Dashboard with Arduino

Tutorial 1: Hello Arduino!

GNG1103 F19

1. Objective

In this tutorial, you will learn how to set up a basic panel on Dashboard and use it to send and receive messages from an Arduino. This will be accomplished with the visual logic built into Dashboard as well as modifying an example in the Arduino IDE. This tutorial will expose students to basic HTTP communications, basic programming, and user interface design.

2. Equipment

- 1x Personal Computer
- 1x NodeMCU
    - Alternatively, you can use any esp8266 module with Arduino but these instructions may be slightly different
- 1x Micro USB cable

3. Downloads and Setup

1. You will need to download Ross Video's *Dashboard* software and install it on the computer that you will use to communicate with the Arduino. You can get this software from: https://www.rossvideo.com/support/software-downloads/dashboard.

2. Next, you will need to setup a mobile hotspot on your computer in order to connect the Arduino to that computer, and be able to communicate with it using its IP address. On Windows, do this as follows (see Figure 1):
    a) Right-click the Start menu icon
    b) Select "Network connections"
    c) Select "Mobile Hotspot"
    d) Now, click the 'Edit' button to set the network name, password, and Network Band. Most Wi-fi enabled Arduino's can only access the 2.4 GHz band. Once, you've set it up, turn the hotspot on by clicking the switch.

Note: This is not necessary if you have a personal private internet connection (for example, if you are at home, connecting to your own personal router), in this case, you can connect both the computer and the Arduino to this same personal network, rather than hosting a network from a PC.

3. You will also need to install the proper board libraries on Arduino. To do this:
    a) Go to File > Preferences and in to the area beside "Additional Boards Manager URLs" enter: http://arduino.esp8266.com/stable/package_esp8266com_index.json
    b) Go to Tools > Boards > Board Manager. In the manager search for "esp8266" and install the "esp8266 by ESP8266 Community" library.

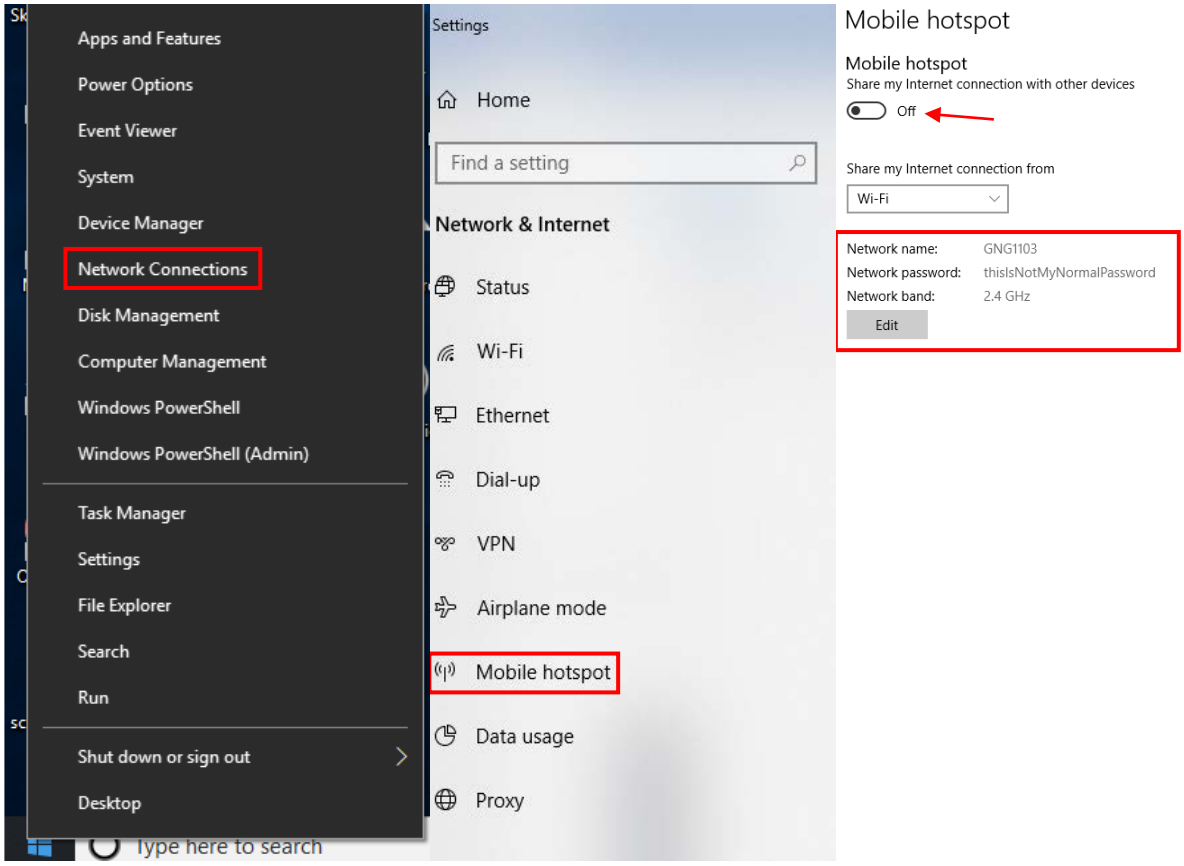


Figure 1: Hosting a Mobile Hotspot on a PC

4. Set up Your First Panel

Now let's start designing a panel on Dashboard. This is how you will create a user interface for your design and control whatever devices you have connected to Dashboard via your network. In this example, we will create a button to send a message to the Arduino and then get a response from the Arduino. We will set up text inputs to assign the Arduino's IP address, and finally provide a location to display a message from the Arduino back to the user.

1. Open Dashboard and create a new panel by using File > New > New CustomPanel File. Set the folder where you want to save your panel and give the file a name that makes sense. Keep the template as "Blank Self-Contained Data Source Panel (XPression)".

   Let's take a quick look at Dashboard (see Figure 2).

   1) This is the main design area where you'll build your panel.

   2) This is the debug information. This is where error messages, or messages that you send yourself will appear to help you while coding.

   3) These are the other panels in your folder.

   4) This is the *edit* button that you use to make/modify a panel. These are the main things to note, but there are other features to explore that may or may not be helpful for your project.

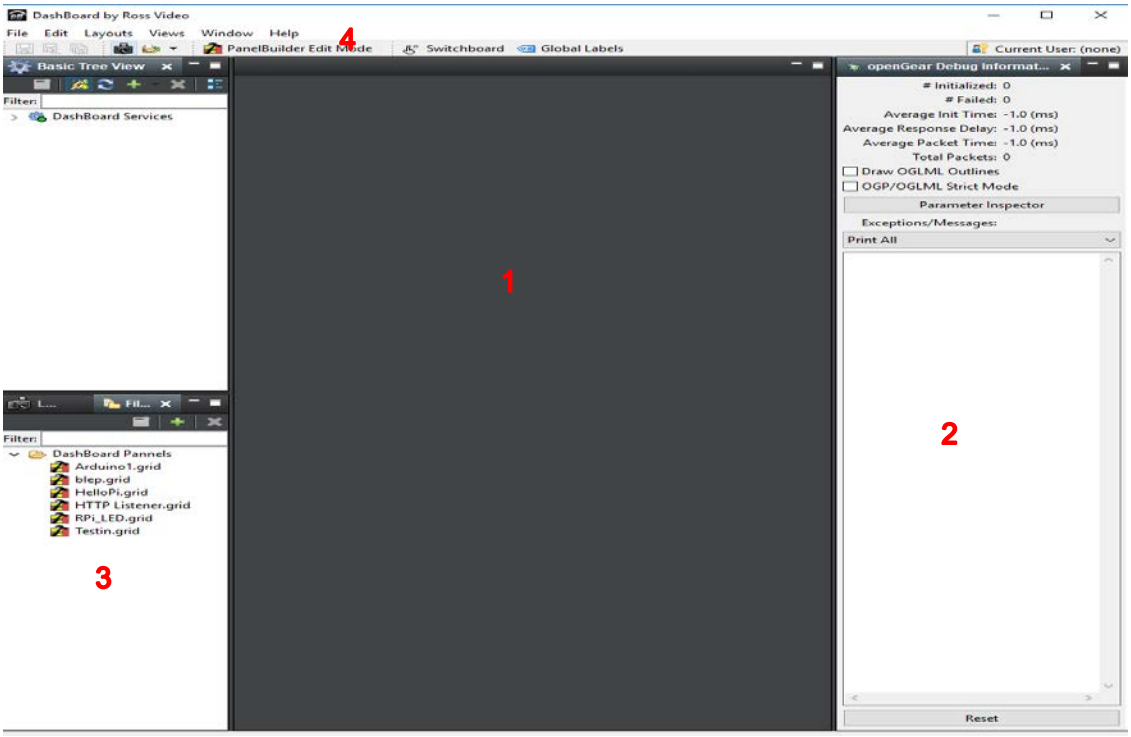


Figure 2: Dashboard Screen Layout

2. Click the PanelBuilder Edit Mode button to begin making the panel. Take a look at the Edit menu that comes up. Here we have options to add a number of different items such as: buttons, tables, labels, etc, as well as tools to move items around and resize them.

3. Click “Button” then click and drag in the panel area to create button.

4. In the menu that comes up, give your button a name, and an ID. The name is what will actually display on the button, and the ID is what we will use to refer to the button in the code. Here we have given the button the name “Click Here” and the ID has been set to “sendMessage”. We can also change the button type, but we will leave it as a push button

5. Next, click the “Param+” button and then the text entry button. This will create a space for the user to enter text, and will keep whatever they enter as a string that we can use in our code.

6. Once again, click and drag in the panel area to create the text entry space.

7. In the menu that comes up, select “Create New Parameter” and give it a name and an initial value. In this case, the name is what we will use in the code, and the initial value is what will initially appear in the text. We have called this parameter “*ardIP*”, and given it the initial value 255.255.255.255, because this is where we will enter the IP address of the Arduino.

8. Change the display type to “Text Entry” and press OK

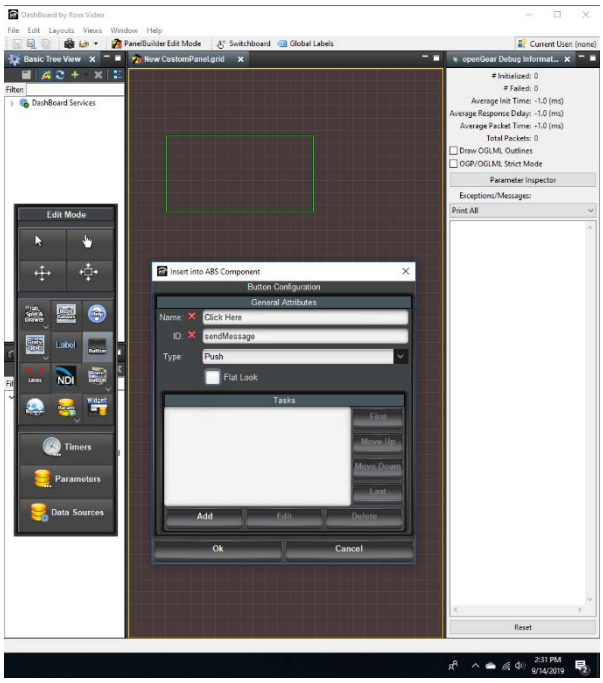


Figure 3: Button Configuration

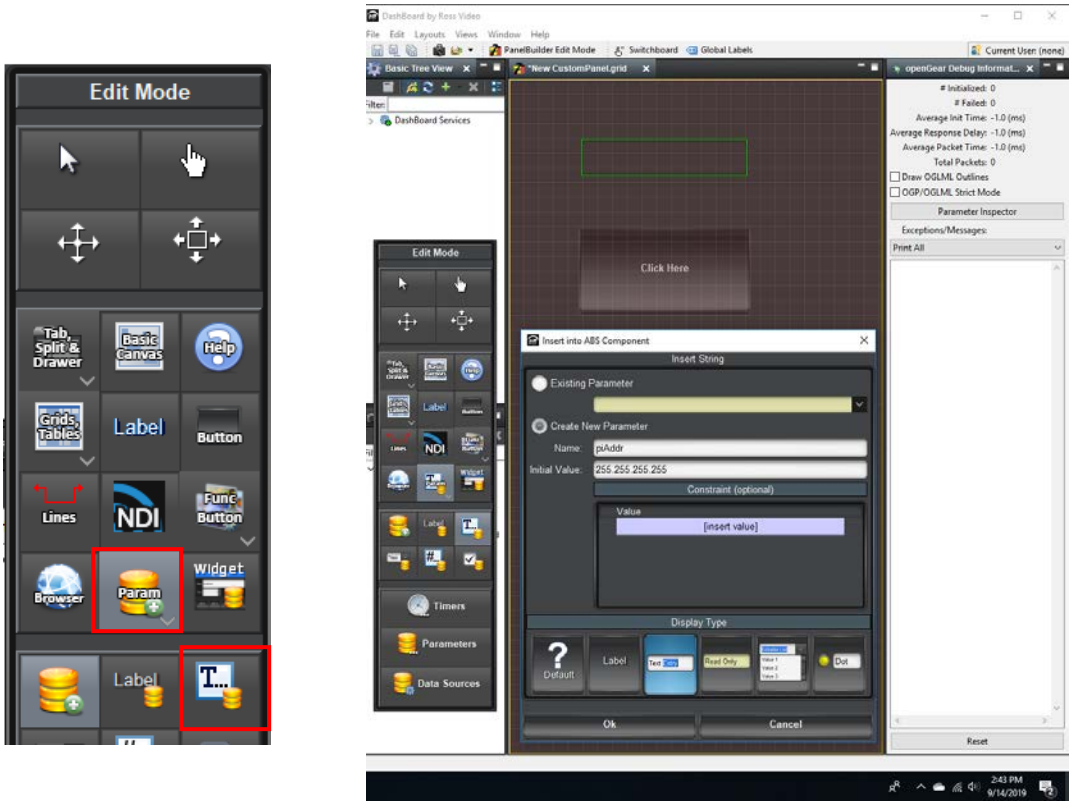


Figure 4: Creating an IP Address

9. Repeat steps 6-8, except this time change the display type to “Read Only”. This is where we will display the reply message sent from the Arduino, so call it something like “ardMessage” and give it some default initial value like “Awaiting Message”.

10. Now we have all the tools in place, but remember you are designing for users that aren’t necessarily you or your group. The user interface is an important consideration that could mean the difference between a good and a great design (for example a button labelled “Click here” probably isn’t a good user interface…). Let’s make our panel a little more user friendly by adding labels to all the text inputs, so people know what they are for. Use the “Label” tool (beside the button tool) to add labels. In this case, the name is what appears on the panel, and the ID is what we can use to modify the label in our code if we wanted.

11. Use the “Move Component” and “Resize Components” to make your panel look nice, maybe something like this:

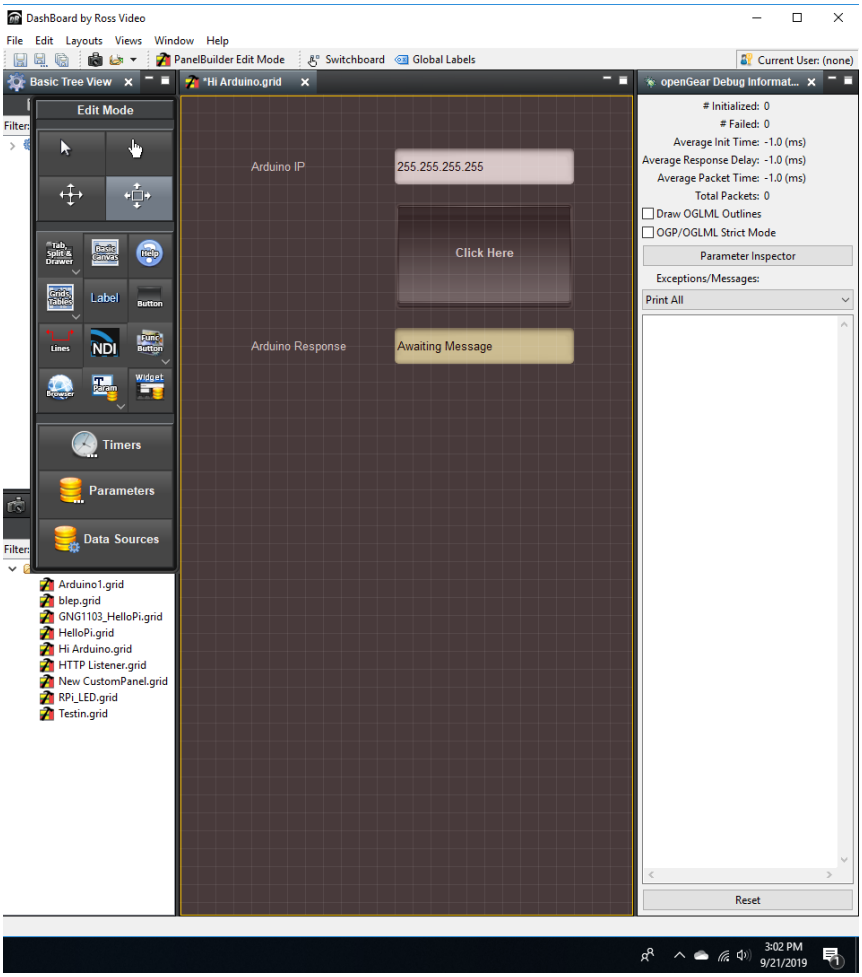


Figure 5: Moving and Resizing Components

12. To add functionality to our button, double click it to open its “Edit Component” window. In this window, you’ll see four Tabs at the top. “Button Attributes” is essentially the same menu you saw when we first created the button, so you can change the name or ID if you want. “Position/Stretch Attributes” allows us the finely tune the size and location of the button which

is helpful is we want to precisely align our components. “Style” allows us to change the aesthetics of the button by changing the background and the font. Finally, “Source” is the script that defines the button on the panel.

13. Most of you will want to program with the Visual Logic. To add a script, go to the “Button Attribute” tab and click “Add” in the “Tasks” window.

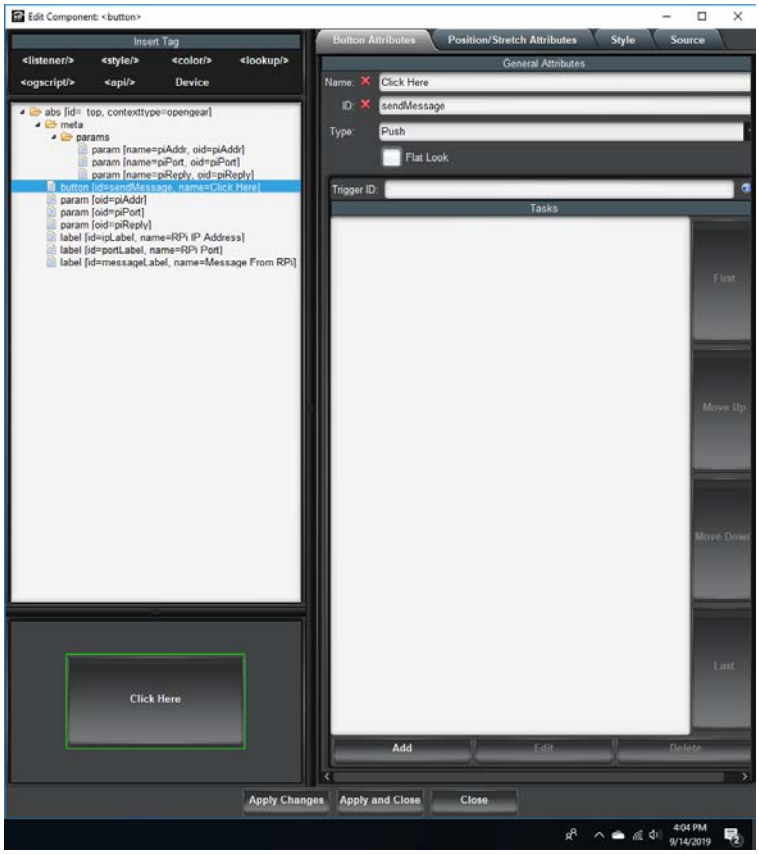


Figure 6: Adding a Script with Visual Logic

14. Now the scripting window will be up. On the right side, you’ll see “Control and API’s”, which is where all of the code blocks are.

15. Double click to select “ogScript” > “Communication”, then double click “HTTP ASync Message” and a block should appear.

16. Now select “Parameters” > “Get Parameter String Value” and add one of those blocks

17. Finally, under “Controls” > “Math” > “Add” double click to add an addition block.

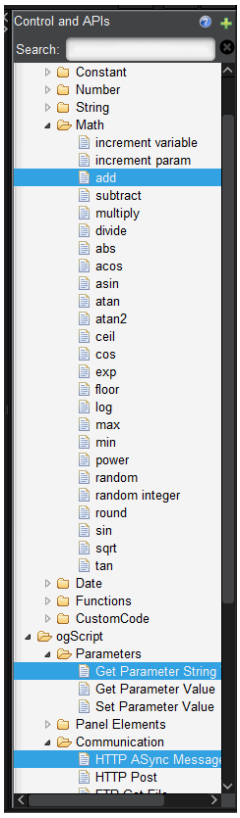


Figure 7: Get Parameter String Value

18. In your “Get Parameter String Value” block use the dropdown menu to select which parameter you want. In this case, we want to reference the IP address text entry. Double click through the folders until you find the parameter you made.

19. We can connect blocks together to have them do something. The **green nubs** are inputs and the **yellow nubs** are outputs.

20. In order to send a message to the HTTP server we need a URL of the following format

    http://**IPADRRESS/MESSAGE**

    So we want to replace the IP and the message with the particular values we want. To do this:

    a) write “http://” in the first part of the “Add” block.

    b) Attach the IP Parameter block to the next spot

    c) Finally, write “/MESSAGE” in the third spot on the “Add” block. In this case we will use “/HiArduino”

21. Attach the “Add” block to the URL port on the “ASync Message” block.

22. Change the method to “POST”

23. Finally, we need to give the command a “callback function”, this is what we can use to handle the response message from the Arduino. First delete the “null” that’s in there already. Then, to simply read the message and display it in the debugger input the following code:

    function callback(response) { ogscript.debug("The response is " + response);}

    ogscript.debug() prints whatever string we put in the function to the debugger so we can see what’s going on.

24. We also want to display the message on our panel since we set up a parameter to do so. To do this, add the highlighted code to the line you just wrote:

    function callback(response) {ogscript.debug("The response is " + response);
    params.setValue("ardMessage",0,response);}

    This function, params.setValue(), is used to change a parameter value. In this case we’re changing “ardMessage”, which was the name given to the “Read Only” parameter we made earlier, to “result”.

    Your code should now look like this:

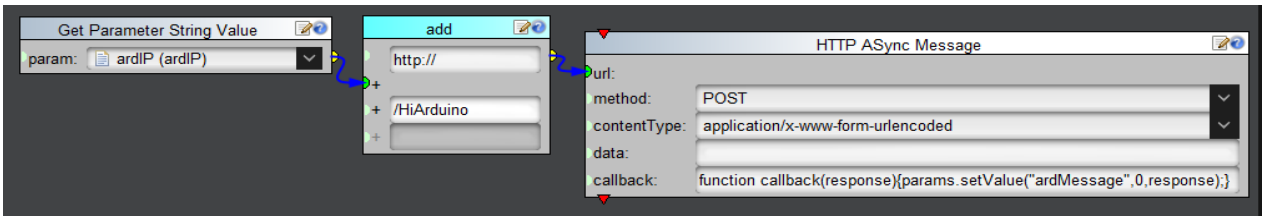


Figure 8: HTTP String Modification

25. Press “OK” at the bottom of the window.

26. Press “Apply and Close” at the bottom of the “Edit Component” window.

27. Now we have a functional panel. Press the “PanelBuilder Edit Mode” button again to exit the edit mode. Time to move over to the Arduino.

5. Code on Arduino

Now we need to write the Arduino side of things. This will be done by modifying an existing example in the new esp8266 libraries that were downloaded when we added the new boards.

1. Open up the IDE and navigate to File > Examples > ESP8266WebServer > HelloServer.

2. Delete anything to do with the on-board LED (the pinMode, and all the digitalWrite functions). Also, delete the contents of the “handleNotFound()” function.

3. Next, change the name of the “handleNotFound()” function to something less terrible, and modify the code to look like the image below. Here we have changed “handleNotFound” to “listenAndRespond”.

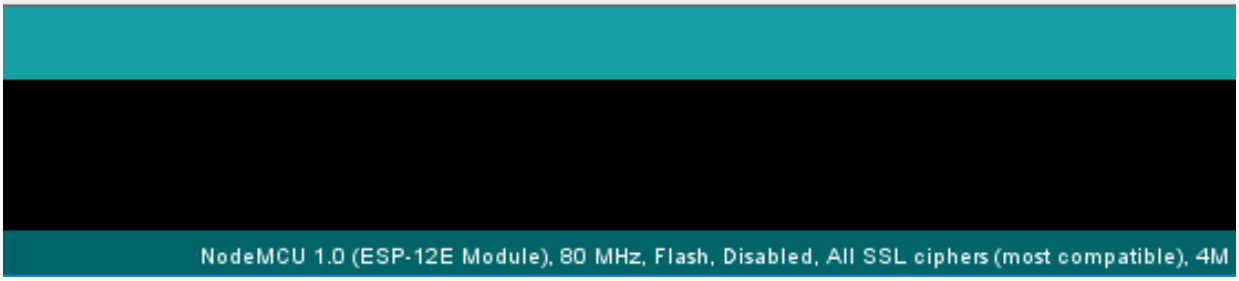
DashboardBasic | Arduino 1.8.9

File Edit Sketch Tools Help

DashboardBasic

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

#ifndef STASSID
#define STASSID "GNG1103"
#define STAPSK  "thisIsNotMyNormalPassword"
#endif

const char* ssid = STASSID;
const char* password = STAPSK;

ESP8266WebServer server(80);

void handleRoot() {
  server.send(200, "text/plain", "hello from esp8266!");
}

void listenAndRespond() {
  if(server.uri() == "/HiArduino"){
    server.send(200, "text/plain", "Hi Dashboard!");
  } else {
    server.send(200, "text/plain", "Wut");
  }

}

void setup(void) {

  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
```

```
  if (MDNS.begin("esp8266")) {
    Serial.println("MDNS responder started");
  }

  server.on("/", handleRoot);

  server.on("/inline", []() {
    server.send(200, "text/plain", "this works as well");
  });

  server.onNotFound(listenAndRespond);

  server.begin();
  Serial.println("HTTP server started");
}

void loop(void) {
  server.handleClient();
  MDNS.update();
}
```



4. Under Tools > Boards, select the appropriate board with WiFi module. In this case it's the NodeMCU 1.0 (ESP-12E Module).

5. Check your port

6. Upload the code to your board.

7. When the upload is completed, open up the Serial and wait for the connection message to show up.

6. Test Your System

Now, with both the Arduino and Dashboard running we can now test the system that we've just made.

1. With the Arduino connected to the mobile hotspot, you will see the Arduino in the list of connected devices

2. Copy the IP address into Dashboard.

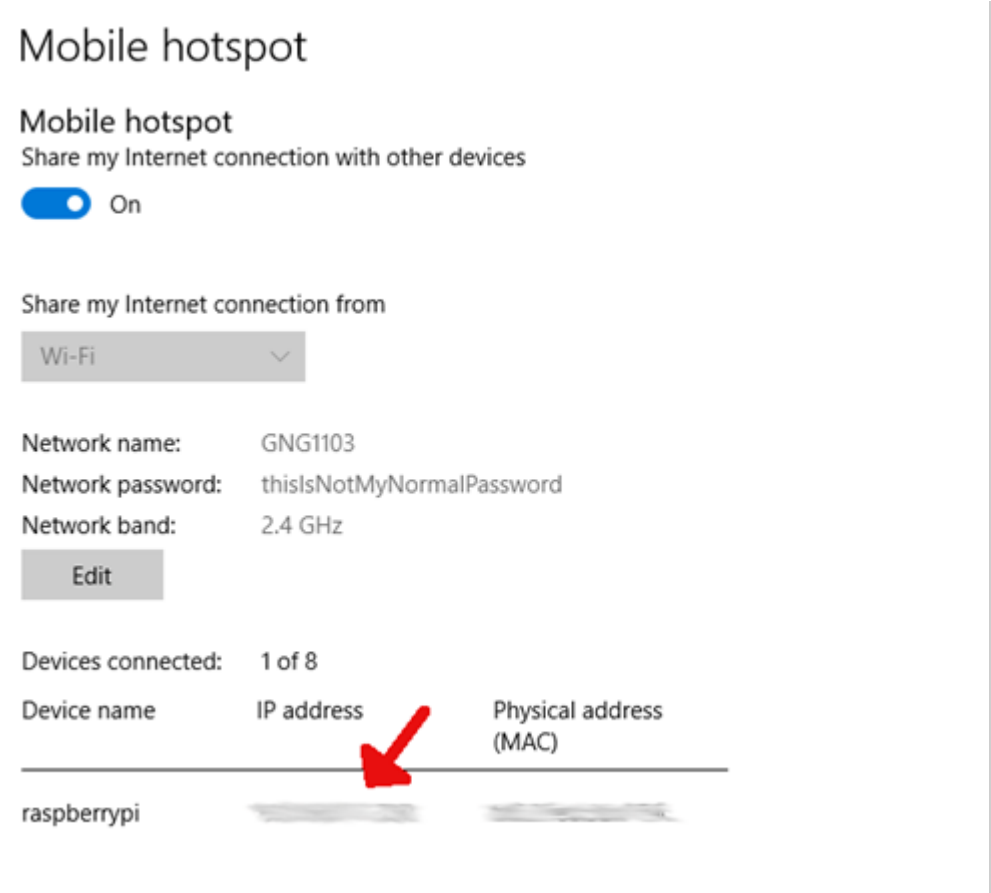


Figure 9: Assigning IP Address to a Physical Address (so it stays the same)

3. Now click the button you made in Dashboard. You should see the specified message in the read only area on your panel and in the debugger.

4. **Success!** Good luck! ☺.