HOME DYNAMIC
AUTOMATION SYSTEM

Home Dynamic System Design Reference Documentation

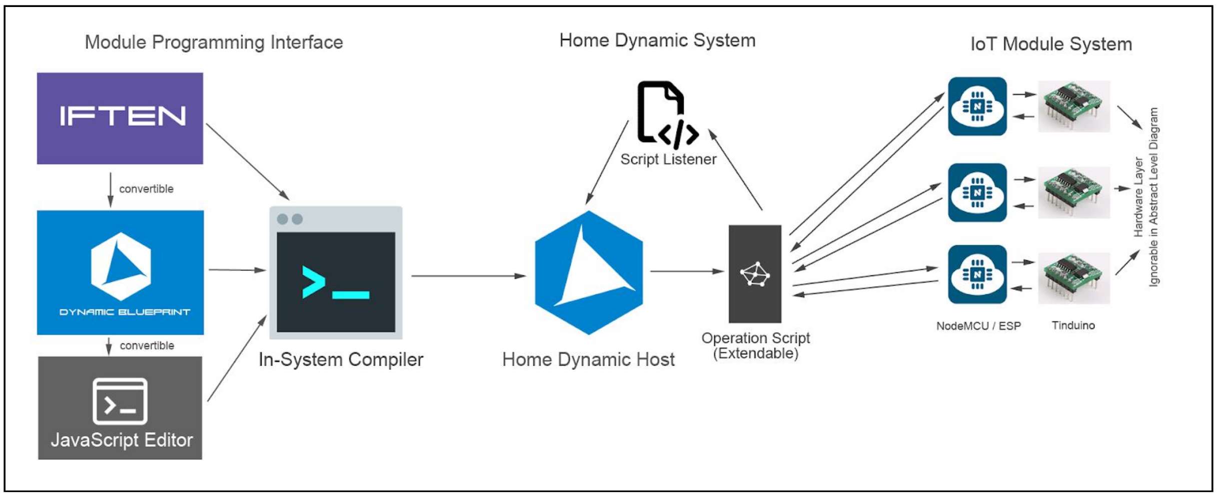
Created for Home Dynamic Home Automation System Prototype, 2017-2018

Introduction to Home Dynamic System

Home Dynamic is a Raspberry Pi Zero W based low cost and easy to setup Home Automation System (or IoT System) design for compact space operation like Hong Kong or other high density cities like London, Tokyo etc.

System Features

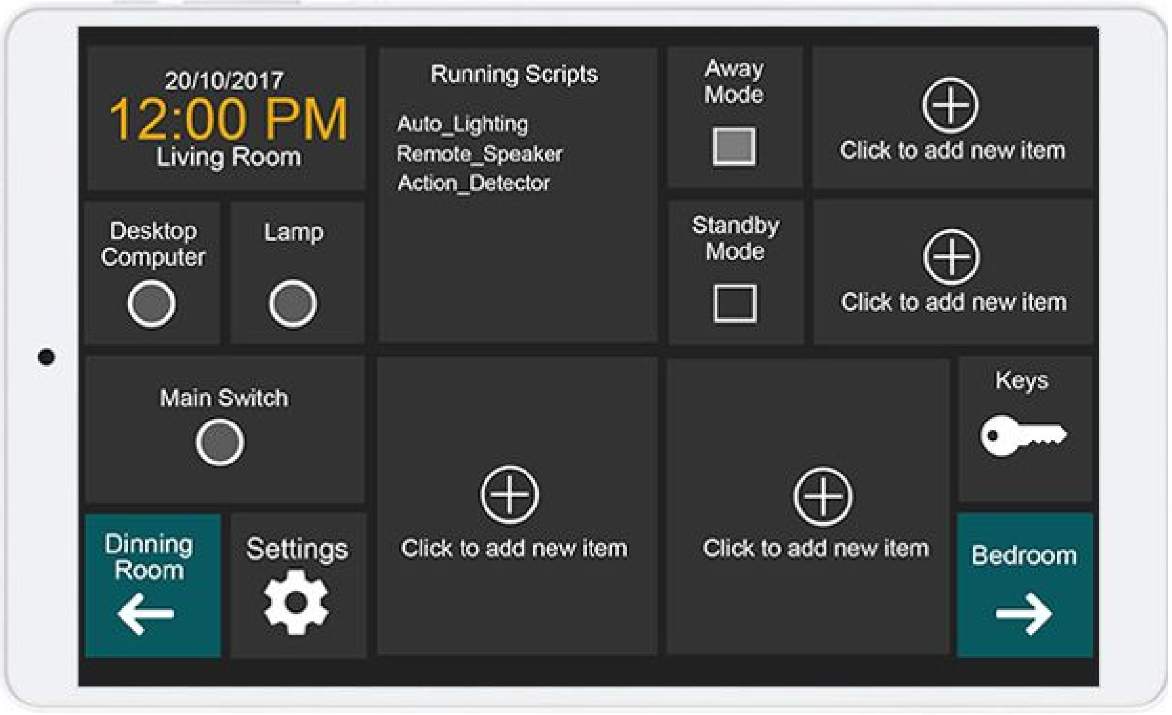
Home Dynamic System features three Module Programming Interface (MPI), compensating the lack of programming interface for current Home Automation System like the OpenHAB and Home Assistant. Home Dynamic will use the **IF-THEN-ELSE-NEXT** (IFTEN), **Dynamic Blueprint™** and standard **JavaScript Programming Interface** for beginners, advanced users and expert. The code generated by the 3 different systems can be converted to different editing mode and exchange between different users.



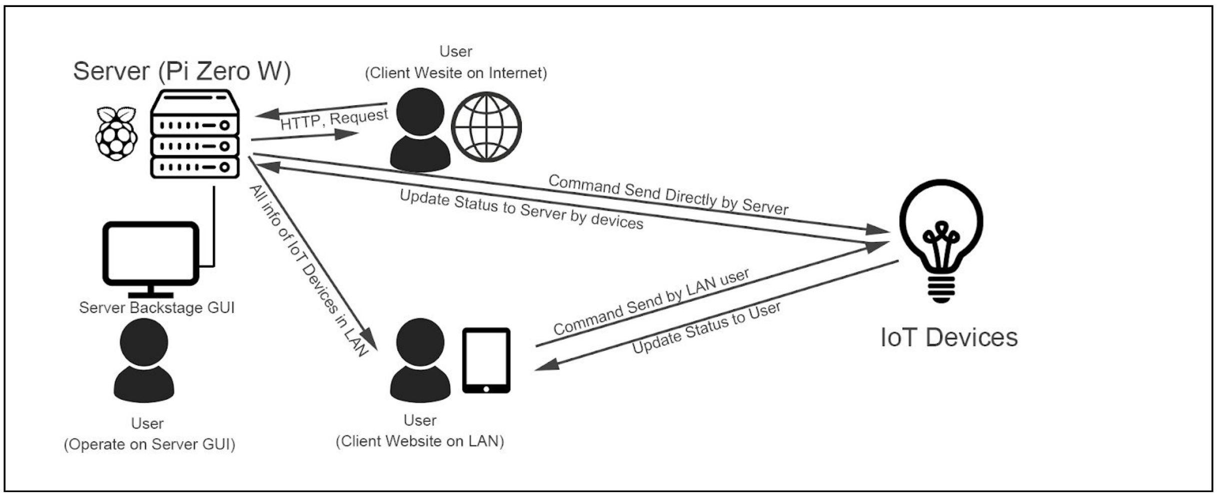
Another feature for Home Dynamic System will be its low cost and easy to set up modules. For each IoT module, we are expecting to sell at **$10USD per unit**. For example, if the user request for a IoT controlled Air Conditioner, assume the user already have his Home Dynamic system hosted and running, the user will need to purchase a Main Power Control Block (For on/off, NodeMCU inclusive) , IR Transceiver (For temperature control, Tinduino inclusive). In total $20USD. The modules can be linked in series, as well as power supply system and access port. The detail will be mentioned in the hardware specification section.

Main Control Interface Structure

The main interface reference the OpenHAB's interface design, but with much more easy to control and similar interface and reduced the number of information displayed on the interface. Material designs are used to design the interface of the Home Dynamic main user interface and the interface can be easily updated using build in functions.
The Home Dynamic Main Control Interface can be opened with browser on all kind of devices support ES6 JavaScript. Including Chrome, Firefox and Edge.



Multi-user control are performed using Client-To-Module linkage instead of central server operation like the other open source Home Automation System did. The structure of control system via interface is represented in the graph below.



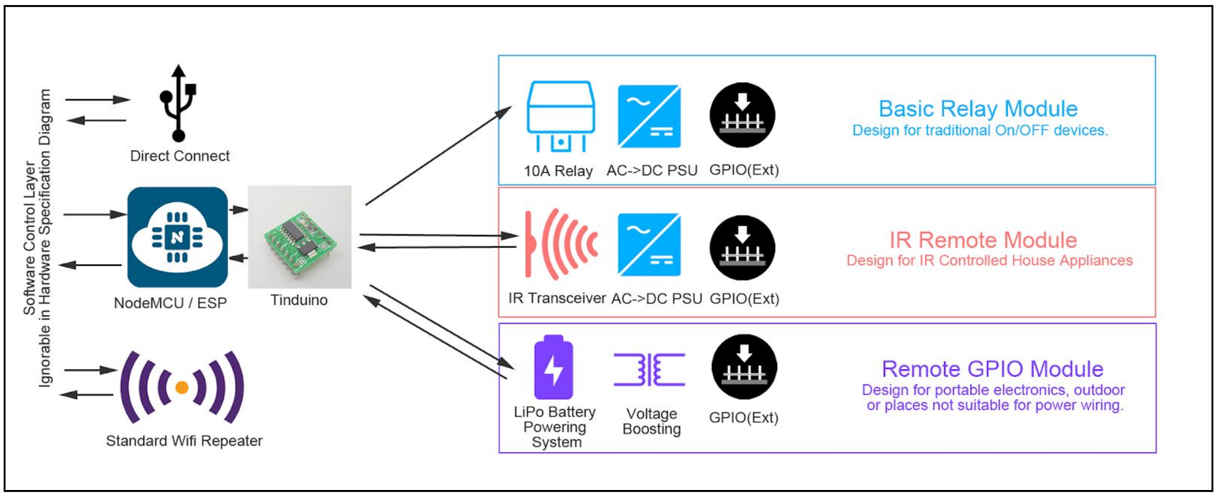
With this design, even if the server's processing power is low, it can still support a number of users from LAN connection as the more client is connected to the system, there are more devices to share the processing load in controlling all devices.

Standard Hardware Specification

The standard module of Home Dynamic will be made up with the following type of devices or repeater.

1. Basic Relay Module
2. Basic IR Module
3. Remote GPIO Module
4. Signal Repeater (Standard Wifi Repeater)

The structure of Standard Hardware Modules are illustrated in the following diagram.



Software Control Layer
Ignorable in Hardware Specification Diagram
Direct Connect
NodeMCU / ESP
Tinduino
Standard Wifi Repeater
10A Relay
AC->DC PSU
GPIO(Ext)
Basic Relay Module
Design for traditional On/OFF devices.
IR Transceiver
AC->DC PSU
GPIO(Ext)
IR Remote Module
Design for IR Controlled House Appliances
LiPo Battery Powering System
Voltage Boosting
GPIO(Ext)
Remote GPIO Module
Design for portable electronics, outdoor or places not suitable for power wiring.

Master - Assistant Microprocessor Architecture

The hardware specification of the ESP8266 and Tinduino are as following.

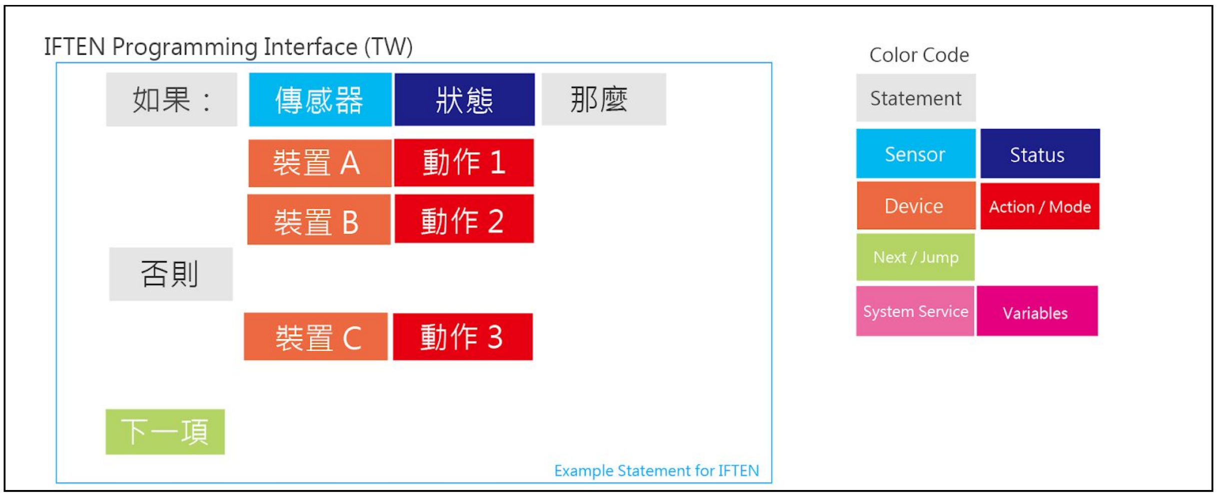
NodeMCU v1.0+ / ESP8266	**Tinduino v3.5+**
Manufacturer Espressif Systems	Manufacturer IMUS Laboratory
Type Microcontroller	Type Microcontroller
CPU @ 80 MHz (default) or 160 MHz Memory	CPU AVR ATtiny44A @ 8Mhz (MIPS)
64 KiB instruction, 96 KiB data	Memory 4KiB instruction, 256 Byte EEPROM
Input 16 GPIO pins	Input 11 GPIO pins
Power 5VDC / 3.3 VDC	Power 5 VDC

The ESP8266 in this system is used to provide HTML interface via HTTP. The command will be sent to Tinduino for further processing. This is designed to be freeze proof and in case the system has no responses, the NodeMCU / ESP8266 can receive command from Home Dynamic Host to terminate any program running on Tinduino. Tinduino can also be an additional processing core for NodeMCU / ESP8266 while computing complicated algorithm or data phrasing. Another benefit for using dual microprocessor will be performing activating mechanism for NodeMCU / ESP Wifi services while Wifi services are in sleep mode under battery mode. This structure is similar to Qualcomm “Big-little technology”.

IF-Then-Else-Next Programming Interface (IFTEN)

The IFTEN interface are designed to be used by non professional users. This programming interface provide drag-drop menu and simple to use selection choices for users to choose from and control their IoT devices connected to the Host. IFTEN language can be later translated into Dynamic Blueprint Programming System or pure JavaScript and open in the system build in JavaScript Editor.

An IFTEN Statement example as follow.



An IFTEN Statement can be extend. However, one If-Then-Else Statement cannot contain another If-Then-Else Statement. This is for the ease of programming for beginners and the system should not be as complicated as real programming languages.
The available command are grabbed from the system host. And the system host will grab the necessary functions from the IoT Module (i.e. NodeMCU) and return all the possible command usable. It is basically a list of usable GPIO and Control Mode for standard modules and custom mode for non-standard modules.

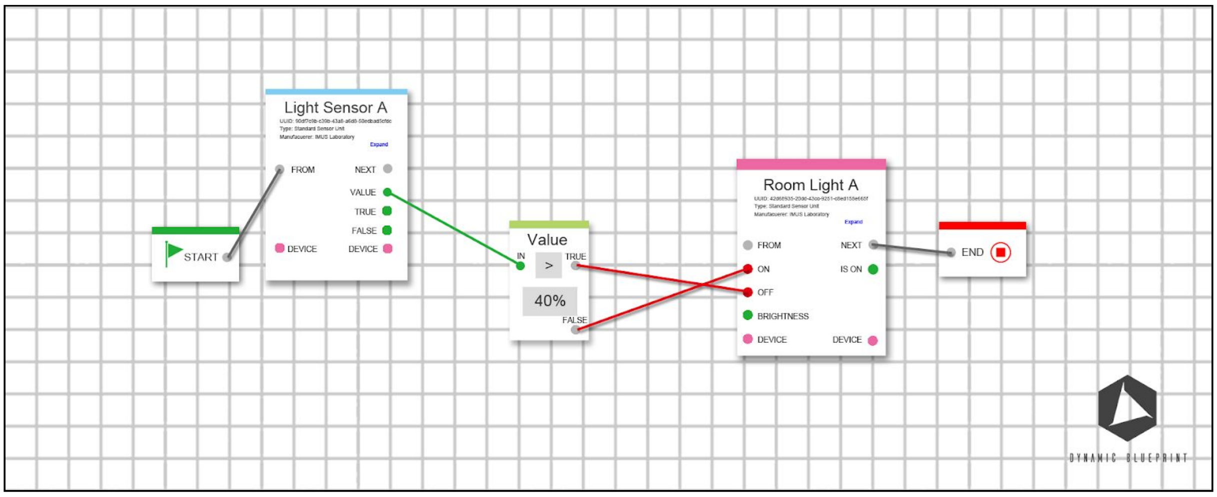
Here are some examples for IFTEN Statements, Next/Jump Instructions and System Services Instructions.

IFTEN Statements	Next / Jump	System Service

IF
THEN
ELSE
AND
OR
NOT
	NEXT
	LABEL(id)
	JUMP(Label, Dropdown Menu for selection)
	END
		ONLINE(Module uuid)
		OFFLINE(Module uuid)
		KILL(Module uuid)
		RUNNING(Module uuid)
		WAIT_FINISH(Module uuid)
		TIME_OUT(Module uuid, time in ms)
		NOTIFY(Text)

Dynamic Blueprint System (Blueprint)

Dynamic Blueprint System is a programming interface for Home Dynamic Project. The Dynamic Blueprint is for advance system users in which they have the basic logic in controlling numbers of IoT devices and control what they need to be automated. Compare to IFTEN, Blueprint gives much more flexibility in creating complicated home management scripts. The script can be easily changed by drag drop and line drawing. One basic statement of the Blueprint system is as follow.



The above statement is equivalent to an IFTEN statement of:

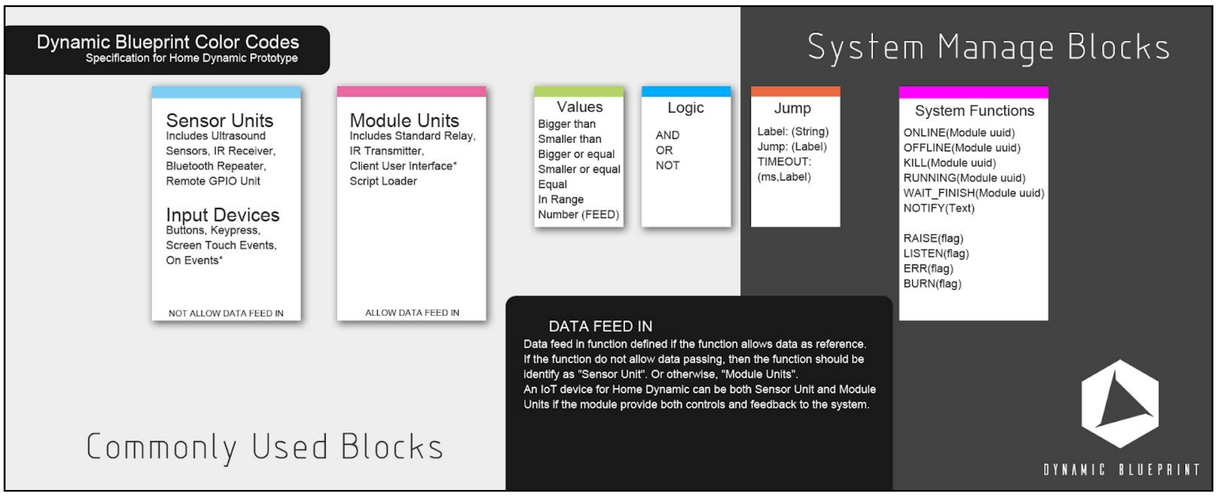
```
IF Light_Sensor_A(value) > 40% THEN Room_Light_A(mode) ON ELSE
Room_Light_A(mode) OFF NEXT
```

However, as IFTEN do not support comparative statements, this cannot be replicated in IFTEN system. Instead, this will become a less sensitive statement as follow:

```
IF Light_Sensor_A=true THEN Room_Light_A ON ELSE Room_Light_A OFF NEXT
```

(p.s. Default true implies the sensor value is greater than the default value plus offsets. It was set by the hardware manufacturer.)

The color code for different function blocks are illustrated in the diagram below.



DATA FEED IN

Data feed in function defined if the function allows data as reference. If the function do not allow data passing, then the function should be identify as "Sensor Unit". Or otherwise, "Module Units". An IoT device for Home Dynamic can be both Sensor Unit and Module Units if the module provide both controls and feedback to the system.

Home Dynamic Transfer Protocol

The Home Dynamic Transfer Protocol (HDTP) is the main data pathway within the Home Dynamic System. The datapath mainly consists of two parts.

1. Home Dynamic Host ⇋ Client User Interface (HC Pathway)
2. Home Dynamic Host ⇋ IoT Modules (HI Pathway)

The specification of **HC Pathway** is json based and **HI Pathway** are plain text over HTTP. The detail of the protocol is specified as follow.

HI Pathway

The HI Pathway divided into two parts. The IoT Module and the Host Caching System. Here is a simple demonstration on the module creating Module Functions Cache in the Host Caching System and rename the module to a new name.

IoT→Host are in blue and Host→IoT is in red, requests are in green.

```
<Module setup finished, first connection to Host>
GET: Module_setup.php (module uuid, module ip, module port, module name)
Request: module_ip:port/info
Plain Text:
<Module-Name>_<Module-uuid>_funct:<funct_Path>_<item>:<values>_<item>:<values>
Request: module_ip:port/<funct_path>
Plain Text: <Module-uuid>_<JavaScript Functions Code>
Request: module_ip:port/rename
GET: <Module-uuid>_<new-name>
Plain Text: <Module-uuid>_DONE
<Repeat Request of /info to get the new devices name and update Host Function Cache>
```

The HI Pathway should support the following standard commands and have followed the following protocol definitions.

Standard Request Path	Supported Response Type
/info /funct/ <function name to call> /uuid /kill /running	State:<key1>,<value1>;<key2>,<value2> Mode:<path>_Option:<option1>,<option2> Variables:<path>_Keywords:<keyword1>,<keyword2>*

*Return in GET variables. E.g. /path/keyword1=text1&keyword2=text2

Module uuid and ip address will be passed to the Home Dynamic Host while initiate or by pressing the reconnect button on the module. The module will make a request for IP update.

```
http://home_dynamic:port/subpath/setIP.php?uuid=<module_uuid>&ip=<module_ipv4>
>DONE
```

HC Pathway

The HC Pathway consists of two main sub-system. Including the general json format for programming interfaces IFTEN and Dynamic Blueprint System, and User Interface Update Protocol.

Home Dynamic Server → Client Programming Interface

The Server to Client Programming Interface relies on the module uuid and gtid. UUID is an unique id for each IoT Modules under the Home Dynamic system and the GTID is the general type id for a group of specified Home Automation Module that use the same firmware and communication methods. Here is an example for the Standard Relay Module information supplied with the uuid page.

```
UUID=ce987b33-3584-4add-80f2-670441822200
GTID=com.imuslab.home_dynamic.standard.relay
```

Extra information can be given in the uuid page including manufacturing date, batch number and simple description on module operation method and contact info etc. A full uuid page example is as follow.

```
UUID=ce987b33-3584-4add-80f2-670441822200
GTID=com.imuslab.home_dynamic.standard.relay
MANUFACTURE_DATE=23/11/2017
DESCRIPTION=The IMUS Standard Relay Module for automated switches!
CONTACT=contact.imuslab.com
```

Blue items are necessary. Purple items are optional information.

And the above information will be used to download an additional javascript library containing all the required functions and protocol for activating the module. Or the user can choose to upload from their file system. The json phrased as follow.

```
{
  "name": "Standard Relay Module",
  "manufacturer": "IMUS Laboratory",
  "uuid": "ce987b33-3584-4add-80f2-670441822200",
  "gtid": "com.imuslab.home_dynamic.standard.relay",
  "IOs": [
          [
          "Toggle",
          "This function is set the mode of the Relay Module, input \"ON\" to turn on and
\"OFF\" to turn off.",
          "select",
          "string"
          ],
          [
          "CheckOnline",
          "This function is to check if the Relay Online",
          "return",
          "boolean"
          ],
          [
          "GetMode",
          "return",
          "boolean"
          ]
  ],
  "functions": [
          "function Toggle(status){\nif (hd.module.connected(uuid)==true){\n//This module is
connected\nswitch(status){\ncase \"ON\": hd.send(uuid,true);break;\ncase \"OFF\":
hd.send(uuid,false);break;\nreturn true;\n}\n",
          "function CheckOnline(){\nif (hd.module.connected(this.uuid) == true){\n//The
module is currently online\nreturn true;\n}else{\nreturn false;\n}",
          "function GetMode(){\nif (hd.module.connected(this.uuid) == true){\n//The module
is currently online\nif (hd.get(uuid,\"/mode\").includes(\"ON\")){\nreturn
\"ON\";\n}else{\nreturn \"OFF\";\n}\n}"
  ]
}
```

**The color code above illustrate how the three function was created and acknowledge the programming interface on how to use the functions.

Then after the programming interface, a json file will be composed and generated. The json file then will get feed into a compiler and the javascript will be stored as a script file on server side. The storage position is depend on the starting block (ON, AT, or START).

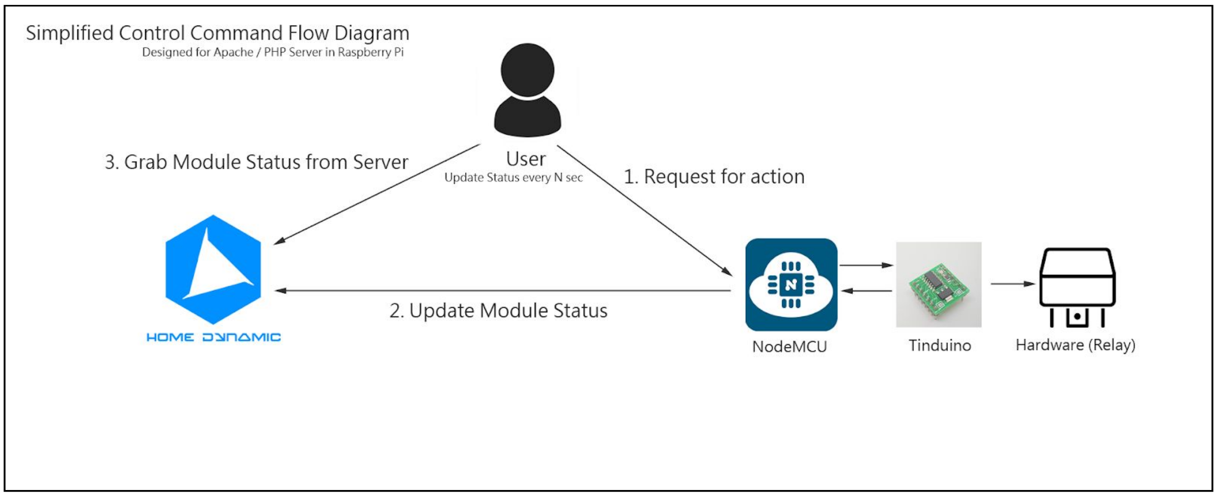
Starting Block Specification

The starting block will define how the program run. There are three type of START event in the Home Dynamic System.

ON events	ON an sensor / detector get triggered. Passive script starting mode.
AT events	AT get trigger at a specific period of time. Active script starting mode.
START events	START get triggered when user press a button. Passive script starting mode.

Home Dynamic Server → User Interface / Interface Update Protocol

The User Interface Update Protocol specify the command needed to update a certain module on the control interface. For example, the switch of an IoT device should goes green after switching on and red after switching off. The status is being stored in server side as a config file and communication are done using http request. Alternatively, the active response system can be done using websocketd and php script on the server side.
(Please refer websocketd on http://websocketd.com/)


Simplified Control Command Flow Diagram
Designed for Apache / PHP Server in Raspberry Pi
User
Update Status every N sec
3. Grab Module Status from Server
1. Request for action
2. Update Module Status
HOME DYNAMIC
NodeMCU
Tinduino
Hardware (Relay)

Additional plugins can be installed to use Node.js with Websocket for update functions. This will be done after the main system finished and made as an external plugin for alternative experimental projects that is designed for the Home Dynamic System.

The json format grabbed by the client is a custom formatted json file with the following format.

```
{
  "uuid": [
    "Module Name",
    [
      "Property Name",
      "Type",
      "Data"
    ]
  ],
  "6549eb01-8c88-41d0-b3b0-0c942054afe0": [
    "Demo Room Desk Lamp",
    [
      "Status",
      "boolean",
      "true"
    ],
    [
      "Brightness",
      "float",
      "0.5"
    ]
  ]
}
```

The information is then phrased to an interface and show all the status information of the system. The modules can be grouped on the web interface. While requesting for the status, the group name can be passed as a variable and the required module information will be packed into a json file for client processing. The following are the examples for group sorting in json phrasing.

```
getStatus.php?group=my%20room
getStatus.php?uuid=6549eb01-8c88-41d0-b3b0-0c942054afe0
getStatus.php?uuid=6549eb01-8c88-41d0-b3b0-0c942054afe0,ce50caee-44ea-4cf8-9a87
-2aed6c5f06b8,a19bc874-b743-4fa7-b13b-d81b262d9e4a
getStatus.php?search=Lamp
```

If the user want to activate an appliances, or module, given the uuid, the client will send an AJAX request to the Home Dynamic Host for advance information on a uuid.

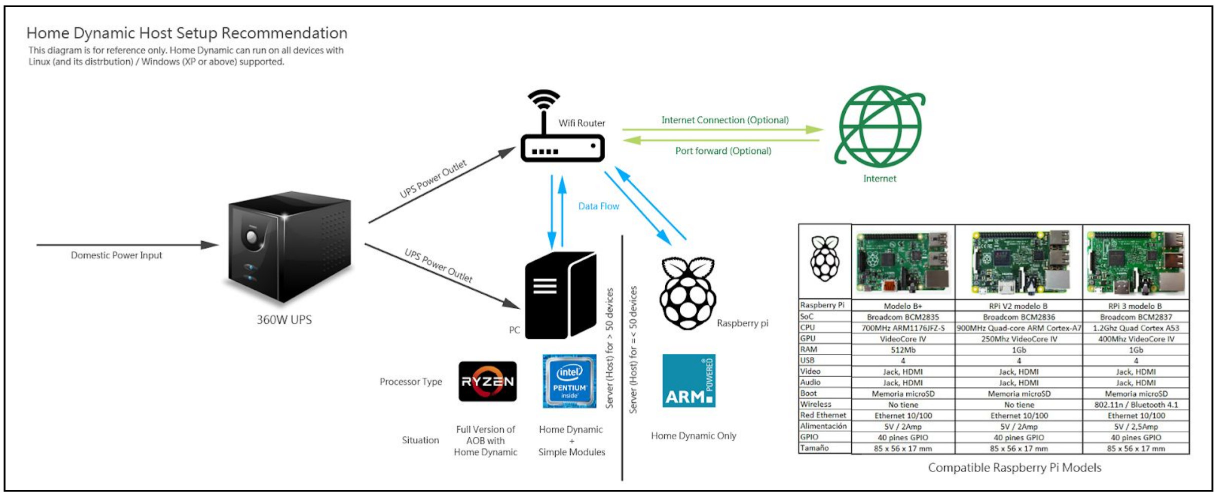
```
getDetail.php?uuid=6549eb01-8c88-41d0-b3b0-0c942054afe0
```

Then the php will return the module’s detail in json format, specifying all the command that is usable on this IoT module. For example, the Desk Lamp will turn on when request 192.168.0.123/ON and turn off when request 192.168.0.123/OFF with basic auth of “username” and “password”, then the json passed from the PHP script to client interface will be as follow.

```
{
  "6549eb01-8c88-41d0-b3b0-0c942054afe0": [
    "Demo Room Desk Lamp",
    "192.168.0.123",
    "username",
    "password",
    [
      "ON",
      "This command turn on the Desk Lamp",
      "/ON"
    ],
    [
      "OFF",
      "This command turn off the Desk Lamp",
      "/OFF"
    ]
  ]
}
```

Home Dynamic Host Specification

The Home Dynamic Host System is a server that provide centralized processing of all the modules. The stability of the server (host) is extremely important. Hence, it will be suggested that the server be setted up as follow.


Home Dynamic Host Setup Recommendation
This diagram is for reference only. Home Dynamic can run on all devices with Linux (and its distrbution) / Windows (XP or above) supported.
Wifi Router
Internet Connection (Optional)
Port forward (Optional)
Internet
UPS Power Outlet
Data Flow
Domestic Power Input
UPS Power Outlet
360W UPS
PC
Server (Host) for > 50 devices
Server (Host) for =< 50 devices
Raspberry pi
Processor Type
RYZEN
intel PENTIUM inside
ARM POWERED
Situation
Full Version of AOB with Home Dynamic
Home Dynamic + Simple Modules
Home Dynamic Only

Raspberry Pi	Modelo B+	RPi V2 modelo B	RPi 3 modelo B
SoC	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2837
CPU	700MHz ARM1176JFZ-S	900MHz Quad-core ARM Cortex-A7	1.2Ghz Quad Cortex A53
GPU	VideoCore IV	250Mhz VideoCore IV	400Mhz VideoCore IV
RAM	512Mb	1Gb	1Gb
USB	4	4	4
Video	Jack, HDMI	Jack, HDMI	Jack, HDMI
Audio	Jack, HDMI	Jack, HDMI	Jack, HDMI
Boot	Memoria microSD	Memoria microSD	Memoria microSD
Wireless	No tiene	No tiene	802.11n / Bluetooth 4.1
Red Ethernet	Ethernet 10/100	Ethernet 10/100	Ethernet 10/100
Alimentación	5V / 2Amp	5V / 2Amp	5V / 2,5Amp
GPIO	40 pines GPIO	40 pines GPIO	40 pines GPIO
Tamaño	85 x 56 x 17 mm	85 x 56 x 17 mm	85 x 56 x 17 mm

Compatible Raspberry Pi Models

Home Dynamic System can work with ArOZ Online BETA to provide a fully personal cloud experience for the user providing with Video Streaming, Music Streaming, Photo Station, Memo, Notification, Manga Reading and many alternative functions. In the case of the user want to install Home Dynamic System with ArOZ Online BETA, there are two choices regarding the installation method.

1. Home Dynamic System installs on a separate server with the ArOZ Online BETA. For example, HDS is installed on 192.168.0.2 and AOB is installed at 192.168.0.3
2. Home Dynamic System is installed as a module of ArOZ Online BETA system. (This installation method might cause some unexpected error or function lossing)