Charith Perera and Nhat Pham (Eds.)

Internet of Things

Systems Design

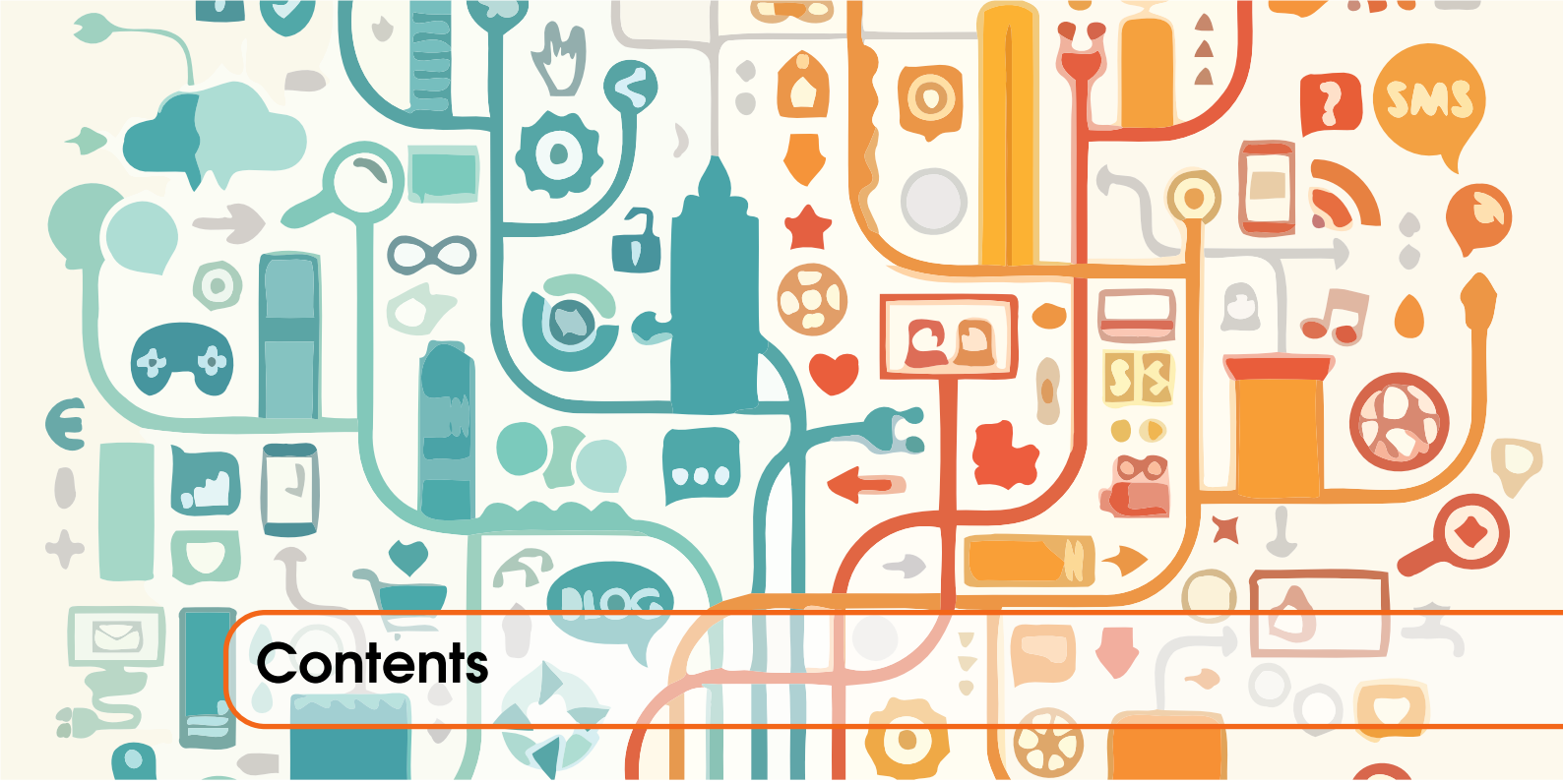
Lab Book with Packet Tracer

PUBLISHED BY CARDIFF IOT GROUP

How to cite this book
Charith Perera and Nhat Pham (Eds.), Internet of Things: Systems Design Lab Book with Packet Tracer, Cardiff IoT Group, 2025

Contributing Authors (alphabetical order)
Isra Ismail, Imran Khan

Version 1.0, (Latest) January 2025



Contents

Preface

This IOT LAB BOOK WITH PACKET TRACER is primarily compiled to support the university courses on *'Internet of Things: Systems Design'* at both undergraduate and postgraduate levels. It is also designed to complement the IOT LAB BOOK. The IOT LAB BOOK primarily focuses on end-to-end IoT systems development, combining microcontrollers, single-board computers, and IoT cloud platforms. This IOT LAB BOOK WITH PACKET TRACER aims to design IoT systems and bridge the gap between the virtual and physical environments.

One of the unique aspects of this LAB BOOK is its emphasis on integrating real-world hardware, such as Raspberry Pis, Arduinos, and microcontrollers, with the simulated environment of Cisco Packet Tracer. While working with physical hardware provides invaluable hands-on experience, it often comes with limitations. These include restricted access to specific types of sensors, actuators, or the number of devices available at a given time. Cisco Packet Tracer addresses these limitations by enabling students to create virtual counterparts of these devices—digital representations of sensors, actuators, and devices—and integrate them seamlessly with physical setups. This integration is achieved by allowing real hardware to communicate with virtual environments through protocols like HTTP, enabling bidirectional data and command flow between the two worlds.

For example, students can develop part of their project using physical hardware, such as Arduinos, and communicate with the virtual environment in Cisco Packet Tracer through HTTP servers or other communication protocols. This allows for sending commands and data back and forth between the physical and virtual worlds. By combining these two realms, students learn to leverage the strengths of both environments, building more robust and scalable IoT solutions.

Cisco Packet Tracer's flexibility extends further, allowing users to create new types of virtual sensors and actuators that may not be accessible due to hardware constraints. For instance, students might design a virtual temperature sensor that interacts with real-world data or create a virtual robotic arm actuator to simulate complex tasks not feasible with limited hardware. These examples help students develop problem-solving skills by exploring practical applications of IoT systems and foster their ability to design comprehensive and innovative solutions. It also offers extensive simulation capabilities that enable students to model and test complex scenarios that would be challenging to replicate in real-world conditions. This lab book bridges the gap between virtual and physical worlds through a series of targeted exercises. These are designed to teach students how to connect these domains effectively, fostering innovative and comprehensive IoT system designs.

It is important to note that ADVANCED LAB BOOK is primarily focused on teaching Cisco Packet Tracer as a standalone tool for learning the Internet of Things, providing a detailed introduction to its features and capabilities. In contrast, this IOT LAB BOOK WITH PACKET TRACER emphasizes integrating physical hardware with virtual environments. By bridging these two domains, this book equips students with the practical experience and conceptual tools necessary to create hybrid IoT systems. Together, the two books complement each other, enabling students to both master the technical aspects of Cisco Packet Tracer and explore innovative ways to combine real and virtual IoT components for comprehensive system designs.

— Further information, links and references. Throughout this lab book, we offer explanations, learning tips, external links, and references to relevant reading materials. If you find certain programming tasks difficult, you are encouraged to explore these resources for additional guidance. Although the suggested readings are optional, they provide valuable opportunities to deepen your understanding of IoT beyond the scope of the provided labs. Finally, please note that this is not a programming course; you are responsible for identifying and addressing any knowledge gaps by consulting the links and materials we have included.

Accessing the Code Repository

All the Packet Tracer (`.pkt`) files and other resources required to complete the labs in this *IOT LAB BOOK WITH PACKET TRACER* can be found in the following GitLab repository:

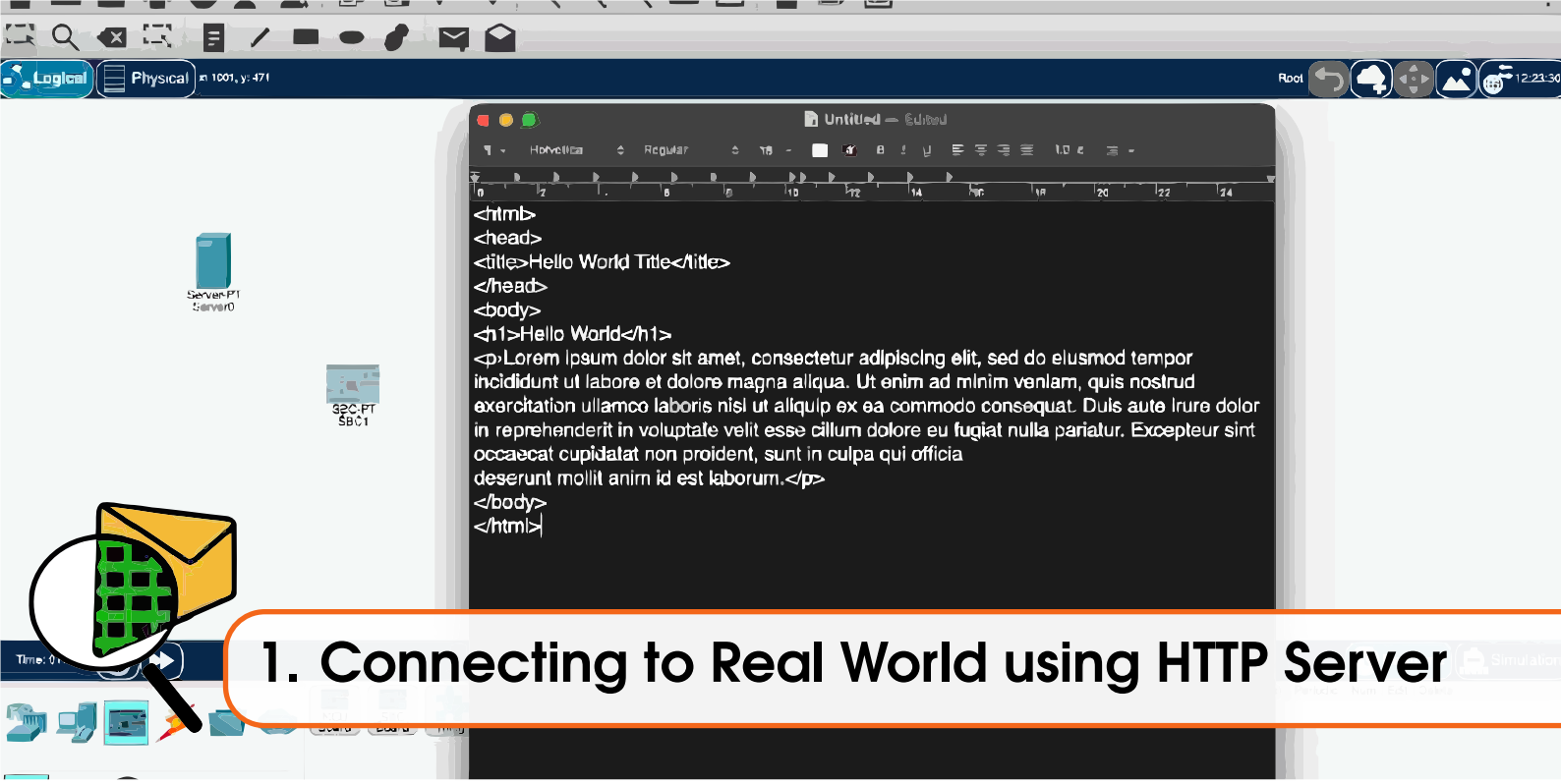
`https://gitlab.com/IOTGarage/iot-lab-book-with-packet-tracer`

This repository includes all lab files, configuration scripts, and supplemental materials referenced throughout the chapters. Whenever you work on a lab exercise, refer to the respective folder or file in the repository to locate the exact examples. We will also post updates, bug fixes, or enhancements here, so please check back periodically for the most recent versions. By visiting the repository, you can:

- **Clone or Download the Files:** Acquire all the relevant `.pkt`lab files, along with any scripts or configurations you need for each lab activity.
- **Review Commit History:** Track changes across different versions and branches, noting any improvements or experimental features.
- **Submit Issues:** If you encounter a bug, need clarification, or have suggestions, open an issue in the repository and collaborate with others to enhance the lab exercises.

Learn More about Cisco Packet Tracer:
If you would like to download Cisco Packet Tracer or learn more about its features, visit:

`https://www.netacad.com/cisco-packet-tracer`



1. Connecting to Real World using HTTP Server

Introduction

In this lab, you will learn how to **configure a JavaScript-based HTTP server** within Cisco Packet Tracer—first on a Single Board Computer (SBC) and then on a standard Server device. You will create routes, embed HTML, and verify responses via your local (or Packet Tracer's built-in) web browser. By the end, you'll see how Packet Tracer can simulate real HTTP connections for simple web hosting, providing a solid foundation for more advanced IoT or networking labs.

Objectives

- **Build a basic topology** in Cisco Packet Tracer (SBC, Server).
- **Create and run** a JavaScript HTTP server on the SBC.
- **Modify** port numbers, routes, and responses (including multi-line HTML).
- **Validate the server** by accessing it through a browser at `127.0.0.1`.
- **Transfer the same concept** to the Server device, using a different port and route.

Lab Plan

A. **Launch Packet Tracer and Observe the Initial Workspace**
B. **Build the Topology**
C. **Programming SBC Board (Creating the Program)**
D. **Programming SBC Board (Modifying the Program)**
E. **Programming SBC Board (Testing the Modification)**
F. **Programming SBC Board (Creating Another Directory)**
G. **Programming SBC Board (Adding to Your Web Page)**
H. **Programming on the Server**

Required Software

- **Cisco Packet Tracer 8.x** (or newer)
- Basic knowledge of navigating the *Programming* tab in Packet Tracer

- Optional: external web browser for testing (`http://127.0.0.1:{port}`)

A. Launch Packet Tracer and Observe the Initial Workspace

1. **Open Packet Tracer:**
   Double-click the Packet Tracer icon on your computer. You will see a blank Logical topology workspace.

A. Launch Packet Tracer and Observe the Initial Workspace

(a) **Open Packet Tracer:**
Double-click the **Packet Tracer** icon on your computer to launch the program. When it starts, you'll see a blank *Logical* topology workspace, as shown in Figure 1.1.

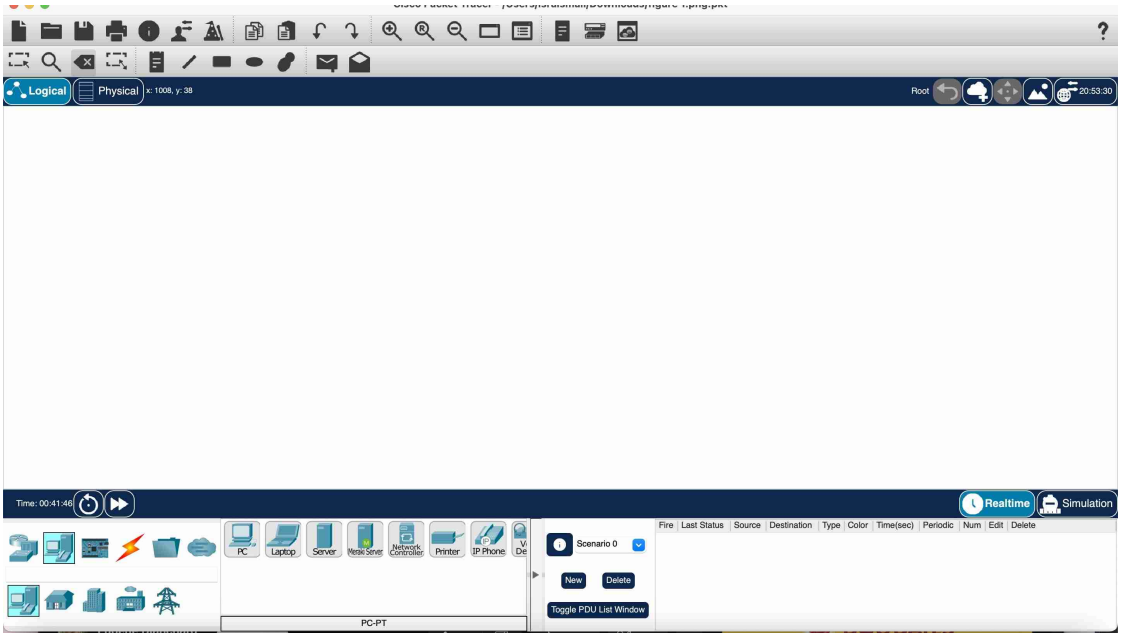


Figure 1.1: Initial workspace with no devices added yet.

B. Build the Topology

2. **Practice the Device-Type Selection Box:**
   In the lower-left corner of Packet Tracer, you'll see two rows of icons:
   - The *top row* shows **device categories** (e.g., *End Devices*, *Components*, *Network Devices*).
   - The *bottom row* shows **subcategories** (e.g., *Servers*, *PCs*, *SBC Boards*).

   Hover your mouse slowly over each category/subcategory to reveal helpful tooltips. This step familiarizes you with the **wide array of devices** you can use to build network scenarios.

Figure 1.2: Using the Device-Type Selection box to pick devices.

Why do this? Understanding how to navigate these categories is essential for quickly finding the right devices (like routers, switches, IoT boards, or servers). It also ensures you're comfortable adding, removing, or replacing components in more complex labs. ■

3. **Add a Server:**

   - In the top row, click *End Devices*.
   - In the bottom row, look for and select **Server**.
   - Move your cursor over the main workspace and click to place the server.

   After a brief moment, you should see the server icon appear with default settings (e.g., *Server0*).

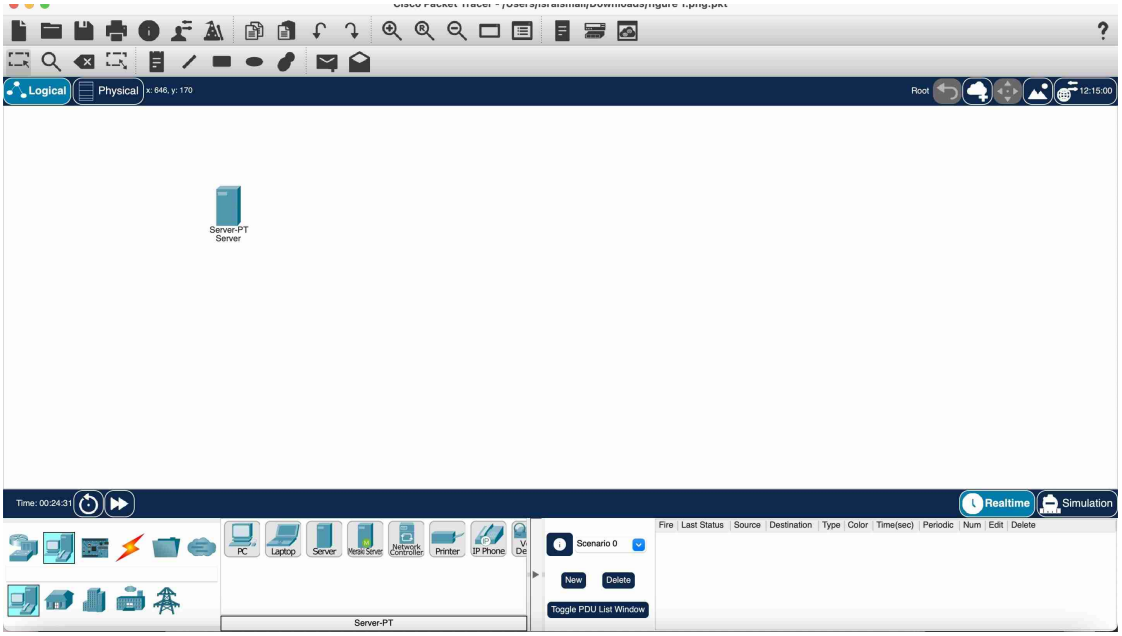


Figure 1.3: Server device added to the empty workspace.

Why a Server? A **Server** in Packet Tracer can host a variety of services (HTTP, DHCP, DNS, etc.). Having one in your topology allows you to simulate real-world network scenarios such as serving web pages or allocating IP addresses. ■

4. **Add an SBC Board:**

   - Still in the top row, click *Components*.
   - In the bottom row, select **SBC** (Single-Board Computer).
   - Click inside the workspace to place the SBC.

   You will now see both the *Server* and *SBC board* in the workspace.

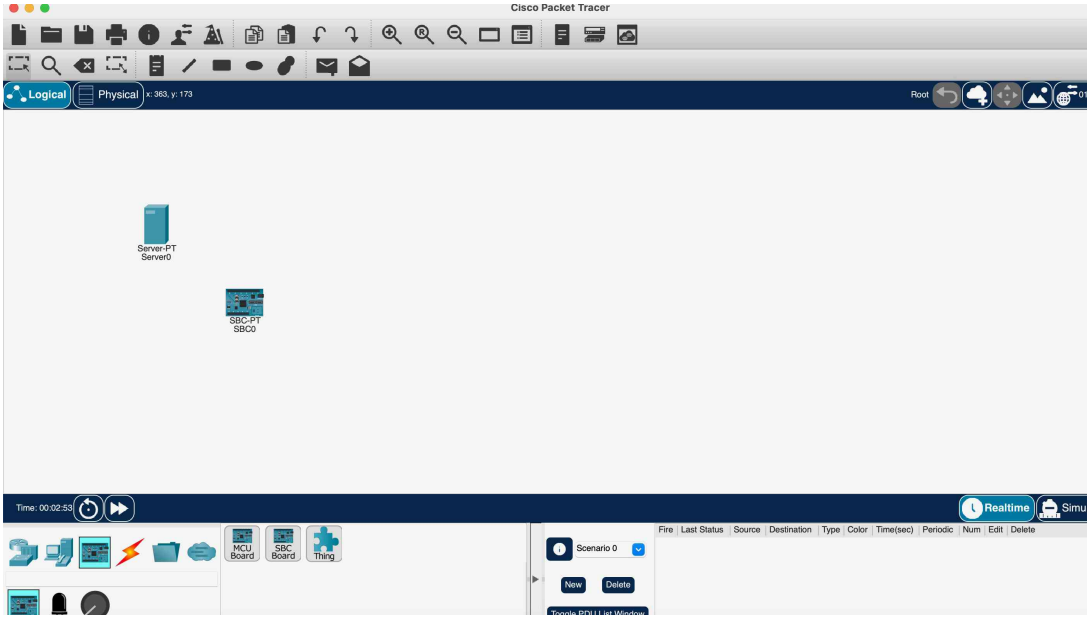


Figure 1.4: Topology now includes a Server and an SBC board.

Why an SBC Board? A **Single-Board Computer (SBC)** can represent a small device or microcontroller in IoT scenarios. This allows you to explore and experiment with networking, programming, and IoT features (e.g., sensors, actuators) within Packet Tracer. ■

C. Programming SBC Board (Creating the Program)

5. **Open the SBC Board Configuration:**
   Click on the **SBC Board** to open its configuration window. Then select the *Programming* tab. You will see a list of existing scripts on the left, and an editor or console area on the right.

   — Why do this?. Accessing the *Programming* tab is the starting point for any custom script you wish to run on the SBC. By default, an SBC may include sample scripts (like *Blink*), which you can edit or remove.

6. **Delete the File Blink (Python):**
   In the left panel, highlight the script named "`Blink (Python)`" and remove it by clicking the *Delete* or *Remove* button.

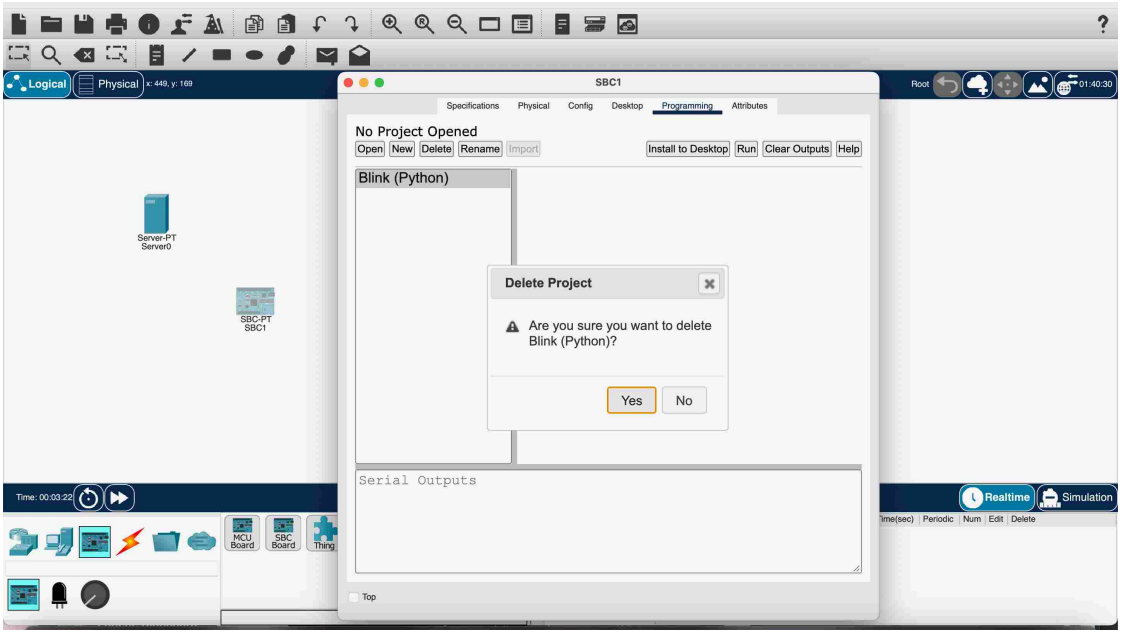


Figure 1.5: Removing the default Blink script from the SBC.

— Why remove Blink?. The `Blink` script is a simple default demonstration. Removing it declutters the interface, so you can start fresh with your own **HTTP server** script.

7. **Create a New HTTP Server Script:**
   Click the **New** button on the left, name the script *HTTP Server*, and select the template `Real HTTP Server - JavaScript` from the dropdown list.

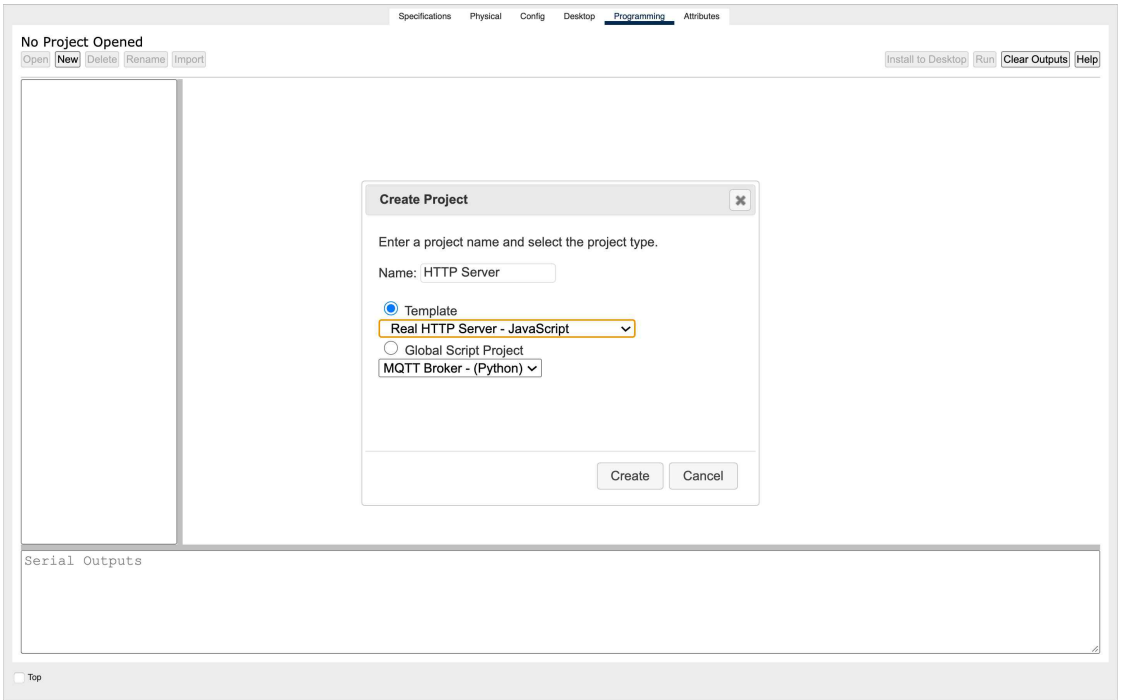


Figure 1.6: Creating a new JavaScript-based HTTP Server project.

— Why JavaScript?. The `Real HTTP Server - JavaScript` template auto-generates boilerplate code (`main.js` plus route handling) that's already set up to listen for HTTP requests. This simplifies creating basic web endpoints quickly.

8. **Open `main.js` and Run the Code:**
   In the left panel, click `main.js`, then switch to the **Run** tab (usually at the top right of the editor window).

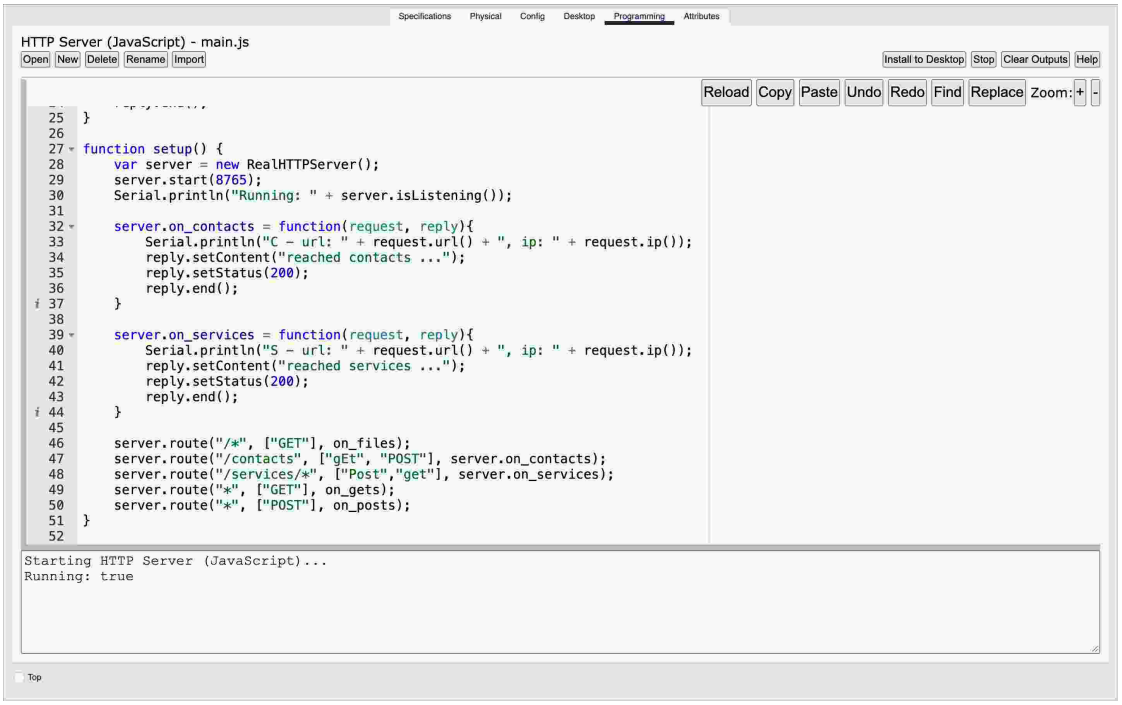


Figure 1.7: Running the newly created `main.js`.

9. **Allow Port 8765:**
   Upon running the script, Packet Tracer will prompt you to allow the SBC to listen on `port 8765`. Click *Allow* or *Yes* when asked.

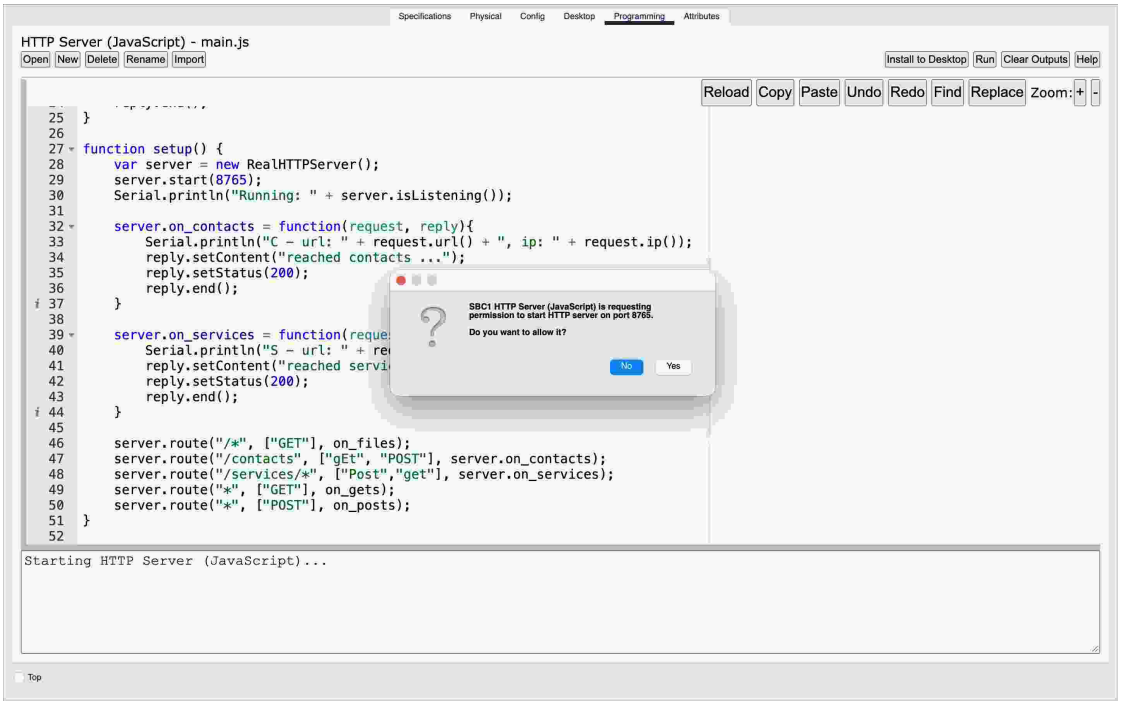


Figure 1.8: Permission pop-up for listening on port 8765.

> **— What is Port 8765?.** Web servers typically run on port 80 (HTTP) or port 443 (HTTPS). Here, we're using **8765** for demonstration. Packet Tracer's built-in firewall prompts you to confirm that this script can bind to that port.

10. **Test via Browser:**
    In your *PC's* or *SBC's* browser (Packet Tracer's built-in browser, or your real system browser if configured), navigate to:

```
127.0.0.1:8765/contacts
```

You should see a response similar to "`reached contacts...`".

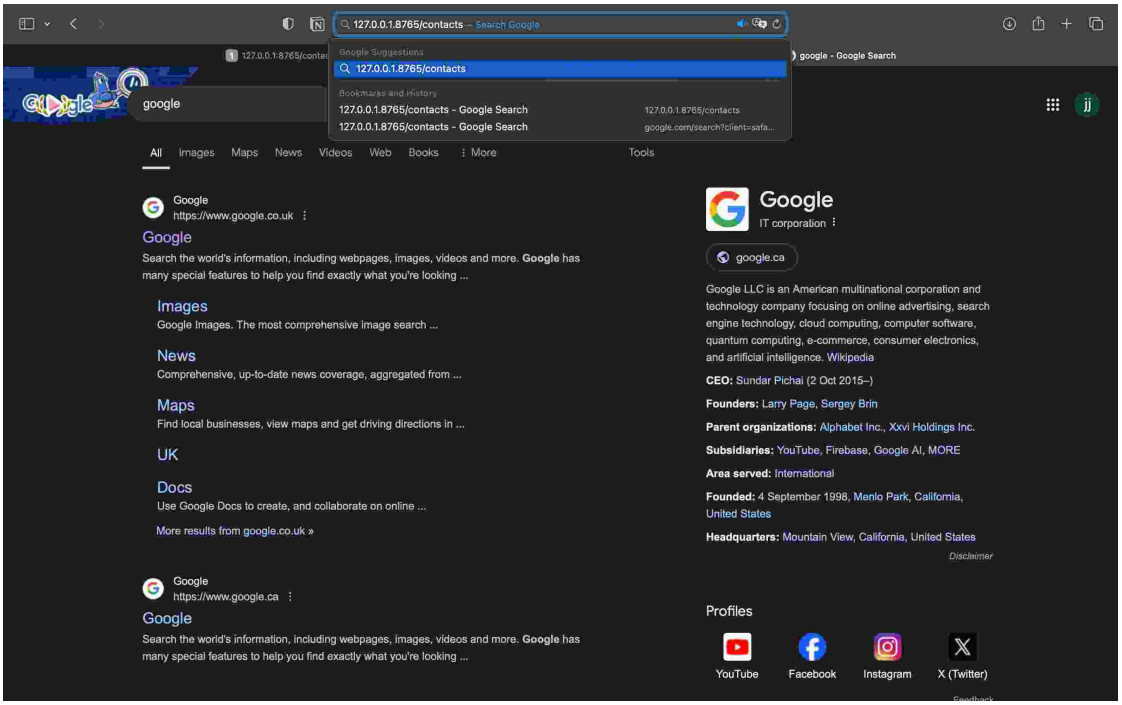


Figure 1.9: Browser pointing to `/contacts` on port 8765.

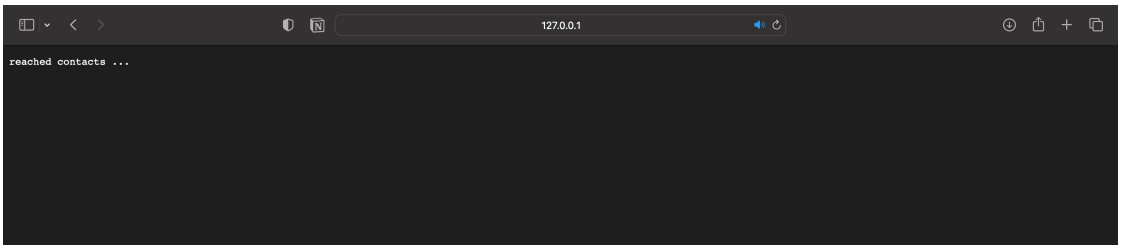


Figure 1.10: A simple web page output from the SBC.

> **— Why `127.0.0.1`?.** `127.0.0.1` is the **loopback address**, meaning the SBC itself is hosting the server. In many real networks, you'd use the SBC's assigned IP (e.g., `192.168.x.x`). Here, we test locally to confirm the server is functioning before exploring external connections.

11. **Check Program Outputs:**
    Open the **Console** or **Log** at the bottom or side of the SBC's *Programming* tab. You will see debug messages showing request handling (e.g., "GET /contacts" or "request from 127.0.0.1").

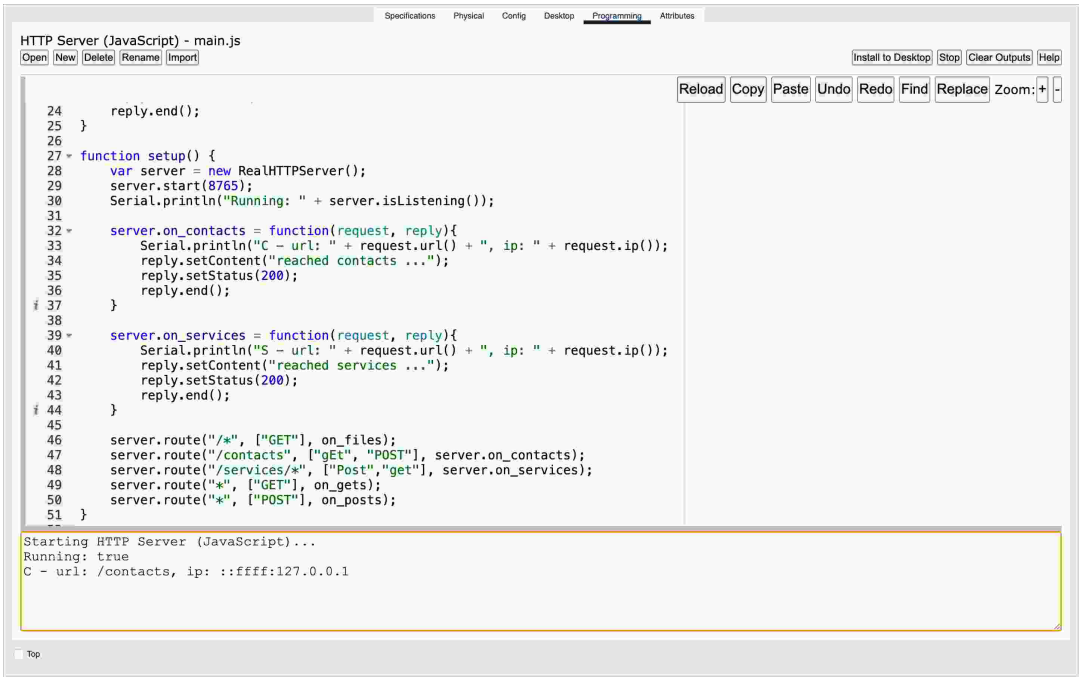


Figure 1.11: SBC console output showing request handling.

> **— Why check the console?.** Viewing the console messages helps you:
> - Confirm that HTTP requests reach your code.
> - Debug any syntax or connection issues (e.g., **port binding** errors).
> - Monitor the incoming URLs or routes that users (or other devices) are requesting.

Tips for Creating the Program:

- **Port Binding Prompts:** If you accidentally deny Packet Tracer's prompt to open port 8765, click **Stop** and re-**Run** the code. You should see the prompt again, allowing you to grant permission.
- **Double-Check the Template:** Always confirm you've selected `Real HTTP Server - JavaScript`. This ensures Packet Tracer generates `main.js`, along with default route-handling functions, simplifying your server setup.

■

D. Programming SBC Board (Modifying the Program)

12. **Change Script Values in `main.js`:**
Using the code editor in the *Programming* tab, locate and modify the following lines:
    - **Line 22:** Change `F` to `"Fail"`
    - **Line 29:** Change `8765` to `8008`
    - **Line 33:** Change `C - url` to `"Connected via url"`
    - **Line 52:** Change `gEt` to `get`

    Each of these updates will alter the behavior or appearance of your HTTP server's responses and port configuration.

> **— Why make these changes?.**
> - **Renaming "F" to "Fail"** (Line 22) might clarify an error message or status code in your script.
> - **Shifting from Port 8765 to 8008** (Line 29) demonstrates how to run an HTTP

server on a different port. This also helps you understand how to free up ports or avoid conflicts in advanced scenarios.

- **Changing "C - url" to "Connected via url"** (Line 33) makes your console or web responses more descriptive, improving readability for debugging or demonstration.
- **Correcting "gEt" to "get"** (Line 52) resolves a case-sensitivity issue. Packet Tracer's JavaScript engine is strict about capital letters in function names and route methods.

Tips for Editing the Script:

- **Syntax and Capitalization:** Packet Tracer's JavaScript engine is case-sensitive. Ensure your replacements use the exact case (e.g., changing `gEt` to `get`).
- **Comment Your Changes:** For clarity, you can add inline comments, for example:

```
// changed port to 8008
// updated gEt to get
```

- **Save Frequently:** While Packet Tracer often auto-saves your script state, it's good practice to explicitly save after each major edit to avoid losing work if you close the project.

E. Programming SBC Board (Testing the Modification)

13. **Stop the Program:**
In the *Programming* tab, click the **Stop** button to halt the currently running script (see Figure 1.12). Stopping the script ensures that any new changes (such as port numbers) will take effect when you re-run it.

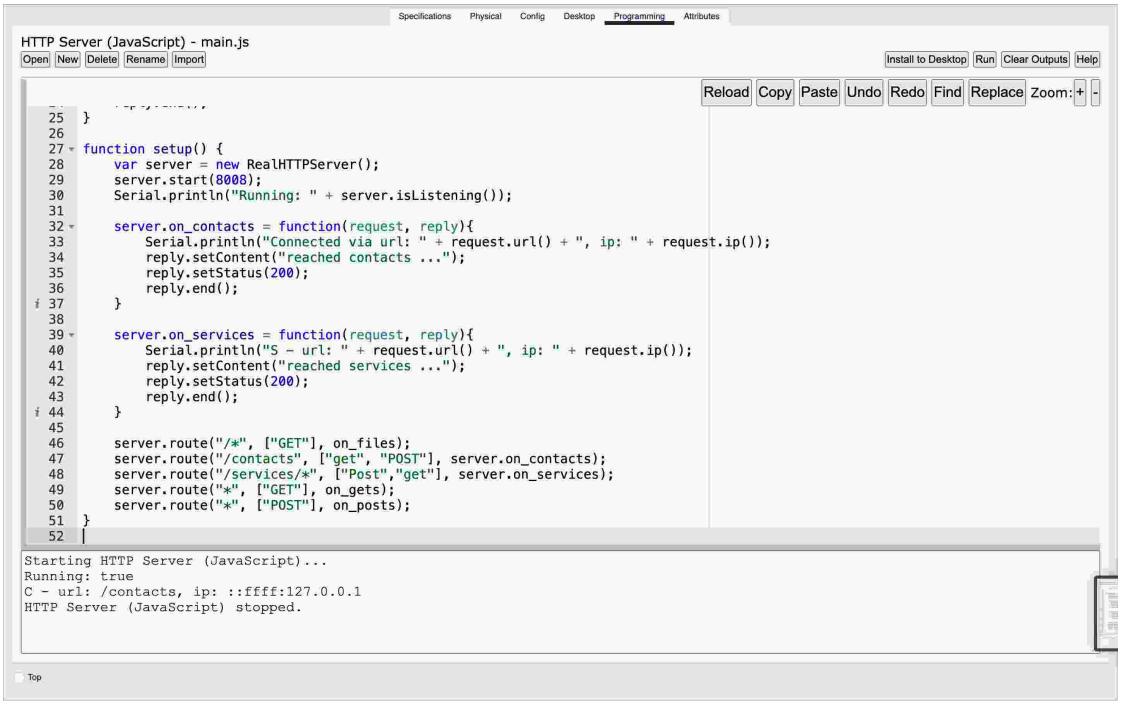


Figure 1.12: Halting the script to apply new port settings.

14. **Run the Program Again:**
    After stopping, select the **Run** button once more. Packet Tracer will prompt you to allow listening on the newly specified port, `8008`. Click **Allow** (or **Yes**) to proceed.

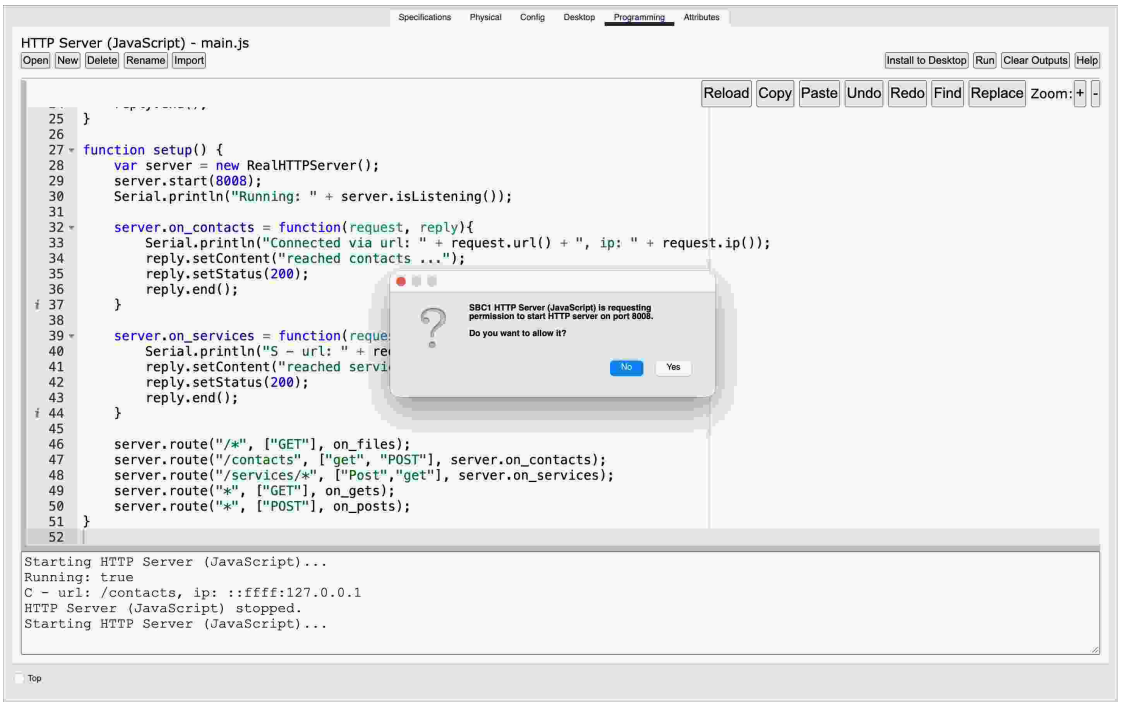


Figure 1.13: Now listening on port 8008 instead of 8765.

15. **Browse to `127.0.0.1:8008/contacts`:**
    Open a web browser (either Packet Tracer's built-in browser or your system browser) and visit:

    ```
    127.0.0.1:8008/contacts
    ```

    You should again see the response:

    ```
    reached contacts...
    ```

    which confirms the server is now running on the new port.

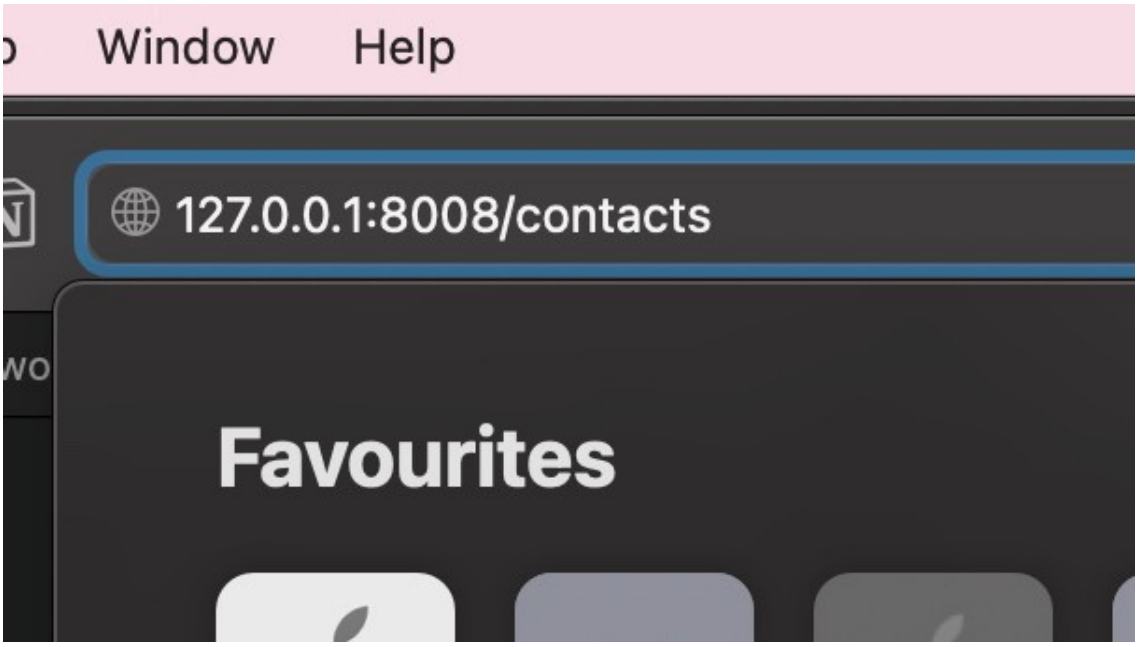


Figure 1.14: Switching to `/contacts` on port 8008.

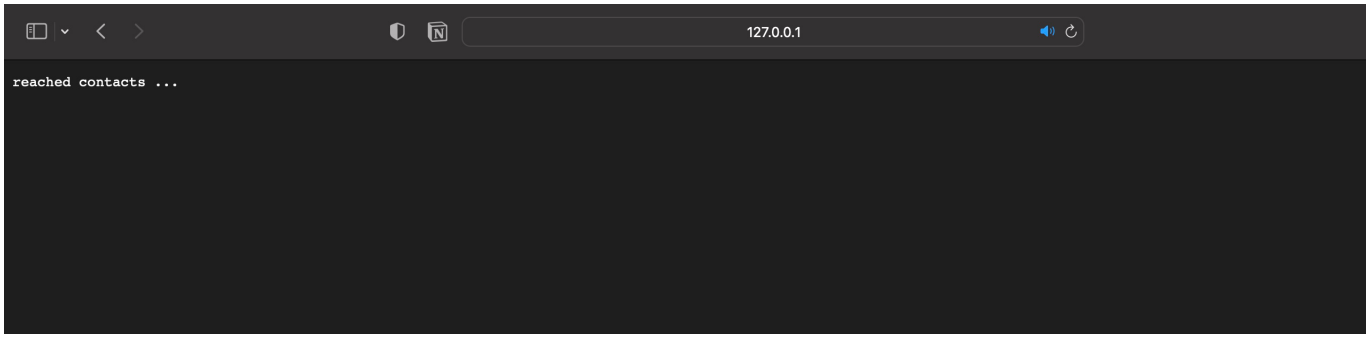


Figure 1.15: The "reached contacts..." output on the new port.

Tips for Testing the Modification:

- **Verify the Console:** In the SBC's log or console output, check for messages stating it is "listening on 8008." This confirms the change took effect.
- **Browser Choice:** You can use the *Packet Tracer Browser* (from the SBC or a local PC) or your actual computer's browser. As long as you allow the port binding, either should work.

F. Programming SBC Board (Creating Another Directory)

16. **Duplicate the `/contacts` Route:**
    In the `main.js` file, locate the line similar to:

    ```
    server.route("/contacts", ["get","POST"], server.on_contacts);
    ```

    Copy this line, then modify it as follows:

```
server.route("/hello", ["get","POST"], server.on_hello);
```

This step defines a brand-new endpoint called `/hello`, which will be handled by the function `server.on_hello`.

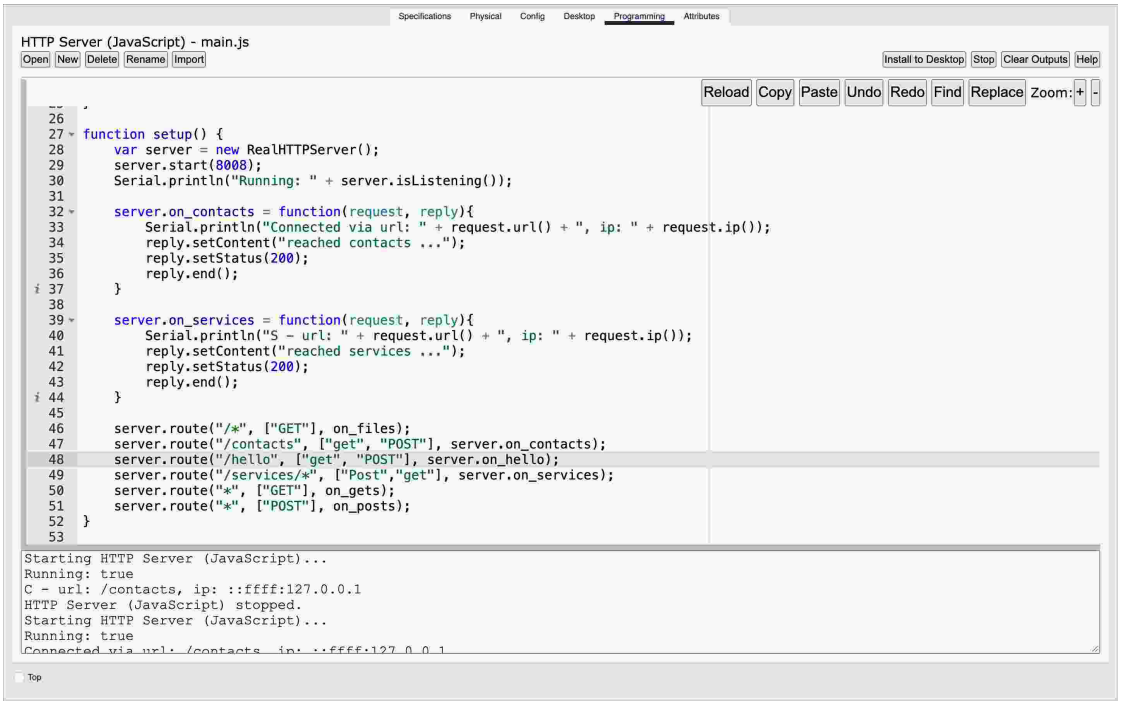


Figure 1.16: Adding a new route `/hello` to complement `/contacts`.

> **— Why another route?.** By creating multiple routes (e.g., `/contacts` and `/hello`), you expand the HTTP server's functionality. Each route can serve different content or handle unique data, much like real-world RESTful APIs or web servers.

17. **Define `server.on_hello`:**
In your code, find the function body for `server.on_contacts` (often near line 32 in the generated template). Copy that entire function, paste it beneath, and rename it to `server.on_hello`. Then change the response text to:

```
"Hello World!"
```

This way, when `/hello` is accessed, you return a distinct message.

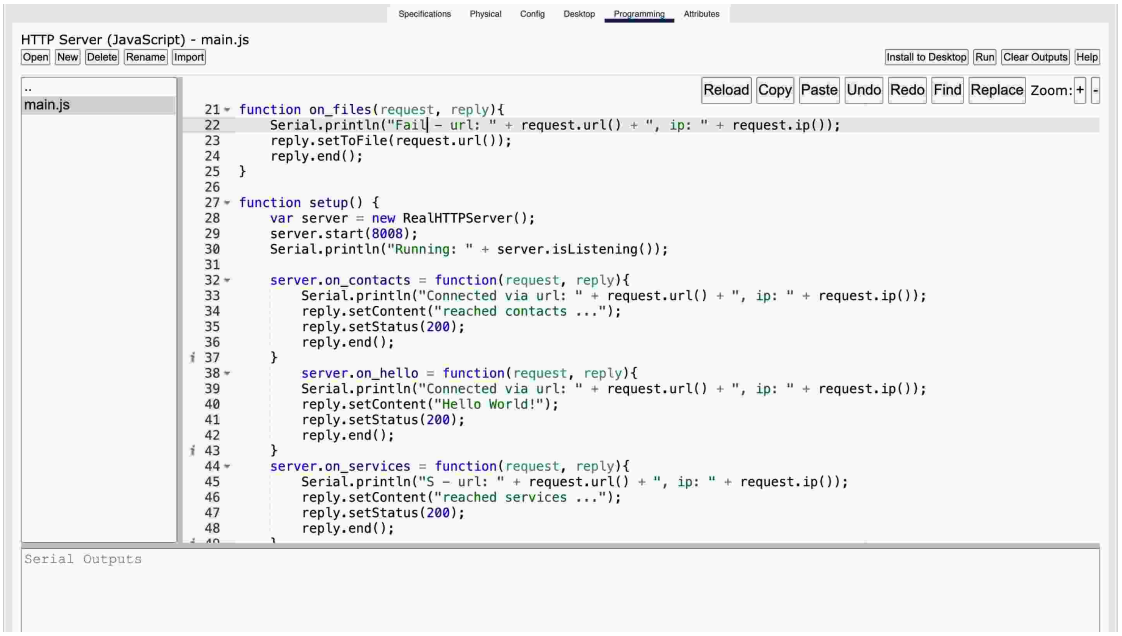


Figure 1.17: Final code block for the new `/hello` route.

> **— Why a separate function?.** Each route needs its own handler function (e.g., `server.on_hello`). This keeps your code organized and helps manage different endpoints independently. Reusing the structure from `server.on_contacts` is a quick way to maintain a consistent coding pattern.

18. **Stop, Re-run, and Test:**

    - Click the **Stop** button to halt the current script.
    - Click **Run** again, allowing it to listen on port 8008.
    - In your browser, visit `127.0.0.1:8008/hello`.
    - You should see the message: `"Hello World!"`

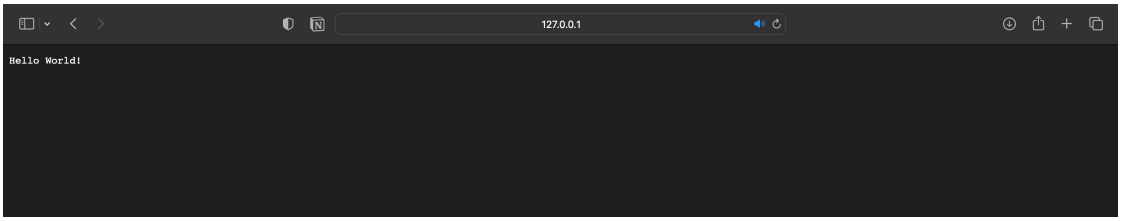


Figure 1.18: Browser now responding with `"Hello World!"` at `/hello`.

Tips for Multiple Routes:

- **Practice Adding More:** Experiment with adding additional routes like `/status` or `/about`. Each new route requires:
  1. A `server.route("/routeName", [...], server.on_routeName);`
  2. A corresponding function `server.on_routeName` in `main.js`.
- **GET vs. POST:** While the route definition includes both `["get","POST"]`, your web browser primarily sends GET requests. To experiment with POST, try tools like Postman

or cURL to send data and see how your server handles various request methods.

G. Programming SBC Board (Adding to Your Web Page)

19. **Copy the Following HTML:**
Below is a sample snippet you can embed into your response:

```
<html><head><title>Hello World Title</title></head>
<body><h1>Hello World</h1>
<p>Lorem ipsum ...</p></body></html>
```

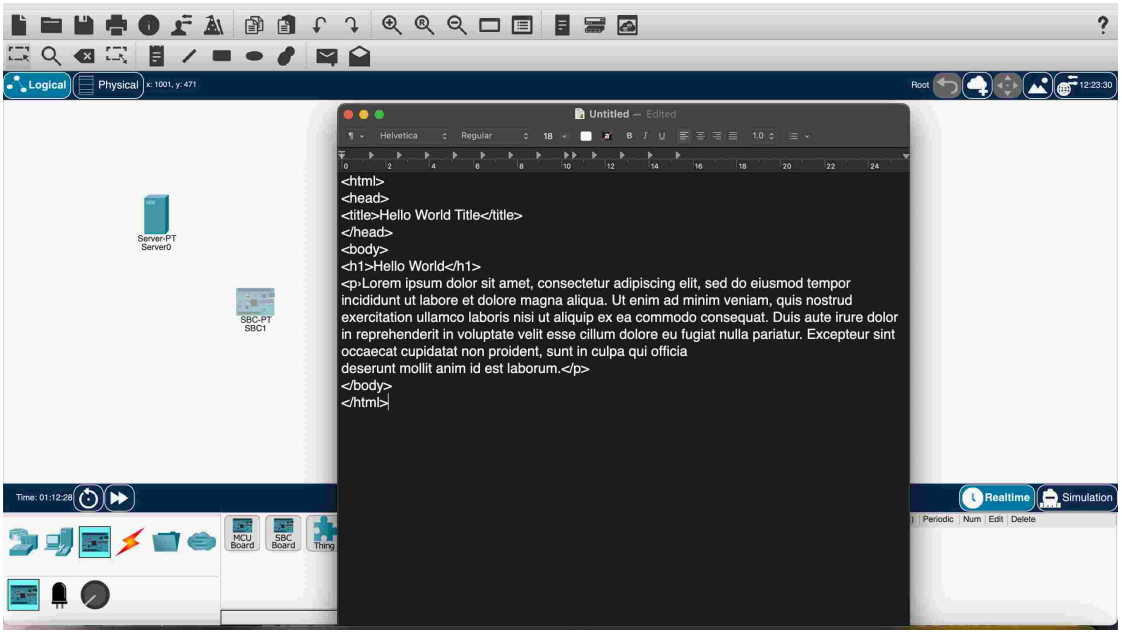


Figure 1.19: Multi-line HTML snippet to embed in the response (for reference).

— Why this HTML snippet?. Adding a simple HTML structure (`<html>`, `<head>`, `<body>`, etc.) introduces you to embedding more complex web content. You can quickly customize headings, paragraphs, or other elements to suit your project's needs.

20. **Replace "Hello World!" in `on_hello`:**
In your `main.js`, locate the `server.on_hello` function. Remove the old `"Hello World!"` text and paste the HTML snippet in its place (Figure 1.20).
Important: Keep the entire HTML code on a single line to avoid "Unclosed string" errors in Packet Tracer.

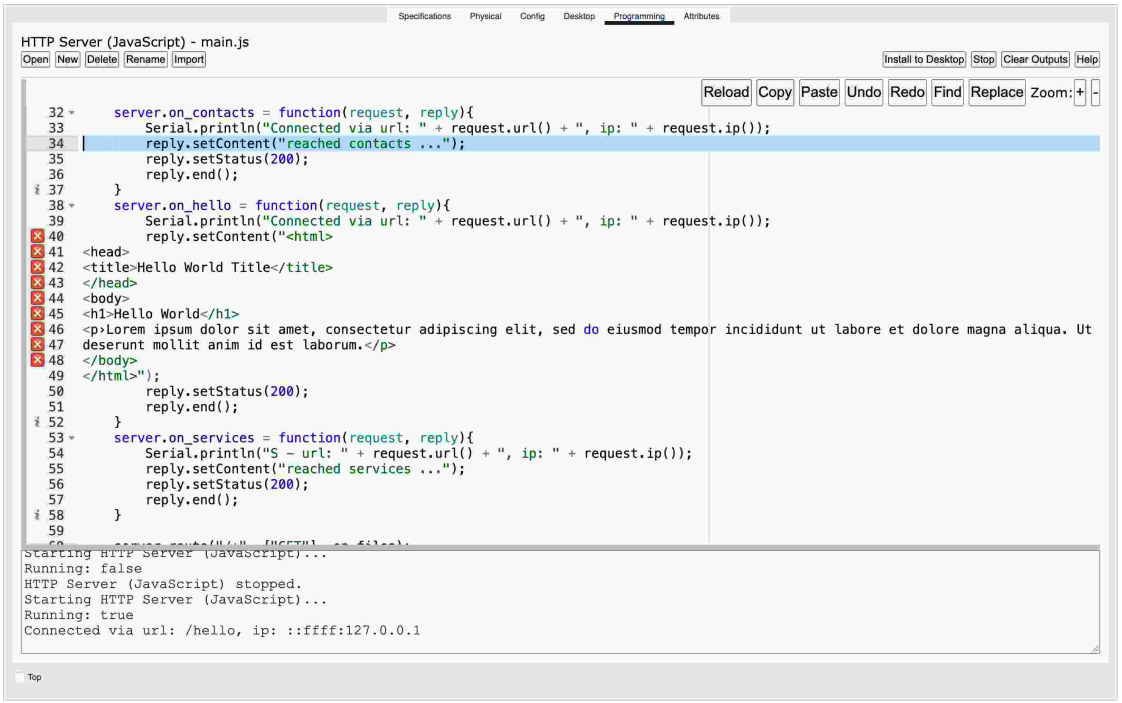


Figure 1.20: HTML pasted into the code on a single line.

> **— Why the one-line requirement?.** Packet Tracer's JavaScript parser treats line breaks within a quoted string as syntax errors. If you need multiple lines, use either string concatenation or the correct escape characters to avoid the "Unclosed string" error.

21. **Stop, Re-run, and Check the Page:**

    - Stop the script to apply your changes.
    - Run it again, ensuring port `8008` is allowed.
    - In your browser, navigate to `127.0.0.1:8008/hello`.
    - You should see the HTML content (title, heading, and paragraph) rendered in the browser.

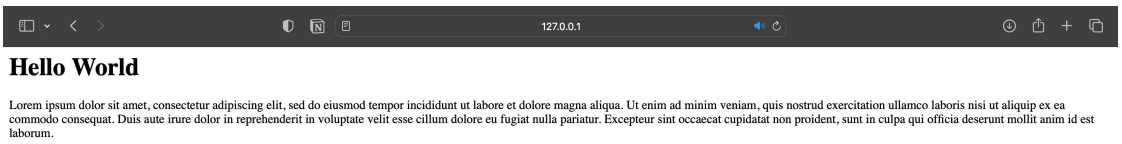


Figure 1.21: Embedded HTML now visible at the `/hello` route.

Tips for Embedding HTML:

- **One-Line Requirement:** If you insert line breaks in the string, Packet Tracer may report an "Unclosed string" error. Keep your snippet on one line or properly escape line breaks with `\n`.
- **Add More Elements:** Experiment with tags like `<h2>`, `<ul>`, `<li>`, or even images to

see how you can enrich your embedded web page. This lets you create more sophisticated interfaces or information displays.

H. Programming on the Server

22. **Why Move the Script to the Server?**
    In Packet Tracer, the standard **Server** device has a built-in file system for storing multiple scripts and files. This makes it especially useful if you're planning more complex or larger-scale labs. You can follow essentially the same steps you did with the SBC, but:
    - Use a different port, such as `8001` (instead of `8765` or `8008`).
    - Provide a different route name, like `/HelloServer0`.

    This way, you can keep your SBC project and the Server's project separate, each serving different HTTP endpoints.
23. **Open Another Browser Tab:**
    After creating and running your JavaScript-based HTTP server on the **Server** device, open a new browser tab (either in the built-in Packet Tracer browser or on your host system, if properly configured). Navigate to:

    ```
    127.0.0.1:8001/HelloServer0
    ```

    to access your newly hosted route.

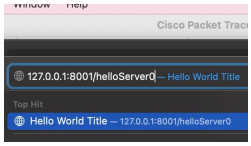


Figure 1.22: Accessing the route `/HelloServer0` on port `8001`.

24. **Confirm "Hello World Server0":**
    If everything is configured correctly, your browser should display a page (Figure 1.23) containing the text `"Hello World Server0"` or whichever custom message you defined in your script.

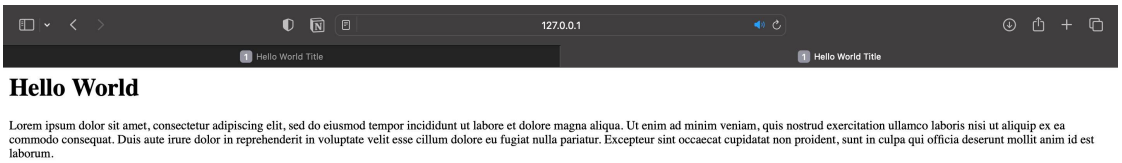


Figure 1.23: Final page served by the standard Server device.

Tips for Using the Server Device:

- **Built-in File System:** The Packet Tracer **Server** device comes with its own file system, allowing you to store multiple scripts or web files. This is more convenient for bigger or multi-service labs, where you might host multiple pages or applications.
- **Multiple Ports in Parallel:** Want to simulate multiple web services? You can continue

running your SBC HTTP server on port `8008`, while simultaneously hosting another script on the **Server** at port `8001`. This is handy for testing how different client devices interact with different endpoints on the same network.

■

Measuring Success

- **SBC's HTTP Server Responses:** You can navigate to `127.0.0.1:8008/hello` and see the HTML content you embedded, confirming that the `/hello` route is operational.
- **Route Flexibility:** If you created a second or third route (e.g., `/about`), you can visit it in the browser and see your custom text—verifying your ability to duplicate and modify route logic.
- **Server-Side Hosting on Port 8001:** You can open `127.0.0.1:8001/HelloServer0` to see "Hello World Server0," demonstrating a successful HTTP service on Packet Tracer's standard Server.
- **No Port or Syntax Errors:** The Packet Tracer console shows no messages like "Failed to bind port" or "Unclosed string," confirming that your code is stable and properly configured.

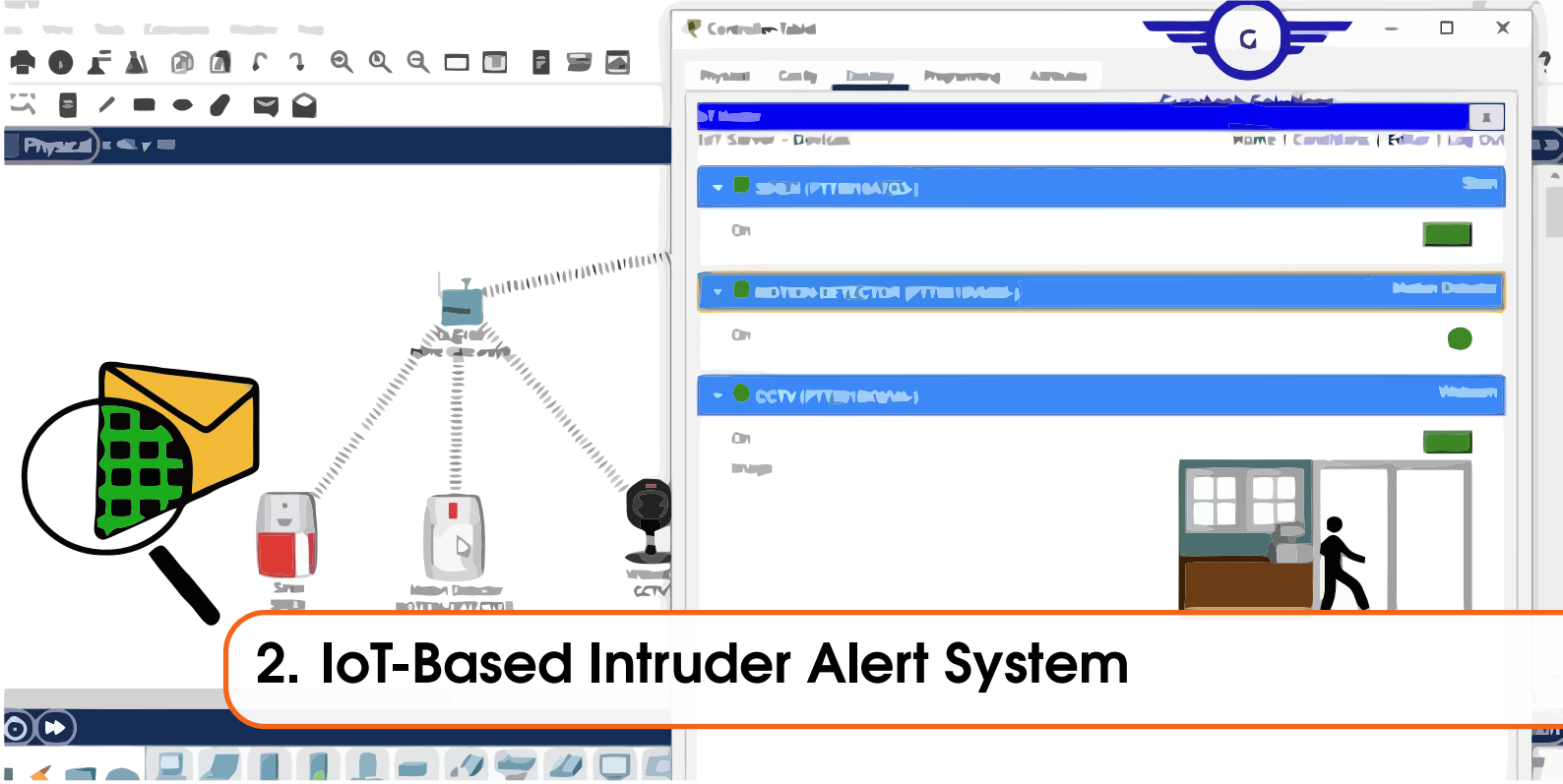
■

Summary

You have successfully **modified an existing script in Packet Tracer** to connect to real-world networks using a JavaScript-based HTTP server. Starting with an SBC board, you added routes (`/contacts` and `/hello`), embedded HTML, changed ports, and verified everything in a web browser. You then repeated a similar process on a standard Server device for comparison, demonstrating how Packet Tracer can simulate a variety of HTTP-based scenarios.

— Further Exploration

- Experiment with additional device states (e.g., `MEDIUM` or `ALERT`).
- Add environment-based triggers (like temperature or motion) to automatically change an IoT device's state or route response.
- Combine multiple IoT devices and scripts to build a more integrated smart home scenario.



2. IoT-Based Intruder Alert System

Introduction

This lab demonstrates how to build and configure an **IoT-based intruder detection system** in Cisco Packet Tracer. You will create a network with motion detectors, sirens, and CCTV cameras, all registering to a Home Gateway server. By defining conditions and programming devices to react to motion, you will simulate a functional security system capable of detecting intruders and triggering alarms.

Objectives

- **Set up** a simulated IoT system in Packet Tracer to detect intruders.
- **Build and configure** a topology with motion detectors, sirens, and CCTV cameras.
- **Register** IoT devices to a Home Gateway server.
- **Program devices** to react to motion, activating the CCTV and Siren.
- **Test the system** by simulating motion detection and ensuring correct responses.

Lab Plan

A. **Launch Packet Tracer**
B. **Build the Topology**
C. **Configure the Tablet (Wireless Settings)**
D. **Register IoT Devices with Home Gateway**
E. **Explore Devices on the Network**
F. **Connecting IoT Devices (Conditions)**
G. **Testing the Motion Detector Rules**

Required Software

- **Cisco Packet Tracer 8.x** (or newer)
- Basic knowledge of IoT device configuration in Packet Tracer
- Optional: external PC browser if you wish to test advanced user interfaces

A. Launch Packet Tracer

1. **Open Packet Tracer:**
   Double-click the Packet Tracer icon on your desktop or locate it in your applications folder to launch. You should see a blank default Logical topology workspace similar to the figure below.

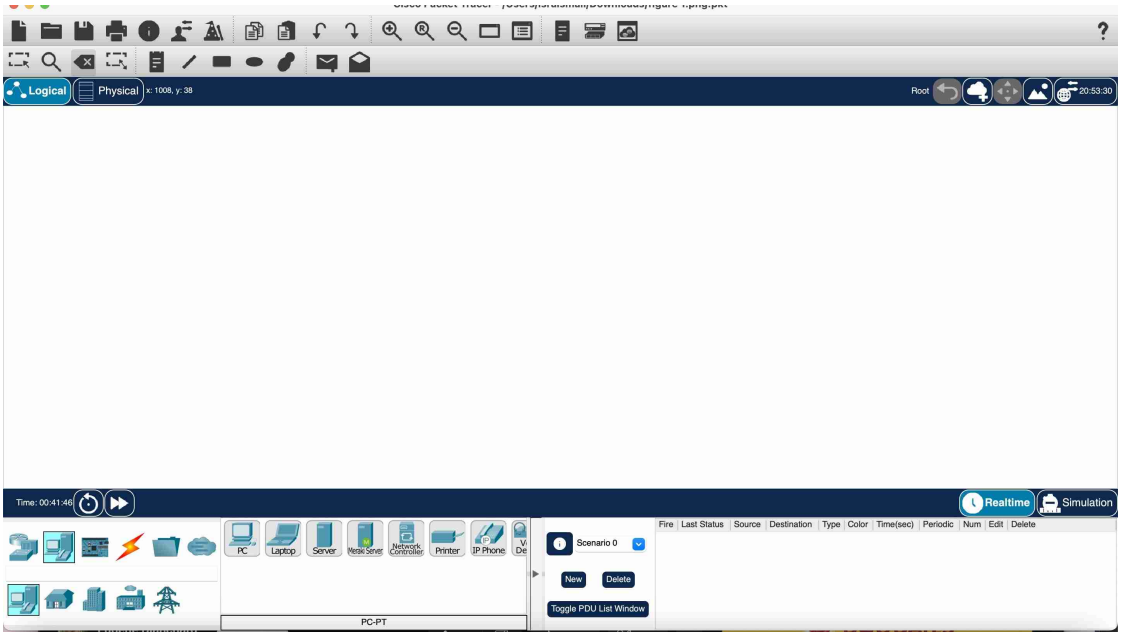


Figure 2.1: Initial Workspace (blank default topology).

B. Build the Topology

2. **Add Network Devices:**
   In Packet Tracer's lower-left pane, hover over the *device categories* and *subcategories* to locate each IoT component. Drag and drop the following onto your workspace:
   - **Home Gateway**
   - **Motion Detectors**
   - **Sirens**
   - **CCTV Cameras**

   Arrange them to reflect how you want the network to appear. See Figure 2.2 for an example layout.

   — Why these devices?.
   - **Home Gateway:** Acts as a central hub or router for IoT devices, often providing wireless connectivity and an IoT registration service.
   - **Motion Detectors, Sirens, and CCTV Cameras:** Common security-focused IoT devices that communicate with the Home Gateway.

   Collectively, these devices form a basic intruder alert or security monitoring system in Packet Tracer's IoT environment.

3. **Rename Devices:**
   After placing each device, click on it, navigate to the **Config** tab, and modify the *Display Name* to something descriptive (e.g., "`Siren1`," "`MotionSensor1`," "`CCTV1`"). Meaningful labels make it easier to differentiate between multiple devices of the same type.

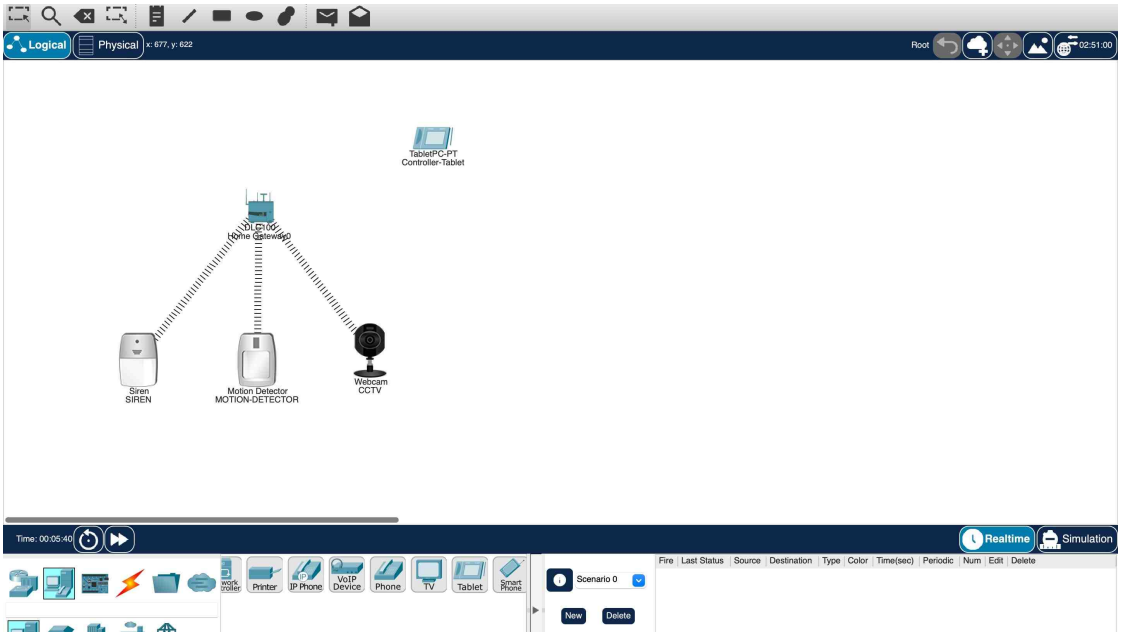


Figure 2.2: Topology with motion detectors, sirens, and cameras.

> **— Why rename?.** In larger or more complex labs, you might have multiple sirens, cameras, and sensors. Assigning clear names (e.g., *CCTV1*, *CCTV2*, etc.) prevents confusion when configuring or troubleshooting. This also simulates real-world best practices for device naming.

C. Configure the Tablet (Wireless Settings)

4. **Open the Tablet's Config Tab:**
   Click on the **Tablet** device to open its configuration window, then select the *Config* tab at the top. The left panel will display available interfaces and settings.

   > **— Why do this?.** The *Config* tab provides a streamlined way to rename devices, adjust network settings, and configure wireless parameters without using the CLI. This is especially helpful if you're new to IoT devices in Packet Tracer.

5. **Change Wireless SSID:**
   In the left pane, click on **Wireless0** (the wireless interface). Change the `SSID` field from `"Default"` to `"HomeGateway"`. This sets the name of the wireless network the Tablet will join.

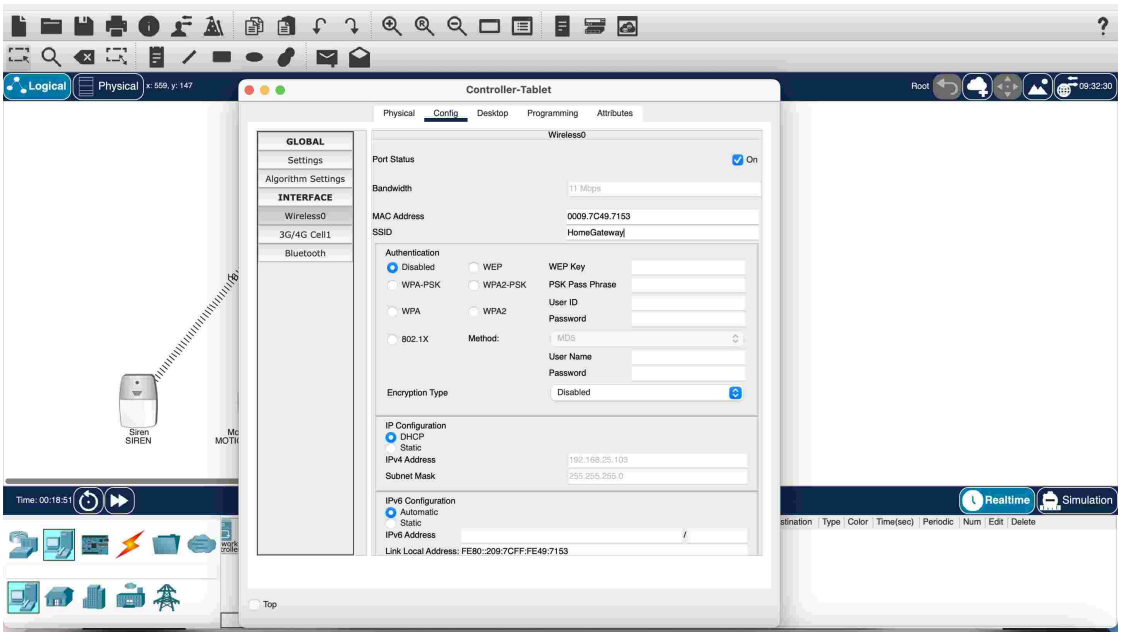


Figure 2.3: Wireless settings on the Tablet.

— Why rename the SSID?.

- Using a clear SSID like `"HomeGateway"` helps differentiate this wireless network from others, especially in complex topologies.
- It ensures that any wireless device (like the Tablet) knows exactly which network to connect to.

6. **Verify Connection:**
   After changing the SSID, you should see a **wireless link** (often shown as a dotted or curved line) automatically forming between the Tablet and the **Home Gateway**, as shown in Figure 4.4.

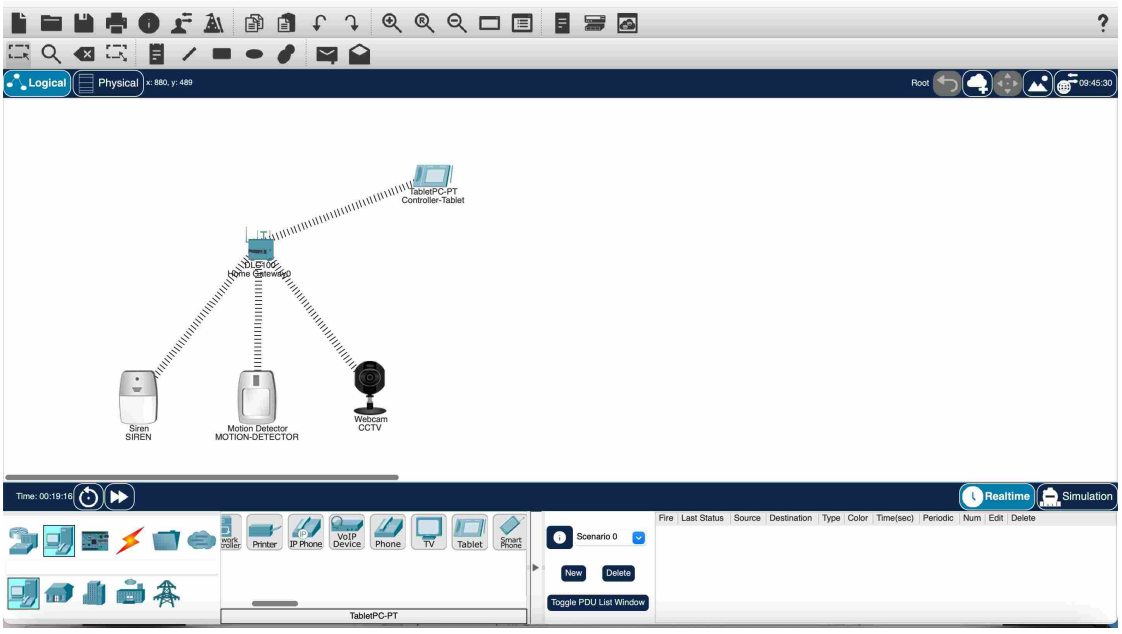


Figure 2.4: Wireless connection established between Tablet and Gateway.

— How to confirm the wireless link?.
- Check for a wireless symbol or a dotted line directly connecting the Tablet to the Gateway in the Logical workspace.
- Alternatively, hover your cursor over the Tablet or Gateway to see if the pop-up information indicates a wireless association (e.g., signal strength or connected SSID).

D. Register IoT Devices with Home Gateway

7. **Siren Registration:**

   - Click on the **Siren** device in the workspace.
   - Go to *Config* → *Settings*.
   - Under **IoT Server** (usually a dropdown or selection box), choose **Home Gateway**.

   This tells the Siren to communicate and register with the Home Gateway, making it controllable via the gateway's IoT services.
8. **Motion Detector Registration:**

   - Click on the **Motion Detector** device.
   - Navigate to *Config* → *Settings*.
   - Under **IoT Server**, select **Home Gateway** again.

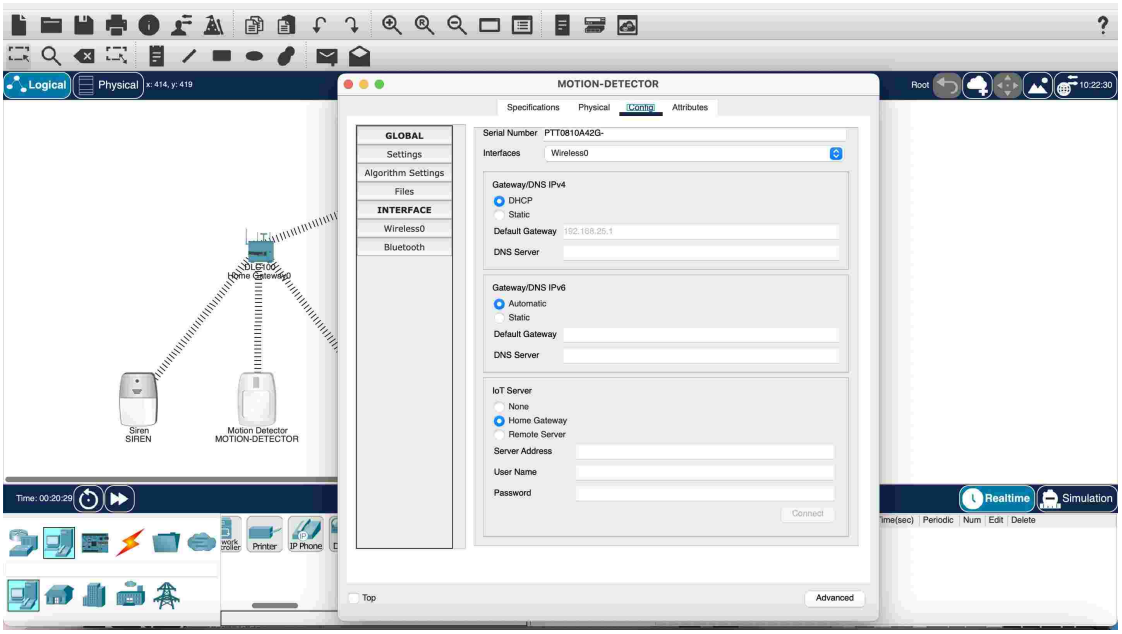


Figure 2.5: Registering devices to Home Gateway (example).

— Why register Siren and Motion Detector?.
- By registering, the devices become visible to the Home Gateway's IoT Monitor.
- It allows you to create automated rules (e.g., *if Motion Detector = triggered, then activate Siren*).

9. **Webcam Registration:**

   - Click on the **Webcam** icon.
   - Go to *Config* → *Settings*.

- Under **IoT Server**, choose **Home Gateway**.

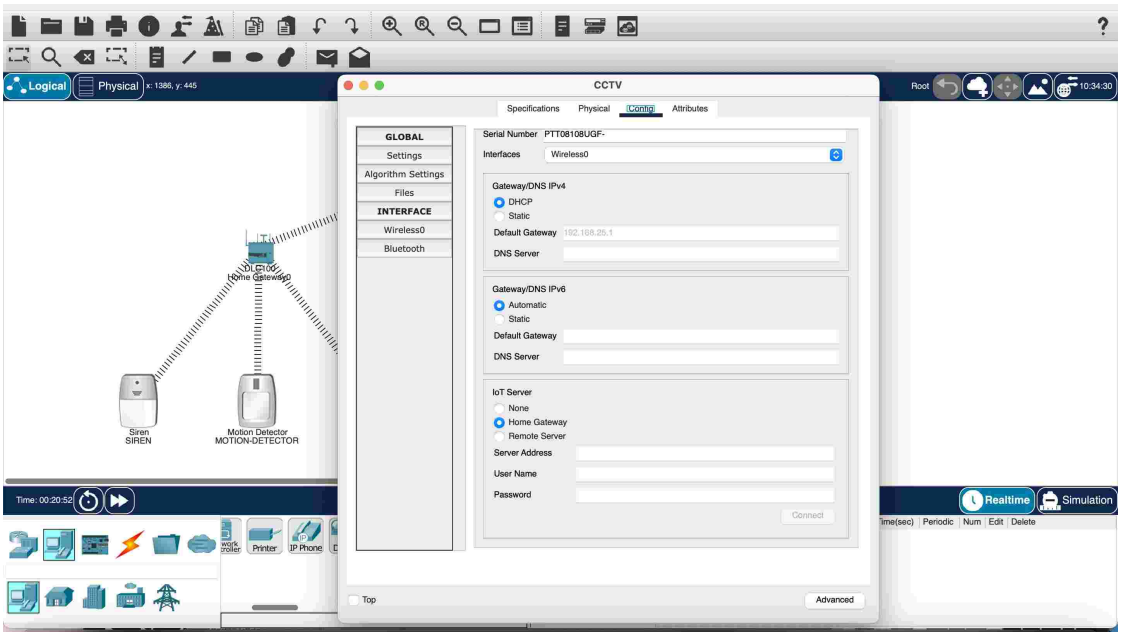


Figure 2.6: Registering the Webcam to Home Gateway.

— Why register the Webcam?.

- The **Webcam** can record or stream footage upon detecting motion or other triggers.
- Linking it to the Home Gateway helps centralize control and monitoring (for example, the gateway can activate recording when motion is detected).

E. Explore Devices on the Network

10. **Activating Devices (Alt-Key):**
    In Packet Tracer, holding down the **Alt** key while hovering your cursor over certain IoT devices can simulate triggering or interacting with them.
    - For example, with a **Motion Detector**, pressing `Alt` causes its *red light* to appear, indicating activation or detection.

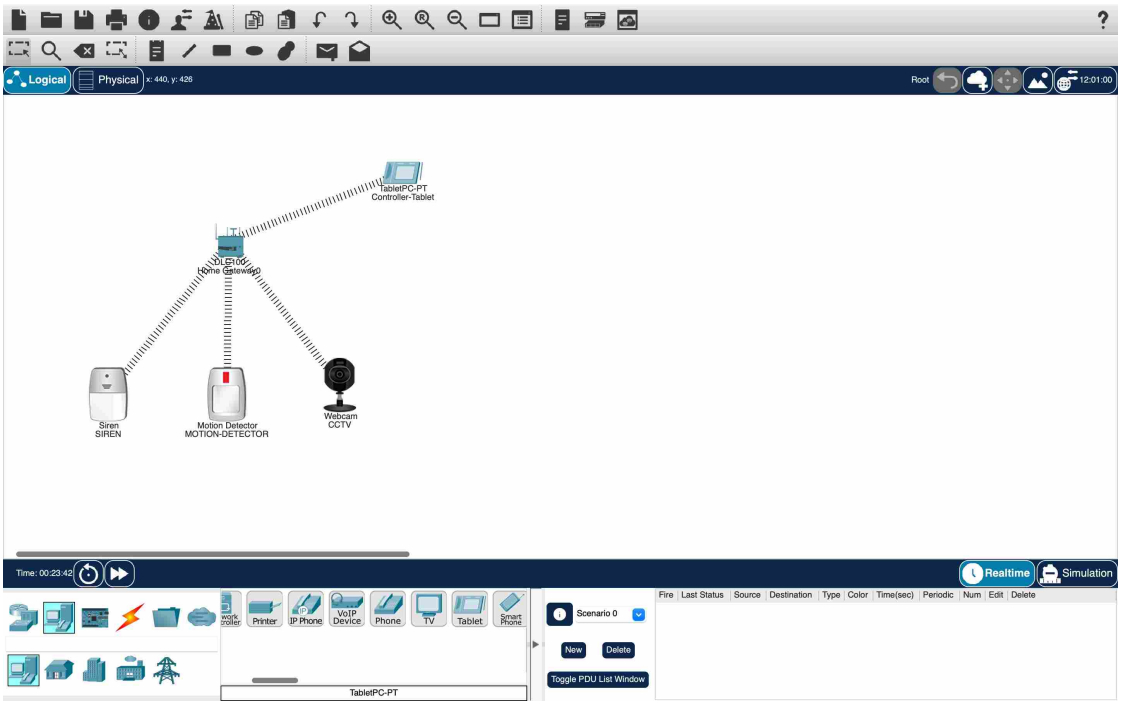


Figure 2.7: Motion detector showing a red light when activated (via Alt + hover).

— Why use `Alt`?.

- **Triggers Device State:** In Packet Tracer's IoT environment, *Alt* + *hover* is a quick way to simulate real-world sensor inputs (e.g., motion or heat).
- **Testing Reactions:** This helps you check if other devices (like sirens or CCTV) respond correctly to a triggered sensor without needing a fully automated script.

11. **Viewing Device Info:**
Simply hover your cursor over a device (e.g., the Tablet) to see a pop-up with basic network data like:

    - **IP address**
    - **MAC address**
    - **Device Name**

    This allows a quick overview of how devices are identified and connected.

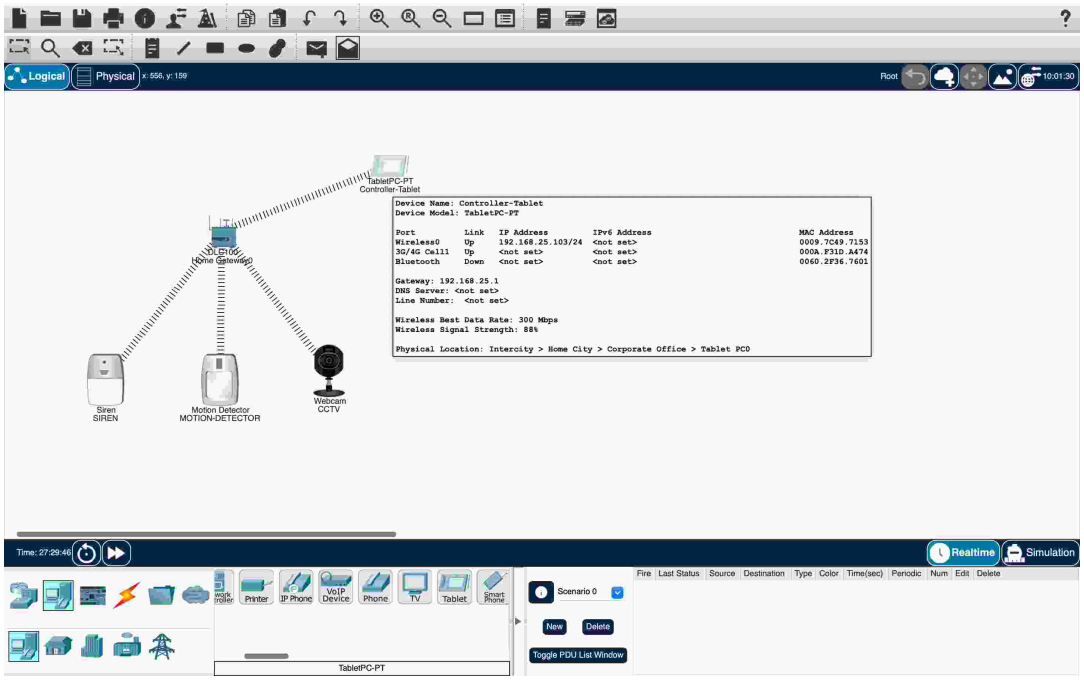


Figure 2.8: Viewing device information (IP, MAC, etc.) by hovering.

12. **Open IoT Monitor on Tablet:**

    - Click the **Tablet**.
    - Switch to the *Desktop* tab.
    - Select *IoT Monitor*.

    The IoT Monitor is where you can log in to the Home Gateway and see or control registered IoT devices.

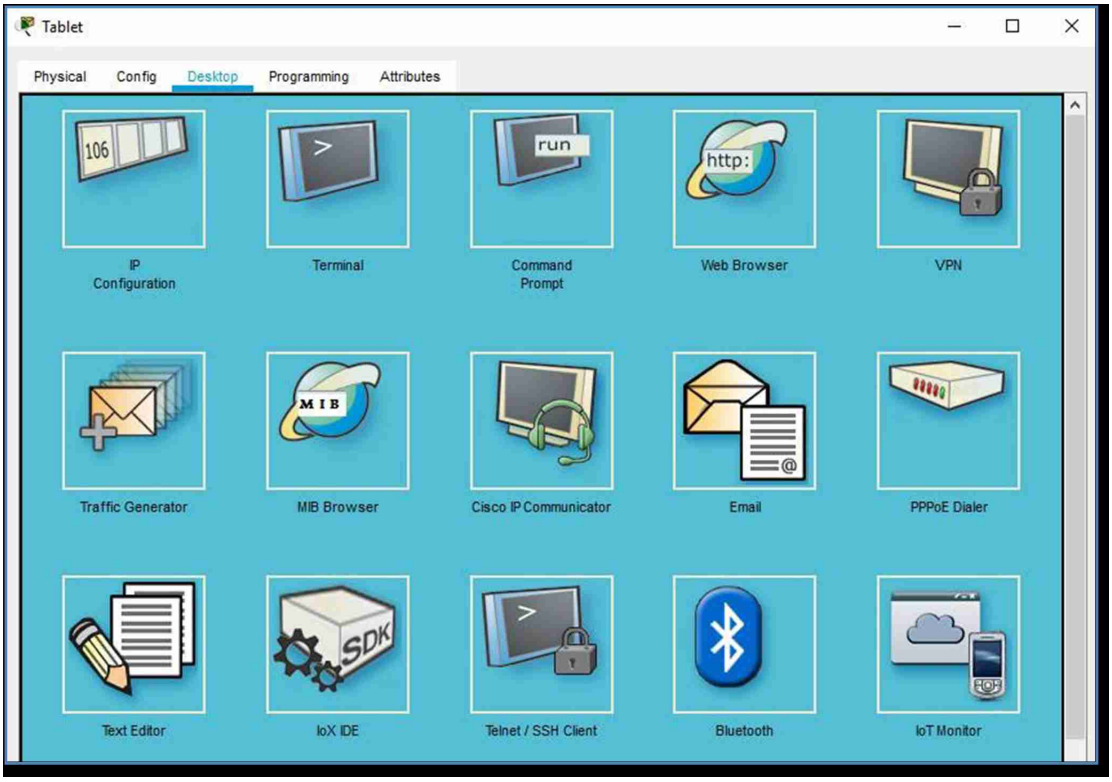


Figure 2.9: Accessing IoT Monitor from the Tablet's desktop.

— What is IoT Monitor?.

- **Central Management**: It gives a web-based interface to review and manage all IoT devices connected to the Home Gateway.
- **Rules & Automation**: You can also configure triggers or actions (e.g., *if motion detected, then turn on siren*).

13. **Log in to Home Gateway:**

- In IoT Monitor, enter the **Home Gateway IP** (e.g., `192.168.25.1`).
- Use default credentials, typically `admin / admin`.

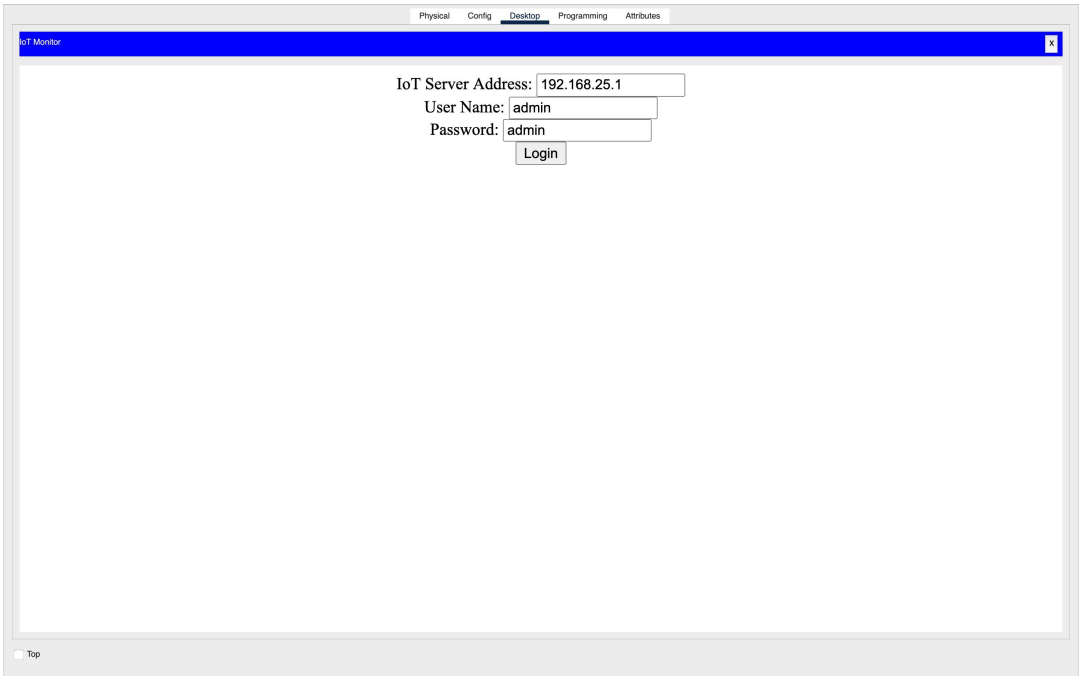


Figure 2.10: Logging into Home Gateway via the Tablet.

14. **View Connected Devices:**
Once logged in, a list of all registered IoT devices (Sirens, Motion Sensors, Webcams, etc.) appears. Select any device to see more details.

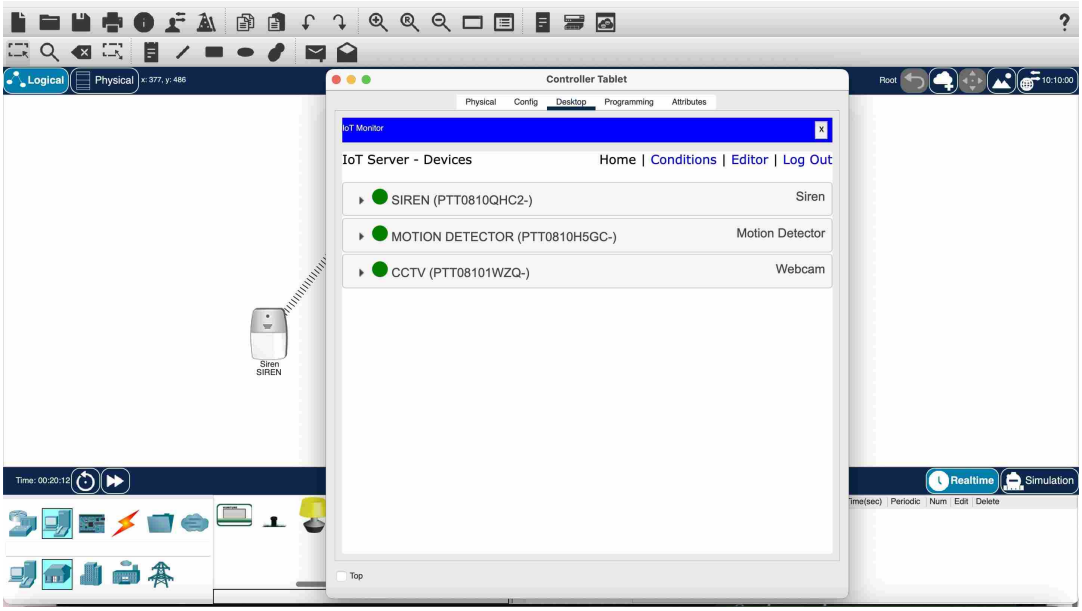


Figure 2.11: List of IoT server devices on Home Gateway.

15. **Check Device Status/Settings:**

- Clicking a device in the IoT Monitor reveals its **Status** and **Settings**.
- This can include *on/off* states, battery levels, or any special configurations specific to

that device.

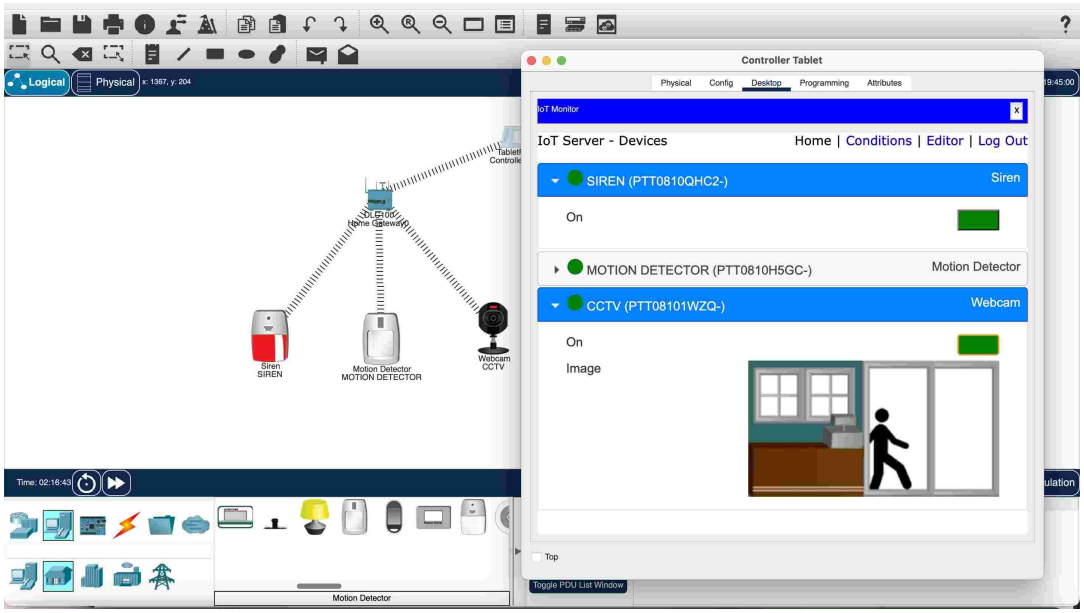


Figure 2.12: Status and configuration options for connected IoT devices.

— How to use these details?.

- **Troubleshoot**: If a device isn't responding, verify here if it's registered, online, or has any error states.
- **Manage Remotely**: Change settings such as sensitivity for motion sensors or volume for sirens without physically "touching" the device in the workspace.

F. Connecting IoT Devices (Conditions)

Goal: Link the CCTV and Siren to the Motion Detector so that when the motion detector is "on," the other devices activate as well.

16. **Add a Motion Detector Rule:**

    - On the Tablet's **IoT Monitor** interface, click on the **Conditions** tab.
    - Select the **Add Rule** button and name your new rule `MOTIONDETECT`.

    This rule will define what happens when the motion sensor detects movement.

17. **Set Condition:**

    - Within the new `MOTIONDETECT` rule, choose "`motion detector`" as the device.
    - Under *State*, pick "`on`," which should auto-fill "`true`" or a similar boolean value indicating motion is detected.

18. **Set Actions:**

    - Under *Actions*, select `SIREN` and set `on` = `true`.
    - Click **Add Action** to also set `CCTV` on = `true`.

    This means that once the **Motion Detector** is `on`, it will automatically turn the **Siren** and **CCTV** `on` as well.

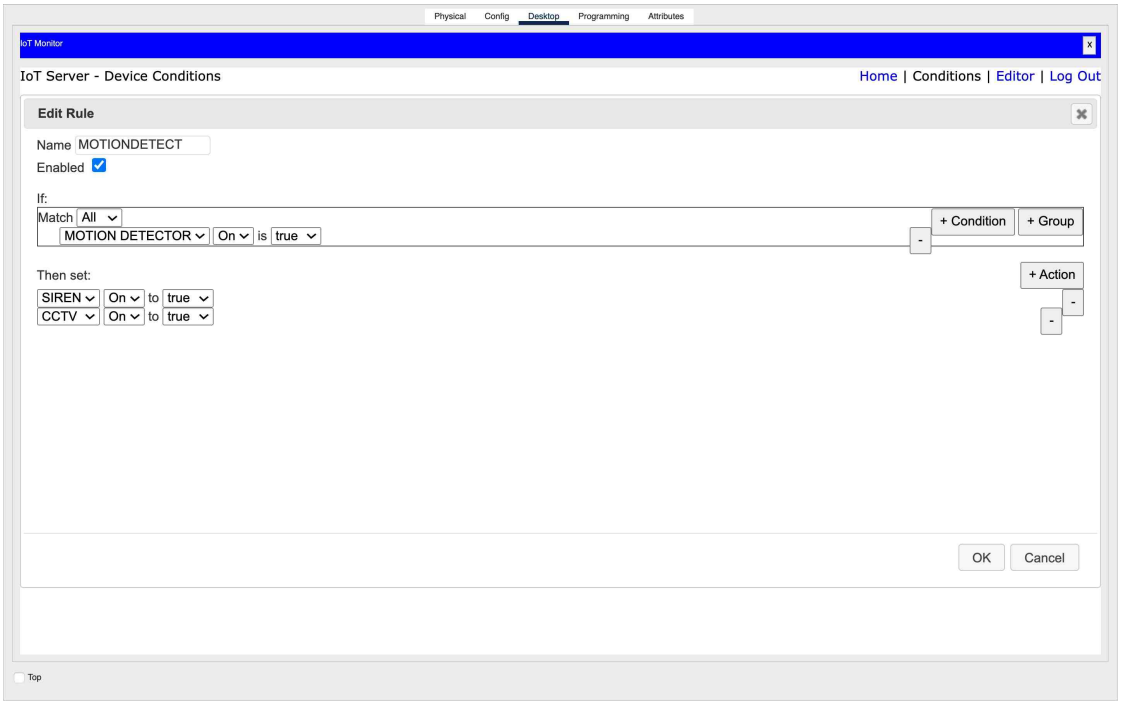


Figure 2.13: Creating the first condition for motion detection.

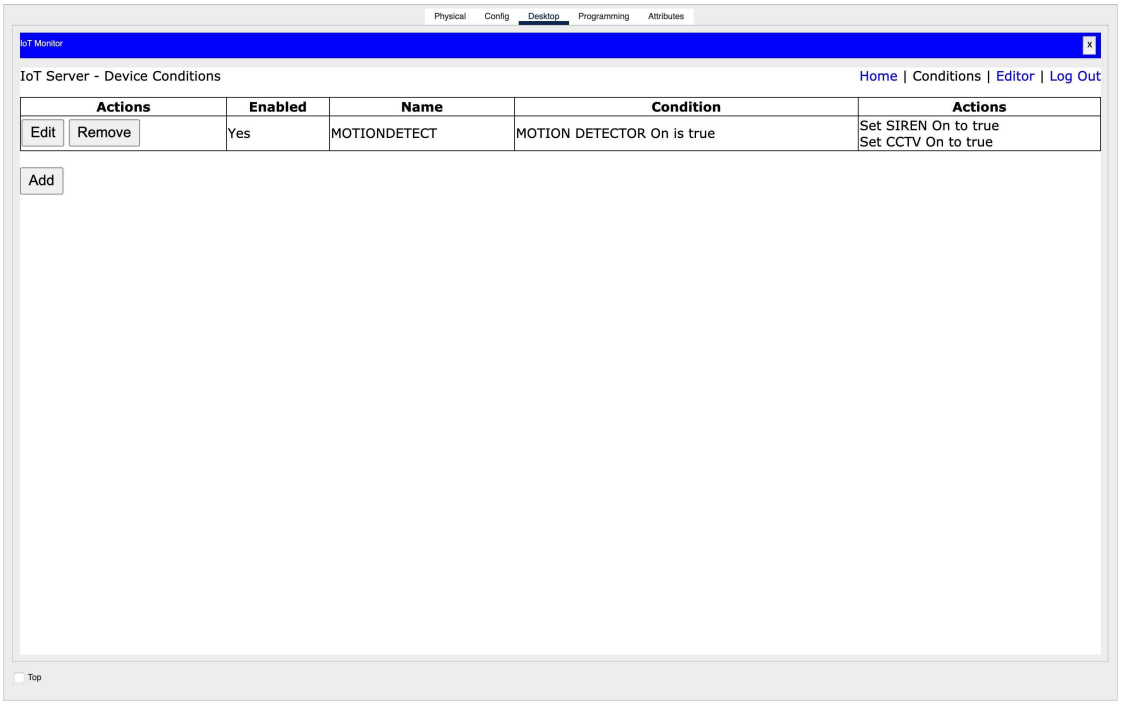


Figure 2.14: CCTV and Siren set to “on” when motion is detected.

19. **Test the Setup:**

- Close the tablet window (or leave it open to observe status updates).
- Hold **Alt/Option** and hover your cursor over the **Motion Detector** device in the

workspace.

- As soon as the motion detector is triggered, both the **Siren** and **CCTV** should switch on.

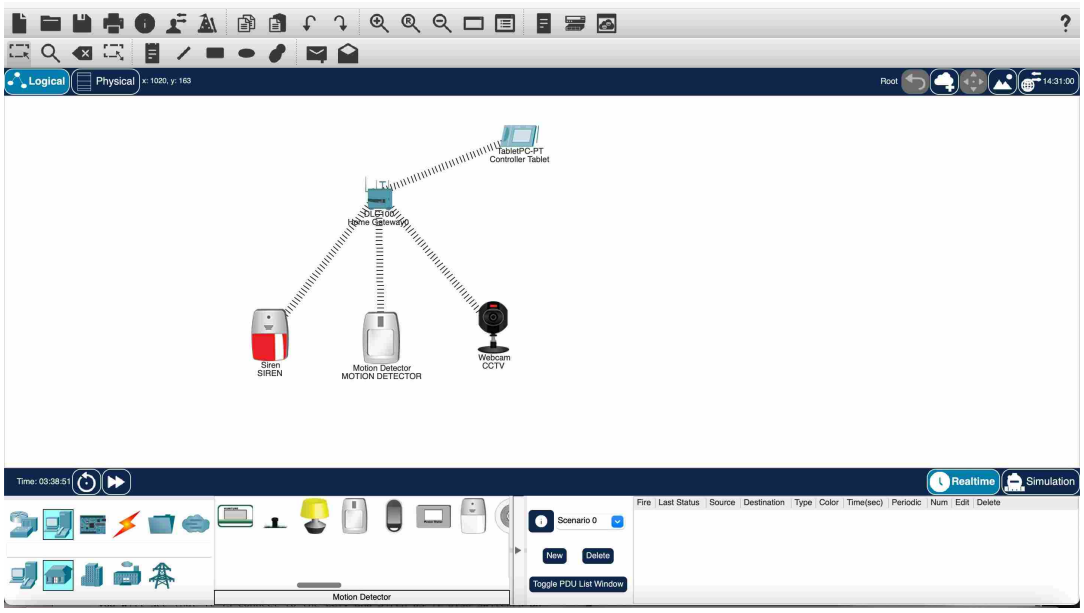


Figure 2.15: Siren and CCTV turning on when motion is detected.

— Why Create a Rule?.

- **Automated Security:** By configuring conditions and actions, you simulate a basic home or office security system where motion automatically triggers alarms and camera recording.
- **Easy Management:** Using *IoT Monitor* rules is more efficient than writing custom scripts to link devices, making it beginner-friendly for rapid prototyping of IoT interactions.

G. Testing the Motion Detector Rules

Observation: After a few seconds, the Motion Detector automatically resets to an `off` state, yet the Siren and CCTV remain `on`. We need a second rule that deactivates them when motion is no longer detected.

20. **Add a "No Movement" Rule:**
    - On the Tablet, open **IOT Monitor** and navigate to the **Conditions** tab again.
    - Click **Add Rule** and name it `NOMOVEMENT`.
21. **Condition:**
    - Set *Device* to "`motion detector`."
    - Under *State*, choose "`on` = `false`."

    This captures the scenario when the motion detector returns to an `off` state (no movement).
22. **Actions:**
    - Set `SIREN` on = `false`.
    - Click **Add Action** to also set `CCTV` on = `false`.
    - Click **Connect** (or **Save**) to finalize the rule.

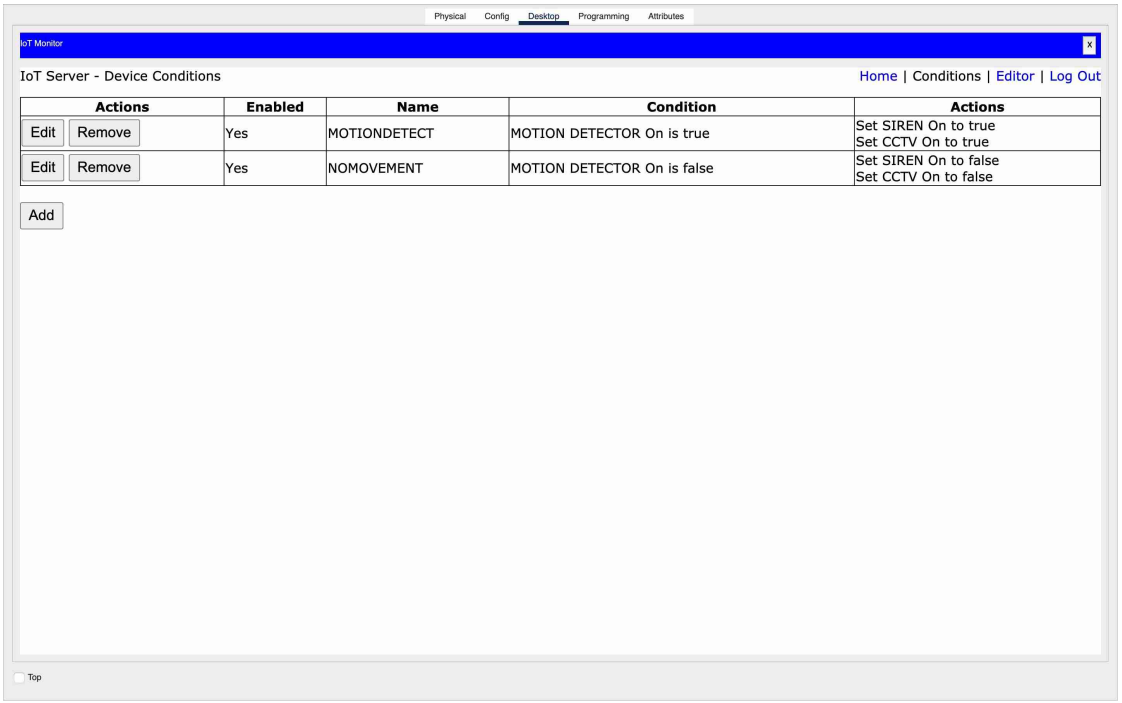


Figure 2.16: Creating the “no movement” condition to turn devices off.

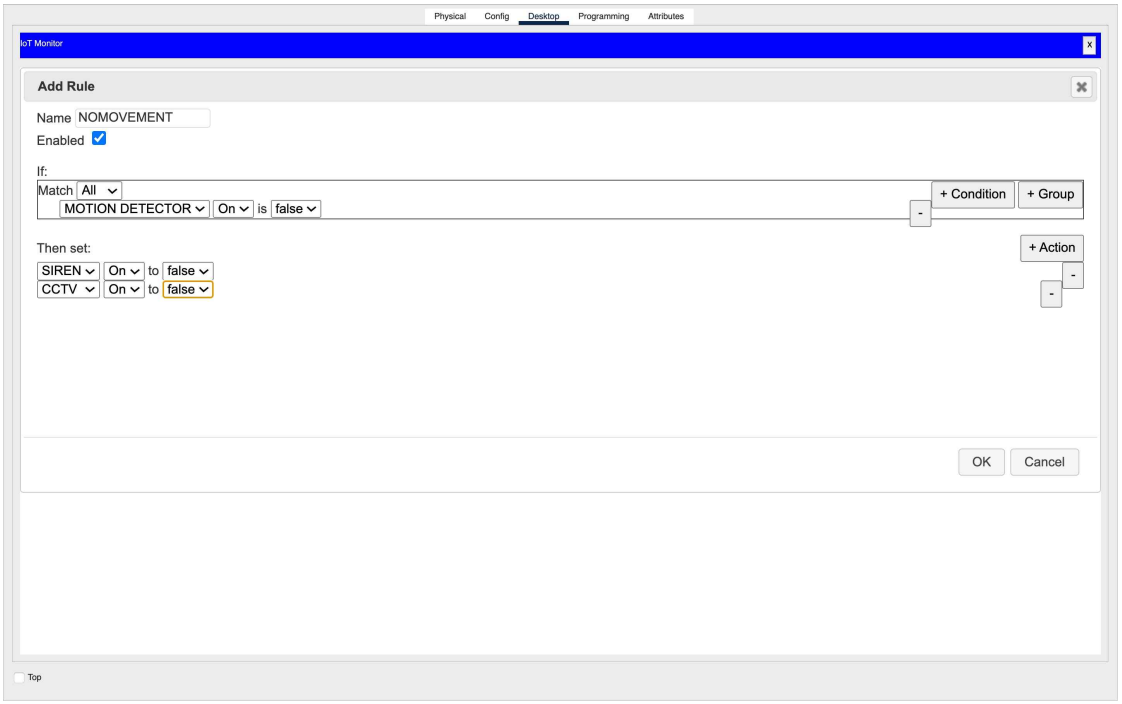


Figure 2.17: Both CCTV and Siren set to “off” when no motion is detected.

23. **Final Test:**

- Exit or minimize the Tablet window.
- Once again, hold **Alt/Option** and hover over the Motion Detector to simulate movement.

- Verify that the CCTV and Siren turn `on` when motion is detected.
- Wait a few seconds for the motion to reset, and confirm that the CCTV and Siren automatically switch `off`.

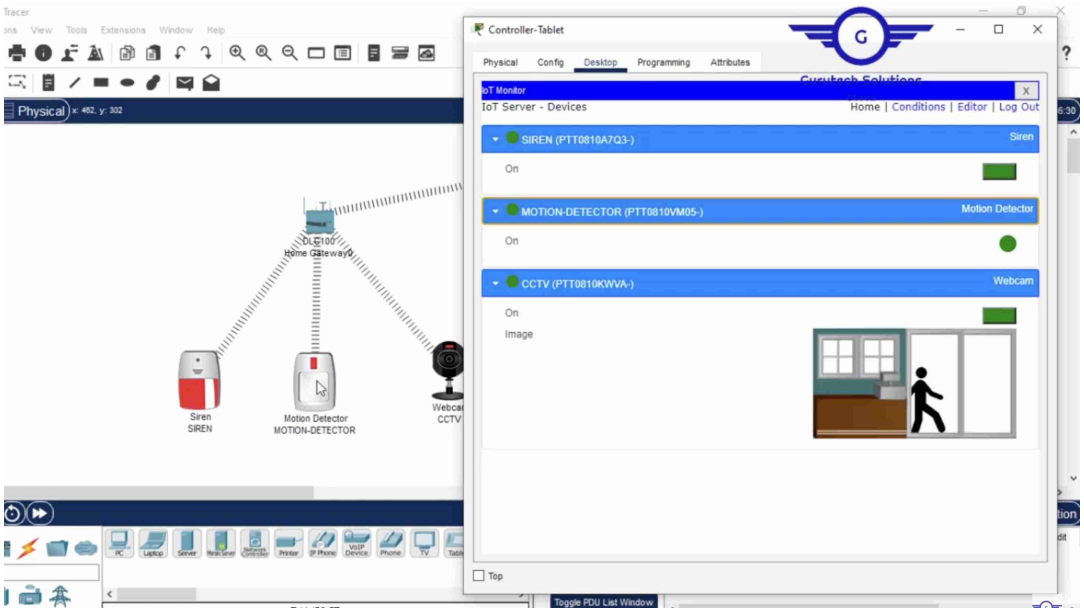


Figure 2.18: CCTV and Siren on while the motion sensor is active.

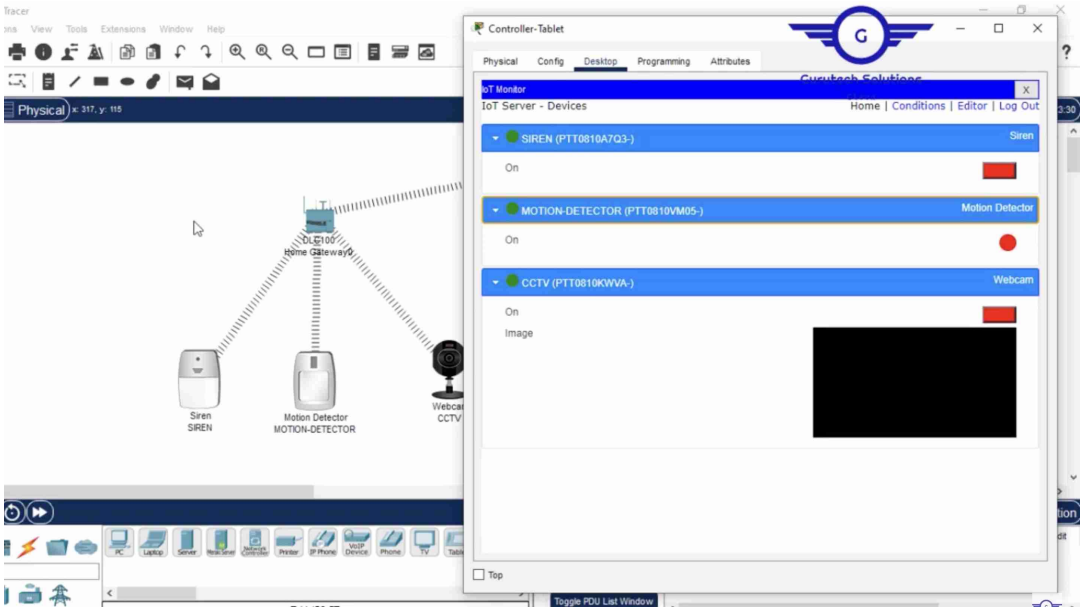


Figure 2.19: Devices turning off after motion ceases.

— Why Two Separate Rules?.

- **Positive Trigger:** The `MOTIONDETECT` rule handles what happens *when* motion is detected (`on = true`).
- **Negative Trigger:** The `NOMOVEMENT` rule handles what happens *when* motion stops (`on = false`), ensuring devices don't stay on indefinitely.
- **Realistic Behavior:** This mimics how a real motion sensor system automatically resets to

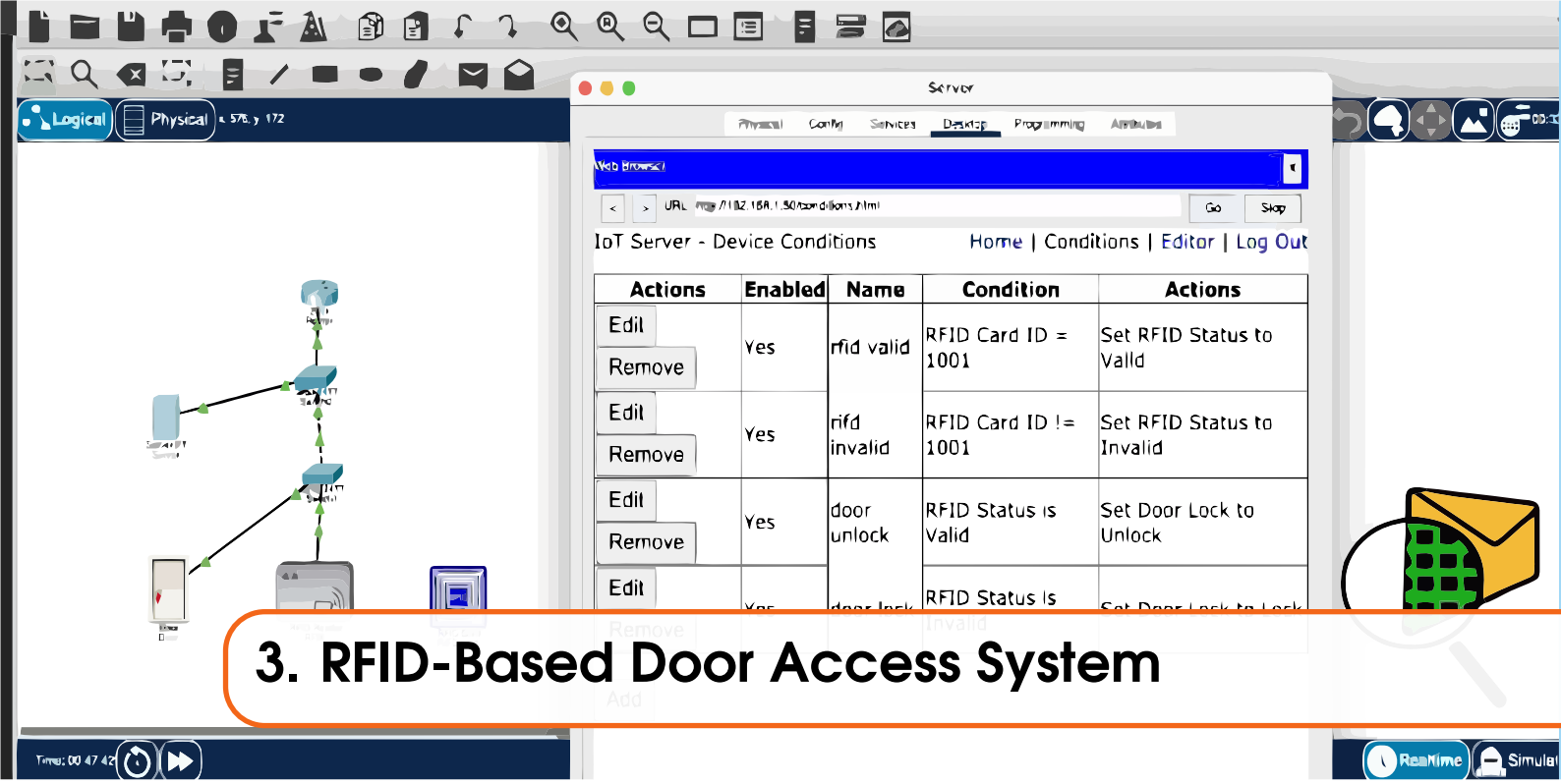
`off`, deactivating alarms and cameras after a period of no movement.

Measuring Success

- **Device Registration:** Each IoT device (Siren, Motion Detector, Webcam) appears in the Home Gateway list, confirming a correct "Home Gateway" registration.
- **Motion Activation:** Holding **Alt** over the motion detector triggers the CCTV and Siren to turn on—verifying the "MOTIONDETECT" condition rule works.
- **Automatic Reset:** After motion stops, the "NOMOVEMENT" rule turns off both CCTV and Siren, indicating a complete on/off cycle is functional.
- **Tablet Monitoring:** The Tablet's IOT Monitor displays all devices with accurate statuses, ensuring you can view and manage them in real time.

Summary

Congratulations! You have successfully built and configured an IoT-based intruder detection system in Cisco Packet Tracer. Through device registration, condition-based rules, and practical testing, you demonstrated how motion sensors can seamlessly trigger CCTV cameras and sirens—forming a functional security setup.



3. RFID-Based Door Access System

Introduction

In this lab, you will learn how to **build and configure a network to control door access using an RFID reader**. You will assign IP addresses, set up a DHCP pool for IoT devices, and register an RFID door system with a central server. By creating conditions based on RFID card IDs, you will control the state of the door (locked or unlocked), providing a simple yet effective example of access control in a Packet Tracer IoT environment.

Objective

- **Build and configure** a network to control door access using an RFID reader.
- **Set up communication** between devices using IP addressing and DHCP.
- **Program and test** device conditions to manage access control based on RFID card ID.

Lab Plan

A. **Launch Packet Tracer and Observe the Initial Workspace**
B. **Build the Topology**
C. **Connect Devices and Prepare the Door**
D. **Configure IP Addresses and DHCP on the Router**
E. **Set IP Addresses for the Server**
F. **Enable IoT Services on the Server**
G. **Create a Server Account (Username and Password)**
H. **Register and Connect IoT Devices to the Server**
I. **Configure RFID-Based Door Access Conditions**

Required Software

- **Cisco Packet Tracer 8.x** (or newer)
- Familiarity with Packet Tracer's *Programming* or *Config* tabs
- Optional: external PC browser for more advanced testing

A. Launch Packet Tracer and Observe the Initial Workspace

1. **Open Packet Tracer:**
   Double-click the Packet Tracer icon on your desktop or locate it in your applications folder to launch. You should see a blank default Logical topology workspace, as shown in Figure 5.1.

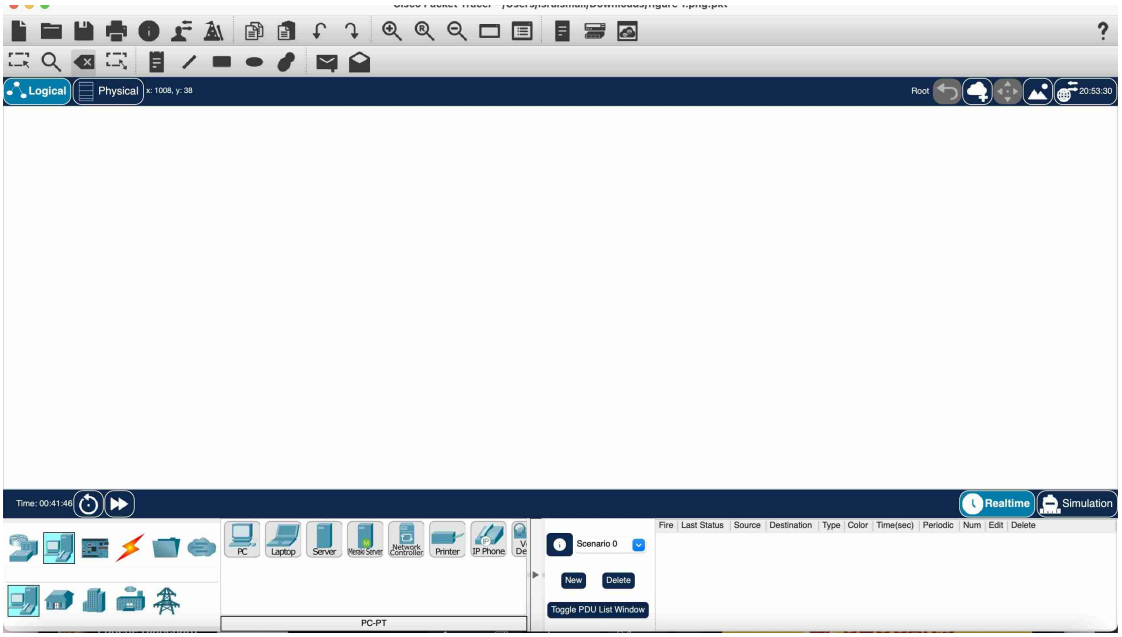


Figure 3.1: Initial Workspace

B. Build the Topology

2. **Add Network Devices:**
   In Packet Tracer, locate the *Device-Type Selection* box in the lower-left corner. You will see a row of **categories** (e.g., *Network Devices*, *End Devices*) and a row of **subcategories** (e.g., *Routers*, *Switches*, *Wireless Devices*). Drag each of the following onto your workspace:
   - **Router** (e.g., *Router0*)
   - **Switches** (e.g., *Switch0*, *Switch1*, as needed)
   - **RFID Reader**
   - **Door**
   - **Server**

   Arrange the devices to reflect the topology you plan to build. For instance, place the router at the center if it's a primary gateway, switches in between for LAN connections, and the RFID reader/door where you can simulate secure access. Figure 3.2 offers an example layout with these devices configured in a network.

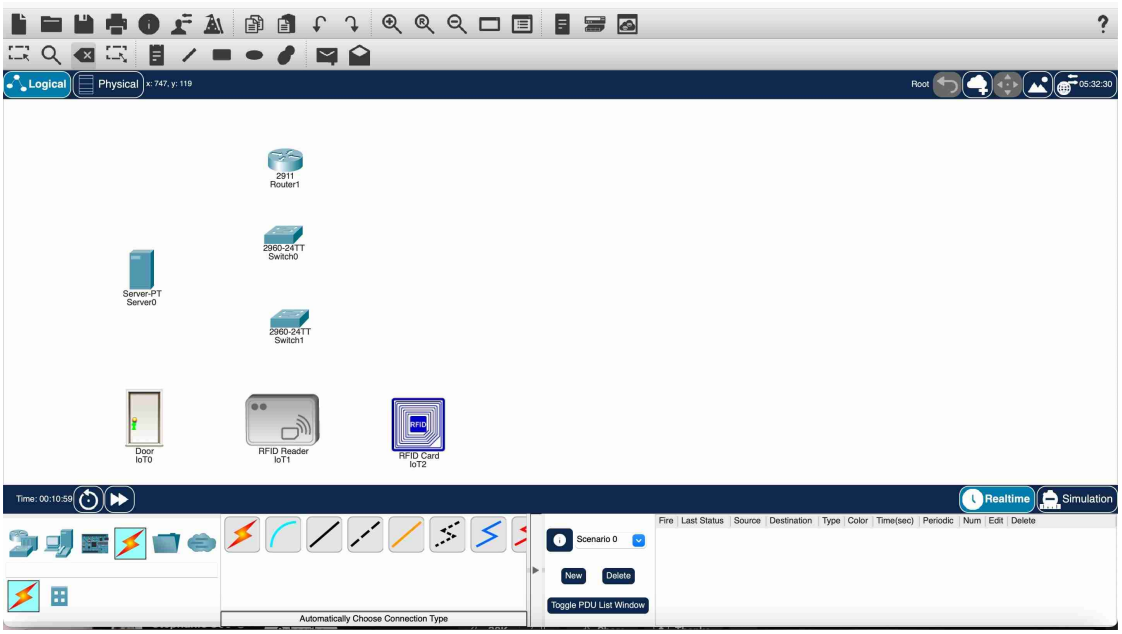


Figure 3.2: Sample topology featuring a router, switches, an RFID reader, a door, and a server.

— Why these devices?.
- **Router:** Central device for managing traffic between different subnets or connecting to external networks.
- **Switches:** Provide LAN connectivity, allowing multiple end devices to communicate within the network.
- **RFID Reader & Door:** Simulate an access control system, where authorized tags unlock the door.
- **Server:** Hosts essential services (e.g., authentication, database, or web services) that integrate with the RFID reader and door for access control logic.

Placing these components together lets you explore how an RFID access system communicates and enforces security within a networked environment.

C. Connect Devices and Prepare the Door

Goal: Ensure all devices (**Router**, **Switches**, **RFID Reader**, **Door**, **Server**) are cabled correctly.

3. **Use the Connections Icon:**
   In Packet Tracer's toolbar (usually on the lower-left side), click the *Connections* icon, which appears as a **lightning bolt**. From the pop-up list, choose **Automatically Choose Connection Type**. This cable tool automatically selects the most appropriate cable (e.g., straight-through or cross-over) based on the devices you are connecting.
4. **Connect Router and Switches:**
   - With the cable tool active, click on the router, then click on **Switch0**.
   - Repeat the process to connect **Switch0** to **Switch1**.
   - After a moment, you should see link lights turn **green** on each connection if the devices are powered on.
5. **Connect Switch1 to RFID Reader and Door:**
   - Attempt to connect **Switch1** to the **RFID Reader** directly with the cable tool.
   - If the cable *does not* attach to the **Door**:

(a) Click on the door device, then select **Advanced**.
(b) In the **I/O Config** tab, change the network adapter to `PT-IOT-NM-1CFE`.
(c) Once changed, use the cable tool again to connect **Switch1** to the door.

- Wait a few seconds for the link lights or dotted lines to appear, indicating an active connection.

6. **Connect Switch0 to the Server:**
Finally, use the same cable tool to link **Switch0** to the **Server**. This ensures the server can communicate with both switches and other devices in the topology.

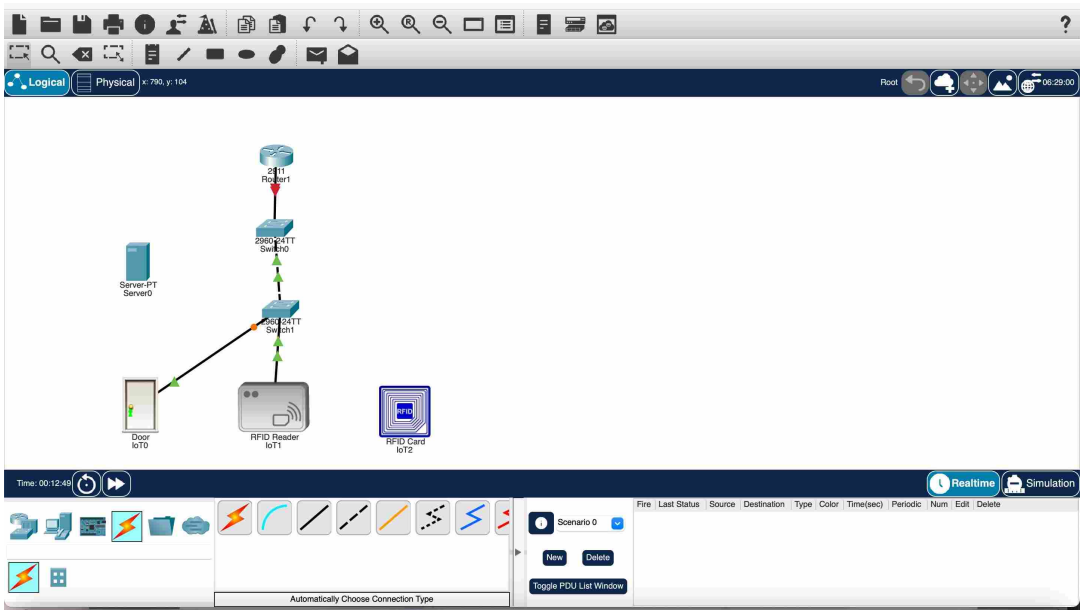


Figure 3.3: Partial topology showing the door (gray icon) and RFID reader connected to Switch1.

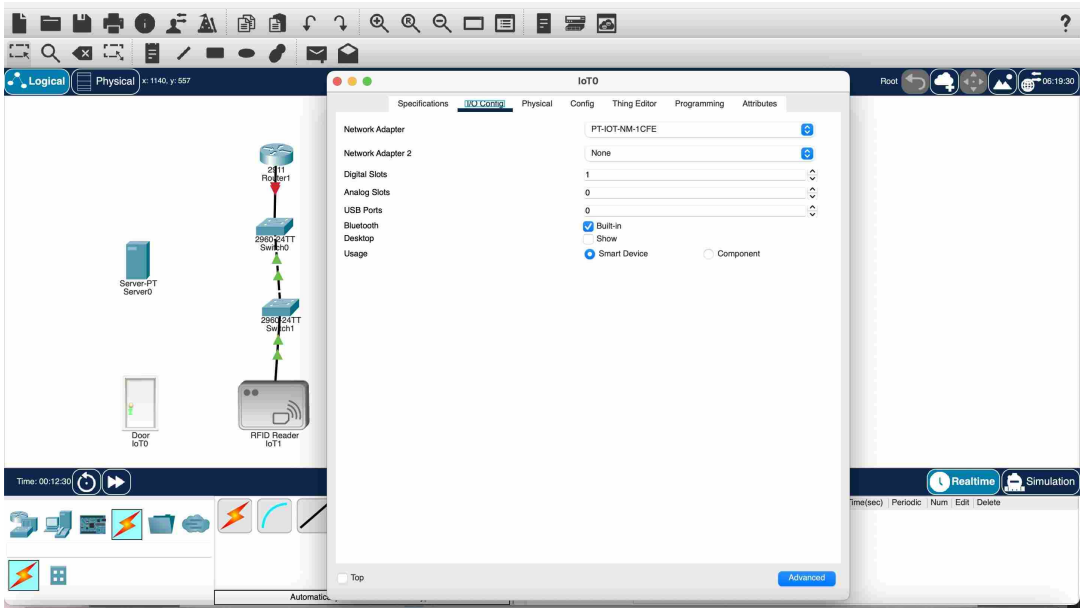


Figure 3.4: Accessing the door's *Advanced* settings to modify the network adapter.

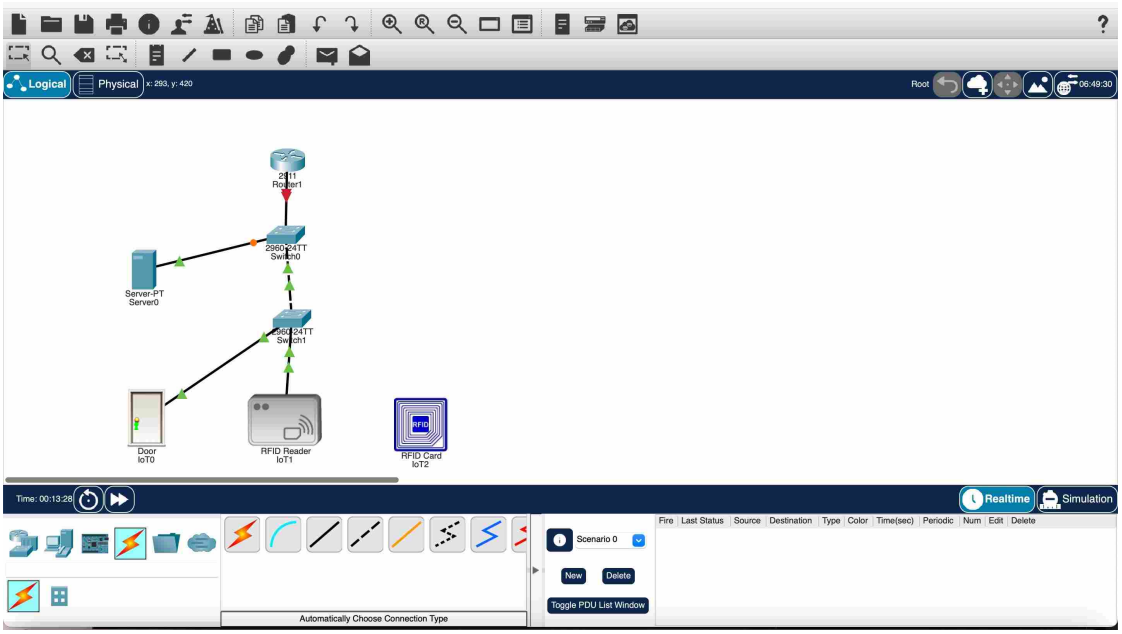


Figure 3.5: Ensuring the door uses the `PT-IOT-NM-1CFE` network adapter.

— Troubleshooting Cable Connections.

- **No Link Lights?**
  - Check the device power status or port enable status under the *Config* or *Physical* tab.
  - Make sure you've used the correct adapter module for IoT devices like the door.
- **Wrong Cable Type?**
  - If you selected a *Manual* cable type (e.g., *Copper Straight-Through*), double-check that it's correct for Router-to-Switch or Switch-to-Switch links.
  - The *Automatically Choose Connection Type* option usually picks the right cable based on the devices.
- **Door Not Showing Port?**
  - Always verify the `PT-IOT-NM-1CFE` or equivalent network interface module is installed under the **Advanced** → **I/O Config** tab.
  - After installing the correct module, power the device off and on (if needed) to ensure the port becomes active.

D. Configure IP Addresses and DHCP on the Router

Goal: Automatically assign IP addresses to IoT devices via DHCP.

7. **Open the Router CLI:**

   - Click the **Router** in the workspace.
   - Choose the **CLI** tab. If prompted, press `Enter` to begin.

   You will see a command-line interface (CLI) similar to a real Cisco router.
8. **Enter Commands:**
   At the router prompt, type the following commands:

```
enable
conf terminal
hostname R1
interface gigabitEthernet 0/0
 ip address 192.168.1.1 255.255.255.0
```

```
 no shutdown
exit

ip dhcp pool iot
 network 192.168.1.0 255.255.255.0
exit
```

— Command Explanation:.

- `enable`: Switches from user mode to privileged mode on the router.
- `conf terminal`: Enters global configuration mode.
- `hostname R1`: Renames the router to `R1`.
- `interface gigabitEthernet 0/0`: Selects the primary interface to configure (may vary by router model).
- `ip address 192.168.1.1 255.255.255.0`: Assigns an IP (`192.168.1.1`) and subnet mask to `GigabitEthernet 0/0`.
- `no shutdown`: Activates the interface (removes default shutdown state).
- `exit`: Returns to global config mode.
- `ip dhcp pool iot`: Creates a DHCP pool named `iot`.
- `network 192.168.1.0 255.255.255.0`: Defines the subnet from which IP addresses are assigned.
- `exit`: Returns to privileged EXEC mode.

9. **DHCP Pool Created:**
   This configures a DHCP pool named `iot` that assigns IP addresses within the `192.168.1.0/24` subnet. Any device requesting an IP via DHCP on this interface will automatically be assigned an address, preventing manual configuration errors and address conflicts.

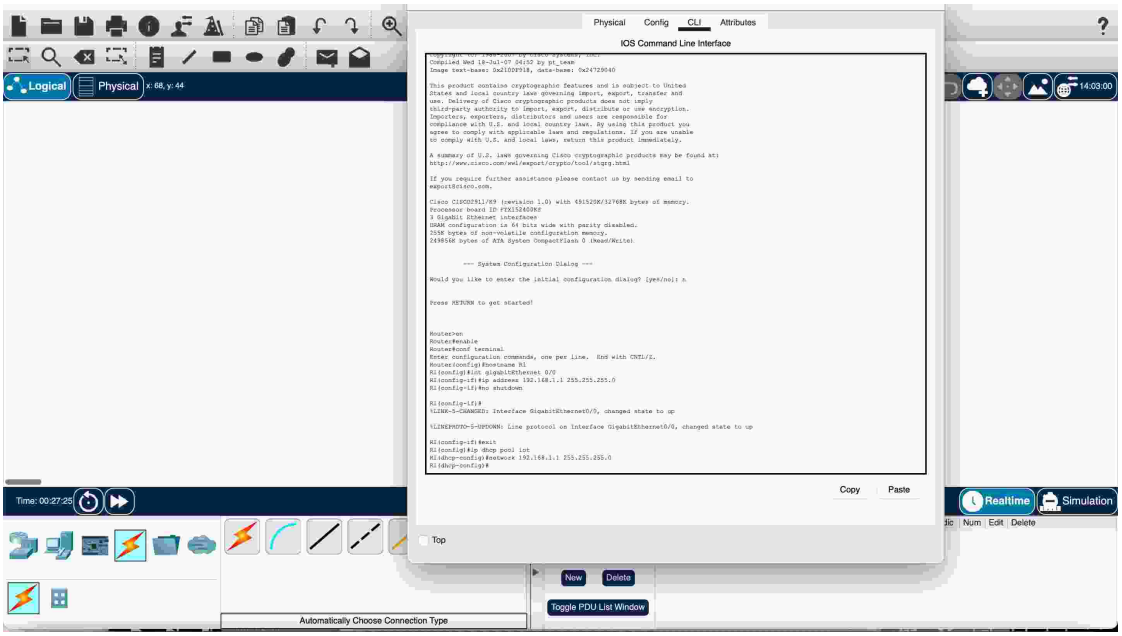


Figure 3.6: Using the router CLI to configure IP addressing and a DHCP pool named “iot.”

— Why Use DHCP on the Router?.

- **Central Management:** You can manage IP addresses from a single point, making it easy to add or remove devices without manual IP

changes.

- **Fewer Conflicts:** Automatically assigning addresses reduces the risk of two devices accidentally sharing the same IP.
- **Scalability:** In larger or busier networks, relying on DHCP can save considerable time and administrative overhead.

E. Set IP Addresses for the Server

10. **Open Server Desktop:**
    - Click on the **Server** device in your Packet Tracer workspace.
    - Select the *Desktop* tab, then choose *IP Configuration*.

    This interface allows you to configure the server's IP settings, much like configuring a real-world server's network adapter.
11. **Assign IP:**
    In the *IP Configuration* window, set:
    - **IPv4 Address:** `192.168.1.50`
    - **Subnet Mask:** `255.255.255.0`
    - **Default Gateway:** `192.168.1.1`

    These values place the server in the same `192.168.1.x` subnet as the router and other network devices.
12. **Confirm and Close:**
    After entering these settings, simply close the configuration window. The **static IP address** ensures the server remains reachable at `192.168.1.50`, making it easy to host IoT services and preventing automatic changes from DHCP.

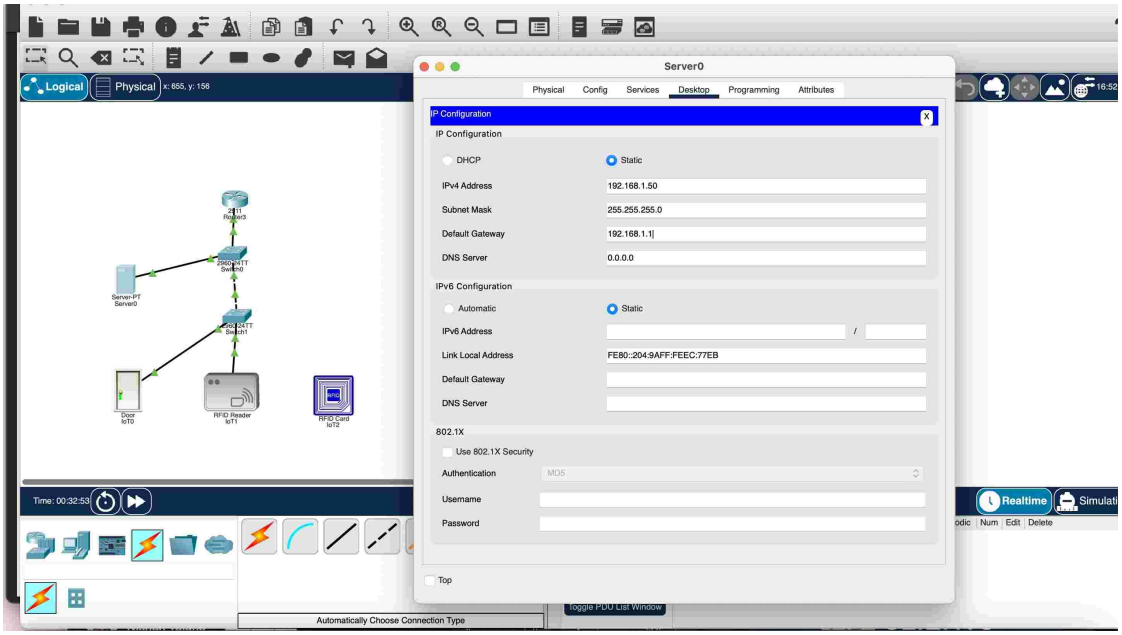


Figure 3.7: Assigning `192.168.1.50` to the server so it can host IoT services.

— Why Use a Static IP for Servers?.

- **Consistent Addressing:** Client devices (and network admins) always know where the server can be reached, important for DNS, HTTP, or other services.

- **Easier Troubleshooting:** A stable IP simplifies pings and route tracing, helping you quickly verify server availability.
- **No DHCP Reliance:** In case DHCP fails or is misconfigured, the server remains unaffected.

F. Enable IoT Services on the Server

13. **Switch to Services:**

    - With the **Server** window still open, select the *Services* tab (usually found at the top).
    - The left-hand panel will list various services (e.g., DHCP, DNS, IoT, HTTP, etc.).

14. **Turn On IoT:**

    - In the left panel, click on **IoT**.
    - Locate the toggle or button to enable the IoT service.
    - Click the button so it reads **On**.

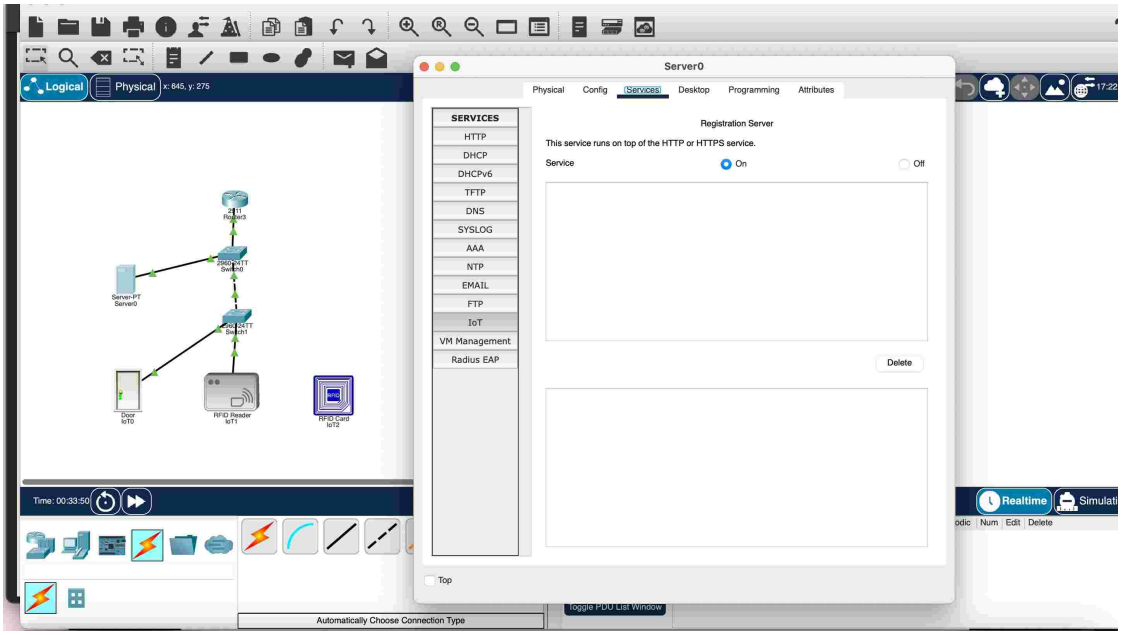


Figure 3.8: Enabling the server's built-in IoT services.

— Why Enable IoT Services?.

- **Device Registration:** When IoT services are active, networked devices such as doors, sensors, or RFID readers can register with this server and appear in its device list.
- **Central Management:** It provides a hub for managing IoT rules, conditions, or triggers across multiple devices—similar to a home automation controller.
- **Scalability:** As your lab grows, you can integrate more IoT devices (cameras, sensors, etc.) under the same server, simplifying configuration and control.

G. Create a Server Account (Username and Password)

15. **Open Web Browser:**

    - While still on the **Server** device, switch to the *Desktop* tab.

- Click on *Web Browser*.

This will open a built-in browser window, enabling you to access the server's local web interface.

16. **Enter the Server IP:** `192.168.1.50`.
    In the browser's address bar, type `192.168.1.50` and press **Enter**. You should see the server's default web interface load.

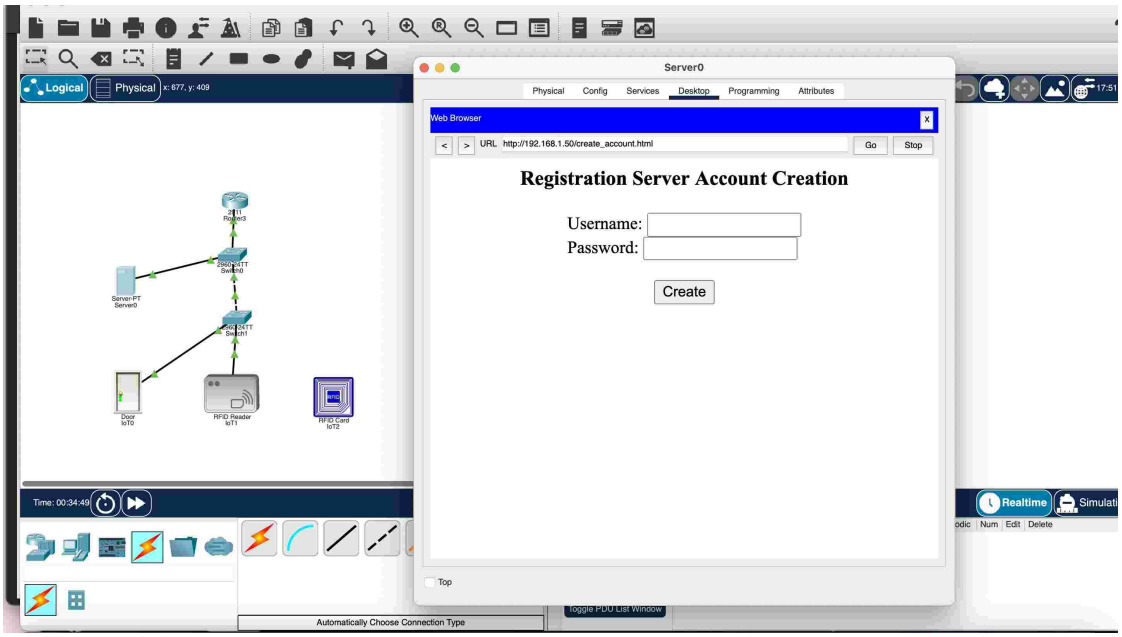


Figure 3.9: Accessing the server's web interface via `192.168.1.50`.

17. **Sign Up:**

- Look for a link or button labeled *"sign up now"* on the login page.
- Provide a username and password (for instance, `admin / admin`) in the corresponding fields.

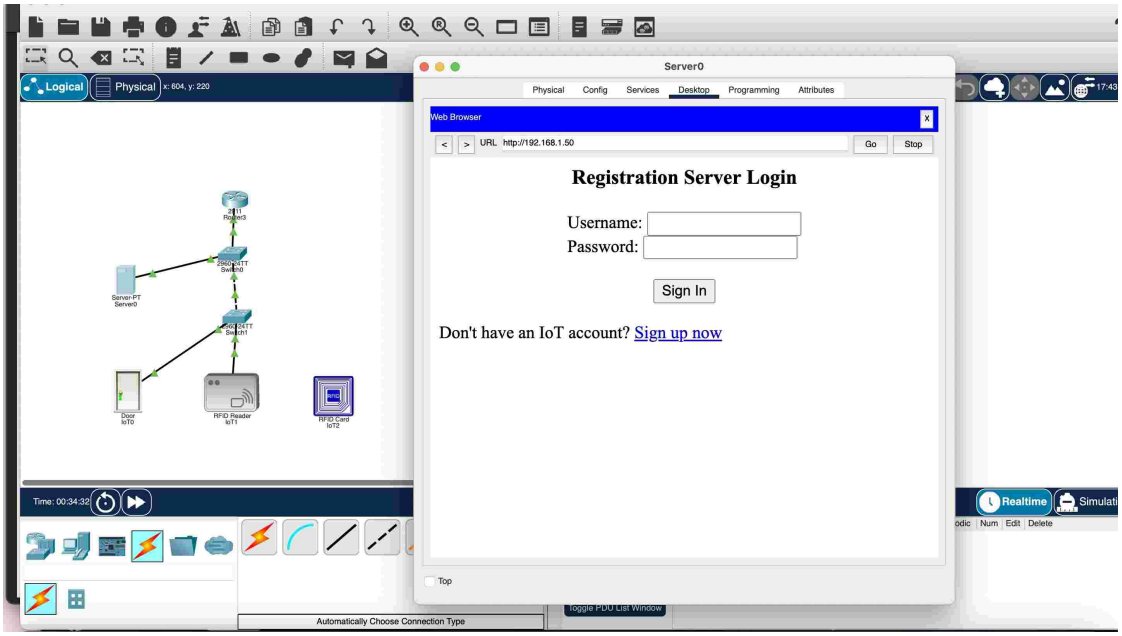


Figure 3.10: Signing up for a new account on the server.

18. **Complete Registration:**

    - Click *Create* (or *Submit*) to finalize your new user account.
    - Once created, you will be logged in or prompted to log in using the new credentials.

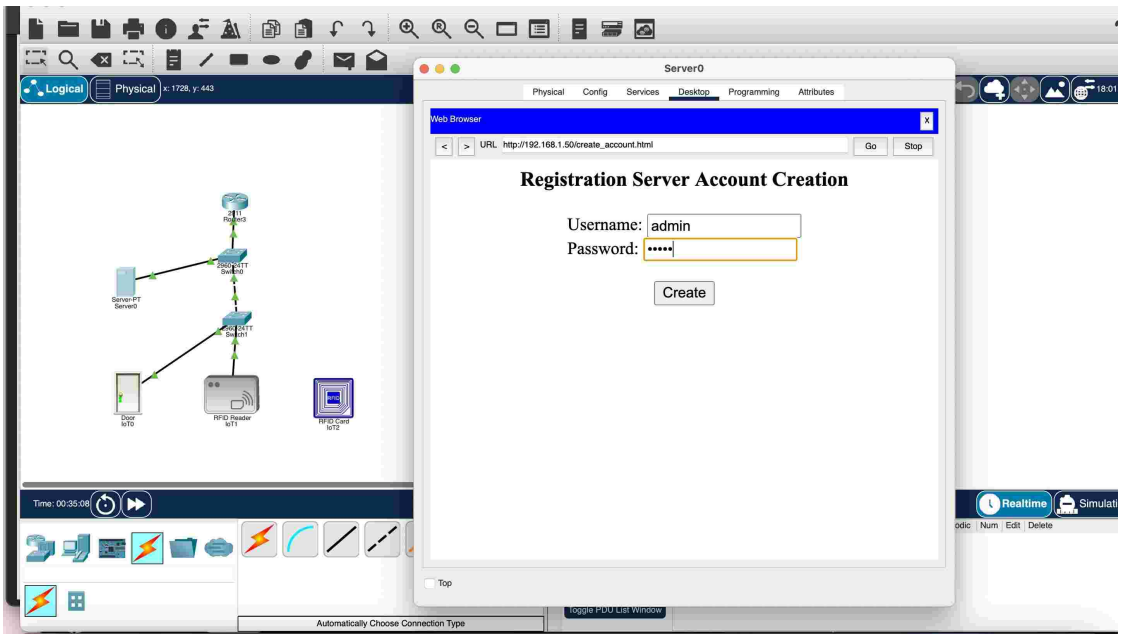


Figure 3.11: Specifying username and password (e.g., “admin/admin”).

— Why Create a User Account?.

- **IoT Registration:** This account allows you to register IoT devices under the server’s management, so they appear in the device list.
- **Security and Customization:** Each user account can have distinct access privileges or

configurations, helpful for managing multiple users or administrative roles.
- **Simplicity for Labs:** Using `admin / admin` is fine for classroom or lab scenarios, but remember to use stronger credentials in real-world setups.

H. Register and Connect IoT Devices to the Server

19. **Door Config:**

    - Click the **Door** device, then select the *Config* tab.
    - Under **Gateway DNS IPv4**, choose `DHCP` to automatically obtain an IP address from the router's DHCP pool.
    - Switch to the **Remote Server** section and enter:
      - **Server Address:** `192.168.1.50`
      - **User Name:** `admin`
      - **Password:** `admin`
    - Click **Connect** to link the door to the server.

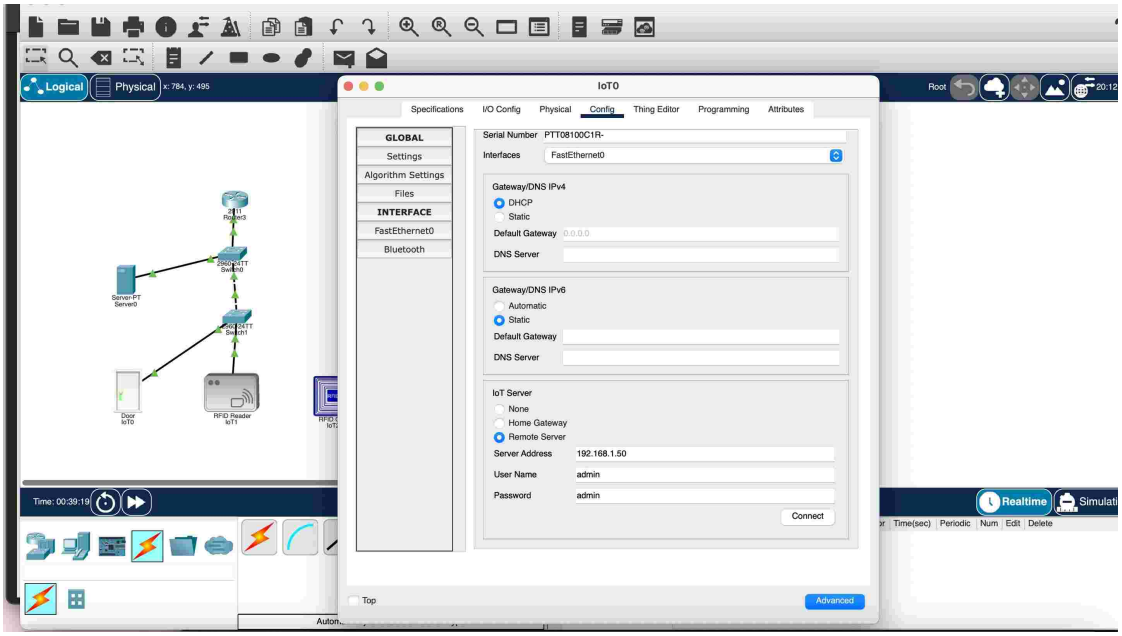


Figure 3.12: Door set to DHCP and pointing to the server at `192.168.1.50`.

20. **RFID Reader Config:**

    - Repeat the same steps: under **Config** → **Settings**, select `DHCP` for IP assignment.
    - Then, in **Remote Server**:
      - **Server Address:** `192.168.1.50`
      - **User Name:** `admin`
      - **Password:** `admin`
    - Click **Connect** to finalize registration.

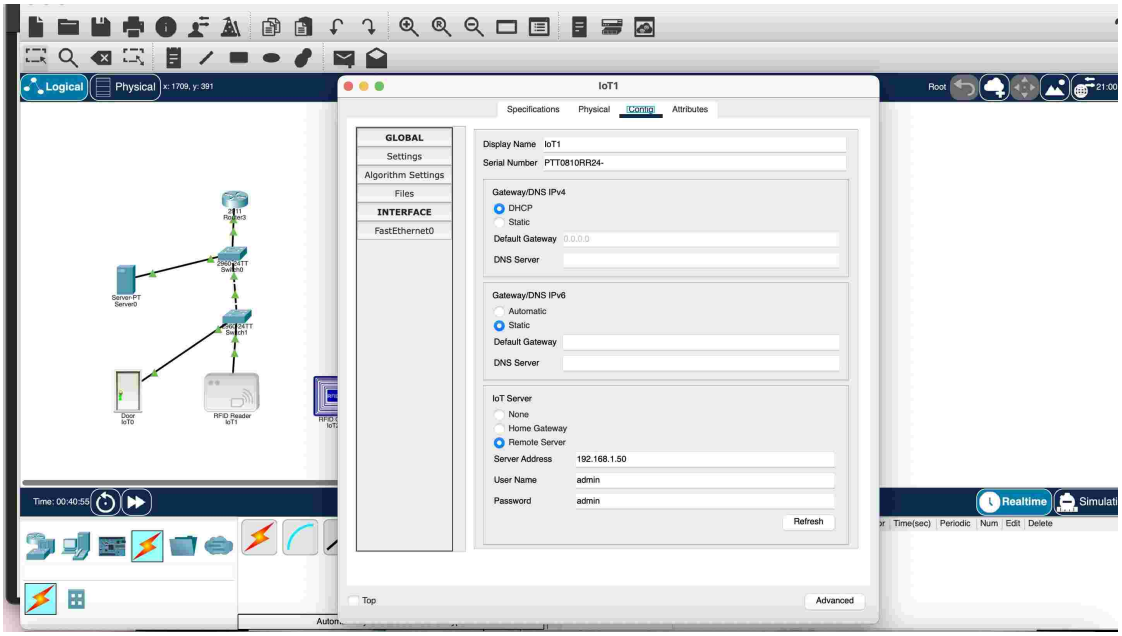


Figure 3.13: RFID reader also connecting remotely to the server.

21. **Verify in IoT Monitor:**

- On the **Server**'s *Desktop* tab, select *IoT Monitor*.
- Log in with the credentials (`admin/admin`) you created earlier.
- Confirm that both the **Door** and the **RFID Reader** appear in the list of connected devices.

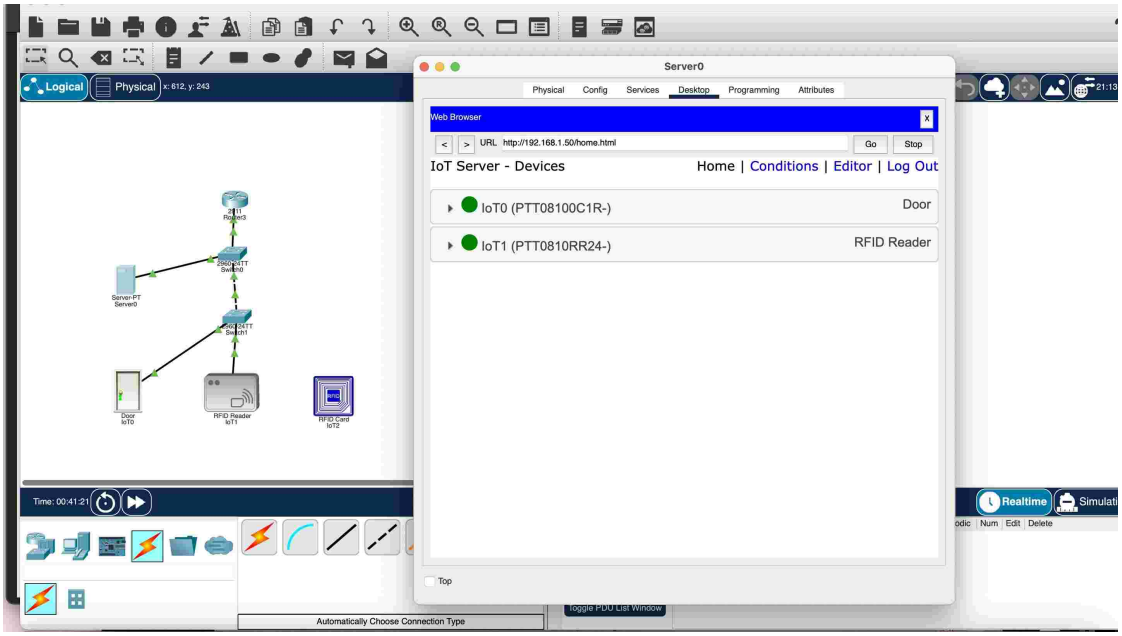


Figure 3.14: IoT Monitor view showing door and RFID status.

22. **Test Basic Functionality:**

- From the IoT Monitor, try locking or unlocking the door to verify the server can control it.
- Alternatively, you can click the door directly in the workspace to ensure changes reflect back in the IoT Monitor.

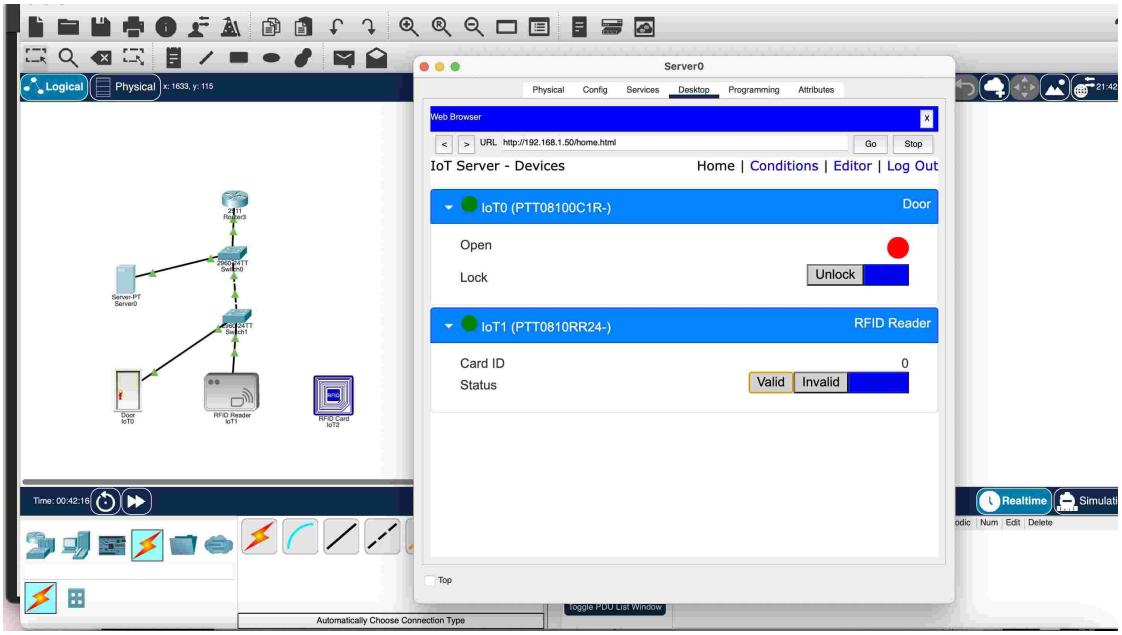


Figure 3.15: Lock/unlock tests confirming the door responds to server commands.

— Why Register IoT Devices with the Server?.

- **Central Control:** Once registered, the server can monitor and send commands to each IoT device (e.g., lock/unlock the door, read RFID events).
- **Automation & Rules:** You can create rules (for example, *if RFID tag is valid, unlock door*) to automate the system.
- **Easier Troubleshooting:** The IoT Monitor on the server provides a consolidated view of device statuses and connections.

I. Configure RFID-Based Door Access Conditions

Goal: Control the door lock based on the RFID card ID.

23. **Open the Server's Web Browser:**

    - Return to the **Server** device, switch to its *Desktop* tab, and launch the *Web Browser*.
    - In the server's web interface, look for the **Conditions** section (or a link labeled "Conditions") to begin setting up your rules.

24. **Add Condition for RFID Card ID = 1001:**

    - Under **Add Condition**, specify:
      - **RFID Card ID**: equals `1001`
      - **Action**: `RFID Status = Valid`
    - This creates a rule stating that whenever the card ID is `1001`, the RFID status should be marked `Valid`.

25. **Add Condition for RFID Card ID != 1001:**

- Similarly, create a second condition:
  - **RFID Card ID**: not equal to `1001`
  - **Action**: `RFID Status` = `Invalid`
- This covers any card ID that differs from `1001`, automatically marking the RFID status `Invalid`.

26. **If RFID Status = Valid:**

- Add or edit an action so that whenever `RFID Status` equals `Valid`, the **Door Lock** is set to `Unlock`.
- This ensures the door opens if the RFID tag is recognized as authorized.

27. **If RFID Status = Invalid:**

- Add another action specifying that if `RFID Status` is `Invalid`, then the **Door Lock** remains `Lock`.
- This keeps the door locked for all unauthorized or unknown RFID IDs.

28. **Save and Exit:**
The final rule set should resemble Figure 3.16, where **Valid** IDs unlock the door, and **Invalid** IDs keep it locked.

Figure 3.16: Condition-based rules controlling the door lock via RFID status.

How it works: If the RFID card ID is `1001`, it's marked `Valid`, unlocking the door. Any other ID is `Invalid`, keeping the door locked.

Tips for Testing RFID Door Conditions:

- **CardID Property:** Click the RFID tag in the workspace, go to *Attributes*, and edit the `CardID`. This lets you simulate both authorized and unauthorized tags.
- **Visual Confirmation:** Some versions of Packet Tracer show a green icon or light if the door unlocks, and a red indicator if the ID is rejected. Observe the door to confirm the lock status changes as expected.

29. **Test the Setup:**

    - Hover the RFID card (ID = `1001`) over the **RFID Reader**.
    - You should see a **green light** (or similar indicator) and the door should unlock.
    - If you change the `CardID` to something else (e.g., `500`), you'll see a **red** indicator and the door will remain locked.

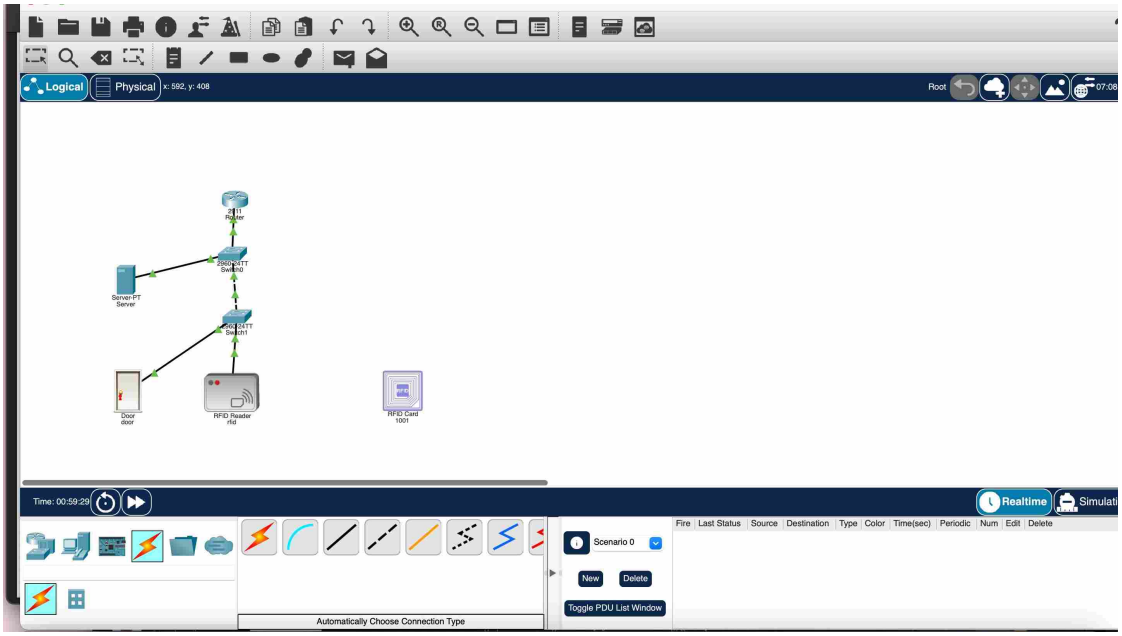


Figure 3.17: Invalid card scenario, red light if mismatch (door remains locked).

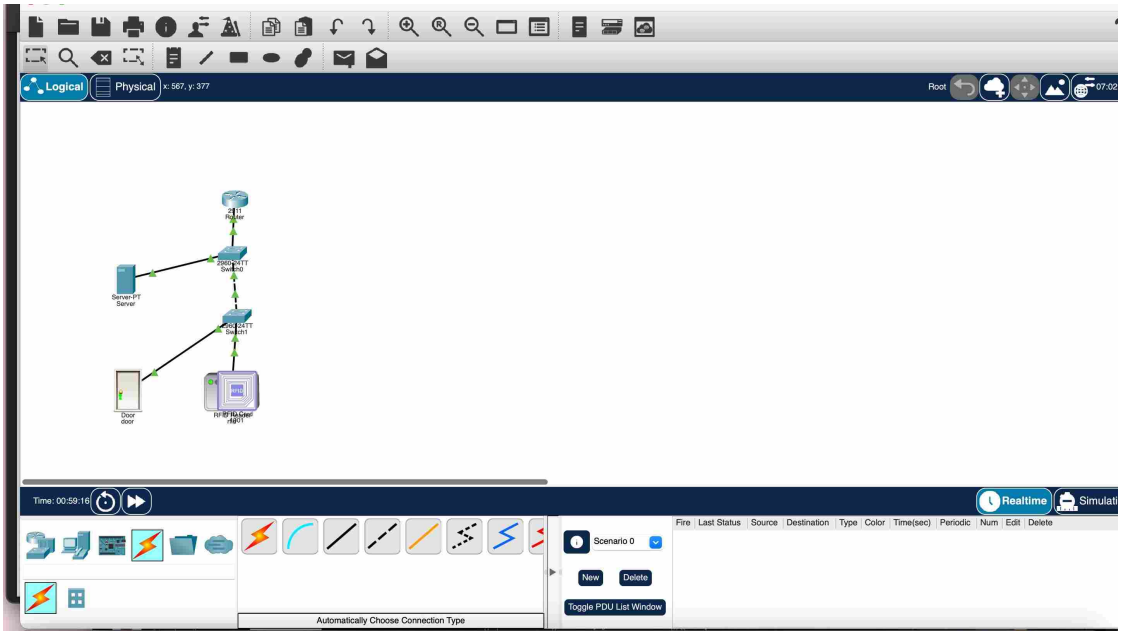


Figure 3.18: Valid card scenario, green light with door unlocked.

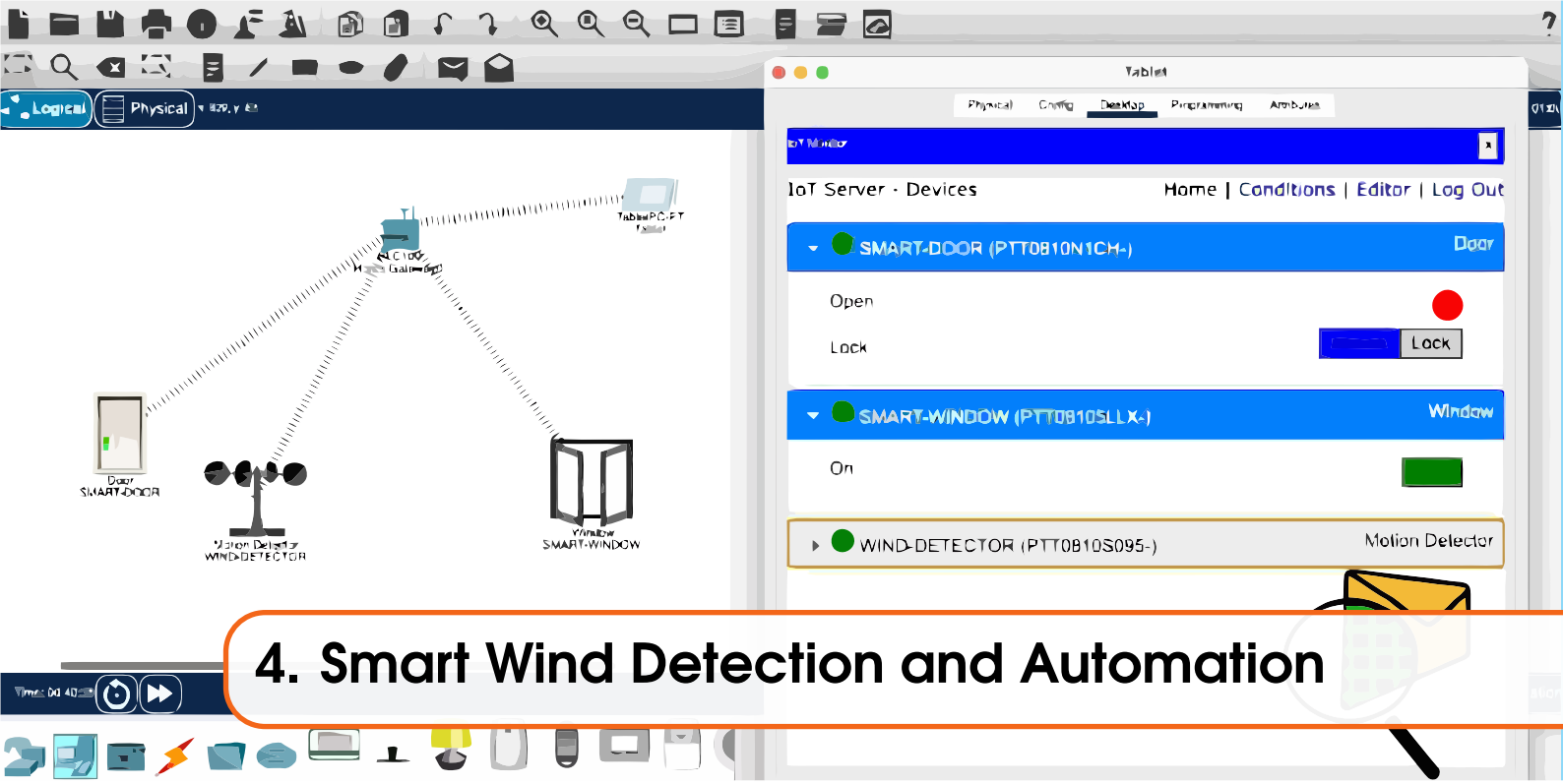
Measuring Success

- **DHCP Setup on Router:** Door and RFID reader successfully obtain IP addresses from `192.168.1.0/24`.
- **IoT Registration:** The door and RFID reader appear in the server's IoT Monitor, verifying that they've connected.
- **Card Validation:** Presenting the correct card ID (`1001`) unlocks the door; any other ID keeps it locked.
- **Real-time Status Updates:** The IoT Monitor or visual icons (lights) show correct lock/unlock states, confirming that conditions are triggered properly.

■

Summary

Congratulations! You have successfully **configured a server to control door access using an RFID reader** in Cisco Packet Tracer. By assigning IP addresses (with DHCP), creating a user account on the server, and setting conditions for valid vs. invalid RFID card IDs, you achieved a functioning access control system. This approach can be expanded or integrated with additional IoT devices to form a more comprehensive security or smart-building scenario.



4. Smart Wind Detection and Automation

Introduction

This lab will guide you through creating a small **smart home system** in Cisco Packet Tracer using a *Wind Detector*, *Smart Door*, and *Smart Window*. You will establish wireless connections to a Home Gateway, configure devices with appropriate IP addresses, and set up automated rules that lock a door or close a window when high wind is detected.

Objectives

- **Simulate and configure** IoT devices: Smart Door, Smart Window, Wind Detector.
- **Establish wireless communication** between IoT devices and the Home Gateway.
- **Create automated conditions** (rules) for device actions based on wind detection.
- **Test and verify** the entire setup via the Packet Tracer interface and Tablet.

Lab Plan

A. **Launch Packet Tracer and Observe Initial Workspace**
B. **Build the Topology**
C. **Configure the Tablet (Wireless Settings)**
D. **Assign IP Addresses to IoT Devices**
E. **Register Devices with the Home Gateway**
F. **Control IoT Devices through the Tablet**
G. **Modify Wind Detector Icons**
H. **Add Wind Detection Conditions**
I. **Test Wind Detector Functionality**

Required Software

- **Cisco Packet Tracer 8.x** (or newer)
- Familiarity with *Config* and *Advanced* tabs in Packet Tracer IoT devices
- Optional: external PC browser (if you want to test beyond Packet Tracer's built-in tools)

A. Launch Packet Tracer and Observe Initial Workspace

1. **Open Packet Tracer:**
   Double-click the Packet Tracer icon (or navigate to its directory) to launch. You should see a blank default Logical topology workspace.

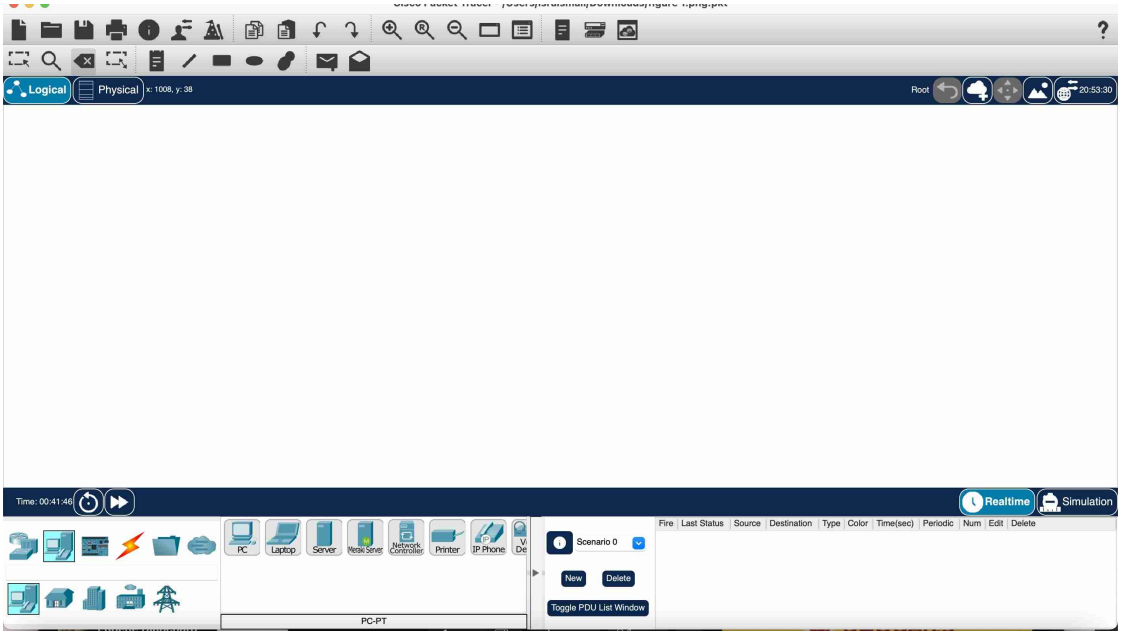


Figure 4.1: Initial Workspace (blank topology).

B. Build the Topology

2. **Add Network Devices:**
   Open the *Device Selection* box in the lower-left corner of Packet Tracer. You will see a row of **categories** (e.g., *Network Devices*, *End Devices*, *Components*) and a row of **subcategories** (e.g., *Home Gateways*, *Smart Devices*, *IoT*). Drag each of the following onto your workspace:
   - **Home Gateway**
   - **Tablet**
   - **Smart Door**
   - **Smart Window**
   - **Wind Detector**

   Arrange them to reflect the scenario you plan to create. For instance, place the *Home Gateway* centrally so all IoT devices can connect to it wirelessly or via a wired link, and position your *Tablet* as the user control interface.

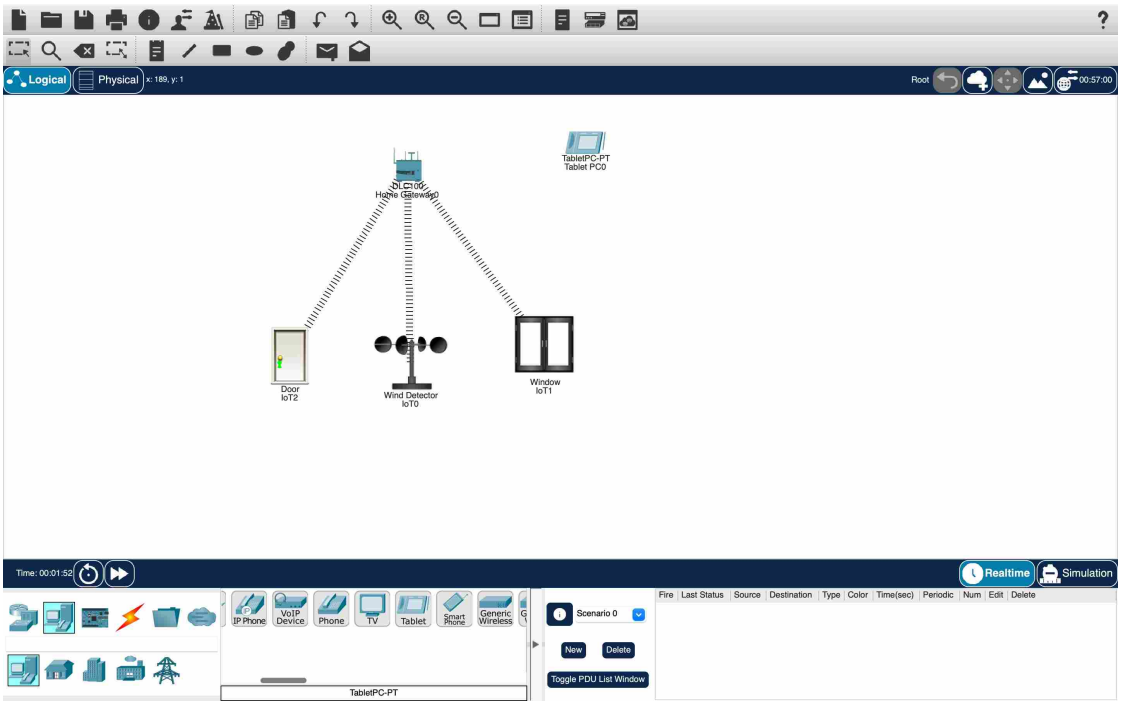


Figure 4.2: Sample topology with IoT devices and a Home Gateway.

— Why these devices?.

- **Home Gateway:** Acts as the central hub for IoT device registration and management, often providing wireless access.
- **Tablet:** Serves as a user interface to monitor and configure IoT devices (e.g., checking sensor status, controlling doors/windows).
- **Smart Door, Smart Window, Wind Detector:** Represent common IoT elements in a smart home. The door and window can be opened/closed via conditions, while the wind detector senses changes in wind speed or direction.

Together, they create a realistic small-scale smart home environment in Packet Tracer.

C. Configure the Tablet (Wireless Settings)

3. **Select the Tablet and Config Tab:**

   - Click on the **Tablet** device in the Logical workspace.
   - At the top of the Tablet's configuration window, choose the *Config* tab.

   This tab provides a simplified interface for modifying the Tablet's hardware and network parameters.
4. **Wireless Configuration:**

   - In the left pane, select **Wireless0**.
   - Change the **SSID** from `"Default"` to `"HomeGateway"` (see Figure 4.3).

   This ensures the Tablet connects to the Home Gateway's wireless network rather than any default or unrelated SSID.

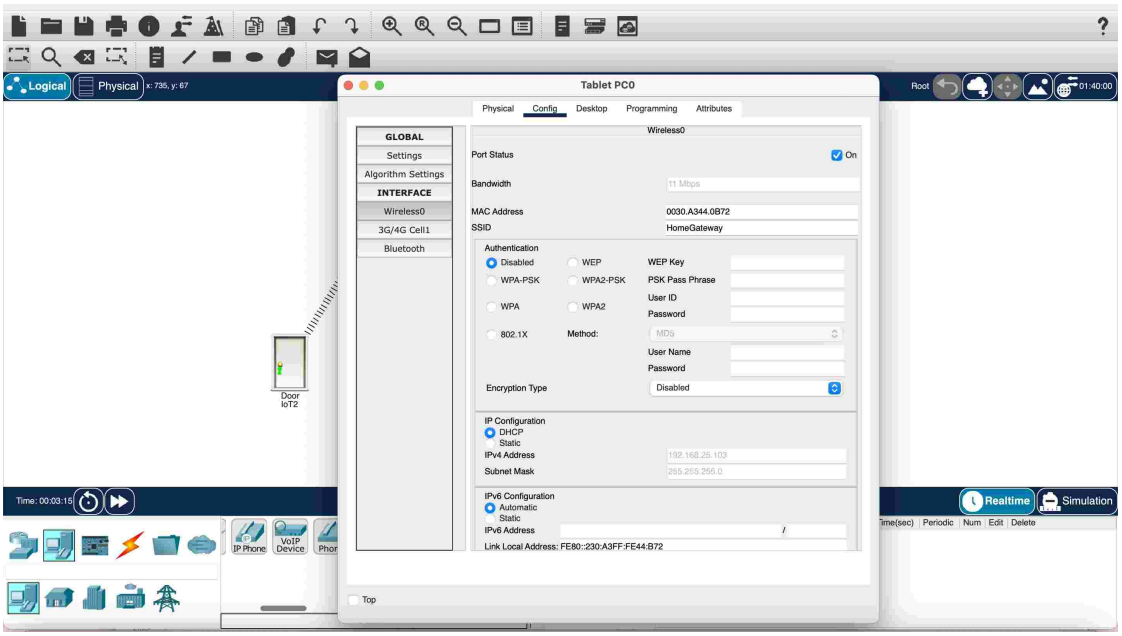


Figure 4.3: Wireless Settings on the Tablet.

5. **Verify the Connection:**

   - After setting the SSID, the Tablet should automatically form a wireless link to the **Home Gateway**.
   - Look for a dotted or curved line connecting the Tablet to the Gateway in the workspace (Figure 4.4).

   If you hover your cursor over the Tablet or Gateway icon, you may see status details (e.g., signal strength, connected SSID) confirming the active connection.

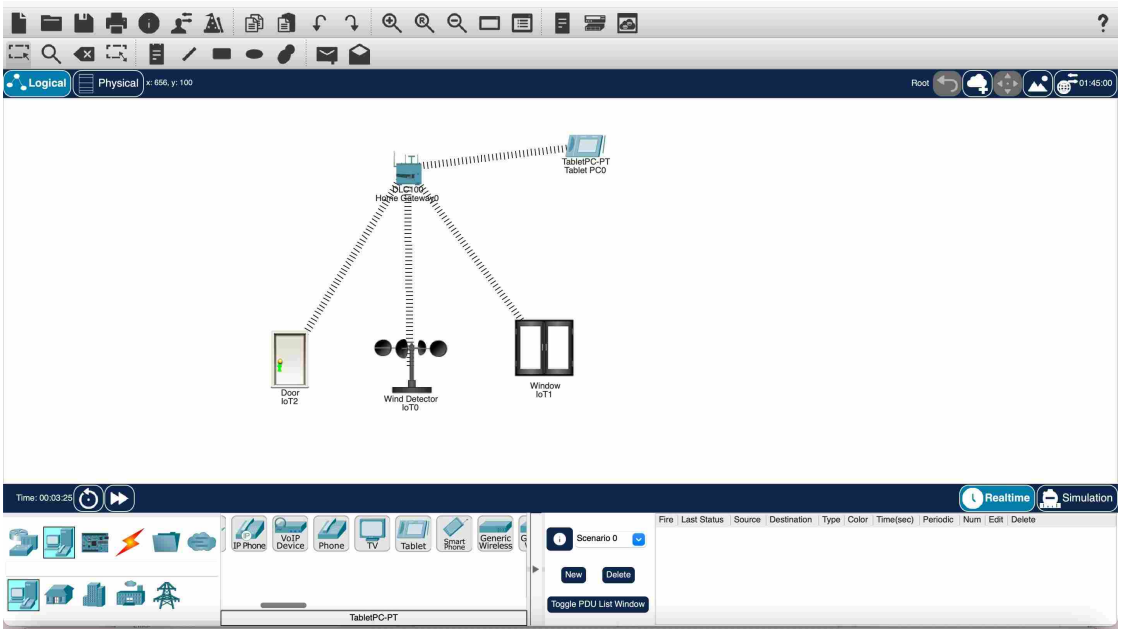


Figure 4.4: Wireless connection established between Tablet and Gateway.

— Why rename the SSID?.

- **Clarity in Multi-Wireless Environments:** Using a unique SSID like `HomeGateway` helps distinguish this network from any other wireless networks in Packet Tracer.
- **Consistency with Other Devices:** All IoT devices that need to connect wirelessly will reference `HomeGateway` in their configuration, making it easier to manage and troubleshoot.

D. Assign IP Addresses to IoT Devices

6. **Check Device IPs:**
   In the Packet Tracer workspace, hover your mouse cursor over each IoT device (see Figure 4.5) to see if it already has an IP address assigned. A small tooltip usually appears, listing the device's **IP Address**, **Subnet Mask**, **Gateway**, and more.

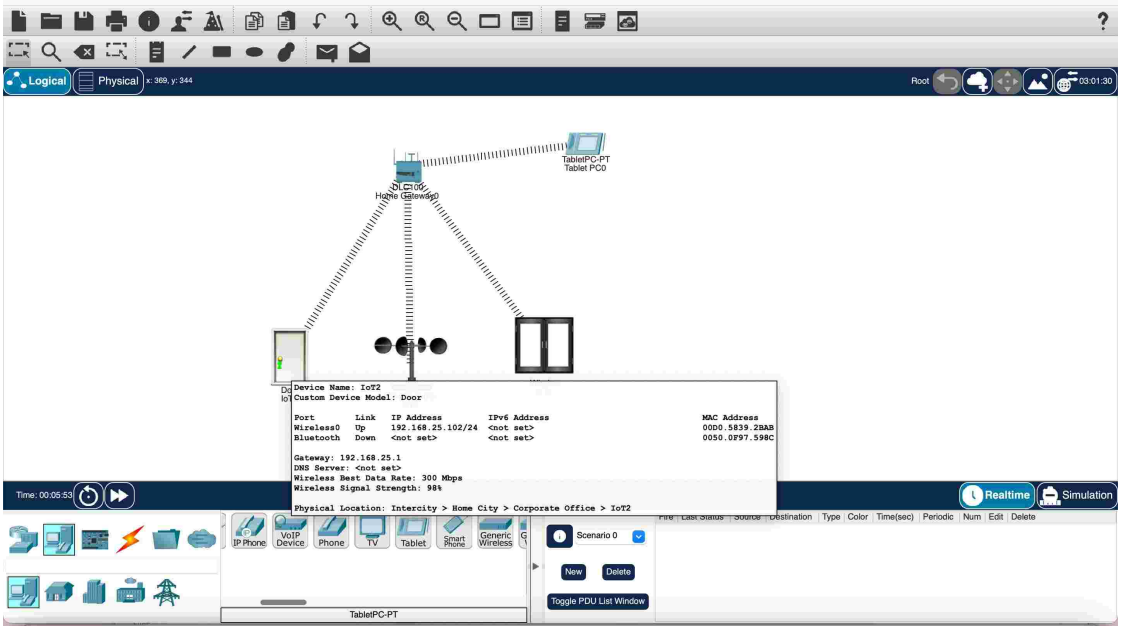


Figure 4.5: Hovering over devices to check IP addresses.

— Why hover for IPs?.

- **Quick Verification:** Without opening each device's *Config* tab, you can rapidly confirm which devices still need an IP.
- **Network Snapshot:** If you see `0.0.0.0` or `169.x.x.x`, it indicates the device hasn't properly obtained an IP (or is using a default/link-local address).

7. **Use DHCP for Each Device:**
   If any device lacks a valid IP address:
   (a) **Click** the device in the workspace.
   (b) Go to **Config** → **Gateway/DNS IPv4**.
   (c) Temporarily switch from `DHCP` to `Static` (if needed), then switch back to `DHCP`.
   This process re-requests a dynamic address from the **Home Gateway**'s DHCP service.

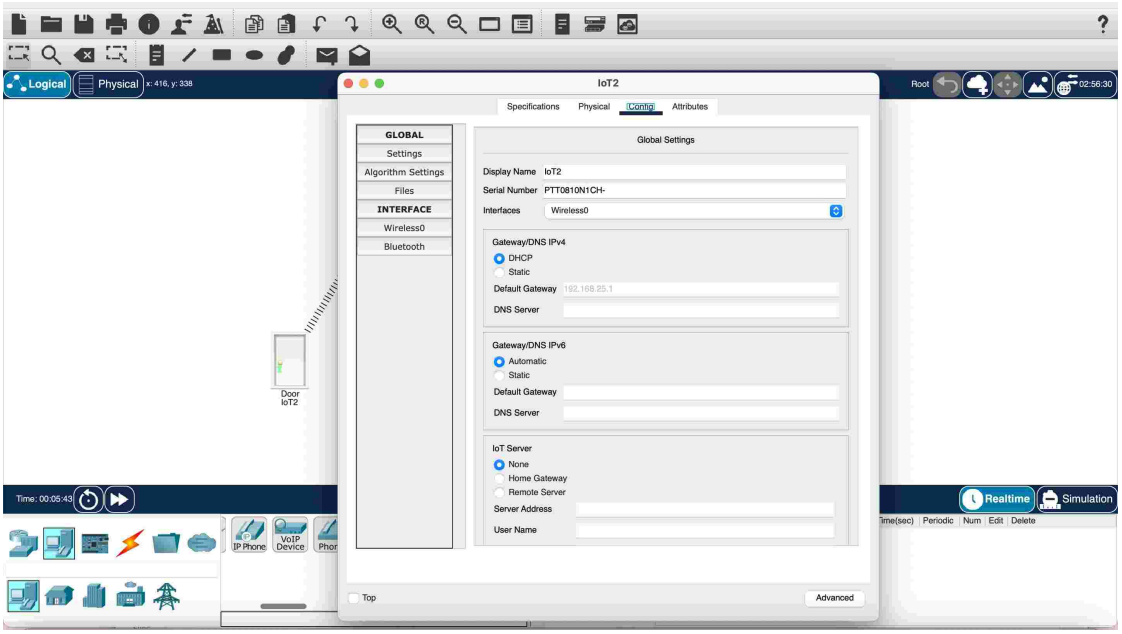


Figure 4.6: Assigning an IP address via DHCP.

— Why switch from Static to DHCP?.

- **Forcing a Refresh:** Packet Tracer devices sometimes need a "nudge" to re-check DHCP. Moving to `Static` and then back to `DHCP` triggers a new request to the gateway.
- **Ensuring Consistent Settings:** This guarantees each IoT device is on the same IP subnet as the **Home Gateway**, simplifying communication and control.

E. Register Devices with the Home Gateway

8. **Door Registration:**

   - Click the **Door** device in the workspace.
   - Go to **Config** → **Settings**.
   - Under **IoT Server** (sometimes labeled "Server" or "IoT Connection"), select **Home Gateway**.

   This tells the Door to register itself with the **Home Gateway**, enabling remote control and monitoring.

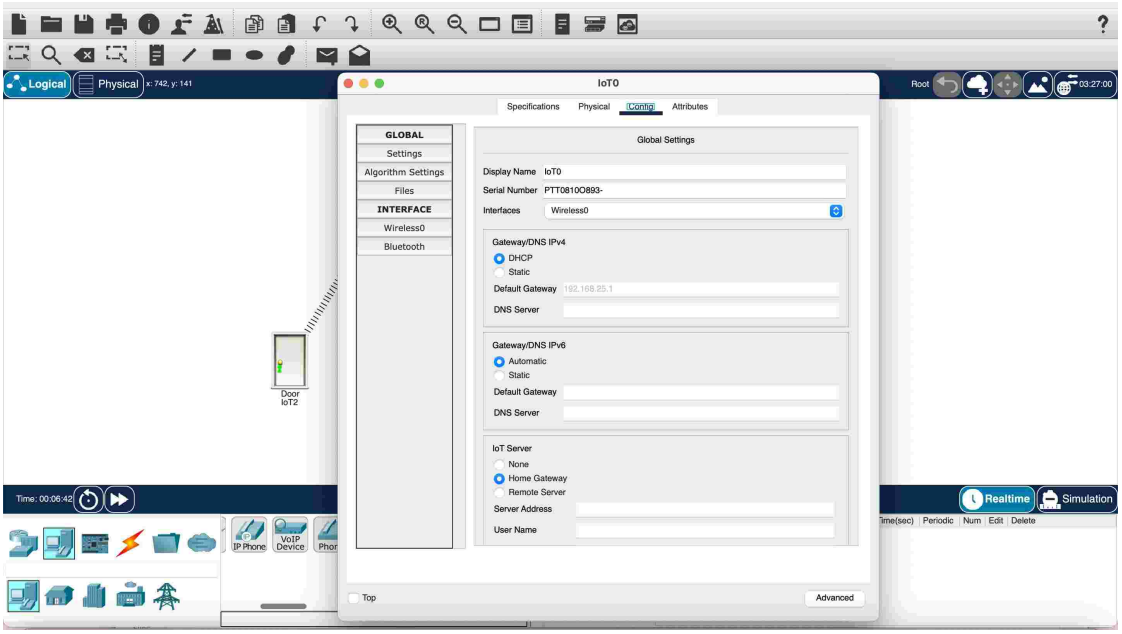


Figure 4.7: Registering a device (e.g., Door) with the Home Gateway.

9. **Wind Detector Registration:**

   - Click the **Wind Detector** device.
   - Navigate to *Config* → **Settings**.
   - Under **IoT Server**, select **Home Gateway**.

   This step ensures the Wind Detector is visible to the Home Gateway and can send wind status updates.

10. **Window Registration:**

    - Click the **Window** device.
    - Go to *Config* → **Settings**.
    - Choose **Home Gateway** under IoT Server.

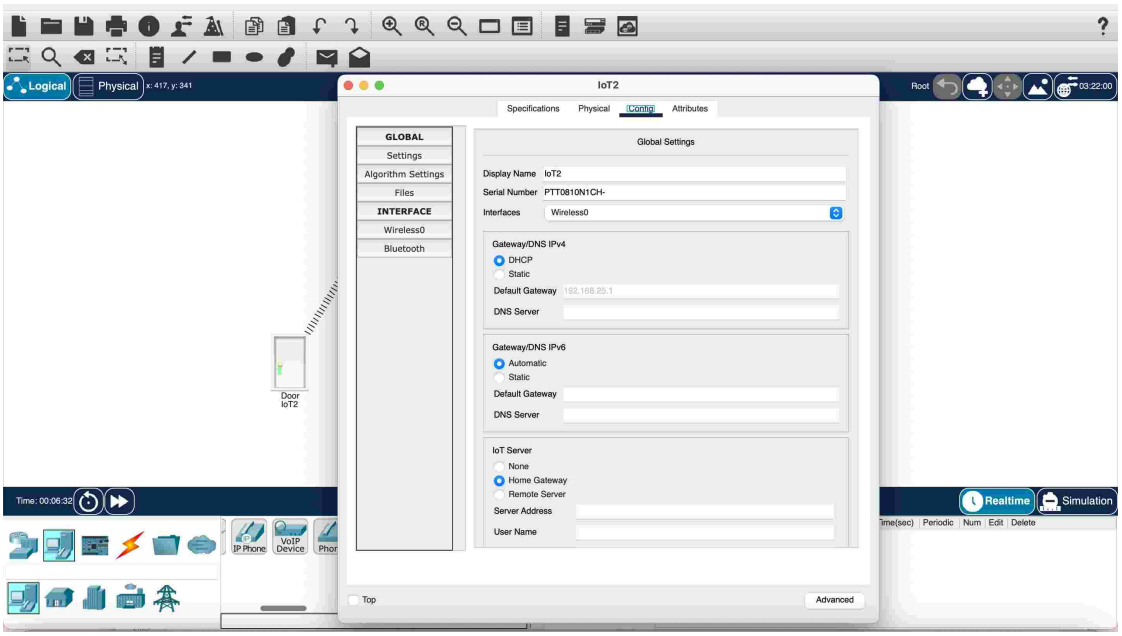


Figure 4.8: Registering multiple IoT devices (Window, Wind Detector, etc.) to the Home Gateway server.

— Why Register These Devices?.

- **Centralized Control:** By registering each IoT device with the Home Gateway, you can manage (lock, unlock, open, close, etc.) all devices from one interface.
- **Automation Possibilities:** The gateway can define rules (for example, *if wind is high, automatically close window*) involving multiple devices.
- **Easy Monitoring:** The Home Gateway's IoT Monitor shows real-time statuses of all connected devices, making it simple to troubleshoot or observe behavior.

F. Control IoT Devices Through the Tablet

11. **Open IoT Monitor:**

    - On the **Tablet**, switch to the *Desktop* tab.
    - Click the **IoT Monitor** icon to launch the web interface that connects to the Home Gateway.

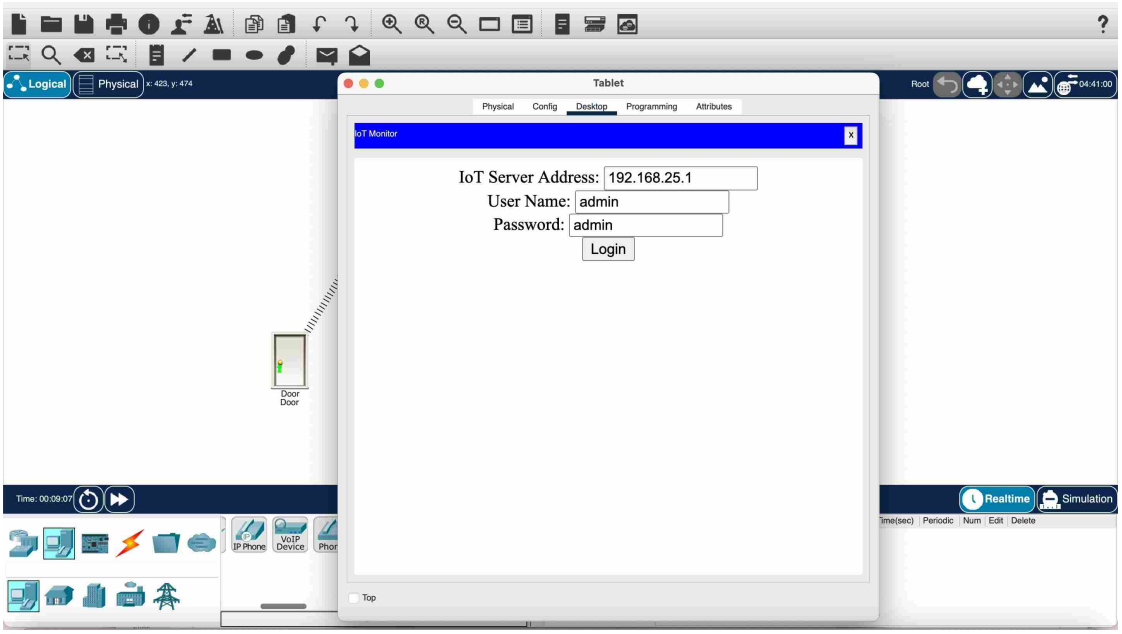


Figure 4.9: Accessing IoT Monitor on the Tablet.

12. **Login:**

- In many cases, the **Server IP**, **Username**, and **Password** may already be auto-filled (*e.g.*, `192.168.x.x`, `admin`, `admin`).
- Click the **Login** button to proceed. You should then see a list of all registered IoT devices under the Home Gateway.

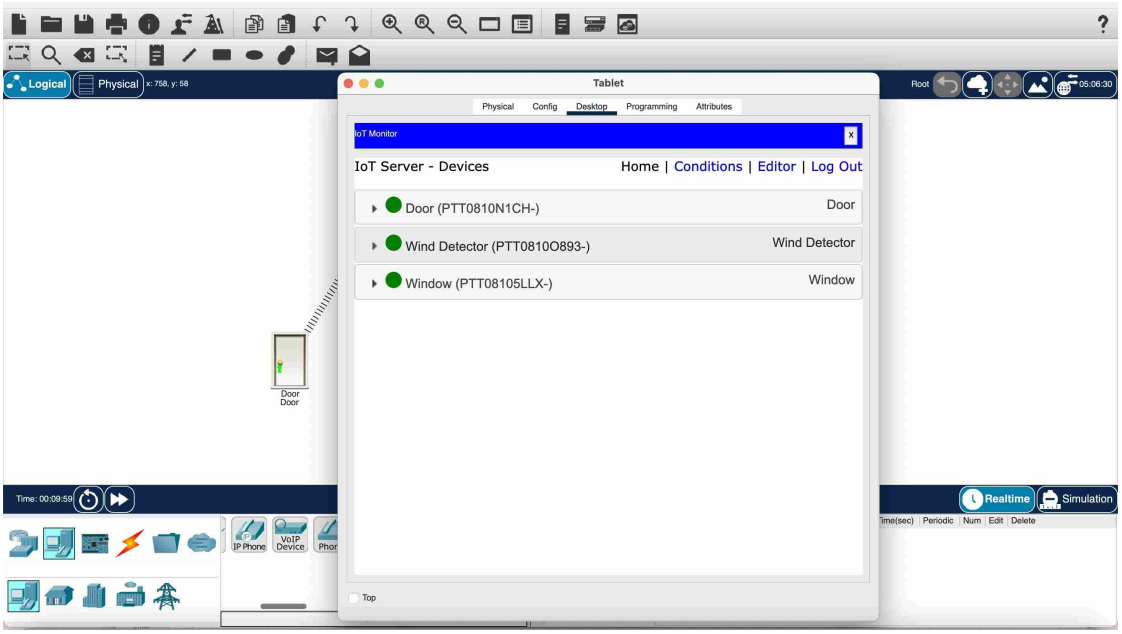


Figure 4.10: IoT devices visible after logging in.

13. **Rename Devices:**

- Within the IoT Monitor interface, locate the **Door**, **Wind Detector**, and **Window**.
- Rename them as follows:
  (a) Door → `SMART-DOOR`
  (b) Wind Detector → `WIND-DETECTOR`
  (c) Window → `SMART-WINDOW`

Using clear, descriptive names helps distinguish devices in larger or more complex topologies.

14. **Interact with the Devices:**

- From the **IoT Monitor**, try locking or unlocking `SMART-DOOR`, or open/close `SMART-WINDOW`.
- Observe that changes made here instantly reflect on the Packet Tracer workspace (e.g., the door icon may change states, or the window shows as open or closed).

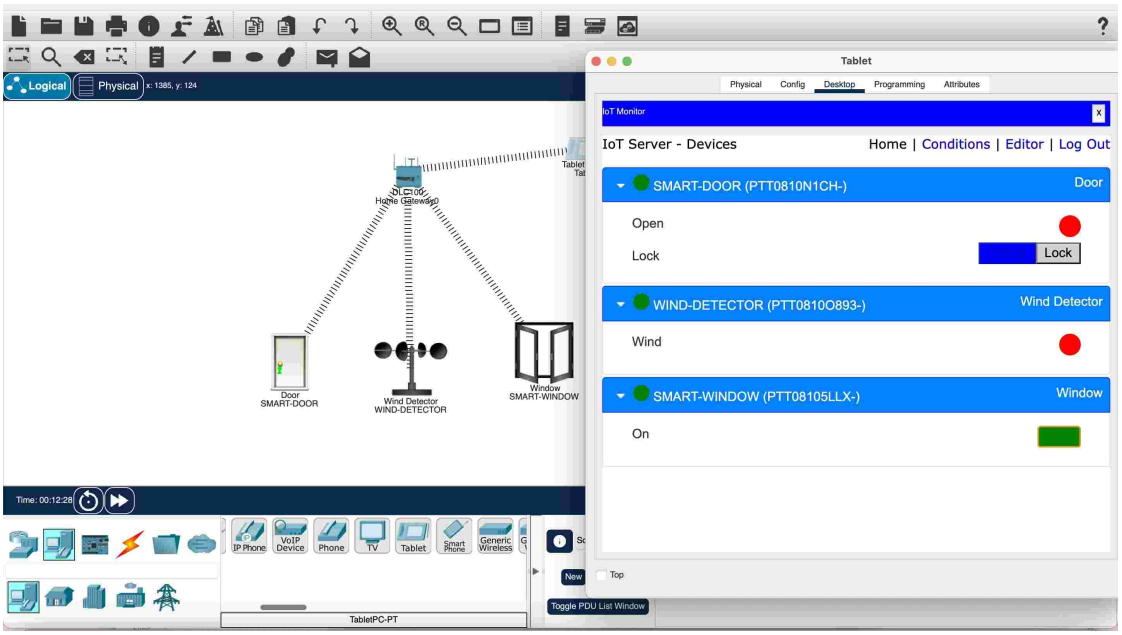


Figure 4.11: Controlling the Smart Door and Window from the Tablet.

— Why Control IoT Devices via Tablet?.

- **Central Access Point:** A Tablet interface mimics a typical user-friendly home automation controller, letting you manage multiple IoT devices at once.
- **Live Feedback:** Actions like locking a door or opening a window take effect in real time, demonstrating how IoT Monitor and Packet Tracer work together.
- **Future Expansion:** You can add rules, schedules, or triggers (e.g., *close windows when wind is strong*) through this same interface.

G. Modify Wind Detector Icons

15. **Use Motion Detector as a Base:**

- In the Device Selection box, go to *End Devices* → *Home Devices*.
- Drag a **Motion Detector** onto your workspace.

Although you'll rename and re-icon this device, it serves as a convenient starting point for creating a custom "Wind Detector."

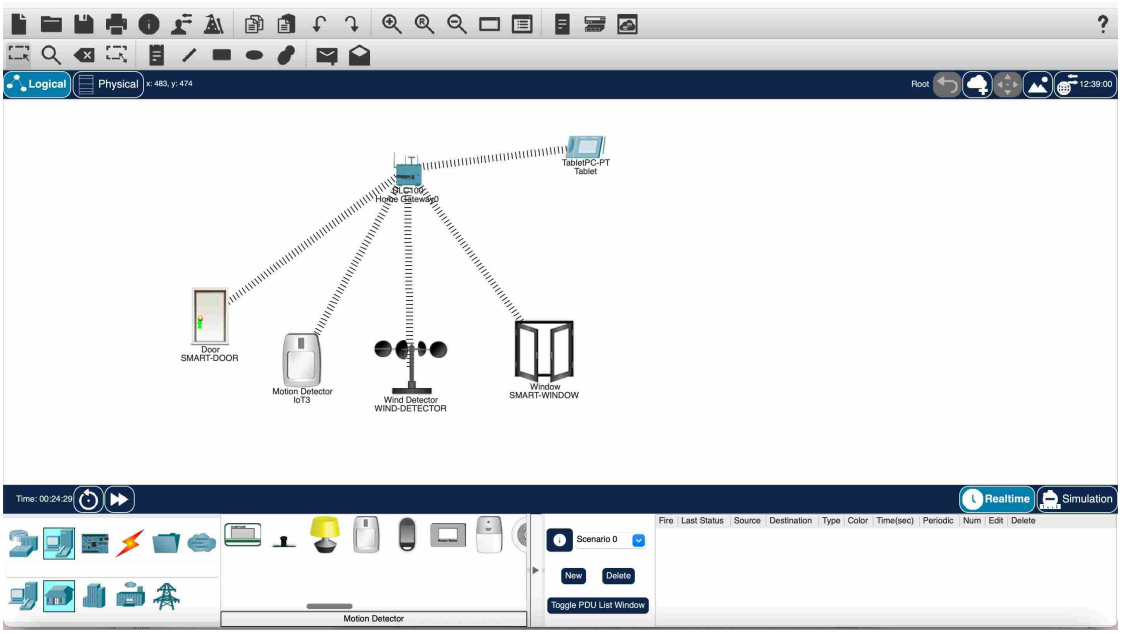


Figure 4.12: Adding a Motion Detector to transform into a Wind Detector.

16. **Open Thing Editor:**

- Click the newly added **Motion Detector**.
- Under **Advanced**, select the **Thing Editor**.
- Click **New** next to the motion detector icon or images list.

This allows you to customize the device's appearance, name, and behavior.

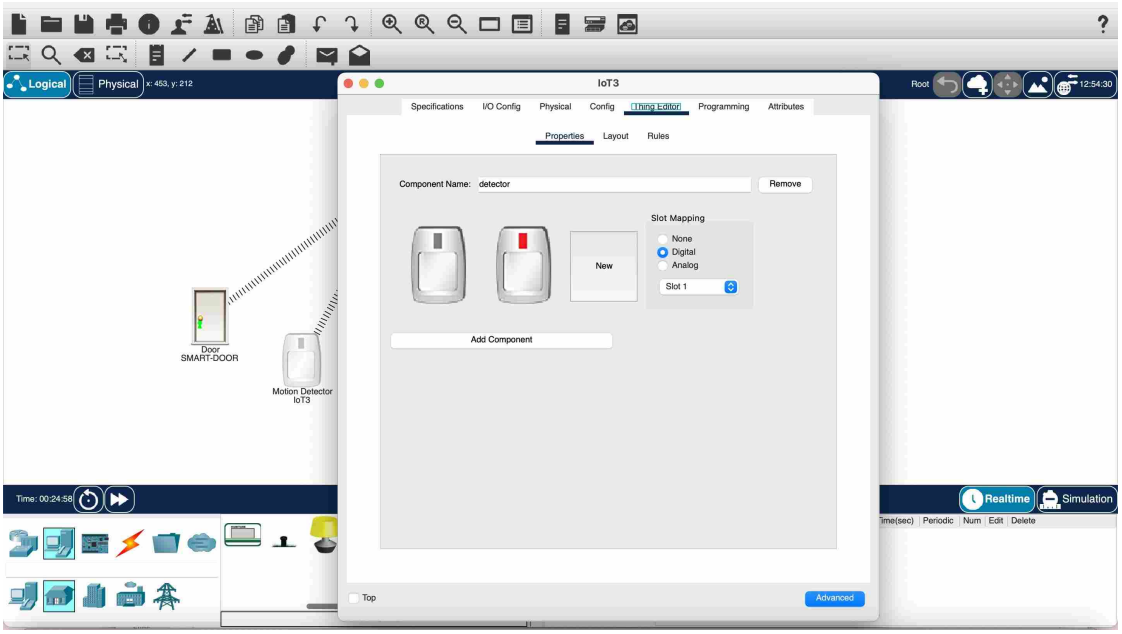


Figure 4.13: Thing Editor: preparing to upload new icons.

17. **Choose Wind Detector Images:**

- In the *Thing Editor*, delete the original motion detector images (often labeled `icon0`, `icon1`, etc.).
- Replace them with the two desired “Wind Detector” images you have, each representing a different wind state (e.g., *low wind*, *high wind*).

Make sure the new icons are clearly identified and saved.

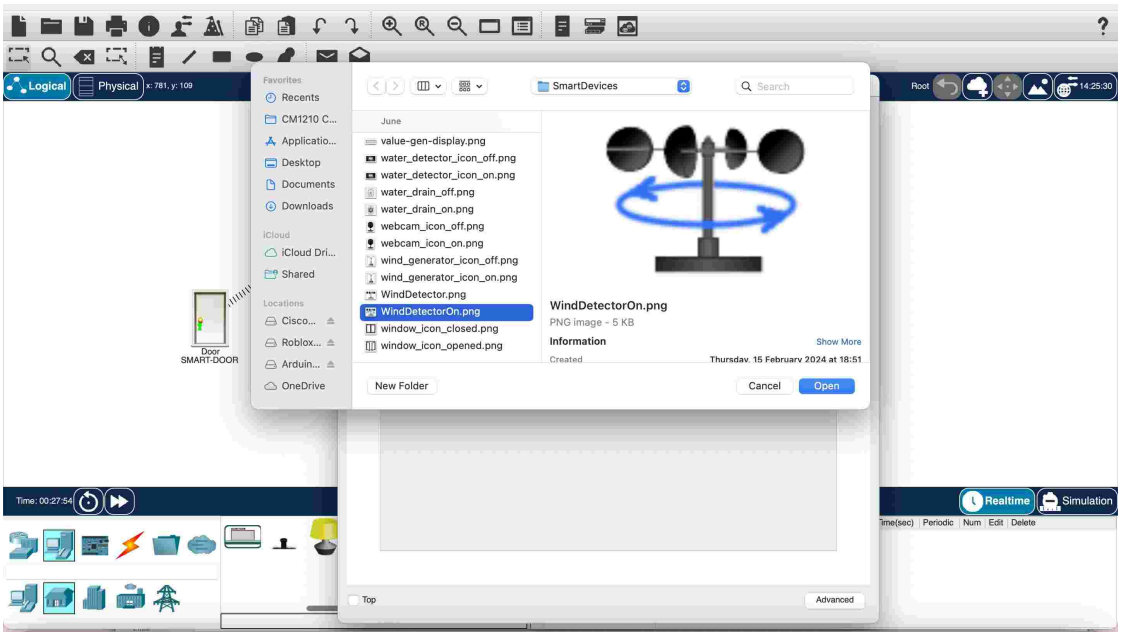


Figure 4.14: Selecting new icons to represent wind states.

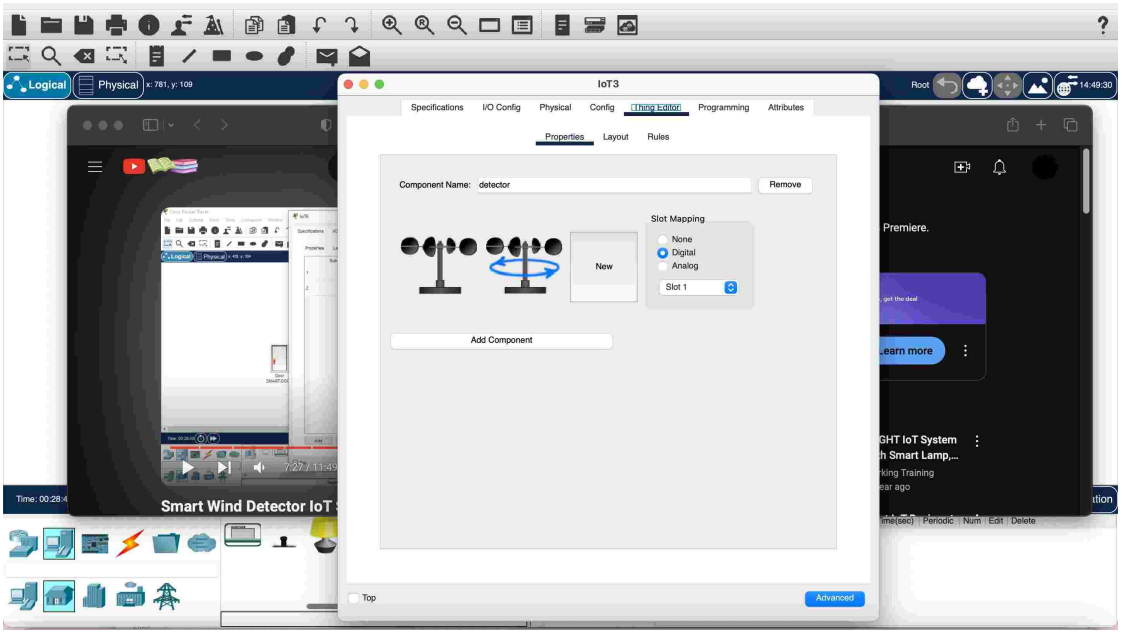


Figure 4.15: Second icon for an alternative wind state.

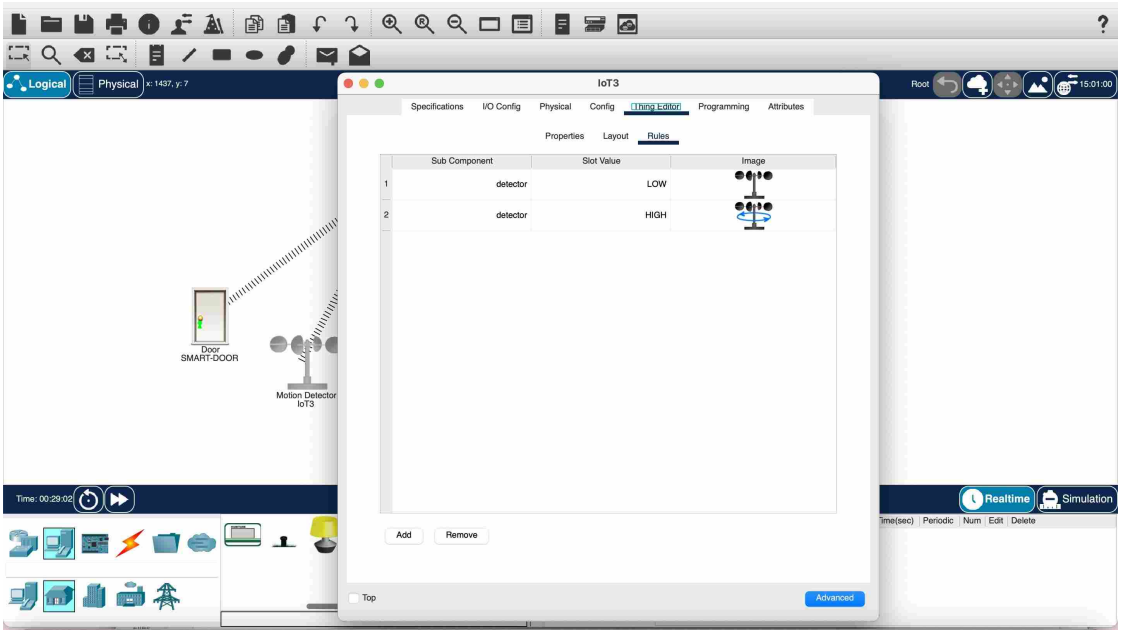


Figure 4.16: Confirming the new icons in the Thing Editor.

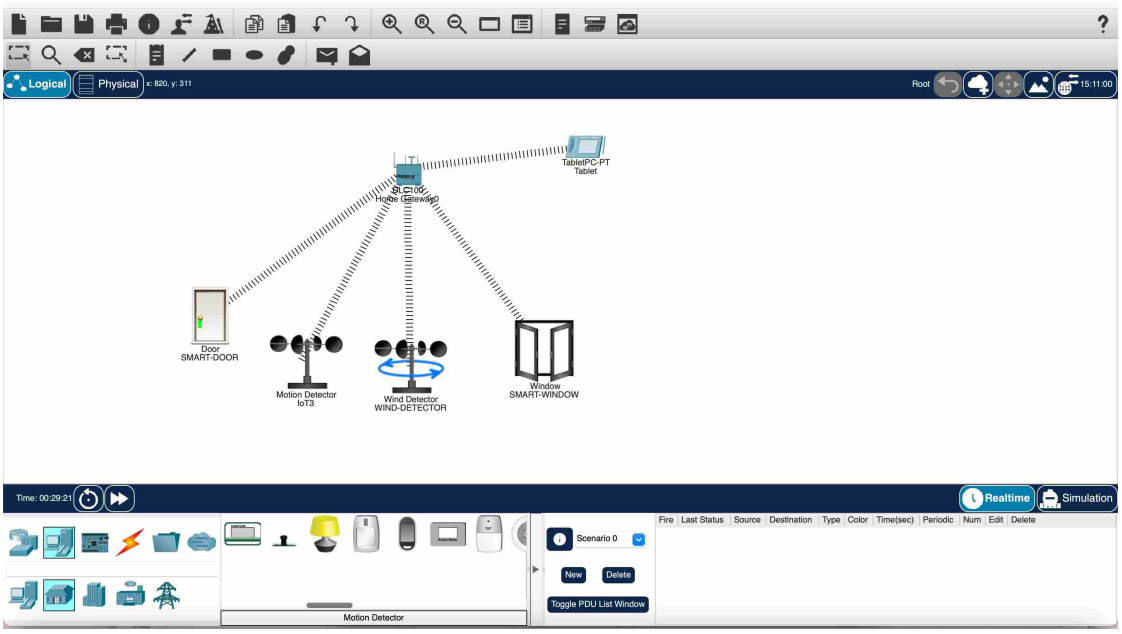


Figure 4.17: Deleting the old Motion Detector icon references.

18. **Replace the Old Device:**

    - Once the images are swapped, **remove the old wind detector** (or the original motion detector if you had one) from the workspace.
    - Place your newly modified device into the workspace.
    - Rename it **WIND-DETECTOR** (or a similar descriptive name).

Now you have a custom *Wind Detector* device that visually reflects different wind states.

— Why Customize the Icons?.

- **Realistic Simulation:** Visual changes in Packet Tracer help identify device states (e.g., high wind or low wind) at a glance.
- **Simplify Troubleshooting:** Clear, custom icons make it easier to see which sensor is detecting what condition.
- **Engage Learners:** Creating custom devices fosters a deeper understanding of how IoT devices can be adapted and integrated into the network simulation.

H. Add Wind Detection Conditions

19. **Register the New `WIND-DETECTOR`:**

    - As with your other IoT devices, click the newly created **WIND-DETECTOR**, then go to **Config** → **Settings**.
    - Under **Gateway/DNS IPv4**, set `DHCP` to acquire an IP automatically.
    - Under **IoT Server** (or "Remote Server"), select **Home Gateway**.
    - Click **Connect** to finalize the registration.

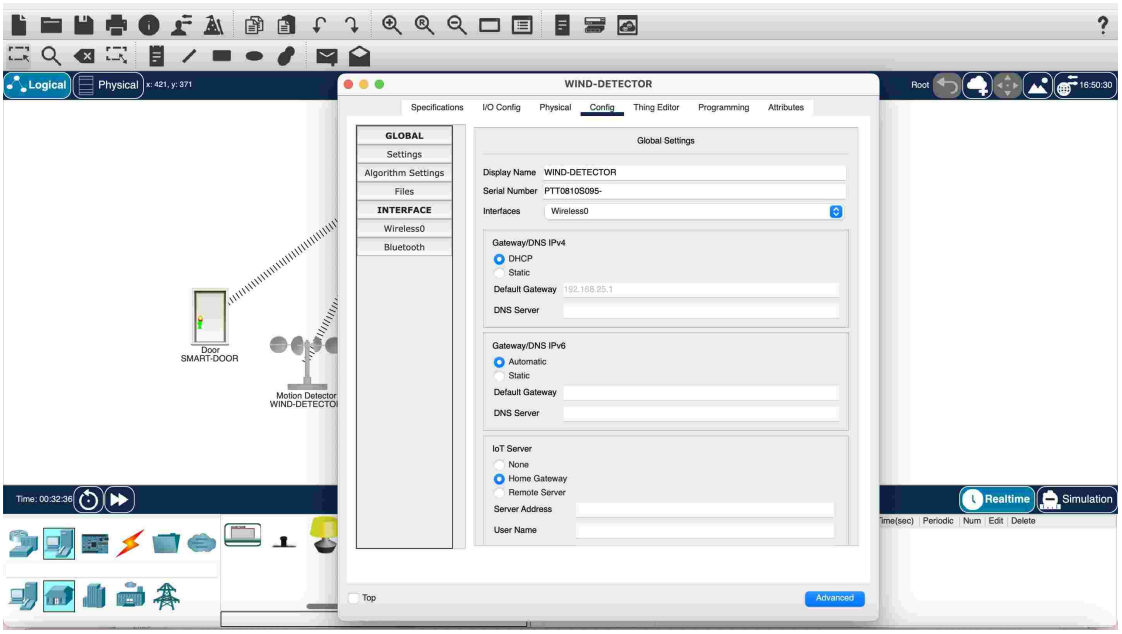


Figure 4.18: Registering the newly created `WIND-DETECTOR` device.

20. **Confirm via Tablet:**

    - Open the **IoT Monitor** on your Tablet (*Desktop* → *IoT Monitor*).
    - Log in if prompted.
    - You should now see **WIND-DETECTOR** listed alongside your other devices.

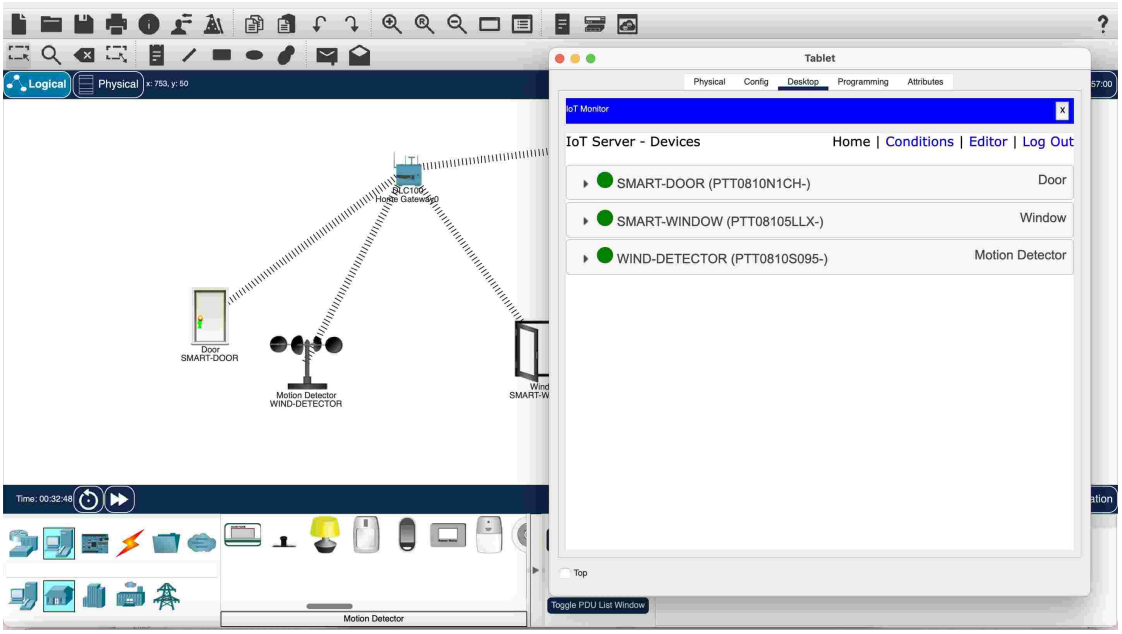


Figure 4.19: `WIND-DETECTOR` now displayed in the Tablet's IoT Monitor.

21. **Add Conditions for Wind Detection:**
    The IoT Monitor allows you to create **Conditions** that trigger specific **Actions** across devices. For example:
    - **Condition 1:** `WIND-DETECT` *If* `WIND-DETECTOR` is **on = true**, → **Lock SMART-DOOR**, **turn off SMART-WINDOW**.
    - **Condition 2:** `NOWIND` *If* `WIND-DETECTOR` is **on = false**, → **Unlock SMART-DOOR**, **turn on SMART-WINDOW**.

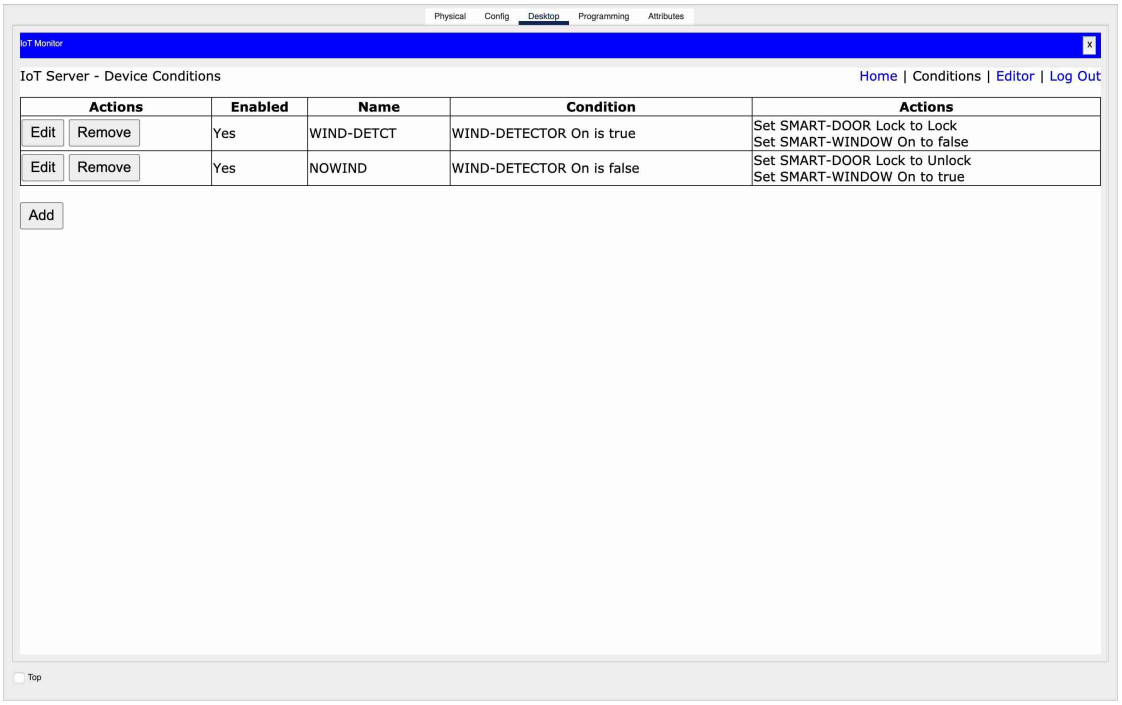


Figure 4.20: Condition 1 for detecting wind: door locked, window off.

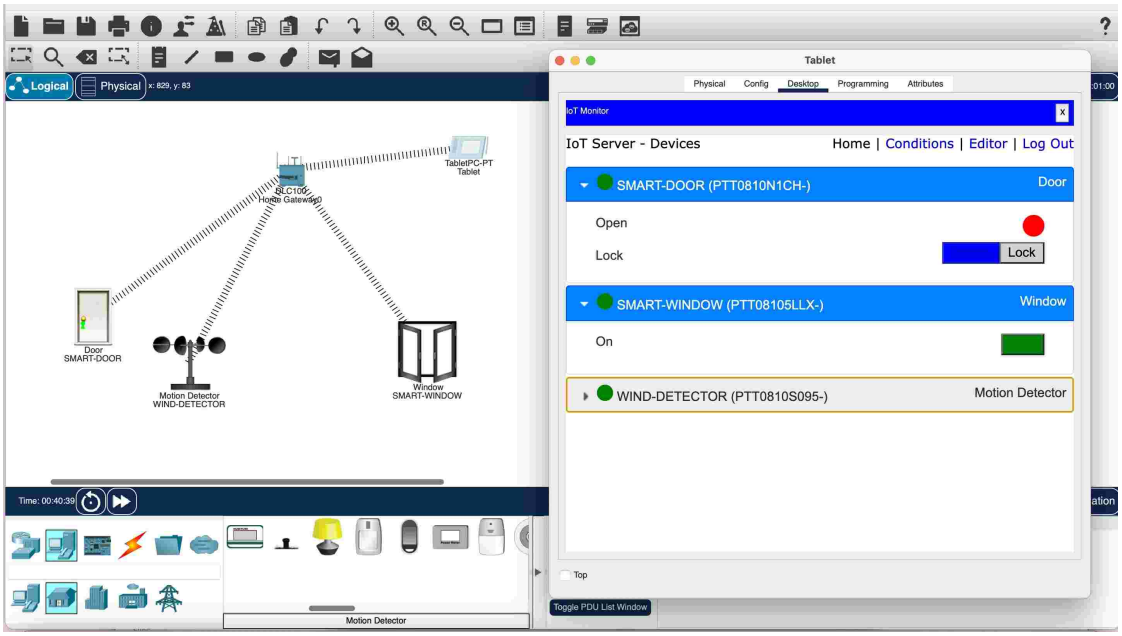


Figure 4.21: Condition 2 for no wind: door unlocked, window on.

— Why Create Wind Detection Rules?.

- **Automated Safety:** High wind could cause damage if doors or windows are left open. Automating their state prevents potential hazards.
- **Hands-Free Control:** No need for manual toggling each time; once the `WIND-DETECTOR` senses wind, the system adjusts devices accordingly.
- **Scalability:** You can add more devices (like `SMART-CURTAINS` or `SMART-SHUTTERS`) to these rules, broadening your smart home's capabilities.

I. Test Wind Detector Functionality

22. **Arrange Devices and Monitor:**

    - In Packet Tracer's *Logical* view, position your `SMART-DOOR`, `SMART-WINDOW`, and `WIND-DETECTOR` relatively close together for easy observation.
    - Open the **IoT Monitor** on your **Tablet** and place it side by side with the main workspace (or toggle between them as needed).

    This setup helps you see real-time changes on both the devices and the IoT Monitor interface.

23. **Toggle States:**

    - From the *IoT Monitor*, experiment with manually *unlocking* the `SMART-DOOR` or *opening* the `SMART-WINDOW`.
    - Then hold **Alt/Option** while hovering over the `WIND-DETECTOR` to simulate the sensor turning *on*.
    - Observe how the system responds in real time.

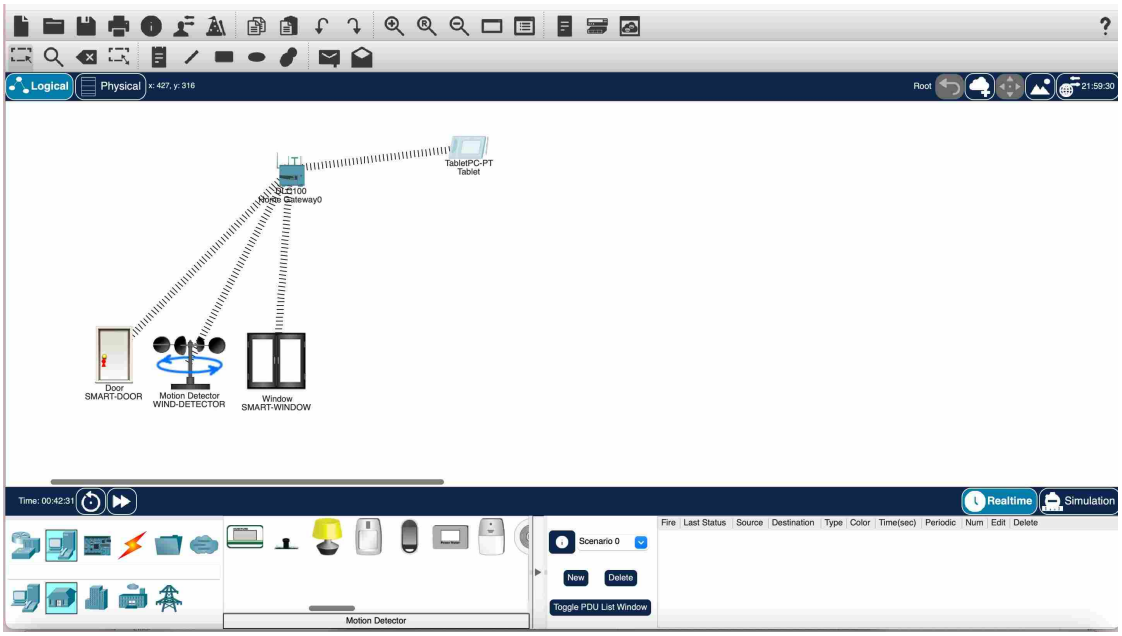


Figure 4.22: Observing the Wind Detector trigger the Smart Door and Window.

24. **Observe Automation:**

- With `WIND-DETECTOR` set to `on`, the `SMART-DOOR` automatically **locks** and the `SMART-WINDOW` **closes** (or toggles `off`), reflecting the conditions you configured in the IoT Monitor.
- Returning the `WIND-DETECTOR` to `off` reverts both devices to their previous states (*door unlocked*, *window open*).
- This demonstrates a fully automated home scenario, where sensor data (wind detection) triggers protective actions (locking or closing openings in the home).

— Testing Tips.

- **Check the IoT Monitor Log:** Some versions of Packet Tracer display logs indicating when a sensor changes state. Verify your rules are firing at the expected times.
- **Experiment with Delays:** If you want a brief waiting period before the window closes, you can add an optional *delay* in the condition settings (where supported).
- **Expand Your System:** Try adding more devices (lights, fans) and linking them to wind detection or other triggers for a comprehensive smart home experience.

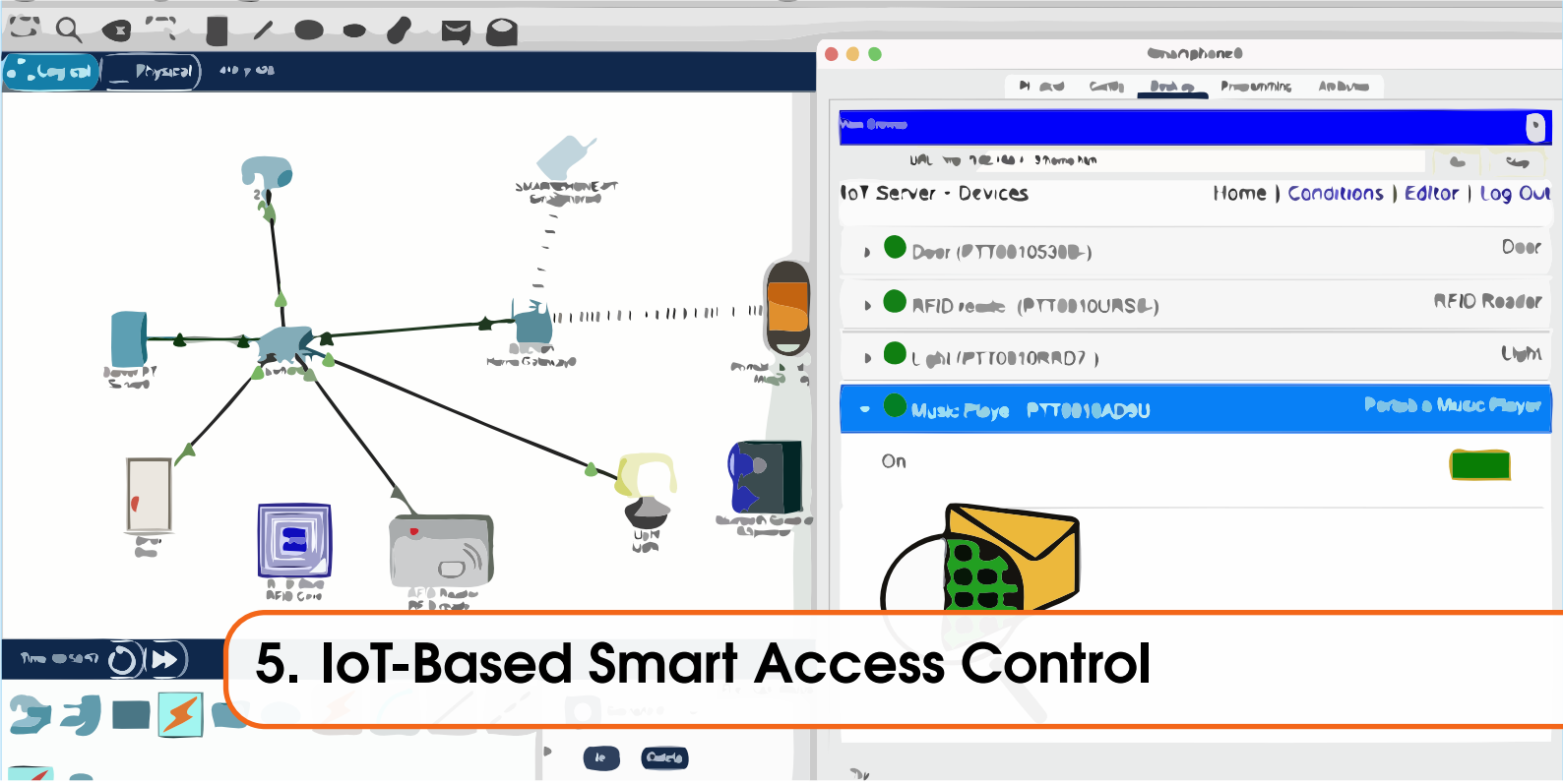
Measuring Success

- **Registered Devices:** All IoT devices (SMART-DOOR, SMART-WINDOW, WIND-DETECTOR) appear under the Home Gateway's IoT Monitor.
- **Wind Detector Reactions:** Hovering **Alt/Option** on WIND-DETECTOR correctly toggles the door lock and window state.
- **Modified Icons:** The custom WIND-DETECTOR icons replace the old motion detector icons successfully, showing correct visuals for each state.
- **Automated Conditions:** The door remains locked and the window off when wind is detected, and revert to normal when no wind is detected—verifying your condition-based actions.

■

Summary

You have successfully created an IoT-based smart home system in Cisco Packet Tracer, integrating a **WIND-DETECTOR** with a **SMART-DOOR** and **SMART-WINDOW**. By setting up wireless connectivity, assigning IP addresses, and defining condition-based rules in the IoT Monitor, you demonstrated how wind detection can automatically secure the home's door and window. This hands-on exercise underscores the flexibility and power of Packet Tracer's IoT functionality.



5. IoT-Based Smart Access Control

Introduction

This lab will show you how to build and configure a **smart access control system** in Cisco Packet Tracer using *RFID*, *Bluetooth*, and a *smartphone*. You will create a network topology that includes an RFID-based door, a Bluetooth music player, and a smartphone for remote control. By the end, you will have tested both RFID access (unlocking a door) and Bluetooth device management (music playback) from a smartphone.

Objective

- **Build and configure** a Packet Tracer network involving RFID, Bluetooth, and smartphone devices.
- **Set up and test connectivity** of IoT devices, ensuring proper IP configuration.
- **Implement and test** RFID access control logic (valid vs. invalid tag IDs).
- **Manage devices** (e.g., light, music player) via Bluetooth and a smartphone interface.

Lab Plan

A. **Launch Packet Tracer and Observe Initial Workspace**
B. **Build the Topology**
C. **Configure the Router (DHCP) and Server**
D. **Activate the IoT Registration Server**
E. **Register Devices (Door, RFIDReader)**
F. **Add Conditions for RFID Access**
G. **Assign and Test RFID Tag ID**
H. **Add Bluetooth Devices (Music Player, Speaker)**
I. **Add Home Gateway and Smartphone**
J. **Test IoT Control via Smartphone**

Required Software

- **Cisco Packet Tracer 8.x** (or newer)
- Basic knowledge of configuring IoT devices (RFID, Bluetooth) in Packet Tracer
- Optional: external PC browser for verifying server-based pages (if needed)

A. Launch Packet Tracer and Observe Initial Workspace

1. **Open Packet Tracer:**
   Double-click the Packet Tracer icon (or find it in your applications) to launch. You should see a blank Logical topology workspace:

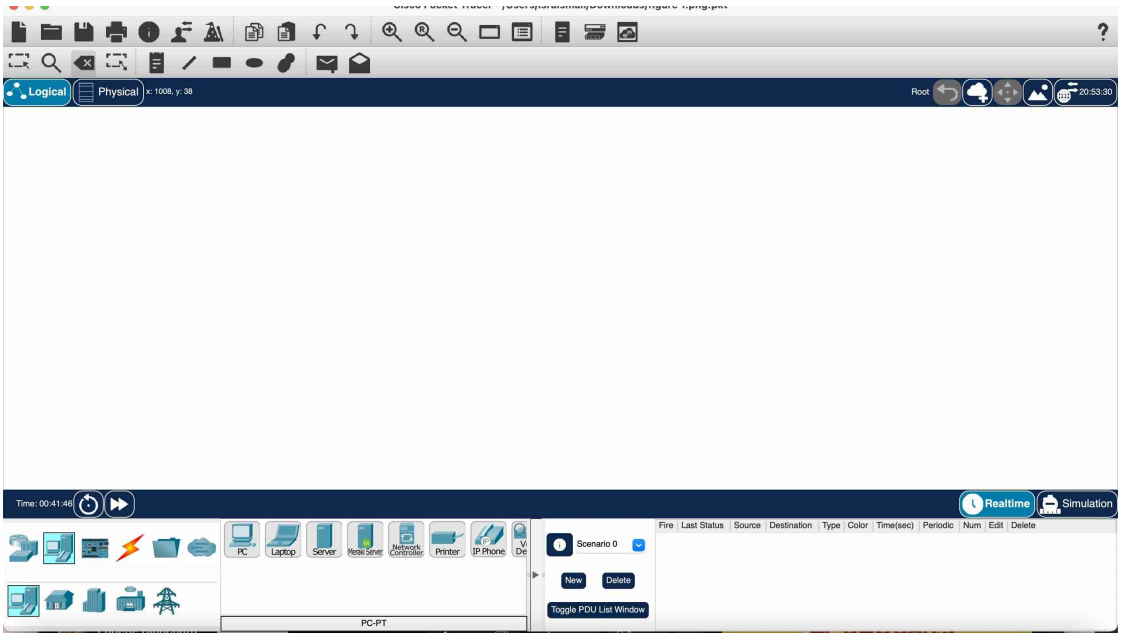


Figure 5.1: Initial Workspace

B. Build the Topology

Adding the Devices to the Workspace

2. **Add network devices:**
   From the Device Selection box, place the Router, Switch, Server, Door (IoT device), RFID Reader, RFID Card, and any other required devices onto the workspace. Refer to the sample topology (Figure 5.2).

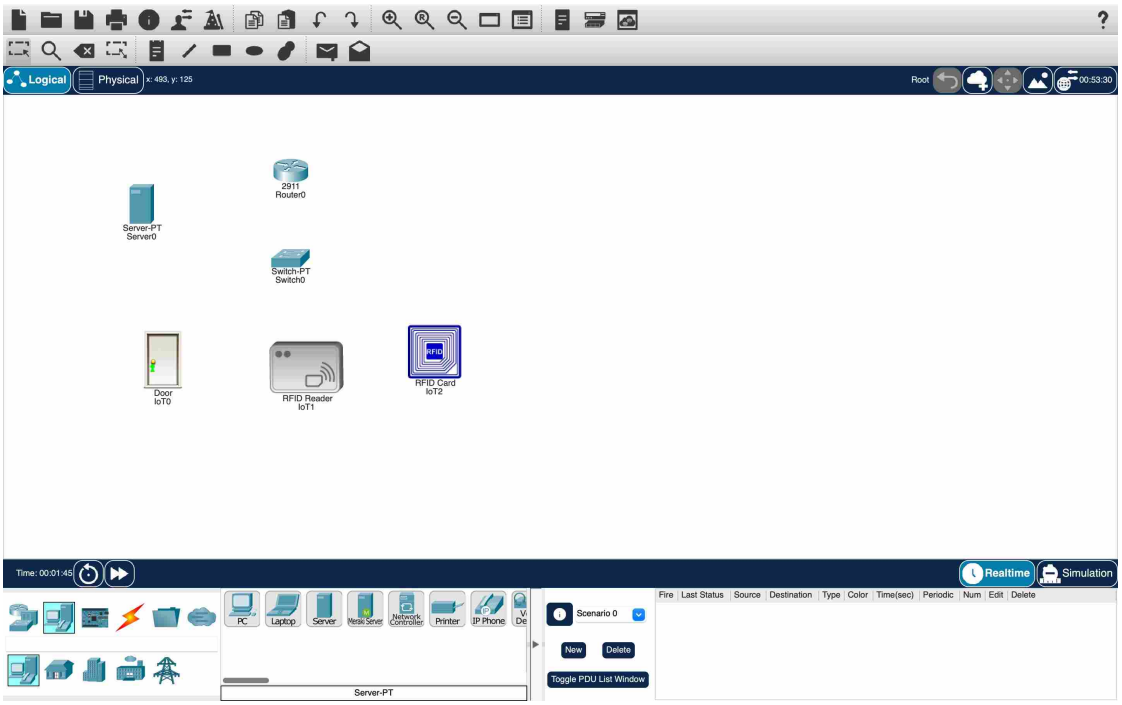


Figure 5.2: Sample topology for IoT-based smart access control.

3. **Renaming IoT devices:**
   For clarity:
   - `Door` → Door
   - `RFID Reader` → RFIDReader
   - `RFID Card` → RFIDTag

   In Packet Tracer, click on each device, go to *Config*, and change the *Display Name*.

Connecting Devices

Use the “Automatically Choose Connection Type” cable (lightning bolt icon). Follow these steps:

4. Connect **Router** to the **Switch**.
5. Connect **Switch** to the **RFIDReader**.
6. Connect **Switch** to the **Door**.
7. Connect **Switch** to the **Server**.

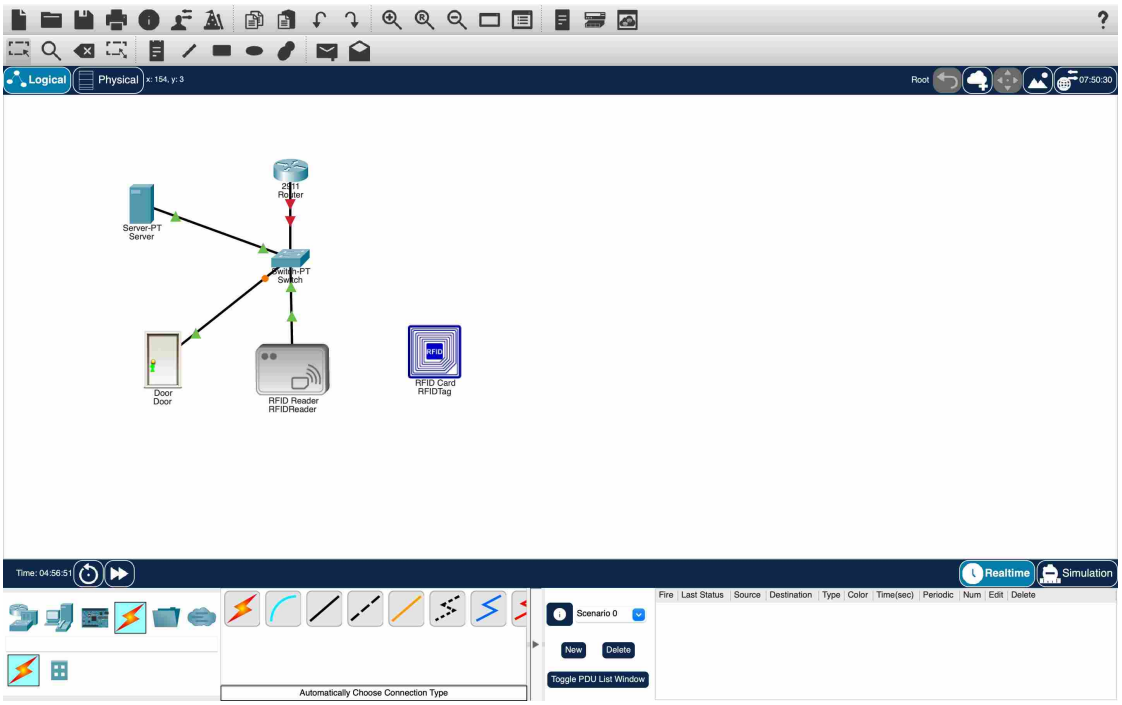


Figure 5.3: Cabling the router, switch, IoT door, and RFID reader.

C. Configure the Router (DHCP) and Server

Configuring the Router

8. **Open the Router (R1):**
   Click the Router icon, go to *Config*.
9. **Rename and Assign IP:**
   Change the Display Name to `R1` and Hostname to `R1`. In **GigabitEthernet0/0**:
   - `IPv4 Address`: 192.168.1.1
   - `Subnet Mask`: 255.255.255.0
   - Check the *Port Status* box to enable.

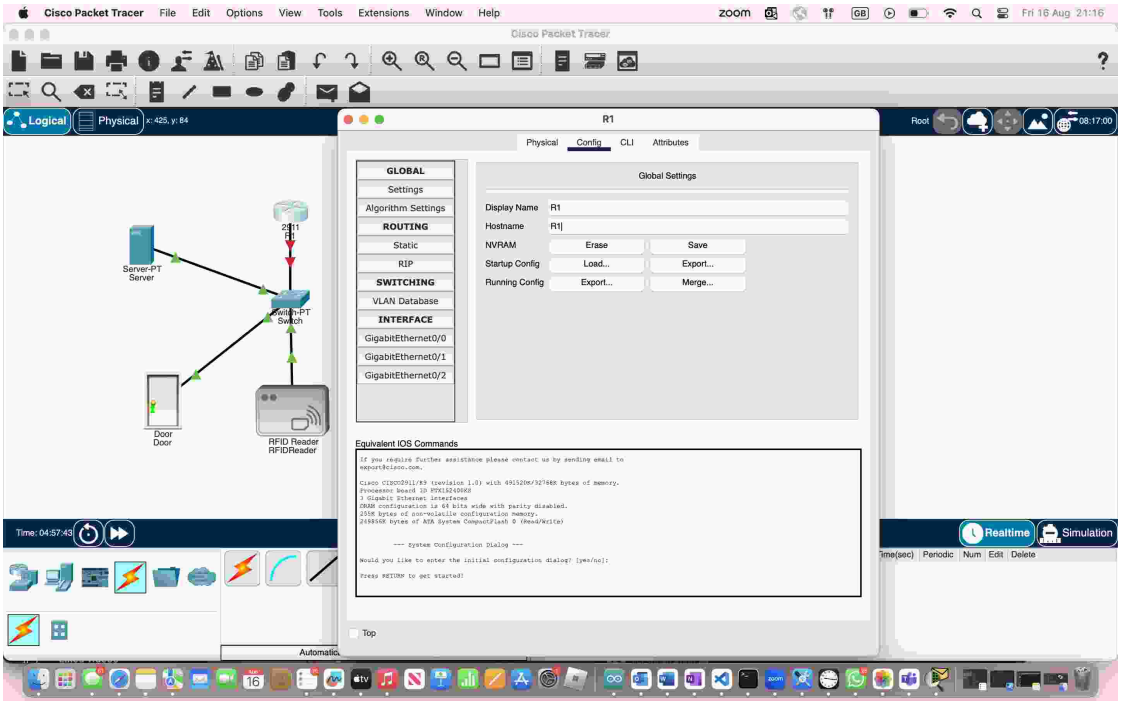


Figure 5.4: Renaming router to R1 and preparing interface config.

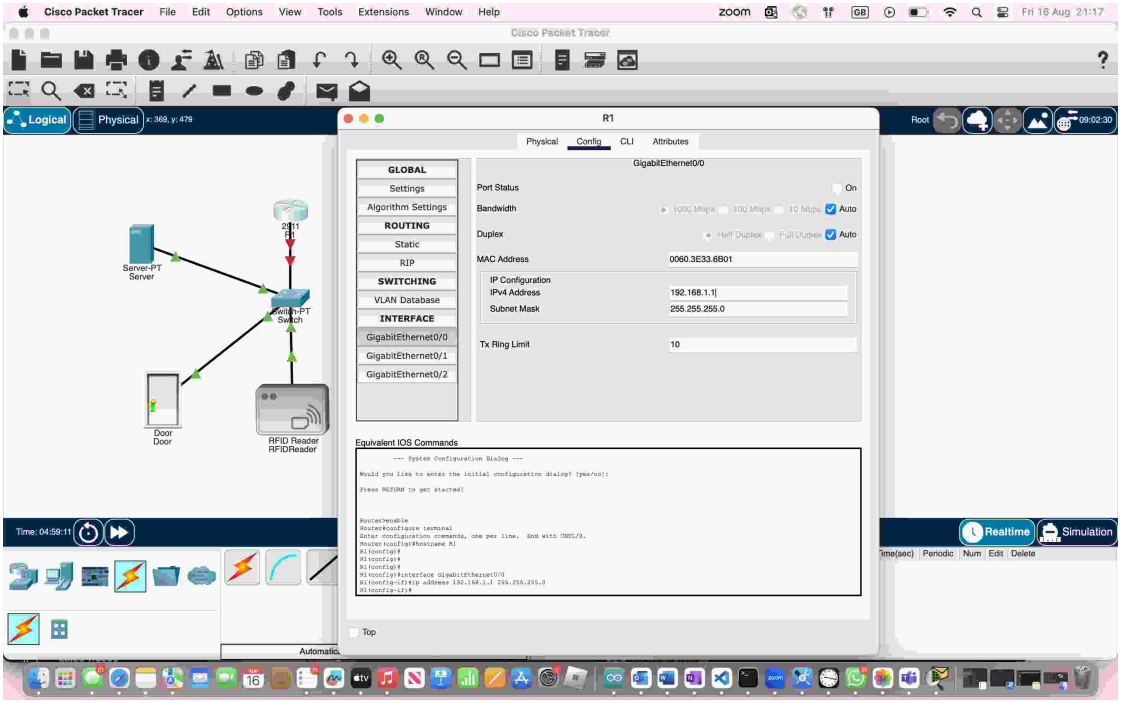


Figure 5.5: Assigning IP 192.168.1.1/24 to GigabitEthernet0/0.

10. **Enable DHCP from CLI:**
    Switch to the **CLI** tab, press Enter, and type:

```
ip dhcp pool iot
network 192.168.1.1 255.255.255.0
```

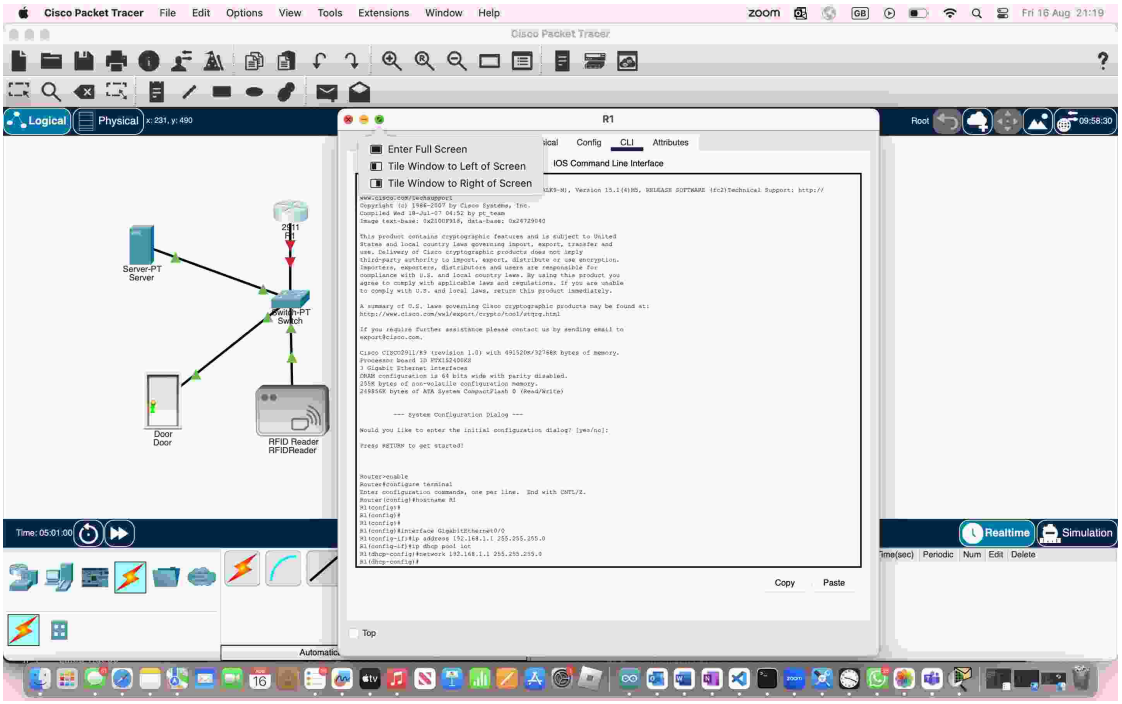


Figure 5.6: CLI commands to enable DHCP for the 192.168.1.0/24 network.

Configuring the Server

11. **Open the Server:**
    Go to *Desktop* → **IP Configuration** and assign:
    - `IPv4 Address`: 192.168.1.15
    - `Subnet Mask`: 255.255.255.0
    - `Default Gateway`: 192.168.1.1

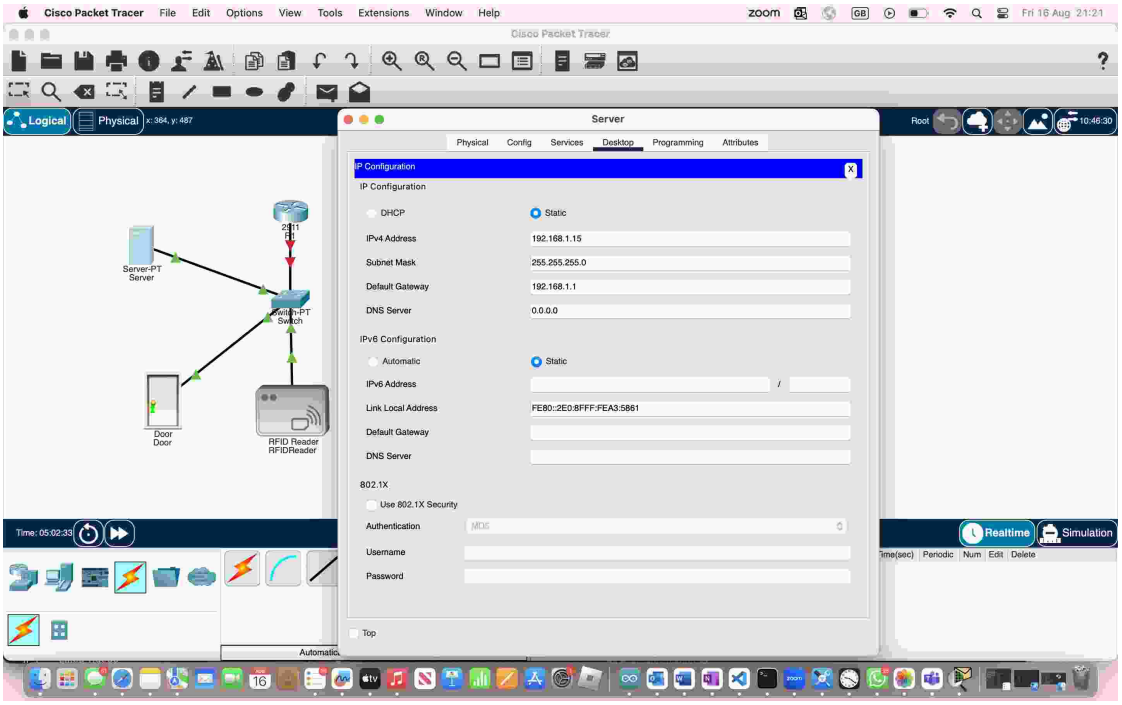


Figure 5.7: Server IP config: 192.168.1.15/24, gateway 192.168.1.1.

12. **Enable IoT Services:**
Go to *Services* → **IoT** and switch On the *Registration Server*.

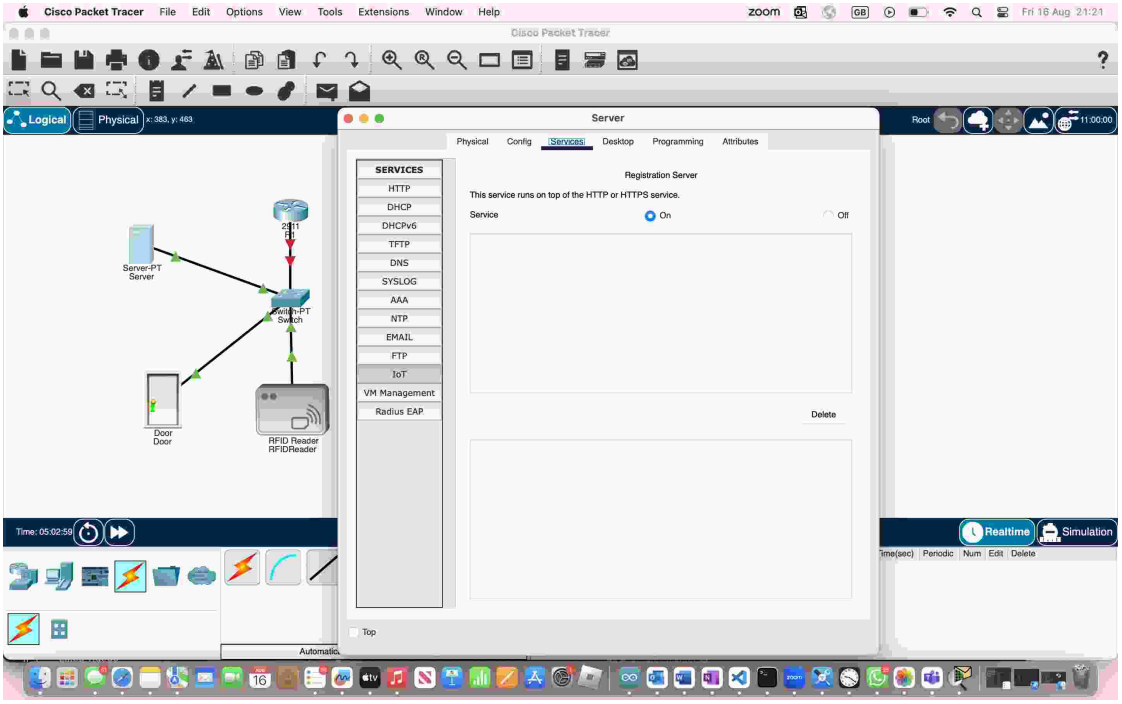


Figure 5.8: Turning on the IoT Registration Server.

D. Activate the IoT Registration Server

13. **Create an Account:**
    On the Server, go to *Desktop* → **Web Browser**. Enter `192.168.1.15` in the URL. Since there's no IoT account yet, click **Sign Up Now**, and use 'admin' / 'admin'.

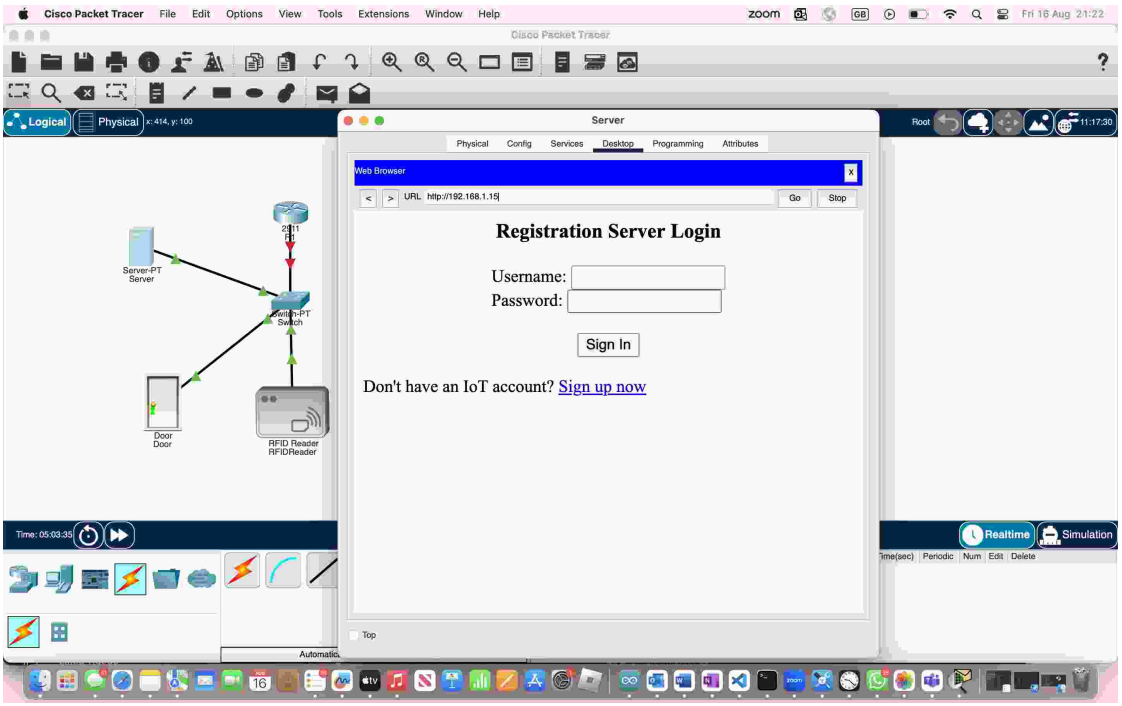


Figure 5.9: Creating a new IoT account with username/password "admin."

14. **Confirm Account Creation:**
    After hitting **Create**, you have an IoT account, but no devices are registered yet.

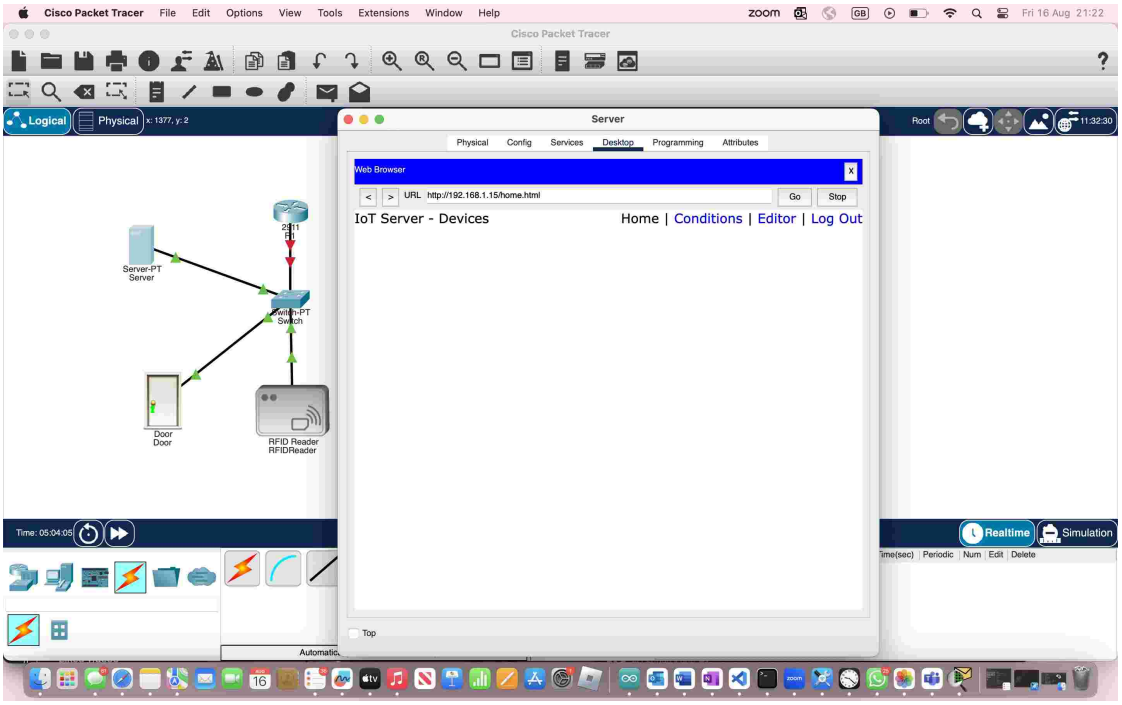


Figure 5.10: Successful account creation on the Registration Server.

E. Register Devices (Door, RFIDReader)

15. **Go to Device Config:**
    For each device (Door, RFIDReader), under *Gateway/DNS IPv4* choose `DHCP` so it picks up an IP from the router.
16. **Select Remote Server:**
    Under IoT Server options, click **Remote Server**, enter:
    - Server IP: 192.168.1.15
    - Username: admin
    - Password: admin

    Then click **Connect**.

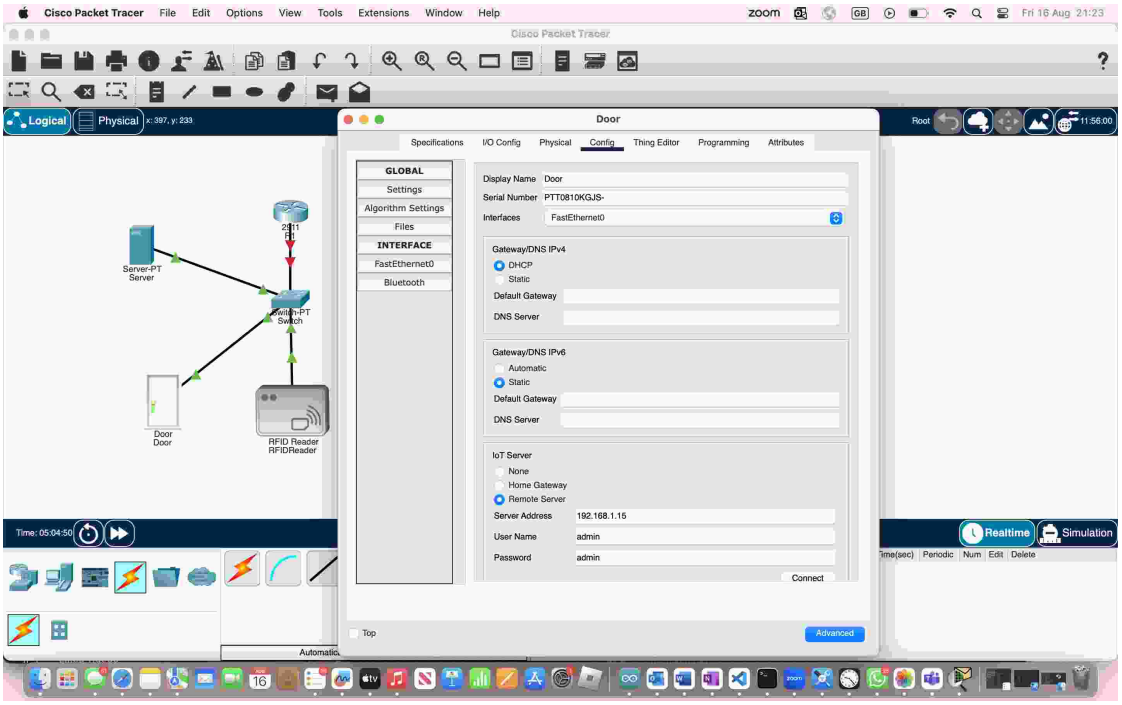


Figure 5.11: Registering the Door with the remote server (192.168.1.15).

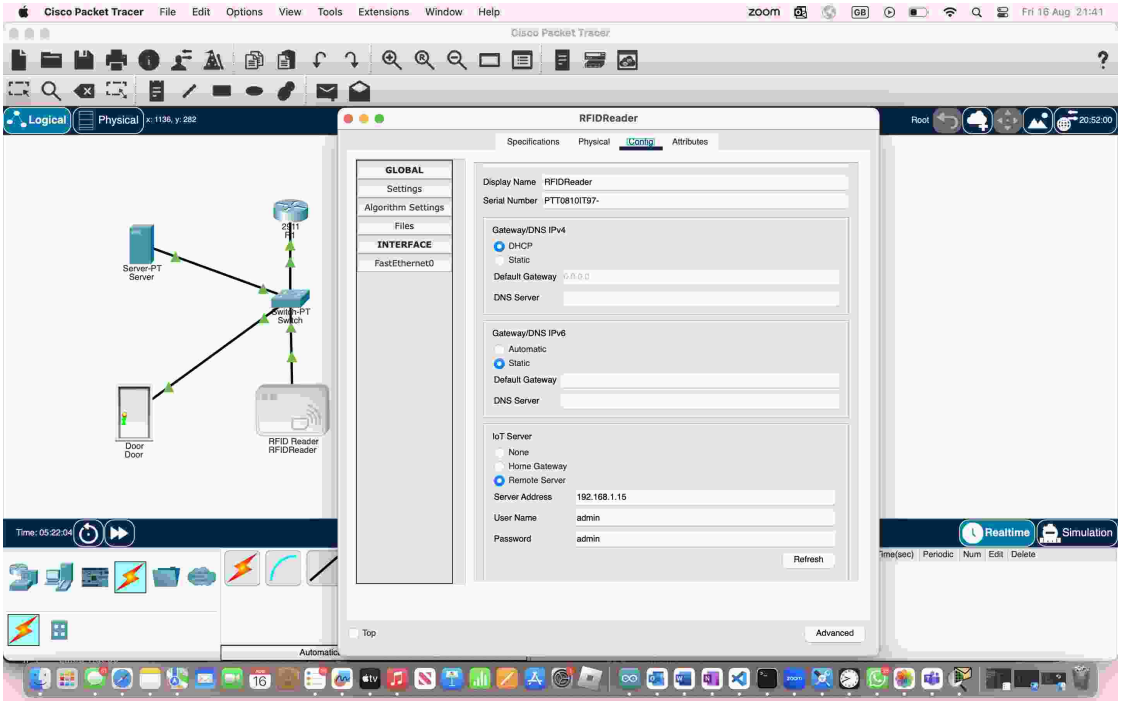


Figure 5.12: Registering the RFID reader similarly.

17. **Confirm Registration:**
Return to the server's **Web Browser**. You should see Door and RFIDReader listed under the IoT devices.

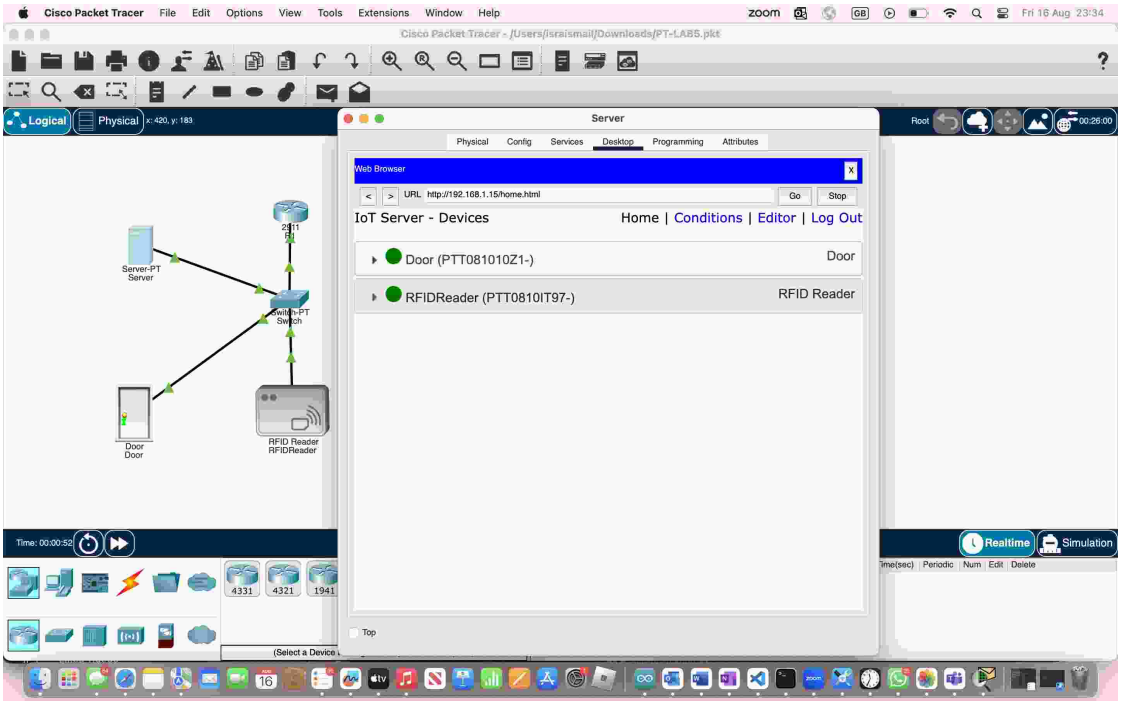


Figure 5.13: Door and RFID reader successfully connected to the IoT server.

F. Add Conditions for RFID Access

18. **Four Rules:**
    In the Registration Server's *Conditions* or *IoT Monitor* interface, add the following:
    - **Condition 1: RFIDValid** Condition: `RFIDReader Card ID == 1001` → *Set RFIDReader Status = Valid*.
    - **Condition 2: RFIDInvalid** Condition: `RFIDReader Card ID != 1001` → *Set RFIDReader Status = Invalid*.
    - **Condition 3: DoorOpen** Condition: `RFIDReader Status = Valid` → *Door Lock = Unlock*.
    - **Condition 4: DoorLock** Condition: `RFIDReader Status = Invalid` → *Door Lock = Lock*.

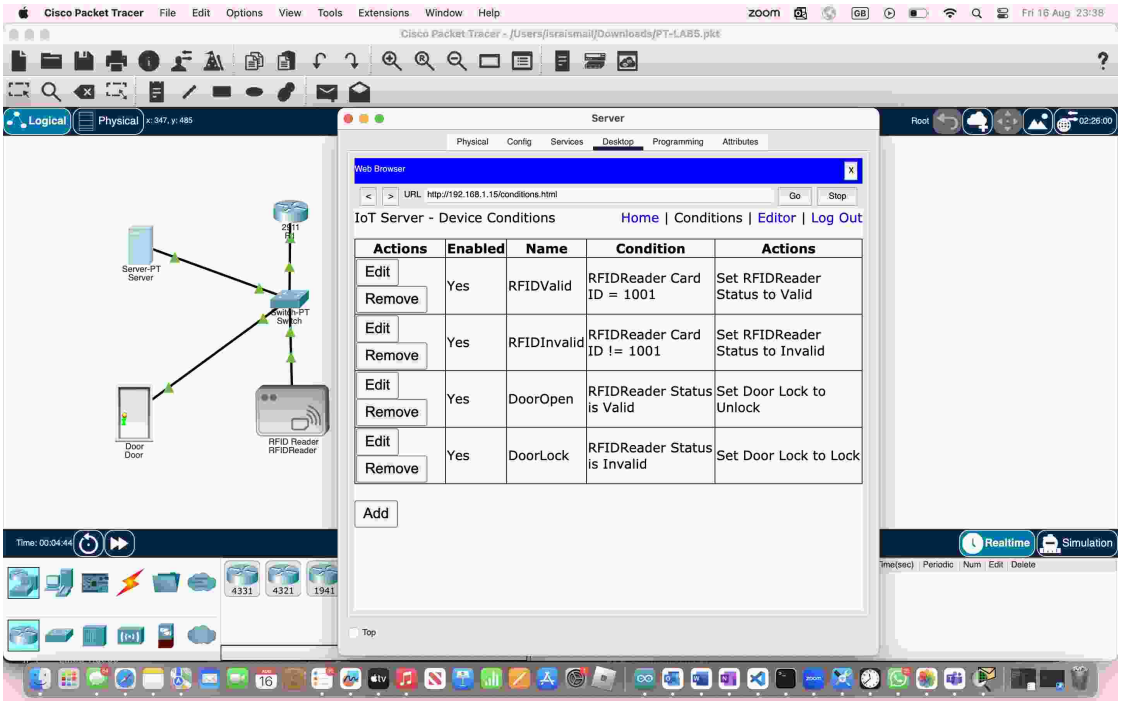


Figure 5.14: Creating conditions to manage door lock based on RFID validity.

G. Assign and Test RFID Tag ID

19. **Modify the RFID Tag:**
    In *Attributes*, set `CardID` from 1001 to 500. Then hover the tag over the reader. The reader should turn red, locking the door (invalid).
20. **Restore Valid ID:**
    Change the `CardID` back to 1001 and test again. This time, the reader turns green and unlocks the door.

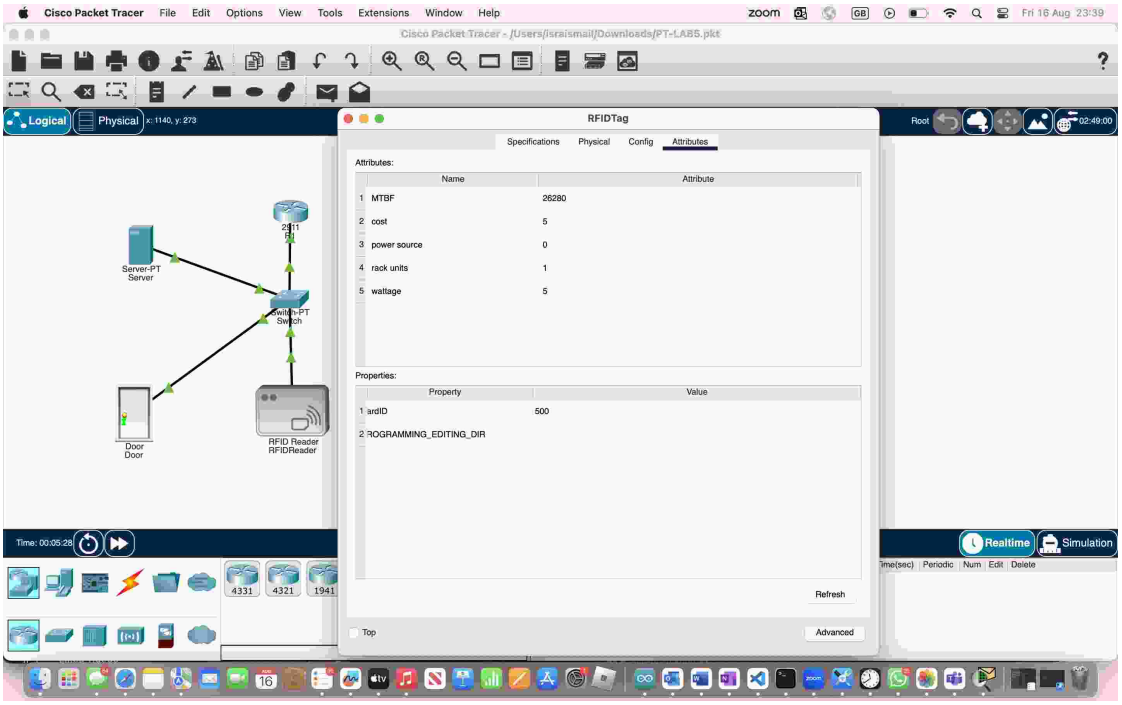


Figure 5.15: Reader denies access (red) for invalid CardID (500).

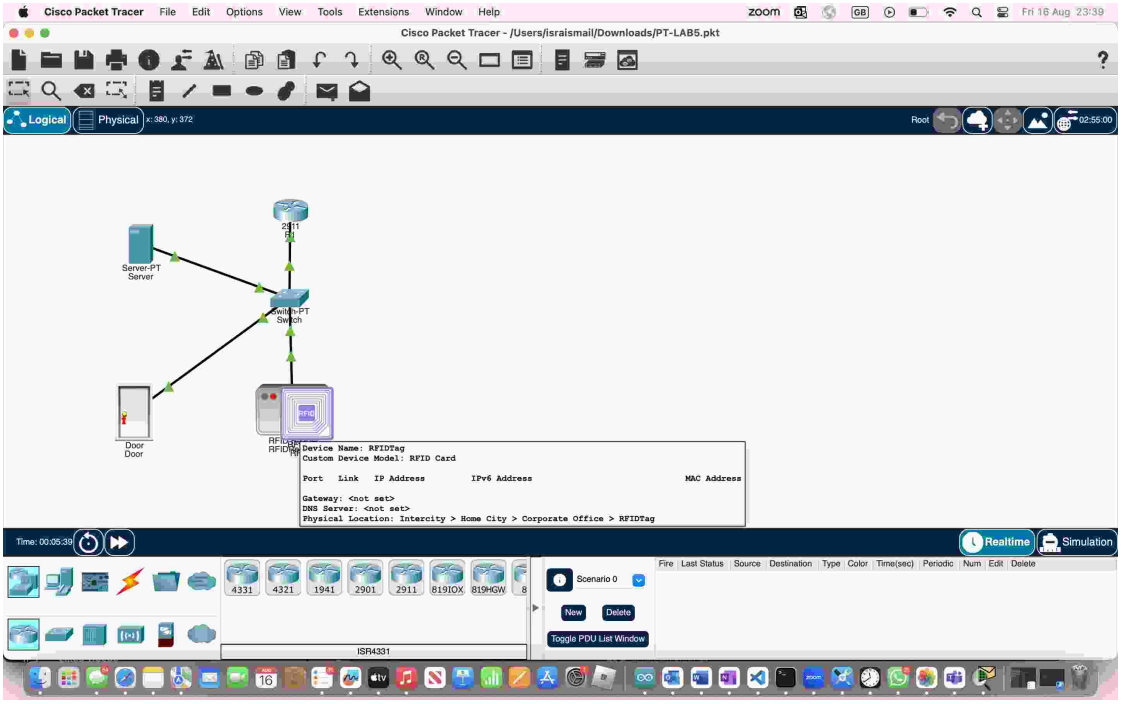


Figure 5.16: Switching card ID back to 1001 to allow green (valid).

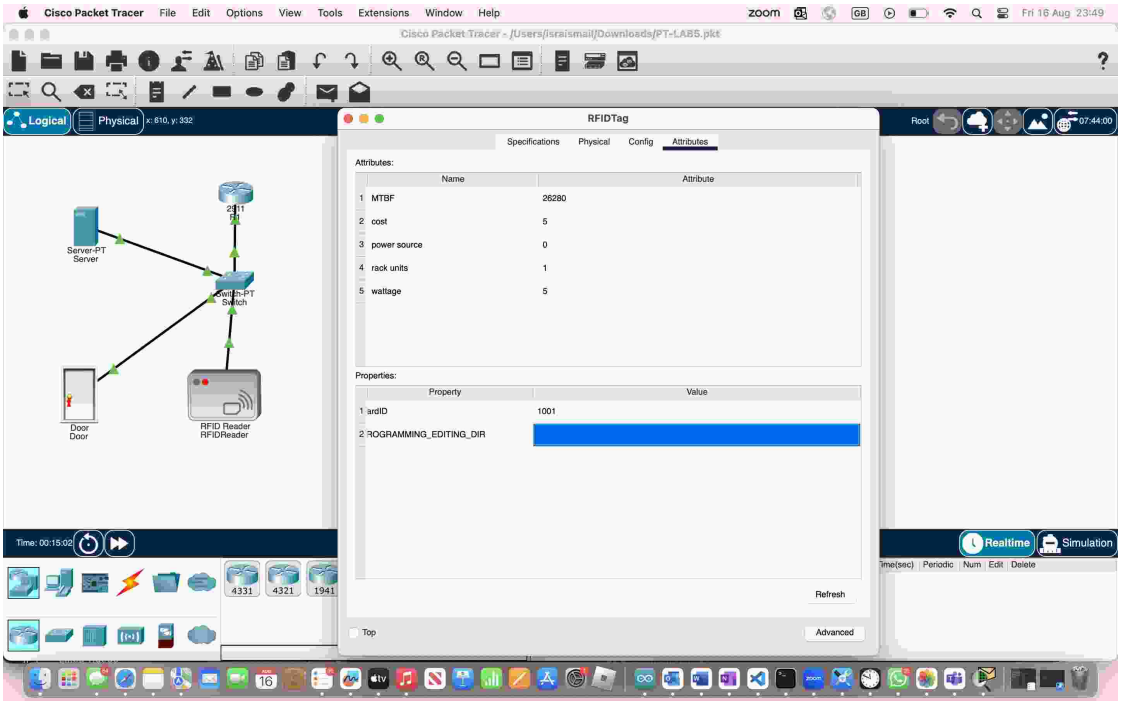


Figure 5.17: Reader turning green and door unlocking.

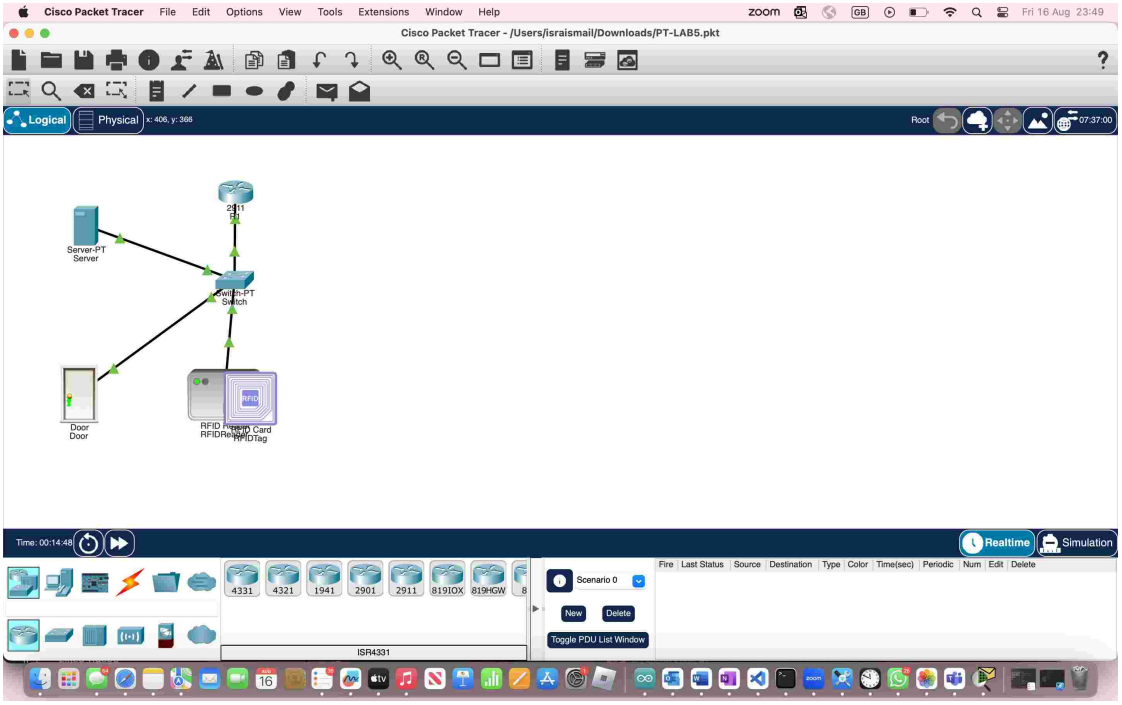


Figure 5.18: Demonstrating RFID validity transitions.

H. Add Bluetooth Devices (Music Player, Speaker)

21. **Add Music Player and Speaker:**
In **End Devices** → **Home Devices**, place a *Music Player* (rename it `MusicPlayer`) and a

Bluetooth Speaker (rename it `BSpeaker`).

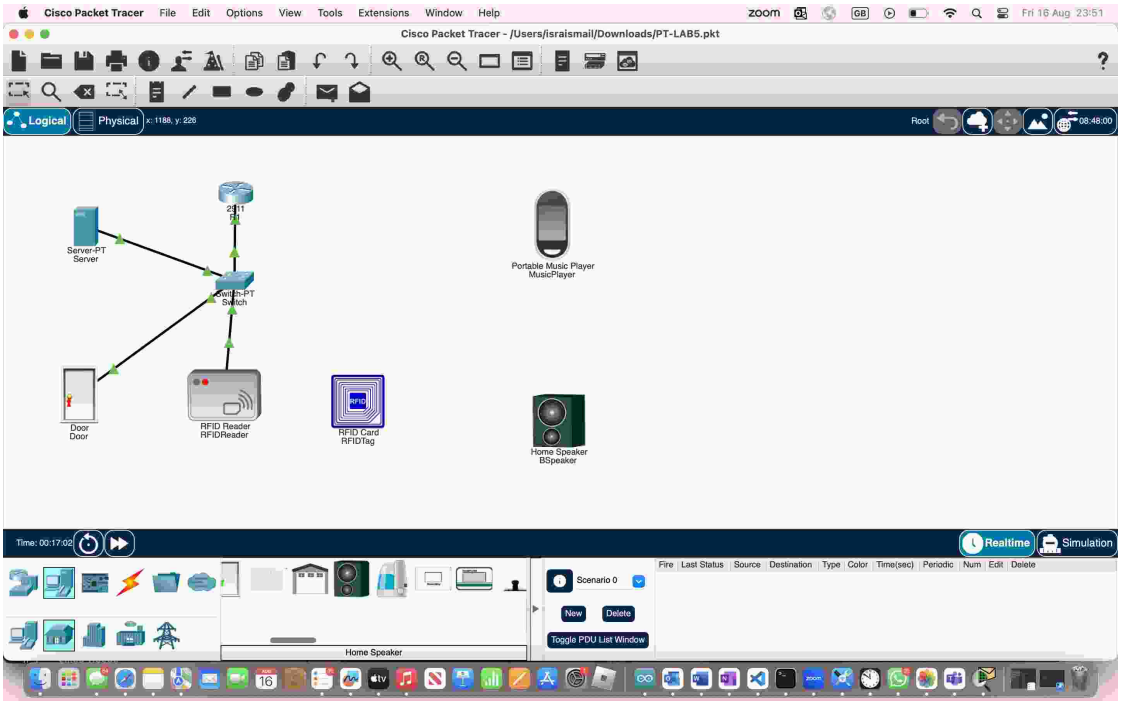


Figure 5.19: Adding Music Player and renaming it for clarity.

22. **Configuring the Music Player (Bluetooth):**

- **Wireless0** tab: Uncheck *Port Status* to disable Wi-Fi.
- **Bluetooth**: Enable *Port Status* so it can pair.

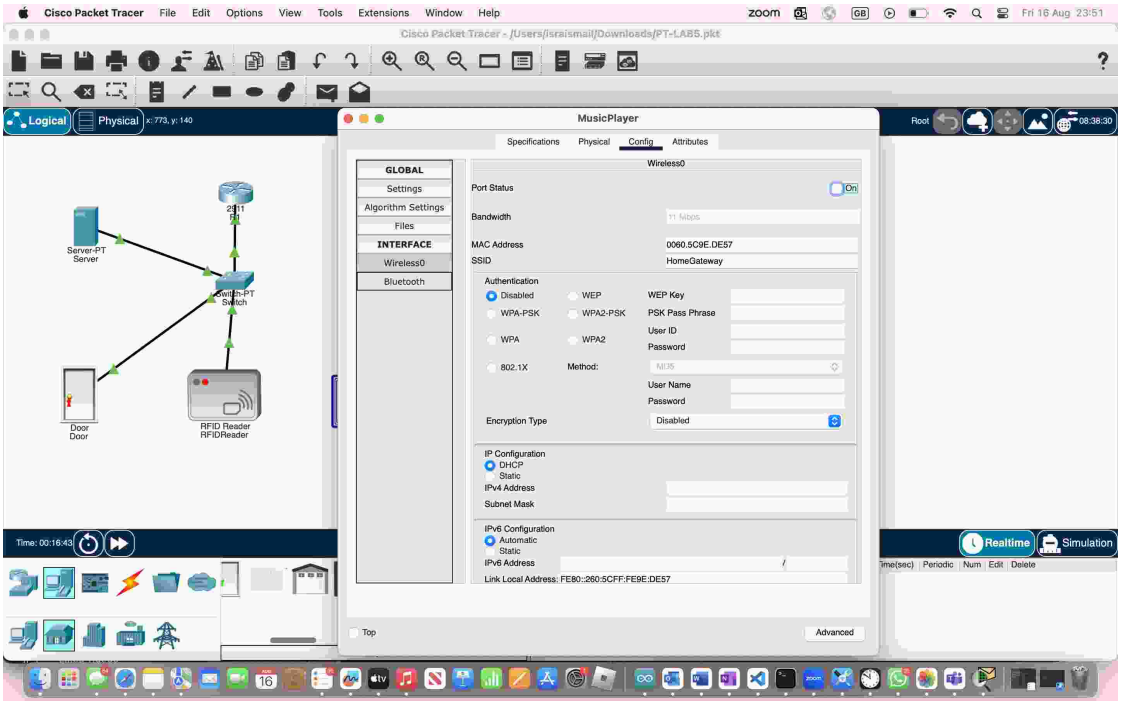


Figure 5.20: Disabling Wi-Fi, enabling Bluetooth on Music Player.

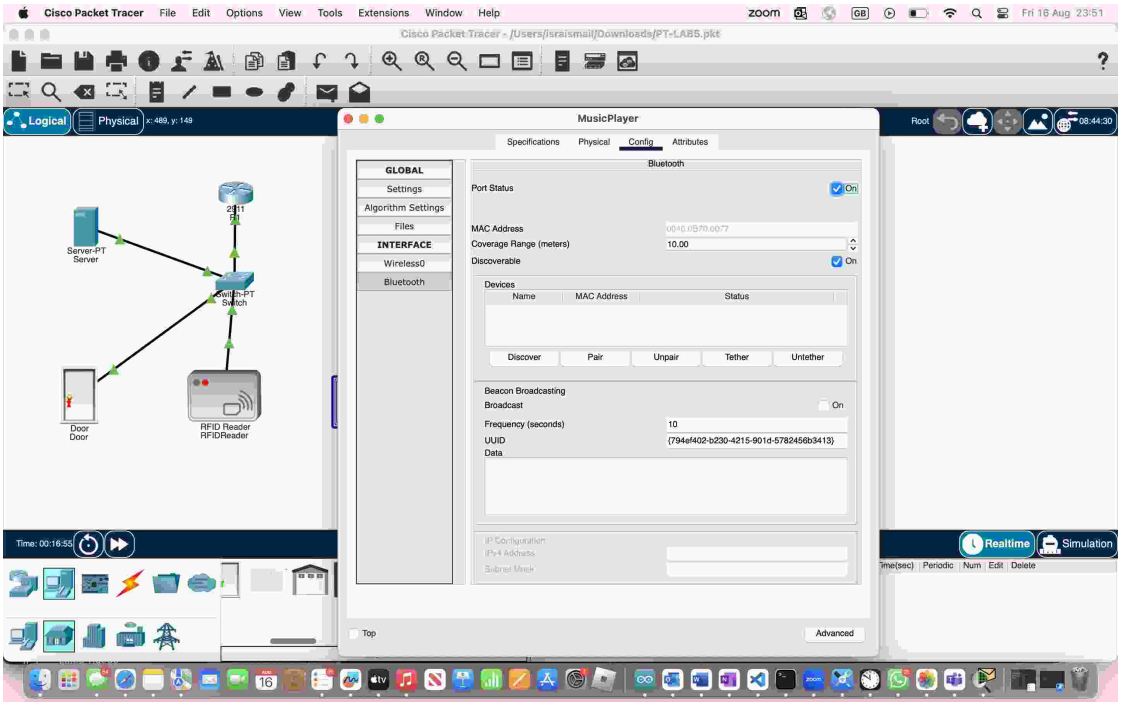


Figure 5.21: Ensuring the Music Player relies solely on Bluetooth.

23. **Configuring the BSpeaker:**
    Similarly, disable **Wireless0** and enable **Bluetooth** in the *Config* tab.
24. **Discover and Pair:**
    In the BSpeaker's **Bluetooth** page, click `Discover` to find the MusicPlayer, then `Pair`.

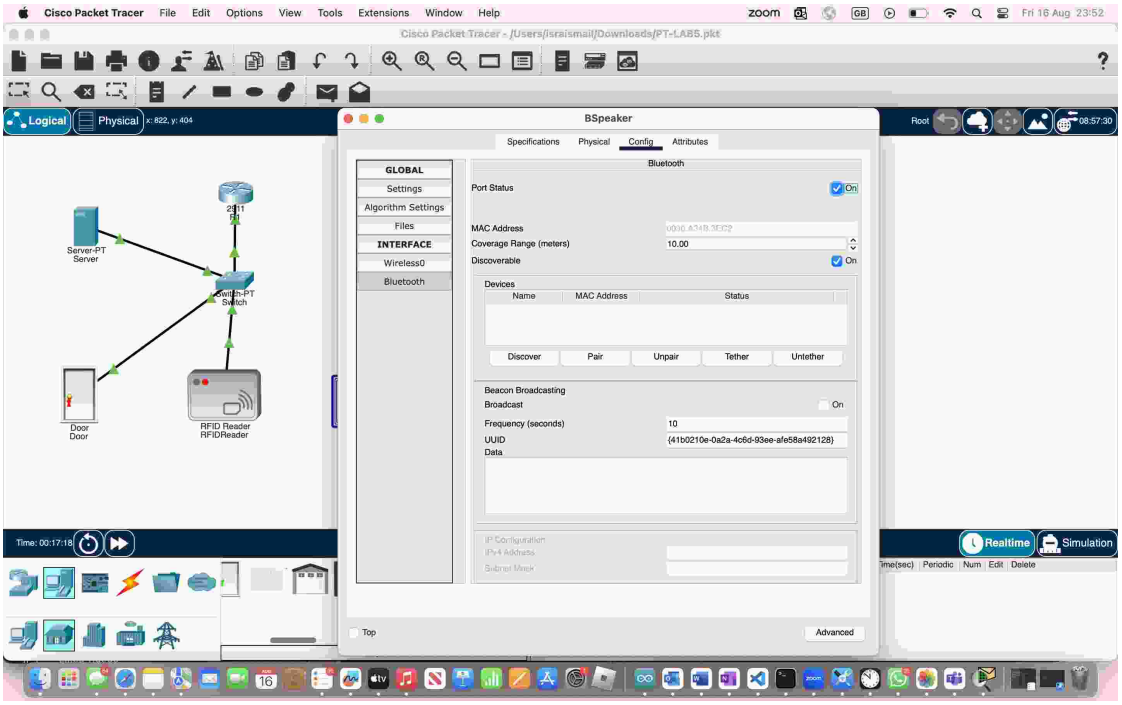


Figure 5.22: Discovering and pairing with MusicPlayer over Bluetooth.

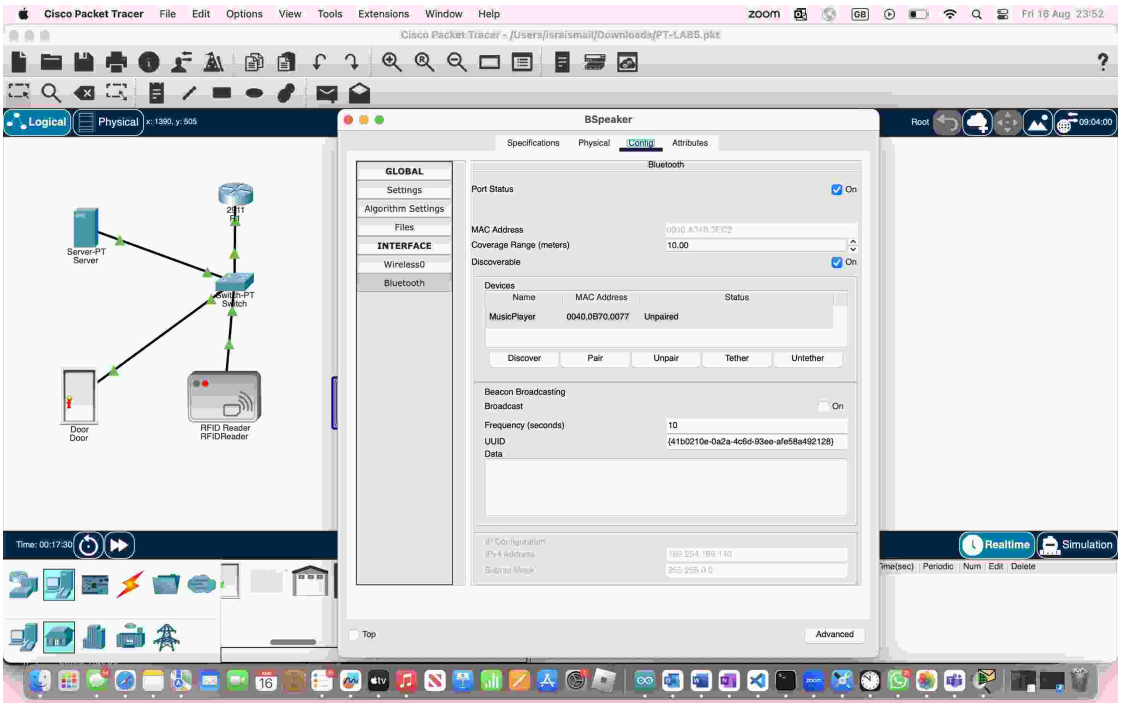


Figure 5.23: MusicPlayer discovered, ready for pairing.

25. **Paired State:**
    The workspace should show a Bluetooth connection between `MusicPlayer` and `BSpeaker`.

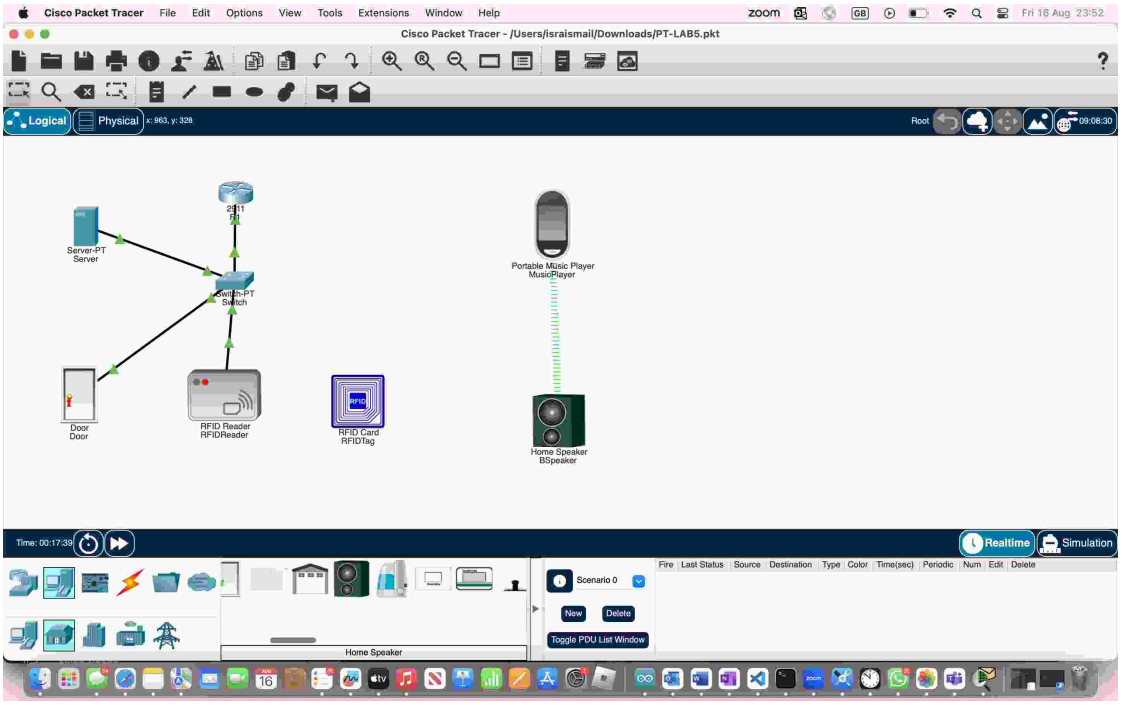


Figure 5.24: Successful Bluetooth link between MusicPlayer and BSpeaker.

I. Add Home Gateway and Smartphone

26. **Place a HomeGateway:**
    In **Network Devices** → **Wireless Devices**, add `HomeGateway` and use a cable to connect it to the switch.
27. **Add a Smartphone:**
    In **End Devices**, select `Smartphone` and place it on the workspace.

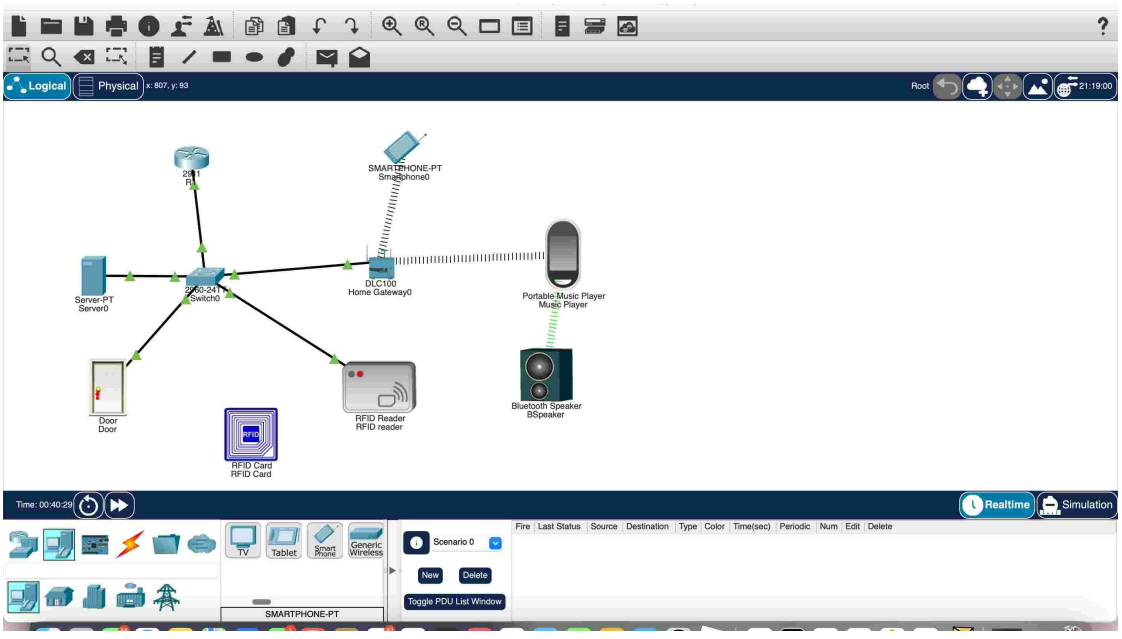


Figure 5.25: Smartphone added to the topology.

28. **Configure the Smartphone:**

    - **Config** tab → **Bluetooth**: Disable Port Status.
    - **3G/4G Cell**: Disable Port Status.
    - **Wireless0**: Set `SSID` to “HomeGateway”, enable Port Status.

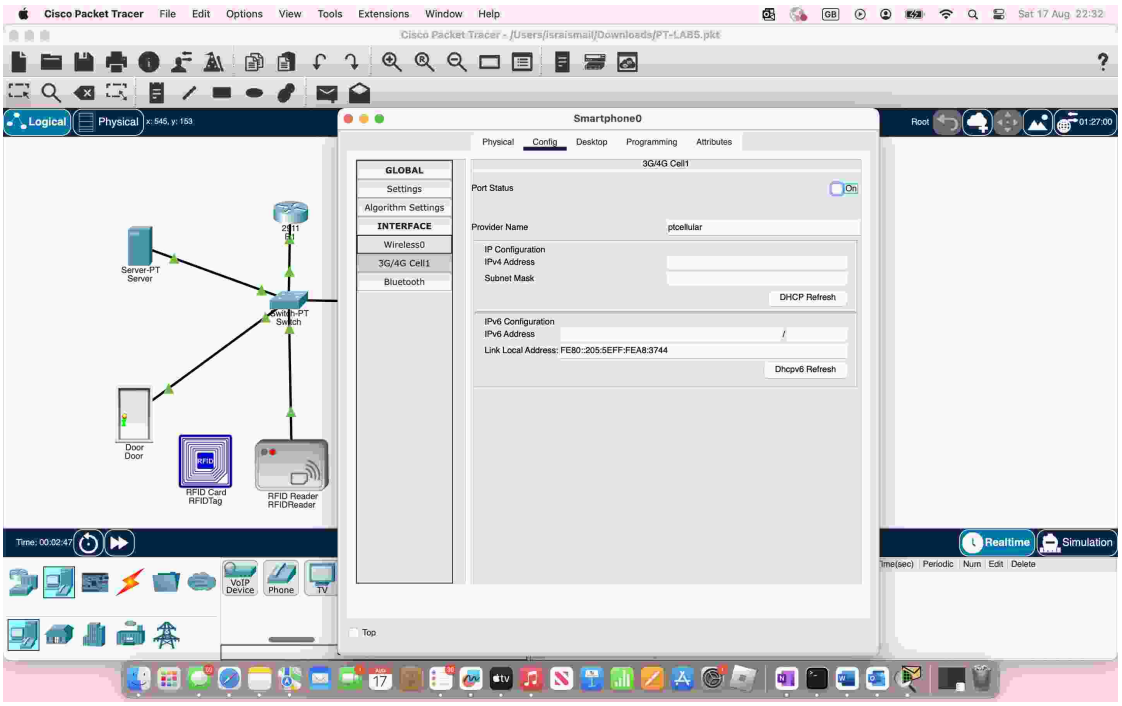


Figure 5.26: Smartphone config: turning off Bluetooth/cellular, turning on Wi-Fi (HomeGateway).

J. Test IoT Control via Smartphone

29. **Enable Wireless for MusicPlayer (Optional):**
    If you want the MusicPlayer to appear under the server, enable `Wireless0` or repeat the *DHCP + Remote Server* steps from earlier. (If purely Bluetooth, it won’t automatically show.)

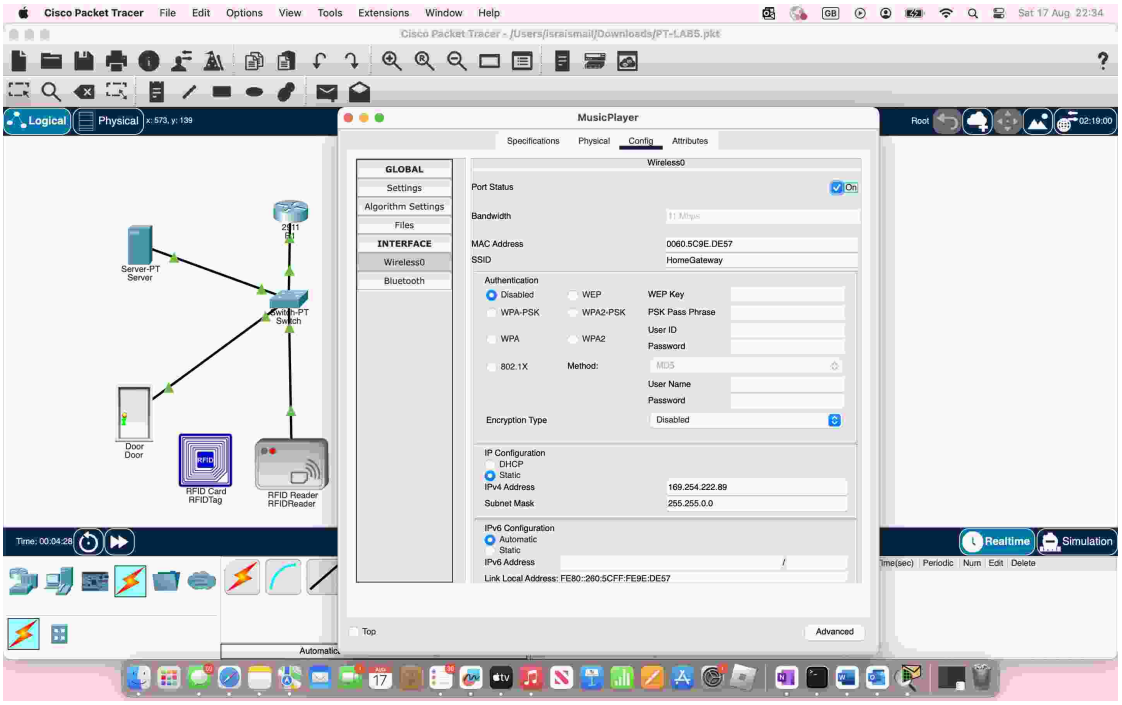


Figure 5.27: Turning on Wi-Fi for MusicPlayer to talk to server, if desired.

30. **Configure the Light (Example):**
    If you have a Light device, set DHCP and *Remote Server* to 192.168.1.15, user “admin”, pass “admin.”

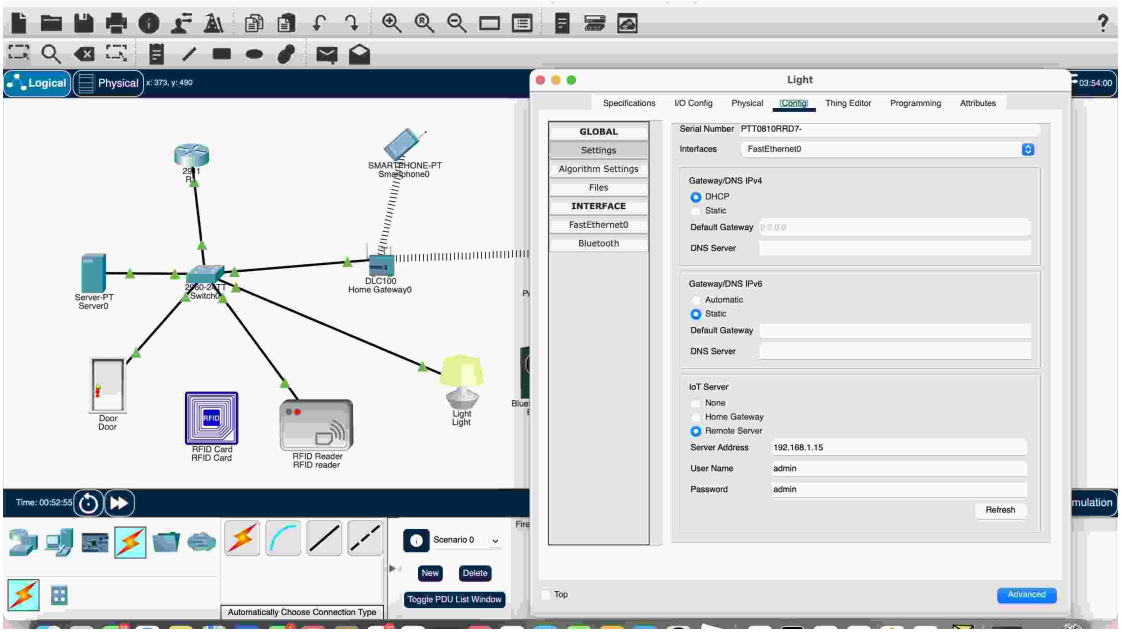


Figure 5.28: Registering the Light with the server.

31. **MusicPlayer to Server:**
    Repeat the same steps:
    - DHCP for IP

- `Remote Server` = 192.168.1.15, user: admin, pass: admin

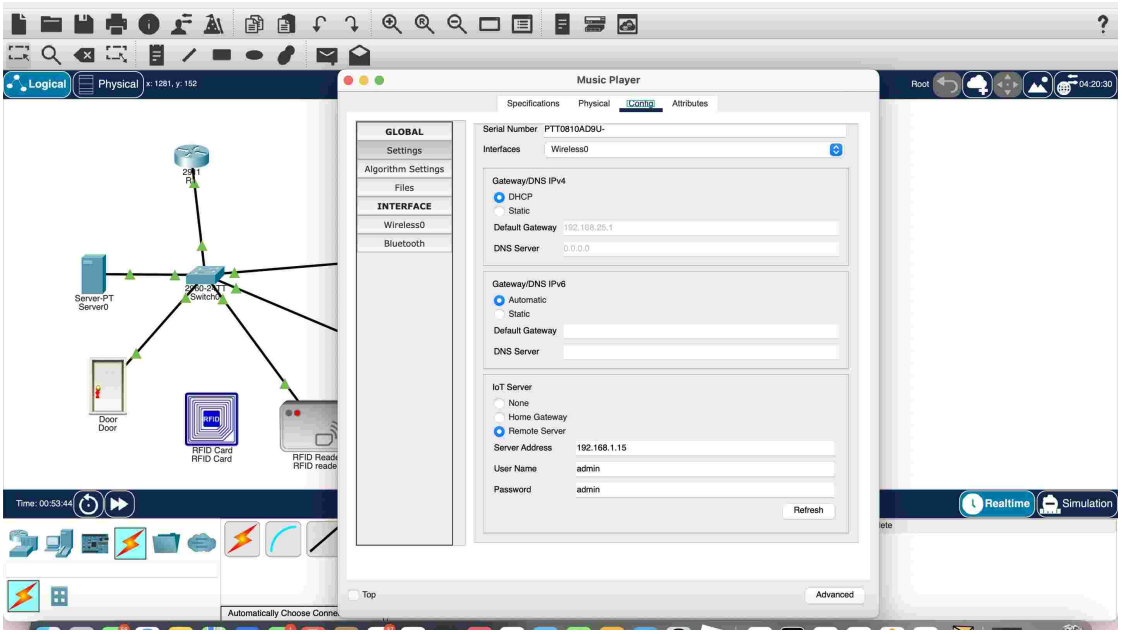


Figure 5.29: Connecting MusicPlayer to server.

32. **Smartphone Web Browser:**
On the Smartphone → **Desktop** → **Web Browser**, enter `192.168.1.15`, credentials "admin" / "admin."

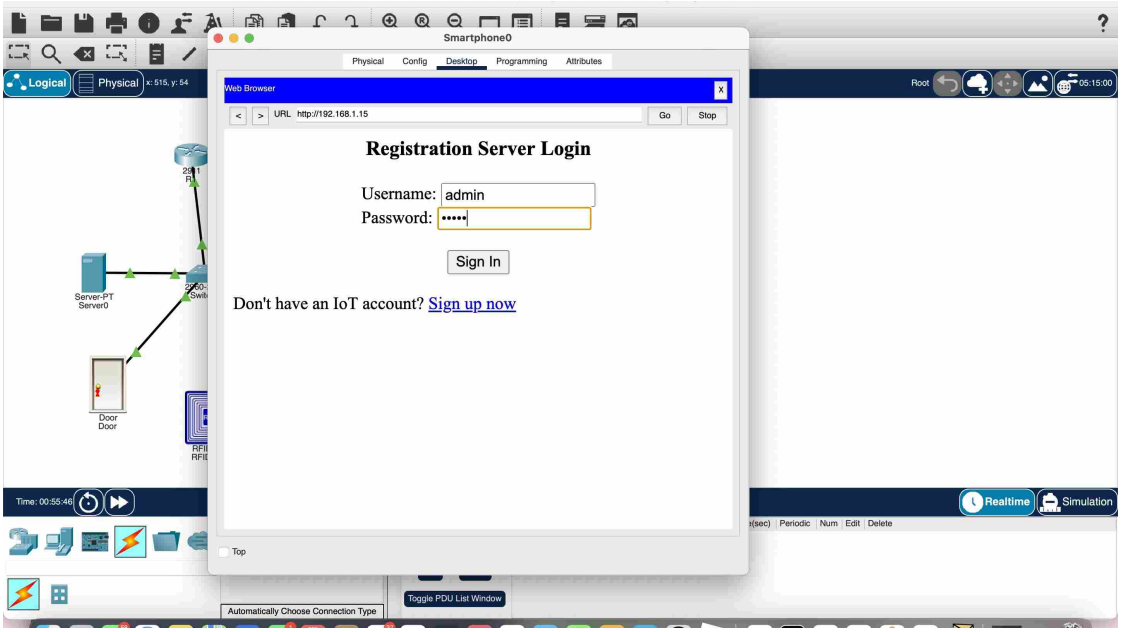


Figure 5.30: Smartphone browser connected to the IoT server.

33. **View IoT Devices:**
All registered devices appear (Door, RFIDReader, Light, MusicPlayer, etc.).

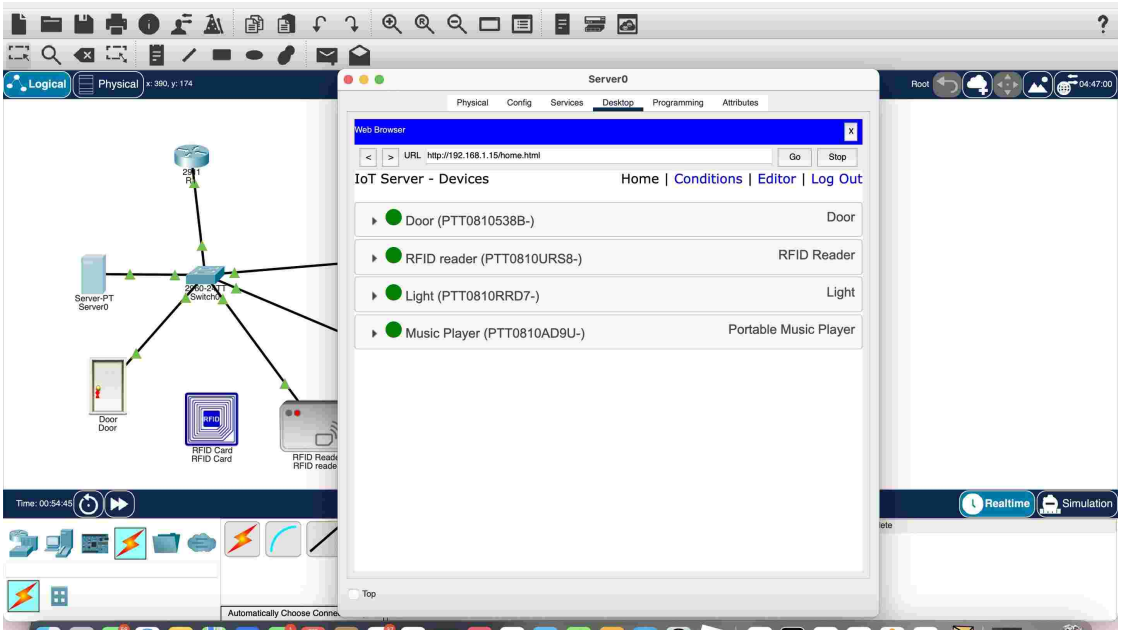


Figure 5.31: IoT devices connected to the server.

34. **Test Light or Music Playback:**
Toggle the Light on/off, or start the MusicPlayer. The Speaker and MusicPlayer are paired via Bluetooth, so audio is routed accordingly.

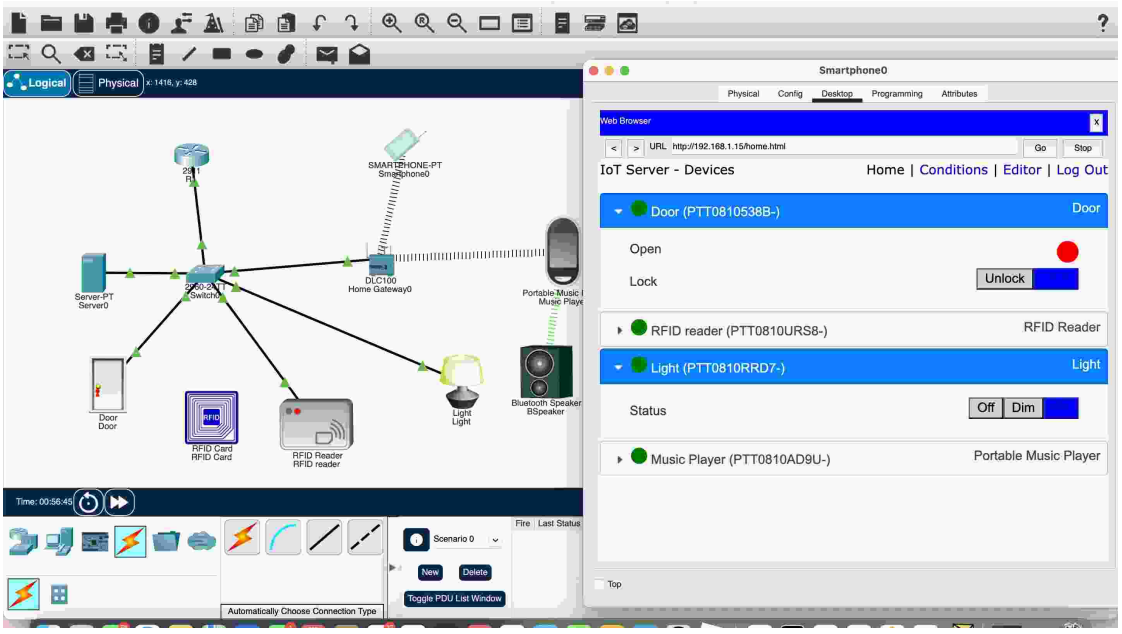


Figure 5.32: Light switched on from the smartphone interface.

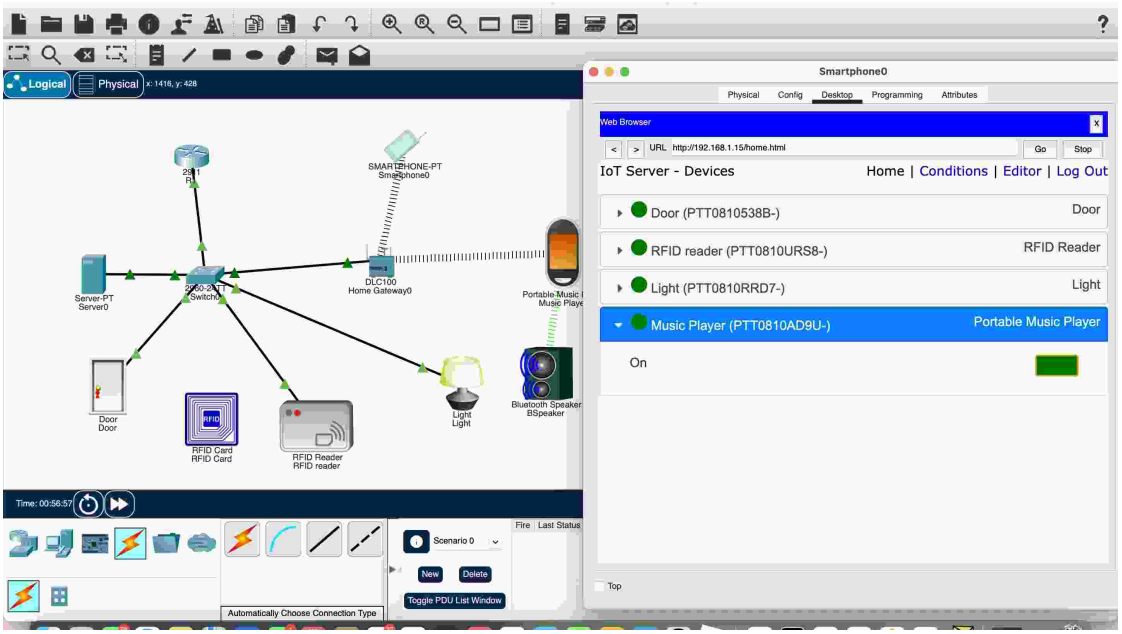


Figure 5.33: Music playing from the MusicPlayer to BSpeaker.

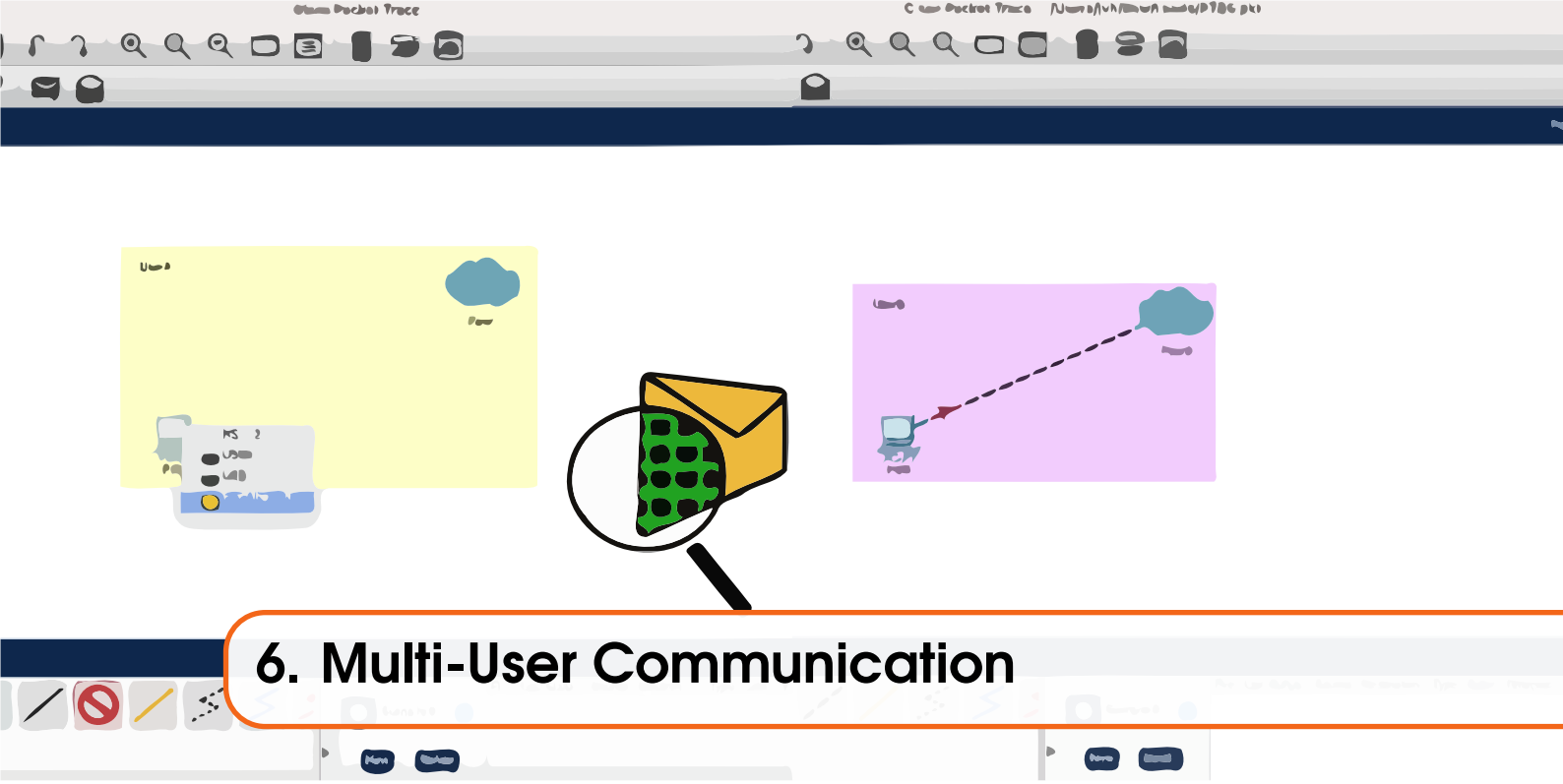
35. **Door Unlocking Logic (RFID):**
The door remains locked unless the RFID tag has the valid ID (1001). Once valid, the *DoorOpen* rule unlocks it.

Measuring Success

- **RFID Access:** Swiping the RFID tag with ID 1001 unlocks the door (green reader), while any other ID keeps it locked (red reader).
- **Bluetooth Pairing:** The MusicPlayer and BSpeaker are discovered, paired, and audio can be played.
- **Smartphone Control:** The phone's IoT Monitor (192.168.1.15) displays all devices (Door, Reader, Light, MusicPlayer) and allows toggling them.
- **No Errors in Registration:** The Packet Tracer console or IoT logs show successful DHCP/remote server connections, with no "bind port" or "unclosed string" errors.

Summary

You have built a **smart access control system** in Cisco Packet Tracer, incorporating *RFID-based unlocking* and *Bluetooth device management*. By configuring a router with DHCP, registering devices to the IoT server, and setting up rules for valid vs. invalid tag IDs, you established fine-grained access control. Additionally, by pairing a *MusicPlayer* and *BSpeaker*, you demonstrated how a smartphone can manage music playback alongside door locks—forming an integrated IoT environment that showcases Packet Tracer's versatile simulation capabilities.



6. Multi-User Communication

Introduction

This lab will show you how to **enable multi-user communication** between two separate Packet Tracer instances (often referred to as *User A* and *User B*). By creating a multi-user connection, you can simulate a distributed network environment where each user has their own Packet Tracer topology, yet both can pass traffic or share a remote connection.

Objective

- **Set up** multi-user communication between two Packet Tracer instances.
- **Simulate** network interaction across different users or separate topologies.

Lab Plan

A. **User A: Create a Topology**
B. **User B: Create a Second Topology**
C. **User B: Listen for Multi-user Connections**
D. **User A: Accept the Remote Connection**
E. **Connect PCs to the Remote Server**
F. **Assign IP Addresses on Each PC**
G. **Verify Connectivity (Ping)**

Required Software

- **Cisco Packet Tracer 8.x** (or newer)
- Basic knowledge of Packet Tracer interface for multi-user mode
- Two separate Packet Tracer instances (on one or multiple computers)

A. User A: Create a Topology

1. **Open Packet Tracer (User A):**

   - Launch **Cisco Packet Tracer** on the first user account (User A).
   - You should see a blank *Logical* topology workspace, ready for creating your network.

2. **Create a Rectangle (Optional Marker):**

   (a) Click the *Draw Shape* Tool (near the top-right toolbar in Packet Tracer, depending on your version).
   (b) Select a **Rectangle** shape and fill it with any color you like.
   (c) Right-click the rectangle and choose *Rename*, calling it `UserA` for easy identification.

   This optional marker helps visually separate or label User A's work area if you're collaborating in a multi-user environment.

3. **Add a PC:**

   (d) In the lower-left device selection, go to *End Devices*.
   (e) Drag a **PC** icon onto the canvas.
   (f) Rename it to `PCA` by clicking on the device, going to the **Config** tab (or right-click → *Rename*), and typing `PCA`.

4. **Add a Remote Network:**

   (g) Scroll to or click the **Multi-user Connection** category (in some Packet Tracer versions, it may appear as "Multiuser").
   (h) Select and place a **Remote Network** object in the workspace.

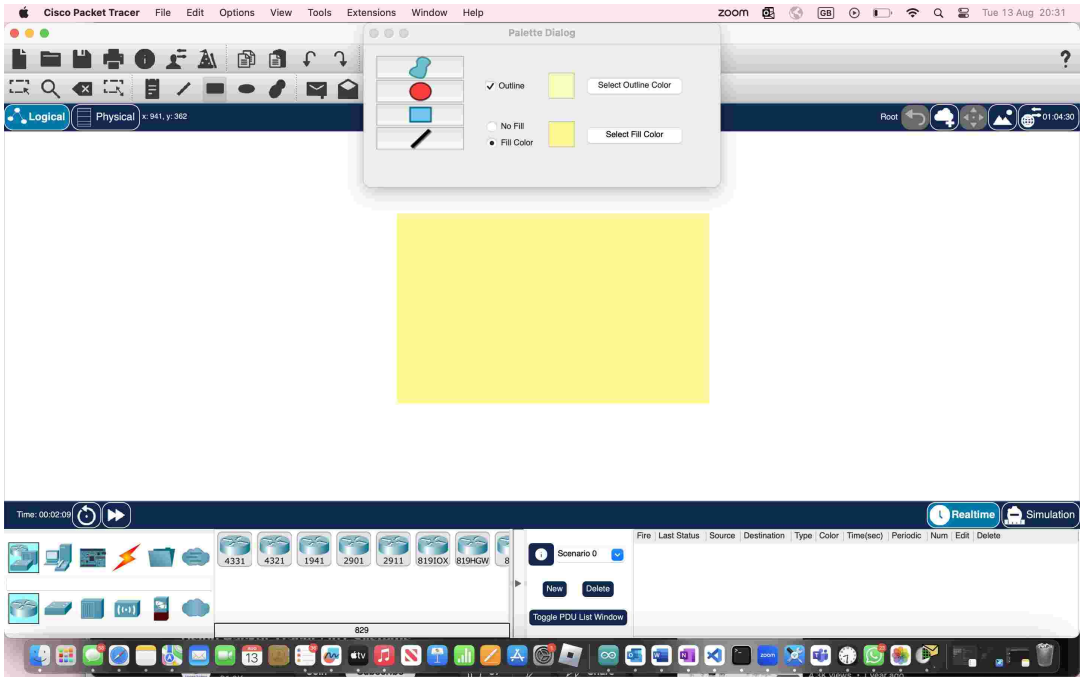


Figure 6.1: Placing a Rectangle and PC for User A's topology.

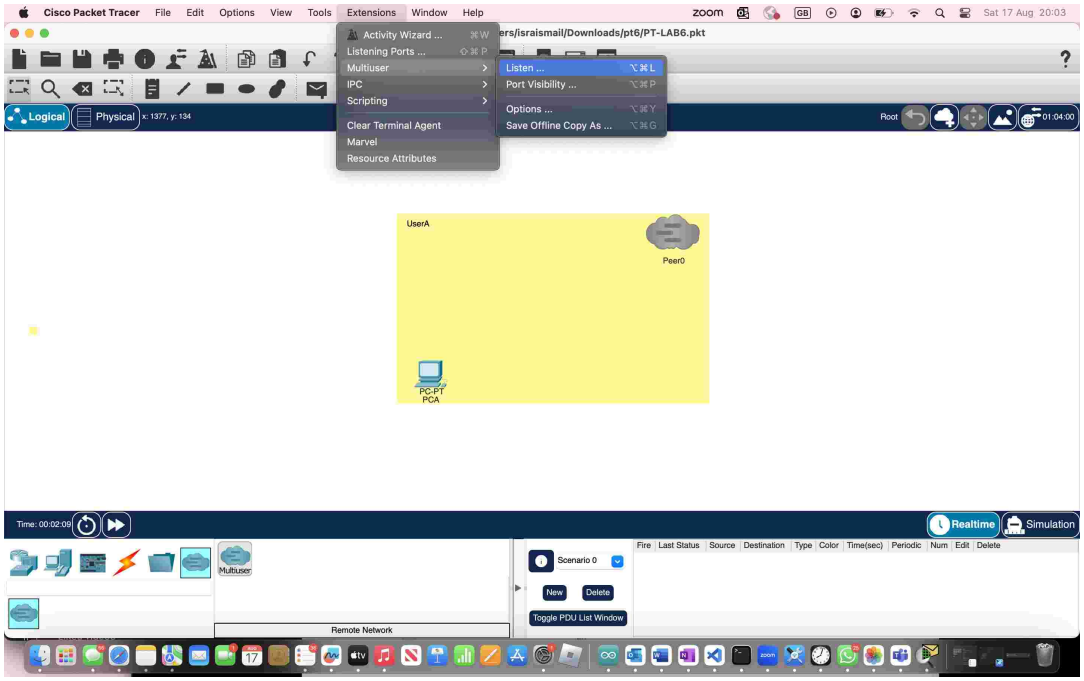


Figure 6.2: Adding a Remote Network object for User A.

— Why a Remote Network Object?.

- **Multi-User Mode:** Packet Tracer's remote network object is specifically used for connecting two or more Packet Tracer instances (e.g., User A and User B) over a shared topology.
- **Labeling Your Area:** By drawing a rectangle named `UserA`, you visually delineate your portion of the simulation if you're collaborating with others in real time.
- **Organized Workflows:** Keeping your devices (like `PCA`) and a remote network reference in one area ensures you can quickly link to User B's topology in the next steps.

B. User B: Create a Second Topology

5. **Open Packet Tracer (User B):**
   - Launch **Cisco Packet Tracer** on the second user account (User B).
   - You should see another blank *Logical* workspace, just as in User A's setup.
6. **New Rectangle:**
   (a) Use the **Draw Shape** Tool (found in the same toolbar as before) to create a rectangle of a different color than User A's.
   (b) Right-click the rectangle and select **Rename**, calling it `UserB`.
   This visually designates your portion of the multi-user topology as belonging to User B.
7. **Add a PC:**
   (a) Under *End Devices* in the lower-left device selection, drag a **PC** into the workspace.
   (b) Rename it `PCB` by clicking on the device and going to the **Config** tab (or right-click → *Rename*).
8. **Add a Remote Network:**
   (a) Scroll to the *Multi-user Connection* category (sometimes labeled "Multiuser").

(b) Drag a **Remote Network** object onto your workspace.

— Why Create a Separate User B Topology?.

- **Multi-User Collaboration:** Each user can build and manage their portion of the network independently, then link them together later.
- **Clarity of Ownership:** By labeling the rectangle `UserB` and naming your PC `PCB`, it's easy to see which devices belong to User B if you're collaborating in real time.
- **Scalable Design:** You can expand your own network—adding routers, switches, or servers—and then connect to User A's network when ready.

C. User B: Listen for Multi-user Connections

9. **Open Listen Mode:**
   In **User B**'s Packet Tracer instance:
   (a) Click on the **Extensions** menu (usually at the top of the Packet Tracer window).
   (b) Hover over **Multiuser** and select **Listen**.
   (c) A prompt or dialog box appears. Choose:
   - *Prompt for both existing and new remote networks*

   (d) Keep the default password as `cisco` (unless you wish to change it) and click **OK**.
   This puts **User B**'s Packet Tracer into a "listening" state, allowing **User A** (or others) to initiate remote connections.
10. **Configure the Remote Network Icon:**

    (a) In **User B**'s workspace, click on the *Remote Network* object you placed earlier.
    (b) In the object's configuration, change the connection type from *Incoming* to *Outgoing*.
    (c) Enter the password `cisco` (or the one you specified in the Listen Mode prompt).
    (d) Click **Connect** to finalize this part of the setup.

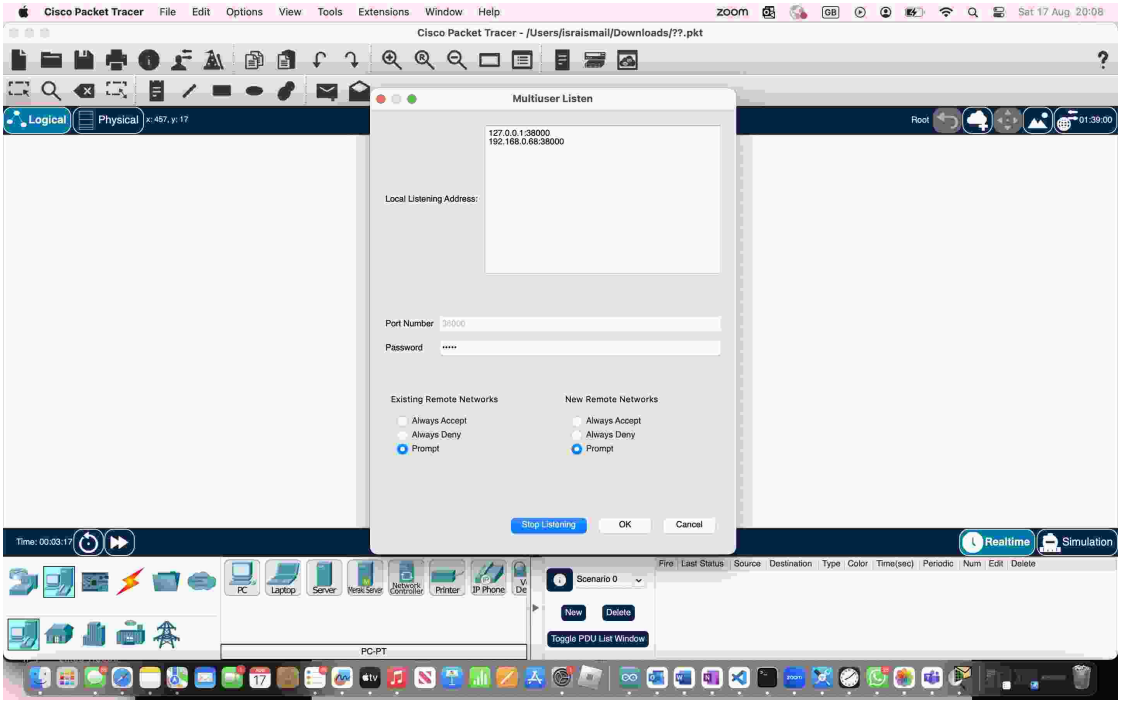


Figure 6.3: User B listening for multi-user connections.

— Why Listen Mode?.

- **Multi-User Collaboration:** By enabling "listen," **User B**'s Packet Tracer session awaits an incoming request from **User A** to link their topologies.
- **Prompting for Both Existing and New Networks:** Ensures that if a new remote network tries to connect, or an existing one changes settings, **User B** is asked for confirmation each time—enhancing control and security.
- **Same Password:** Both sides must use the same multi-user password (default `cisco` here) for the connection to succeed. This keeps unauthorized users from connecting unannounced.

D. User A: Accept the Remote Connection

11. **Pop-up Connection Request:**
    On **User A**'s Packet Tracer instance, a prompt appears, such as:
    "A peer (Guest at (1)) wants to make a new remote network connection to you. Do you want to accept this connection?"
    Click **Yes** to grant permission. This links **User B**'s "listening" session to your own topology.
12. **New Remote Server Appears:**
    After accepting the request, you should see a new *Remote Server* icon automatically appear in **User A**'s workspace. This object is Packet Tracer's way of representing a successful multi-user connection.
13. **Delete the Original Remote Network (If Needed):**

    - Keep the newly created *Remote Server* (the one that just showed up after accepting the connection).
    - Remove the old "Remote Network" object (the one you initially placed in **User A**'s topology), since it's no longer needed or is not connected.

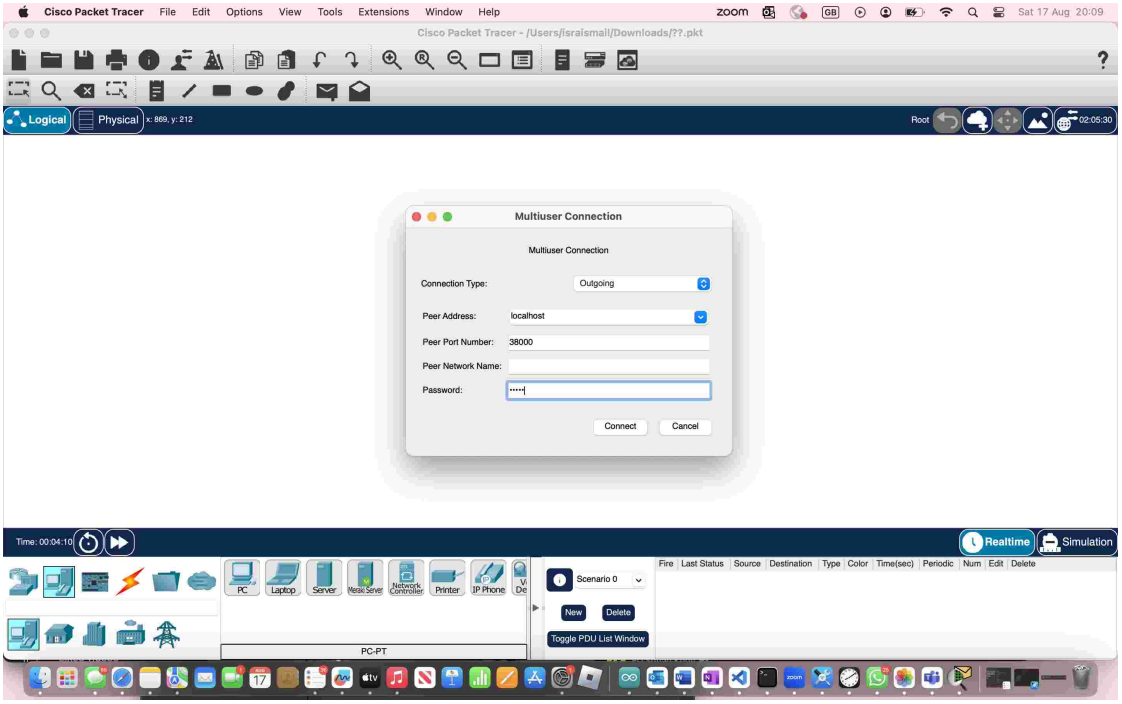


Figure 6.4: User A receiving a new remote connection prompt.

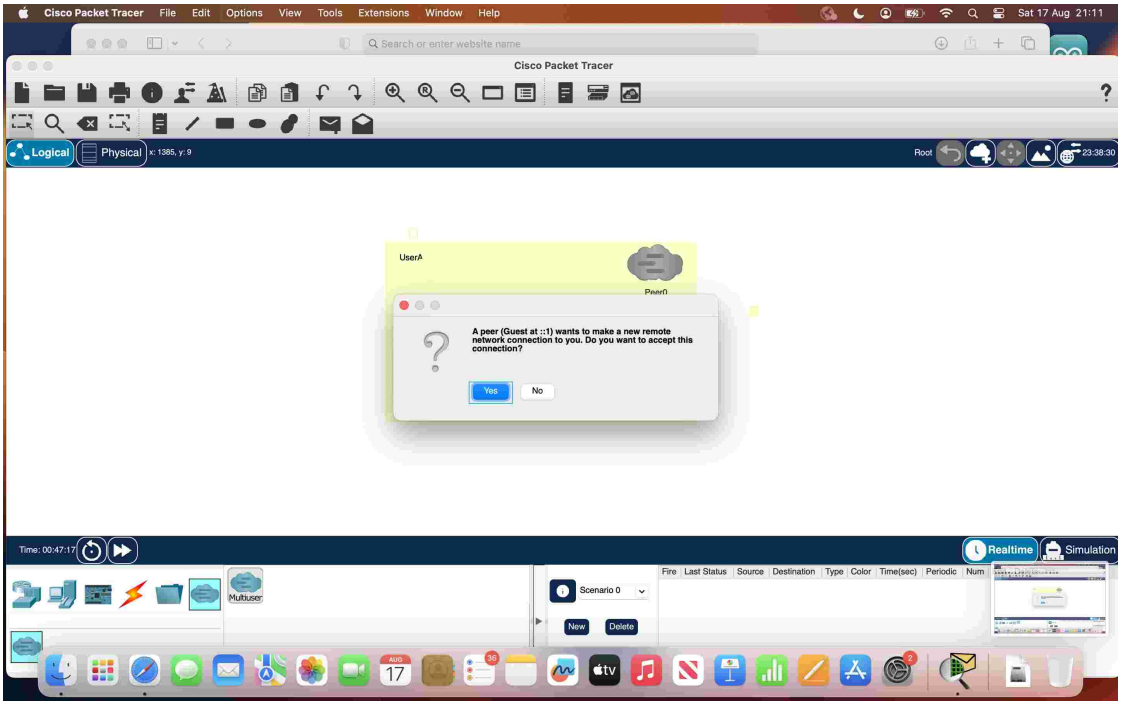


Figure 6.5: Accepting the multi-user connection creates a new Remote Server object.

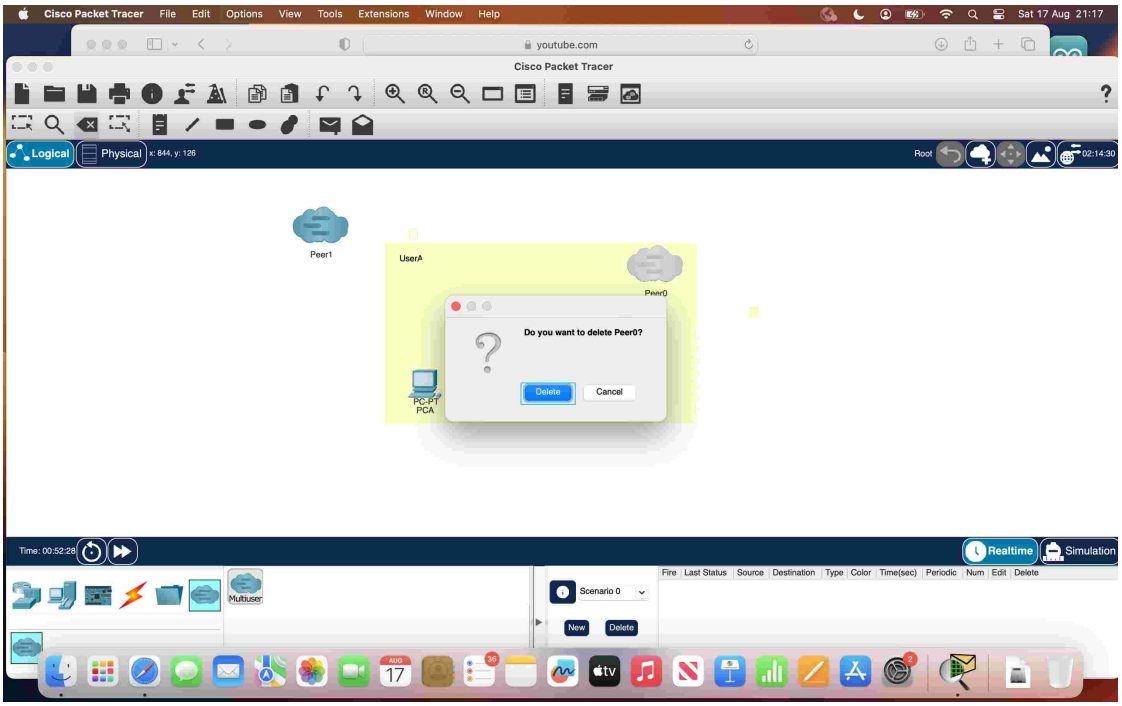


Figure 6.6: Removing the old unused Remote Network object.

— **Why Remove the Original "Remote Network"?.**

- **Auto-Created Remote Server:** Once you accept the connection from **User B**, Packet Tracer generates a *Remote Server* automatically in your workspace. This new object effectively replaces the initial "Remote

Network" placeholder you added.

- **Prevent Confusion:** Having multiple "Remote Network" objects can be confusing. Removing the unused one streamlines your topology, ensuring you only keep the active multi-user connection.
- **Clear Roles:** The new *Remote Server* is now your link to **User B**'s topology, while the leftover "Remote Network" no longer serves any purpose.

E. Connect PCs to the Remote Server

14. **In User A:**
    (a) Select a **Copper Cross Cable** from the *Connections* toolbar (often at the bottom-left).
    (b) Connect **PCA**'s *FastEthernet0* port to one of the available *link ports* on the newly created *Remote Server*.
    (c) Wait a moment to see if link lights appear or turn green, indicating a successful cable connection.
15. **In User B:**
    (a) Also choose a **Copper Cross Cable**.
    (b) Connect **PCB**'s *FastEthernet0* port to another *link port* on the same *Remote Server* object.
    (c) Observe the link status—if both ends of the connection are active, you'll see link lights or a green status.

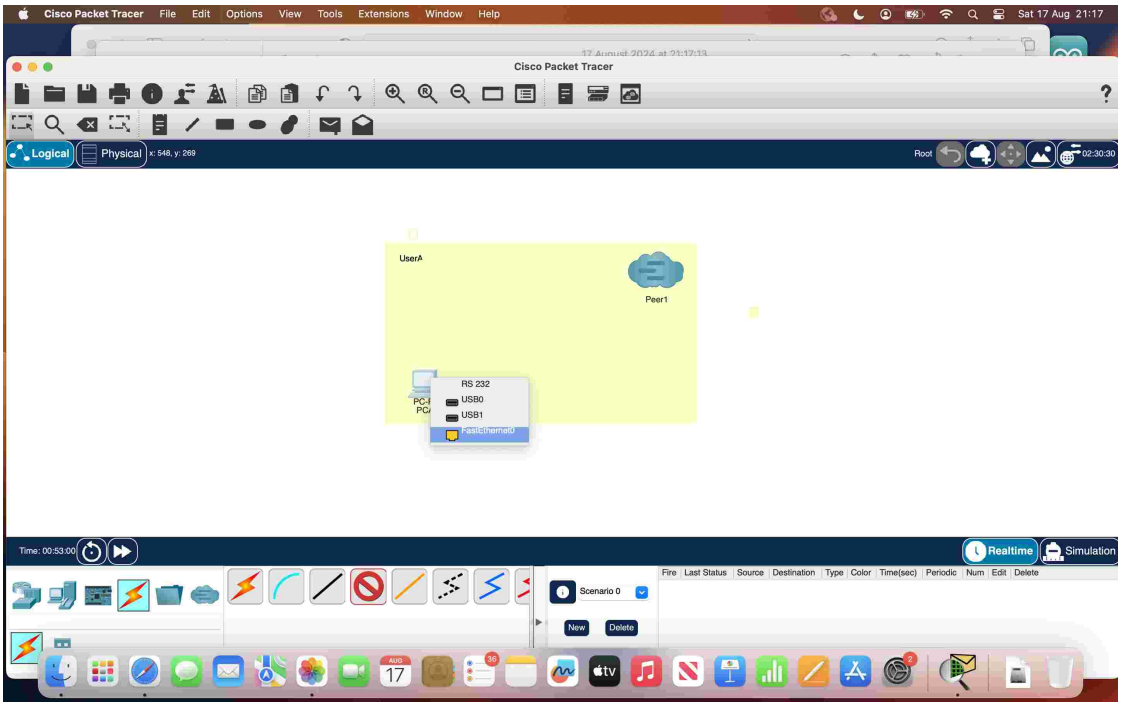


Figure 6.7: User A connecting PCA to the newly created remote server.

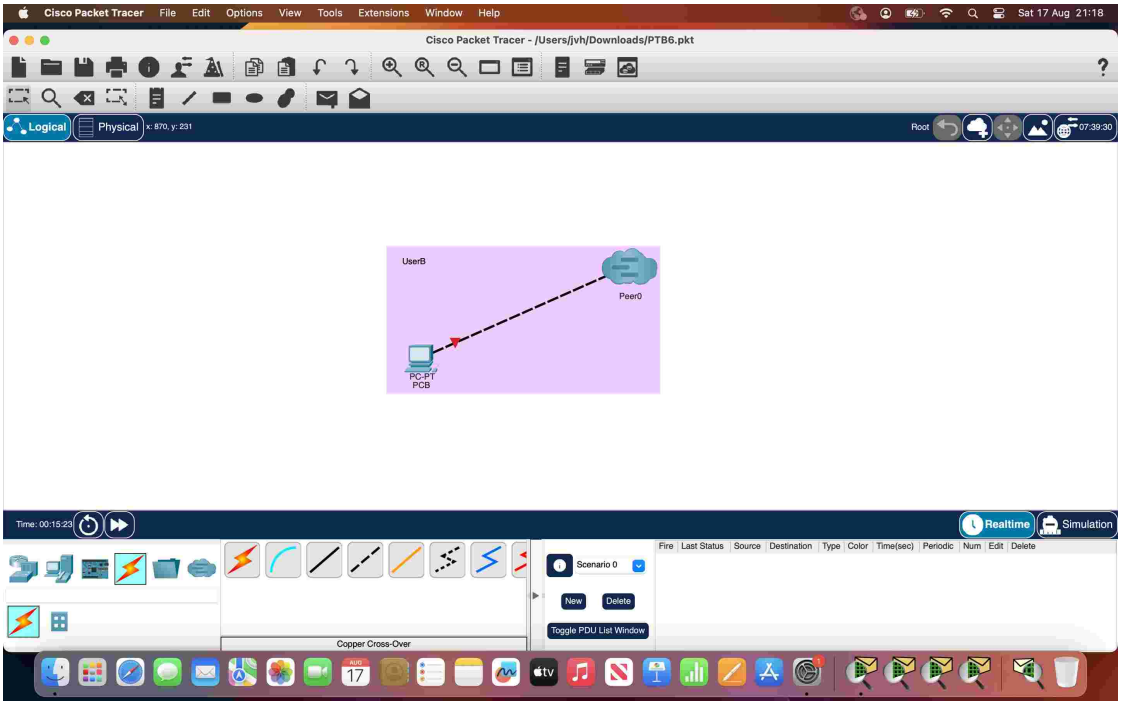


Figure 6.8: User B connecting PCB to the same remote server link.

— Why Use a Copper Cross Cable?.

- **Multi-User Connection Object:** Packet Tracer often requires a *Cross Cable* to link physical ports to the *Remote Server* interface (though *Automatically Choose Connection Type* can work, too).
- **Virtual "Link Ports"**: The *Remote Server* object simulates a "gateway" for User A and User B's topologies, so each PC needs to connect with a cable type that Packet Tracer recognizes for direct host-to-host or host-to-virtual server connections.
- **Confirm Link Lights:** If you don't see the link lights, re-check the ports (e.g., `FastEthernet0`), or try *Switching* to another cable type if Packet Tracer misidentifies the connection.

F. Assign IP Addresses on Each PC

16. **PCA Configuration (User A):**

    (a) **Open PCA's Desktop**: Click on PCA, then select *Desktop* → **IP Configuration**.
    (b) **Assign IP Details**:
        - **IP Address**: `10.0.0.1`
        - **Subnet Mask**: Let Packet Tracer auto-fill (likely `255.0.0.0` or `255.255.255.0`).
    (c) **No Gateway Needed**: Since we're directly testing connectivity between PCA and PCB over a *Remote Server* link, a default gateway is not strictly necessary unless you expand the network.
    (d) **Close Configuration**: Once set, exit the IP Configuration window.

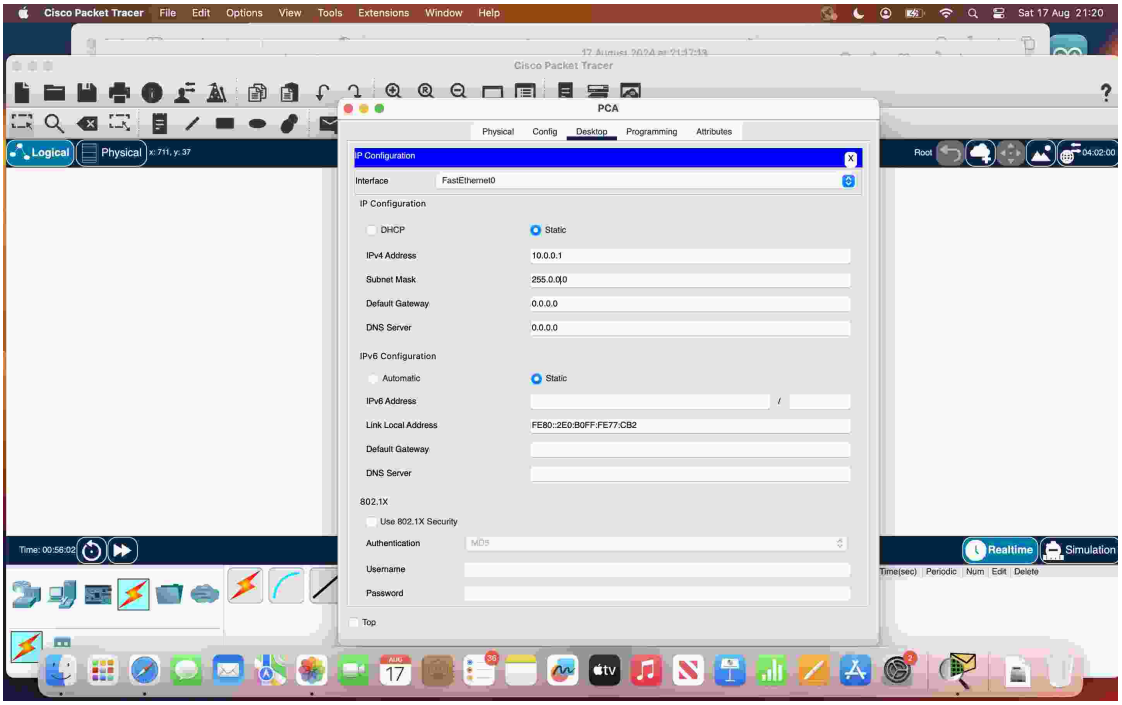


Figure 6.9: Assigning `10.0.0.1` to PCA.

17. **PCB Configuration (User B):**

    (a) **Open PCB's Desktop**: Click on `PCB`, then go to *Desktop* → **IP Configuration**.
    (b) **Assign IP Details**:

    - **IP Address**: `10.0.0.2`
    - **Subnet Mask**: Same as `PCA`'s (e.g., `255.0.0.0` or `255.255.255.0`).

    (c) **No Gateway Needed**: Similar to `PCA`, if you're only testing basic cross-connectivity with `User A`'s PC, a gateway is not required unless you plan a larger routed network.
    (d) **Close Configuration**: After entering the IP information, exit the IP Configuration.

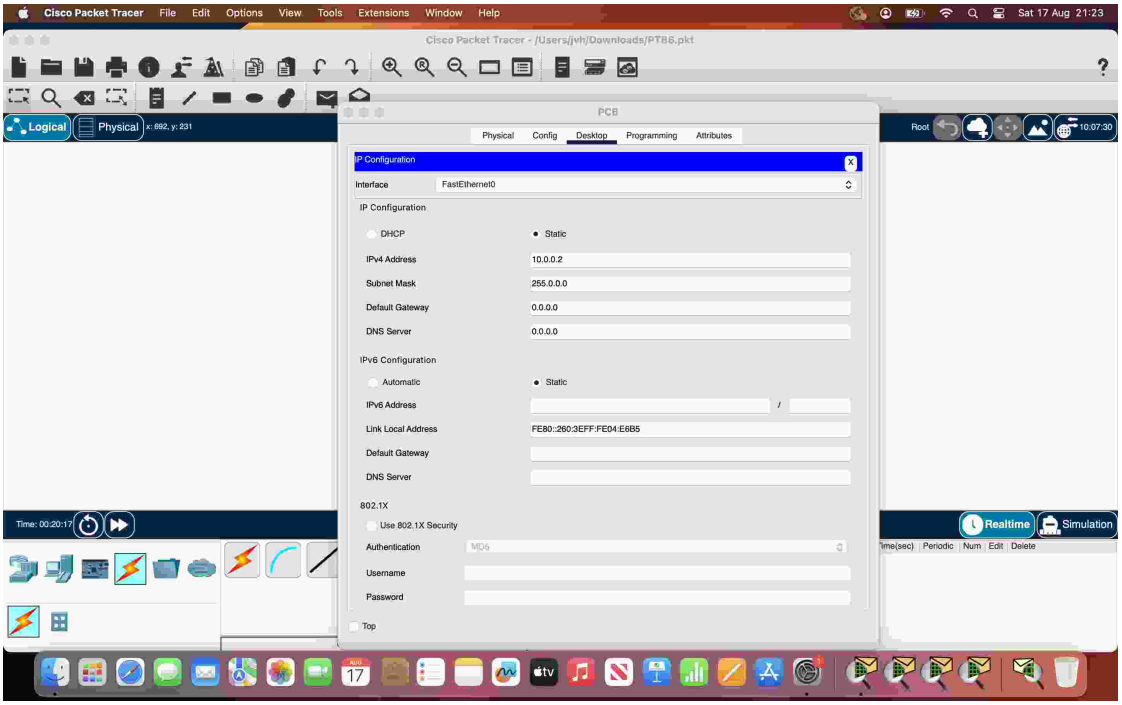


Figure 6.10: Assigning `10.0.0.2` to PCB.

— Why 10.0.0.x Addresses?.

- **Simple Private Network**: The `10.0.0.0/8` range is a common private IP space, ideal for point-to-point or lab scenarios.
- **Easy Memorization**: Short IP addresses like `10.0.0.1` and `10.0.0.2` are simple to recall and reduce typing errors when pinging or configuring devices.
- **No Overlaps**: Using a dedicated subnet for multi-user mode ensures it won't conflict with other subnets in your broader Packet Tracer labs.

G. Verify Connectivity (Ping)

18. **Ping from PCA to PCB:**
    (a) On `PCA`, open the **Command Prompt** from the *Desktop* tab.
    (b) Type:

```
ping 10.0.0.2
```

    (c) If you see replies like *Reply from 10.0.0.2: bytes=32 time<1ms TTL=128*, then multi-user communication is successfully established between **User A** and **User B**.

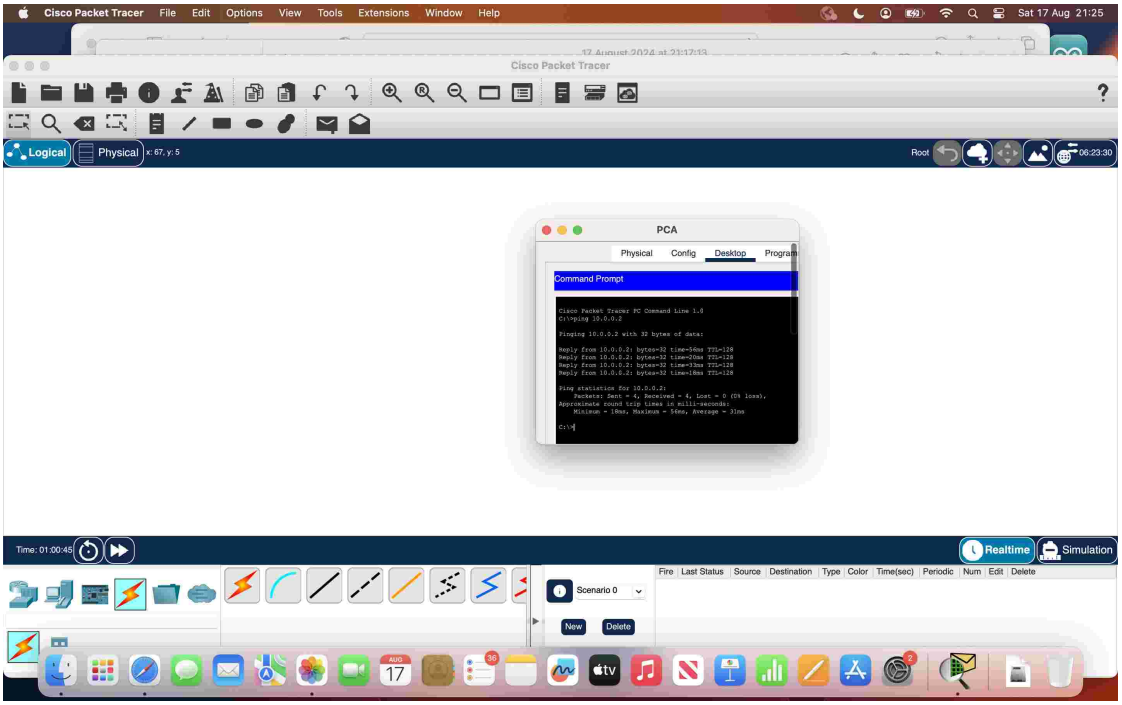


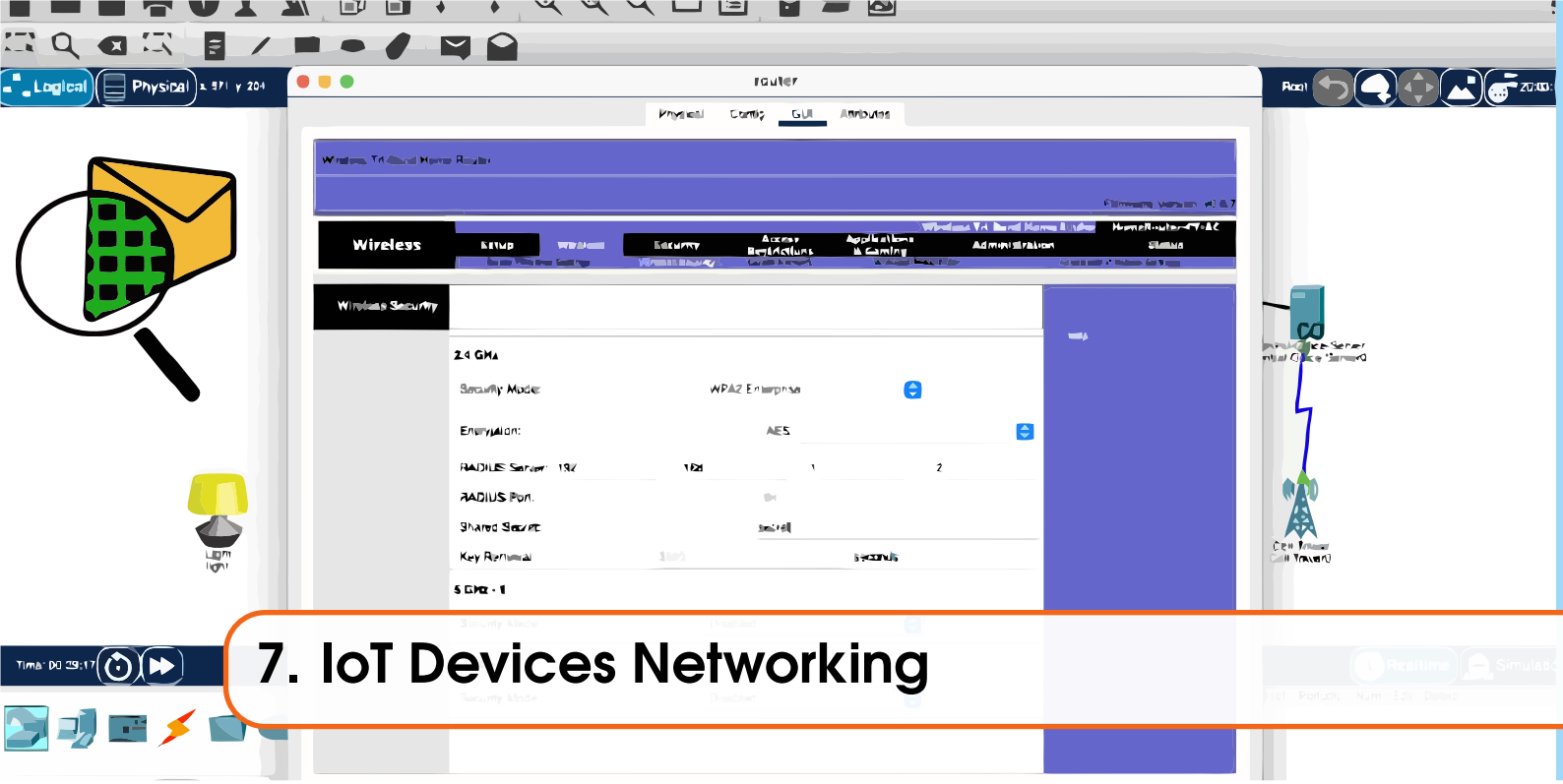
Figure 6.11: Successful ping from `10.0.0.1` (PCA) to `10.0.0.2` (PCB).

— **Summary of the Multi-User Connection.**

- **Physical Representation:** Each user builds a separate Packet Tracer topology (User A, User B). The *Remote Network* or *Remote Server* object links them.
- **IP Configuration:** Assigning `10.0.0.1` and `10.0.0.2` on respective PCs ensures both are on the same subnet.
- **Ping Test:** Verifies that the cross-connection in multi-user mode works as if both PCs were in the same LAN.
- **Further Exploration:** You can now exchange packets, run other protocols, or build more complex labs (e.g., routers, switches, IoT devices) and still test them collaboratively in real time.

Summary

You have successfully **configured multi-user communication** between two Packet Tracer instances (User A and User B). By *listening* on one side and *accepting* on the other, you created a shared remote server object that both PCs connect to. Using IP addresses on the 10.0.0.x subnet, you verified they can ping each other—demonstrating that both topologies are effectively joined into a single multi-user simulation. This setup allows you to simulate distributed networks or collaborative lab environments in real time.



7. IoT Devices Networking

Introduction

In this lab, you will create a **smart home network** in Cisco Packet Tracer using both *traditional networking devices* (router, switches) and *IoT devices* (fan, lamp, garage door). You will configure an IoT server for authentication (AAA), verify device connectivity (wired and wireless), and test remote control of the IoT devices through a laptop and smartphone.

Objective

- **Create a basic network topology** with IoT and traditional networking devices.
- **Configure the IoT server and devices** (AAA security, IP settings).
- **Test and ensure connectivity** of IoT devices using a laptop and smartphone.

Lab Plan

A. **Add Network Devices** (Switches, IoT Server, Fan, Garage Door, Lamp)
B. **Connect the Network Devices** (Switch-to-switch, router, external servers)
C. **Set Up the IoT Server** (Registration Server, AAA)
D. **Configure Switches and Router**
E. **Set Up Wireless (SSID, WPA2 Enterprise)**
F. **Laptop Setup** (testing network and IoT server)
G. **Smartphone Setup** (final device configuration and testing)

Required Software

- **Cisco Packet Tracer 8.x** (or newer)
- Familiarity with *Config* and *Services* tabs for IoT server settings
- Optional smartphone or laptop knowledge to test final connectivity

A. Add Network Devices

Step 1: Add Two Switches

1. **Open Cisco Packet Tracer:**
   Double-click the Packet Tracer icon on your computer or launch it from your applications. A blank *Logical* workspace should appear.
2. **Add Two Switches:**
   - In the lower-left device categories, click on *Network Devices* and then *Switches*.
   - Drag two **Switches** onto the workspace.
3. **Place the First Switch in the Center:**
   - Position the first switch roughly in the center, leaving space on either side for more devices.
   - This ensures you can easily connect devices on both the left (e.g., IoT Server) and right (e.g., IoT sensors or additional switches).

Step 2: Add IoT Devices

4. **From IoT Devices, Add a Fan, Garage Door, and Lamp:**
   - In the Packet Tracer categories, select *End Devices* or *IoT Devices*, depending on your version.
   - Drag a **Fan**, a **Garage Door**, and a **Lamp** onto the workspace.
5. **Position Them Near the First Switch:**
   - Arrange the fan, garage door, and lamp near your central switch for short, uncluttered cable runs.
   - Leave room for any additional sensors or devices you may add later.

Step 3: Add IoT Server

6. **From Servers, Add an IoT Server Near the First Switch:**
   - In *Network Devices* $\rightarrow$ *Servers*, select the **IoT Server**.
   - Place it on the workspace, ensuring it's relatively close to your main switch for simpler cabling.
7. **Purpose of IoT Server:**
   - This device will manage and authenticate all IoT devices, such as the fan, garage door, and lamp.
   - It can also provide additional services (e.g., DHCP, DNS) if needed, although you may configure a router for those tasks in more complex topologies.

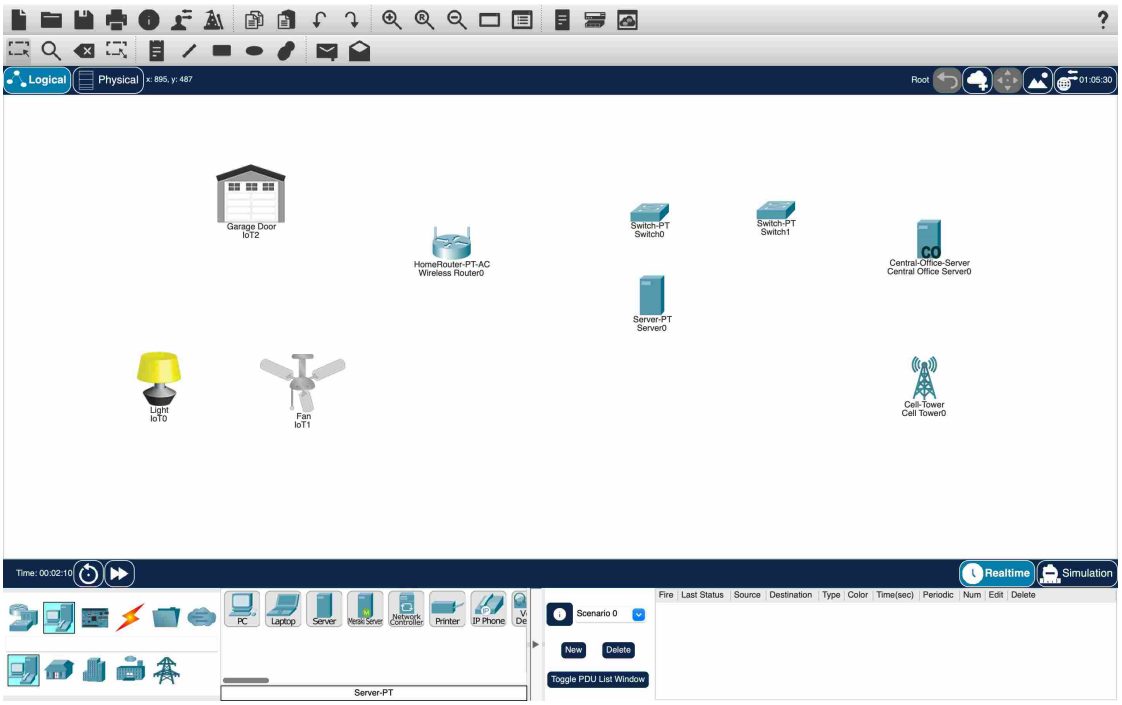


Figure 7.1: Switches and IoT devices (fan, garage door, lamp, IoT server).

— Why These Devices?.

- **Switches:** Allow multiple endpoints (IoT devices, servers) to connect locally on a LAN.
- **Fan, Garage Door, Lamp:** Simulate common smart-home or small-office IoT components that can be monitored or controlled remotely.
- **IoT Server:** Acts as a central "brain" for device registration, authentication, and potential automation logic (e.g., triggers/conditions for each device).

B. Connect the Network Devices

Step 1: Add Central Office Server and Cell Tower

1. **Add a Central Office Server and a Cell Tower:**
   - In Packet Tracer's *Network Devices* or *End Devices* (depending on the version), locate and place:
     - A **Central Office Server**
     - A **Cell Tower**
   - These represent external network infrastructure, simulating an ISP or mobile provider that your network may connect to for broader access.

Step 2: Connecting the Switches

2. **Use the Automatically Selected Cable:**
   - Click on the *Connections* icon (lightning bolt).
   - Choose "Automatically Choose Connection Type" so Packet Tracer picks the correct cable (usually a copper straight-through).
   - Click on the first switch, then the second switch, to link them together.
3. **Place the Second Switch on the Right:**
   - Position **Switch2** (or however you label it) on the right side of the workspace.

- This switch will connect to external devices such as the Central Office Server and Cell Tower.

Step 3: Add a Router

4. **Add a Router to the Workspace and Connect It:**

   - Drag a **Router** from the *Network Devices* → *Routers* category.
   - Use the cable tool again (either *Automatically Choose Connection Type* or *Copper Straight-Through*) to connect the router to the first switch.
   - Wait a moment to see if link lights turn green, indicating a successful connection.

Step 4: Connect IoT Server and Switches

5. **Link the IoT Server to the First Switch:**

   - If you placed the IoT Server near the first switch, select an empty port on the switch (e.g., `FastEthernet0/3`) and connect it to the server's `FastEthernet0`.
   - This ensures the IoT Server can communicate with any devices on Switch1.

6. **Connect the Second Switch to the Central Office Server, and Then to the Cell Tower:**

   - Use a *Copper Straight-Through* cable (or *Automatically Choose Connection Type*) to link Switch2 with the **Central Office Server**.
   - Connect the **Central Office Server** to the **Cell Tower**, simulating a path to external/-mobile infrastructure.

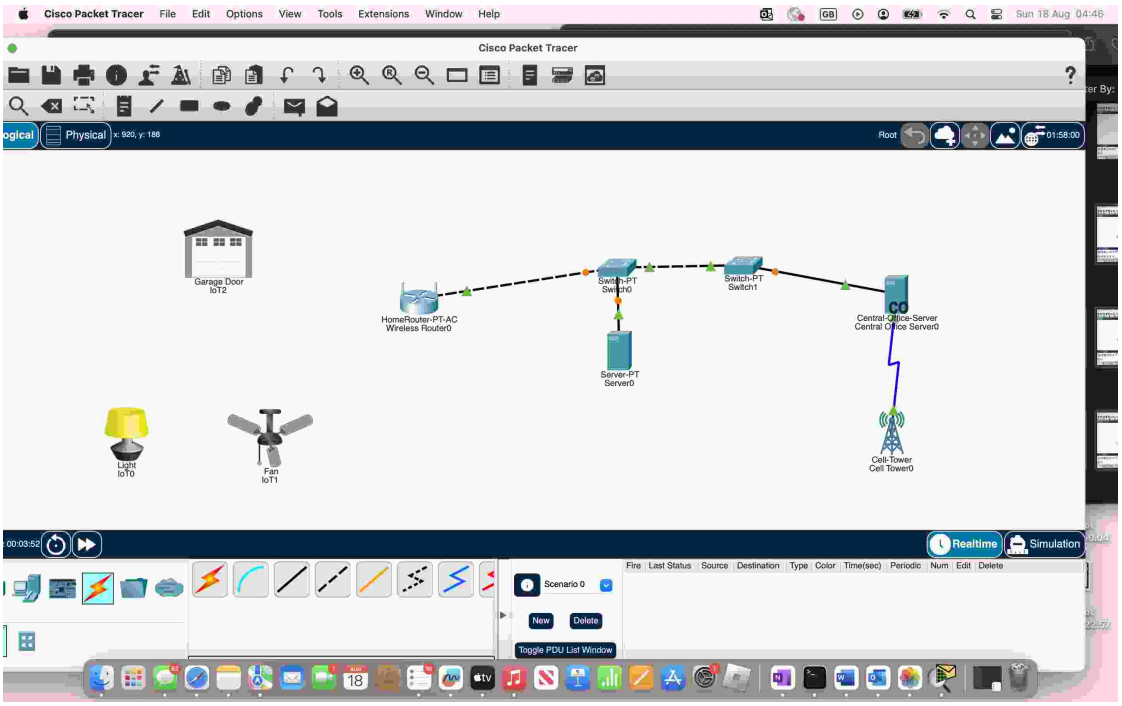


Figure 7.2: All devices connected, including router, switches, and external servers.

At this point, all essential IoT devices (fan, garage door, lamp), the IoT server, and the traditional network components (switches, router, external servers) are physically connected. In the next steps, you'll configure the IoT server (e.g., enabling registration, AAA security) and network elements (IP addressing, DHCP, etc.) to allow seamless communication and device management.

C. IoT Server Setup

Step 1: Set IoT Server as Registration Server

1. **Open the IoT Server:**
   - Click on the **IoT Server** device in the workspace.
   - Select the *Services* tab at the top of the server's configuration window.
2. **Enable the Registration Server:**
   - In the left-hand panel, click on **IoT**.
   - Toggle or click the option to enable the **Registration Server**.
   - This allows IoT devices (like the fan, garage door, lamp) to register themselves with this server.

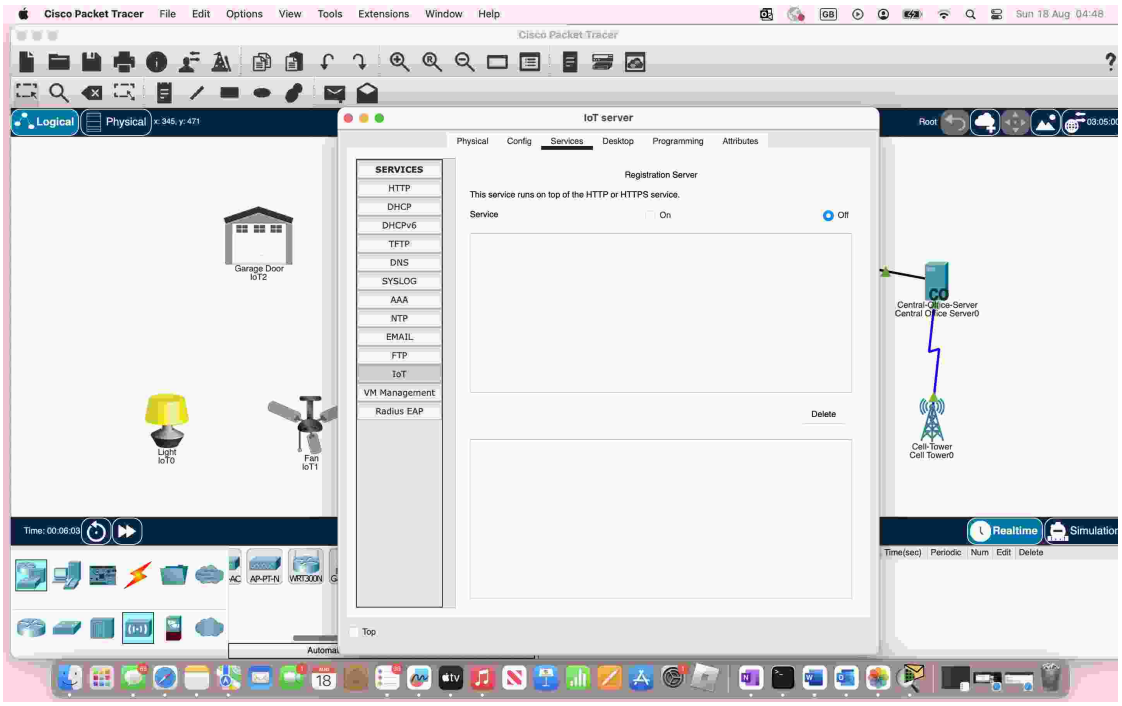


Figure 7.3: Opening the IoT server's Services tab.

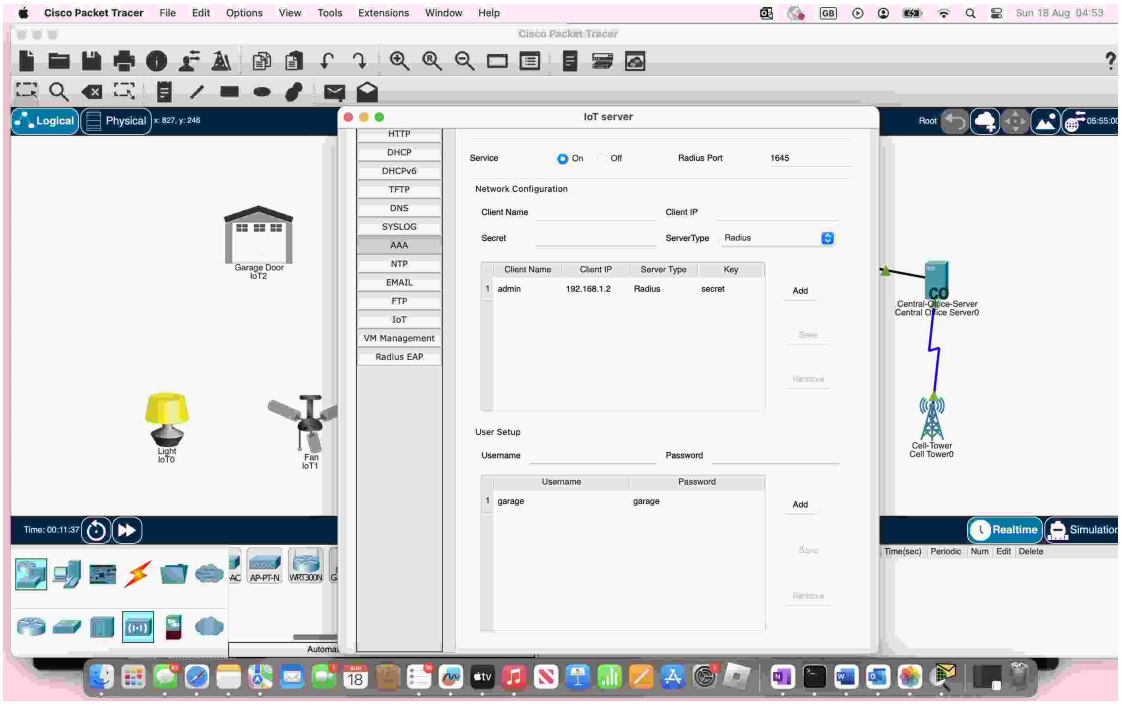


Figure 7.4: Enabling the Registration Server.

Step 2: Enable AAA Server

3. **Stay in the IoT Server Settings:**
   - Within the same **IoT** configuration panel, also enable the **AAA Server** (Authentication, Authorization, and Accounting).
4. **Enter Credentials:**
   - **Client Name:** `admin`
   - **Client IP:** `192.168.1.2`
   - **Secret:** `secret`

   Click **Add** to register these AAA credentials.
5. **Why AAA?**
   - **Authentication:** Ensures only recognized IoT devices/users can access the network.
   - **Authorization:** Controls what each device/user can do once authenticated.
   - **Accounting:** (Optional in Packet Tracer) Could track activity or usage if fully implemented.

Step 3: Configure User Setup for IoT Devices

6. **Assign Usernames/Passwords:**
   - Under the same IoT/AAA configuration, create or edit user credentials for each IoT device:
     - **Garage Door:** user `garage`, pass `garage123`
     - **Fan:** user `fan`, pass `fan123`
     - **Lamp:** user `lamp`, pass `lamp123`
   - These login details allow each device to authenticate against the AAA server when they come online.
7. **Save Your Configuration:**
   - Packet Tracer often auto-saves these settings within the project.
   - It's good practice to manually save the Packet Tracer file (*File* → *Save*) after major changes.

8. **Next Steps:**
    - With the IoT server acting as both a registration server and an AAA server, your IoT devices can now securely register and authenticate.
    - Proceed to configure the network devices (switches, router) so they can route traffic and manage IP addressing (e.g., DHCP).

— Why IoT + AAA Integration?.
- **Centralized Management:** All IoT devices register and authenticate in one place, making it simpler to monitor or revoke access if needed.
- **Security Enhancement:** Prevents unauthorized devices from joining the IoT network, especially important in production or large-scale environments.
- **Scalability:** As you add more IoT devices, the server can track all device credentials, status, and usage from a single interface.

D. Configure Switches and Router

Step 1: Configure Default Gateway on `Switch 0`

1. **Open the CLI:**
    - Click on `Switch 0` in the workspace.
    - Select the *CLI* tab at the top of the switch's configuration window.
    - Press `Enter` if prompted.
2. **Enter the Commands:**

```
en
conf t
int vlan 1
ip add 192.168.1.1 255.255.255.0
no shut
exit
```

    - `en` puts you in *privileged EXEC mode*.
    - `conf t` (configure terminal) enters *global configuration mode*.
    - `int vlan 1` selects the **VLAN 1** interface (the default logical interface on a Cisco switch).
    - `ip add 192.168.1.1 255.255.255.0` assigns an IP address (`192.168.1.1`) and a subnet mask to *VLAN 1*.
    - `no shut` brings the interface up (it is *shutdown* by default).
    - `exit` returns to *global config* mode.
3. **Result:**
    - Setting `192.168.1.1` on VLAN 1 provides the switch an IP for remote management and serves as a gateway reference for connected devices (though in many designs, you'd use a separate router IP for the default gateway).
    - Check the CLI output for messages indicating the interface is now active.

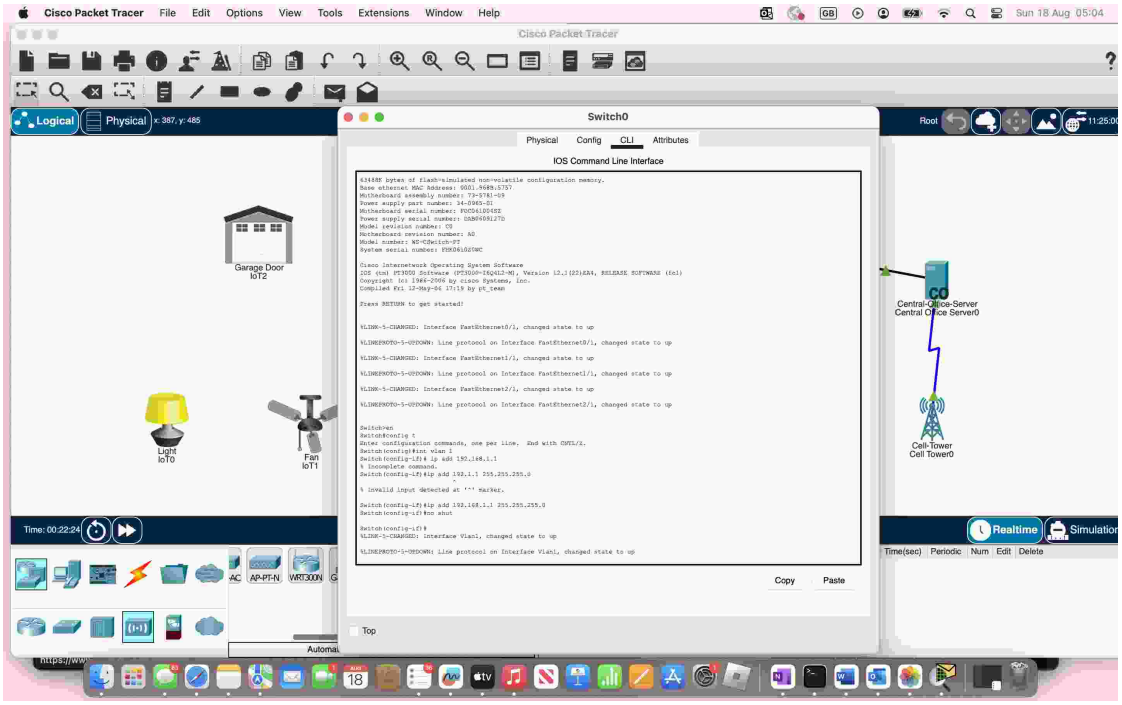


Figure 7.5: Setting VLAN 1 IP on `Switch 0`.

Step 2: Configure the Router

4. **Open the Router Configuration:**
   - Click the **Router** device in Packet Tracer.
   - Go to *Configure* (or *Config*) → **Internet** (this may vary by router model or Packet Tracer version).
5. **Select Static IP and Assign:**
   - **IP Address:** `192.168.1.2`
   - **Default Gateway:** `192.168.1.1`

   If there's a field for **Subnet Mask**, ensure it's set to `255.255.255.0`, matching `192.168.1.x`.
6. **Confirm and Close:**
   - These static IP settings ensure the router is reachable on the `192.168.1.x` network.
   - You can verify connectivity by later `pinging` `192.168.1.2` from the switch or other end devices.

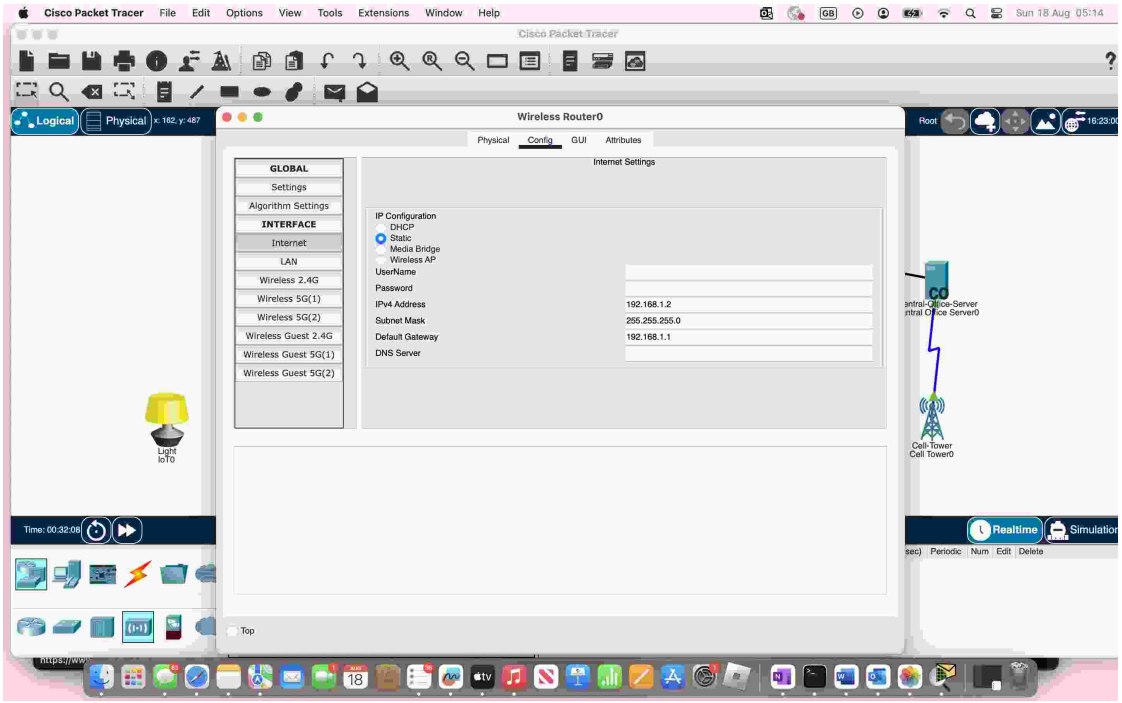


Figure 7.6: Router static IP: `192.168.1.2`, gateway `192.168.1.1`.

With these IP settings, both the switch and router have static addresses in the `192.168.1.0/24` subnet, and IoT devices can reference them for authentication or registration. Next, you can proceed to set up additional services (e.g., wireless parameters, DHCP) for IoT devices.

E. Set Up Wireless (SSID and Security)

Step 1: Check SSID on Router

1. **On the Router's GUI:**
   - Click on the router, go to the *GUI* tab (sometimes labeled `Setup` or `Web_GUI`).
   - Under *Internet Connection Type*, select `Automatic Configuration - DHCP` (or the appropriate type, depending on your lab setup).
2. **Edit the Wireless SSID:**
   - Navigate to the **Wireless** section in the router's GUI.
   - If the default SSID is `default`, rename it to `home`.
   - Click **Save** or **Apply** to confirm changes.

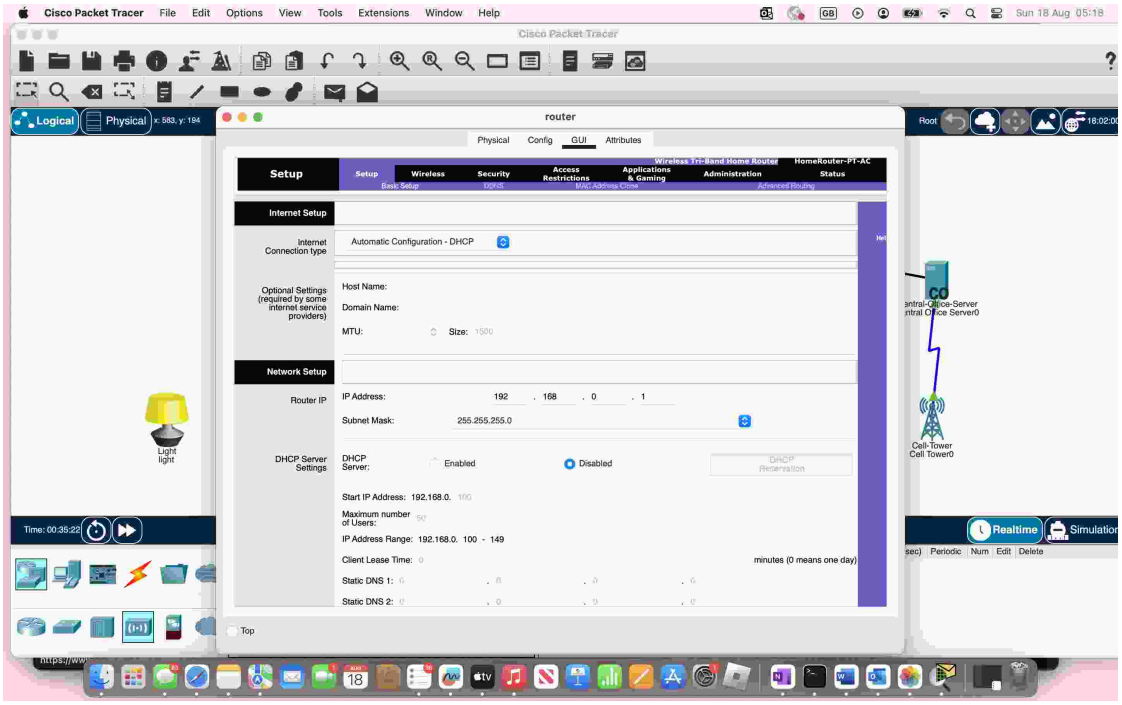


Figure 7.7: Router GUI for SSID "home".

Step 2: Connect the Garage Door to Wireless

3. **Configure Garage Door for Wireless:**
   - Click the **Garage Door** device.
   - Go to *Config* → **Wireless0**.
   - In the SSID field, type `home`.
   - The Garage Door should automatically associate with the router's `home` network if they share the same channel settings by default.
4. **Verify Connection:**
   - Check for a dotted (wireless) line between the Garage Door and the router in Packet Tracer's workspace.
   - Hover your cursor over the Garage Door to see if the pop-up indicates it is connected to `home`.

Step 3: Enable WPA2 Enterprise Security

5. **Wireless Security on the Router:**
   - In the router's GUI, look for a **Wireless Security** or **Security** tab.
   - Choose `WPA2 Enterprise` as the security mode.
   - **Radius Server IP**: `192.168.1.2` (This should match your router's IP for AAA, or if you're using a separate RADIUS server, enter its IP here.)
   - **Shared Secret**: `secret`
   - Click **Save** or **Apply**.

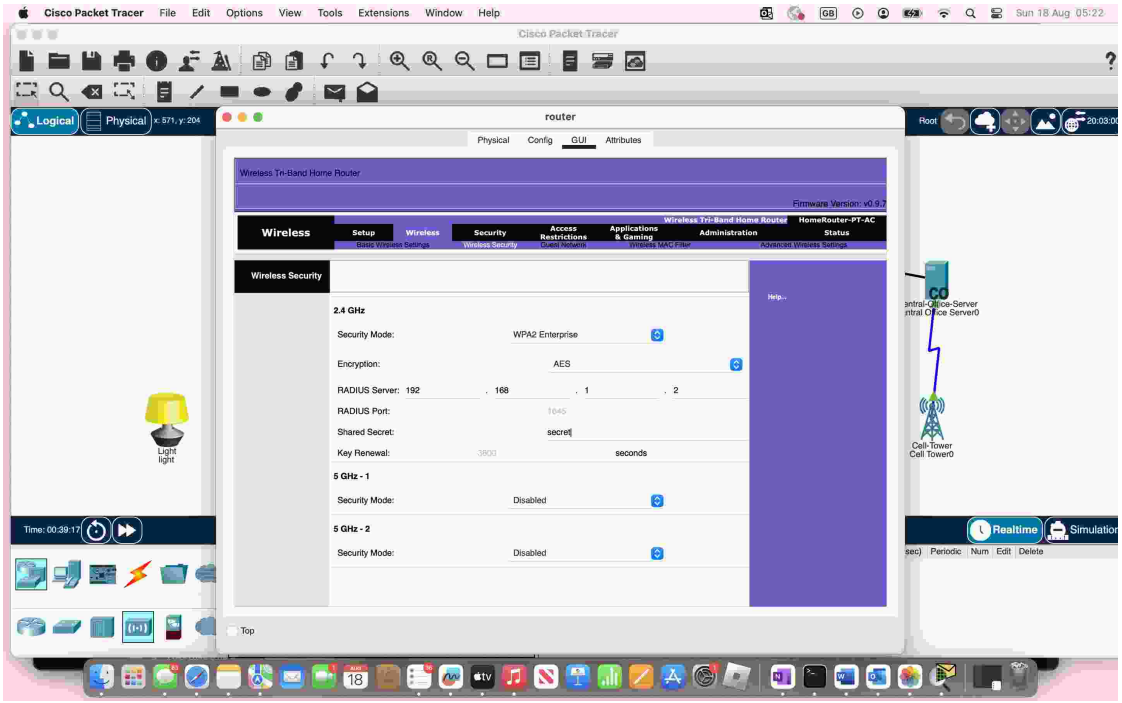


Figure 7.8: Enabling WPA2 Enterprise with RADIUS server.

Step 4: Authenticate the Garage Door

6. **Garage Door WPA2 Configuration:**
   - Return to *Config* → **Wireless0** on the Garage Door.
   - Under **Authentication**, select `WPA2`.
   - Enter:
     - **User ID**: `garage`
     - **Password**: `garage123`
   - This matches the AAA credentials you set up earlier (`garage / garage123`).
7. **Confirm Association:**
   - The Garage Door now attempts to authenticate via WPA2 Enterprise using the RADIUS server at `192.168.1.2`.
   - If everything is correct, the device should successfully connect, and you may see a green status indicating *authenticated*.

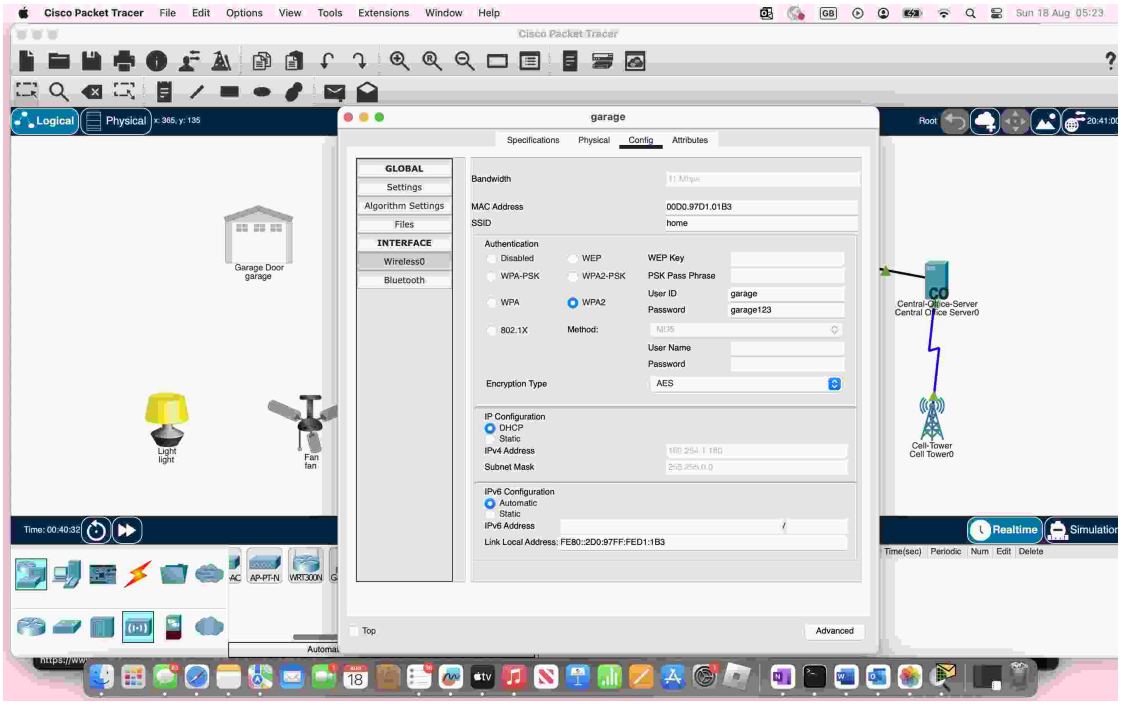


Figure 7.9: **Garage Door** WPA2 Enterprise credentials.

Note: Repeat this process for other wireless IoT devices (e.g., `fan` or `lamp`) if they also require WPA2 Enterprise authentication. Each device will need a unique username/password from your AAA server configuration.

— Why WPA2 Enterprise?.

- **Enhanced Security:** Using RADIUS-based authentication is more robust than WPA2 Personal (PSK), allowing separate credentials for each device.
- **Centralized Management:** The AAA server can track which device logs in, revoke access if necessary, and maintain unique passwords—improving control over your IoT network.
- **Scalability:** If you add more IoT devices, you can simply assign each one new credentials without changing a shared Wi-Fi passphrase across the board.

F. Laptop Setup

Step 1: Connect Laptop to the IoT Server

1. **Add a Laptop:**
   - From *End Devices* (or a similar category), drag a **Laptop** onto the workspace.
   - Position it near either the IoT server or the nearest switch to simplify cabling.
2. **Wired Connection:**
   - Use the **Copper Straight-Through** cable (or *Automatically Choose Connection Type*) to connect the Laptop's `FastEthernet0` port to:
     - The IoT server's `FastEthernet0` port, **or**
     - An available `FastEthernet` port on the switch that is already linked to the IoT server.
3. **Assign an IP:**
   - Click the Laptop, then go to *Config* → **FastEthernet0**.
   - Enter:
     - **IP Address:** `192.168.1.4`

- **Subnet Mask:** `255.255.255.0` (assuming the same /24 network)
- **Default Gateway:** `192.168.1.1` or `192.168.1.2`, depending on your router setup.

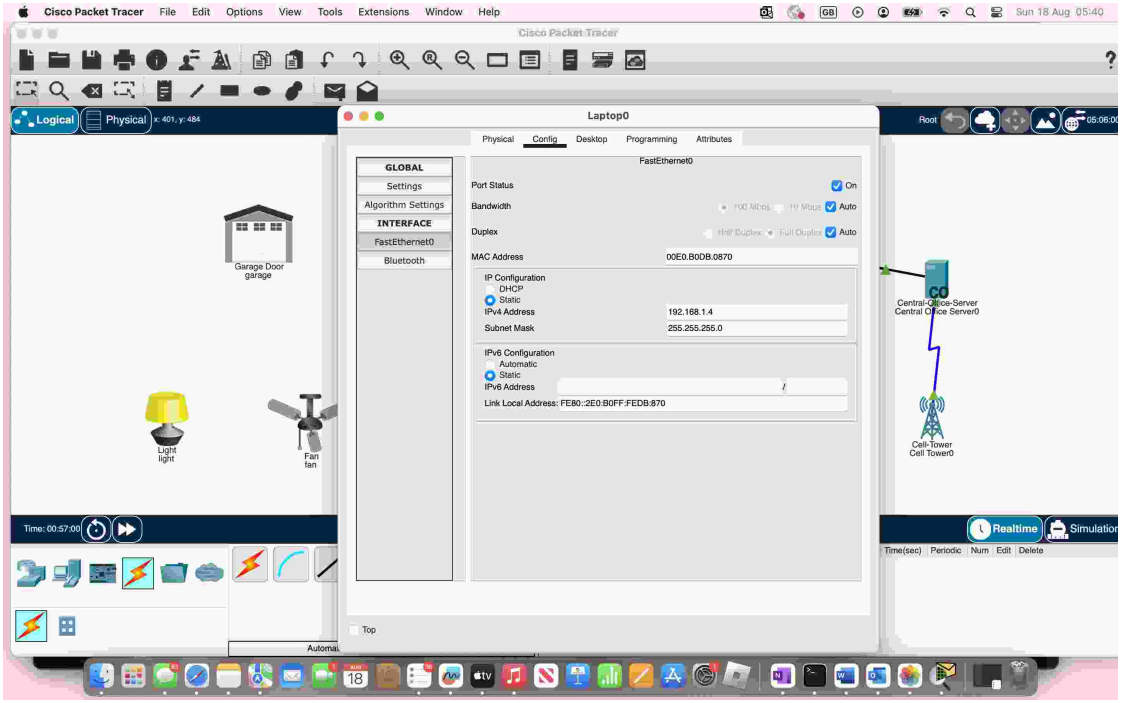


Figure 7.10: Laptop connected to the IoT server or switch with IP `192.168.1.4`.

Step 2: Ping the IoT Server

4. **Open Command Prompt on the Laptop:**
   - Click the *Desktop* tab on the Laptop, then select **Command Prompt**.
5. **Test Connectivity:**
   - Type `ping 192.168.1.2` to verify connectivity with the router or IoT server (depending on which IP you set up for the server or router).
   - Alternatively, `ping 192.168.1.1` if your gateway or server is on that IP.
6. **Successful Ping:**
   - If replies come back with `Reply from...`, the Laptop can communicate with the network, confirming correct IP settings and active connections.
   - If you receive `Request timed out`, re-check your cable connections, IP addresses, and gateway settings.

— Why Connect a Laptop?.

- **Management Station:** The Laptop can be used to configure or manage the IoT devices (e.g., via a web interface, SSH, or IoT Monitor).
- **Troubleshooting:** Having a PC or laptop in the network allows you to `ping`, `traceroute`, or run other diagnostic commands to verify connectivity.
- **Expandability:** You could add more end-user devices (phones, additional PCs) to simulate a realistic network with multiple users managing or interacting with IoT devices.

G. Configuring the Smartphone

Step 1: Add Smartphone to the Network

1. **Place a Smartphone in the Topology:**

- From *End Devices* (or *Smart Devices*) in Packet Tracer, drag a **Smartphone** onto the workspace.

2. **Access IoT Monitor:**
   - Click the **Smartphone**.
   - Switch to the *Desktop* tab, then select **IoT Monitor**.
3. **Set IoT Server Address:**
   - In **IoT Monitor**, locate the *Server Address* or *IoT Server* field.
   - Type `192.168.1.2` (the IP of your IoT server, for example).
   - Log in using credentials you've established (e.g., user `home`, pass `home123`).

Step 2: Wireless Security for Smartphone

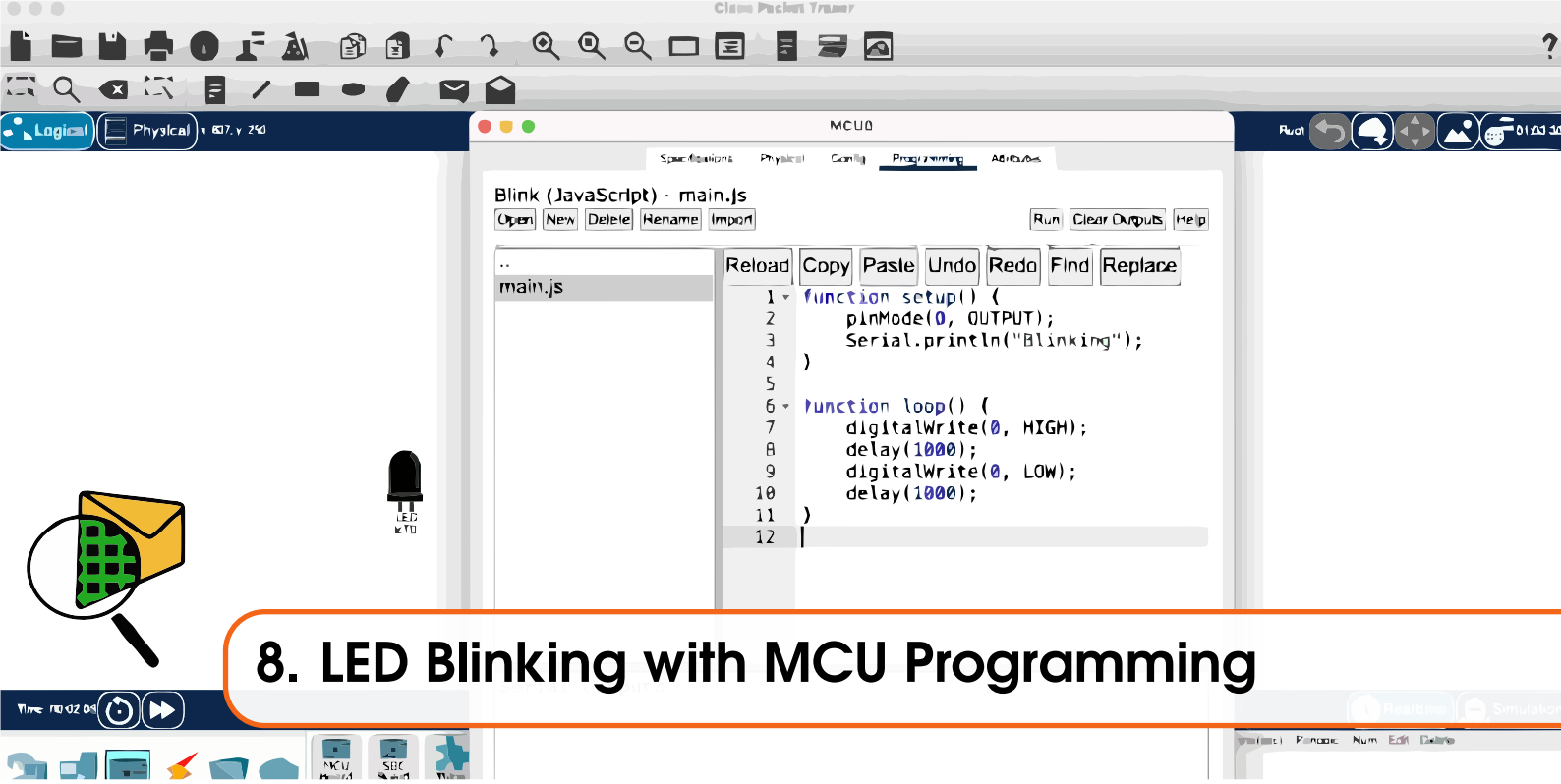
4. **Disable Cellular, Enable Wi-Fi:**
   - Under the Smartphone's *Config* tab, click **3G/4G Cell** and turn `Port Status` **Off**.
   - Go to **Wireless**, set:
     - **SSID = home**
     - **Security**: `WPA2`
     - **User ID**: `smartphone`
     - **Password**: `smartphone123`
   - This matches the `WPA2 Enterprise` setup on your router.
5. **Verify Connection:**
   - The Smartphone should now connect wirelessly to the SSID `home` and authenticate via `WPA2`.
   - Check for a dotted wireless line in the workspace, indicating successful association.
   - If you open **IoT Monitor** again, you can confirm the phone is recognized by the IoT server at `192.168.1.2`.

With this configuration, your **Smartphone** can manage and control IoT devices via Wi-Fi, just like the Laptop. You may now access the IoT Monitor from the phone's desktop and interact with any connected IoT devices (fan, lamp, garage door, etc.) in real time.

— Why Use a Smartphone?.
- **Mobile Control:** Simulates a user on a mobile device controlling or monitoring IoT devices anywhere in the network.
- **Wireless Authentication Demo:** Emphasizes `WPA2 Enterprise` usage, separate from cellular data, to practice secure Wi-Fi connections.
- **Additional Testing:** Allows you to verify that devices can be managed from multiple endpoints (Laptop, Smartphone), ensuring consistent network-wide IoT management.

Summary

You have successfully set up a **smart home IoT network** in Packet Tracer. By creating a centralized IoT server (with AAA), configuring switches/routers for IP addressing, and enabling WPA2 Enterprise security, you ensured that devices such as the garage door, fan, and lamp can join securely. A laptop and smartphone can both control these IoT devices—validating your network's connectivity, authentication, and remote management capabilities.



8. LED Blinking with MCU Programming

Introduction

In this lab, you will **create a simple IoT topology** in Cisco Packet Tracer, connecting a *Microcontroller Unit (MCU)* to an *LED* and programming the MCU with JavaScript so the LED blinks at regular intervals. This exercise demonstrates basic IoT programming concepts—showing how to toggle an LED on and off using digital pins and simple timing logic.

Objective

- **Create a basic IoT topology** with an MCU and an LED.
- **Wire the MCU** to the LED using a custom IoT cable.
- **Write and execute** a simple JavaScript program on the MCU.
- **Observe and verify** the LED blinking at a 1-second interval.

Lab Plan

A. **Add an MCU and LED to the Workspace**
B. **Connect the Devices (MCU and LED)**
C. **Program the MCU** (JavaScript)
D. **Test the Program**

Required Software

- **Cisco Packet Tracer 8.x** (or newer)
- Familiarity with the *Programming* tab for IoT boards in Packet Tracer
- Basic understanding of JavaScript syntax for microcontroller loops

A. Add an MCU and LED to the Workspace

1. **Open Packet Tracer:**

- Launch Packet Tracer on your computer.
- In the lower-left device categories, select *End Devices* and then choose the subcategory *Components*.

This section contains various IoT modules and sensors you can use in your simulation.

2. **Add an MCU and an LED:**

   - From the *Components* subcategory, drag an **MCU (Microcontroller Unit)** onto the workspace.
   - Similarly, select an **LED** and place it near the MCU so that you can wire them together easily.

   The MCU will act as your programmable controller, and the LED will serve as a simple output device.

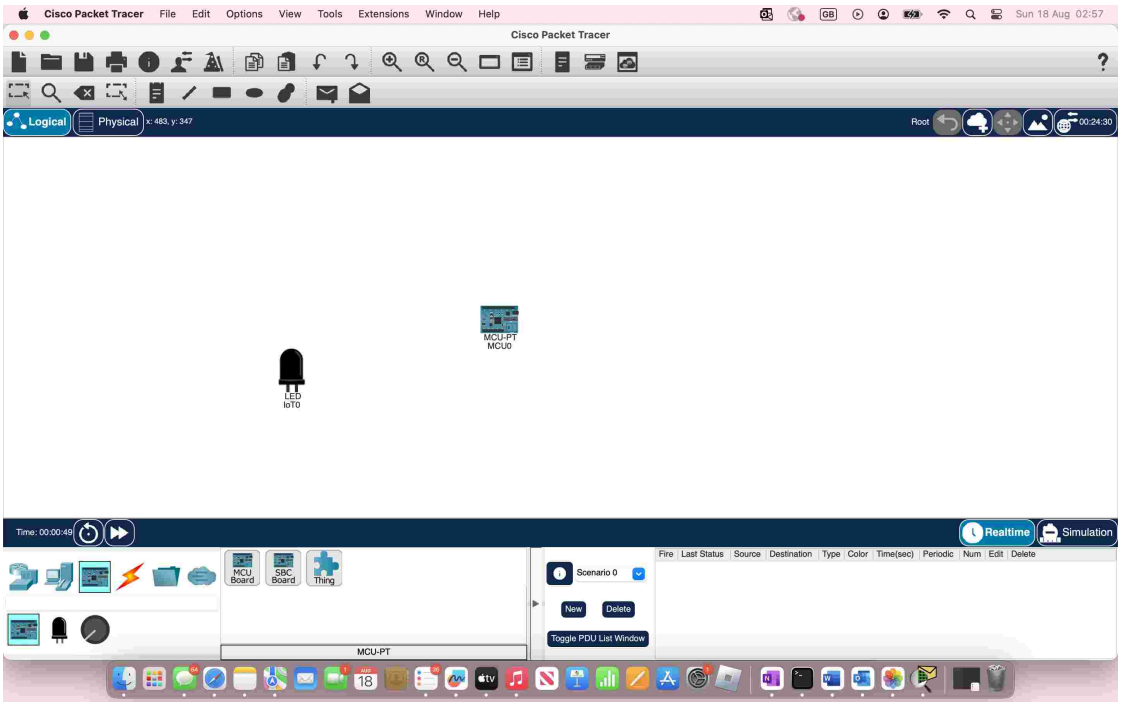


Figure 8.1: MCU and LED added to the workspace.

— Why an MCU and an LED?.

- **Basic IoT Setup:** An MCU controlling an LED is a fundamental example of how microcontrollers interface with outputs in IoT.
- **Hands-On Practice:** Programming the MCU to blink the LED provides hands-on experience in writing, uploading, and testing simple scripts in Packet Tracer.
- **Scalable Concept:** The same principles apply if you replace the LED with more complex actuators (motors, alarms) or add input sensors.

B. Connect the Devices (MCU and LED)

3. **Use the IoT Custom Cable:**

   - In Packet Tracer's *Connections* panel (often at the bottom-left), select the **IoT custom cable**.
   - This cable type is designed for direct pin-to-pin connections between IoT components

(e.g., sensors and microcontrollers).

4. **Wire the LED to the MCU:**

   (a) Connect the `D0` pin of the **LED** to the `D0` pin of the **MCU**.
   (b) Verify that the cable line appears, confirming a successful connection.
   (c) This wiring allows the MCU's `D0` digital output to send signals (e.g., *HIGH* or *LOW*) to the LED.

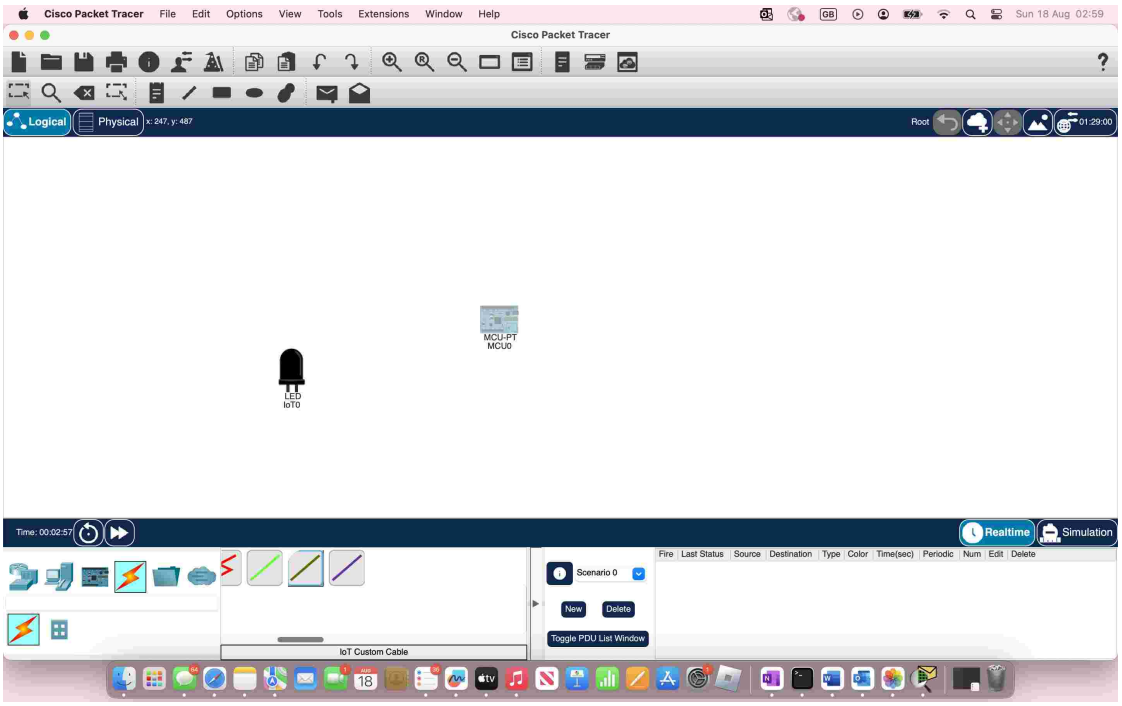


Figure 8.2: Wiring the LED's `D0` pin to the MCU's `D0` pin.

— Tips for Proper Wiring.

- **Matching Pin Numbers:** In your code, ensure you reference the correct MCU pin (here `D0`). If you change the wiring to `D1`, update your script accordingly.
- **Polarity Considerations:** Although Packet Tracer abstracts some real-world aspects (like an LED's anode/cathode), make sure you consistently follow the intended direction of signal flow.
- **Multiple Outputs:** If you add more LEDs or actuators, each should be connected to a different digital pin (e.g., `D1`, `D2`, etc.).

C. Program the MCU (JavaScript)

5. **Open the MCU Configuration:**

   - Click on the **MCU** in the workspace.
   - In the device window, switch to the *Programming* tab to see available script files and editor options.

6. **Select `main.js`:**

   - In the left panel, highlight `main.js` (or whichever default file is present).
   - You will edit this file to implement the blinking LED logic.

7. **Function Setup:**

   (a) **Initialize pin D0 as output**:

   `pinMode(0, OUTPUT);`

   This configures `pin 0` as an `OUTPUT` pin, enabling the MCU to drive the LED with either `HIGH` or `LOW` signals.

   (b) **Print a confirmation message in the console**:

   `Serial.println("Blinking");`

   This helps you verify in the console that the script is indeed running.

8. **Function Loop:**
   To blink the LED, toggle the pin `HIGH` (on) and `LOW` (off), pausing one second between each state:

```
// Turn LED on
digitalWrite(0, HIGH);
delay(1000);  // wait 1 second

// Turn LED off
digitalWrite(0, LOW);
delay(1000);  // wait 1 second
```

   This simple sequence continuously powers the LED for one second and then turns it off for one second.

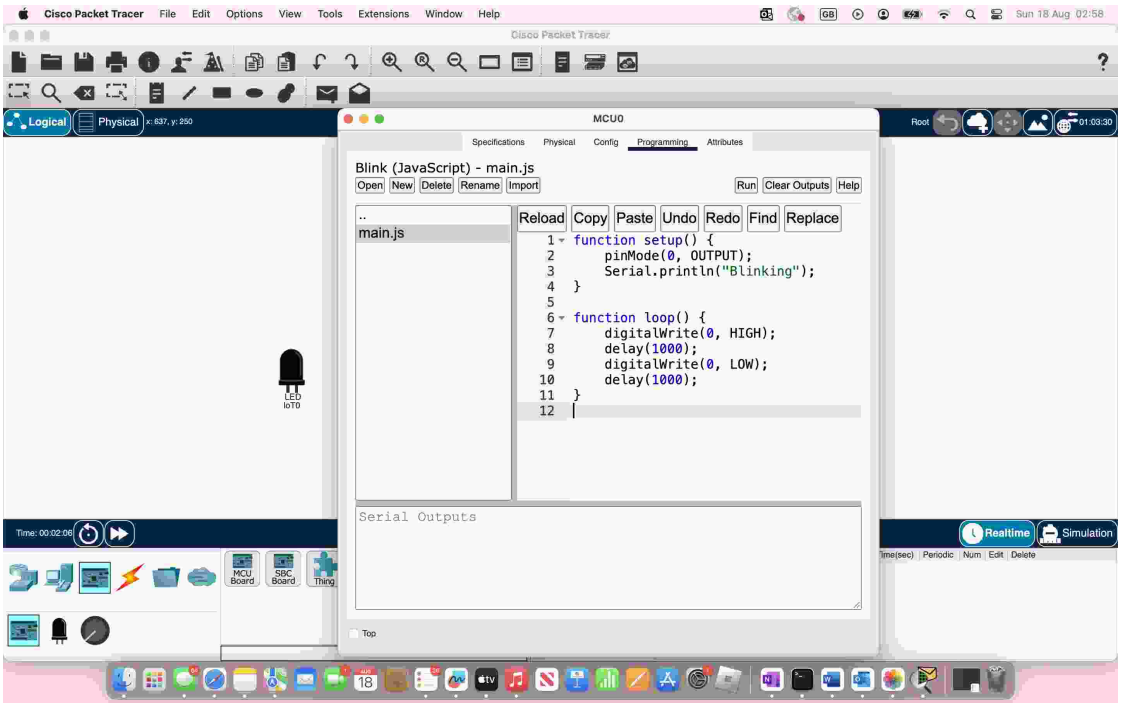


Figure 8.3: JavaScript code snippet controlling the LED on pin D0.

— How the Code Works.

- `pinMode(0, OUTPUT)`: Declares that `pin 0` drives an external component (in this case, the LED).
- `digitalWrite(0, HIGH)`: Sends a `5V` (or `3.3V`, depending on the MCU) signal on pin

D0, lighting the LED.
- `delay(1000)`: Pauses execution for 1,000 milliseconds (1 second).
- `digitalWrite(0, LOW)`: Drops the pin to 0V, turning the LED off.
- **Infinite Loop**: Typically, these commands are placed in the main loop (or repeated cycle) so the LED keeps blinking indefinitely.

D. Test the Program

9. **Run the Code:**
    - In the MCU's *Programming* tab, click the **Run** button to start executing your JavaScript code.
    - If you have a `Serial.println("Blinking");` statement in your setup, the console output should read *"Blinking"* or similar, confirming the script is active.
10. **Observe the LED:**
    - Watch the LED on the workspace. It should alternate *on* (HIGH) and *off* (LOW) every second.
    - If it's not blinking, double-check your wiring (ensure both ends are connected to D0), code syntax, or delays.

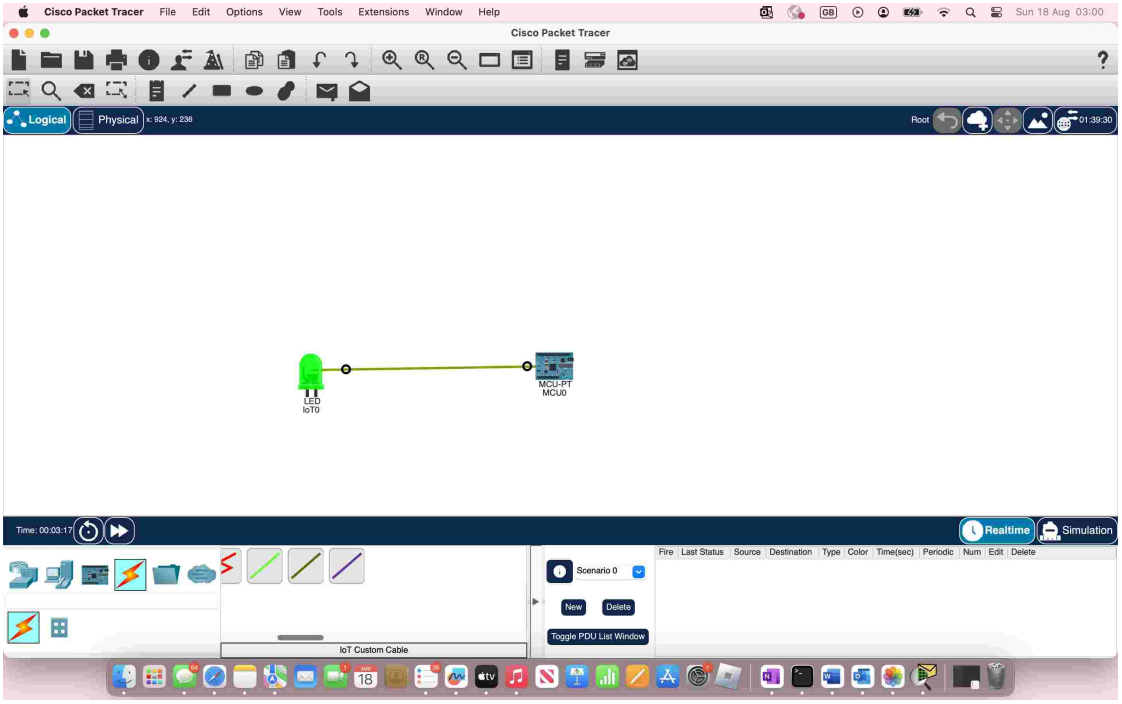
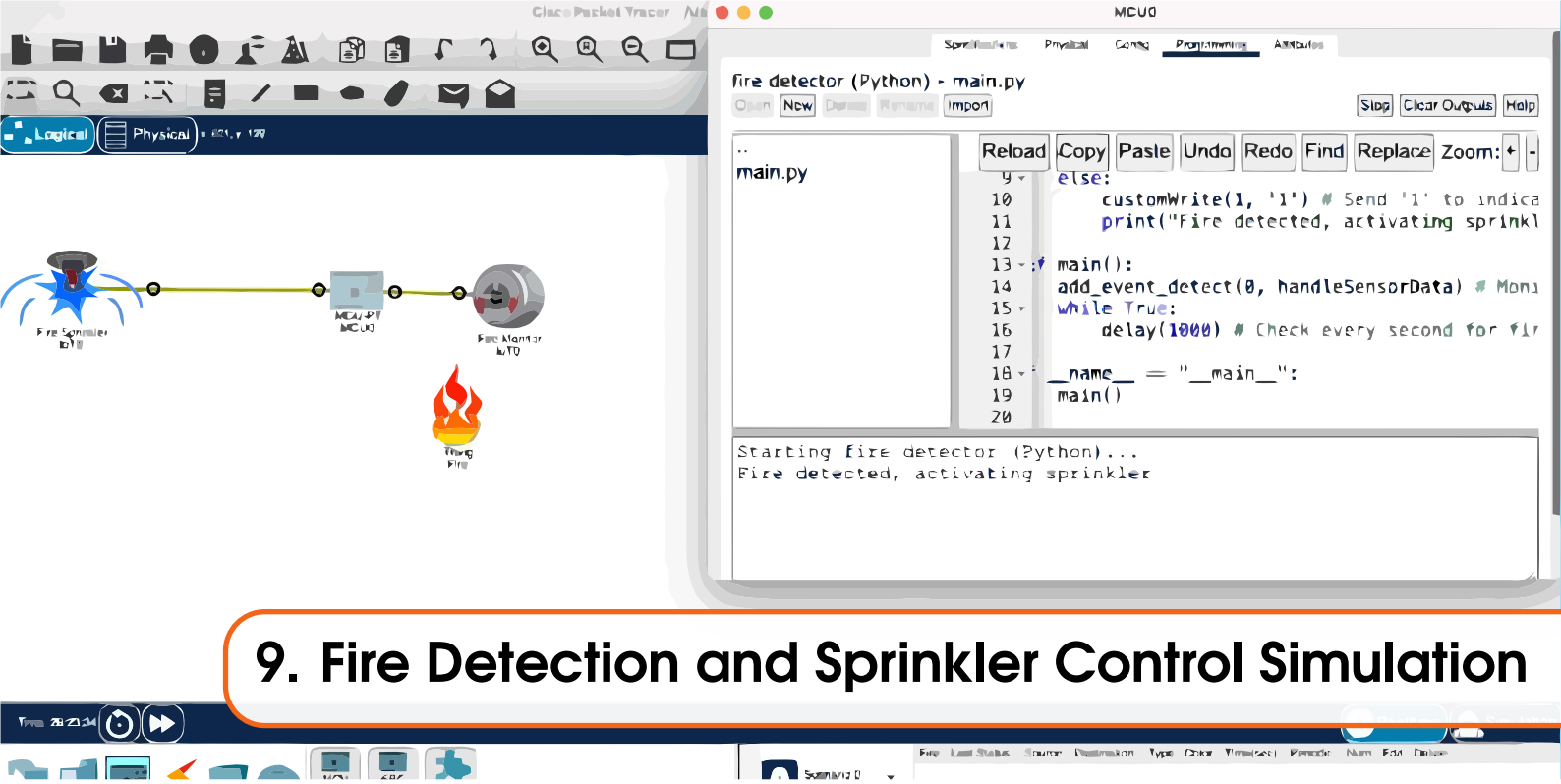


Figure 8.4: LED blinking at a 1-second interval, toggled by the MCU.

— **Troubleshooting LED Issues.**
- **Wiring Check:** Ensure the LED's D0 pin is actually connected to the MCU's D0 pin with the IoT custom cable.
- **Case Sensitivity in Code:** Make sure all function names (`pinMode`, `digitalWrite`, `delay`) are spelled correctly and use the exact case Packet Tracer requires.
- **Console Logs:** Use `Serial.println` statements to see if the loop is running (e.g., "LED on," "LED off") for clearer debugging.

Summary

You have successfully **created a topology**, **wired the MCU and LED**, **programmed the MCU** to blink the LED, and **observed the LED** turning on and off every second. This lab demonstrates how a simple JavaScript loop can toggle an MCU's digital output—highlighting the basics of Packet Tracer IoT programming for physical components. You can extend this project by adding more sensors, altering blink intervals, or incorporating user input to control the LED state.



9. Fire Detection and Sprinkler Control Simulation

Introduction

In this lab, you will build and configure a **fire monitoring system** in Cisco Packet Tracer using an *MCU*, *fire monitor*, *ceiling sprinkler*, and a simulated *heating element*. By programming the MCU to detect fire and activate the sprinkler, you will see how basic IoT logic can respond in real time to sensor input.

Objective

- **Create** a fire monitoring system with MCU, Fire Monitor, and Sprinkler.
- **Program** the MCU to detect fire and automatically trigger the sprinkler.
- **Learn** how to connect and configure IoT devices in Packet Tracer.
- **Test** the system by simulating a fire event and observing the sprinkler response.

Lab Plan

A. **Set Up Packet Tracer Workspace**
B. **Create the Topology** (MCU, Fire Monitor, Sprinkler, Heating Element)
C. **Connect Devices** (IoT custom cables)
D. **Program the MCU** (Python/JavaScript logic)
E. **Test the Program** (Simulate fire, verify sprinkler activation)

Required Software

- **Cisco Packet Tracer 8.x** (or newer)
- Familiarity with *Components* and *Advanced* tabs for IoT devices
- Basic Python or JavaScript coding knowledge for MCU scripts

A. Set Up Packet Tracer Workspace

1. **Open Packet Tracer:**
   Launch the application and ensure you have a blank Logical topology workspace.

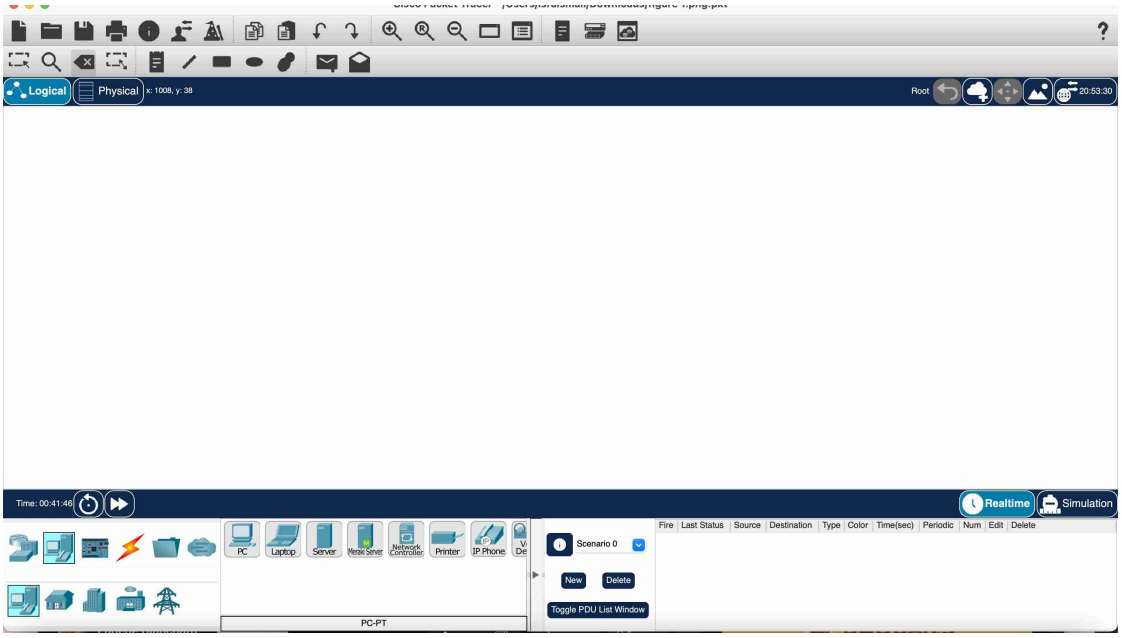


Figure 9.1: Blank Packet Tracer workspace for the fire monitoring system.

B. Create the Topology

2. **Add the Following Devices:**
   In the Packet Tracer *Device Selection* box, locate and drag these components into the workspace:
   - **Fire Sprinkler**
   - **MCU** (Microcontroller Unit)
   - **Fire Monitor**

   These will form the core of the fire detection system.
3. **Position Them:**
   Arrange the three devices so that the **MCU** can easily connect to both the **Fire Monitor** (acting as the input sensor) and the **Fire Sprinkler** (acting as the output/actuator). Ensure there is enough space around each device for cabling and any additional elements you might add later.

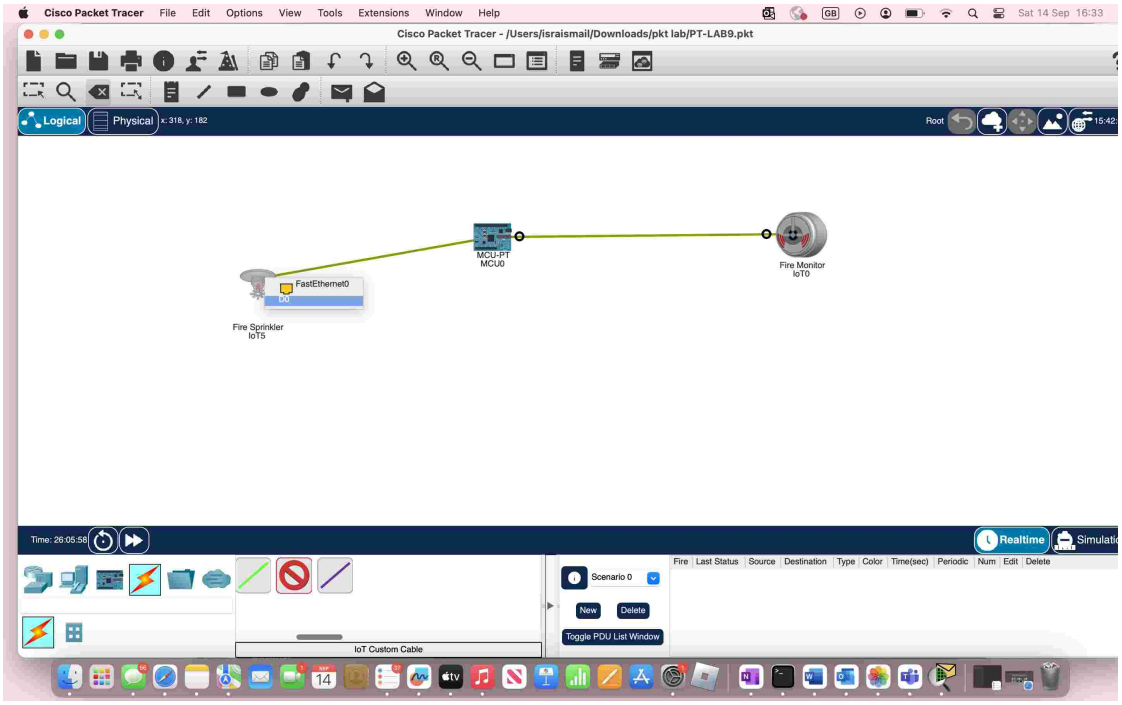


Figure 9.2: MCU, Fire Monitor, and Fire Sprinkler placed in the workspace.

— Why These Three Devices?.

- **MCU (Microcontroller Unit):** Orchestrates logic, reading sensor data from the Fire Monitor and controlling the Sprinkler based on threshold or detection events.
- **Fire Monitor (Sensor):** Detects heat or smoke, sending signals to the MCU when fire conditions are met.
- **Fire Sprinkler (Actuator):** Responds to the MCU's commands by spraying water if a fire is detected, simulating a real-life fire suppression mechanism.

This minimal setup illustrates a classic **sensor-actuator** workflow, with the MCU acting as the control logic in between.

C. Connect the Devices (IoT Custom Cables)

4. **Use IoT Custom Cable:**

   - In the Packet Tracer toolbar (often at the bottom-left), select the *Connections* icon.
   - From the list of cables, choose the **IoT Custom Cable** (it may appear as "Custom Cable" or "IoT Cable").

   This cable type allows direct pin-to-pin connections for IoT sensors and actuators.
5. **Wiring:**

   - **Fire Monitor to MCU**: Connect the `D0` pin of the **Fire Monitor** to the `D0` pin on the **MCU**. This link carries the sensor's "fire detected" signal as an *input* to the MCU.
   - **MCU to Fire Sprinkler**: Connect the `D1` pin of the **MCU** (an *output* pin) to the `D0` pin on the **Fire Sprinkler**. When the MCU determines a fire, it drives `D1` high (or sends a signal) to activate the sprinkler.

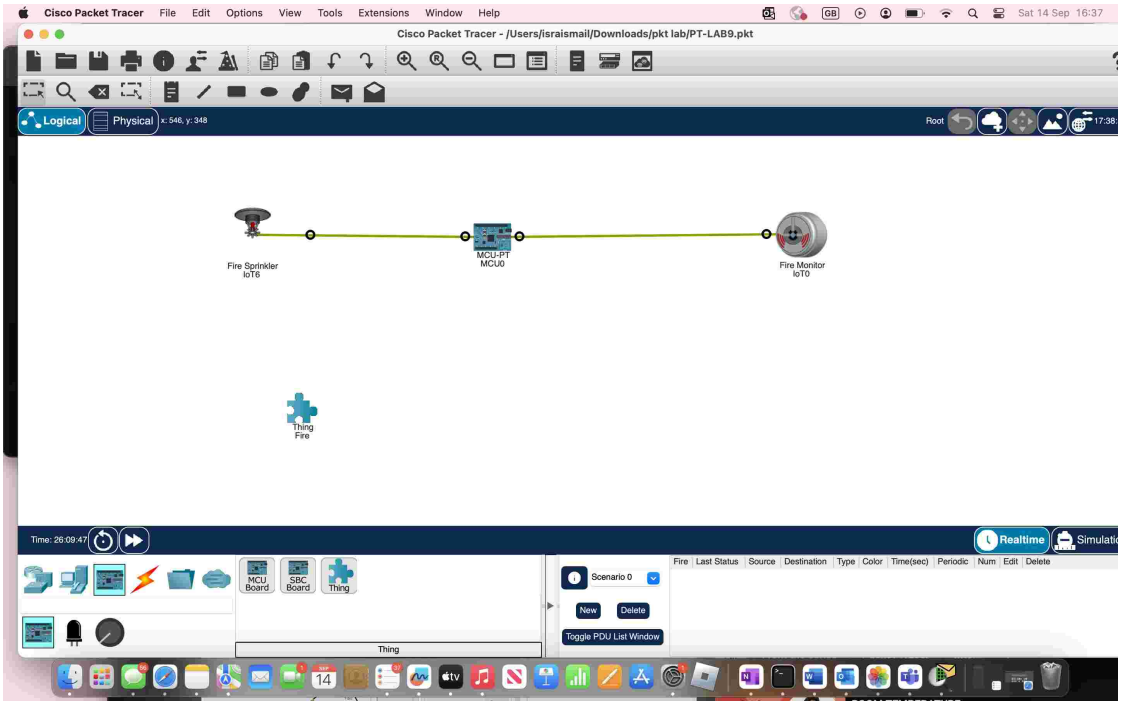


Figure 9.3: Wiring the Fire Monitor and Sprinkler to the MCU.

— Tips for Proper Wiring.

- **Pin Matching:** Ensure the correct pins match the roles:
  - **Sensor** → `DO` **(input)** on the MCU.
  - **MCU Output** → `DO` on the **actuator** (sprinkler).
- **Check Pin Labels:** Some devices allow multiple digital pins (D0, D1, D2, etc.). Make sure the pin numbers in your code later correspond to how you've wired them here.
- **Save Frequently:** If you have multiple cables and devices, consider saving the Packet Tracer file after each major step to avoid redoing connections.

D. Create the Heating Element (Simulated Fire)

6. **Add a New "Thing" Device:**

   - In the **Components** category (lower-left pane of Packet Tracer), find and drag a device labeled **Thing** onto the workspace.
   - This generic device will act as a simulated heat source or "fire" for the lab.

7. **Rename to "Fire":**

   (a) Click the newly added **Thing**.
   (b) Go to *Config* → **Global Settings**.
   (c) Change the **Display Name** from its default (e.g., "Thing1") to `Fire`.

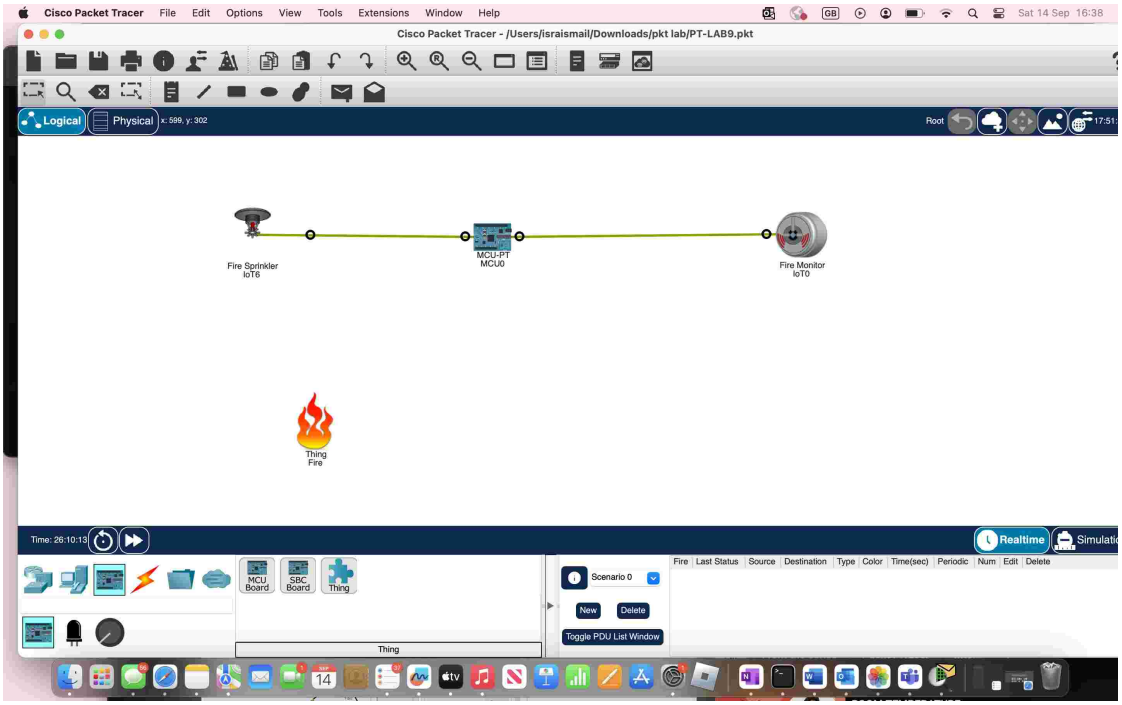


Figure 9.4: Renaming a generic Thing to “Fire.”

8. **Advanced Configuration:**

   (a) Click on *Advanced* → **Thing Editor**, then select **Properties**.
   (b) Change the **Component Name** to `Fire`.
   (c) (Optional) Click **New** to upload or add a custom icon representing fire, if you wish to make it visually distinct.

   This allows you to identify and manage this device as your heat source within the Packet Tracer workspace.

— Why Create a “Fire” Thing?.

- **Simulation of Heat Source:** By naming it `Fire`, you can hover over or “move” this device near the *Fire Monitor* to simulate real fire conditions.
- **Flexible Customization:** Packet Tracer’s *Thing Editor* lets you change icons, behaviors, and properties, making it useful for a wide range of IoT scenarios.
- **Enhanced Visualization:** Using a distinctive icon (like a flame) helps visually remind you that this device represents heat or combustion in the simulation.

E. Program the MCU

9. **Open the MCU Configuration:**

   - On the **MCU** device, switch to the *Programming* tab.
   - Click **New** to create a fresh project file (e.g., name it `Fire`).

   This project will contain the logic for detecting fire from the sensor and triggering the sprinkler.

10. **Template Selection:**

    - Depending on your lab’s instructions, choose either *Empty - JavaScript* or *MQTT Broker (Python)* in the drop-down.

- Although Packet Tracer might label the main file as `main.js`, you can still write Python code if you selected a Python-based template.

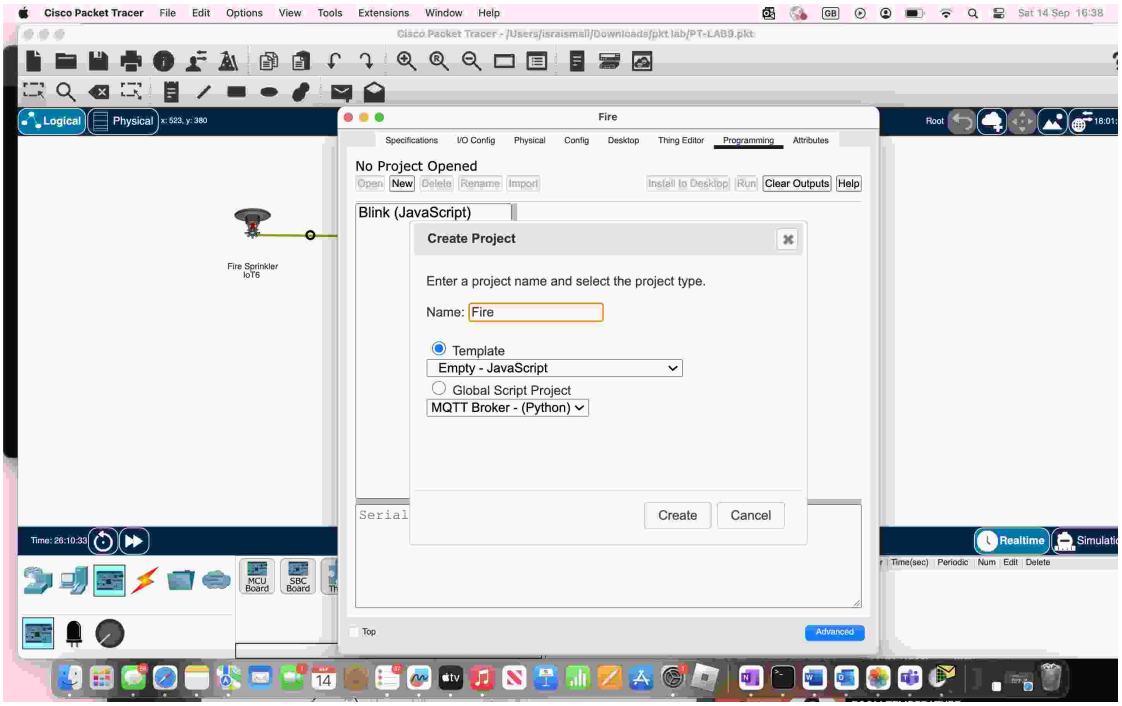


Figure 9.5: Creating a new Python-based project for the MCU.

11. **Open** `main.js`**:**

- Even if you chose a Python template, Packet Tracer names the default script file `main.js`.
- Simply open `main.js` in the editor to begin writing or pasting code.

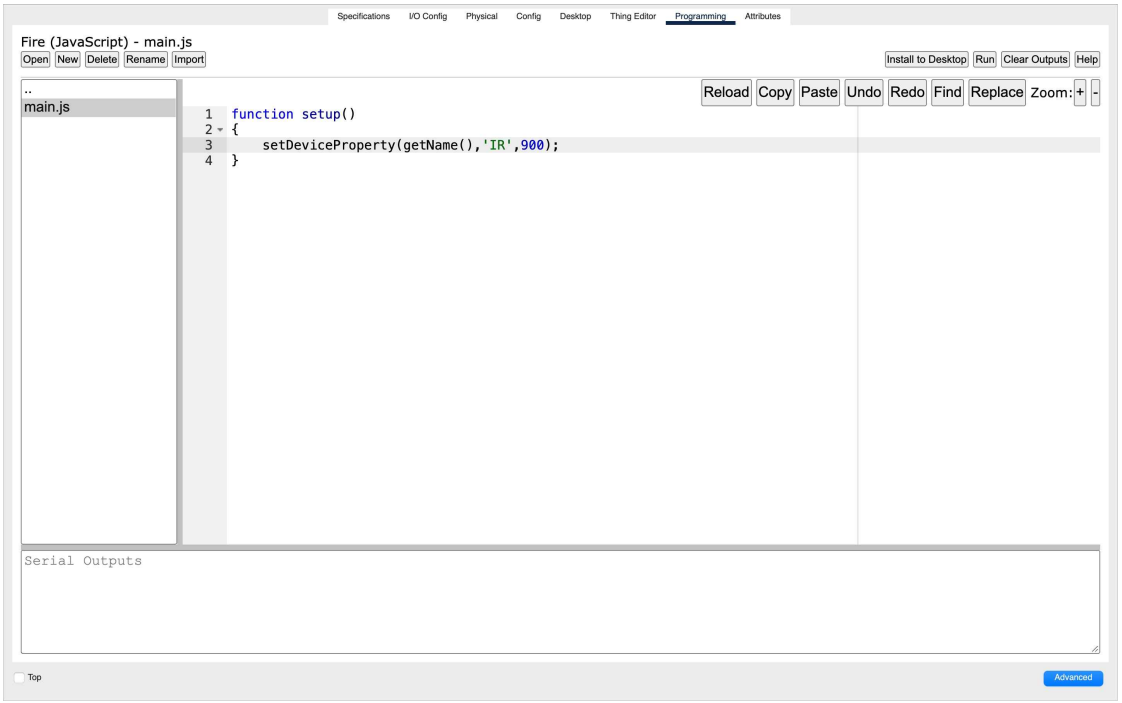


Figure 9.6: Editing the `main.js` file for the MCU's code.

Writing the Python Code

12. **Paste the Following Code:**

```
from gpio import input, output
from time import sleep

def handleSensorData():
    value = digitalRead(0) # Read from fire monitor sensor on pin D0
    if value == 0:
        customWrite(1, '0') # Send '0' to indicate no fire
        print("No fire detected")
    else:
        customWrite(1, '1') # Send '1' to indicate fire detected
        print("Fire detected, activating sprinkler")

def main():
    add_event_detect(0, handleSensorData) # Monitor fire monitor input on D0
    while True:
        delay(1000) # Check every second for fire

if __name__ == "main":
    main()
```

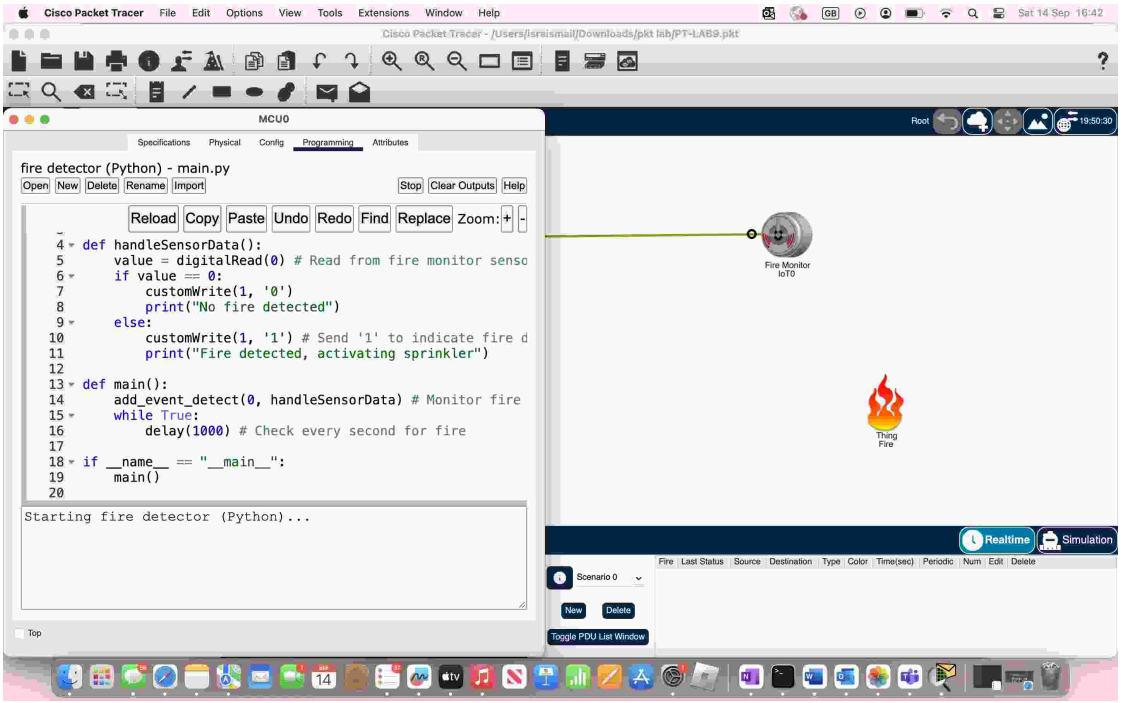


Figure 9.7: Python code reading the fire sensor and activating the sprinkler.

— How the Code Works.

- `handleSensorData()`: Reads pin `D0` (the fire sensor). If the sensor returns `1`, it signifies "fire detected," and the code sends `'1'` to pin `D1` (sprinkler). Otherwise, it sends `'0'` to indicate no fire.
- `add_event_detect(0, handleSensorData)`: Watches `D0` continuously and calls `handleSensorData()` whenever its state changes.
- `while True: delay(1000)`: Keeps the script running, checking for sensor updates every second.
- `customWrite(1, '1')`: A specialized Packet Tracer function that writes data to pin `D1`, where the sprinkler is connected, effectively turning it on or off.

F. Test the Program

13. **Run the Code:**

    - In the MCU's *Programming* tab, click the **Run** button to start the script.
    - The code now continuously listens on `D0` for any changes from the **Fire Monitor**.

14. **Simulate Fire:**

    - In the workspace, move the "**Fire**" device (your simulated heat source) closer to the **Fire Monitor**.
    - If the sensor detects fire (i.e., the **Fire Monitor** outputs a signal of "1" on `D0`), the MCU sends `'1'` to pin `D1`, indicating the sprinkler should turn on.

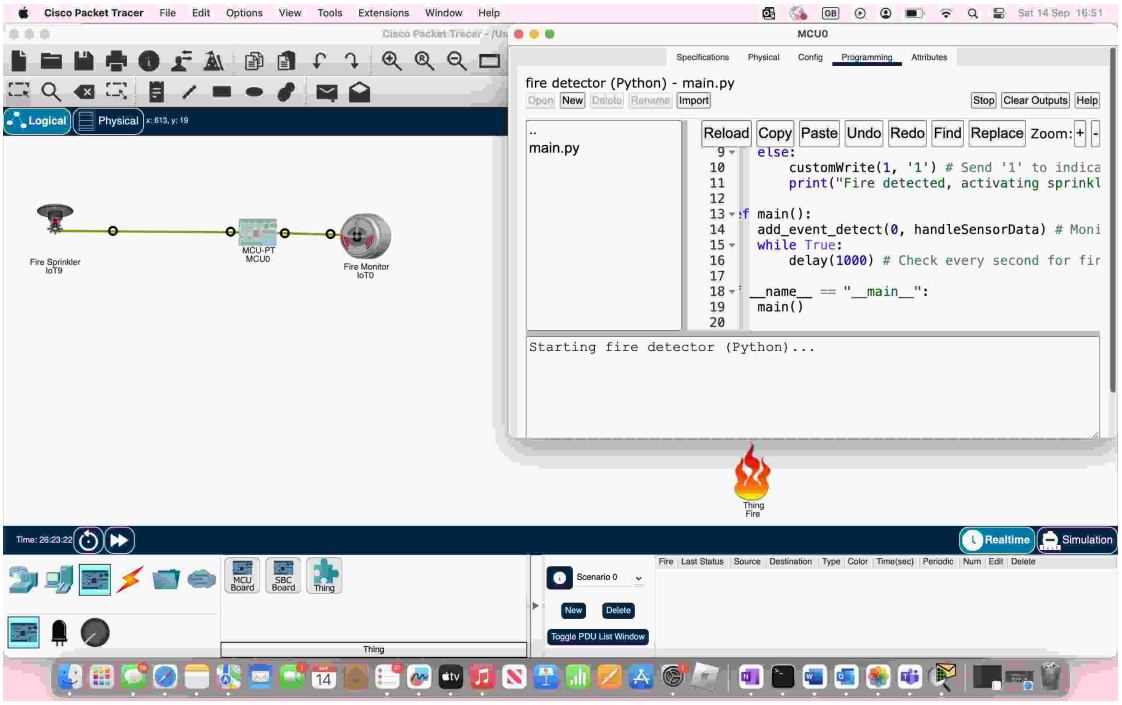


Figure 9.8: Starting the fire detector program on the MCU.

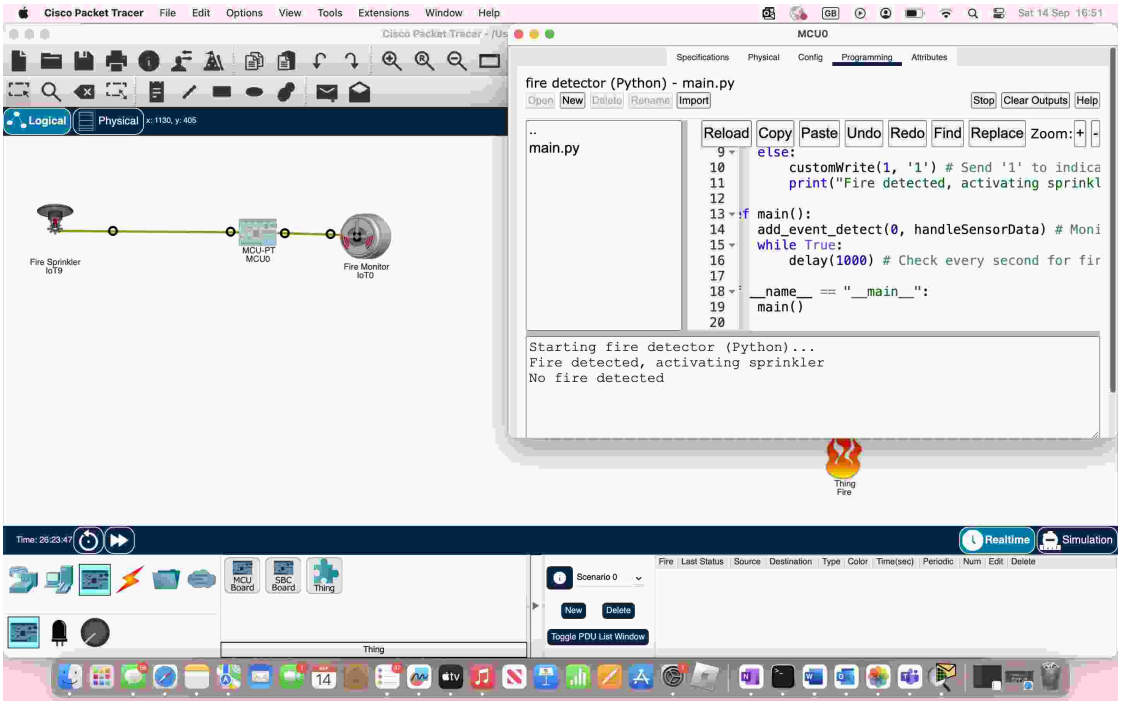


Figure 9.9: Fire is undetected when too far from the sensor.

15. **Observe Sprinkler Activation:**

- Moving "**Fire**" within range of the **Fire Monitor** triggers the sensor to read "fire."
- The MCU's code then writes `'1'` to the sprinkler pin, causing it to activate (e.g., visually

"spray water" or switch an icon from off to on).

- If you move "**Fire**" away, the monitor outputs "0," and the MCU turns the sprinkler off.

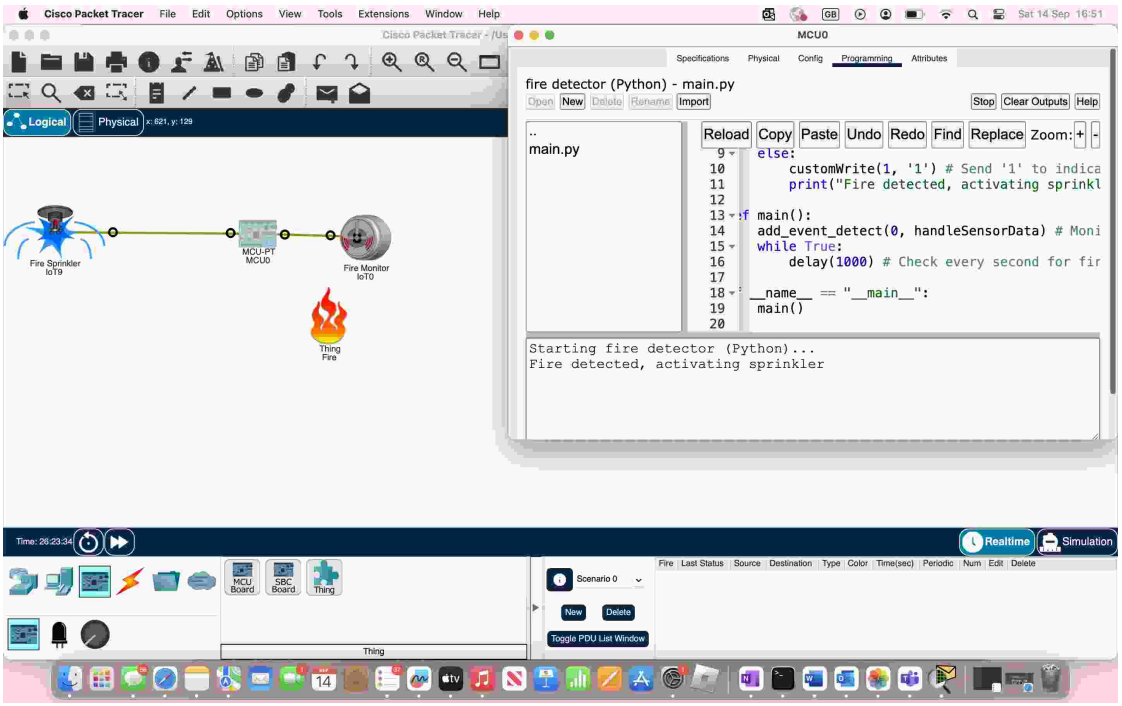


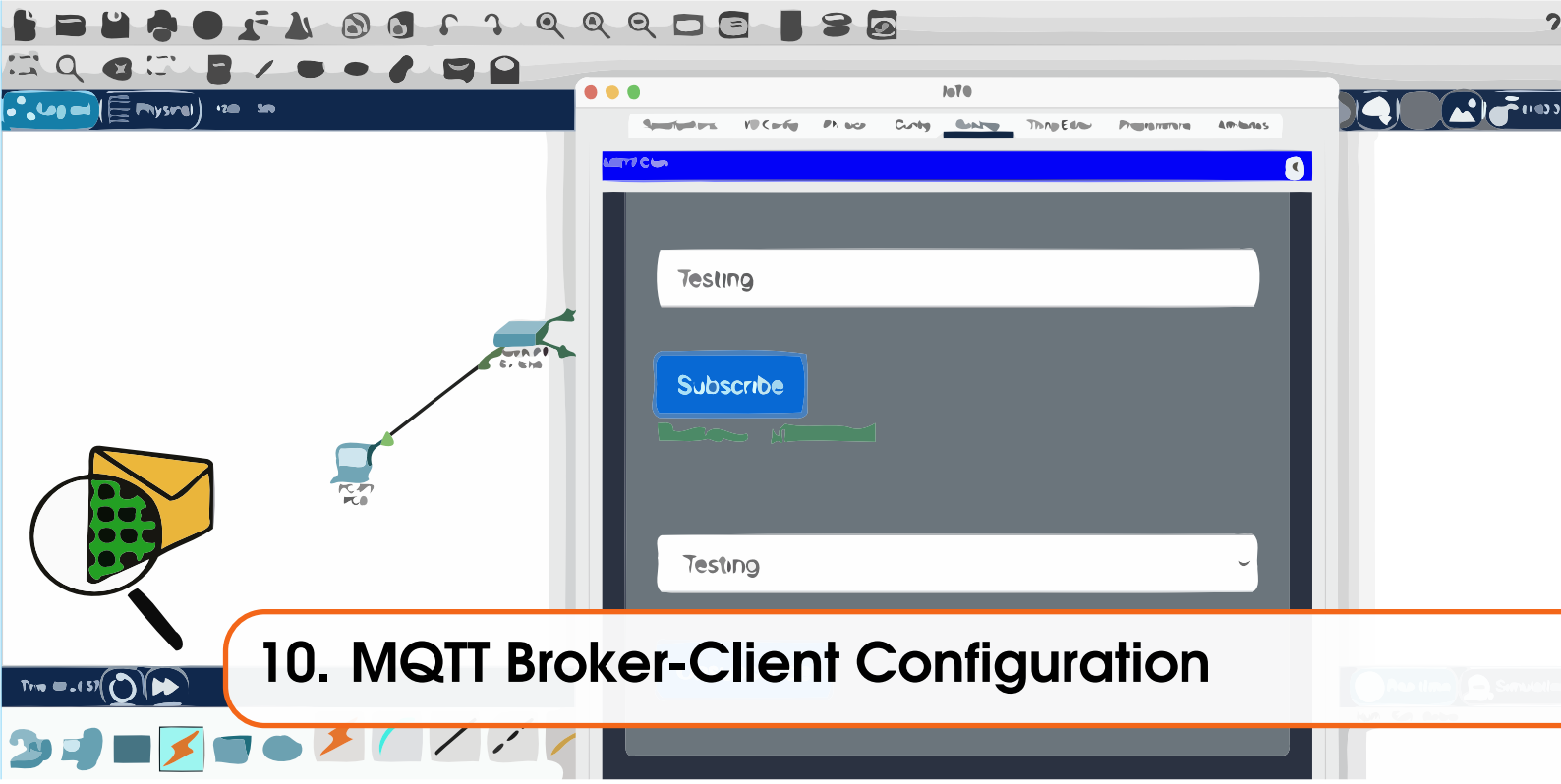
Figure 9.10: Sprinkler activates when fire is detected.

— Testing and Troubleshooting.

- **Console Output:** In the *Programming* tab's console, look for messages like `"No fire detected"` or `"Fire detected, activating sprinkler"` to confirm the code is running.
- **Sensor Placement:** If the **Fire Monitor** never reads fire, adjust the distance of the "**Fire**" device or check the *Thing Editor* settings for sensitivity.
- **Pin Conflicts:** Ensure no other scripts or hardware conflict with `D0` and `D1`. Each pin should be dedicated to the Fire Monitor or Sprinkler, respectively.

Summary

You have successfully built a **fire detection and sprinkler control system** in Cisco Packet Tracer. By **programming the MCU** in Python (or JavaScript syntax) to read the fire monitor input, you created a responsive scenario where the sprinkler automatically activates upon detecting fire. This demonstrates how IoT devices can collaborate in real time to handle emergencies—paving the way for more complex automation in Packet Tracer.



10. MQTT Broker-Client Configuration

Introduction

In this tutorial, you will learn how to **configure an MQTT broker and client** in Cisco Packet Tracer. You will create a small network with a router, switch, PC, and IoT device. The PC will act as the MQTT broker, while the IoT device will serve as the MQTT client. By the end, you will have tested publishing and subscribing to MQTT topics between the broker and client.

Objective

- **Create** a network topology (router, switch, PC, IoT Thing).
- **Configure** all devices with IP settings for communication.
- **Install and set up** an MQTT broker on the PC.
- **Install and set up** an MQTT client on the IoT device.
- **Verify** message exchange (topics) between the broker and client.

Lab Plan

A. **Creating the Topology** (Add router, switch, PC, IoT device)
B. **Connecting Devices** (PC–Switch, IoT–Switch, Router–Switch)
C. **Configuring the Router**
D. **Configuring the PC (MQTT Broker)**
E. **Configuring the IoT Device (MQTT Client)**
F. **Testing the MQTT Connection** (ping test, broker/client topics)

Required Software

- **Cisco Packet Tracer 8.x** (or newer)
- Basic knowledge of IP addressing and device configuration in Packet Tracer
- Familiarity with MQTT concepts (broker, client, topics)

A. Creating the Topology

1. **Add Devices:**
   From the *Device-Type Selection* box in Packet Tracer, locate and drag the following devices onto the workspace:
   - **Switch** (*Network Devices* → *Switches*)
   - **PC** (*End Devices*)
   - **IoT Thing** (*IoT Devices*)
   - **Router** (*Network Devices* → *Routers*)

   These are the core devices for setting up a basic MQTT demo network.
2. **Layout:**
   Place the **Router**, **Switch**, **PC**, and **IoT Thing** in the workspace so that cabling between them is straightforward. For instance:
   - *Router* on the left,
   - *Switch* in the middle,
   - *PC* and *IoT Thing* on the right.

   Your arrangement might vary, but aim to keep enough space for easy viewing and cable connections.

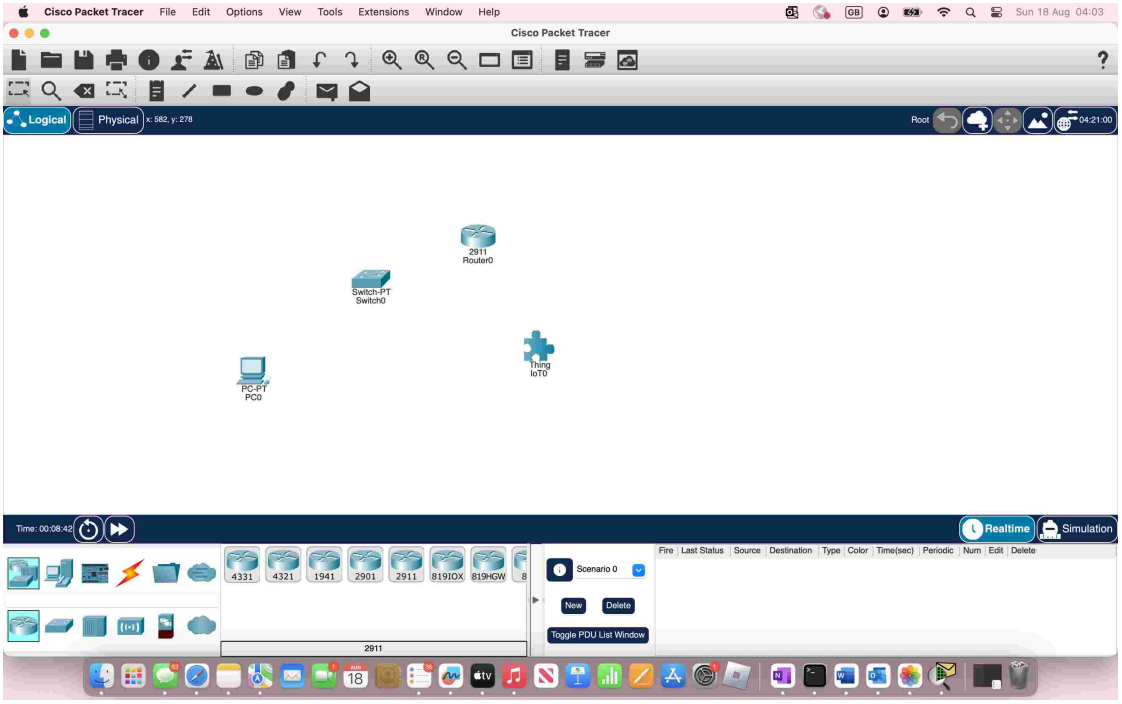


Figure 10.1: Initial topology with a router, switch, PC, and IoT Thing device.

— Why These Devices?.
- **Router:** Provides an IP gateway and possible DHCP services for the network.
- **Switch:** Links multiple devices on the LAN.
- **PC:** Will act as an MQTT Broker or client, depending on your lab setup.
- **IoT Thing:** Serves as the MQTT client, simulating an IoT sensor or actuator.

This selection ensures a minimal yet functional network to demonstrate MQTT messaging in Packet Tracer.

B. Connecting Devices

3. **PC and Switch:**

   - Select the *Connections* tool in Packet Tracer (the lightning bolt icon).
   - Choose **Automatically Choose Connection Type**.
   - Click on the PC's *FastEthernet0* port, then click on an available *FastEthernet* port on the **Switch**.

   The system should automatically pick the correct cable (typically a *straight-through*) for a PC-to-switch link.
4. **IoT0 and Switch:**

   - If you cannot directly connect **IoT0** to the switch, ensure the IoT device has the correct network adapter modules:
     (a) Click **IoT0**, then go to *Advanced* → **I/O Config**.
     (b) For `Network Adapter 1`, select `PT-IOT-NM-1W` (wireless module).
     (c) For `Network Adapter 2`, select `PT-IOT-NM-1CE` (wired module).
   - After confirming the correct adapters, click on **IoT0** and the **Switch** with *Automatically Choose Connection Type* enabled. A valid link should form.

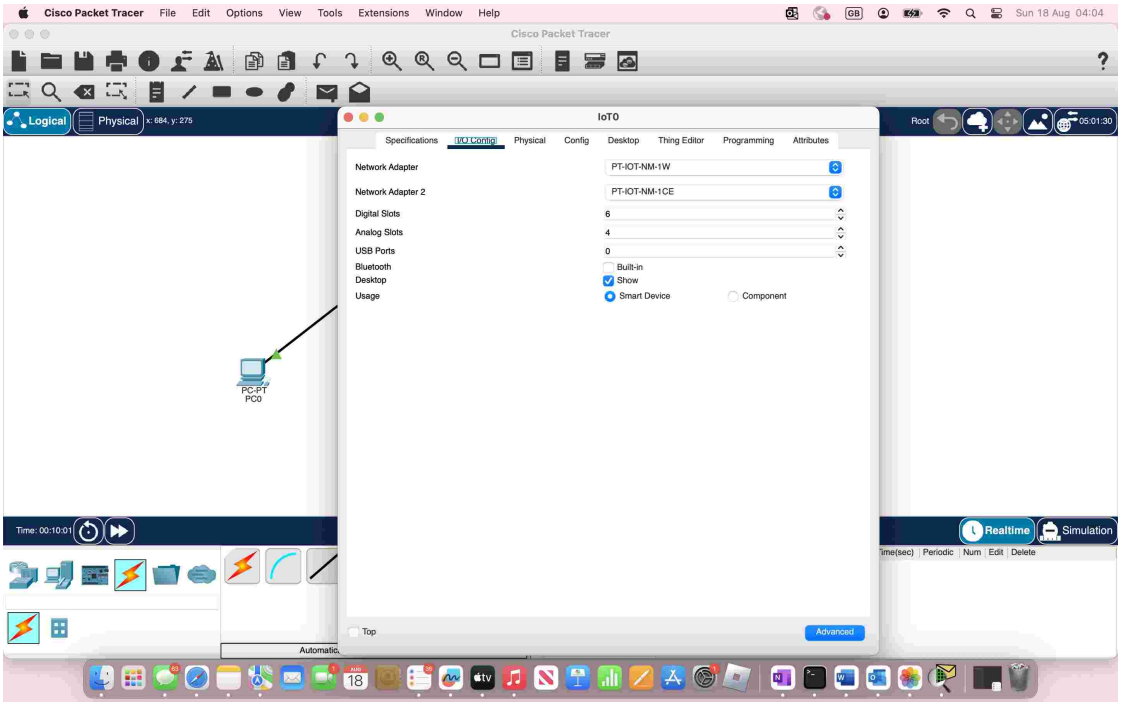


Figure 10.2: Configuring the IoT0 device to ensure it has the correct network adapters.

5. **Router and Switch:**

   - Again, use **Automatically Choose Connection Type**.
   - Click on the **Router**'s *GigabitEthernet0/0* port, then on an available *FastEthernet* or *GigabitEthernet* port on the **Switch**.
   - Wait a moment for the link lights to turn green, indicating a successful connection.

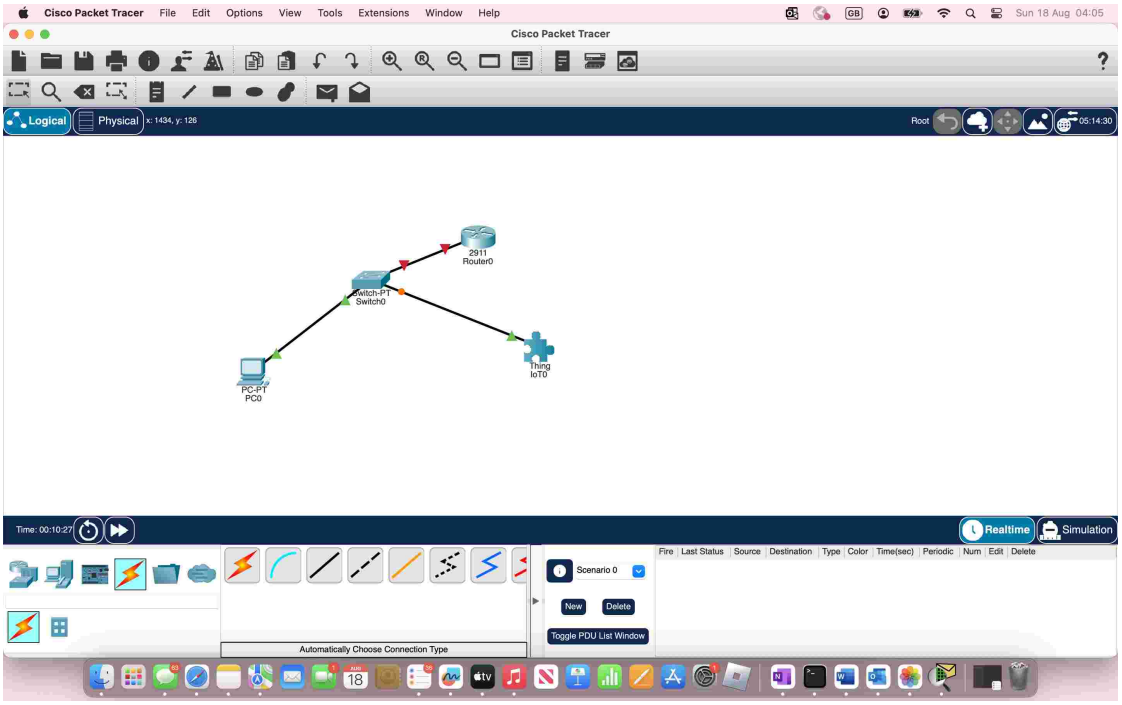


Figure 10.3: Connecting the router to the switch.

— Troubleshooting Connections.

- **No Link Lights?**
  - Check that both devices are powered on and the correct port is selected.
  - Verify the *Power* button is on (if visible) in the *Physical* tab for IoT devices.
- **Incorrect Cable Type?**
  - Manually choose a *Copper Straight-Through* cable if *Automatically Choose Connection Type* fails or picks the wrong option.
- **Missing Ports or Adapters?**
  - For IoT devices, confirm you've added the correct modules (`PT-IOT-NM-1W`, `PT-IOT-NM-1CE`) under *Advanced* → *I/O Config*.
  - For routers, ensure `GigabitEthernet0/0` is enabled in the *Config* tab or via the CLI.

C. Configuring the Router

6. **Open the Router:**

   - Click on the **Router** in your Packet Tracer workspace.
   - Select the *Config* tab at the top of the router's configuration window.

   You'll see a list of interfaces and global settings.
7. **Select `GigabitEthernet0/0`:**

   - Check the box or toggle labeled `Port Status` to turn the interface **On**.
   - In the **IP Address** field, enter `192.168.10.1`.
   - In the **Subnet Mask** field, enter `255.255.255.0`.

   This assigns the router's main interface an IP on the `192.168.10.x` subnet.
8. **Exit the Router Config:**
   Once you've entered the IP address and enabled the port, you can close the configuration

window or switch to another tab as needed.

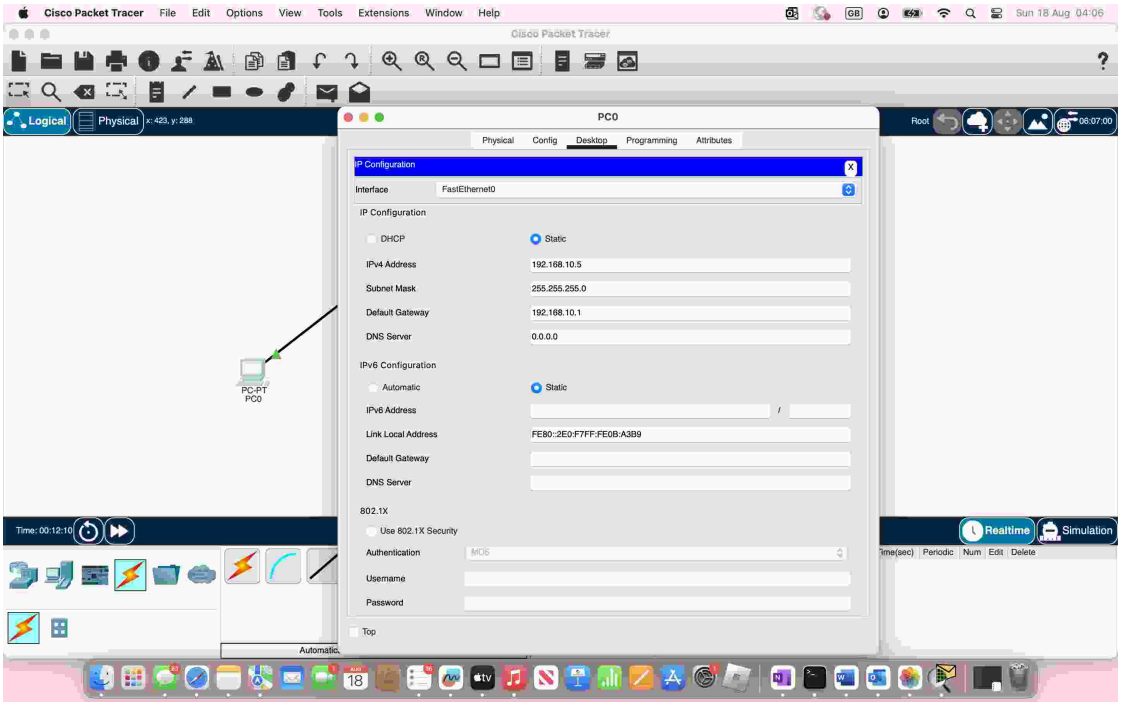


Figure 10.4: Assigning IP `192.168.10.1/24` on the router's interface.

— Why Configure the Router?.

- **Gateway for Devices:** By assigning `192.168.10.1` to `GigabitEthernet0/0`, the router becomes the default gateway for the network.
- **Essential for Internet or WAN Access:** If you expand this lab to connect to the broader internet or another subnet, the router directs traffic outside the local segment.
- **DHCP / Additional Services:** Later, you can enable DHCP, NAT, or other router functionalities to further manage the network.

D. Configuring the PC (MQTT Broker)

D.1 IP Settings

1. **Click the PC:**

   - On the **PC** device, select the *Desktop* tab.
   - Open **IP Configuration**.
   - Assign the following:
     - **IP Address:** `192.168.10.5`
     - **Subnet Mask:** `255.255.255.0`
     - **Default Gateway:** `192.168.10.1`

   This ensures the PC is on the same network (`192.168.10.x`) as the router and can route traffic through `192.168.10.1` if needed.

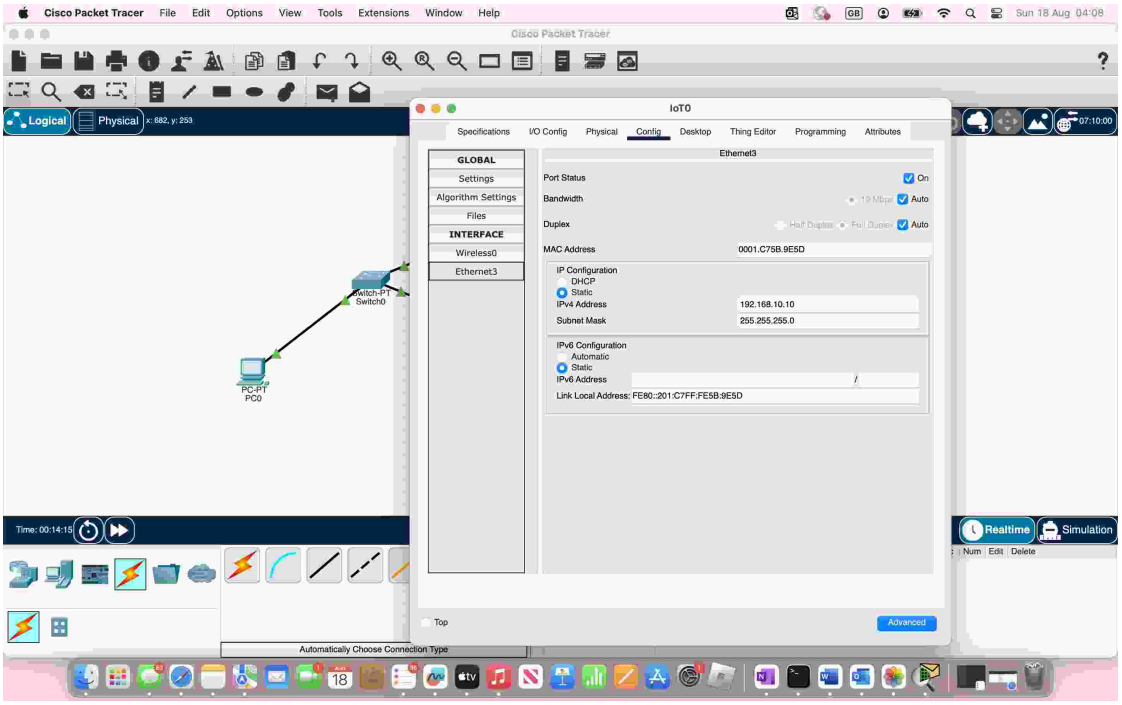


Figure 10.5: PC's IP settings: `192.168.10.5/24` with gateway `192.168.10.1`.

D.2 Installing the MQTT Broker

2. **User Apps Manager:**

   - Still on the **PC** device, open the *Desktop* tab.
   - Choose **User Apps Manager**.
   - Under the **Available** apps list, locate and install **MQTT Broker**.

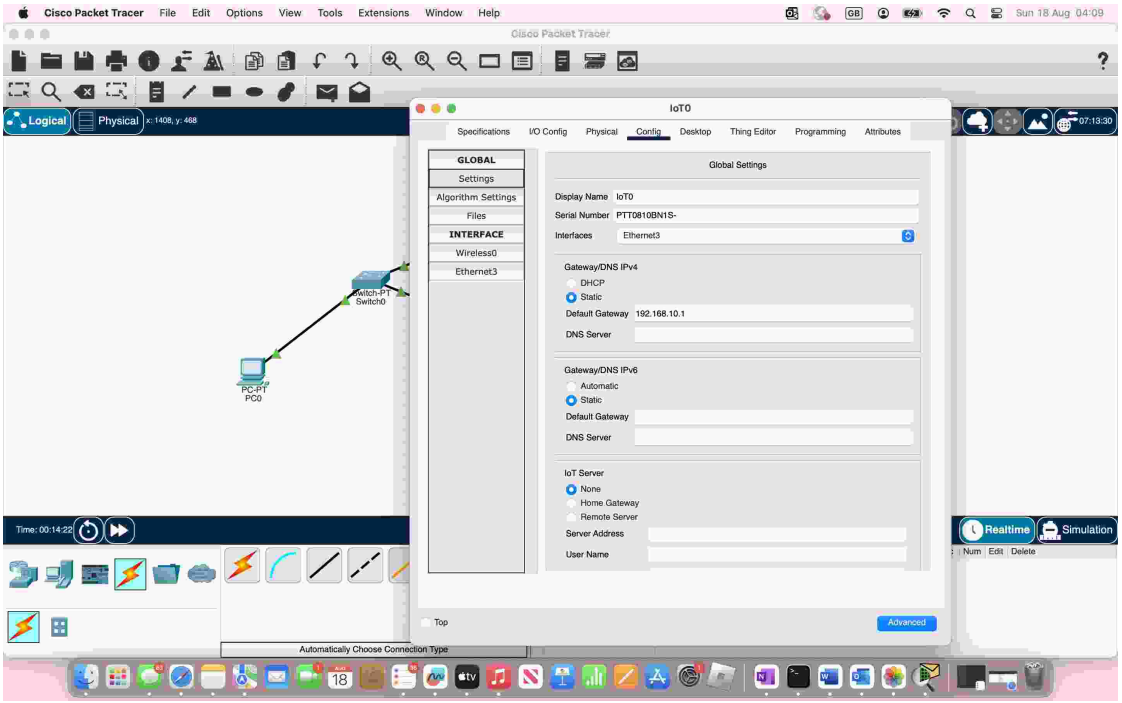


Figure 10.6: Installing the MQTT Broker application on the PC.

3. **Exit PC config** once installed.
   You've now set up the PC to serve as an **MQTT Broker**, ready to handle MQTT publish/subscribe traffic from IoT devices.

— Why Install an MQTT Broker?.

- **Central Messaging Hub:** The broker handles incoming messages (from "publishers") and routes them to subscribed "clients," enabling efficient decoupling of senders and receivers.
- **IoT Standard Protocol:** MQTT is widely used in IoT for lightweight messaging, making this lab a solid introduction to real-world IoT scenarios.
- **Scalable Setup:** Multiple devices can connect to the same broker, each subscribing to or publishing different topics (e.g., `"sensors/temperature"`, `"actuators/motor"`).

E. Configuring the IoT0 Device (MQTT Client)

E.1 IP Settings

1. **Click IoT0:**

   - In the Device Configuration window, select **Config** → **Ethernet3**.
   - Assign:
     - **IP Address**: `192.168.10.10`
     - **Subnet Mask**: `255.255.255.0`
     - **Gateway/DNS IPv4**: `192.168.10.1`

   This ensures **IoT0** is on the same `192.168.10.x` subnet and can reach the router at `192.168.10.1` for traffic outside its local segment.

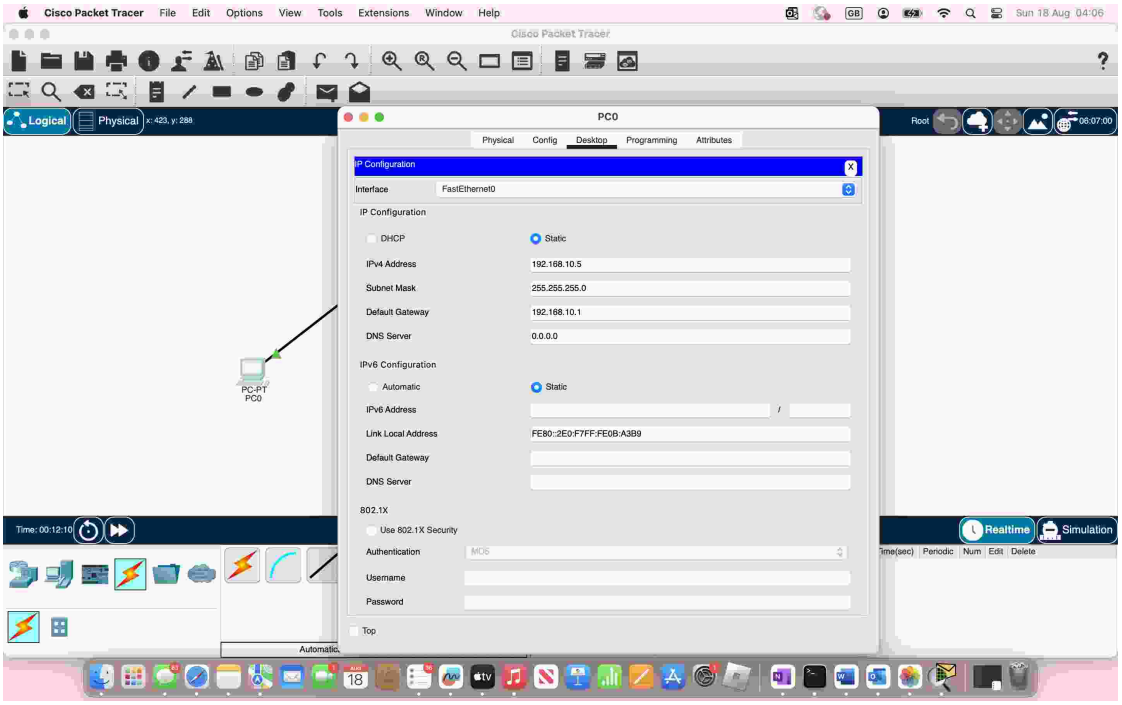


Figure 10.7: Assigning IP details to **IoT0** under **Ethernet3** (example figure).

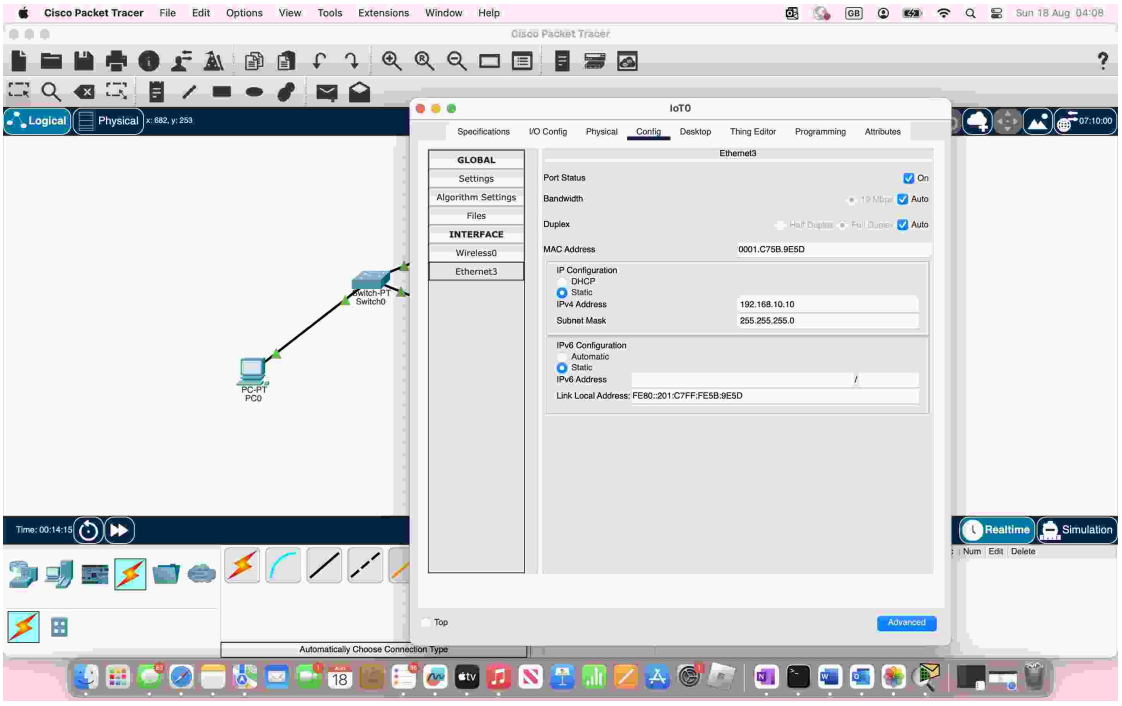


Figure 10.8: Verifying IP configuration in the IoT device (example figure).

E.2 Installing the MQTT Client

2. **Thing's Desktop → User Apps Manager:**

   - On the **IoT0** device, go to **Desktop → User Apps Manager**.

- Locate **MQTT Client** under the **Available** list and click to install it.

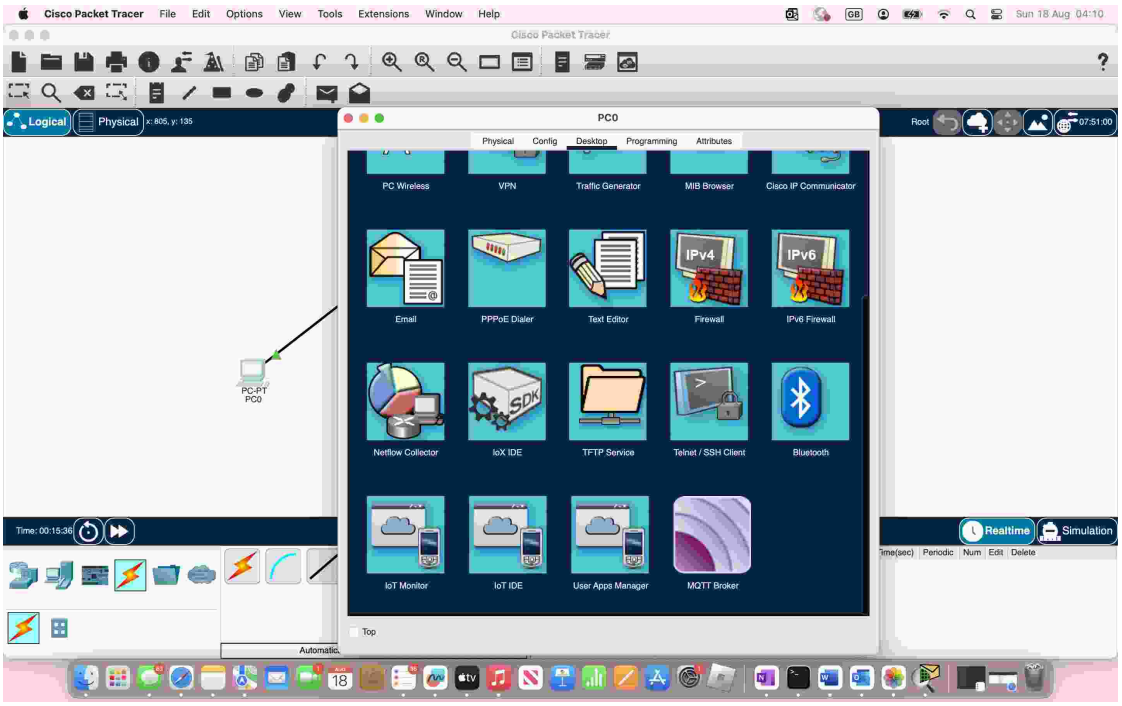


Figure 10.9: Installing MQTT Client on the IoT device.

3. **Exit the IoT device config** after installation.
   At this point, **IoT0** is ready to act as an MQTT client, capable of connecting to the **MQTT Broker** on the PC.

— Why an MQTT Client on IoT0?.

- **Simulated Sensor/Actuator:** The IoT device can publish messages (like sensor readings) or subscribe to topics (like commands from the broker).
- **Lightweight Protocol:** MQTT is well-suited for IoT due to its minimal overhead, making it perfect for resource-constrained devices.
- **Hands-On Learning:** This setup demonstrates real-world IoT patterns—devices sending data to a central broker that routes messages to subscribers.

F. Configuring the MQTT Broker on the PC

4. **Open the MQTT Broker App:**

   - On the PC (which you set up as the MQTT Broker), switch to the *Desktop* tab.
   - Click on **MQTT Broker** to launch the application interface.

5. **Create a New User:**

   - In the MQTT Broker window, look for a **Username** and **Password** field.
   - Enter `admin` for both:
     - **Username:** `admin`
     - **Password:** `admin`
   - Click **Add** to register this new user.

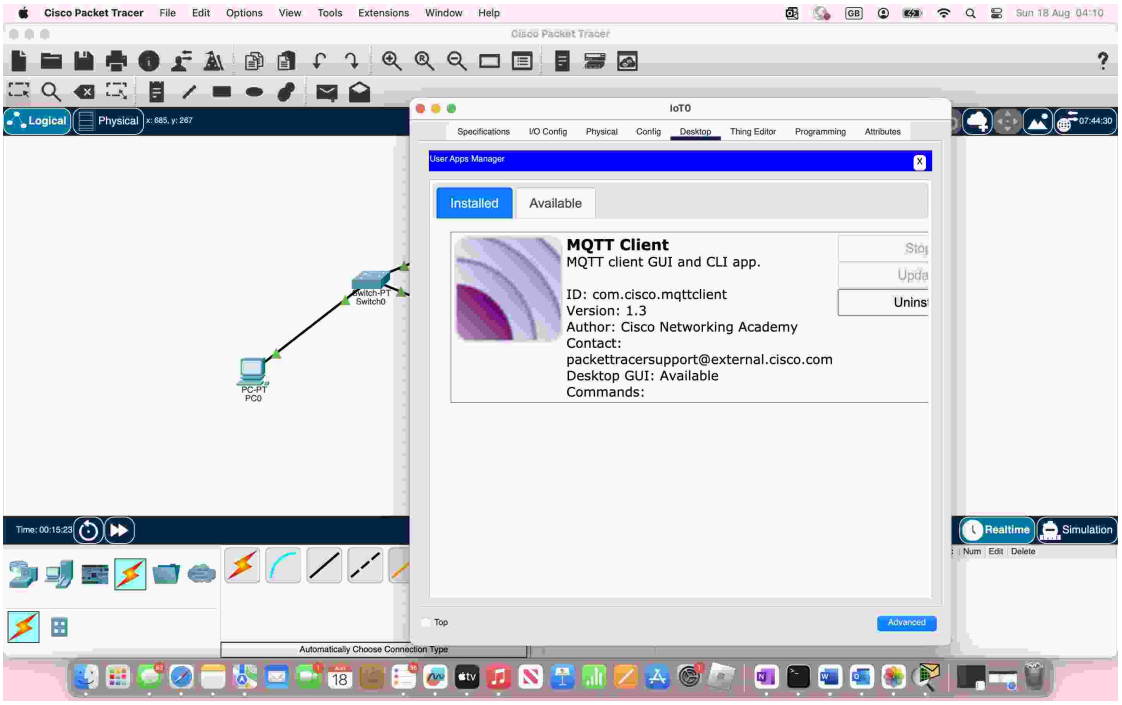


Figure 10.10: MQTT Broker application on the PC.

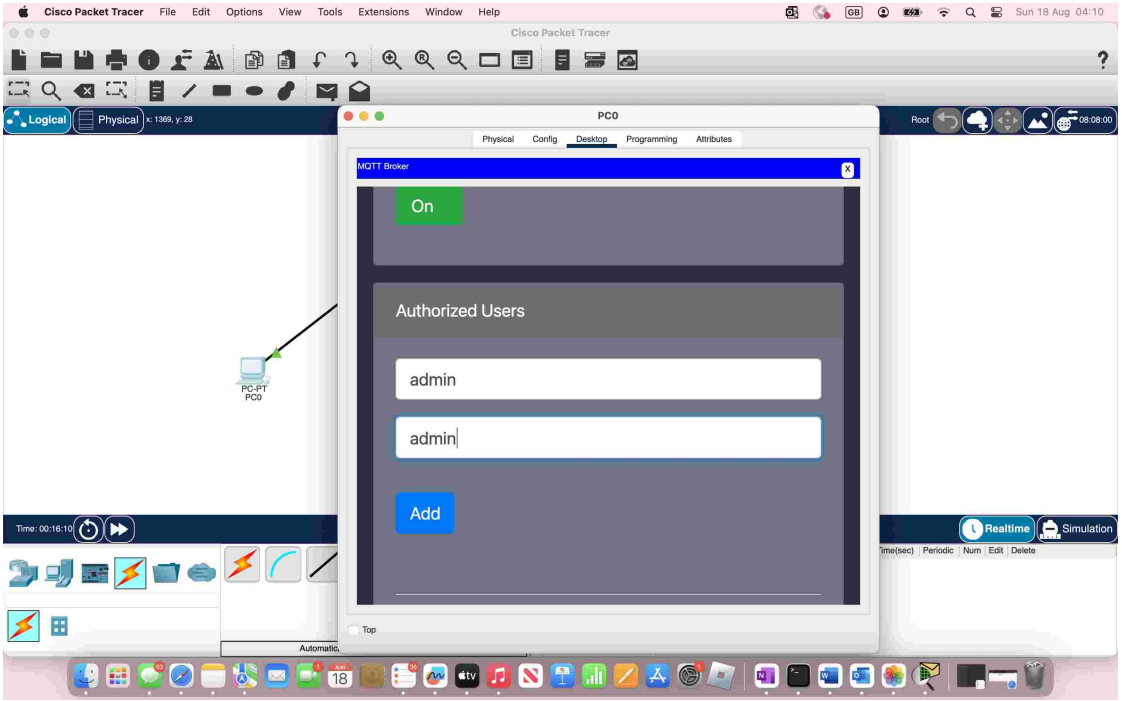


Figure 10.11: Creating credentials (`admin`/`admin`) for the MQTT broker.

— Why Add MQTT Broker Credentials?.

- **Secure Access:** User credentials (`admin`/`admin`) allow only authorized clients to connect, subscribe, or publish to topics.
- **Basic Authentication Demo:** Although it's a simple lab setup, in real-world scenarios

you'd use unique usernames/passwords or tokens for improved security.
- **Multi-User Scenarios:** Having multiple accounts lets different IoT devices or groups connect to the broker with distinct privileges.

G. Configuring the MQTT Client on the IoT Thing

6. **IoT Device (MQTT Client Settings):**

   - On the **IoT Thing** device, open the *Desktop* (or *Applications*) tab where the **MQTT Client** was installed.
   - Set the **Broker Address** to `192.168.10.5`, which is the PC's IP address (i.e., where the MQTT Broker is running).
   - Enter the **Username** and **Password** you created on the broker (e.g., `admin/admin`).
   - Click **Connect** to establish the MQTT connection.

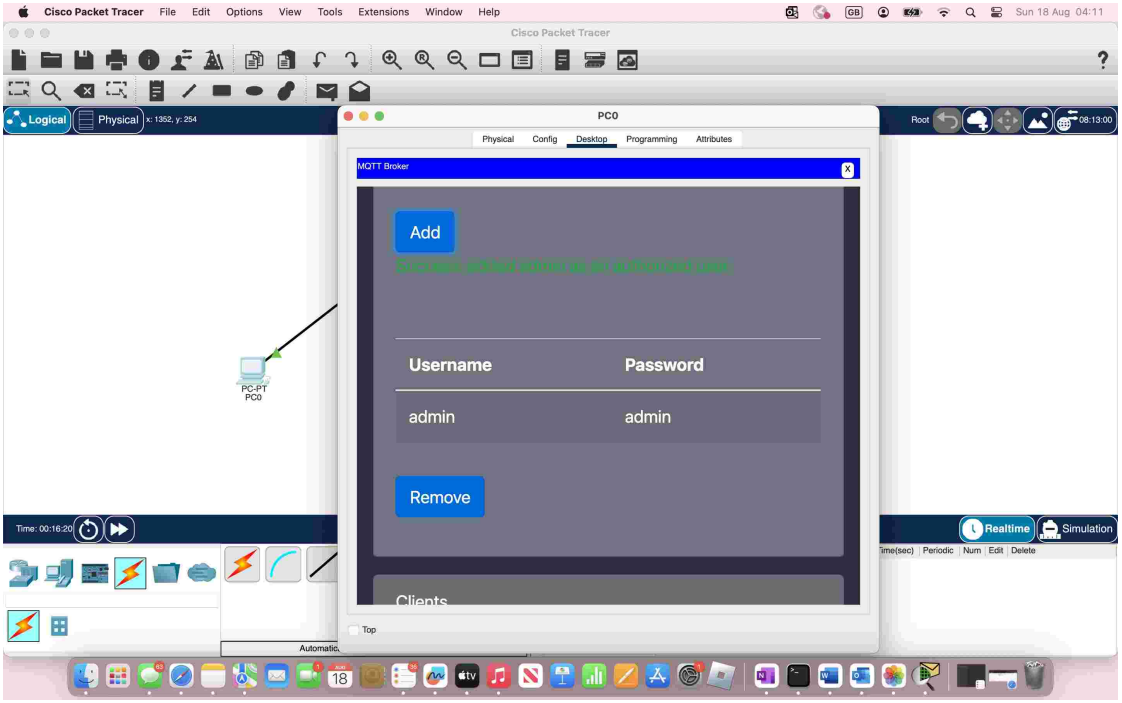


Figure 10.12: Configuring the MQTT client on the IoT device.

— **Why Configure MQTT on the IoT Device?.**
- **Message Publishing/Subscribing:** Once connected, this IoT device can *publish* sensor data (e.g., temperature readings) to the broker or *subscribe* to topics for receiving commands (e.g., turn on/off).
- **Verification of Connectivity:** If the client successfully connects, it confirms that your network settings (IP, gateway) and broker credentials are correct.
- **Scalable IoT Setup:** You can add more IoT clients (with different addresses or on different topics) to simulate a fully functional MQTT-based IoT environment.

H. Testing the MQTT Connection

H.1 Ping Test

1. **On the PC:**

- Go to the PC's *Desktop* → **Command Prompt**.
- Verify connectivity by pinging:
  - `ping 192.168.10.1` (the router's IP)
  - `ping 192.168.10.10` (the IoT device's IP)

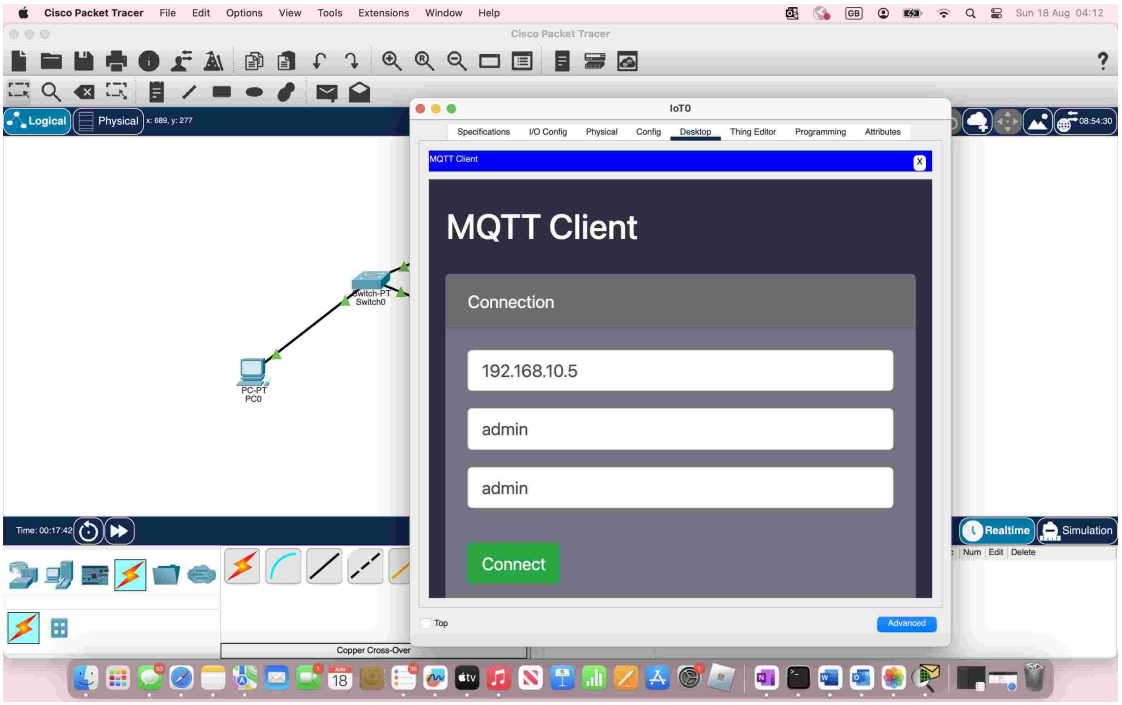


Figure 10.13: Verifying network connectivity via ping commands.

— Why Ping?.
- **Layer 3 Confirmation:** A successful ping ensures that the devices can reach each other at the IP layer.
- **Troubleshoot Early:** If pings fail, check cabling, IP assignments, or gateway settings before proceeding with MQTT configuration.

H.2 Verifying MQTT Connection

2. **Check the IoT Device:**
   - On the IoT device (where the *MQTT Client* app is installed), ensure it displays a status of **Connected** to the broker.
   - If not, re-check the broker's IP address, port, username, or password for any typos.
3. **Check the Broker:**
   - On the PC's *MQTT Broker* window, look under "Clients" or "Connected Clients" to see if the IoT device is listed.
   - A recognized client indicates a successful handshake between the IoT device and the broker.

Topic Test

4. **Set a Topic on the IoT Device:**
   - In the IoT device's MQTT Client interface, find the `Topic` field.

- Type a test topic name such as `Testing` (or `myTopic`).
- Publish a sample message to confirm the broker receives it.

5. **Broker View:**

- On the broker's console, you should see a log entry or a topic list entry reflecting "`Testing`."
- If you have another client subscribed to "`Testing`," it can receive any published messages from the IoT device.

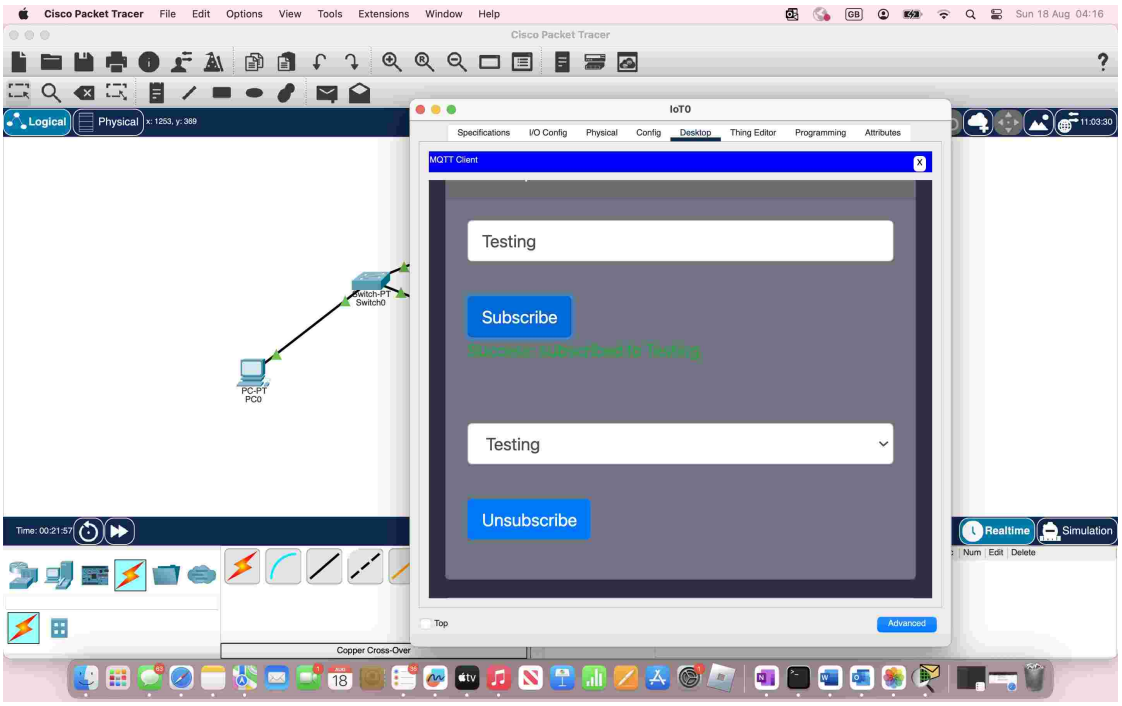


Figure 10.14: IoT device publishing/subscribing to topic "Testing."

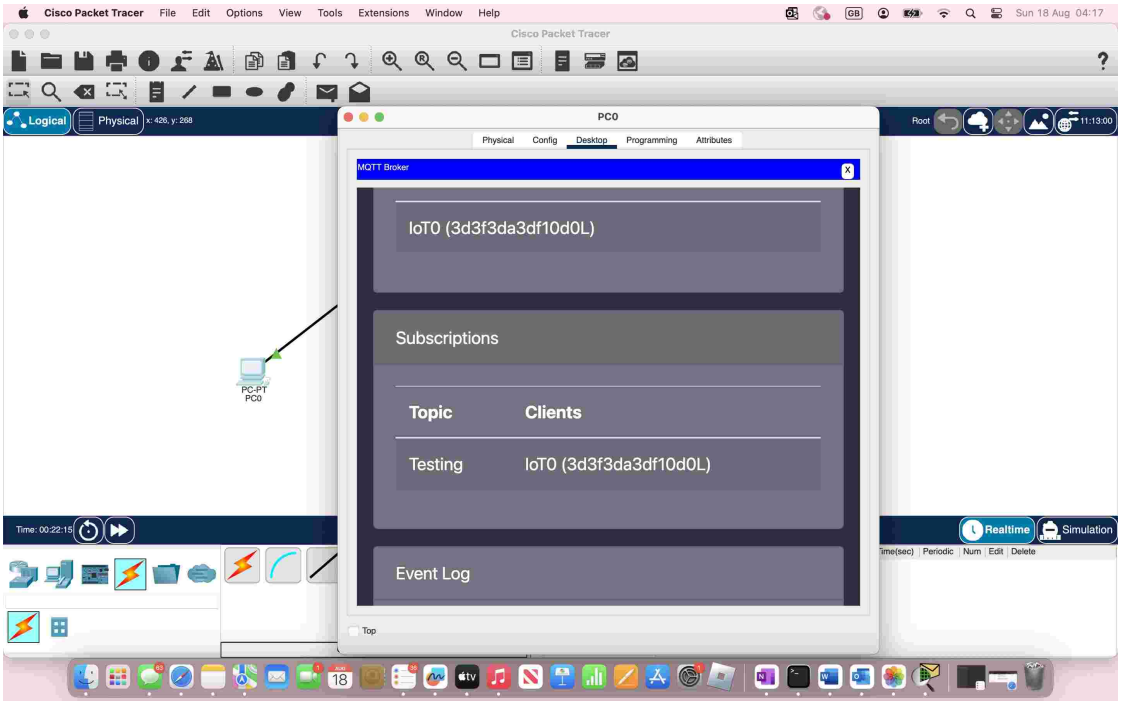


Figure 10.15: Broker console reflecting the new topic exchange.

— **MQTT Troubleshooting Tips.**

- **Check the Broker's IP and Port:** Some MQTT setups use port 1883 by default. Ensure Packet Tracer's broker is listening on that (or whichever port is configured).
- **Credentials Mismatch:** If your IoT device can't connect, confirm the username/password match those added on the broker.
- **Firewall or Security Settings:** In certain scenarios, Packet Tracer might prompt to allow ports. Confirm you allowed the port binding.
- **Multiple Clients:** Try adding a second IoT device or even another PC client. Subscribe one to a topic, publish from the other, and watch the message pass through the broker.

Summary

Congratulations! You have successfully **created an MQTT broker-client setup** in Packet Tracer. After **configuring the router** (192.168.10.x network), **assigning IPs** to the PC and IoT device, **installing** the MQTT broker on the PC, and **installing** the MQTT client on the IoT device, you tested the connection by publishing/subscribing to a topic. This verifies that your MQTT communication is **fully operational**—enabling IoT messaging and data exchange in Packet Tracer.