



Internet of Things with NodeMCU and MIT App Inventor

Dr Fauzan Khairi Che Harun
www.fauzankhairi.com

Universiti Teknologi Malaysia

Tinkercode

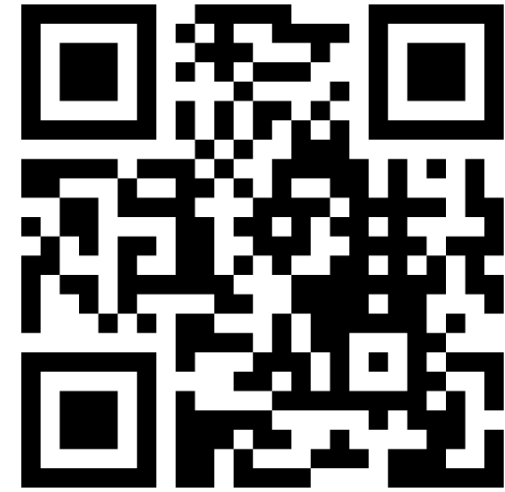
<https://tinkercode.my/>

The screenshot displays the Tinkercode web application interface. The browser address bar shows the URL `app.tinkercode.my/?url=../examples/./ESP32/ESP32IOT4u.xml`. The interface includes a sidebar with categories like Logic, Loops, Math, Text, Variables, Functions, Base Controller, and Robot Kit. The main workspace shows an Arduino IDE with the following code blocks:

- Setup** block.
- Arduino loop forever:**
 - set** block: `reading` to `MHET ESP32 AnalogRead PIN# 2` as `Number`.
 - print new line data on serial port:** `reading`.
 - Get Request** block: `create text with` containing:
 - `" http://iot4u.my/app/json/index.php? "`
 - `" project_key= "`
 - `" e449ceefc0463b1553197b7e1e0f68b9 "`
 - `" &infrared= "`
 - `reading`
 - if** block: `valid Response`
 - do** block: `set payload` to `Get Response` as `Text`.
 - print new line data on serial port:** `payload`.
 - Response String** block: `payload`.
 - if in the response string we find** block: `"onButton"`
 - do** block: `MHET ESP32 DigitalWrite PIN# 2` Stat `HIGH`.
 - on next find** block: `"offButton"`
 - do** block: `MHET ESP32 DigitalWrite PIN# 2` Stat `LOW`.
 - delay (in ms)** block: `1000`.

On the right, an **ESP32** component is shown with the following configuration:

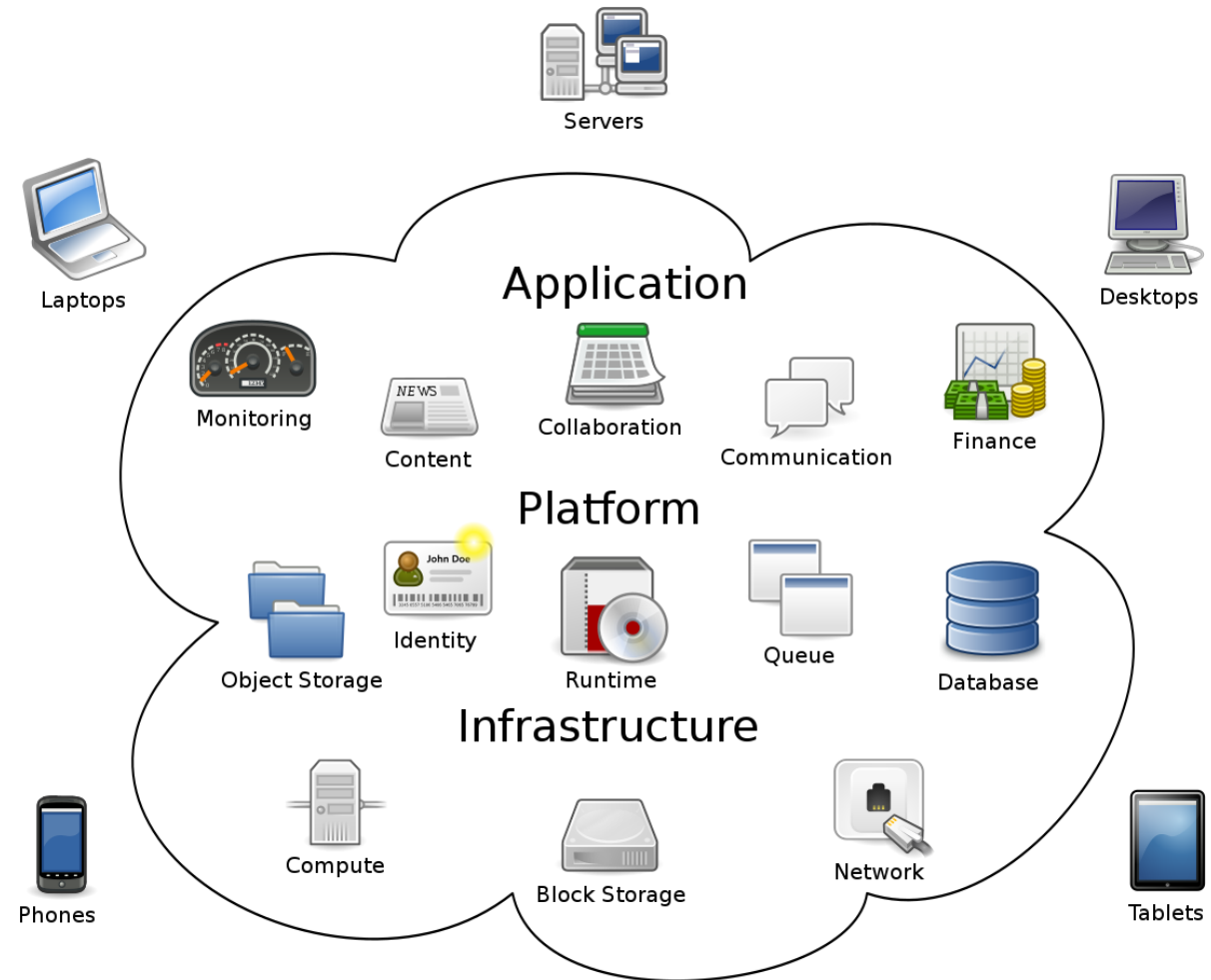
- `addressing`: `dynamic`
- `ssid`: `your_WIFI_SSID`
- `key`: `your_WIFI_KEY`
- `client`: `client`



Flow of Class

<https://www.menti.com/bn2wbvg7nb>

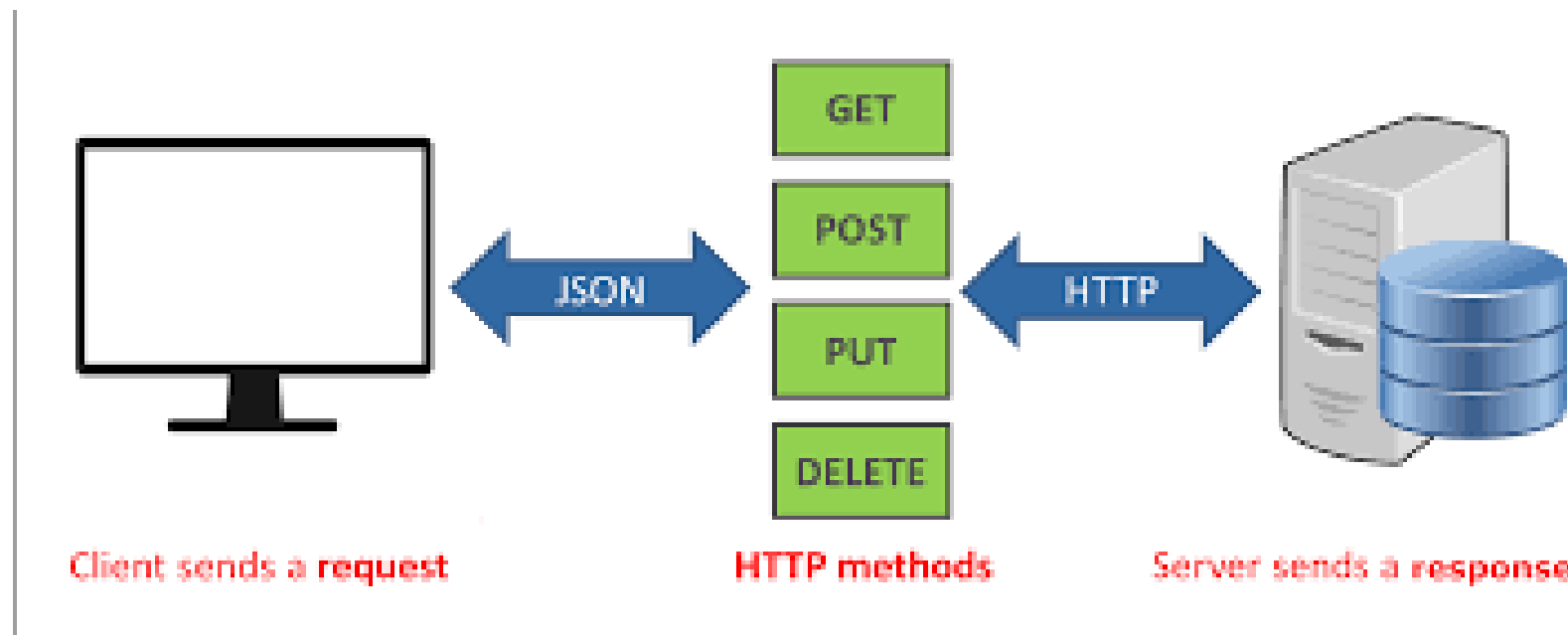
Introduction to Internet of Things



Computer and Server



Introduction to REST API



Today's Task

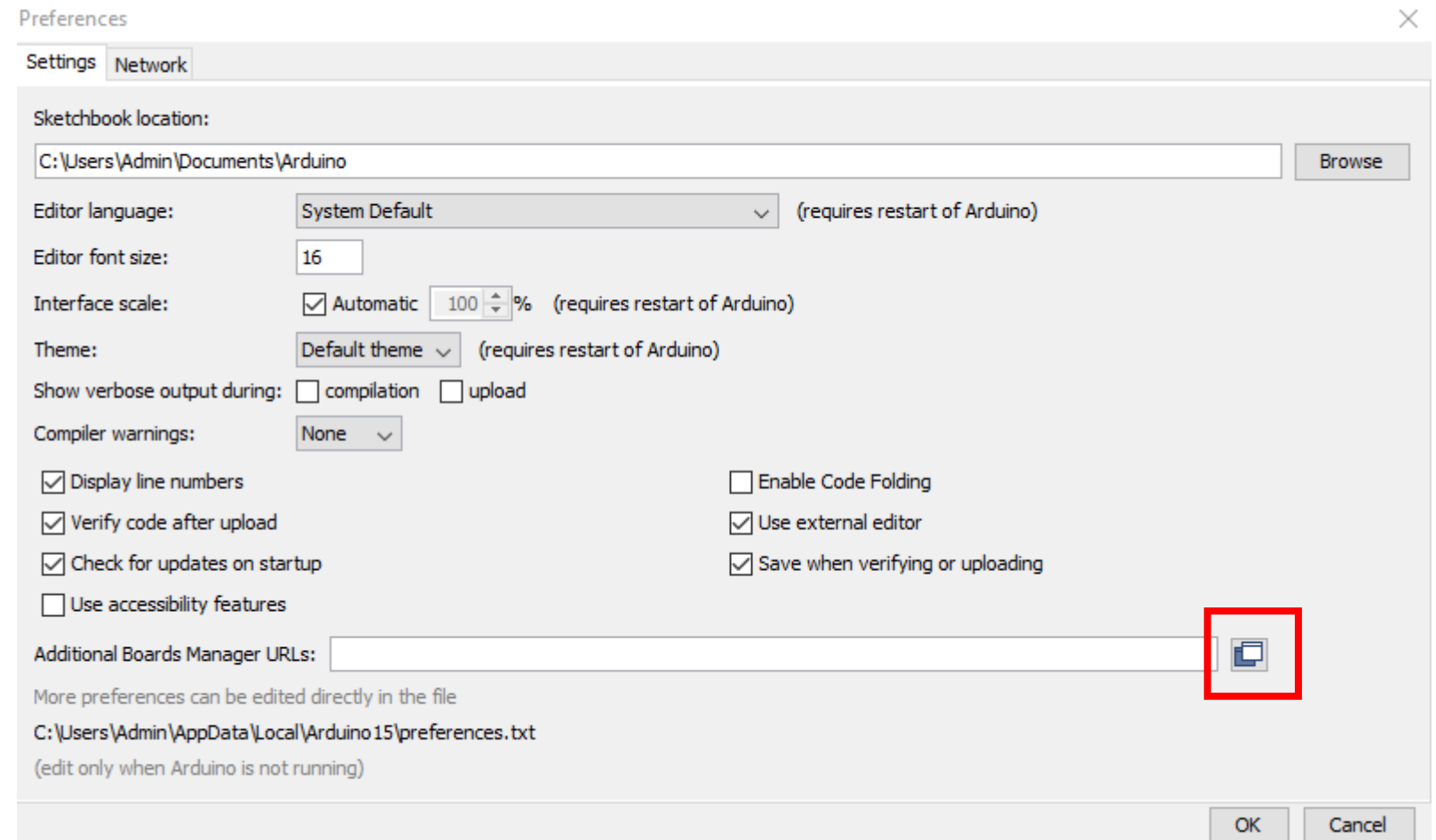
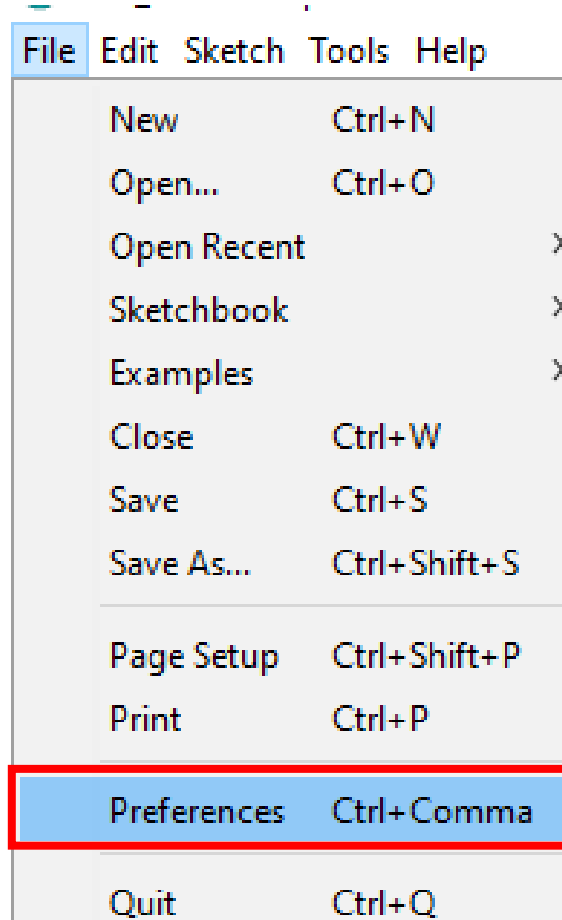


Arduino IDE Setup for ESP8266

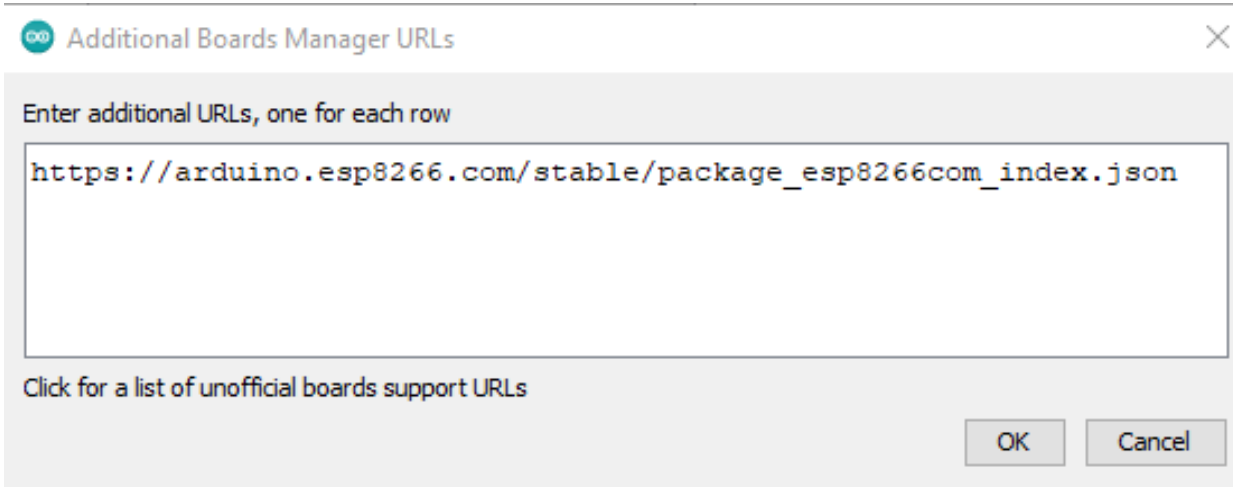
Fauzan Khairi Che Harun

Add ESP8266 to Arduino IDE

Open File-> Preferences, and click the button to add Additional Board URLs.



Copy this link https://arduino.esp8266.com/stable/package_esp8266com_index.json to
Additional Board Manager URLs



Additional Boards Manager URLs

Enter additional URLs, one for each row

```
https://arduino.esp8266.com/stable/package_esp8266com_index.json
```

[Click for a list of unofficial boards support URLs](#)

OK Cancel

If you previously use ESP32 boards or any other boards, just press Enter and paste the link on new line



Additional Boards Manager URLs

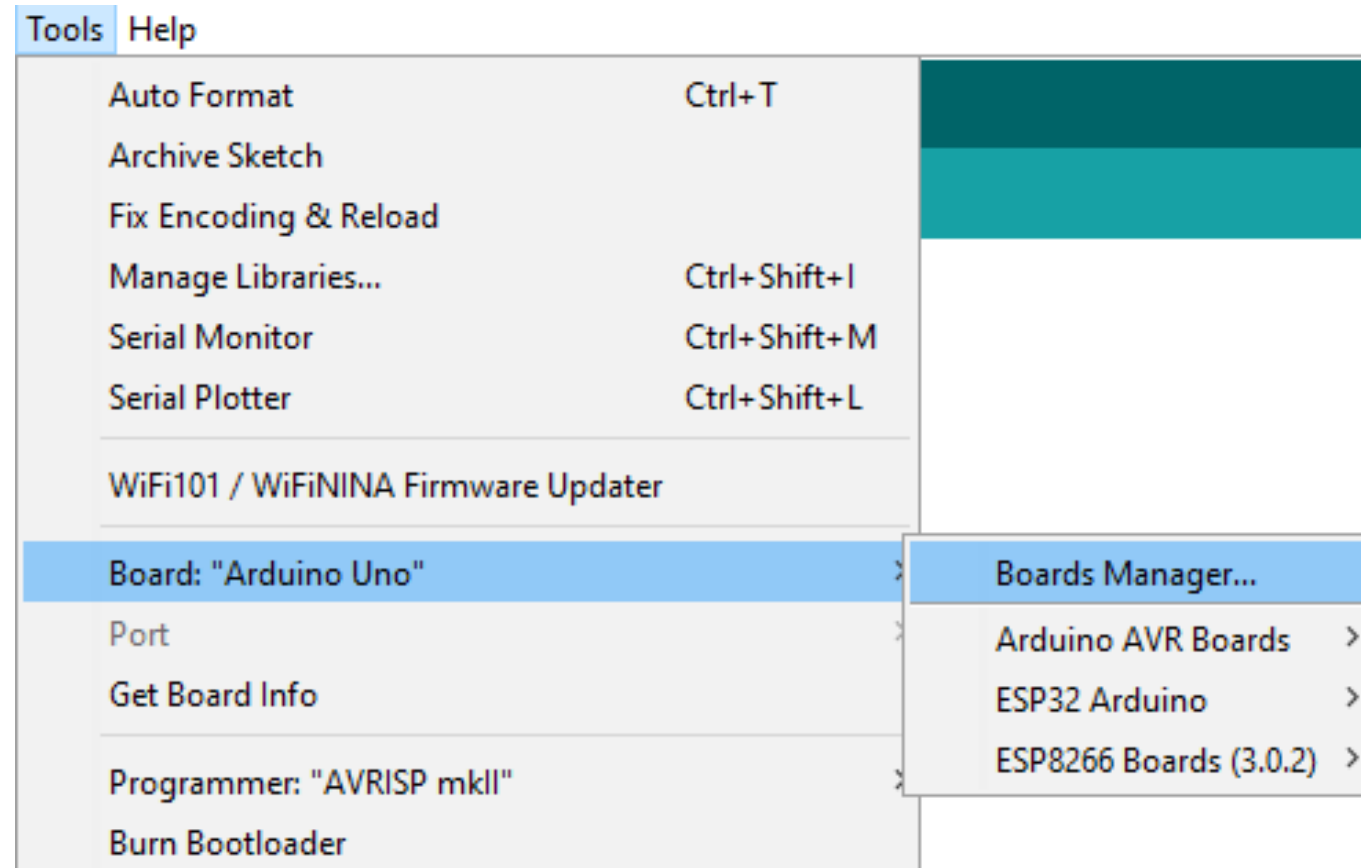
Enter additional URLs, one for each row

```
https://dl.espressif.com/dl/package_esp32_index.json  
https://arduino.esp8266.com/stable/package_esp8266com_index.json
```

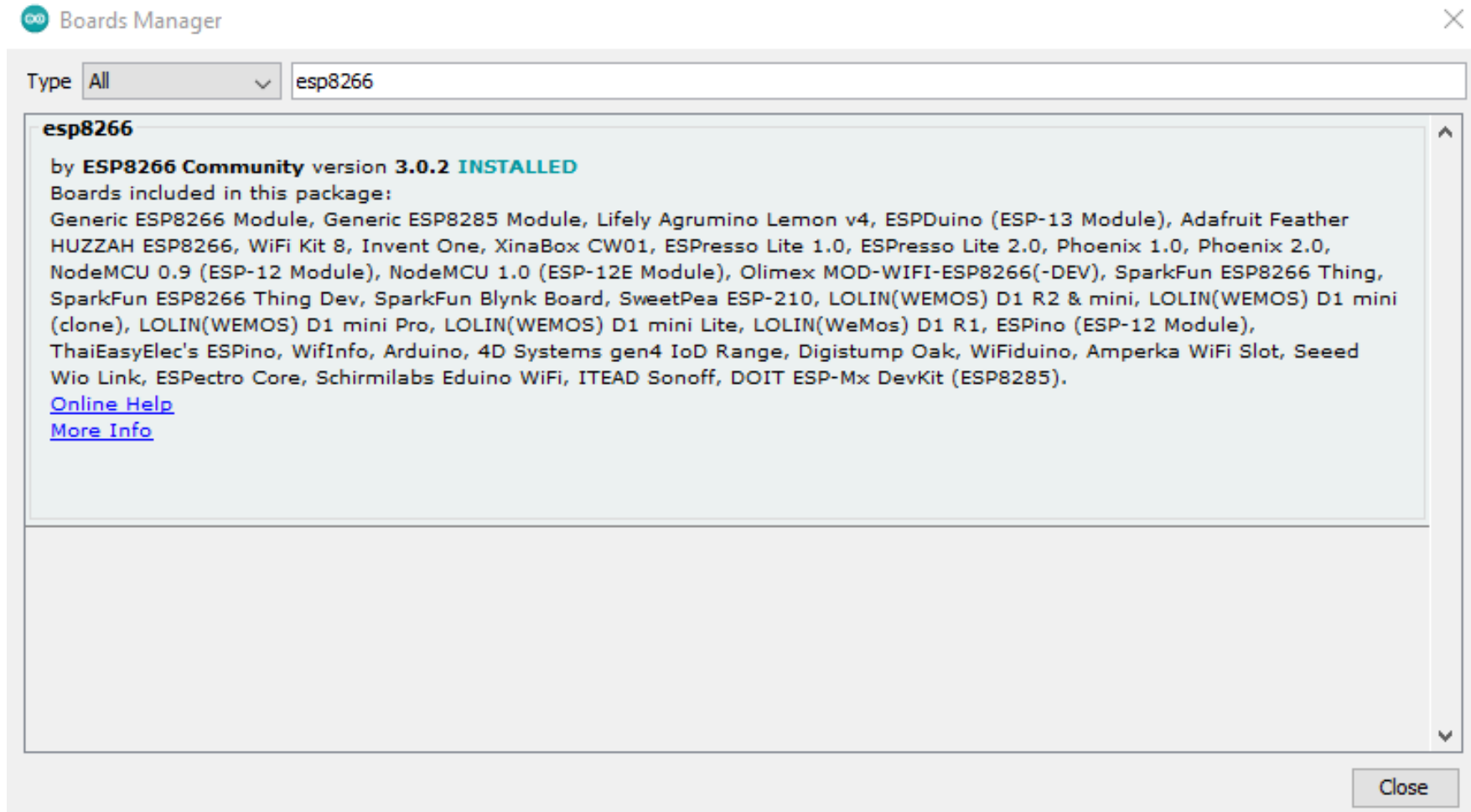
[Click for a list of unofficial boards support URLs](#)

OK Cancel

Select Tools-> Board -> Board Manager



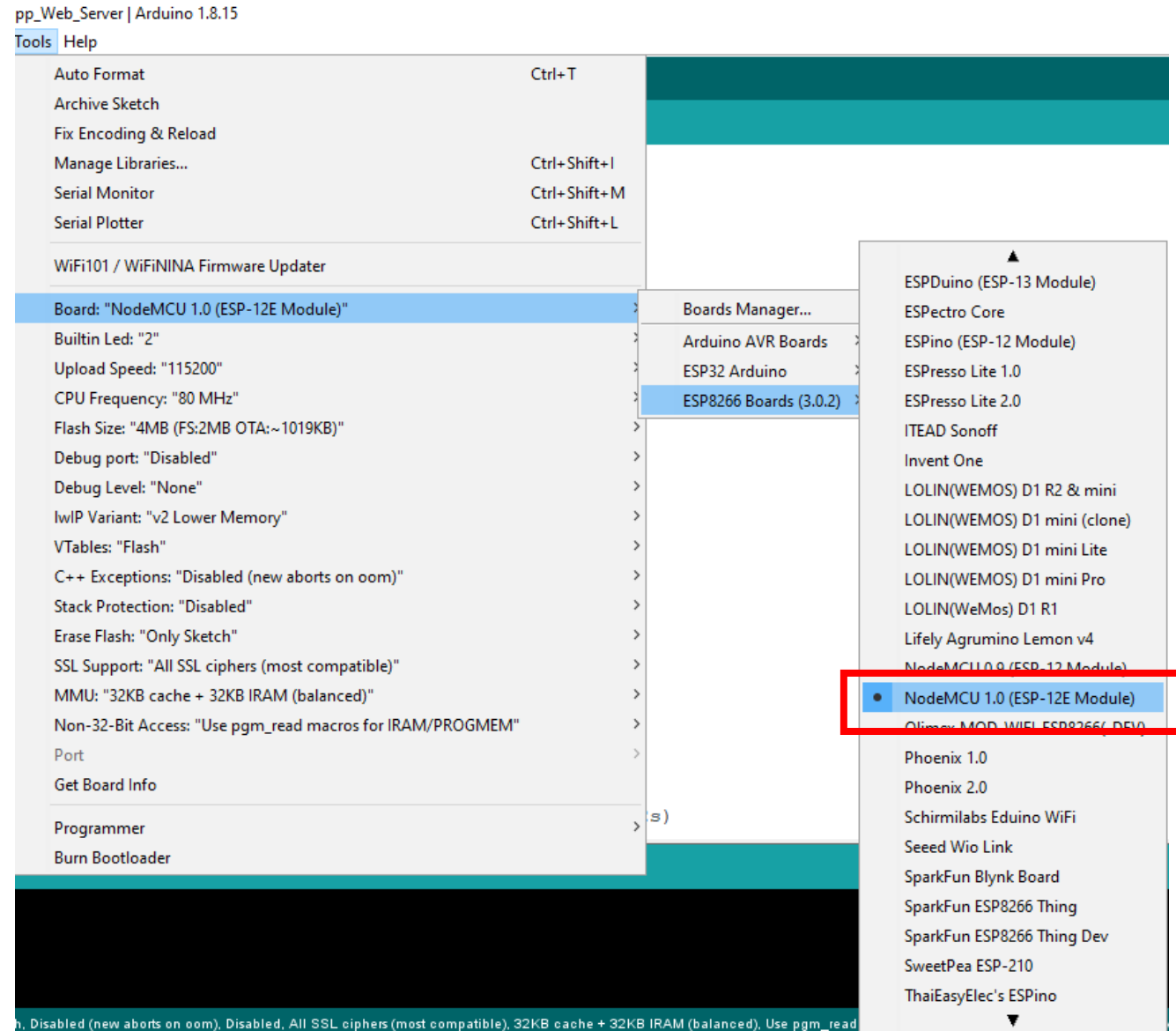
Search for ESP8266, then click install



Web Server With ESP8266

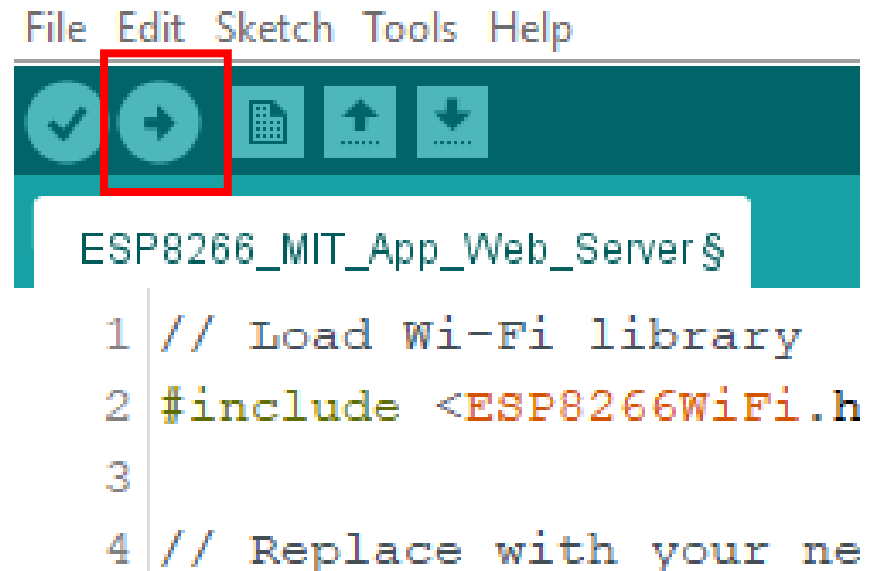
https://github.com/apauaie/esp8266_webserver

Copy the program from this link, select the correct board, and press upload

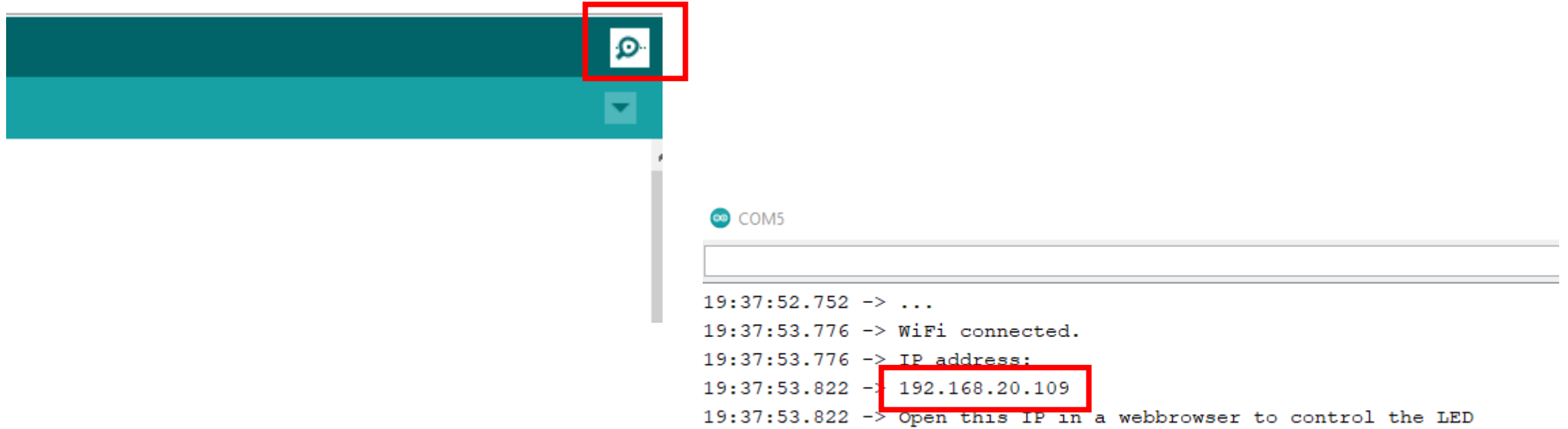


Copy the program from this https://github.com/apauaie/esp8266_webserver, fill in your Wifi name and password, then press upload

```
ESP8266_MIT_App_Web_Server$  
1 // Load Wi-Fi library  
2 #include <ESP8266WiFi.h>  
3  
4 // Replace with your network credentials  
5 char ssid[] = "Your Wifi Name";  
6 char password[] = "YourWifiPassowrd";  
7  
8 // Set web server port number to 80  
9 WiFiServer server(80);  
10
```



Open Serial Monitor, and copy the ip address



Paste the ip address in your web browser, and now you can test the button to turn the LED on and off. Dont worry if you see a Sensor Value 0 there, you will learn about it later.



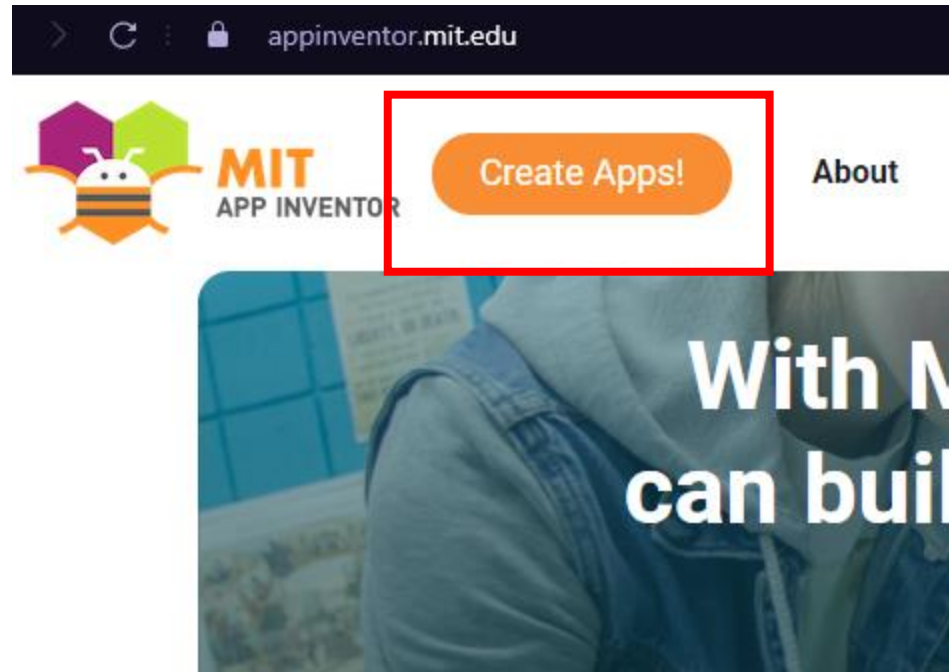
ESP32 Web Server

GPIO 2 - State off

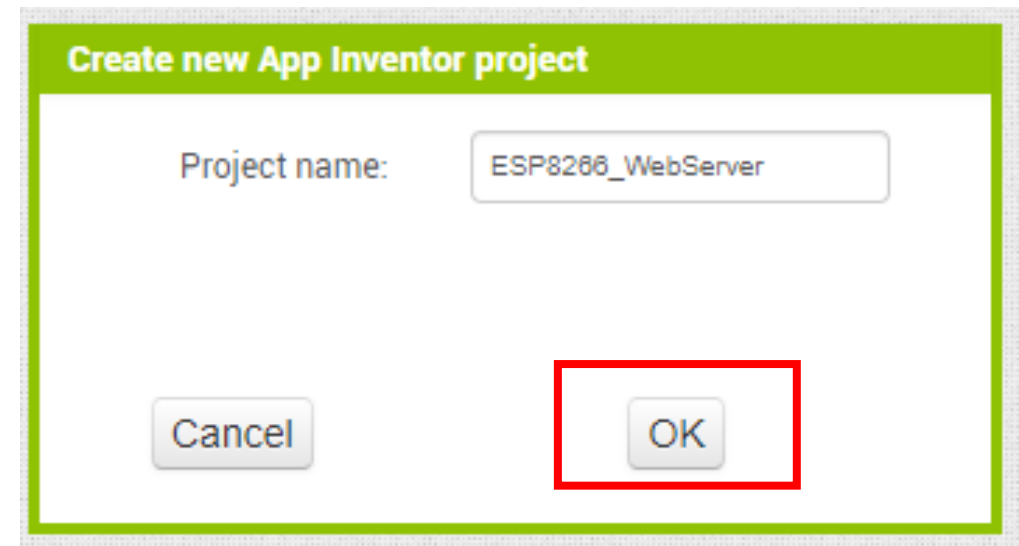
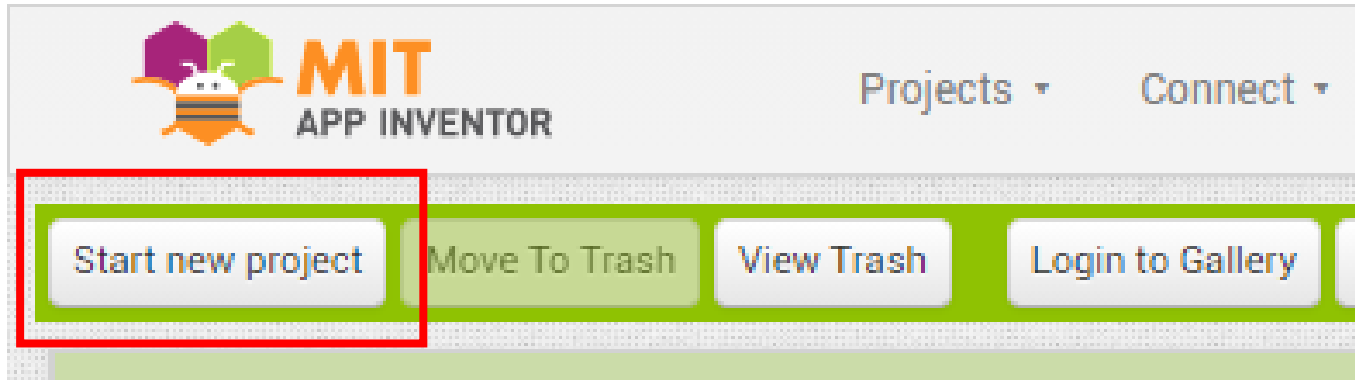


Using MIT App Inventor with the Web Server

Go to <https://appinventor.mit.edu/> , press Create Apps and login with your email.

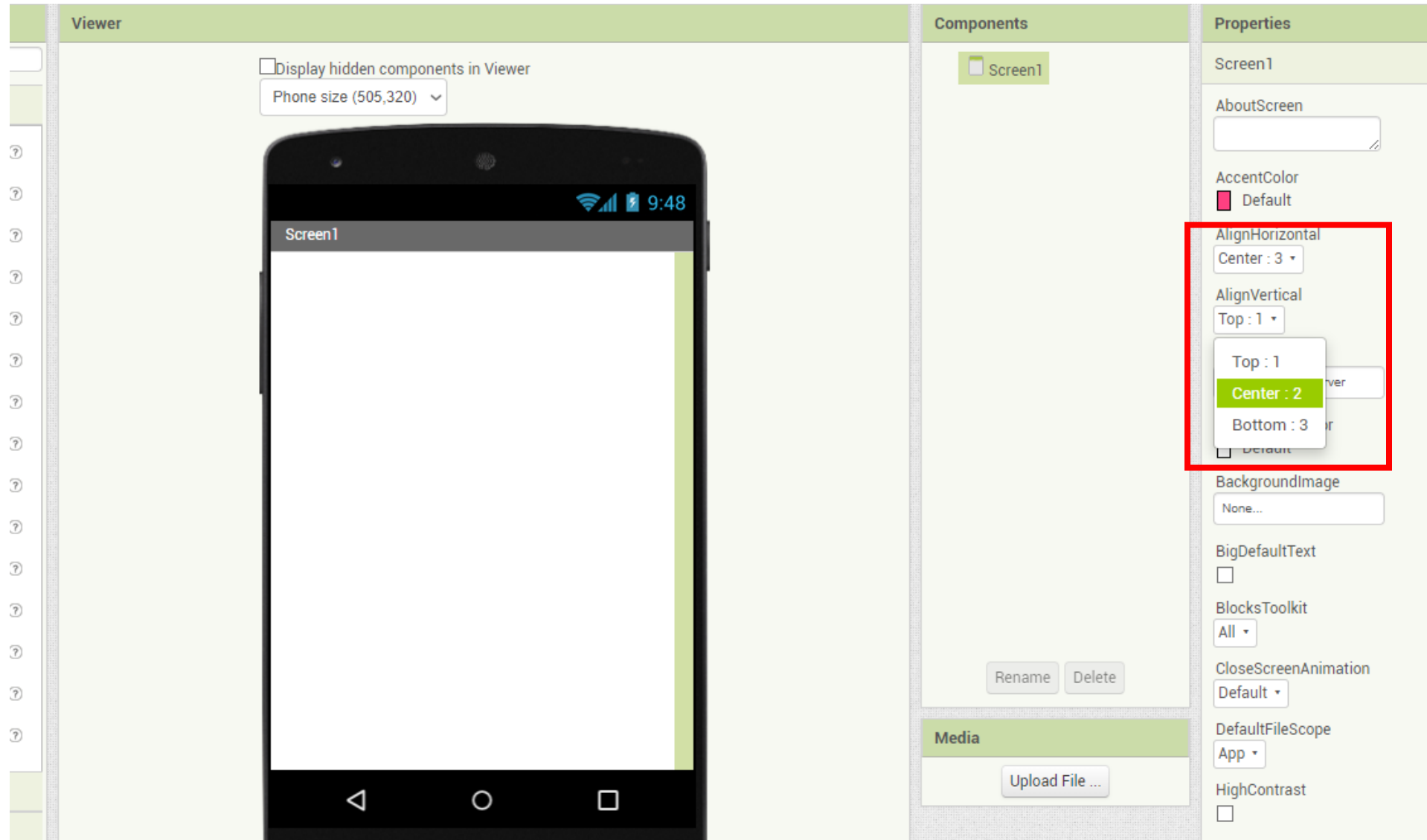


Give your project a nice name

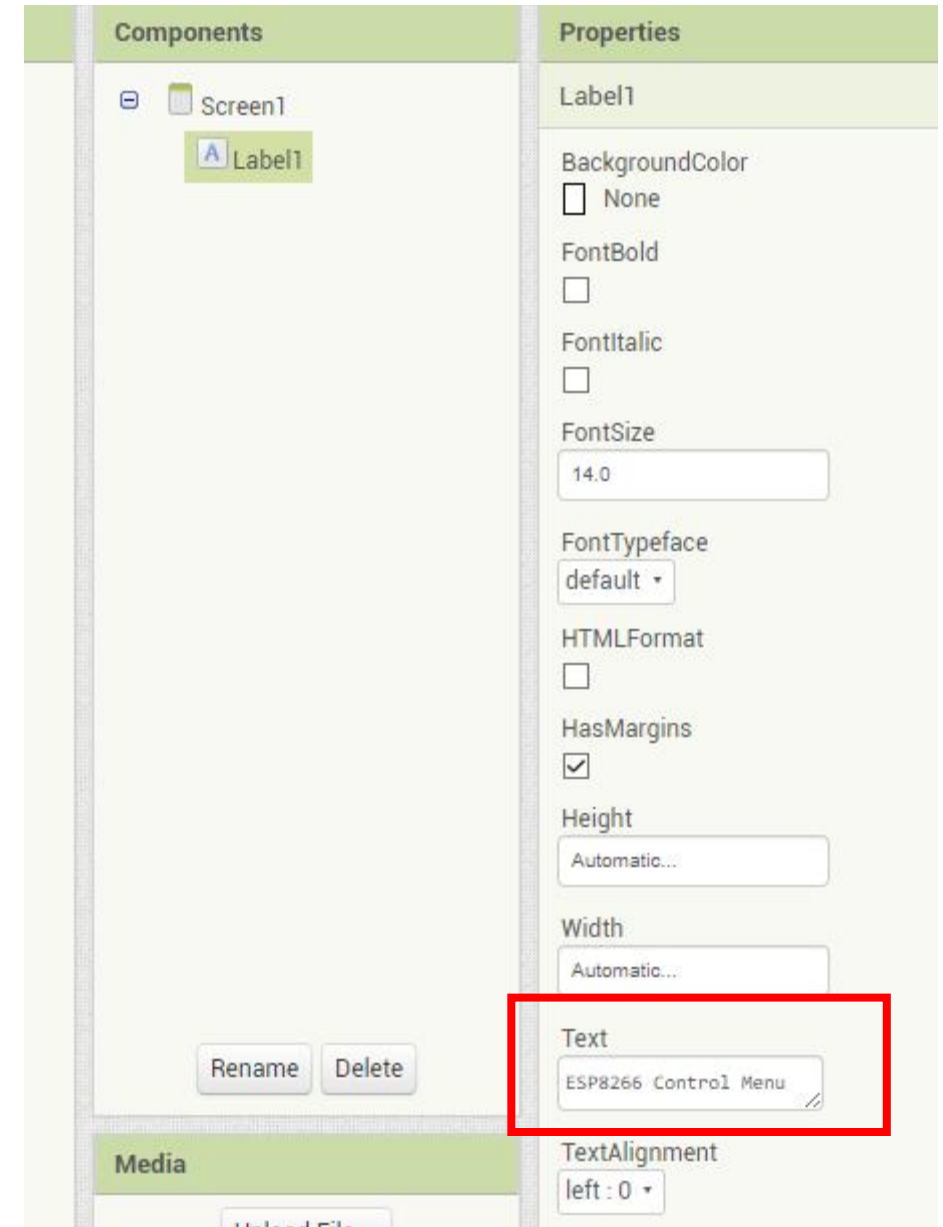
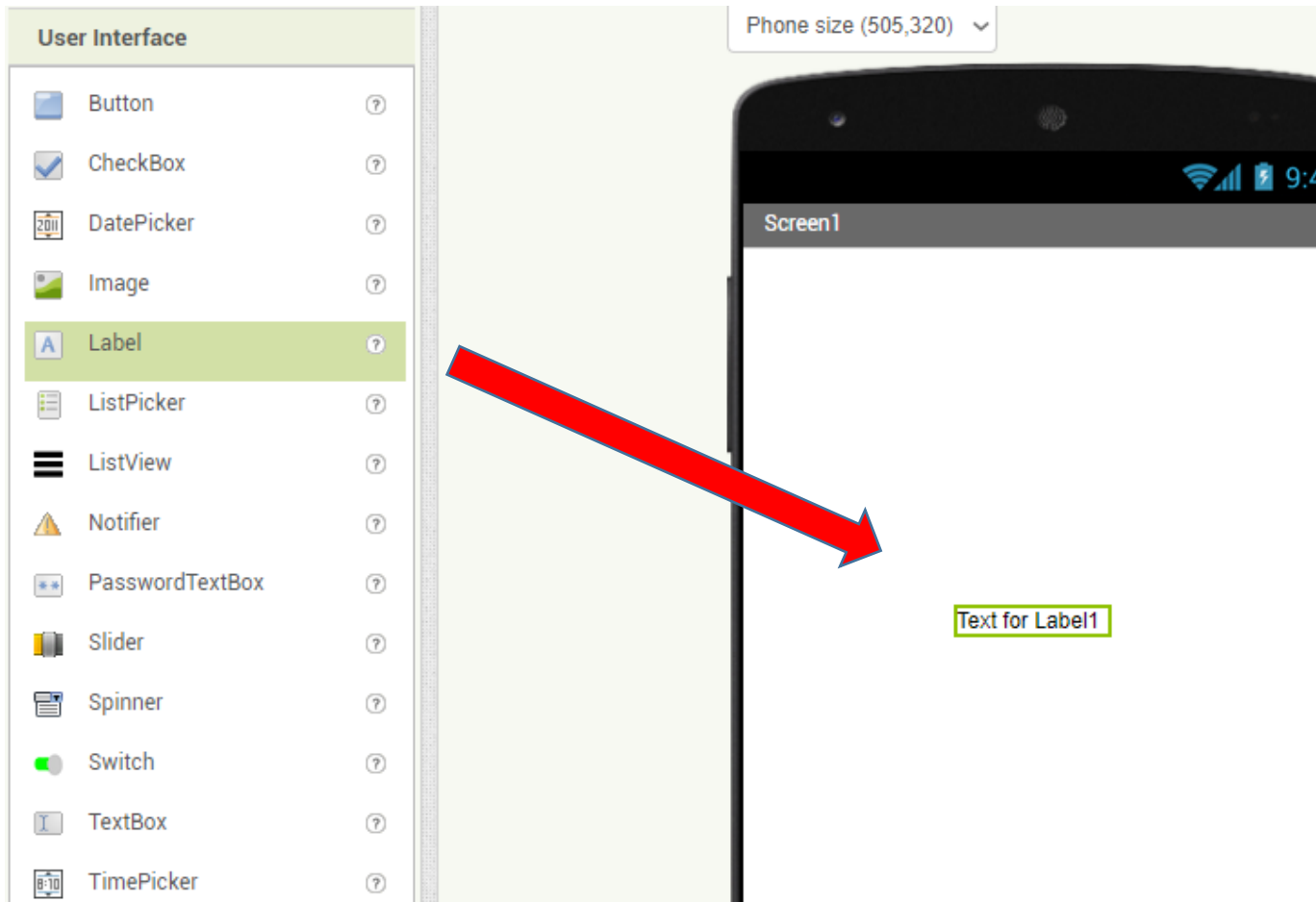


The image shows a dialog box titled 'Create new App Inventor project'. It has a green header bar. Inside, there is a label 'Project name:' followed by a text input field containing 'ESP8266_WebServer'. At the bottom, there are two buttons: 'Cancel' and 'OK'. The 'OK' button is highlighted with a red rectangular box.

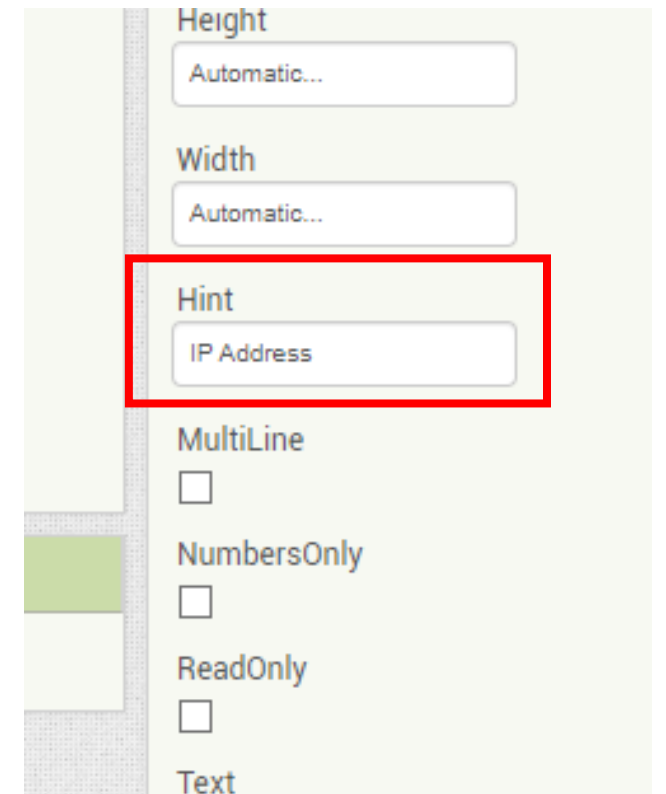
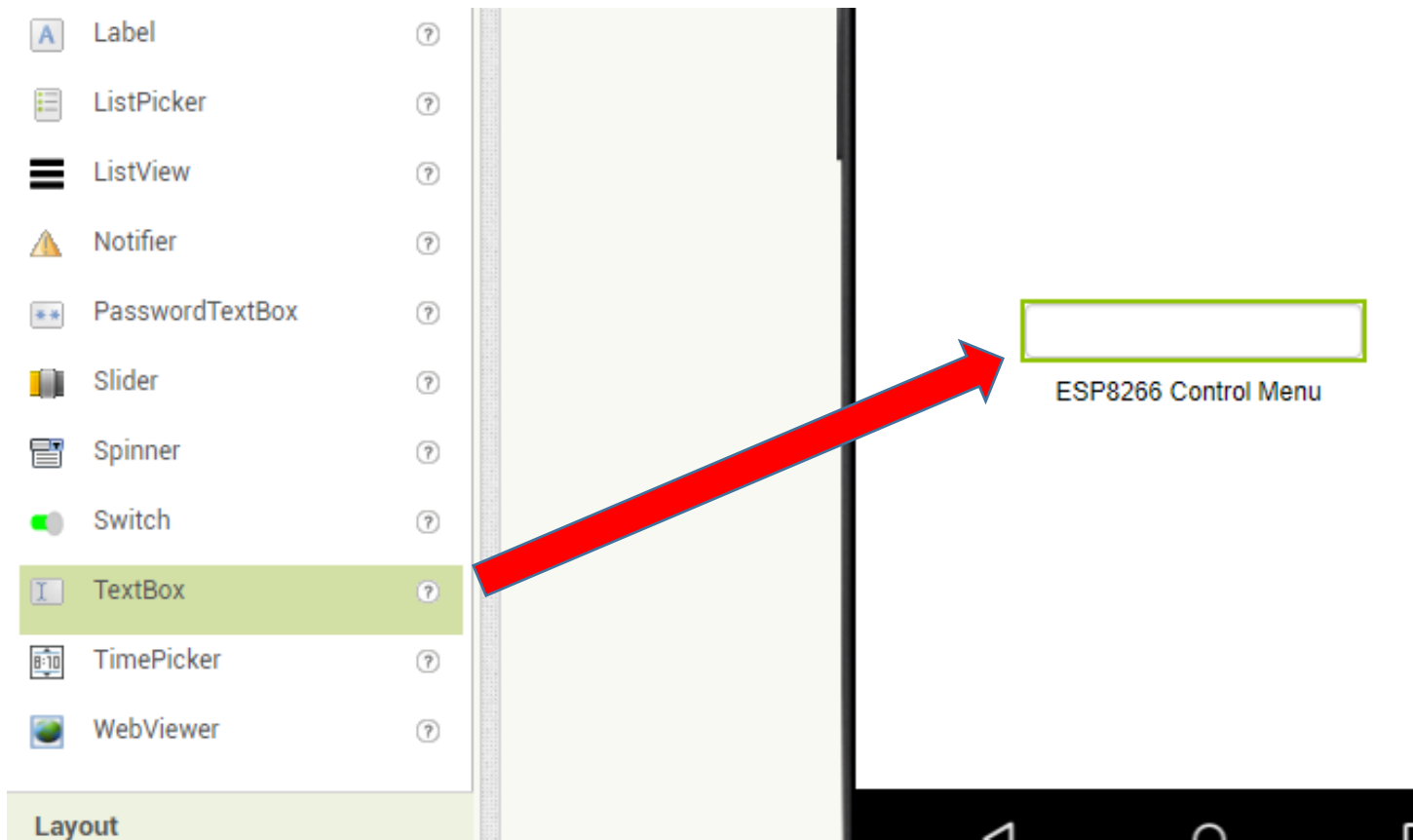
On the far right side of your screen, set these two to Center. This will make your app look nice.



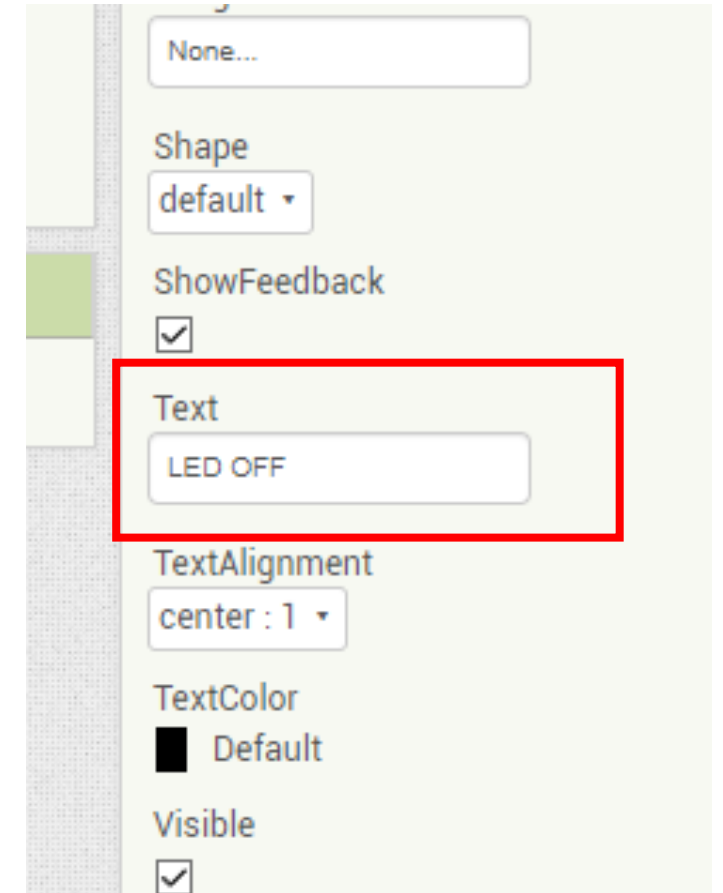
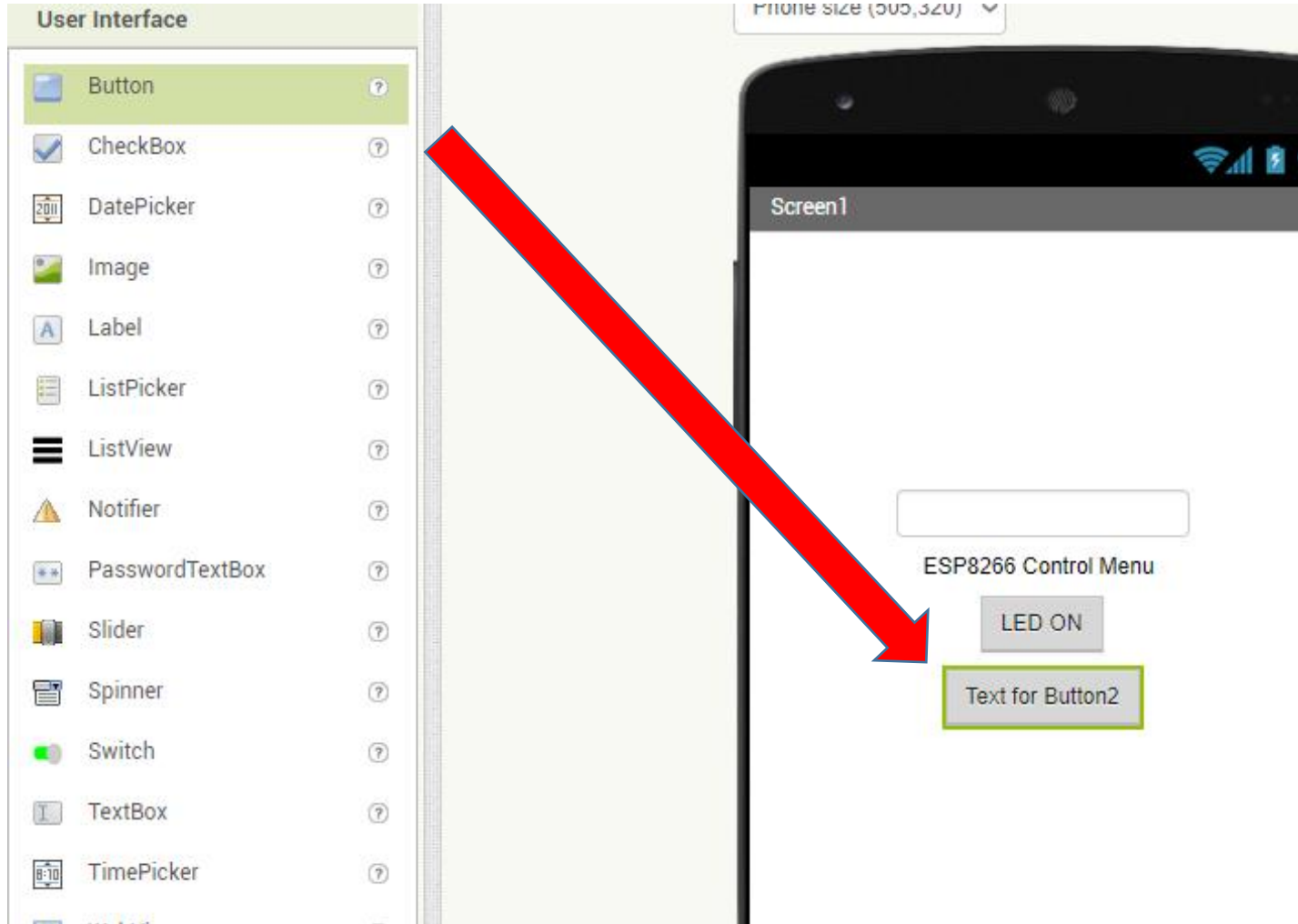
On User Interface Tab, drag a Label into your Virtual Phone Screen. Change the text to be displayed



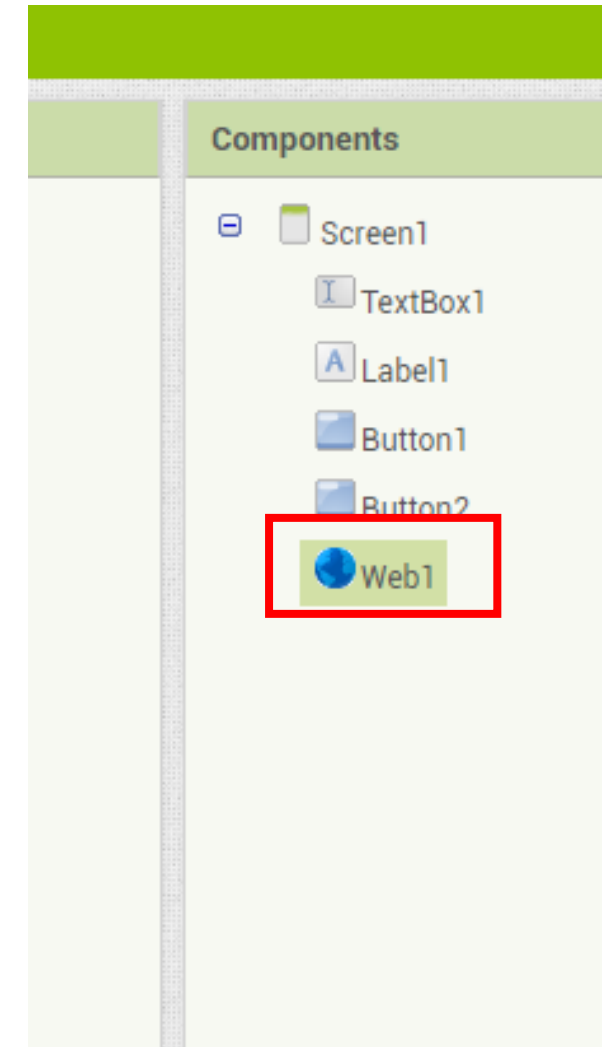
Drag a Textbox on top of your Label. Change the Hint to IP Address



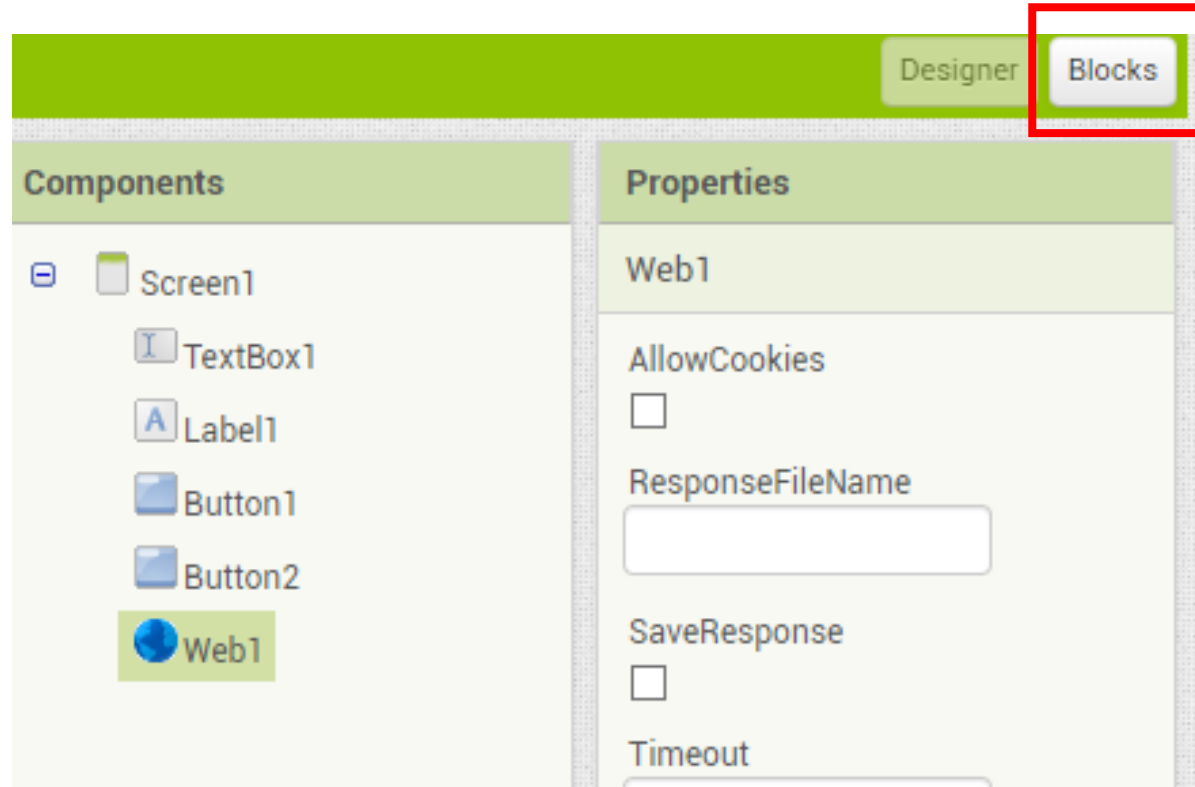
Drag a 2 Buttons into your screen. Change the Text to LED ON and LED OFF



Scroll down to Connectivity Tab, drag a Web into your Screen. It will show nothing on your screen, but it will be listed in your Components Tab.



On the upper right corner, press Blocks to open up the programming interface. Lets program our apps!



Click Button1. Drag when...Click block into the programming space.

The image shows a programming environment with two main panels: **Blocks** and **Viewer**.

Blocks Panel: This panel is on the left and contains a tree view of available blocks. Under the **Built-in** category, there are several categories listed: Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, and Procedures. Under the **Screen1** category, there are **TextBox1**, **Label1**, **Button1**, **Button2**, and **Web1**. The **Button1** block is highlighted with a red rectangle.

Viewer Panel: This panel is on the right and shows a list of event-driven blocks for **Button1**. The blocks are:

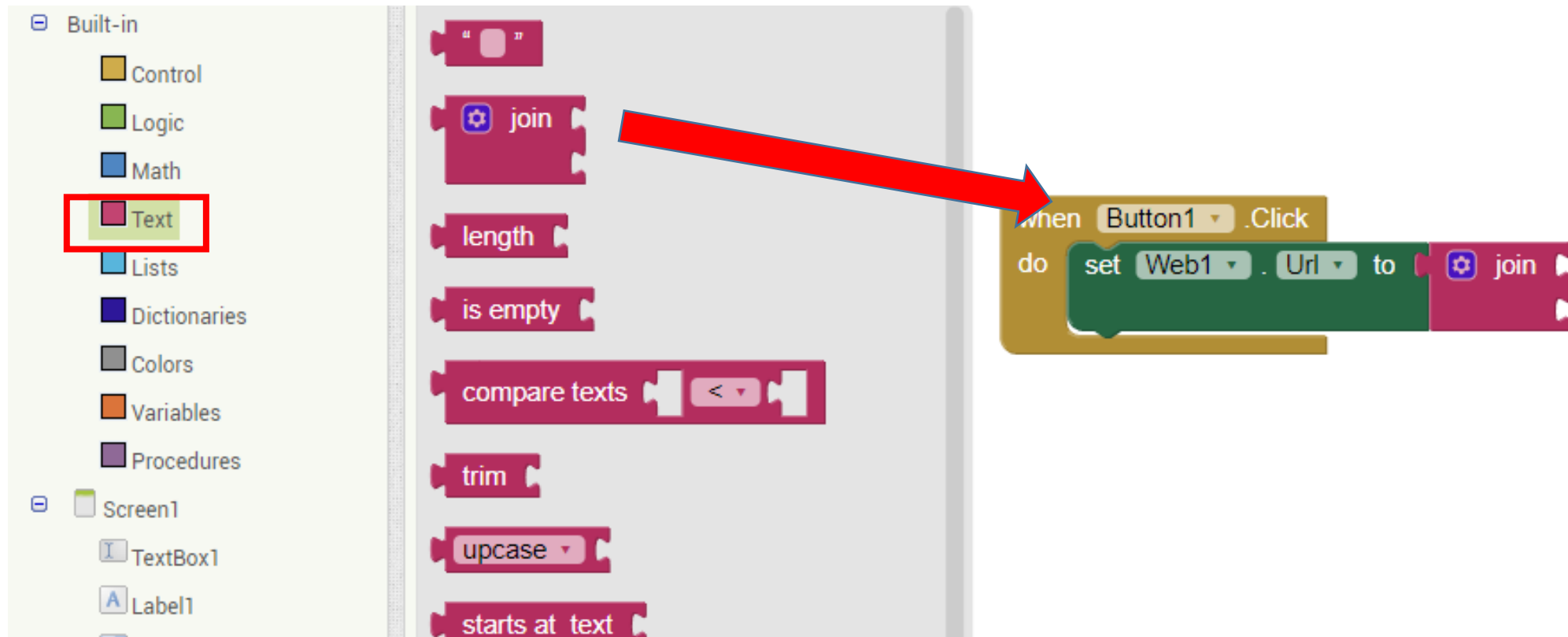
- when **Button1** .Click
- when **Button1** .GotFocus
- when **Button1** .LongClick
- when **Button1** .LostFocus
- when **Button1** .TouchDown
- when **Button1** .TouchUp

A red arrow points from the **when Button1 .Click** block in the Viewer panel to a floating **when Button1 .Click** block on the right side of the screen.

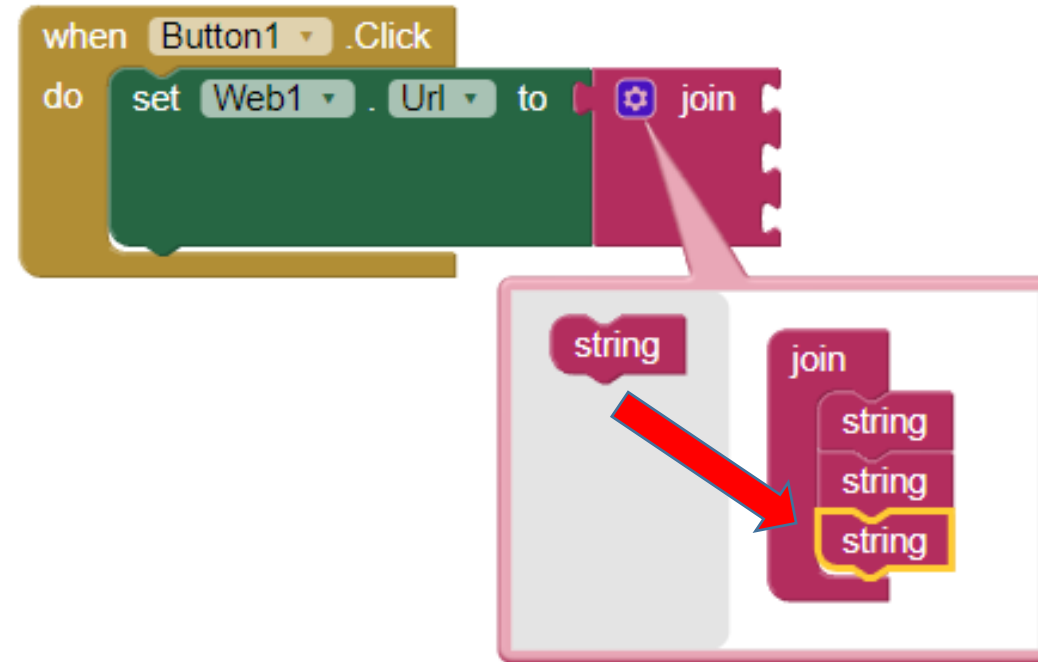
Click Web1. Scroll down, find the set..Url block, drag in into the code.

The image shows a visual programming environment. On the left is a component palette with categories: Variables, Procedures, Screen1, TextBox1, Label1, Button1, Button2, and Web1. The 'Web1' component, represented by a blue globe icon, is highlighted with a red rectangle. Below the palette are 'Rename' and 'Delete' buttons. At the bottom left is an 'Upload File ...' button. The main workspace on the right contains a sequence of code blocks for 'Web1': 'RequestHeaders', 'ResponseFileName', 'SaveResponse', 'Timeout', and 'Url'. A red arrow originates from the 'Web1' component in the palette and points to the 'set Web1 . Url to' block in the workspace. To the right of the workspace, there is a separate block structure: 'when Button1 . Click' followed by 'do' and a block containing a red 'X' icon and 'set Web1 . Url to'.

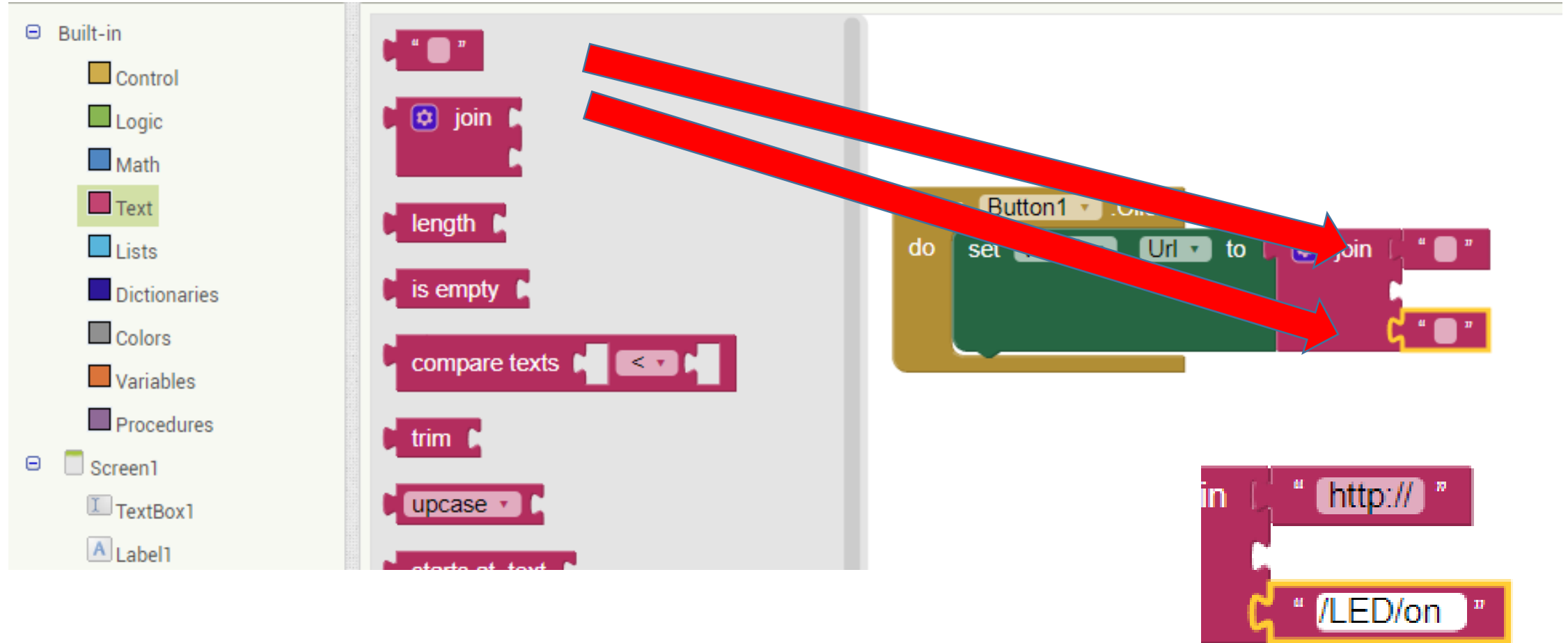
Click Text. Drag the join block.



Click the gear icon on the join block. Add a third string block into it.



Still on Text Tab, drag the empty text blocks into the first and third position of the join block.



Write “http://” on the first box, and “/LED/on” on the third, without the double quotes “ symbols.

Click Textbox1, find Textbox1 Text block, and put it at the second position of the join block.

The image displays the Scratch code editor interface. On the left, the 'Sprite' palette shows a list of objects: Screen1, Textbox1 (highlighted with a red rectangle), Label1, Button1, and Button2. The main workspace contains a script area with several blocks for 'Textbox1': 'NumbersOnly', 'ReadOnly', 'Text' (highlighted with a yellow rectangle), and 'TextColor'. A red arrow points from the 'Textbox1 Text' block in the workspace to the 'join' block in the 'do' section of a 'when Button1 Click' event. The 'join' block is currently empty and is positioned between 'set Web1 Url to' and two string blocks: 'http://' and '/LED/on'. The 'do' section also includes a 'set Web1 Url to' block.

Click Web1, drag the call..Get block

The image shows a visual programming environment with a component palette on the left and a script editor on the right.

Component Palette (Left):

- Variables
- Procedures
- Screen1
 - TextBox1
 - Label1
 - Button1
 - Button2
 - Web1** (highlighted with a red box)
- Any component
- Media
 - Unload File

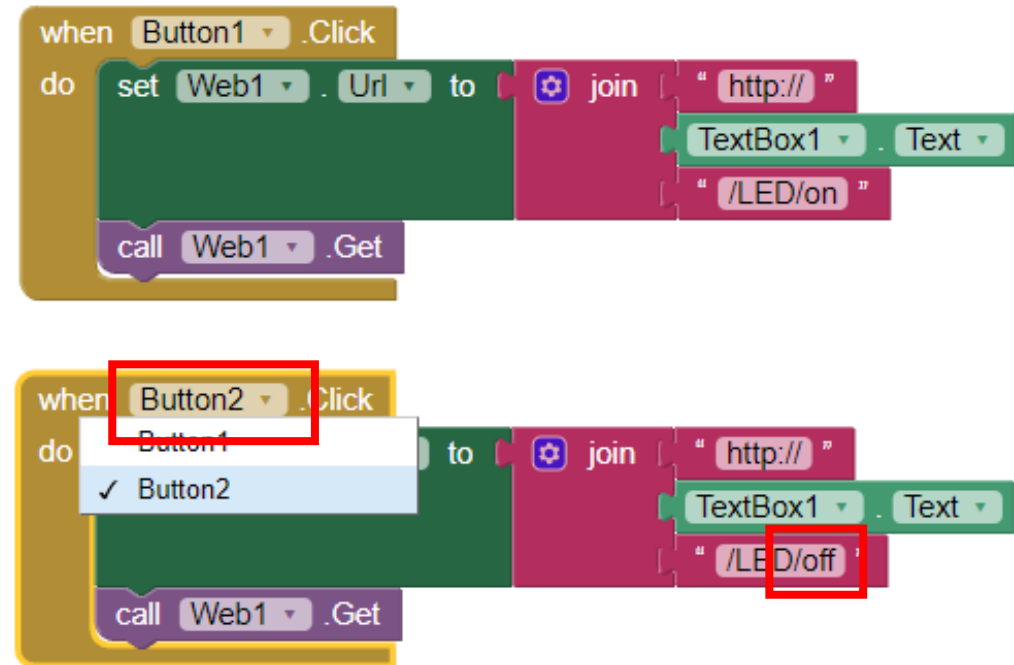
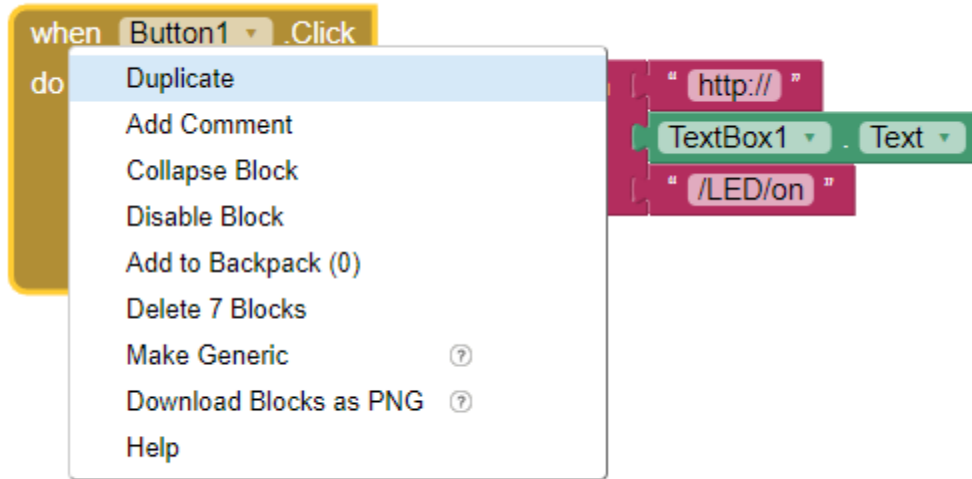
Script Editor (Right):

The script is organized into two event-driven blocks:

- when Web1 .TimedOut**
 - do
 - call Web1 .BuildRequestData list
 - call Web1 .ClearCookies
 - call Web1 .Delete
 - call Web1 .Get
 - call Web1 .HtmlTextDecode htmlText
- when Button1 .Click**
 - do
 - set Web1 .Url to join "http://" TextBox1 .Text "/LED/on"
 - call Web1 .Get

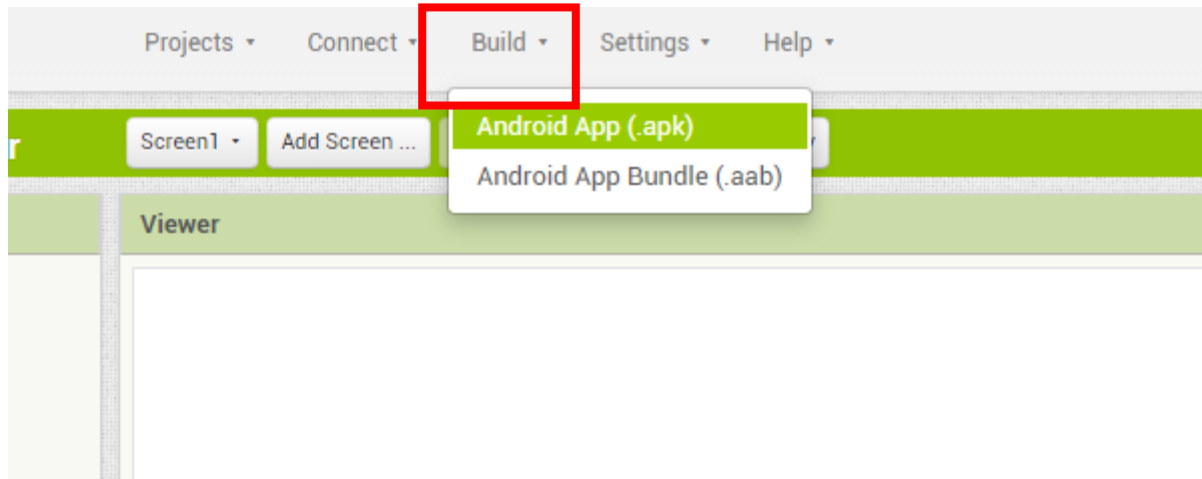
A red arrow points from the **Web1** component in the palette to the **call Web1 .Get** block in the script.

Click on the when block, press Right Click on your mouse, duplicate the whole block.

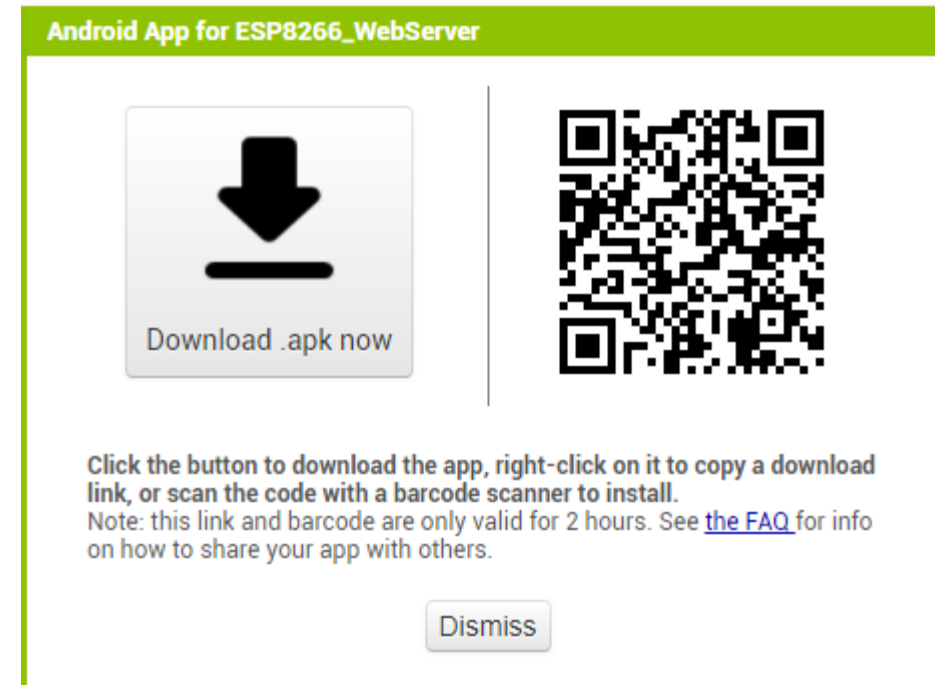


Drag the blocks to a suitable position. Change Button1 to Button2, and the “on” text, to “off”

**Now you have finish building your apps. Let's install it on your phone.
Select Build - Android App and scan the QR Code to download and
install your app**

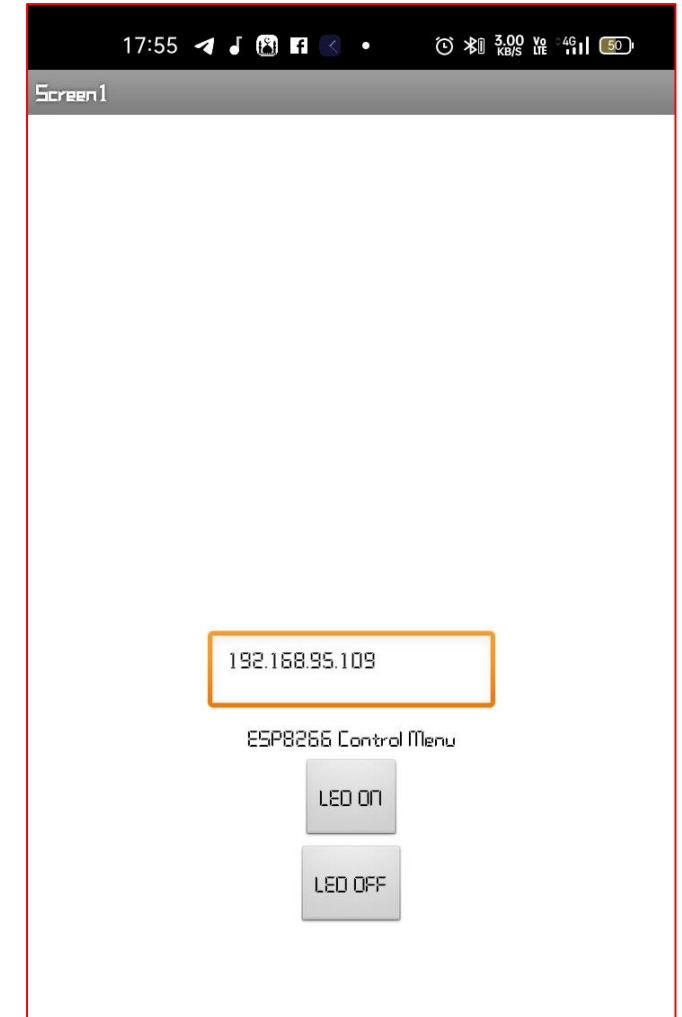
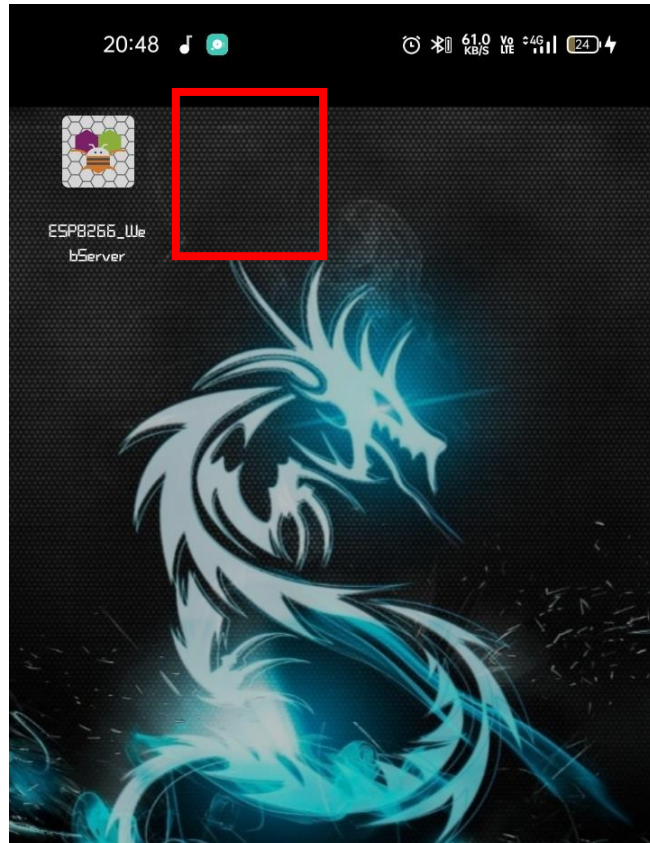


You might get some warning when installing on your phone. Its ok to ignore and proceed. You make your own apps, theres nothing to worry.

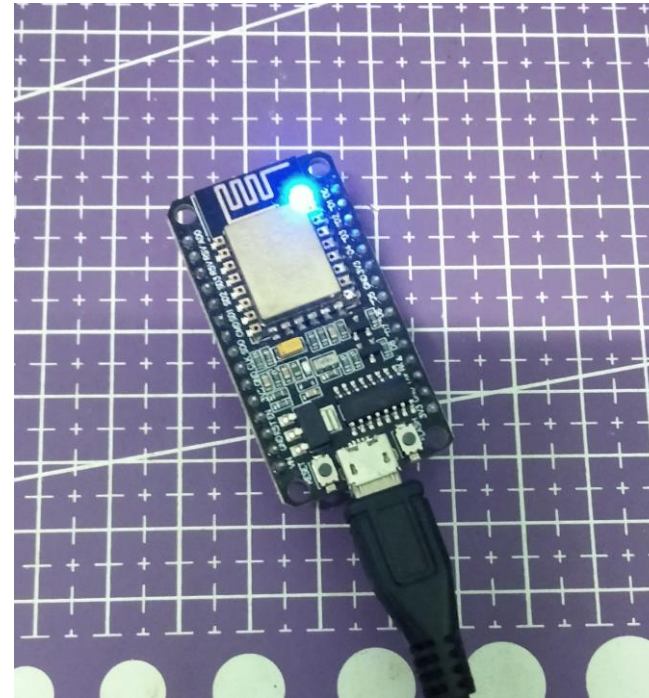
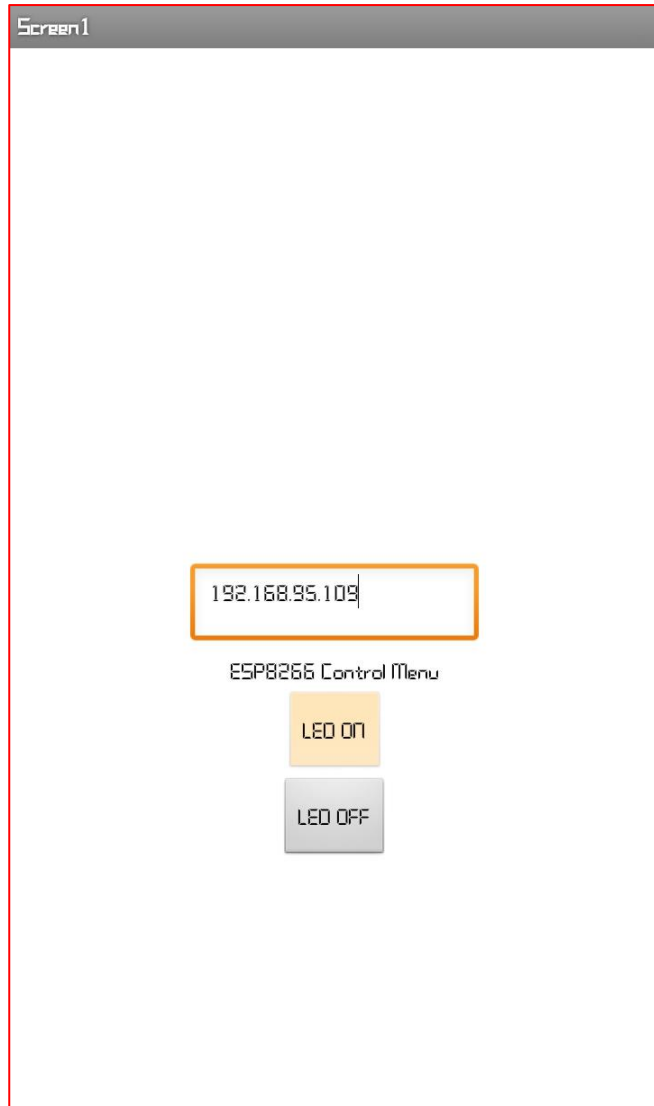


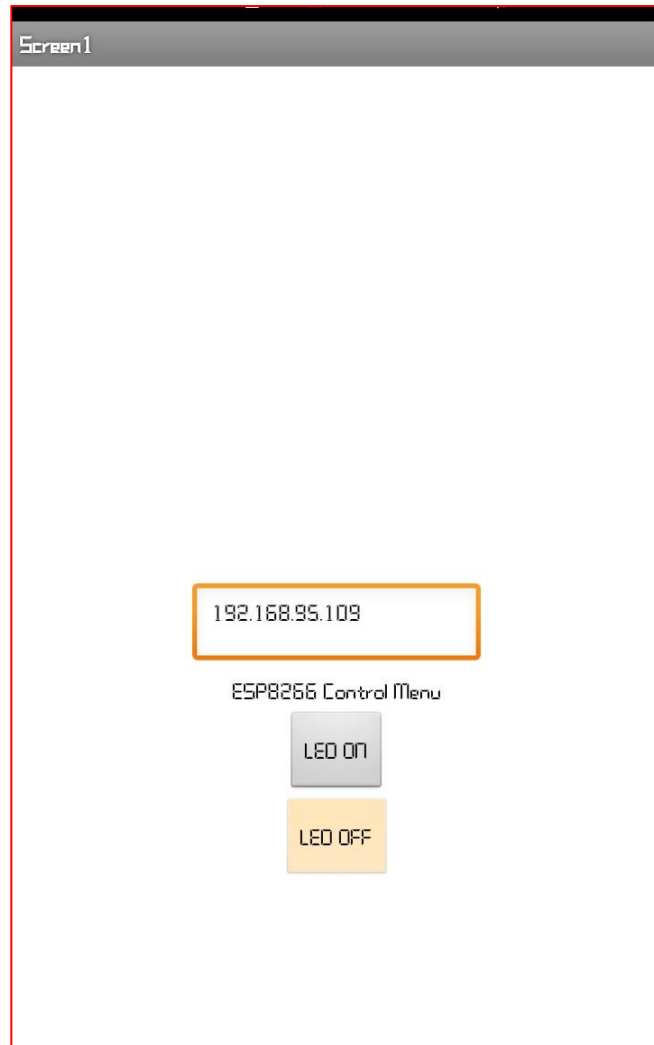
Now let's open your app. Make sure your phone is connected to the same Wifi with your ESP8266 Board. Mobile Hotspot also works.

After you input your IP Address, you can straight away play with the buttons and see the built-in LED turns On and Off

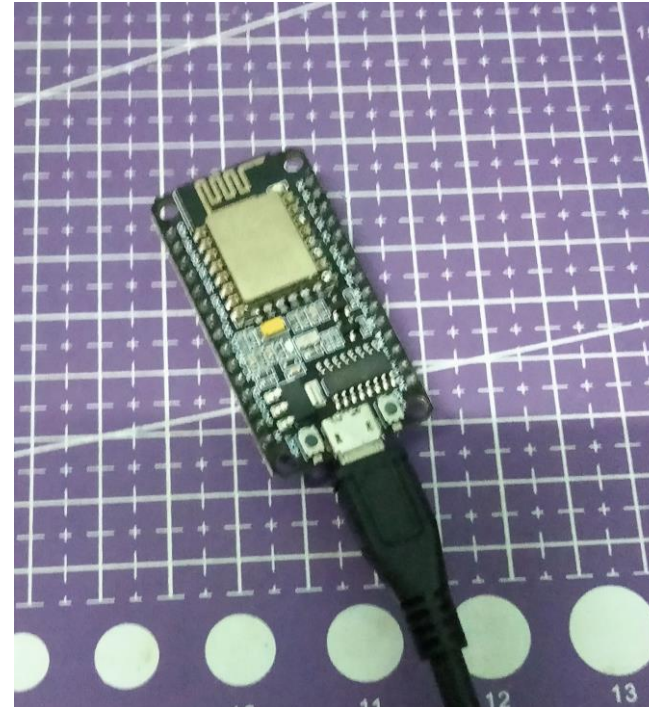


LED On



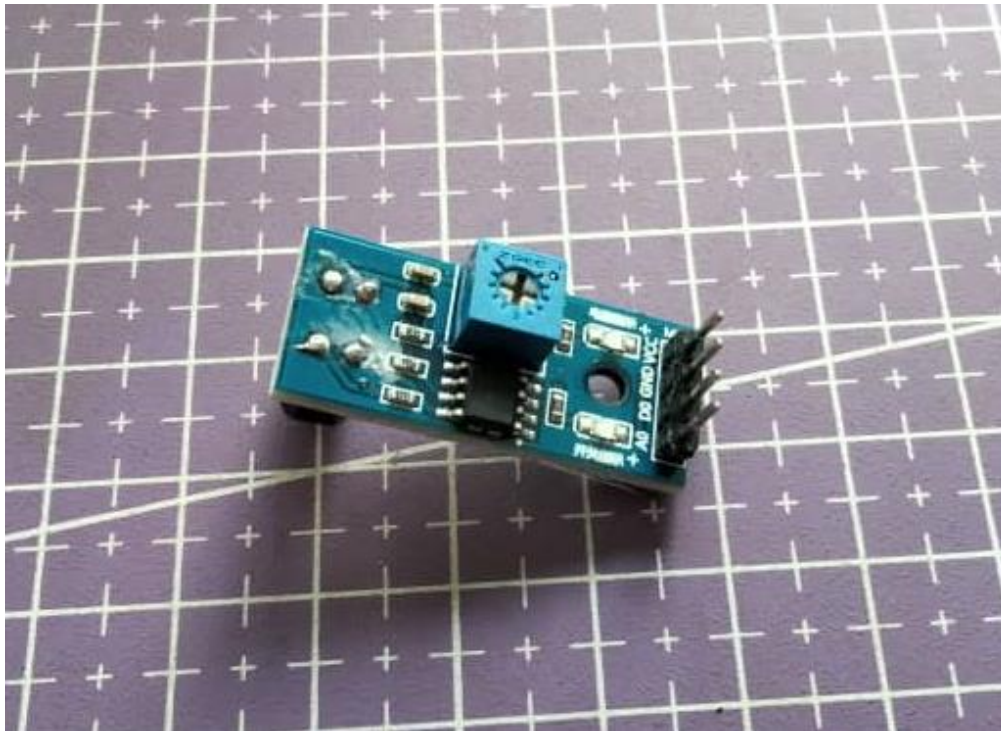


LED Off



Adding Analog Sensor Reading into your Project

We will use Infrared Line Sensor in this example. But you can use any basic analog sensors like obstacle sensors, ultrasonics, temperature sensors, or as simple as a potentiometer.

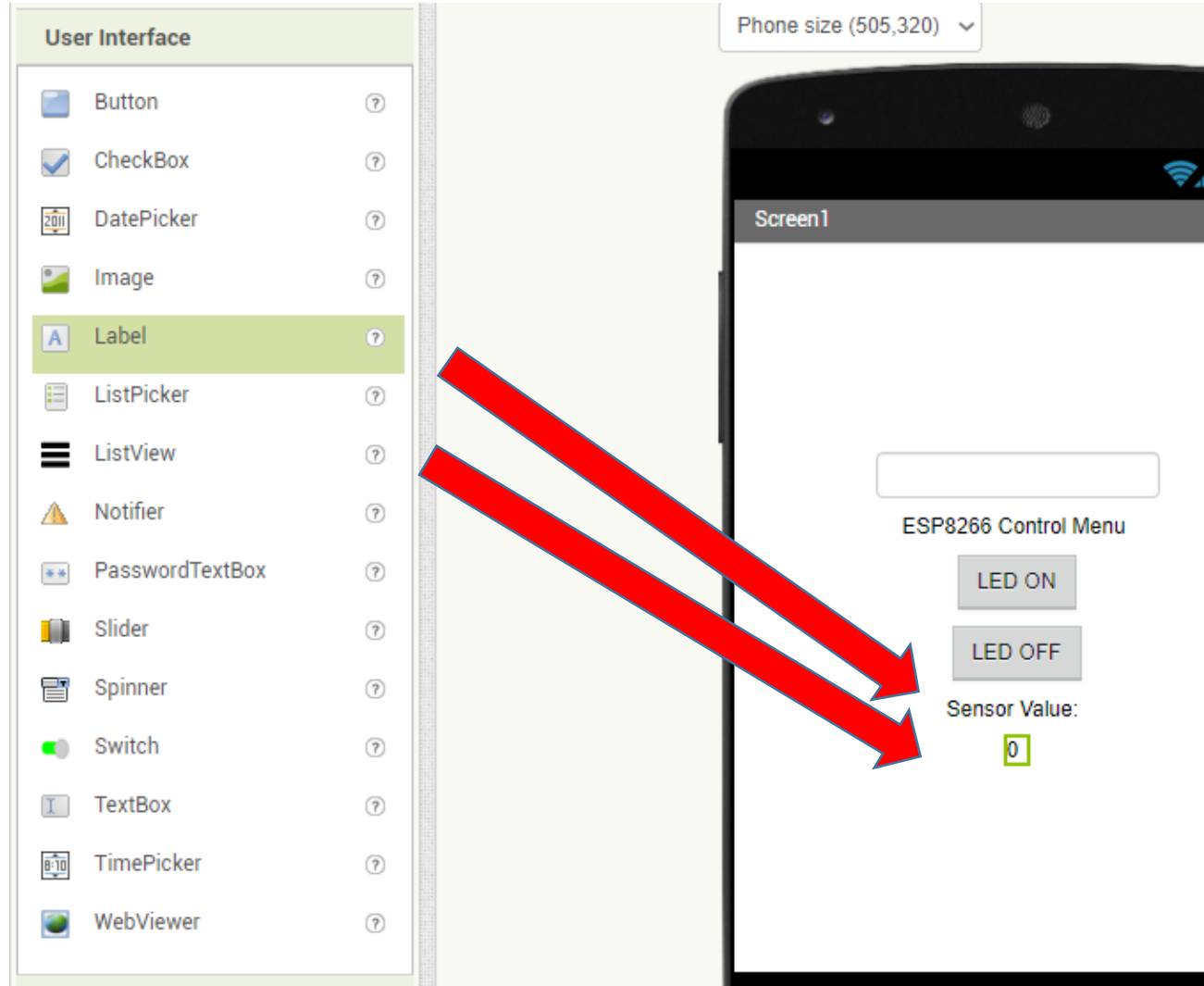


Connect the pins from the sensor to ESP8266 board:

Sensor Pin		ESP8266 Pin
VCC / + / 5v	->	3V3
A0 / Data	->	A0
GND / -	->	GND

You dont need to change anything on the ESP8266 coding. All already being set. We just need to expand our apps program.

On your MIT inventor website, at the top right, go back to Designer tab. Drag two labels into your screen. Change the text to Sensor Value:, and 0.



Go to Sensors Tab, Drag a Clock into your screen. Same as Web, you will not see anything on your screen, but on your Components tab. Change the TimeInterval to 3000.

The image shows a three-part screenshot of an IDE interface. On the left, the 'Sensors' tab is selected in the component palette, with the 'Clock' component highlighted. A large red arrow points from the 'Clock' component to the main design area. The main design area shows a screen titled 'ESP8266 Control Menu' with a text input field, two buttons labeled 'LED ON' and 'LED OFF', and a label 'Sensor Value:' with the value '0'. On the right, the 'Properties' panel for the 'Clock1' component is shown, with the 'TimeInterval' property set to '3000' and highlighted by a red box.

Maps

Sensors

- AccelerometerSensor
- BarcodeScanner
- Barometer
- Clock**
- GyroscopeSensor
- Hygrometer
- LightSensor
- LocationSensor
- MagneticFieldSensor
- NearField

ESP8266 Control Menu

LED ON

LED OFF

Sensor Value:
0

Designer Blocks

Properties

Clock1

TimerAlwaysFires
☒

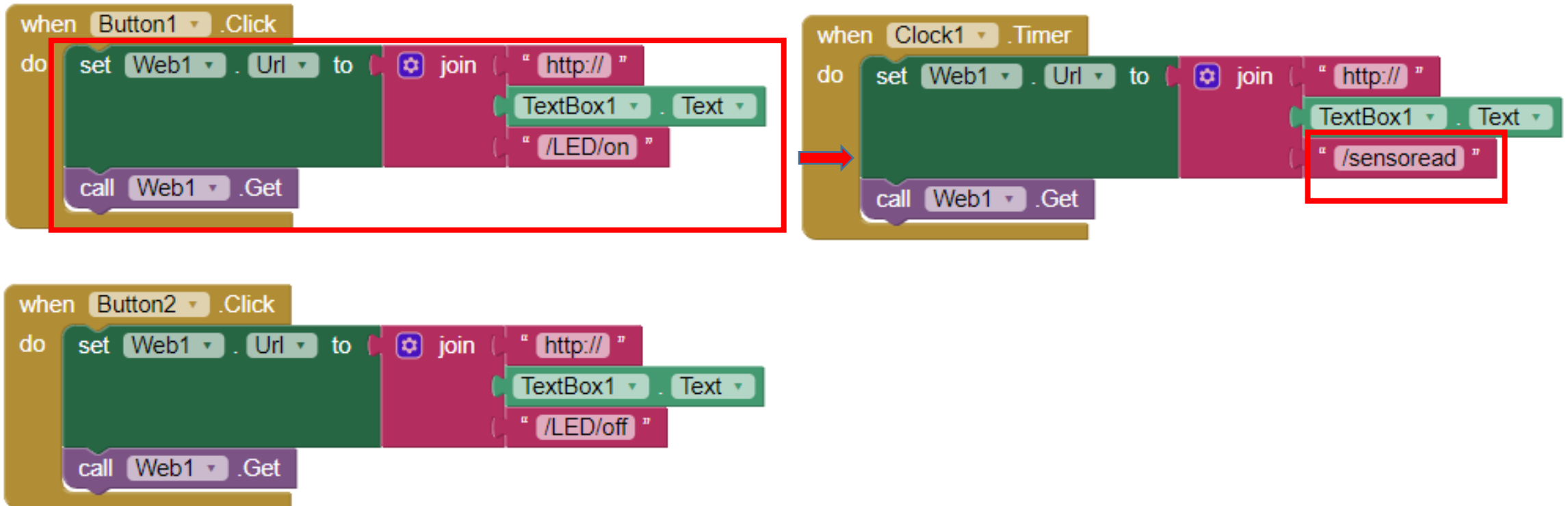
TimerEnabled
☒

TimeInterval
3000

Lets go back to our Blocks Tab. Find some blank space to continue our coding. Click on Clock1, drag the when..Time block into our code.

The image shows the Visual Studio Code Blocks editor interface. On the left, the 'Blocks' panel lists various categories: Logic, Math, Text, Lists, Dictionaries, Colors, Variables, and Procedures. Under the 'Screen1' category, 'Web1' and 'Clock1' are listed, with 'Clock1' highlighted by a red box. On the right, the 'Viewer' panel shows a sequence of blocks: a 'when Clock1.Timer' block followed by a 'do' block containing several 'call Clock1.Add' blocks (Days, Duration, Hours, Minutes, Months) and a 'join' block. A red arrow points from the 'when Clock1.Timer' block in the Viewer to a new 'when Clock1.Timer' block being dragged into the code area.

Remember our previous blocks? Duplicate the set..Url and call..Get blocks into our when..Timer block. Change the text on the 3rd position of join block, into “/sensoread”.



Next, click on Web1. Drag when..GotText block.

The image shows a visual programming environment with two main panels: 'Blocks' on the left and 'Viewer' on the right.

Blocks Panel:

- Categories: Logic, Math, Text, Lists, Dictionaries, Colors, Variables, Procedures.
- Components: Screen1, TextBox1, Label1, Button1, Button2, Label2, Label3, **Web1** (highlighted with a red box), Clock1, Any component.

Viewer Panel:

- Contains several event-driven blocks for 'Web1' and 'Clock1'.
 - when Web1 .GotFile** block with fields: url, responseCode, responseType, fileName. The 'do' section contains a 'TextBox1 .Text' block.
 - when Web1 .GotText** block with fields: url, responseCode, responseType, responseContent. The 'do' section is empty.
 - when Button2 .Click** block.
 - when Web1 .TimedOut** block with fields: url. The 'do' section contains a 'join' block with 'http://' and a 'TextBox1 .Text' block.
 - call Web1 .BuildRequestData** block with a 'list' output.
 - call Web1 .ClearCookies** block.
 - call Web1 .Delete** block.
- Clock1 .Timer** block with a 'do' section containing:
 - set Web1 .Url** to a 'join' block with 'http://' and a 'TextBox1 .Text' block.
 - call Web1 .Get** block.

A red arrow points from the 'when Web1 .GotText' block in the Viewer panel to a separate, highlighted 'when Web1 .GotText' block on the right, indicating it is the block being dragged.

Click on Control Tab, drag if..then block

The image shows a block-based programming environment with two main panels: 'Blocks' on the left and 'Viewer' on the right.

Blocks Panel:

- Under the 'Built-in' category, the 'Control' tab is highlighted with a red rectangle.
- Other categories visible include Logic, Math, Text, Lists, Dictionaries, Colors, Variables, and Procedures.
- Under the 'Screen1' category, various UI elements like TextBox1, Label1, and Button1 are listed.

Viewer Panel:

- It displays a script with several blocks, including 'if..then' blocks and 'when' blocks.
- A red arrow points from the 'if..then' block in the 'Blocks' panel to the 'if..then' block in the 'Viewer' panel, indicating a drag-and-drop action.

Script Details:

- Top Script:** A 'when Clock1.Timer' block followed by a 'do' block containing 'set Web1.Url to' and 'join' blocks.
- Bottom Script:** A 'when Web1.GotText' block followed by a 'do' block containing an 'if..then' block.

Click on Logic Tab, drag the = block into the position.

The image shows a block-based programming environment. On the left, a 'Built-in' library is open, with the 'Logic' tab highlighted by a red rectangle. Below the library, a list of components is visible: Screen1, TextBox1, Label1, Button1, and Button2. The main workspace contains several code blocks. A red arrow points from an equals (=) block in the Logic tab to an 'if' block within a 'when Web1 GotText' event handler. The 'if' block is currently empty, and the equals block is being positioned to be placed inside it. Other visible code blocks include 'true' and 'false' blocks, 'join' blocks for constructing URLs, 'set Web1 Url to' blocks, 'TextBox1 Text' blocks, and 'call Web1 Get' blocks. The overall layout is typical of a visual programming IDE like MIT App Inventor.

Click on Variables Tab, drag the get block into the first position of = block.

The screenshot displays a visual programming environment with two main panels: 'Blocks' on the left and 'Viewer' on the right.

Blocks Panel:

- Built-in:**
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Dictionaries
 - Colors
 - Variables** (highlighted with a red box)
 - Procedures
- Screen1:**
 - TextBox1
 - Label1
 - Button1
 - Button2
 - Label2

Viewer Panel:

The 'Viewer' panel shows a script with several blocks. A red arrow points from the 'get' block in the 'Variables' tab to the first position of the '=' block in a script.

The script in the 'Viewer' panel includes:

- Initialize Global:** 'initialize global name to' block.
- Do:**
 - 'set Web1 . Url to' block.
 - 'join' block with 'http://' and 'TextBox1 . Text'.
 - '/LED/on' block.
 - 'call Web1 . Get' block.
- Initialize Local:** 'initialize local name to' block.
- In:** 'in' block.
- Initialize Local:** 'initialize local name to' block.
- In:** 'in' block.
- Join:** 'join' block with 'http://' and 'TextBox1 . Text'.
- /LED/off:** block.
- Call:** 'call Web1 . Get' block.

The script also includes a 'when Clock1 . Timer' block and a 'when Web1 . GotText' block, both containing 'do' loops with 'set Web1 . Url to' and 'call Web1 . Get' blocks.

Drag the empty text block from Text Tab into the second position of the = block.

The image displays the Scratch interface with the 'Blocks' palette on the left and the 'Viewer' area on the right. In the 'Blocks' palette, the 'Text' category is highlighted with a red box. A red arrow points from an empty text block in the 'Text' category to the second position of an equals block in the code editor.

Blocks Palette:

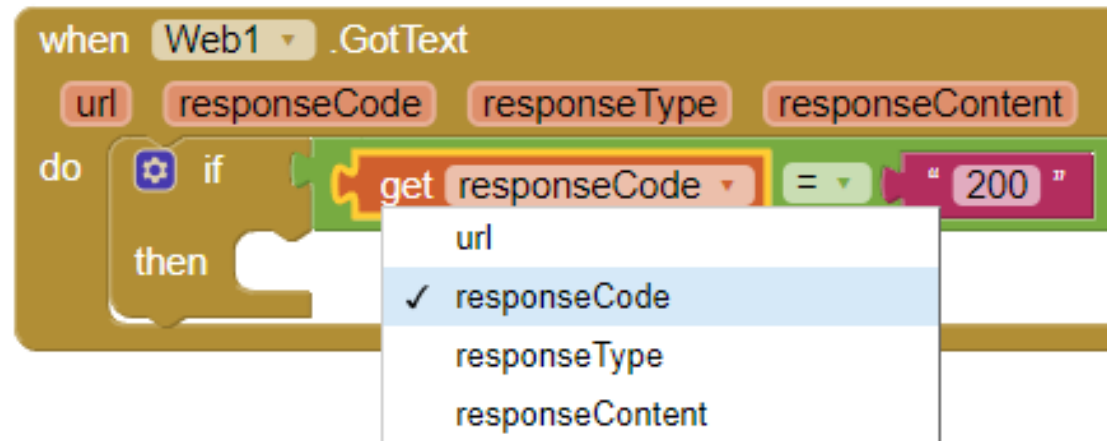
- Built-in
 - Control
 - Logic
 - Math
 - Text**
 - Lists
 - Dictionaries
 - Colors
 - Variables
 - Procedures
- Screen1
 - TextBox1
 - Label1
 - Button1
 - Button2

Viewer Area:

The code editor shows several scripts. A red arrow points from an empty text block in the 'Text' category to the second position of an equals block in the following script:

```
when Web1 . Text
  url responseCode type responseContent
do
  if
  then
    get =
```

Choose the `responseCode` on the `Get` block, and input the value `"200"` in the empty text block.



Click on Label3, drag the set..text..to block

The image shows a Scratch-like IDE with a project titled "Screen1". The "Objects" palette on the left lists "TextBox1", "Label1", "Button1", "Button2", "Label2", and "Label3". "Label3" is highlighted with a red rectangle. The "Scripts" area shows a script for "Label3" with the following blocks: "HTMLContent", "HasMargins", "set HasMargins to", "call Web1 .Get", "Height", "set Height to", "set HeightPercent to", "Text", and "set Text to". A red arrow points from the "set Text to" block to a "when Web1 .GotText" event listener script. This script has a "do" block with an "if" condition: "get responseCode = 200". If true, it executes "set Label3 . Text to".

Dictionaryes
Colors
Variables
Procedures

Screen1

TextBox1
Label1
Button1
Button2
Label2
Label3
Web1

Rename Delete

Label3 . HTMLContent
do set Web1 . Url to join "http://" TextBox1 . Text " /LED/off "
set Label3 . HasMargins to
call Web1 .Get
Label3 . Height
set Label3 . Height to
set Label3 . HeightPercent to
Label3 . Text
set Label3 . Text to

when Web1 .GotText
url responseCode responseType responseContent
do if get responseCode = 200
then set Label3 . Text to

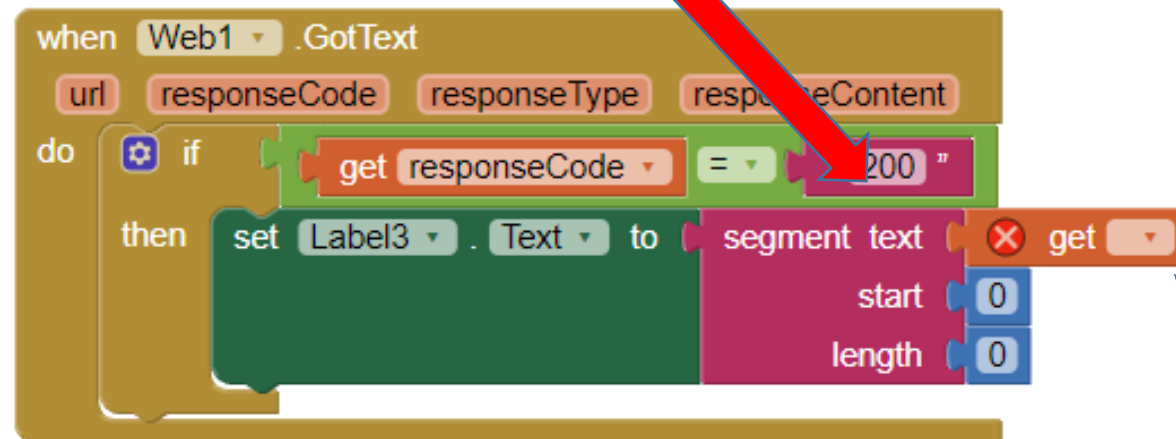
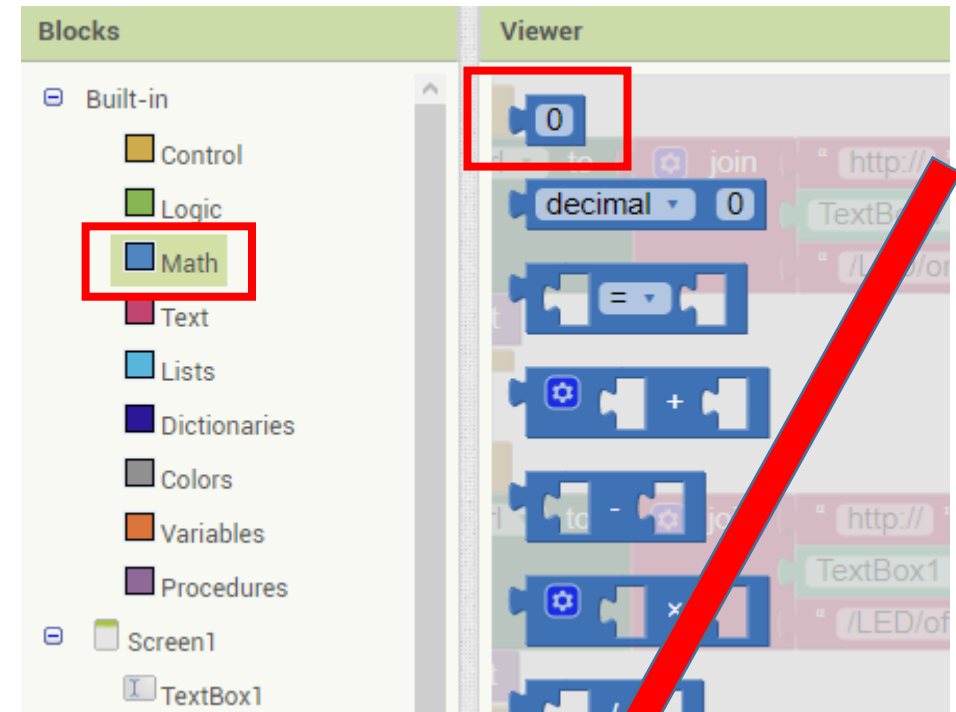
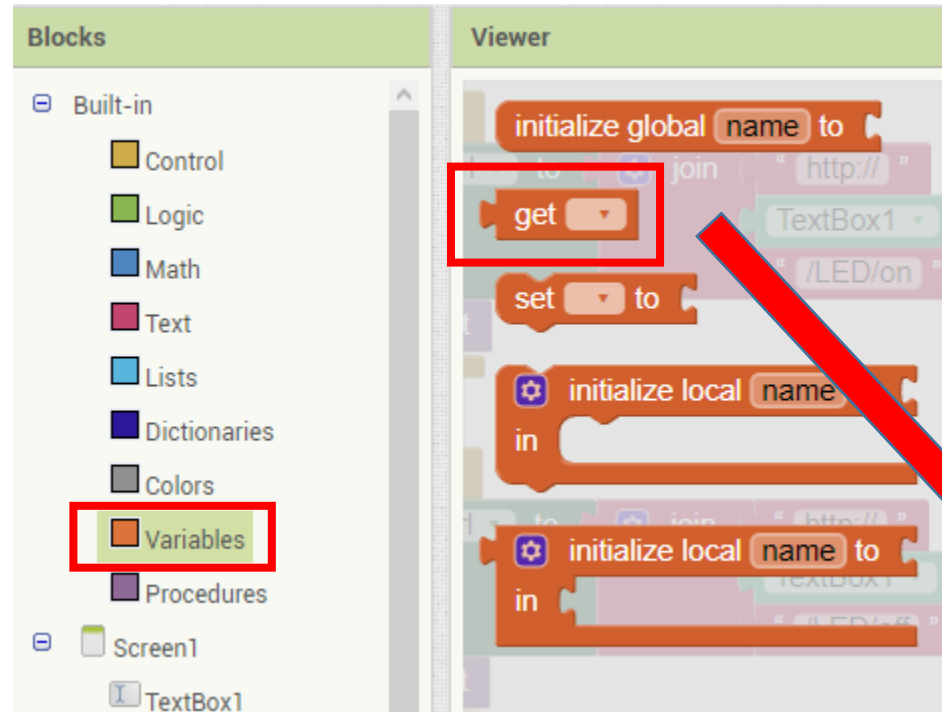
Click on Text Tab, drag the segment block

The image shows a visual programming environment with a left sidebar containing various categories: Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, and Procedures. The 'Text' category is highlighted with a red box. Below the categories is a list of objects: Screen1, TextBox1, Label1, Button1, Button2, Label2, Label3, and Web1. At the bottom of the sidebar are 'Rename' and 'Delete' buttons.

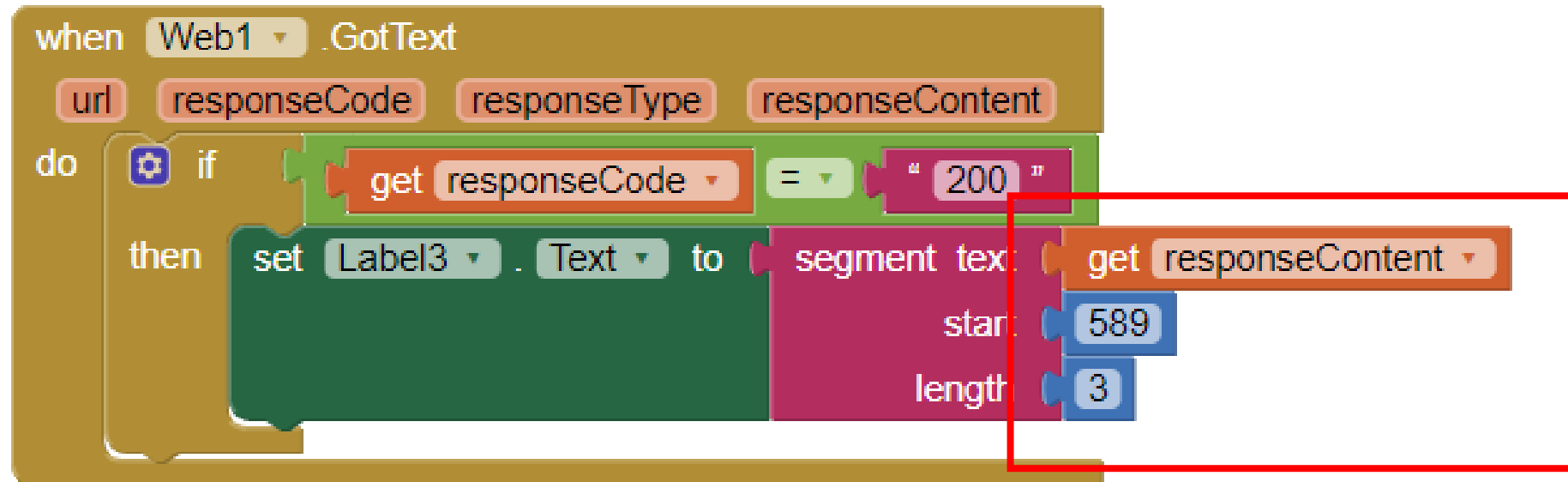
The main workspace displays two scripts. The top script is triggered by 'when Clock1.Timer' and contains blocks for 'do', 'set Web1.Url to', 'join', 'TextBox1.Text', and 'call Web1.Get'. The bottom script is triggered by 'when Web1.GotText' and contains blocks for 'url', 'responseCode', 'responseType', 'responseContent', 'do', 'if', 'get responseCode', '=', '200', 'then', 'set Label3.Text to', and 'segment text'. A red arrow points from the 'segment text' block in the bottom script to the 'segment text' block in the left sidebar, indicating it is being dragged.

The 'segment text' block has three input fields: 'start', 'length', and 'length'.

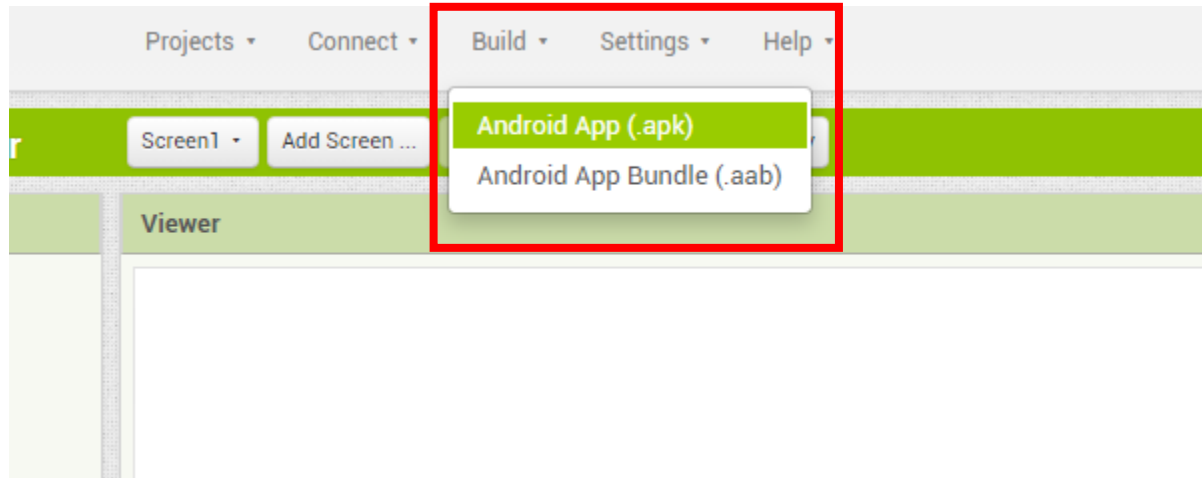
Drag the get block from Variables Tab, and number blocks from Math Tab.




Change the value as shown below.




Done! Now on your real mobile phone, uninstall your apps. Then download and install a new, updated one.



Android App for ESP8266_WebServer

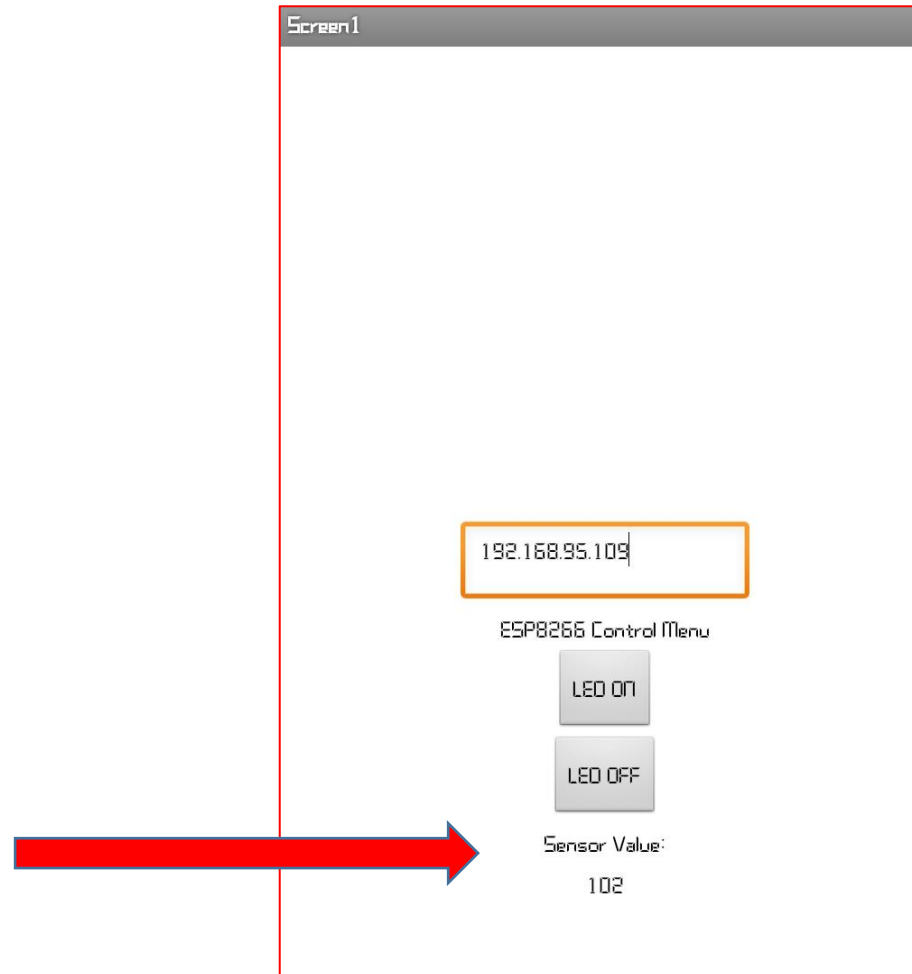

Download .apk now



Click the button to download the app, right-click on it to copy a download link, or scan the code with a barcode scanner to install.
Note: this link and barcode are only valid for 2 hours. See [the FAQ](#) for info on how to share your app with others.

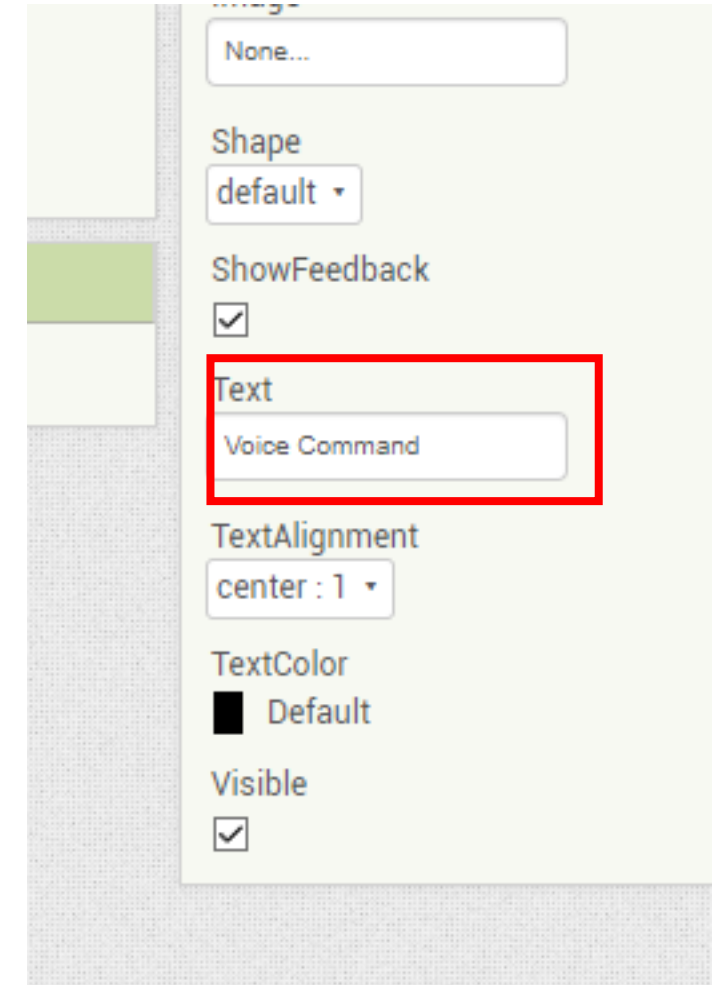
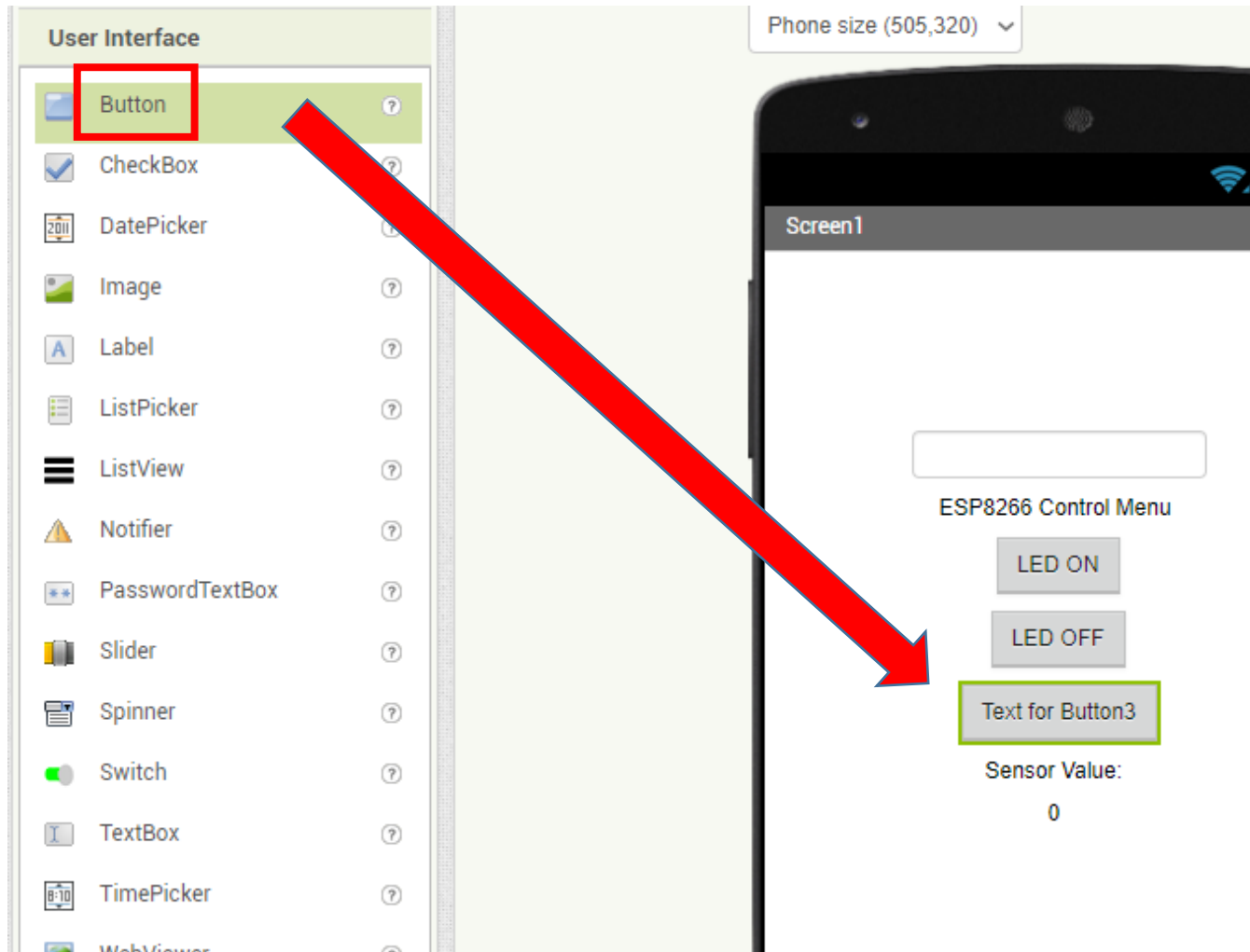
Dismiss

Open your apps. And you should see a new interface, a sensor reading in there. Play with your sensors, and watch the reading changes and update in real time through the internet! Dont forget to input your IP Address from before.

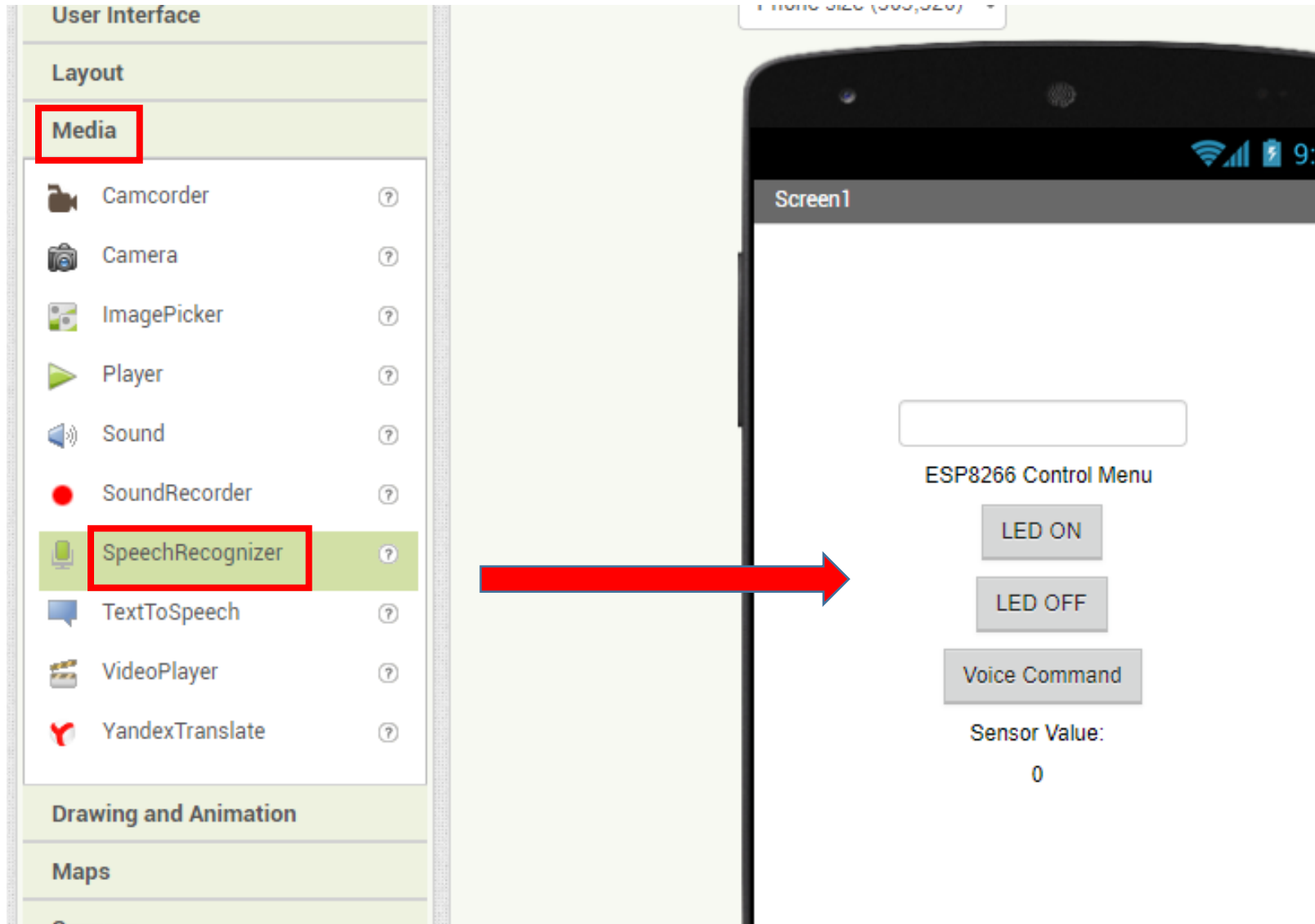


Adding Voice Recognition into your Project

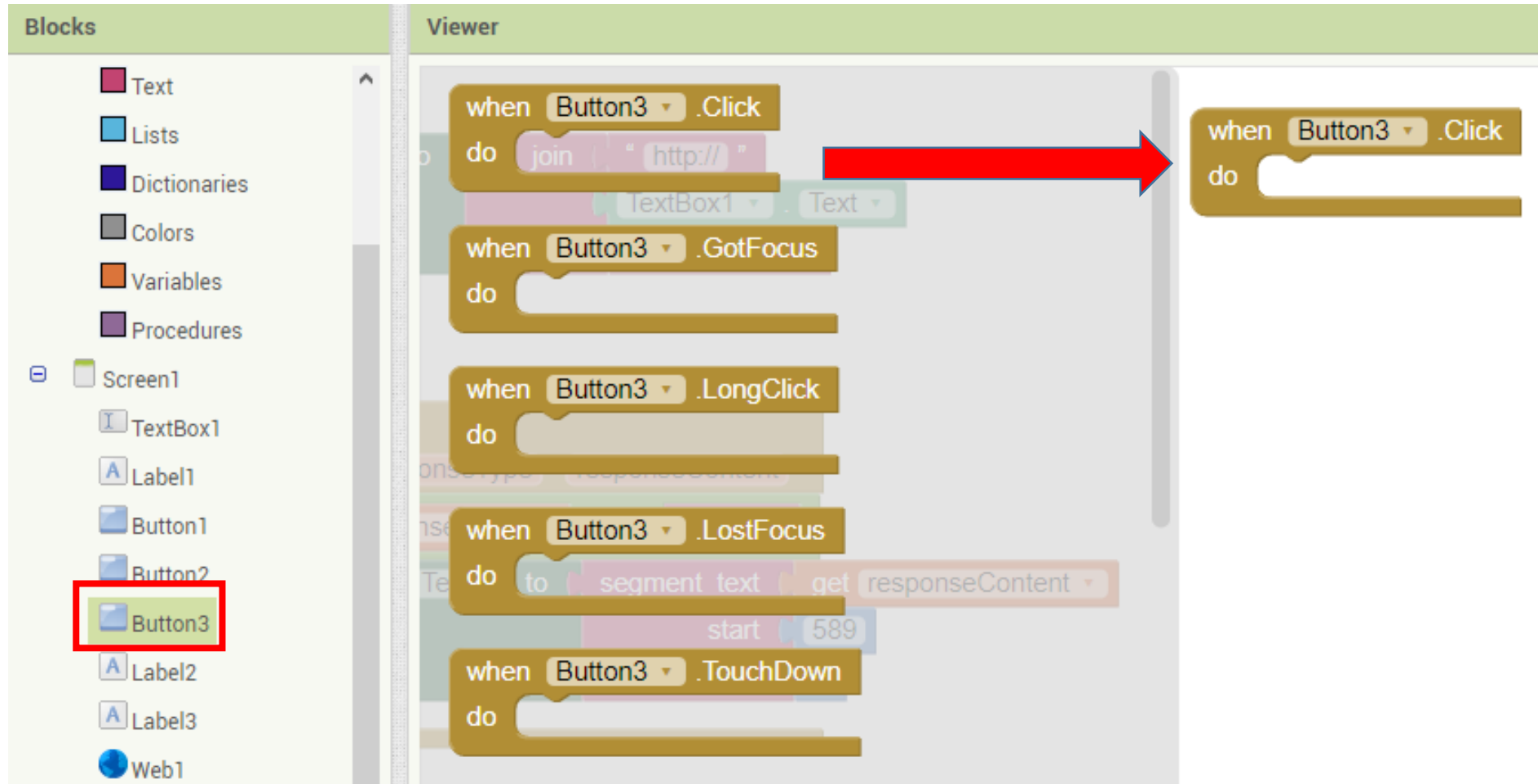
Lets get back on to Designer Tab on your MIT website. Drag a button into your screen. Change the text.



On Media Tab, drag SpeechRecognizer into your screen. Same as Web and Clock, nothing will show on your screen.



Lets continue to our Blocks Tab. Make sure you have some space. Click Button3, drag when..Click block into your code.



Click SpeechRecognizer1 Tab, and drag call..GetText into your code.

The screenshot displays a visual programming environment with two main panes: **Blocks** and **Viewer**.

Blocks Pane: A list of components is shown on the left. The **SpeechRecognizer1** component is highlighted with a red rectangle.

Viewer Pane: The right pane shows a sequence of blocks. A red arrow points from the **SpeechRecognizer1** component in the Blocks pane to a **call SpeechRecognizer1 .GetText** block in the Viewer pane, indicating it is being dragged into the code.

The Viewer pane contains the following blocks (from top to bottom):

- when SpeechRecognizer1 .AfterGettingText** (yellow block)
 - result partial** (orange block)
 - do** (purple block) containing **TextBox1 .Text** (light blue block)
- when SpeechRecognizer1 .BeforeGettingText** (yellow block)
 - do** (purple block)
- call SpeechRecognizer1 .GetText** (purple block) - This block is being dragged from the Blocks pane.
- call SpeechRecognizer1 .Stop** (purple block)
- SpeechRecognizer1 .Language** (green block)
- set SpeechRecognizer1 .Language to** (green block)
- SpeechRecognizer1 .Result** (green block)
- SpeechRecognizer1 .UseLegacy** (green block)
- set SpeechRecognizer1 .UseLegacy to** (green block)

On SpeechRecognizer1 Tab, and drag when..AfterGettingText block into your code.

The image shows the Visual Studio Code interface with the Blocks and Viewer panes. The Blocks pane on the left lists various components, with 'SpeechRecognizer1' highlighted in a red box. The Viewer pane on the right shows a sequence of blocks for the SpeechRecognizer1 component, including 'when SpeechRecognizer1 .AfterGettingText' and 'when SpeechRecognizer1 .BeforeGettingText' blocks. A red arrow points from the 'when SpeechRecognizer1 .AfterGettingText' block in the Viewer pane to a similar block in a separate, highlighted area on the right.

Blocks Pane:

- Text
- Lists
- Dictionaries
- Colors
- Variables
- Procedures
- Screen1
 - TextBox1
 - Label1
 - Button1
 - Button2
 - Button3
 - Label2
 - Label3
 - Web1
 - Clock1
 - SpeechRecognizer1**
- Any component

Viewer Pane:

```
when SpeechRecognizer1 .AfterGettingText
  result partial
do
  TextBox1 .Text

when SpeechRecognizer1 .BeforeGettingText
do

call SpeechRecognizer1 .GetText
responseType responseContent

call SpeechRecognizer1 .Stop

SpeechRecognizer1 .Language
start 589

set SpeechRecognizer1 .Language to

SpeechRecognizer1 .Result

SpeechRecognizer1 .UseLegacy

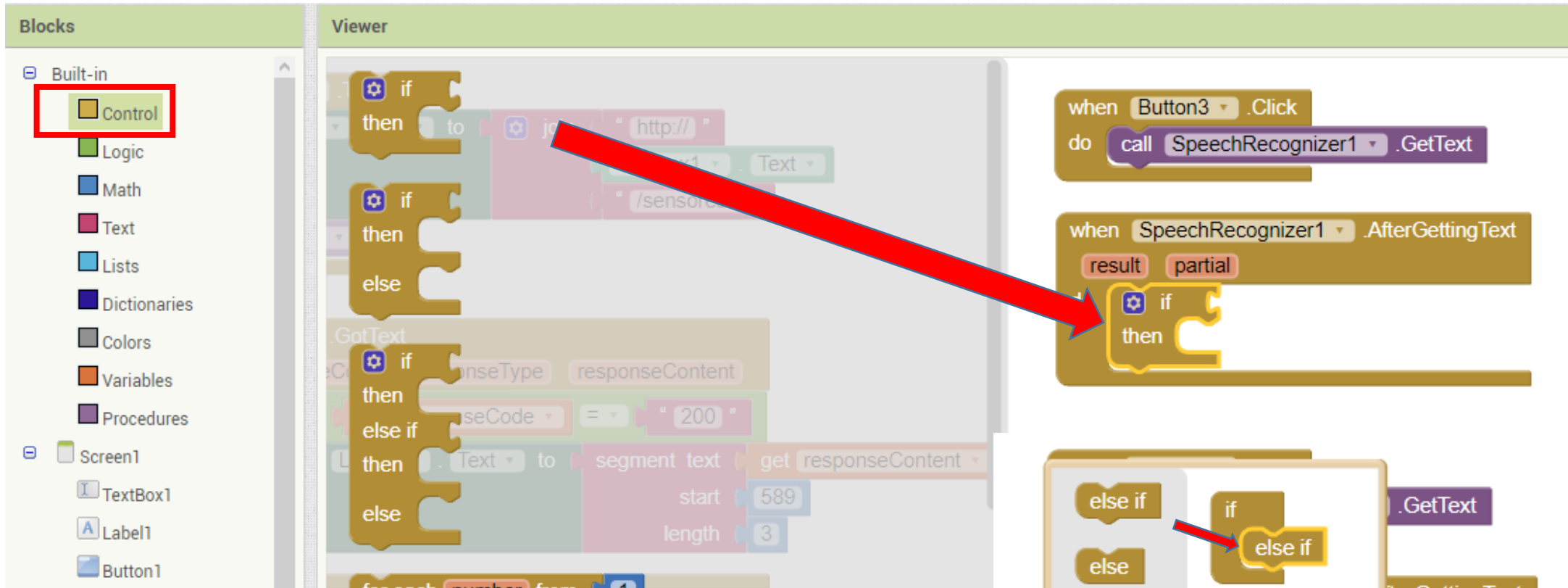
set SpeechRecognizer1 .UseLegacy to
```

Highlighted Block:

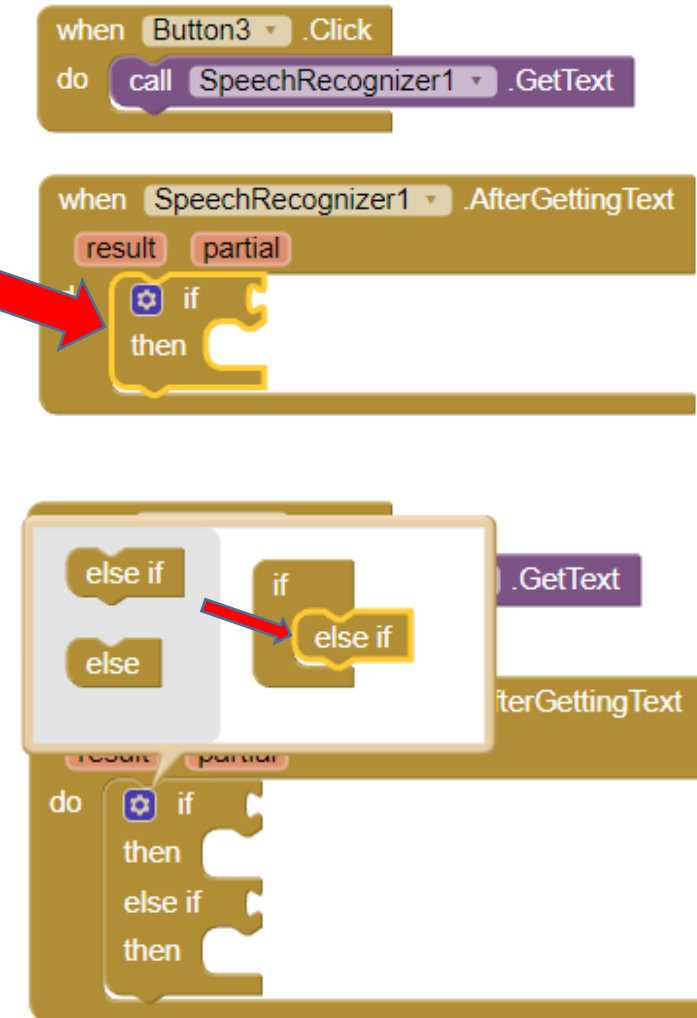
```
when Button3 .Click
do call SpeechRecognizer1 .GetText

when SpeechRecognizer1 .AfterGettingText
  result partial
do
```

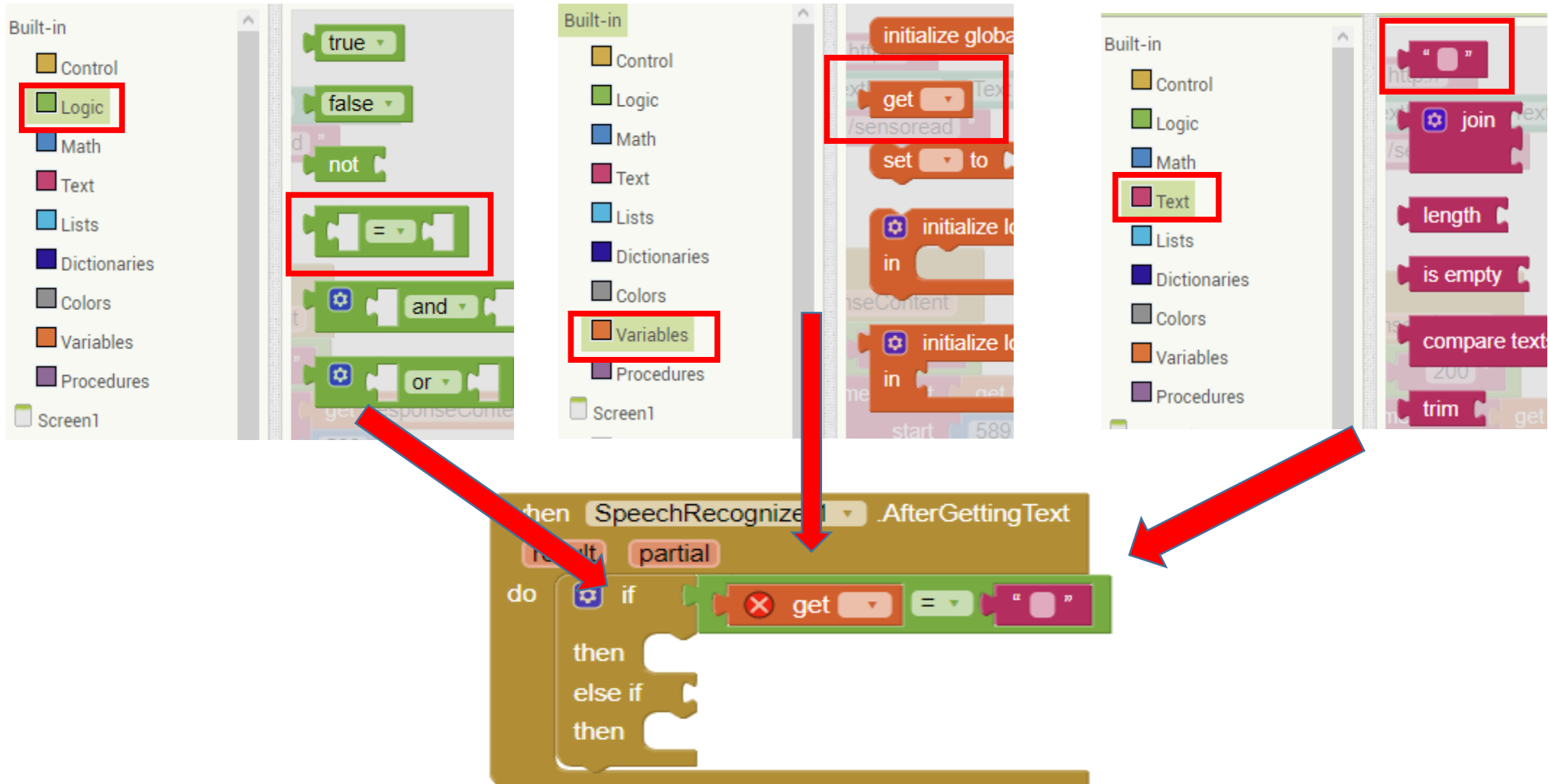
Click Control Tab, drag if..then block into your code.



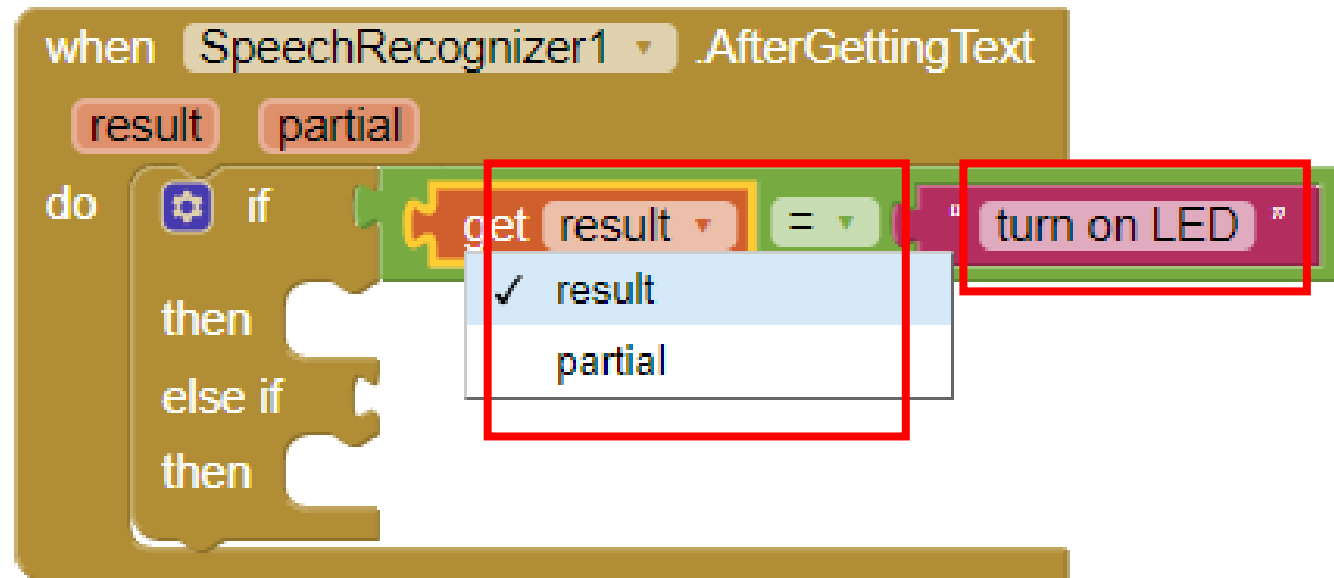
Click the gear icon on the if..then block. Add else if block into it.



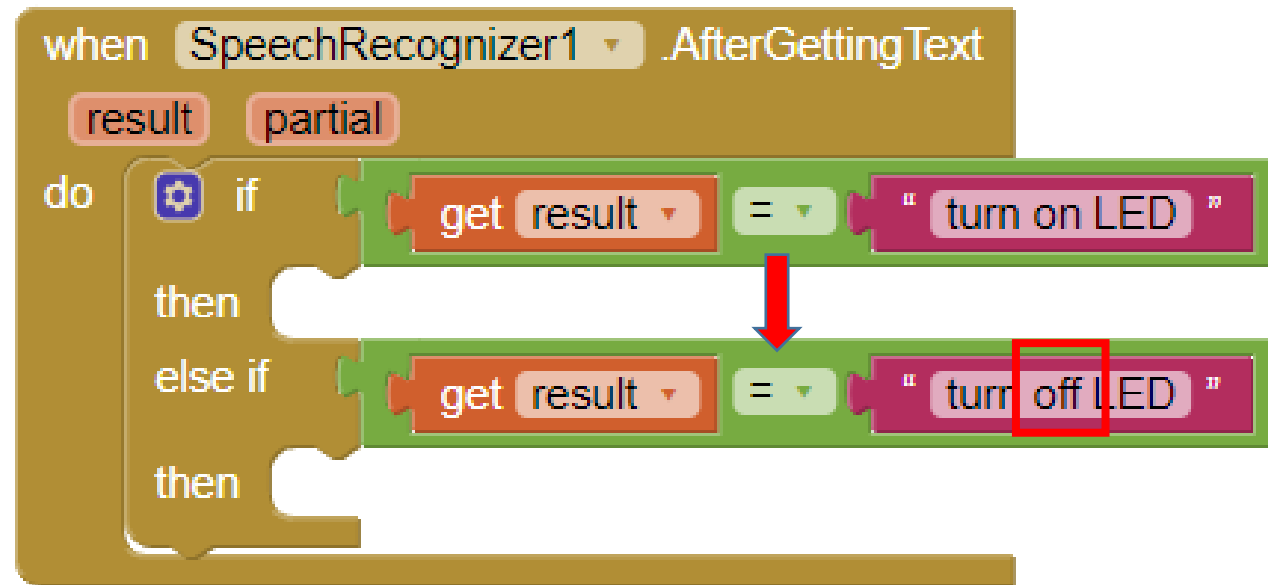
Get the = block from Logic Tab, get block from Variables, and empty text block from Text.



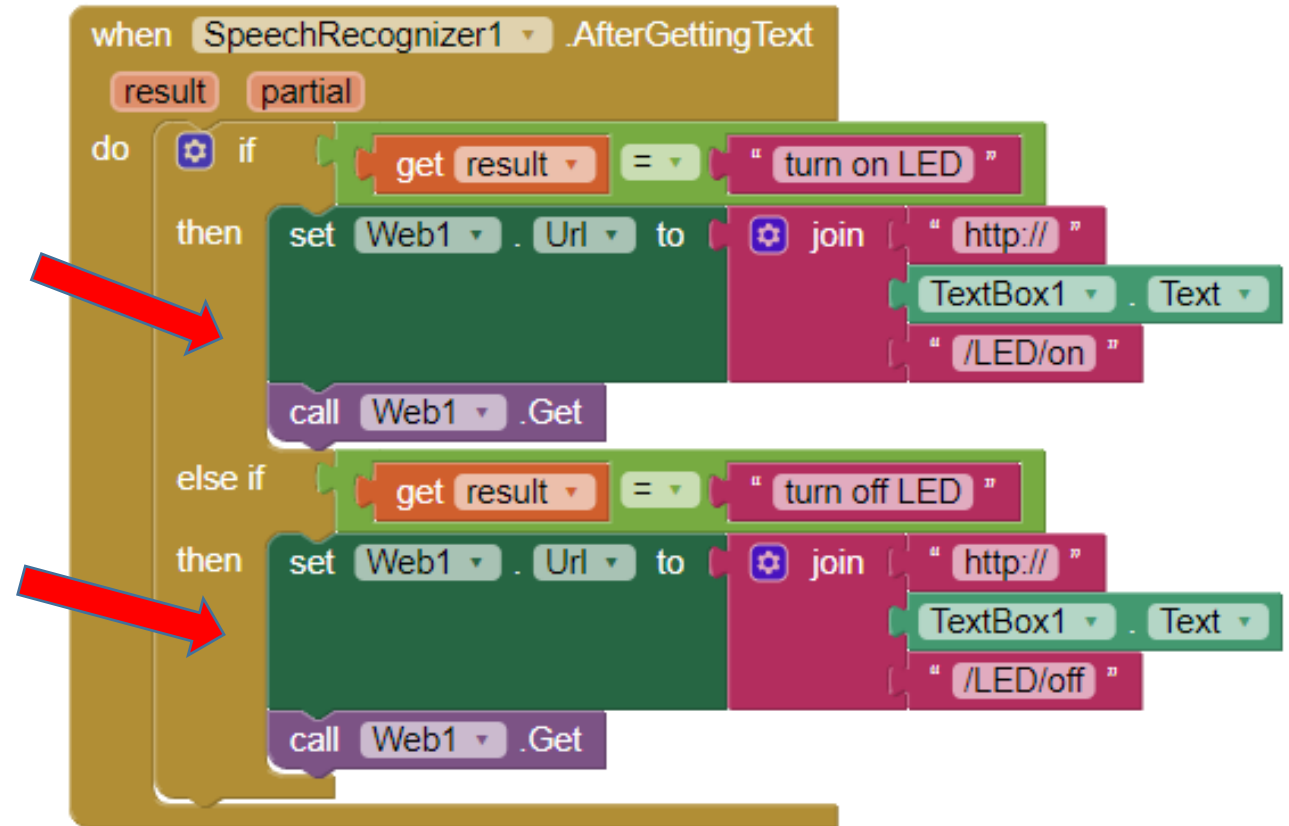
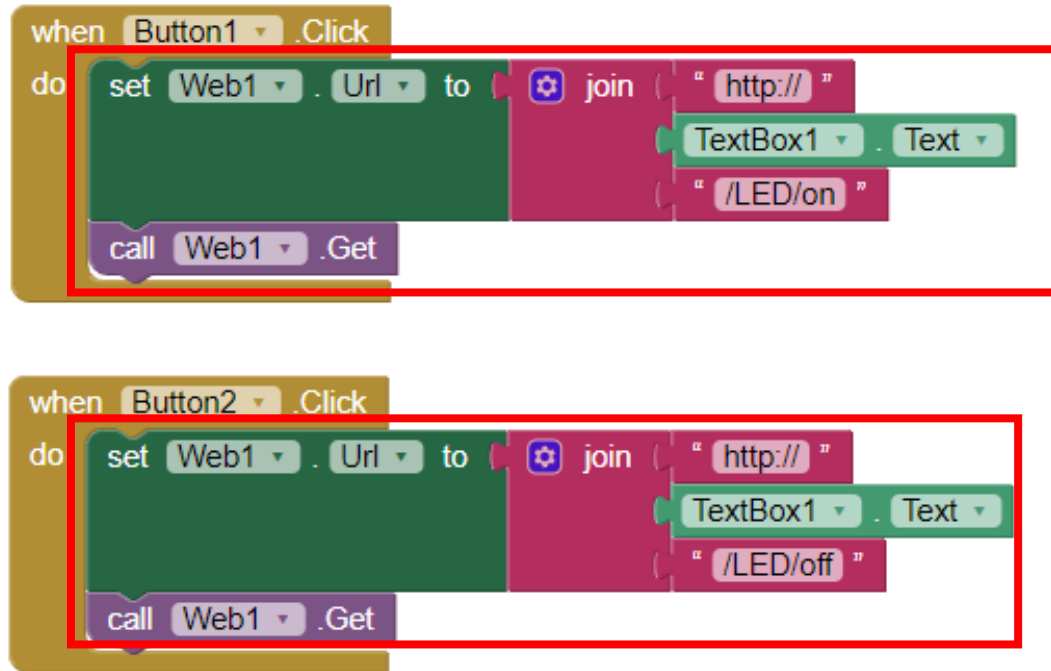
Choose result for the get block. And fill the text block with you desired command.
This is the voice command you will use for turning on the LED.



Duplicate the = block, and drag it to the else if position. Change the command. This command is for turning off the LED.

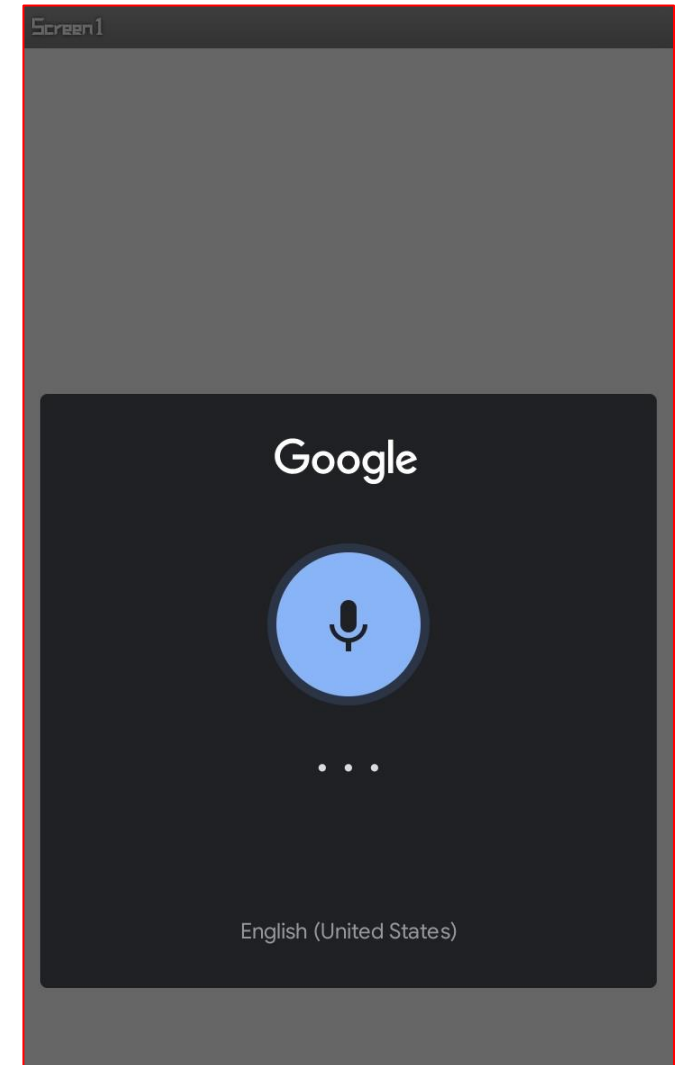
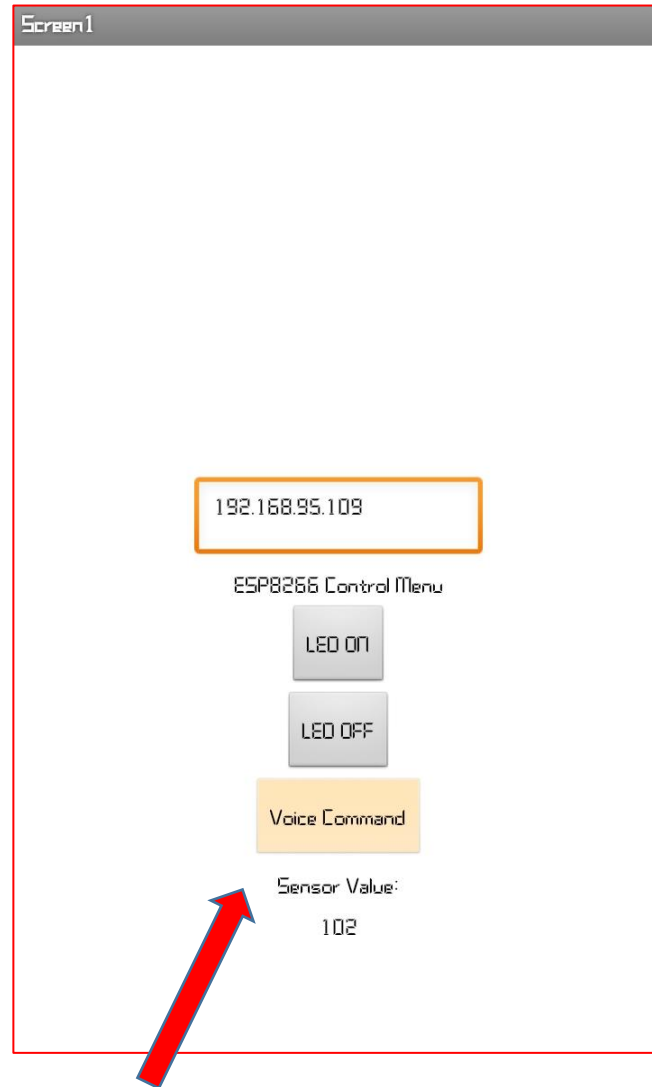
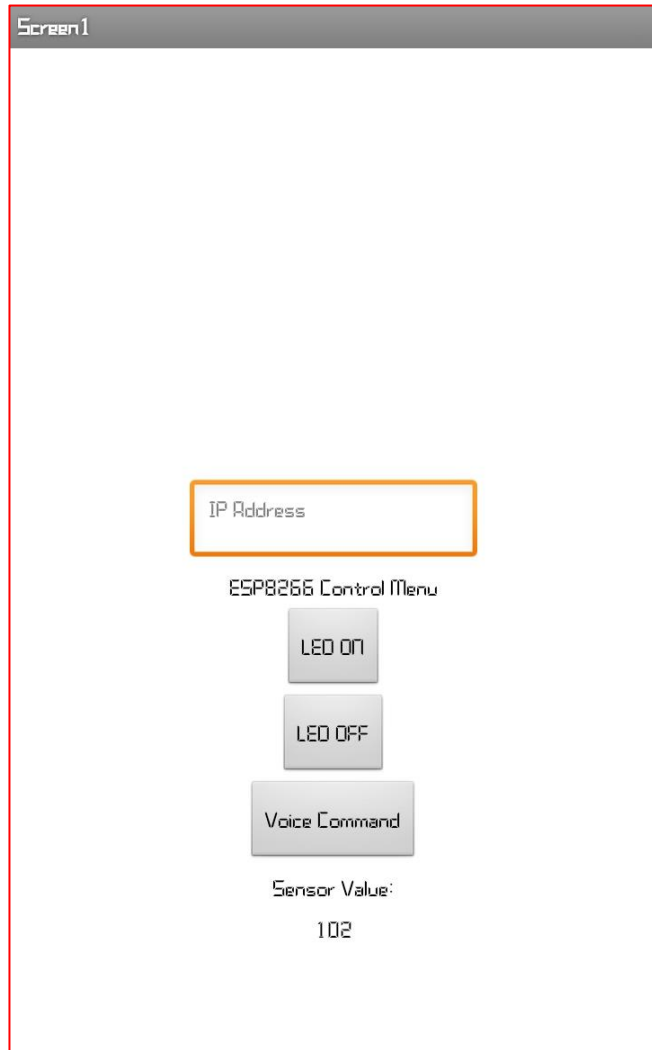


Remember our previous blocks for turning the LED on and off using buttons?
Duplicate those blocks into this block.

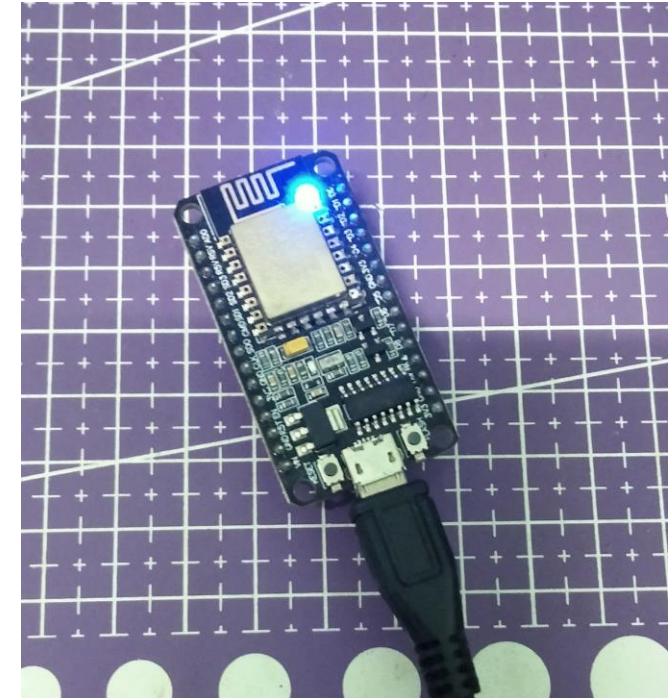
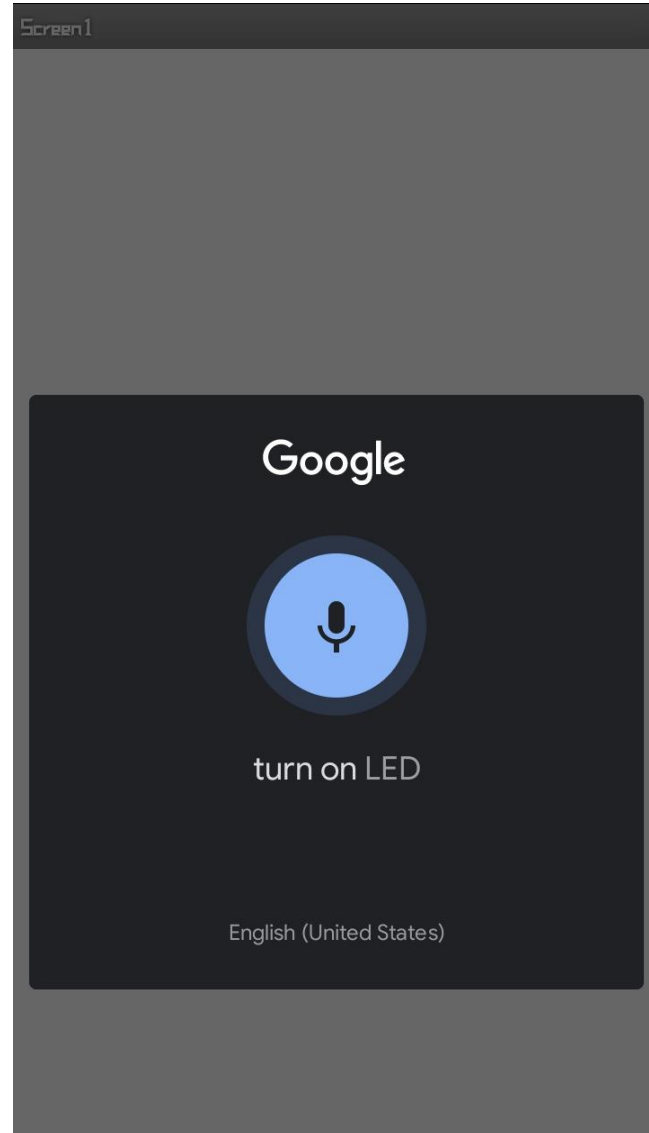
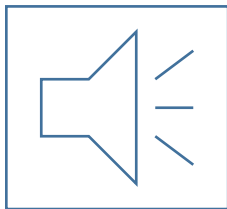


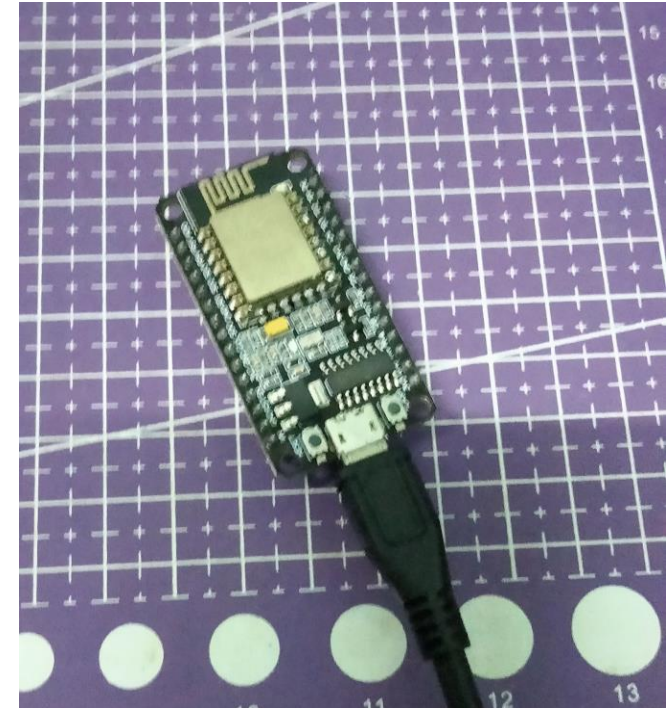
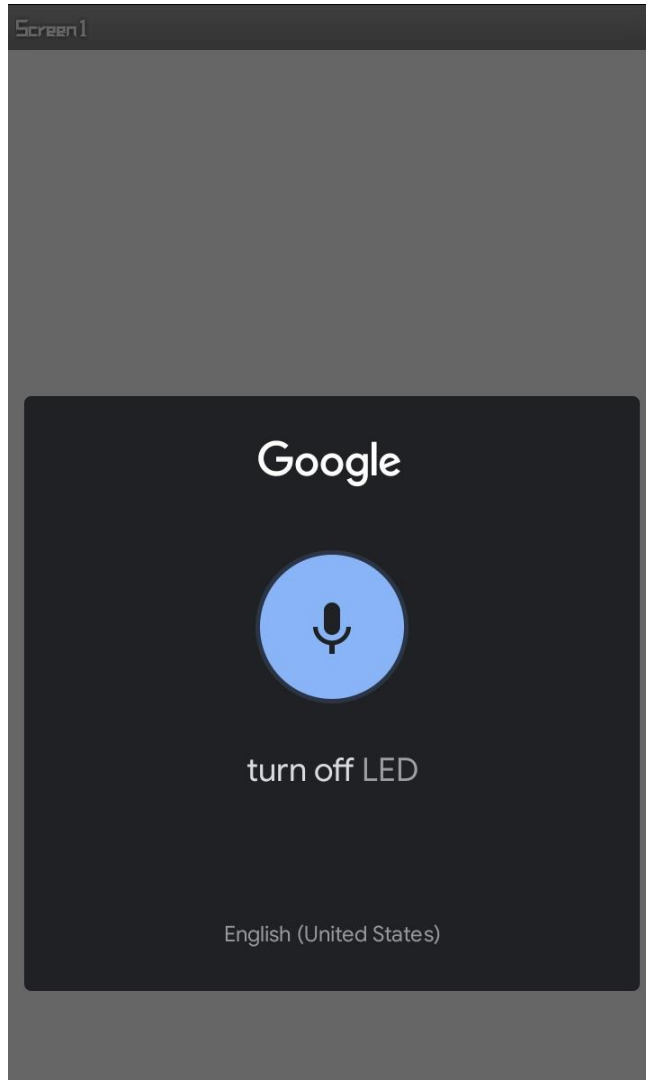
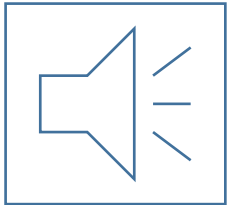
Make sure to **ONLY** duplicate the set..Url and call..get blocks only, **NOT** when..do block. Also make sure your blocks in **CORRECT** position, the /LED/on should be on your turn on voice command, and vice versa.

Now reinstall your apps. Open it and you should see a brand new Voice Command button. Insert your IP Address. Press the voice command button and use your voice control the LED.



Make sure you said exactly what you put in your apps coding. For this example, “turn on LED” and “turn off LED”.





Question and Answer



Conclusion

