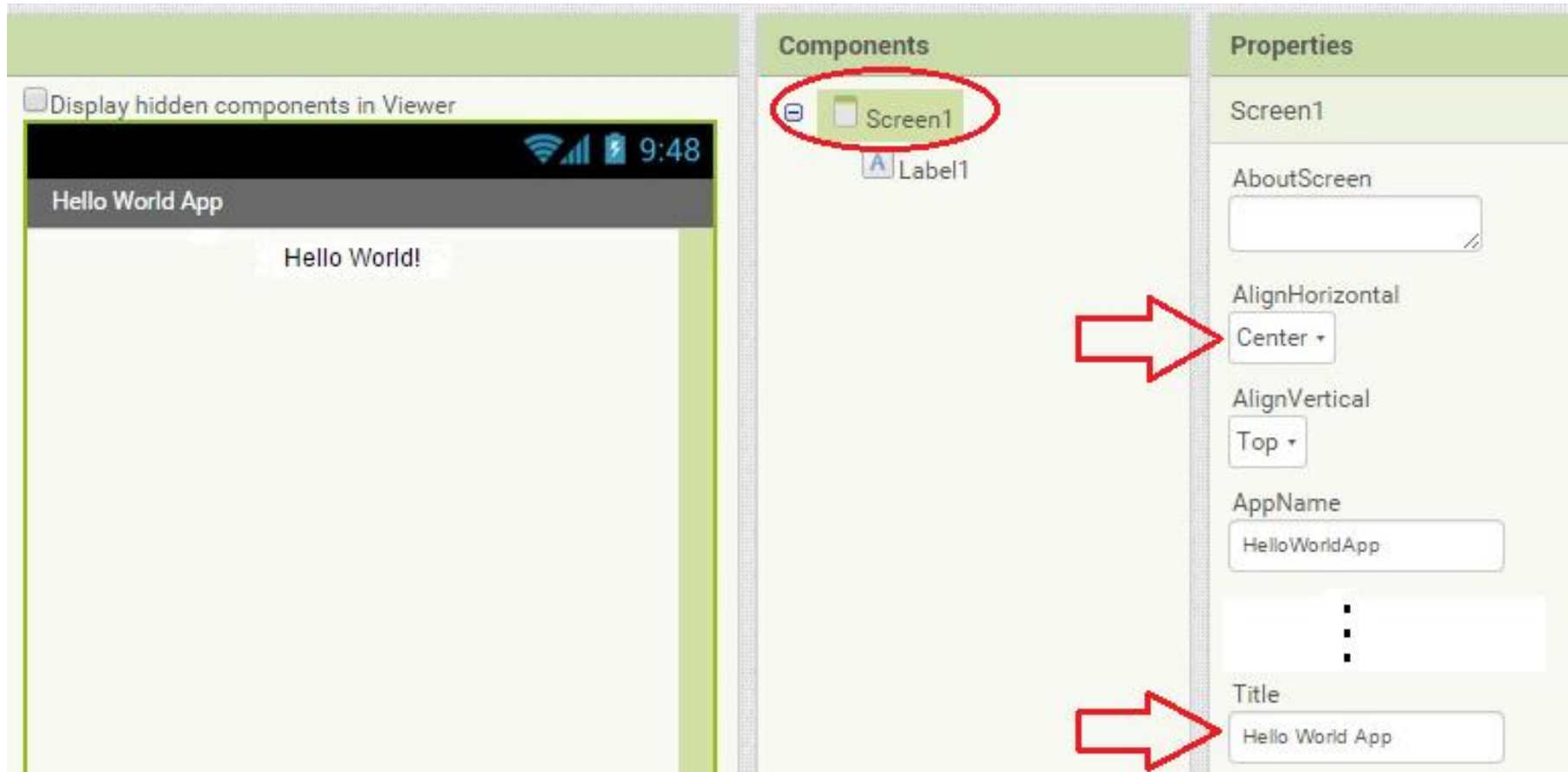
- Viewer:** On the left, it shows a mobile device screen with a status bar at the top displaying signal strength, Wi-Fi, and battery icons, along with the time 9:48. Below the status bar is a header labeled "Screen1". The main content area of the screen displays the text "Hello World!" in a green box.
- Components Panel:** In the center, it lists the components of the application. "Screen1" is the root component, and "Label1" is a child component. "Label1" is highlighted with a red oval.
- Properties Panel:** On the right, it shows the properties of the selected "Label1" component. The "Text" property is set to "Hello World!". A red arrow points to the "Text" field.

The "Properties" panel includes the following settings for "Label1":

- BackgroundColor: ☐ None
- FontBold: ☐
- FontItalic: ☐
- FontSize: 14.0
- FontTypeface: default
- HasMargins: ☒
- Height: Automatic...
- Width: Automatic...
- Text: Hello World!

Hello World Application (5)

5. Change settings of main screen. (Click “Screen1” in the “Components” panel, then in “Properties” panel change “AlignHorizontal” field to “Center” and type “Hello World App” in the “Title” field.)



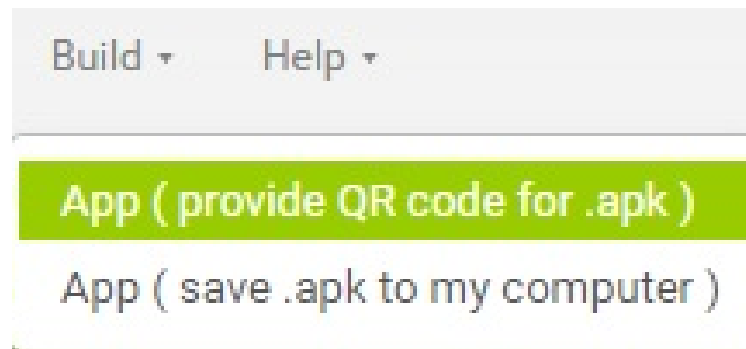
The screenshot displays the application development environment. On the left, the 'Viewer' pane shows a mobile app interface with a status bar at the top displaying signal strength, battery, and the time 9:48. Below the status bar is a dark header with the text 'Hello World App'. The main content area is white and contains a single text label that says 'Hello World!'. Above the viewer, there is a checkbox labeled 'Display hidden components in Viewer'.

In the center, the 'Components' pane lists the app's components. 'Screen1' is highlighted with a red oval, and 'Label1' is listed below it.

On the right, the 'Properties' pane shows the settings for 'Screen1'. A red arrow points to the 'AlignHorizontal' property, which is set to 'Center'. Another red arrow points to the 'Title' property, which is set to 'Hello World App'. Other visible properties include 'AboutScreen' (with an empty text field), 'AlignVertical' (set to 'Top'), and 'AppName' (set to 'HelloWorldApp').

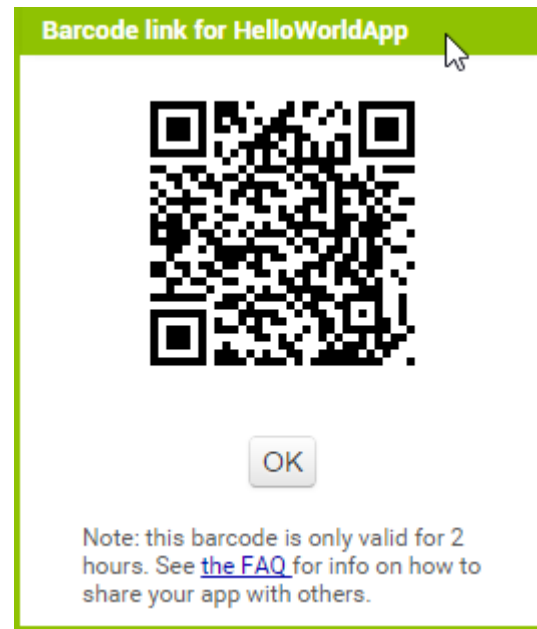
Hello World Application (6)

6. Create an .apk and install the application in your phone. (Select “App (provide QR code for .apk)” from Build menu.



Hello World Application (7)

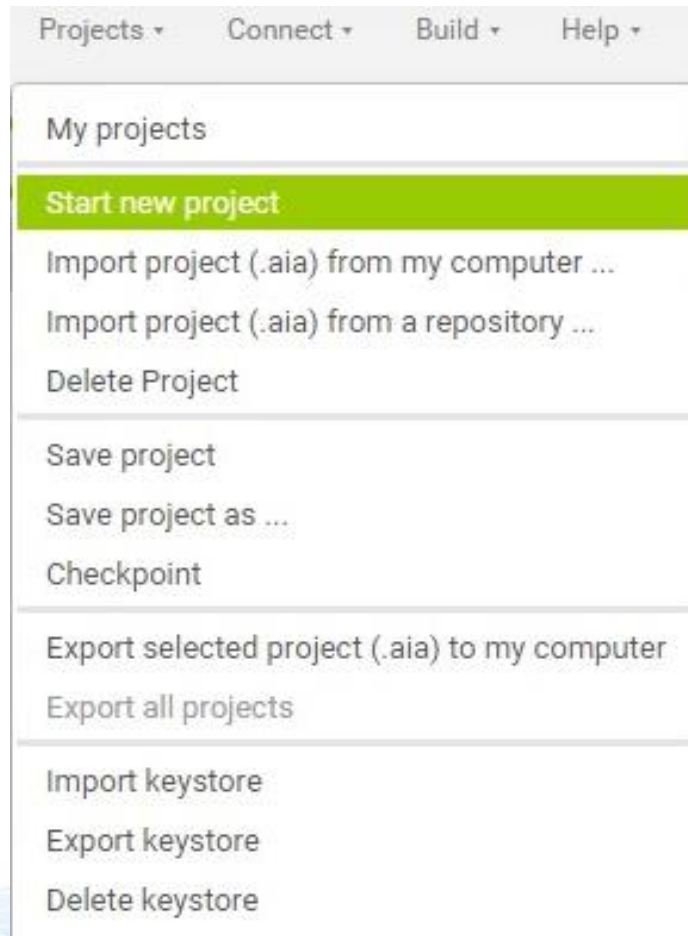
7. Read the provided QR code using any QR code reader app in your phone. This action will download an .apk file into your phone. After download just install it and run it.



CHECKPOINT 5!

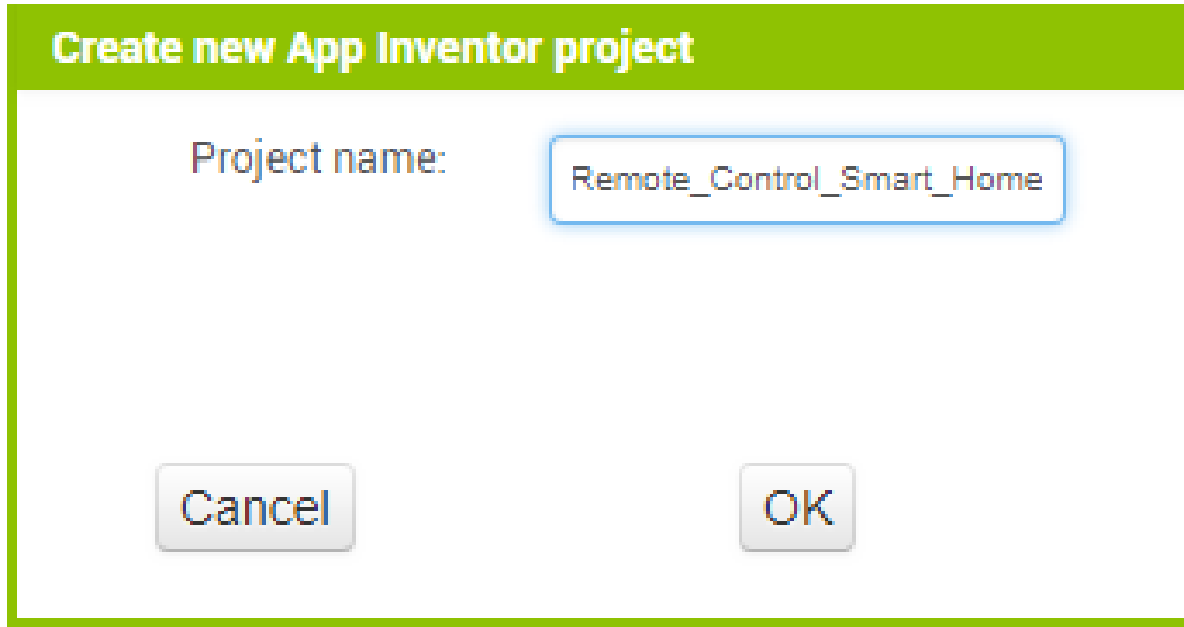
Remote Control Smart Home App

Create a new project



Remote Control Smart Home App

Create a new project called “Remote_Control_Smart_Home”.

A screenshot of the "Create new App Inventor project" dialog box. The dialog has a green title bar with the text "Create new App Inventor project". Below the title bar, there is a label "Project name:" followed by a text input field containing the text "Remote_Control_Smart_Home". At the bottom of the dialog, there are two buttons: "Cancel" on the left and "OK" on the right.

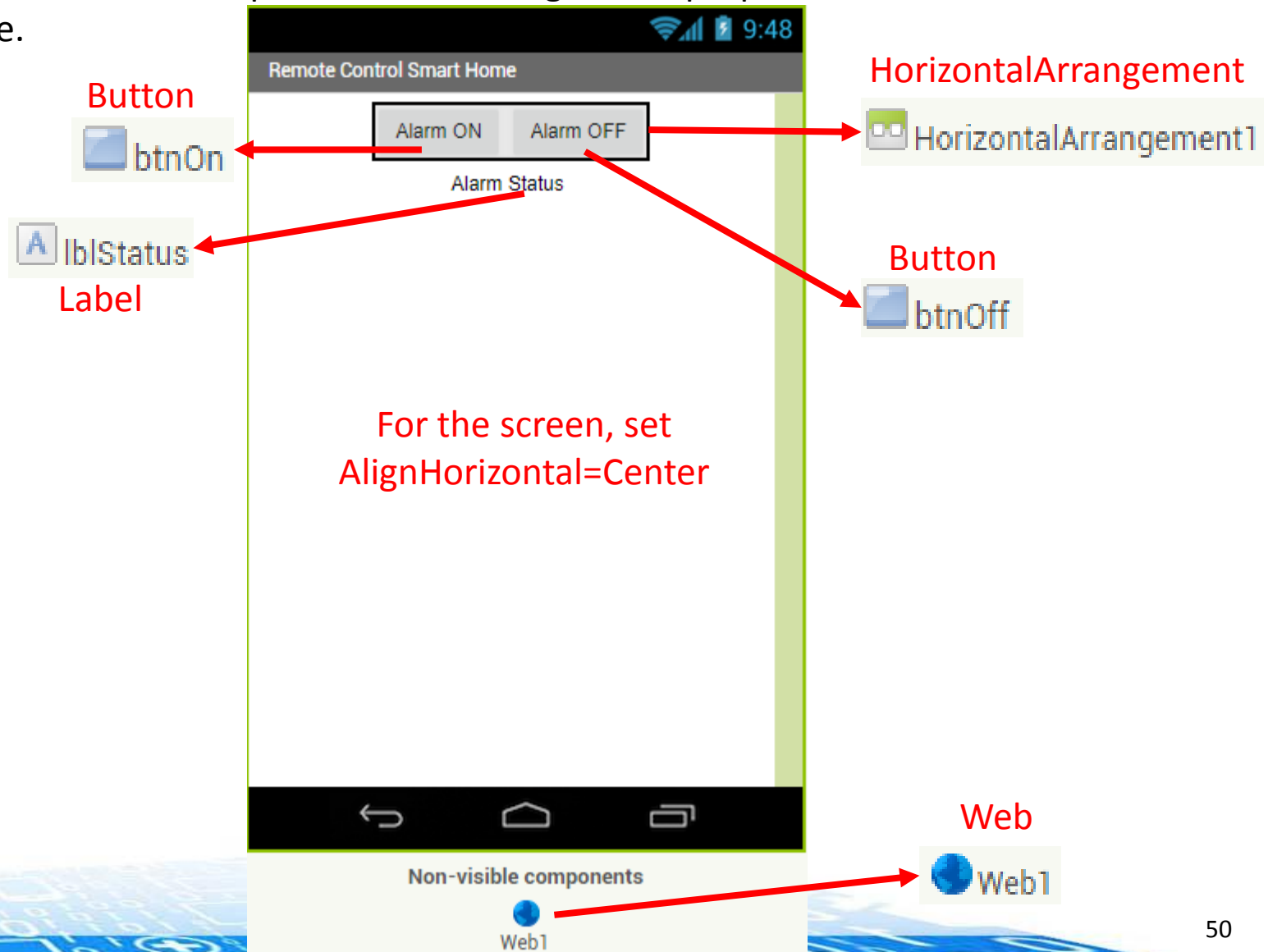
Create new App Inventor project

Project name: Remote_Control_Smart_Home

Cancel OK

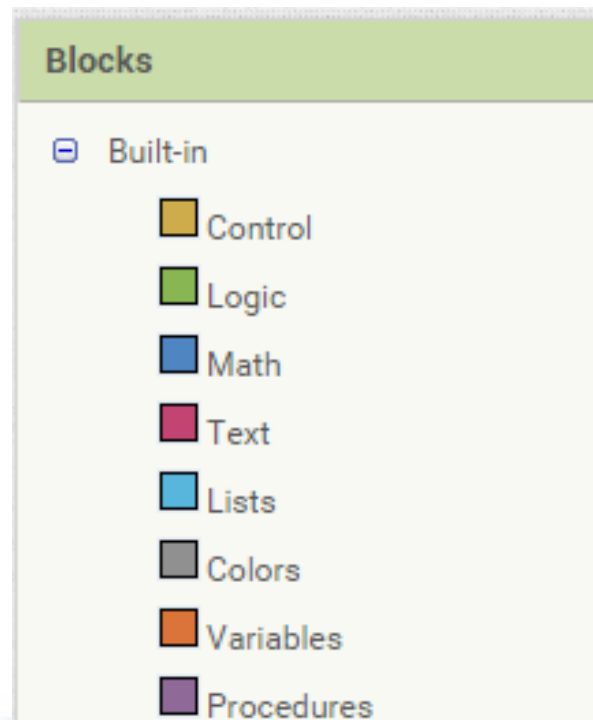
Remote Control Smart Home App

3. Draw the GUI components and change their properties as showed in the picture.



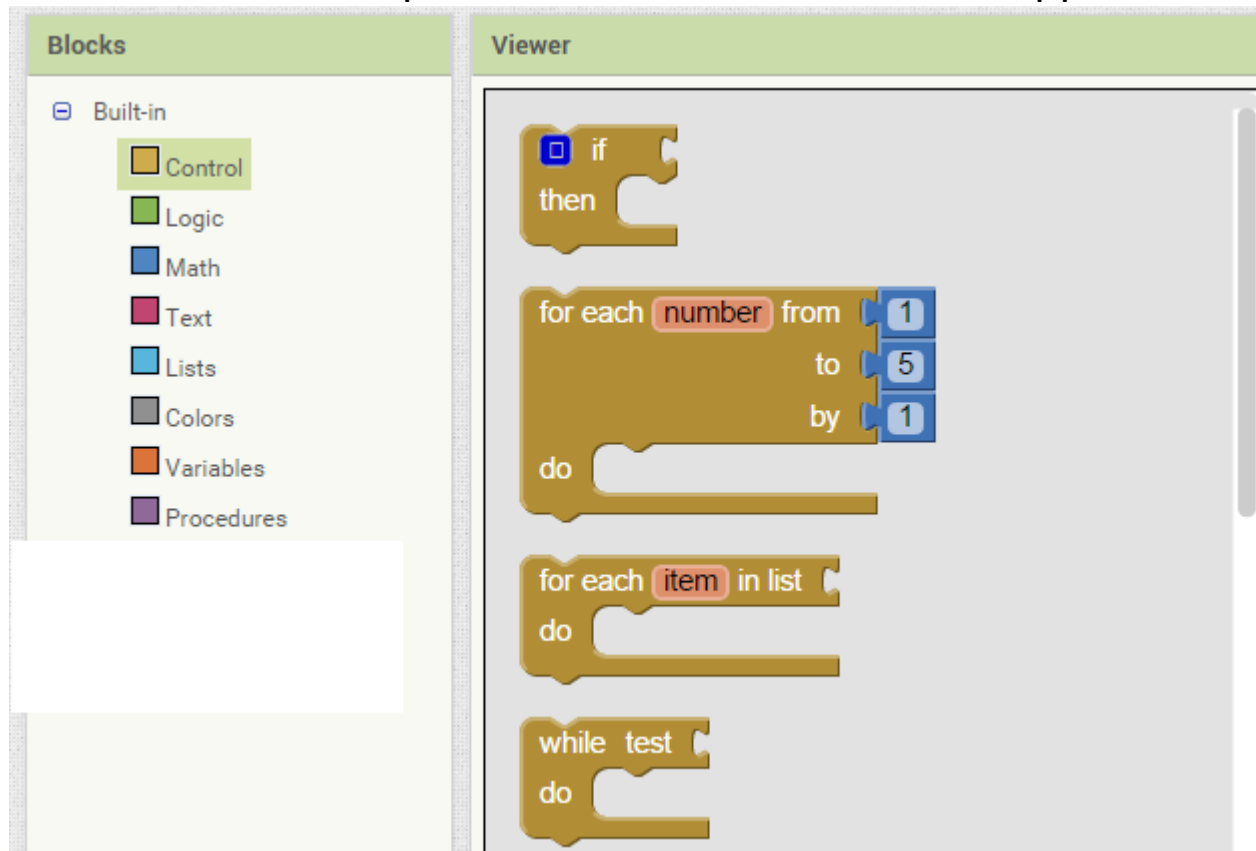
Coding with Blocks (1)

- The Blocks editor has different components that can be used to build our application logic.
- The components follow the drag and drop paradigm.



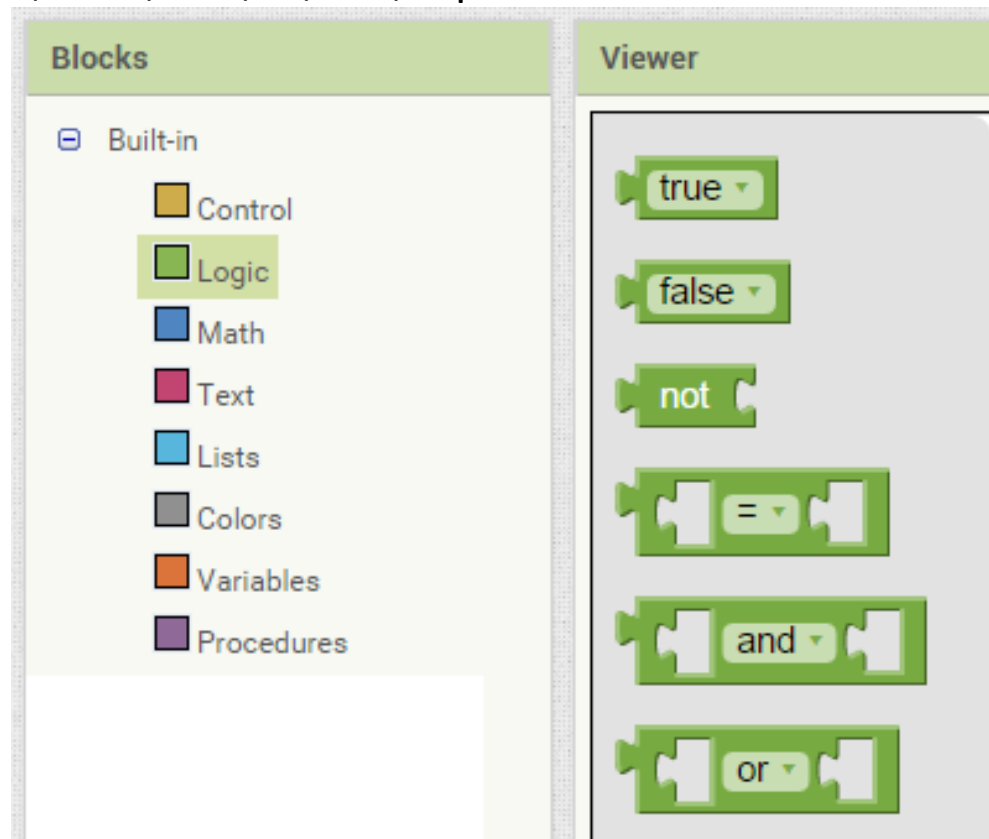
Coding with Blocks (2)

- Control: If, for, while, open screen, close screen, close application, etc.



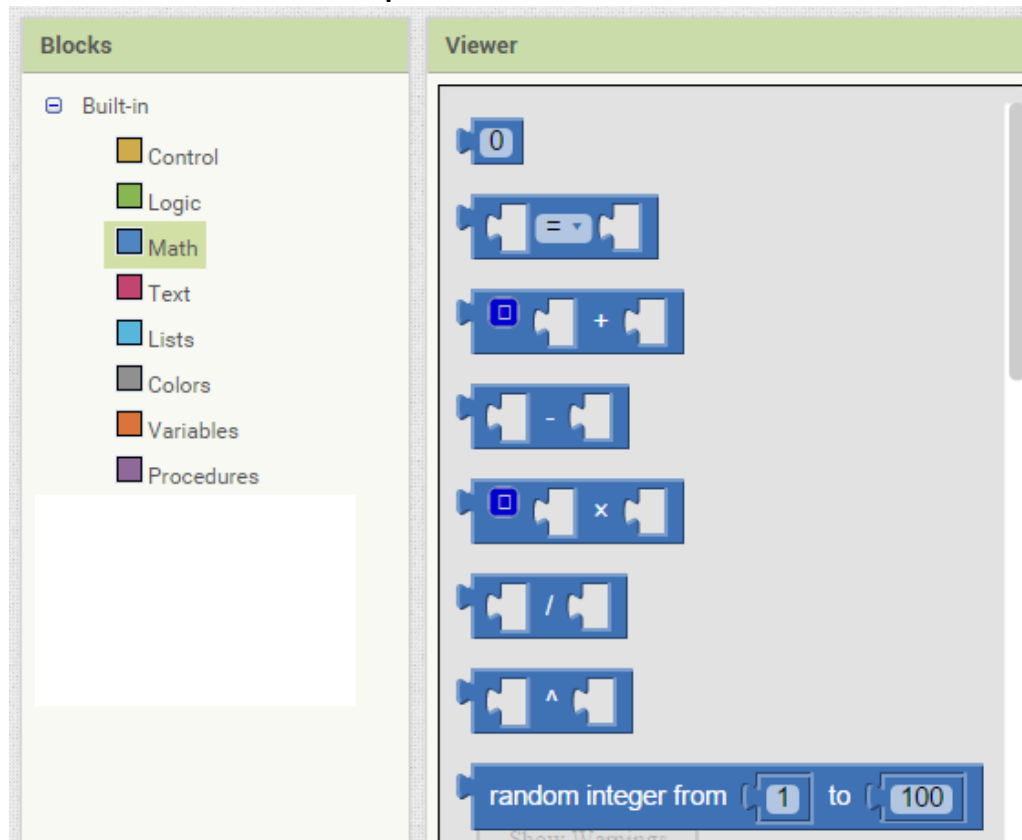
Coding with Blocks (3)

- Logic: true, false, and, or, not, equals.



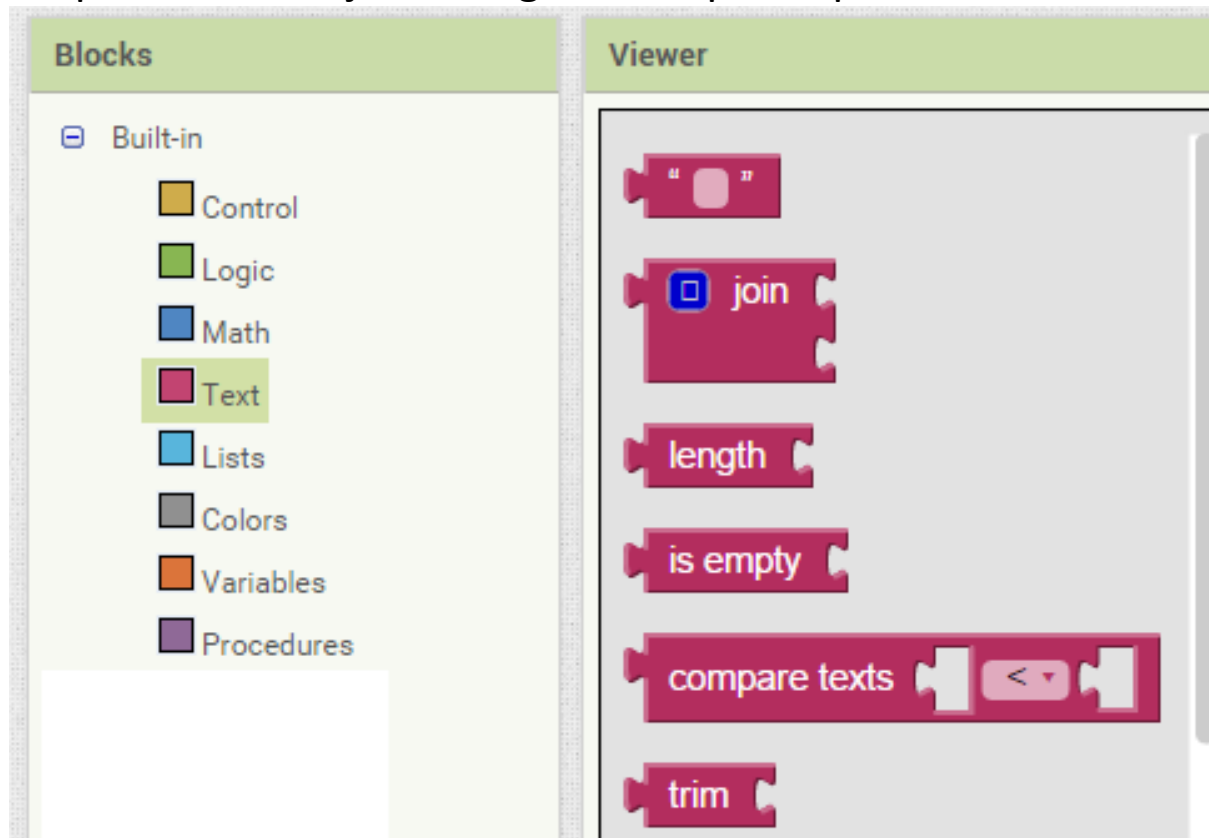
Coding with Blocks (4)

- Math: numbers, numerical operators, random number, abs, sin, cos, etc.



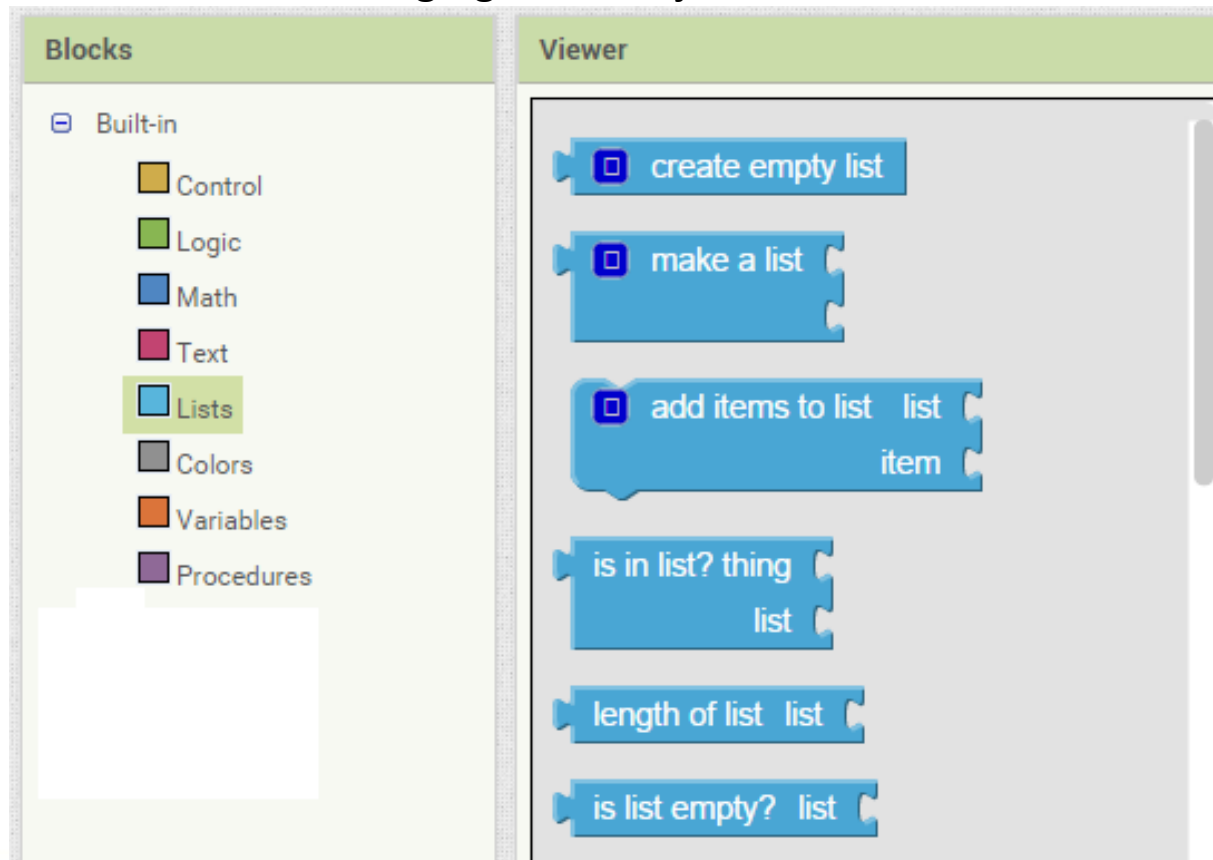
Coding with Blocks (5)

- Text: input user text, join strings, trim, split, replace, etc.



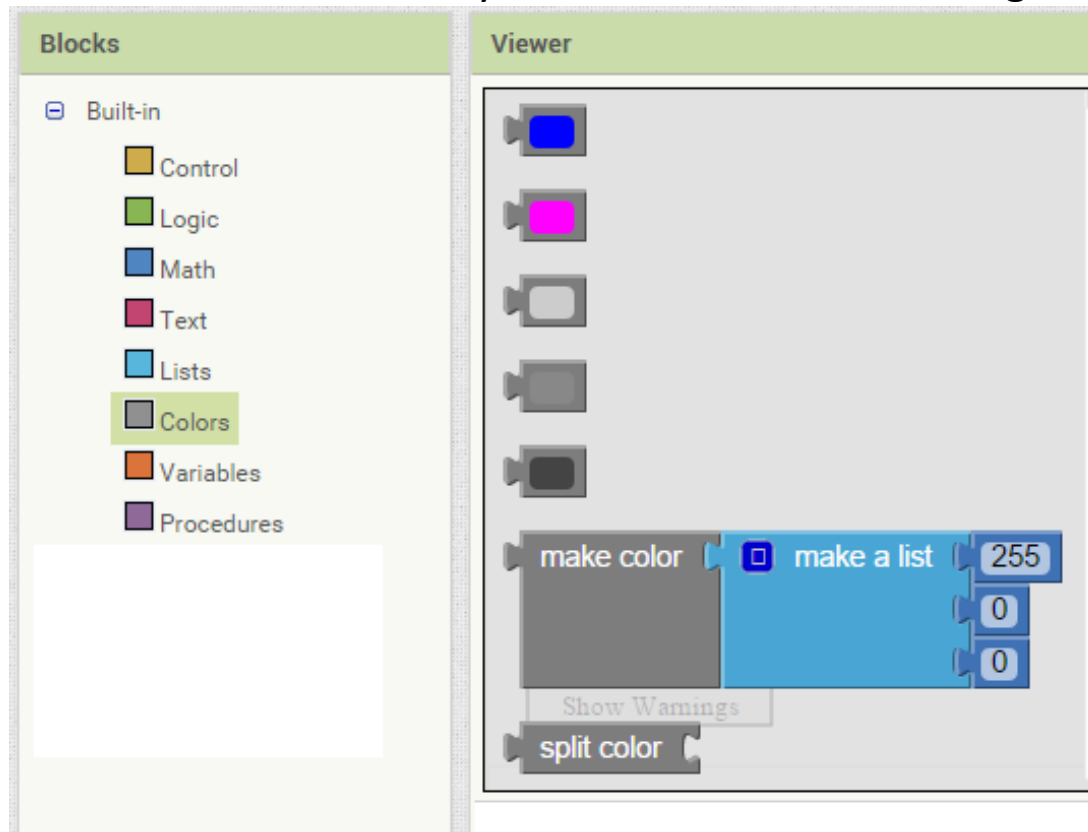
Coding with Blocks (6)

- Lists: functions for managing list of objects.



Coding with Blocks (7)

- Colors: common colors, create your own color based on r, g, b.



Coding with Blocks (8)

- Variables: local variables, global variables, get value, set value.

A screenshot of a coding interface with two main panels: "Blocks" and "Viewer". The "Blocks" panel on the left has a green header and contains a "Built-in" category with a list of block types: Control (yellow), Logic (green), Math (blue), Text (pink), Lists (light blue), Colors (grey), Variables (orange), and Procedures (purple). The "Variables" category is highlighted with a green background. The "Viewer" panel on the right has a green header and displays a sequence of code blocks. The first block is "initialize global name to". The second block is "get" with a dropdown arrow. The third block is "set" with a dropdown arrow and "to". The fourth block is "initialize local name to" with a blue square icon and "in". The fifth block is "initialize local name to" with a blue square icon and "in".

Blocks

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Viewer

initialize global name to

get

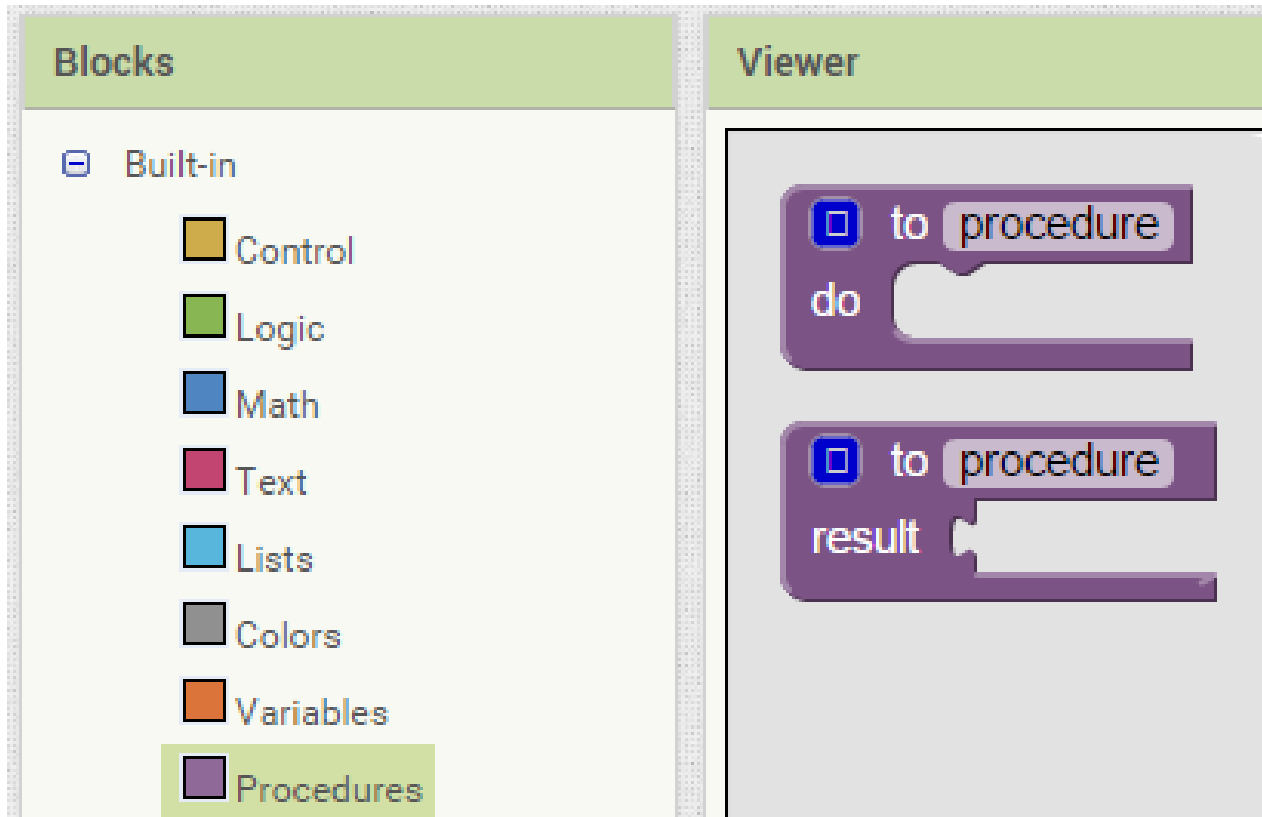
set to

initialize local name to in

initialize local name to in

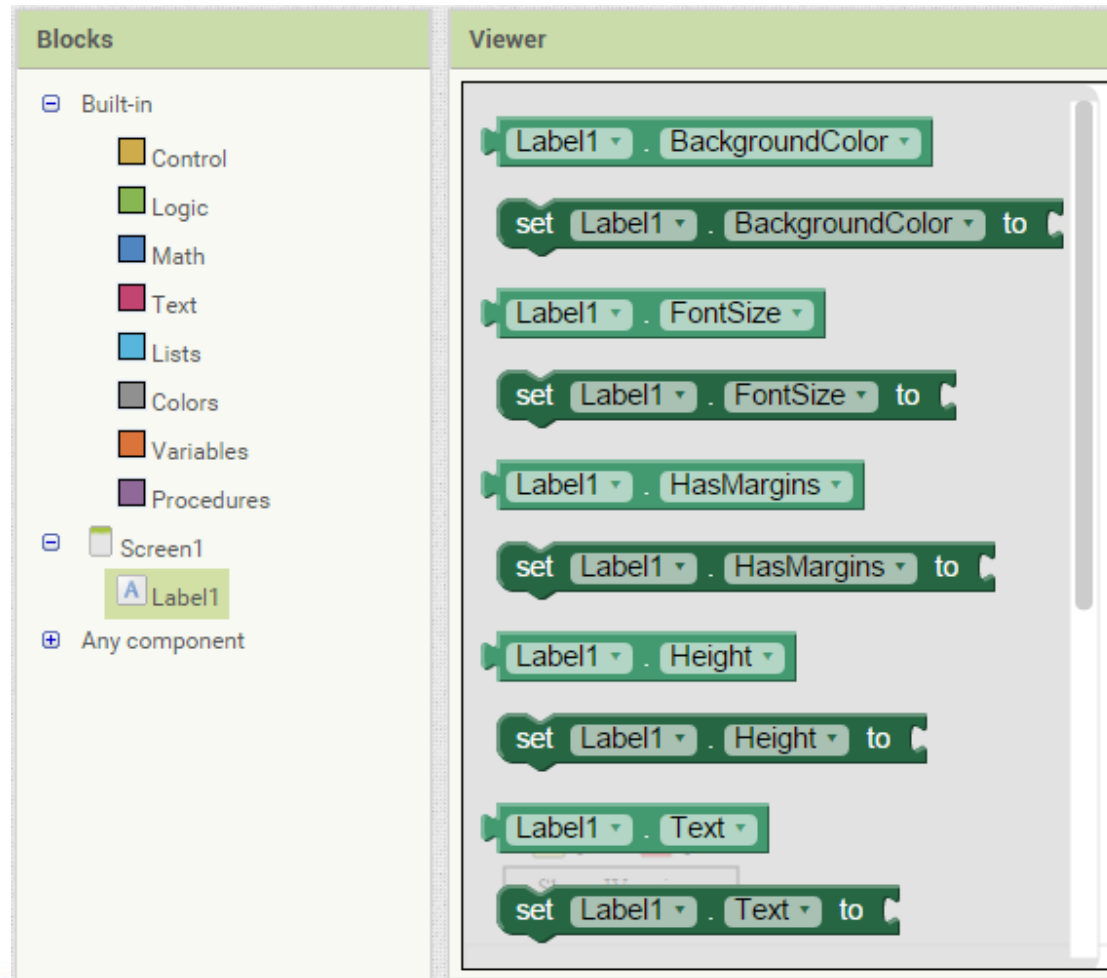
Coding with Blocks (9)

- Procedures: allows reusability of code creating functions.

The image shows a screenshot of the Scratch programming environment. On the left is the "Blocks" palette, which is organized into categories: Built-in, Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. The "Procedures" category is currently selected, highlighted in green. On the right is the "Viewer" area, which displays two "to procedure" blocks. The first block is a "do" block, and the second block is a "result" block. Both blocks have a small blue square icon with a white "x" in the top-left corner.

Coding with Blocks (10)

- Additionally, each GUI component has its own blocks. For example, a label has blocks for getting and setting its text, getting and setting its font height and size, etc.

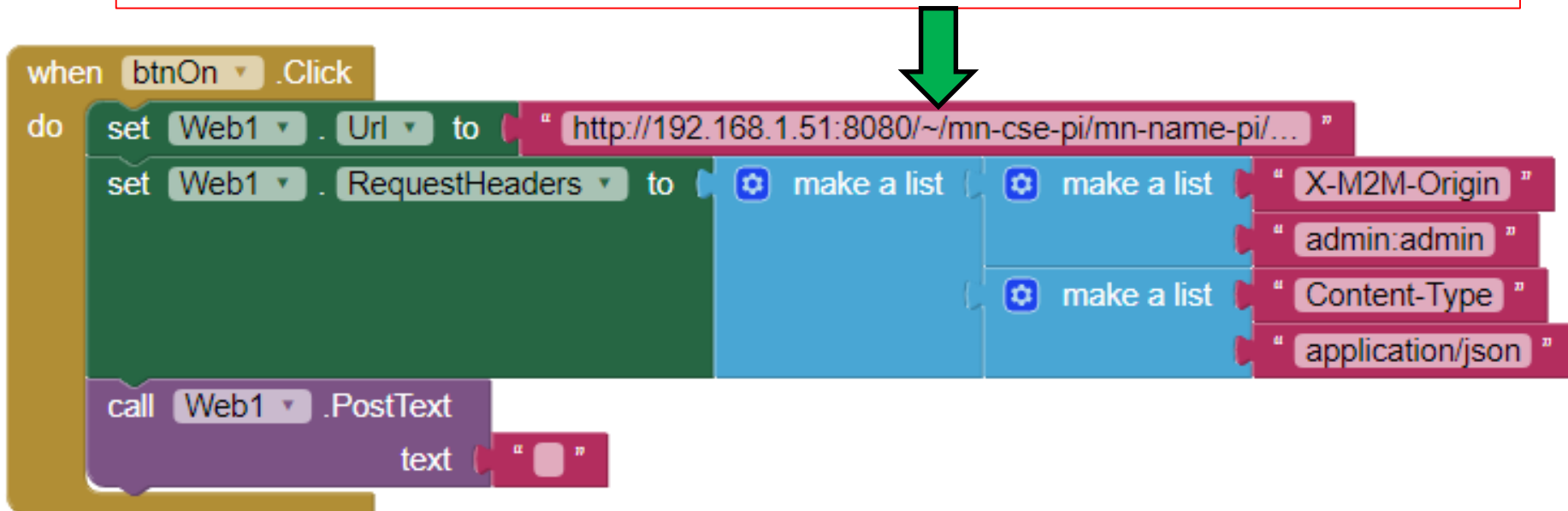


Remote Control Smart Home App

In the blocks area, draw the corresponding objects for:

a. Turn ON the Alarm

`http://YOUR.VM.IP.ADDRESS:8080/~mn-cse-pi/mn-name-pi/ALARM_ON`

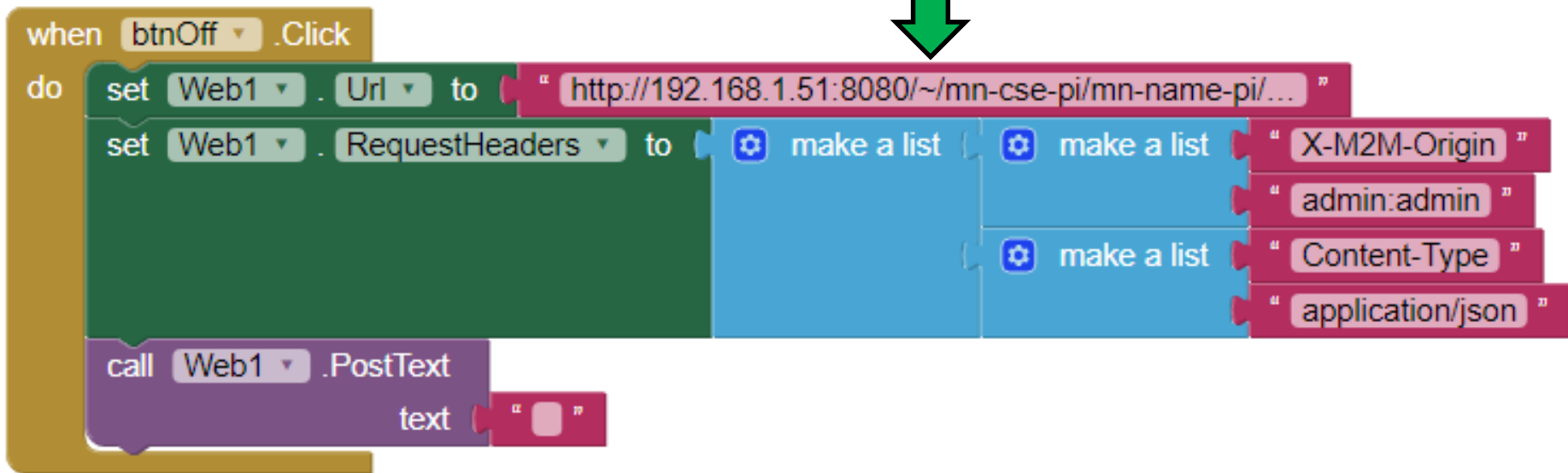


Remote Control Smart Home App

In the blocks area, draw the corresponding objects for:

b. Turn OFF the Alarm

`http://YOUR.VM.IP.ADDRESS:8080/~mn-cse-pi/mn-name-pi/ALARM_OFF`

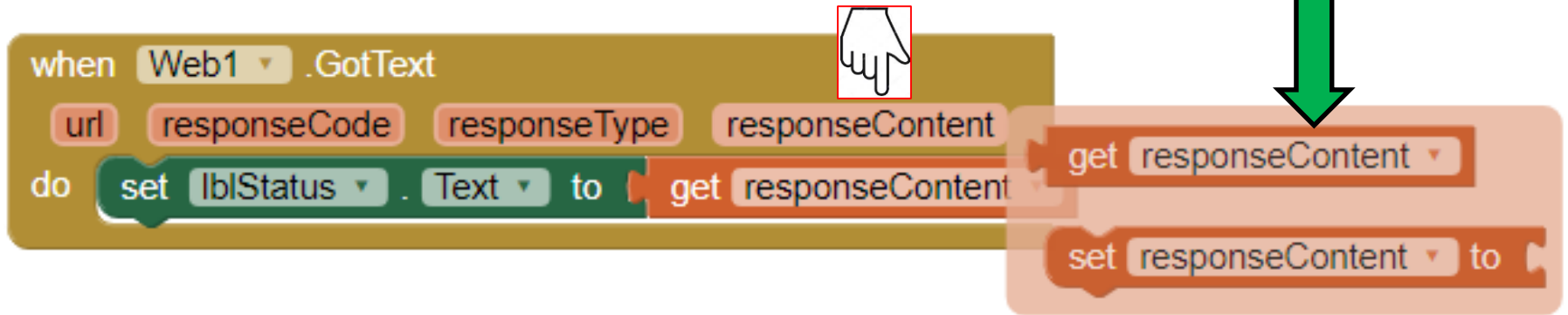


Remote Control Smart Home App

In the blocks area, draw the corresponding objects for:

c. Get Server Response.

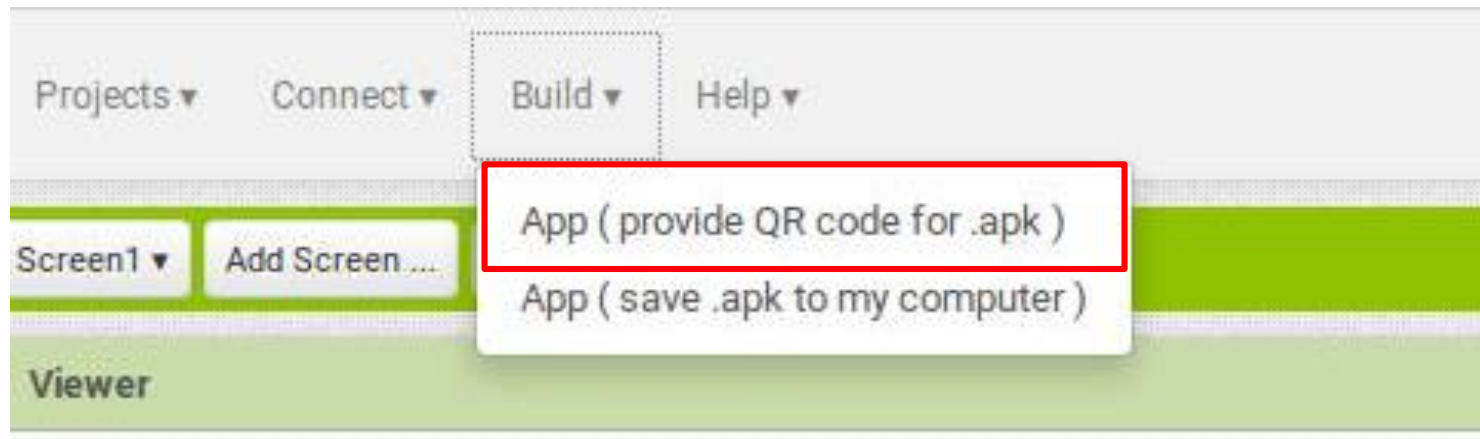
Put your mouse pointer over responseContent and then drag and drop “get ResponseContent”



The image shows a Scratch code editor with a 'when Web1 GotText' event block. The 'url' block has a dropdown menu showing 'responseCode', 'responseType', and 'responseContent'. A mouse cursor is hovering over 'responseContent'. Below the event block is a 'do' block containing a 'set lblStatus Text to' block followed by a 'get responseContent' block. To the right, a separate block contains a 'get responseContent' block and a 'set responseContent to' block. A green arrow points from the text box to the 'get responseContent' block in the separate block.

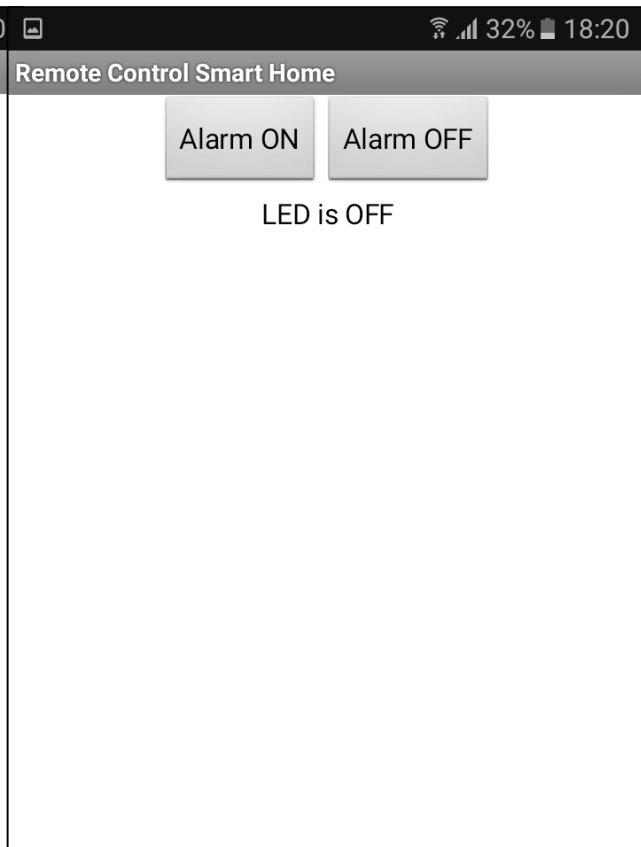
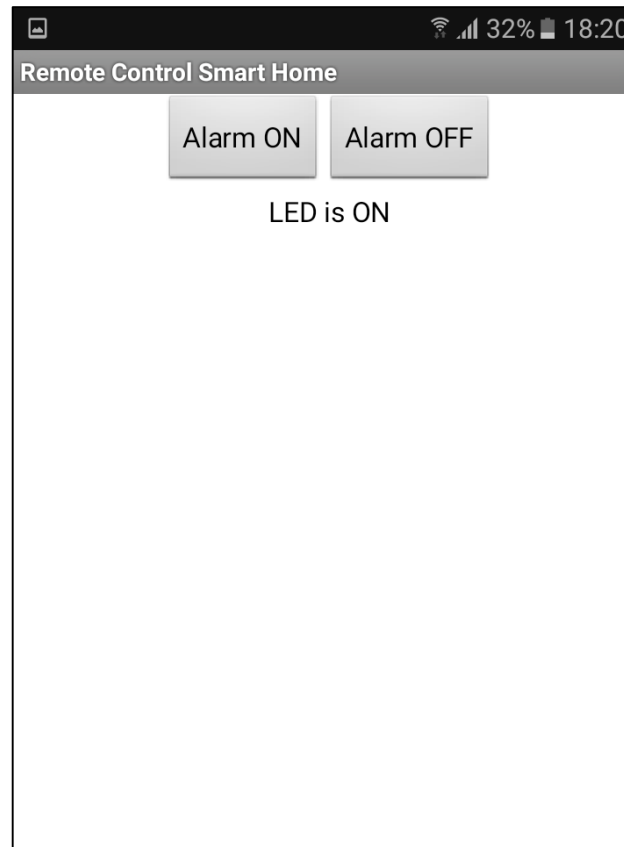
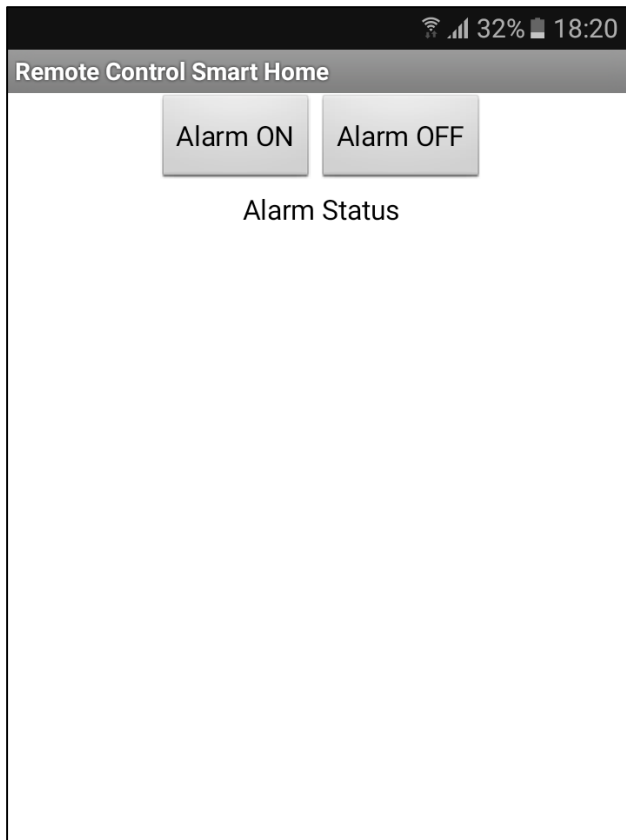
Remote Control Smart Home App

Build and Install the App in your smartphone.



Remote Control Smart Home App

Run the app.



CHECKPOINT 6!