



## Stichting NIOC en de NIOC kennisbank

Stichting NIOC ([www.nioc.nl](http://www.nioc.nl)) stelt zich conform zijn statuten tot doel: het realiseren van congressen over informatica onderwijs en voorts al hetgeen met een en ander rechtstreeks of zijdelings verband houdt of daartoe bevorderlijk kan zijn, alles in de ruimste zin des woords.

De stichting NIOC neemt de archivering van de resultaten van de congressen voor zijn rekening. De website [www.nioc.nl](http://www.nioc.nl) ontsluit onder "Eerdere congressen" de gearchiveerde websites van eerdere congressen. De vele afzonderlijke congresbijdragen zijn opgenomen in een kennisbank die via dezelfde website onder "NIOC kennisbank" ontsloten wordt.

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2025, gehouden op donderdag 27 maart 2025 jl. en georganiseerd door Hogeschool Windesheim). Bij elkaar zo'n 1500 bijdragen!

We roepen je op, na het lezen van het document dat door jou is gedownload, de auteur(s) feedback te geven. Dit kan door je te registreren als gebruiker van de NIOC kennisbank. Na registratie krijg je bericht hoe in te loggen op de NIOC kennisbank.

Het eerstvolgende NIOC vindt plaats in 2027 en wordt dan georganiseerd door HAN University of Applied Sciences. Zodra daarover meer informatie beschikbaar is, is deze hier te vinden.

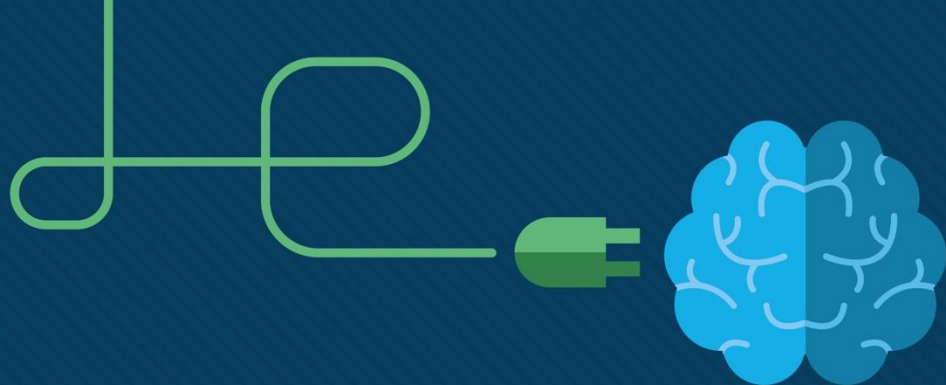
Wil je op de hoogte blijven van de ontwikkeling rond Stichting NIOC en de NIOC kennisbank, schrijf je dan in op de nieuwsbrief via

[www.nioc.nl/nioc-kennisbank/aanmelden\\_nieuwsbrief](http://www.nioc.nl/nioc-kennisbank/aanmelden_nieuwsbrief)

Reacties over de NIOC kennisbank en de inhoud daarvan kun je richten aan de beheerder:

R. Smedinga [kennisbank@nioc.nl](mailto:kennisbank@nioc.nl).

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.



# Connecting Smart Things in PT7

Cisco Packet Tracer Workshop

Eugene Morozov, Cisco Netacad Technical Manager  
Yvan Rooseleer, Odisee University Brussels/ BiASC/ NASC

December 2017 – Stockholm – all parts  
March 2018 – Leeuwarden – part 1



# Agenda

1

IoT Features

2

IoT Systems Demo

3

Build Your Own Device!

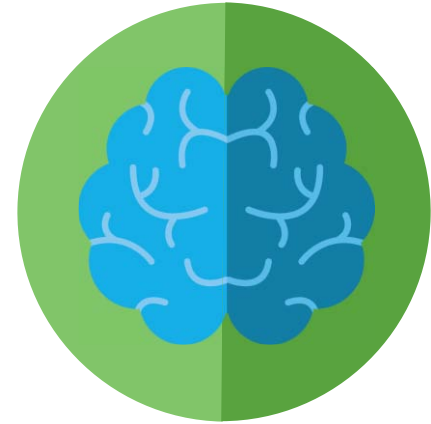
# New in Version 7



New network  
protocols



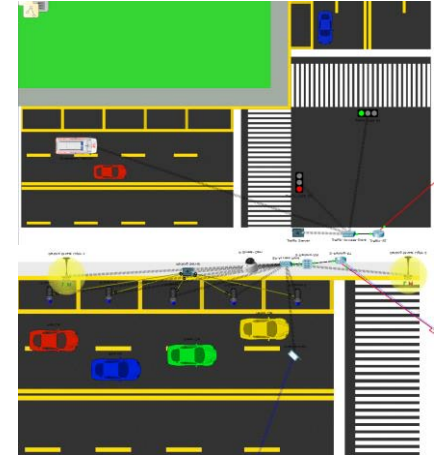
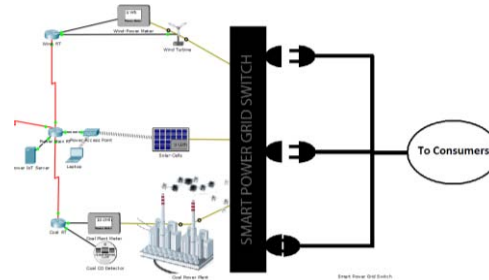
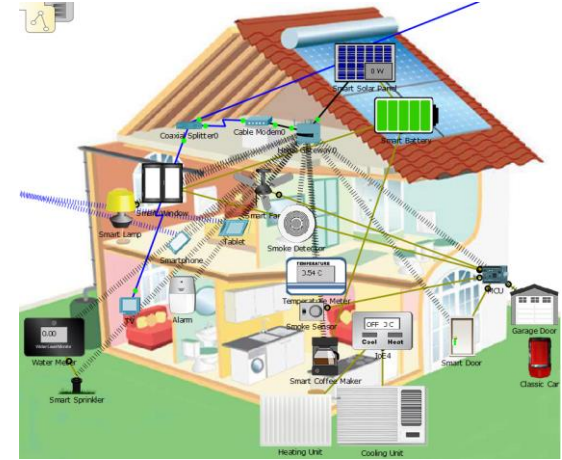
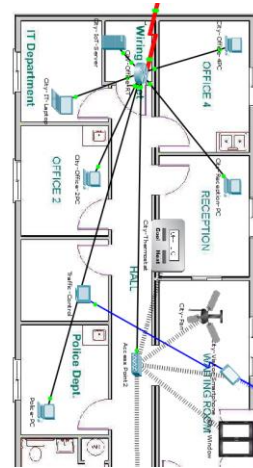
Environment and  
IoT



Programming

# IoT Highlights 7.0

- Physical Environments
- Smart devices, sensors and actuators
- Smart Home, Smart City, Industrial, Power Grid
- Edit existing or program your own devices
- Python, Javascript, Blockly
- SBC and MCU
- Home Gateway and Reg Server
- Rules for devices to work together
- Routers 819 and 829



# Basic Demo

# Build your own device

# Steps to make

1

Create smart device

2

Connect to registration server

3

Real environment challenge  
Data visualization challenge



# Creating smart device

# Project: Smart Greenhouse



- Monitors temperature
- If temperature is too high, opens an articulated vent
- Threshold temperature can be set from server

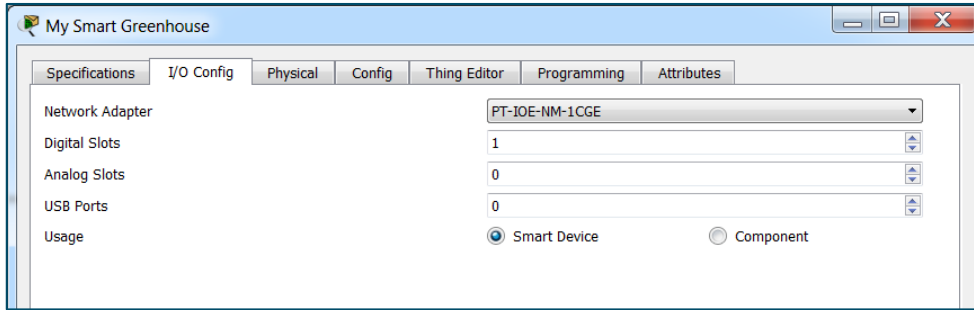
# Create the thing



- Add new generic “Thing”
- Give it a better, unique name

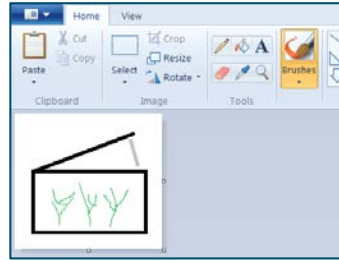
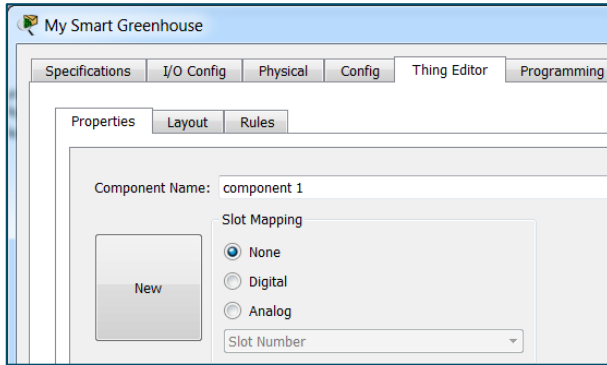


# Configure input/output



- Go to I/O Config tab
- Configure interfaces:
  - wired network
  - one digital slot
  - no analog or USB ports
- Usage: Smart Device

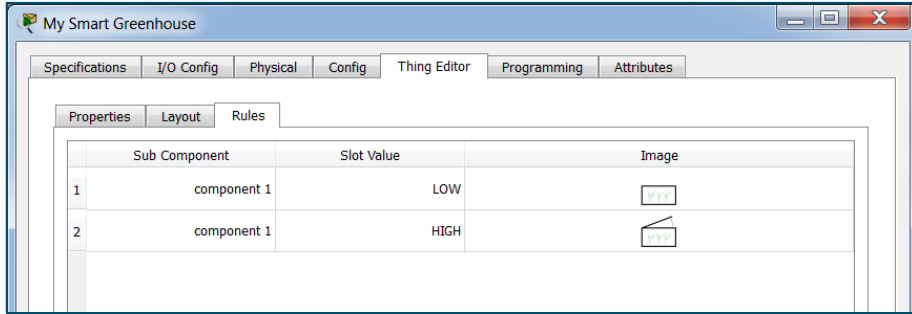
# Set the images



- Go to Thing Editor tab
- Your device will have only one component
- Open any graphics editor, like Paint
- Create two matching images, one for closed state and another for open state
- Click New to add images
- Both images should be added into component 1
- Map images to digital slot 0.

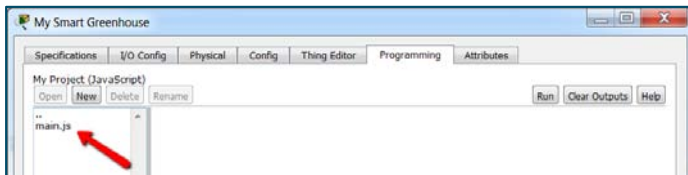
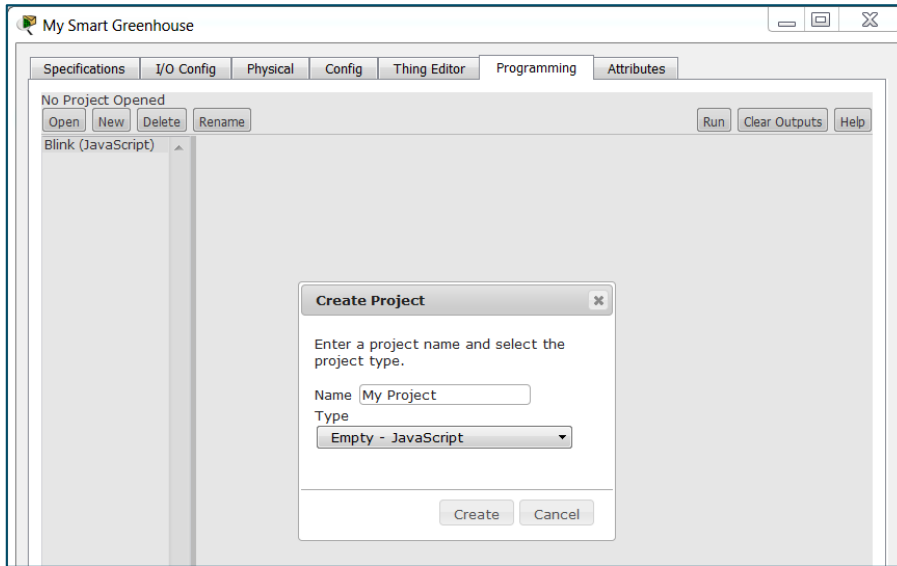


# Assign images to slot values



- Images will change when the slot value changes
- Go to Thing Editor - Rules
- Create two rules:
  - First rule maps LOW value to “closed” image
  - Second rule maps HIGH value to “open” image

# Programming



- That's it, we are ready to write programming code for this device
- On the Programming tab, create new project in Javascript or Python (this example uses Javascript)
- Open your project, then main.js file which should be empty

# Coding for articulation

```
1 var state = 0;           // 0 closed, 1 open
2 var temp = 0;           // current temperature
3 var threshold = 30;     // value of temperature to open the vent
4
5 function setup() {
6   digitalWrite(0, state); // set the initial state which is 0=closed
7 }
8
9
10 function loop() {
11   var temp = Environment.get("Ambient Temperature");
12
13   Serial.println("Current temperature: "+temp);
14
15   if (temp > threshold) {
16     state = 1;
17   } else {
18     state = 0;
19   }
20
21   digitalWrite(0, state);
22   delay(1000);
23 }
24
```

This variable will store the current state

This variable will store the current temperature

This variable defines the threshold temperature to open the vent

Read the temperature value from environment

Output the temperature to console

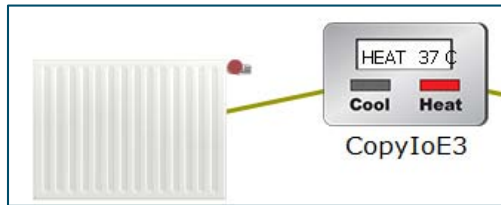
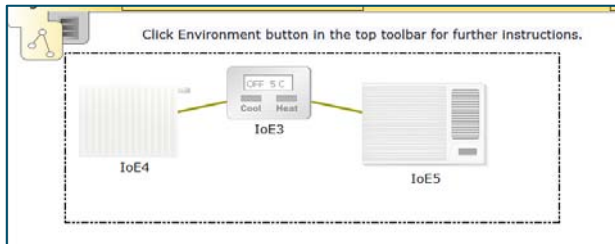
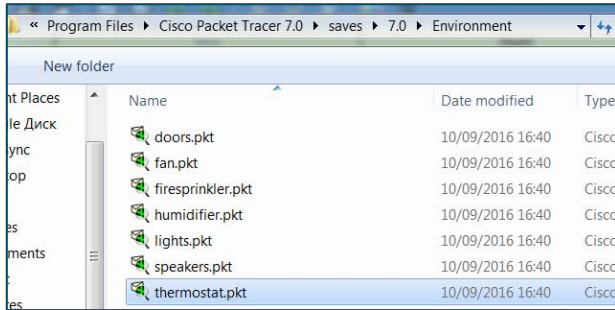
If temp is too high, change the state

Make current state effective by writing to slot 0

Wait for 1 sec



# Testing



- You device should now work – automatically open the vent if it's too hot
- We set the threshold temperature to 30 degrees
- To test it, we need to make our environment hotter:
  - Launch a new instance of Packet Tracer
  - Go to File – Open Samples – 7.0 – Environment – thermostat.pkt
  - Select and copy [Ctrl-C] all three devices
  - Paste [Ctrl-V] into your PT next to the smart device
  - Alt-click the Thermostat device to change to HEAT mode
- Heat and cool the room and observe how your device reacts

# Coding for Reg Server

```
IoEClient.setup(  
  {  
    type: "Smart Greenhouse",  
    states: [{  
      name: "Status",  
      type: "options",  
      options: {  
        "0": "Closed",  
        "1": "Open",  
      }  
    }],  
    {  
      name: "Temperature",  
      type: "number",  
      unit: "&deg;C",  
      decimalDigits: 1  
    },  
    {  
      name: "Threshold Temperature",  
      type: "number",  
      unit: "&deg;C",  
      decimalDigits: 1,  
      controllable: true,  
      minValue: 10,  
      maxValue: 100  
    }  
  ]  
});  
  
IoEClient.onInputReceive = function(input) {  
  if (input) {  
    input = input.split(",");  
    threshold = parseInt(input[2]);  
  }  
};
```

Three parameters are exchanged with Reg Server:

- Current state (open/closed)
- Current temperature
- Threshold temperature

Threshold temperature can be changed from the Reg Server

This is to catch when threshold value changed from server

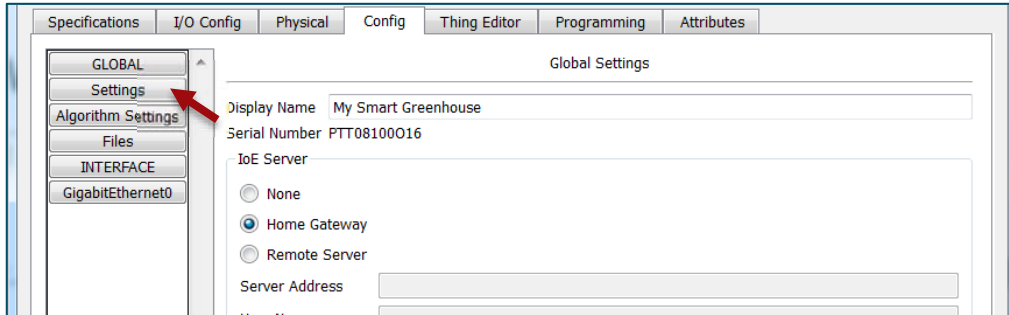
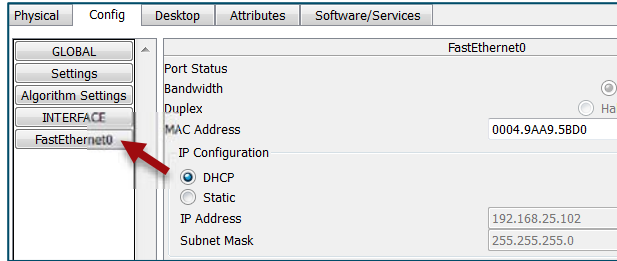
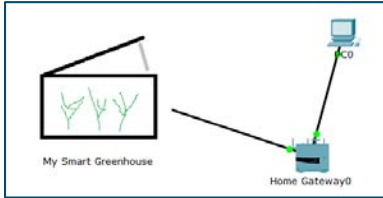
- To make your device talk with Registration server, you need to add additional instructions
- Add this code anywhere within the setup() {...} function body (between curly brackets)
- This piece of code defines parameters exchanged with Reg Server

# Coding for Reg Server (cont)

```
IoEClient.reportStates(state+ "," +temp+ "," +threshold);
```

- This small line is needed to report current values of three parameters (state, temp, threshold) to Reg server
- It should be executed every second
- Put it in the end of the loop() {...} function body before “delay(1000);”

# Testing with Home Gateway (optional)



- Wire your device to Home Gateway
- Configure it to receive IP address automatically
- Set it to talk the IoT language with it's gateway
- Wire a PC to Home gateway and configure it for DHCP as well
- On the PC, browse to 192.168.25.1, use "admin" as username and password

# Connecting to server

# Set up Multiuser



Multiuser Connection

Multiuser Connection

Connection Type:

Peer Address:

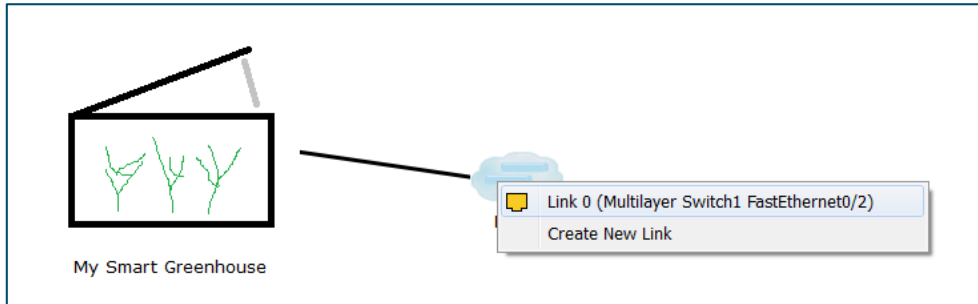
Peer Port Number:

Peer Network Name:

Password:

- Connect to real WiFi network
  - SSID: netacad
  - Password: workshop
- Add multiuser cloud to workspace
- Configure multiuser cloud:
  - Type: Outgoing
  - Peer Address: 192.168.1.100
  - Port: 38000
  - Peer name: pick a unique name
  - No password
- Ask facilitator to connect your multiuser link

# Connect to Reg Server



IP Configuration

DHCP  
 Static

Default Gateway

IP Address

IoE Server

None  
 Home Gateway  
 Remote Server

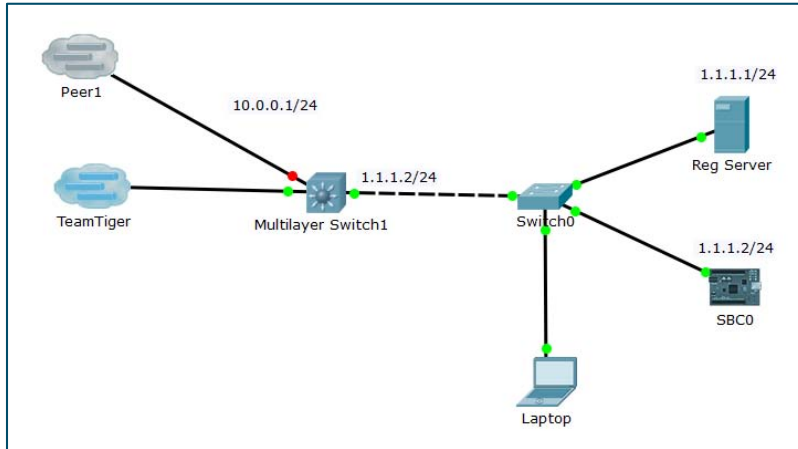
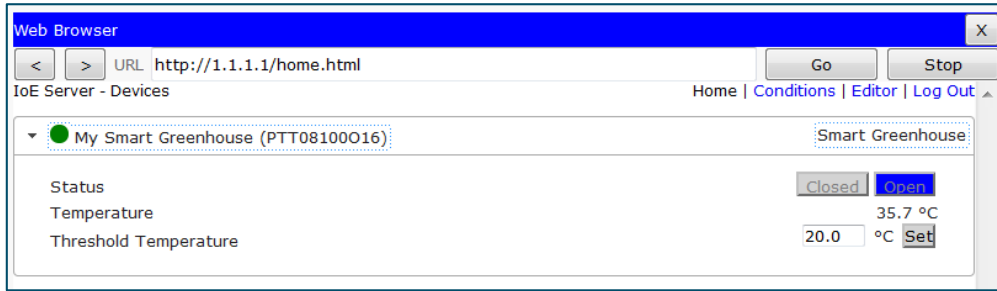
Server Address

User Name

Password

- When your connection is wired on the other side, it will become gold then blue
- Wire your smart device to the multiuser cloud using the existing link
- Configure your device to use DHCP
- Configure your device to use Registration server:
  - Address: 1.1.1.1
  - User Name: IoT
  - Password: system
- **Click Connect**

# Connect to Reg Server



- Your device should become visible on the IoT Server
- Your device reports current temperature and status
- Threshold temperature can be controlled from the server
- Ask facilitator to verify your device



# Challenges



Read data from  
real sensors



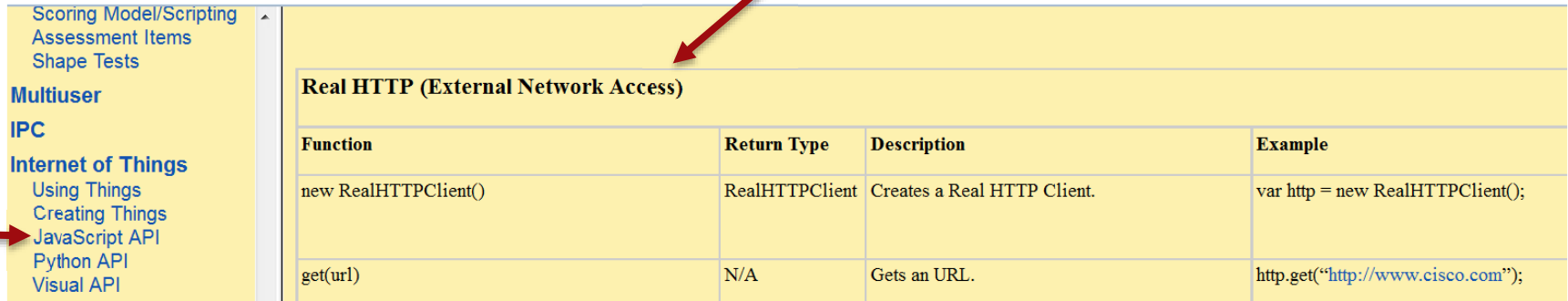
Draw a graph of  
your data flow

Let's go!

# Real environment challenge

# “Real” Javascript API

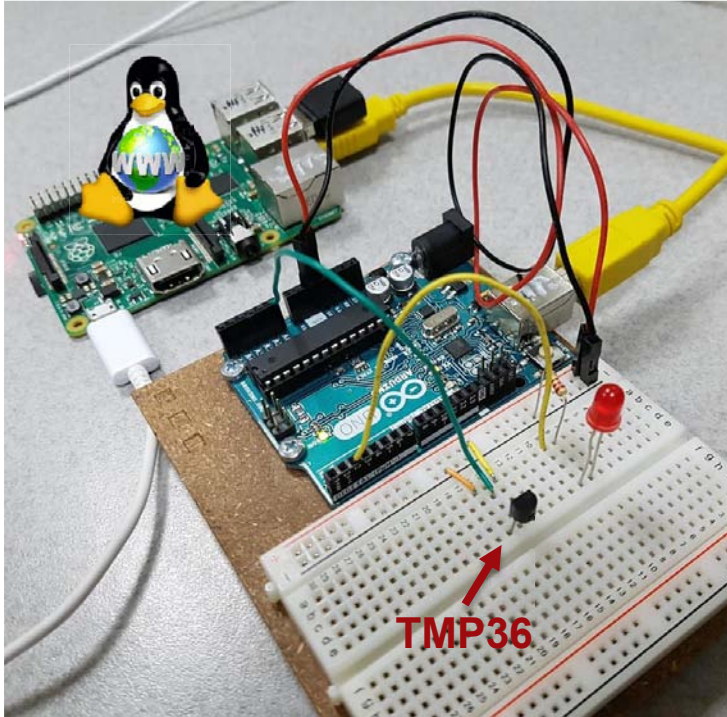
- In Packet Tracer 7, it is possible to communicate with real world using TCP, UDP, and HTTP protocols. Functions that help to do that described in Javascript API:



Real HTTP (External Network Access)			
Function	Return Type	Description	Example
new RealHTTPClient()	RealHTTPClient	Creates a Real HTTP Client.	var http = new RealHTTPClient();
get(url)	N/A	Gets an URL.	http.get("http://www.cisco.com");

- PT7 → Help → Contents

# Real data source



- Webserver runs on Raspberry Pi
- Arduino gets room temperature with TMP36 sensor

```
Start Arduino on port "/dev/ttyACM0"  
Web Server GET route "/temp" return text temp  
  set volts to [Analog read pin 0] x 5 ÷ 1024  
  set temp to [volts] - 0.5 x 100  
  Print on screen temp  
Run webserver
```

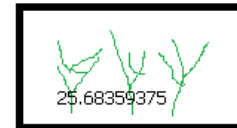
# Coding for real sensor

- Find the line where you read temperature value from PT simulated environment and replace it with call to the real HTTP server:

```
//var temp = Environment.get("Ambient Temperature");  
RealHTTPClient.get("http://192.168.1.101:5000/temp", function(status, data) {temp = data;});
```

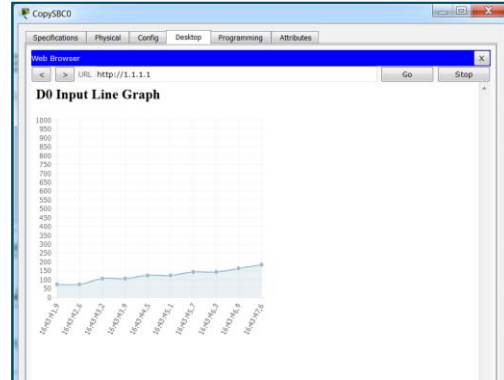
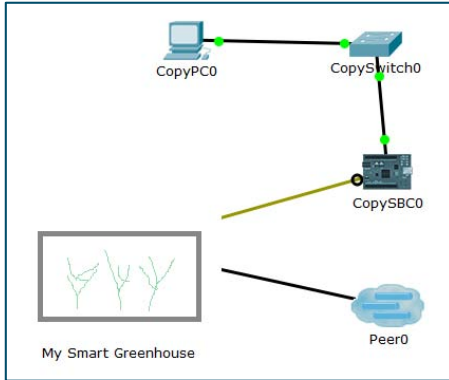
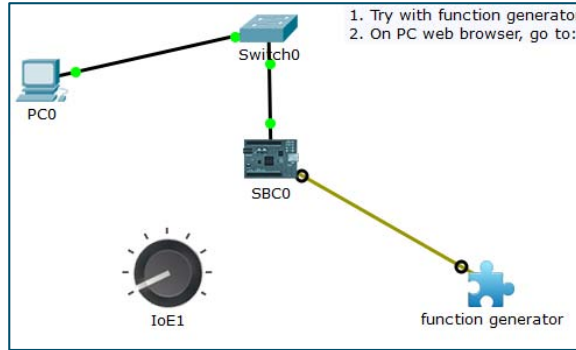
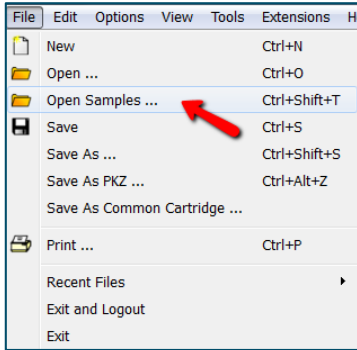
- Use IP address 192.168.1.101 and port 5000
- Your device now should act based on the real room temperature**
- Optionally, your device can directly display the current temperature:

```
setCustomText(50, 130, 100, 20, temp);
```



# Data visualization challenge

# Visualize your data using SBC



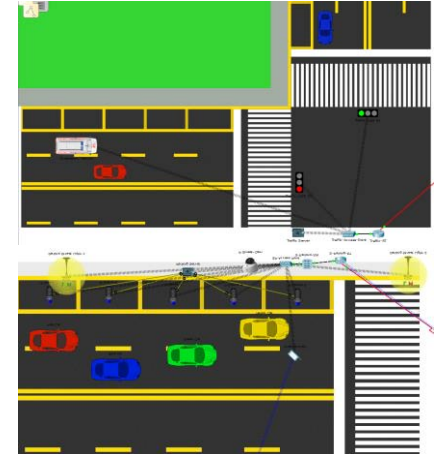
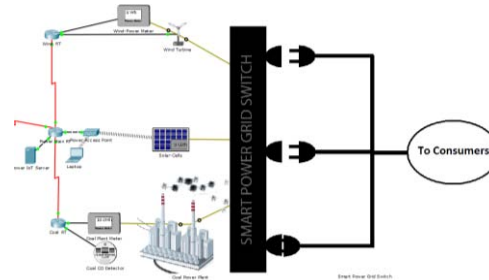
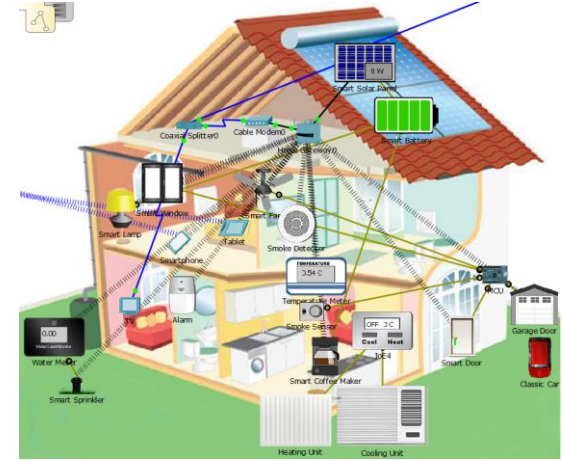
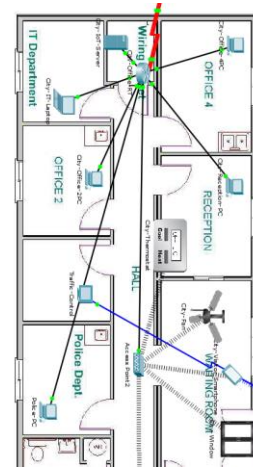
- Open Graphing sample from “7.0 – ioe – graphing.pkt”
- Explore the code on SBC0 and function generator
- Notice function generator sends data to SBC over digital connection
- Copy SBC to your PKT file
- Modify your device I/O by adding another digital slot to connect to SBC
- Modify your device code to send data to SBC for visualization



# Summary

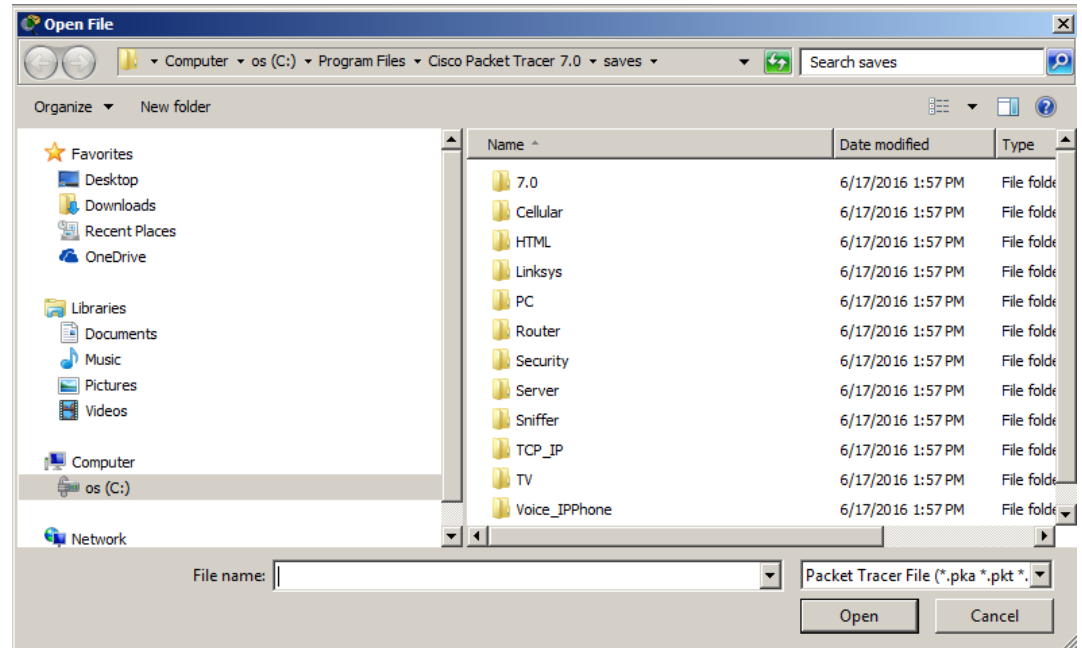
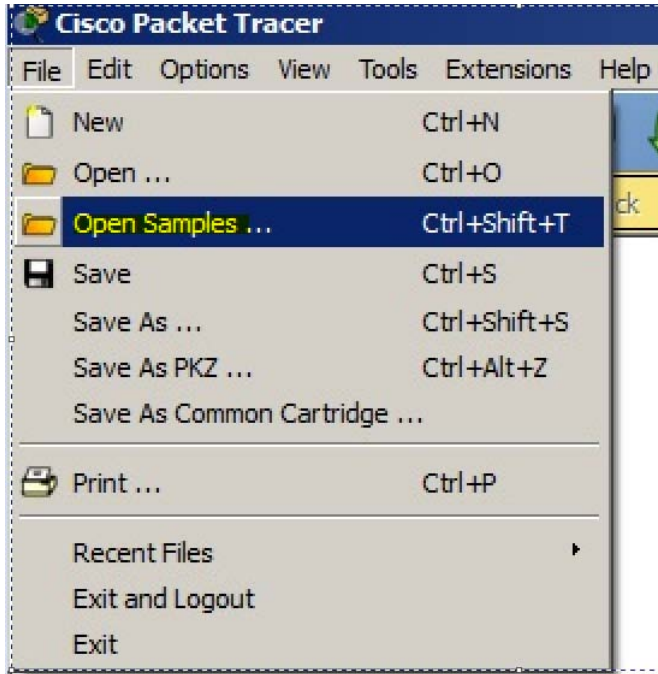
# IoT Highlights 7.0

- Physical Environments
- Smart devices, sensors and actuators
- Smart Home, Smart City, Industrial, Power Grid
- Edit existing or program your own devices
- Python, Javascript, Blockly
- SBC and MCU
- Home Gateway and Reg Server
- Rules for devices to work together
- Routers 819 and 829



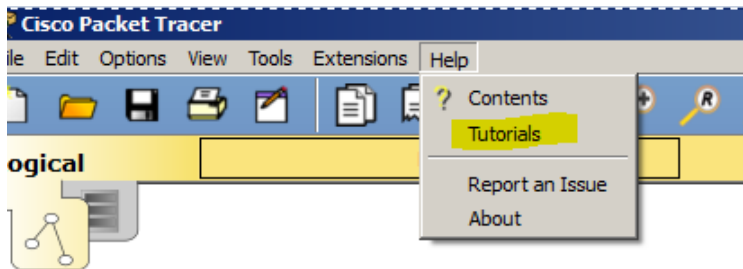
# Sample Files

To find the Sample files



# Tutorials

Packet Tracer includes a lot of tutorials to help the user to understand new features



## 14 New 7.0 Features

### 14-1 Whats New in Packet Tracer 7.0

- Introduce the latest capabilities of Packet Tracer 7.0.

### 14-2 Actuators, Sensors and Smart Devices

- Explore the new IoT devices.

### 14-3 Smart Home

- How to create a Smart Home with the new IoT features.

### 14-4 New Network Devices

- Explore the new devices and protocols in 7.0.

### 14-5 Home Gateway and Registration Server

- Connect your IoT device to a server for remote control.

### 14-6 Physical Workspace

- Learn how to use containers, bendpoints, and other physical workspace capabilities.

### 14-7 Web Server

- See what you can do with the HTTP server now.

### 14-8 Thing Editor and Device Manager

- Learn how to create your own custom IoT device.

### 14-9 Email Client and MCU

- Learn how to use the email client with the MCU.

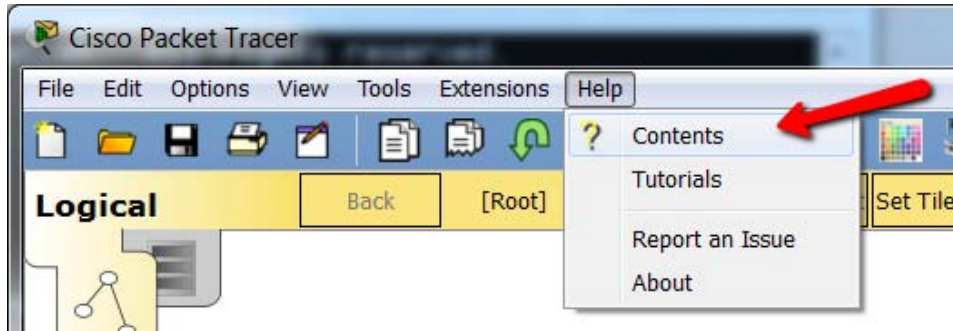
### 14-10 Environment

- See how the environment affects the sensors.

### 14-11 Environment Key Frames

- Control the environment variables to create a dynamic world.

# Help Contents



## JavaScript API

Program Structure and Events			
Function	Return Type	Description	Example
setup()	N/A	If defined, this function is called once when the program starts.	<pre>function setup() {   pinMode(0, INPUT); }</pre>
loop()	N/A	If defined, this function is called continuously when the program is running. The frequency of the calls depends on the complexity of this function, the number of other devices running programs and their complexity, and the machine's processing power.	<pre>function loop() {   Serial.println(digitalRead(0)); }</pre>
cleanUp()	N/A	If defined, this function is called once just before the program stops.	<pre>function cleanUp() {   Serial.println("program is stopping."); }</pre>
mouseEvent(pressed, x, y)	N/A	If defined, this function is called when the user clicks and/or moves the mouse on the workspace icon of this device.	<pre>function mouseEvent(pressed, x, y, firstPress) {   if (firstPress)     doSomething(); }</pre>



## Multuser IPC

### Internet of Things

- Using Things
- Creating Things
- JavaScript API
- Python API
- Visual API

### Environment

### Script Modules

- Scripting Interface
- Script Engine
- Web Views
- Data Store
- Data Store Editor
- Custom UDP Processes
- Tips

### Sample Files &

### Design Patterns

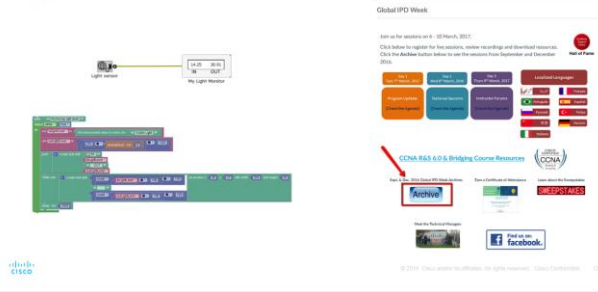
- Concept Builders
- Skill Builders
- Design Challenges
- Troubleshooting

Using t  
The help f  
order (con

# PT7 in Global IPD Week

## PT 7.0 and the Real World March GIPD Week

### Programming Smart Devices in PT 7.0 Part I September GIPD Week



Global IPD Week

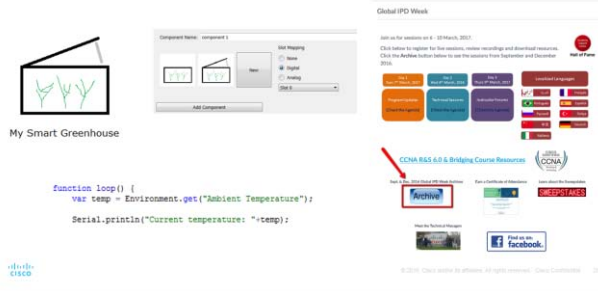
Join us for sessions on 6 - 10 March, 2017.  
Click below to register for live sessions, review recordings and download resources.  
Click the Archive button below to see the sessions from September and December 2016.

CCNA R&S 6.0 & Bridging Course Resources

Archive

Facebook

### Programming Smart Devices in PT 7.0 Part II December GIPD Week



Global IPD Week

Join us for sessions on 6 - 10 March, 2017.  
Click below to register for live sessions, review recordings and download resources.  
Click the Archive button below to see the sessions from September and December 2016.

CCNA R&S 6.0 & Bridging Course Resources

Archive

Facebook

#### Use online data

- 1 Take an existing device
- 2 Read weather data from an online server
- 3 Parse the data and display

#### Use room environment

- 1 A real temperature measuring device
- 2 Web-server to report the data
- 3 Simulated device to read the data

#### Output to the real world

- 1 Use existing simulated motion detector
- 2 Send current state to web server
- 3 Device to turn on a signal light

#### Global IPD Week

Join us for sessions on 6 - 10 March, 2017.

Click below to register for live sessions, review recordings and download resources.  
Click the Archive button below to see the sessions from September and December 2016.



Day 1  
Tues 7<sup>th</sup> March, 2017

Day 2  
Wed 8<sup>th</sup> March, 2017

Day 3  
Thurs 9<sup>th</sup> March, 2017

Localized Languages

- Arabic
- French
- Portuguese
- Spanish
- Pytsvul
- Turkçe
- 中文
- Deutsch
- Italiano

Program Updates [Check the Agenda]

Technical Sessions [Check the Agenda]

Instructor Forums [Check the Agenda]

[CCNA R&S 6.0 & Bridging Course Resources](#)



6<sup>th</sup> & Dec. 2016 Global IPD Week Archives



Earn a Certificate of Attendance



Learn about the Sweepstakes

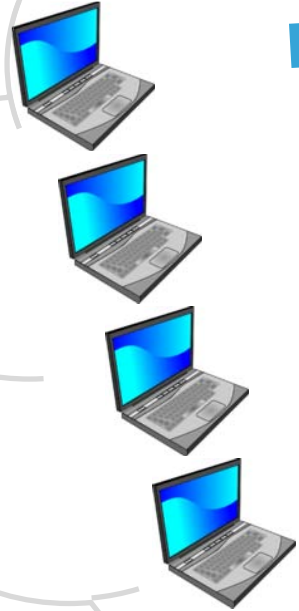


**Recordings Available**  
<http://cs.co/GIPD17>





# Network setup



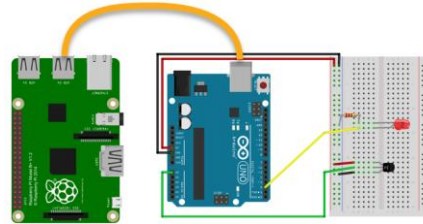
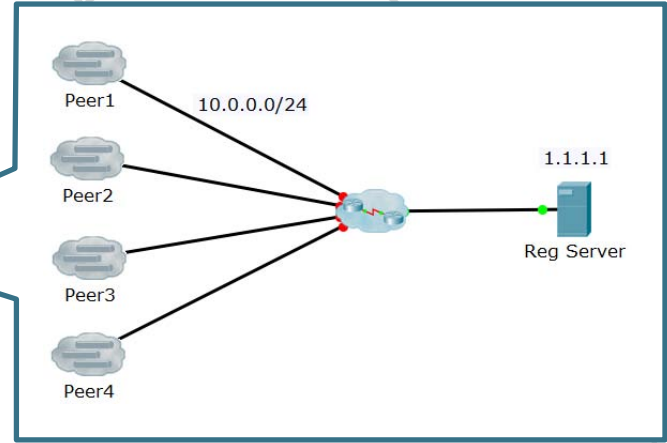
192.168.1.XX



SSID: netacad  
password: workshop



192.168.1.100



192.168.1.101