

BACH KHOA UNIVERSITY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

AUTOMATIC WATERING

[IoT group's project]

Group members:

Nguyen Quang Cuong - 1410420

Nguyen Thanh Van - 1450395

Dinh Gia Bao - 1450010

Nguyen Hien Quang - 1450124

Mai Hoang Phi - 1552276

Tran Duc Trung - 1552402

Table of Contents

I.	INTRODUCTION.....	2
1.	INTRODUCTION OF GROUP IDEA.....	2
2.	REASON	3
a.	Hardware	3
b.	Software.....	3
II.	DESIGN	4
III.	IMPLEMENTATION	5
1.	Hardware	5
2.	Server	5
3.	UI.....	13
IV.	DEMO	26
1.	Hardware	26
2.	Server	27
3.	Website UI.....	28

I. INTRODUCTION

1. INTRODUCTION OF GROUP IDEA

Viet Nam is one of country which is developed base on agriculture, such as: export café and rice. For the convenience and best result at the end of season, watering and measurement in agriculture are the most important aspect. Due to the demand and increase the efficiency of the product, our group decided to apply our class subject-IoT –to help farmers have best qualified products.



IoTs – Internet of Things- is a system which connect hardware and software to improve human life. Not the connection only between hardware and software, IoTs help people easy to use and convenience in create or upgrade their product with low cost and high quality.

Our project's name is Automatic Watering. This project is based on Android Things to measure the temperature and humidity, which is collected from the garden outside. These data will be pushed into the Server which is created by our group for storing and display in the UI.



2. REASON

The project components:

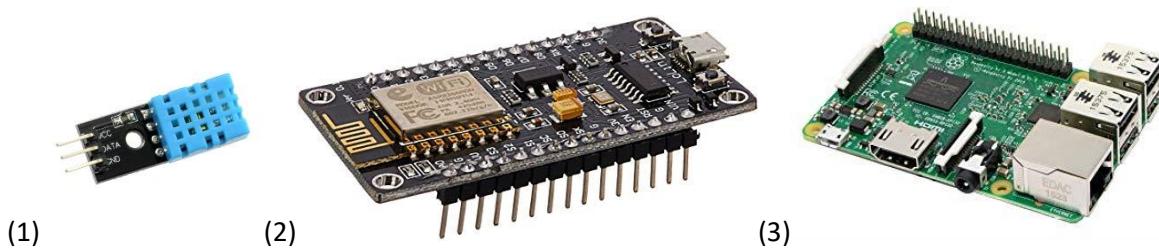
a. Hardware

DHT-11: Collect humidity and temperature in the garden. (1)

Node MCU esp8266: Connect to Wi-Fi and transmit data, which is collected from sensor DHT-11, to Server for UI display. (2)

Raspberry pi 3 model B: Control the sensor condition and number of node connect to the system. If the data value is higher than the condition, led will blink. (3)

LED: Blink when the data value is higher than the condition.



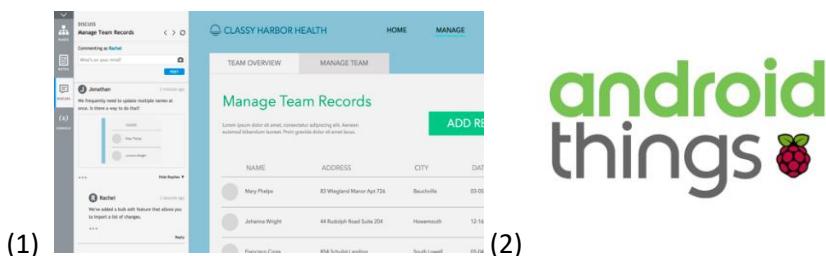
b. Software

OS: Android Things, easy for user to use and we are learning in class. This OS is developed by Google and has same function as android mobile.

Server:

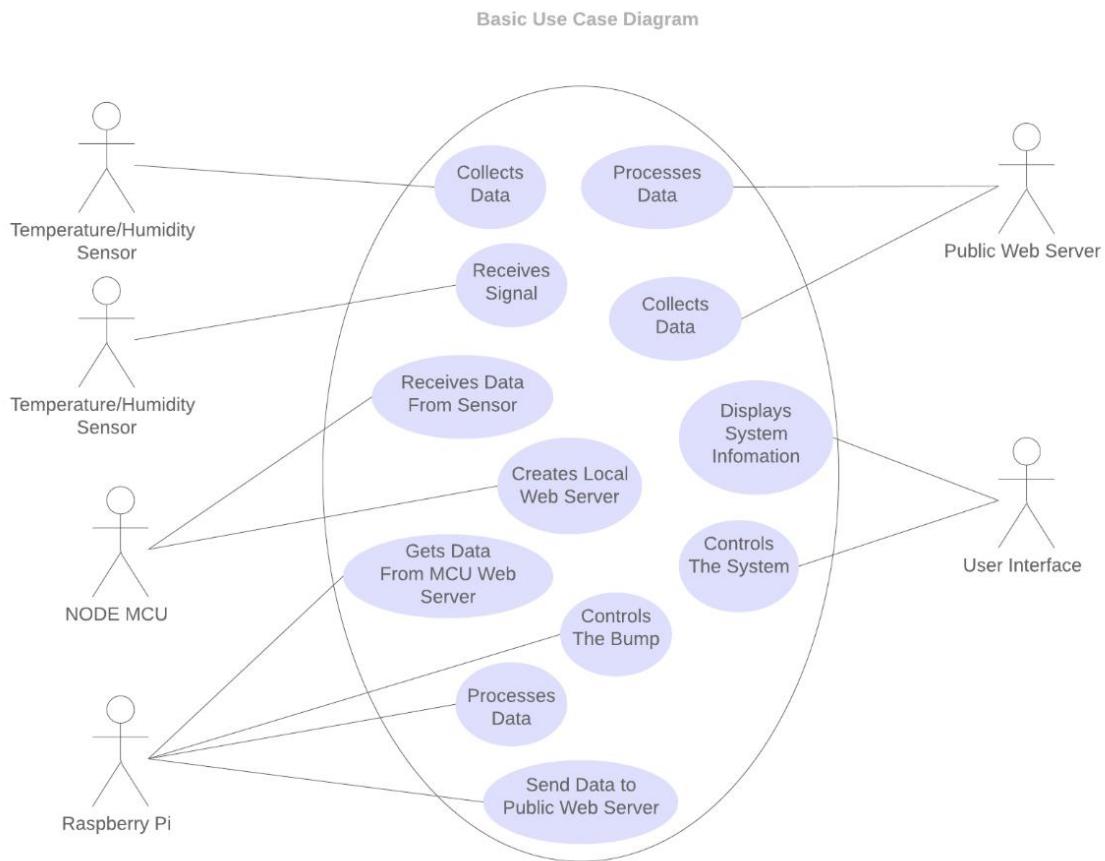
- Language of back-end: Javascript (Nodejs platform)
- Database: MongoDB (Mongoose integrated)
- Server framework: Expressjs
- Cloud platform server: Heroku
- Language in front-end for getting data: php, Javascript,...

Website: Axure RP 8 is one of most famous application for designing website UI. It is easy to share, diagramming, documentation and easier team collaboration.



II. DESIGN

Use case diagram



- Sensors have to collect data from the environment then push it to node MCU
- Node MCU collect data from sensors, then create a local server for transferring data to the gateway.
- Raspberry Pi gets data from node MCU server. After that, the data is formatted and push to the Web server. If the data is higher than the condition, which is set on the server by user, Raspberry Pi will control the bump to bump water if the plan needs water.
- Public web server will receive data, which is sent by the Pi, then display it to the UI.
- UI will display value for user easy to see the temperature and humidity; see current connecting node MCU. Moreover, user can adjust the condition then send back to the Pi.

III. IMPLEMENTATION

1. Hardware

- Connect DHT-11 to node MCU.
- Run Raspberry Pi: connect to Wi-fi, Server and nod MCU.
- Setup node MCU esp8266: connect to Wi-Fi and raspberry.

2. Server

Detail:

- Create a skeleton for back-end (language, database type, database structure, local server...).
- Write API for retrieving specify data.
- Deploy back-end into a host site (includes site for api and database online) so it can be accessed easily for testing purpose.
- Write code in front-end pages to get data from server.

Database structure (tables): At the moment, our subject has 2 tables in the database:

```
- Condition: {  
    temp_condition:Number,  
    humid_condition:Number,  
    mcu_id:Number,  
    mode:String,  
    timestamp: Date  
}
```

where

temp_condition is the condition temporary for bumping water.

humid_condition is the condition humidity for bumping water.

mcu_id is uid of specify mcu.

mode is status of ON or OFF of bumping machine.

timestamp is the time that user set or change condition.

```

    28     type: Number
    29   },
    30   timestamp: Date
    31 });
    32 });
    33 const Condition = mongoose.model('Condition', {
    34   temp: {
    35     type: Number,
    36     min: 0,
    37     max: 100
    38   },
    39   humidity: {
    40     type: Number,
    41     min: 0,
    42     max: 100
    43   },
    44   mcu_id: {
    45     type: Number
    46   },
    47   mode: {
    48     type: String,
    49     enum: ['on', 'off'],
    50     default: 'off'
    51   },
    52   timestamp: Date
    53 });

```

- App: {

 temporary:Number,

 humidity:Number,

 mcu_id:Number,

 timestamp: Date

}

where

 temporary is the current temporary of ground.

 humidity is the current humidity of ground.

 mcu_id is uid of specify mcu.

 timestamp is the time the machine send data.

```

    1 var mongoose = require('mongoose');
    2
    3 // connect to data base
    4 mongoose.connect(
    5   'mongodb://ductruong:cudin5897@ds237713.mlab.com:37713/iot',
    6   function (err) {
    7     if (err) throw err;
    8     console.log('Successfully connected to data');
    9   }
    10 );
    11
    12 const App = mongoose.model('App', {
    13   temporary: {
    14     type: Number,
    15     min: 0,
    16     max: 100
    17   },
    18   humidity: {
    19     type: Number,
    20     min: 0,
    21     max: 100
    22   },
    23   mcu_id: {
    24     type: Number
    25   },
    26   timestamp: Date
    27 });

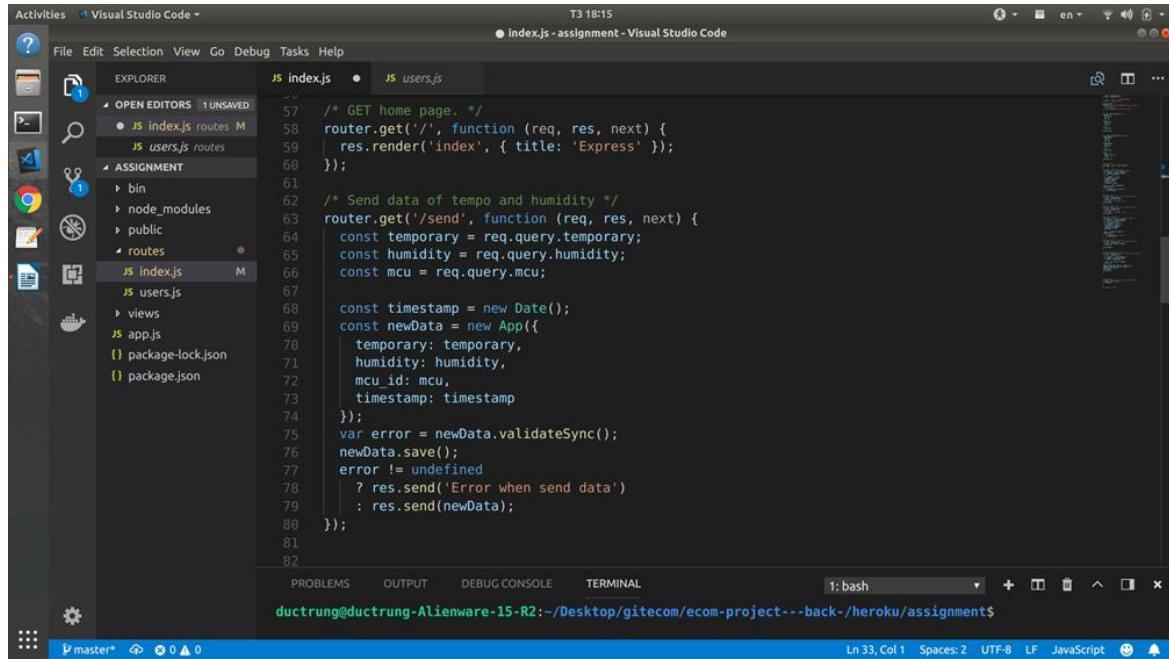
```

2) Write API:

In this project, raspberry pi will receive and send data to server.

- Write api for sending data to database: which requests user to input current temporary, humidity, mcu_id. Example for this api:

<https://iot-assignment.herokuapp.com/send?temporary=30&humidity=60&uid=3488A17988A139>



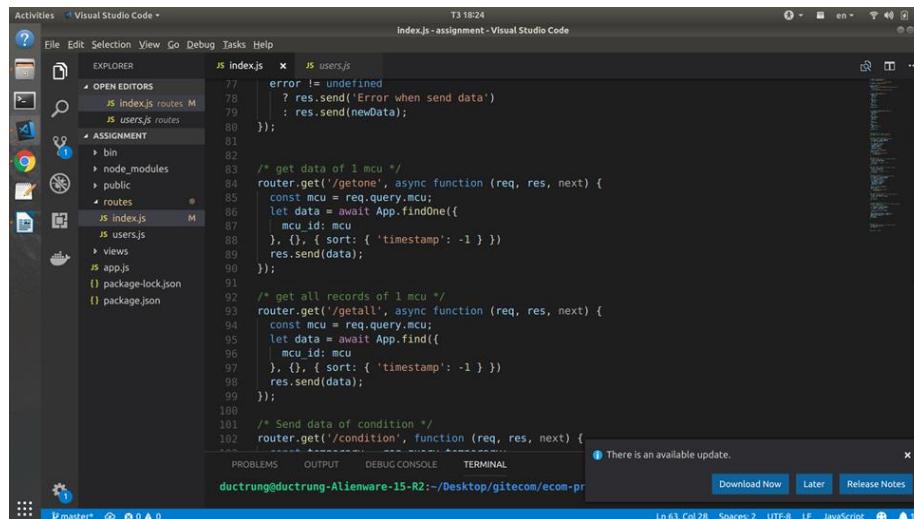
```
Activities Visual Studio Code • T3 18:15
File Edit Selection View Go Debug Tasks Help
OPEN EDITORS 1 UNSAVED
JS index.js • JS users.js
EXPLORER
ASSIGNMENT
routes
bin
node_modules
public
routes
JS index.js M
JS users.js
views
JS app.js
() package-lock.json
() package.json
57 /* GET home page. */
58 router.get('/', function (req, res, next) {
59 | res.render('index', { title: 'Express' });
60 });
61
62 /* Send data of tempo and humidity */
63 router.get('/send', function (req, res, next) {
64 | const temporary = req.query.temporary;
65 | const humidity = req.query.humidity;
66 | const mcu = req.query.mcu;
67
68 | const timestamp = new Date();
69 | const newData = new App({
70 | temporary: temporary,
71 | humidity: humidity,
72 | mcu_id: mcu,
73 | timestamp: timestamp
74 });
75 | var error = newData.validateSync();
76 | newData.save();
77 | error != undefined
78 | ? res.send('Error when send data')
79 | : res.send(newData);
80 });
81
82
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
ductrung@ductrung-Alienware-15-R2:~/Desktop/gitecom/ecom-project---back-/heroku/assignments$ 1:bash
Ln 33, Col 1 Spaces:2 UTF-8 LF JavaScript
master* ↵ 0 0 0
ductrung@ductrung-Alienware-15-R2:~/Desktop/gitecom/ecom-project---back-/heroku/assignments$
```

- Api for getting the latest data of mcu due to its uid which request uid of mcu

Example for this api: <https://iot-assignment.herokuapp.com/getone?mcu=3488A17988A139>

- And another api for getting all data of that mcu:

Example for this api: <https://iot-assignment.herokuapp.com/getall?mcu=123>

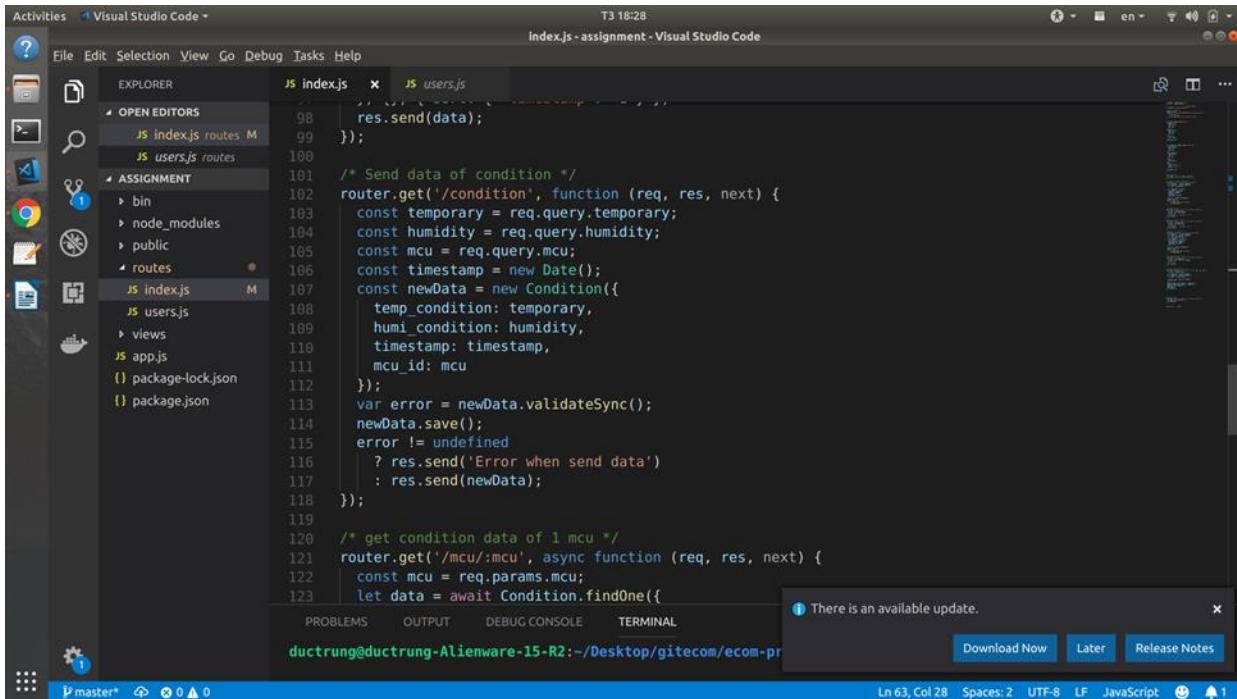


```
Activities Visual Studio Code • T3 18:24
File Edit Selection View Go Debug Tasks Help
OPEN EDITORS 1 UNSAVED
JS index.js x JS users.js
EXPLORER
ASSIGNMENT
routes
bin
node_modules
public
routes
JS index.js M
JS users.js
views
JS app.js
() package-lock.json
() package.json
77 | error != undefined
78 | ? res.send('Error when send data')
79 | : res.send(newData);
80 });
81
82
83 /* get data of 1 mcu */
84 router.get('/getone', async function (req, res, next) {
85 | const mcu = req.query.mcu;
86 | let data = await App.findOne({
87 | mcu_id: mcu
88 | }, {}, { sort: { 'timestamp': -1 } })
89 | res.send(data);
90 });
91
92 /* get all records of 1 mcu */
93 router.get('/getall', async function (req, res, next) {
94 | const mcu = req.query.mcu;
95 | let data = await App.find({
96 | mcu_id: mcu
97 | }, {}, { sort: { 'timestamp': -1 } })
98 | res.send(data);
99 });
100
101 /* Send data of condition */
102 router.get('/condition', function (req, res, next) {
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
ductrung@ductrung-Alienware-15-R2:~/Desktop/gitecom/ecom-pr
There is an available update.
Download Now Later Release Notes
Ln 63, Col 28 Spaces:2 UTF-8 LF JavaScript
master* ↵ 0 0 0
ductrung@ductrung-Alienware-15-R2:~/Desktop/gitecom/ecom-pr
```

- Api to set the condition of temporary and humidity for the bump machine to water.

It requests temporary condition, humidity condition, mcu id. Example for this api:

<https://iot-assignment.herokuapp.com/condition?temporary=28&humidity=58&mcu=3488A17988A139>

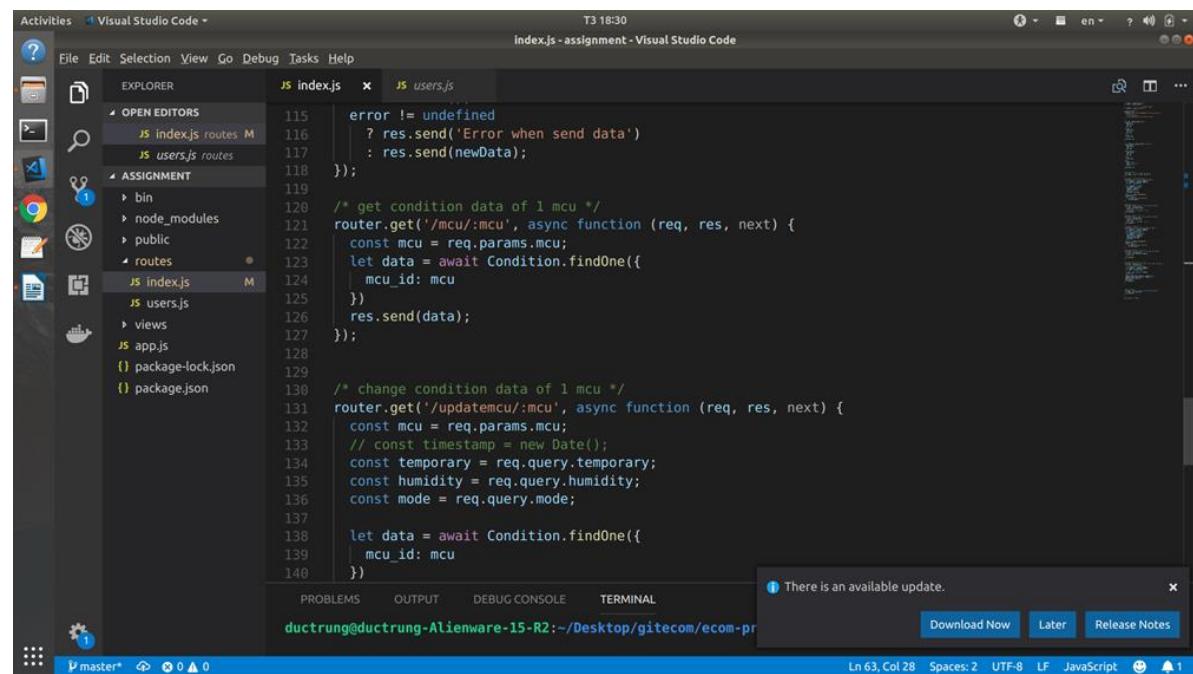


The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files like index.js, users.js, routes, and app.js.
- Editor:** Displays a portion of the index.js file containing code to handle a POST request for setting conditions. The code includes variables for temporary, humidity, and mcu_id, and uses a Condition model to save the data.
- Bottom Status Bar:** Shows the command `ductrung@ductrung-Alienware-15-R2:~/Desktop/gitecom/ecom-pr` and status information like Ln 63, Col 28, Spaces: 2, UTF-8, LF, JavaScript.

- Api to get condition of one mcu which requests mcu uid. Example for this api:

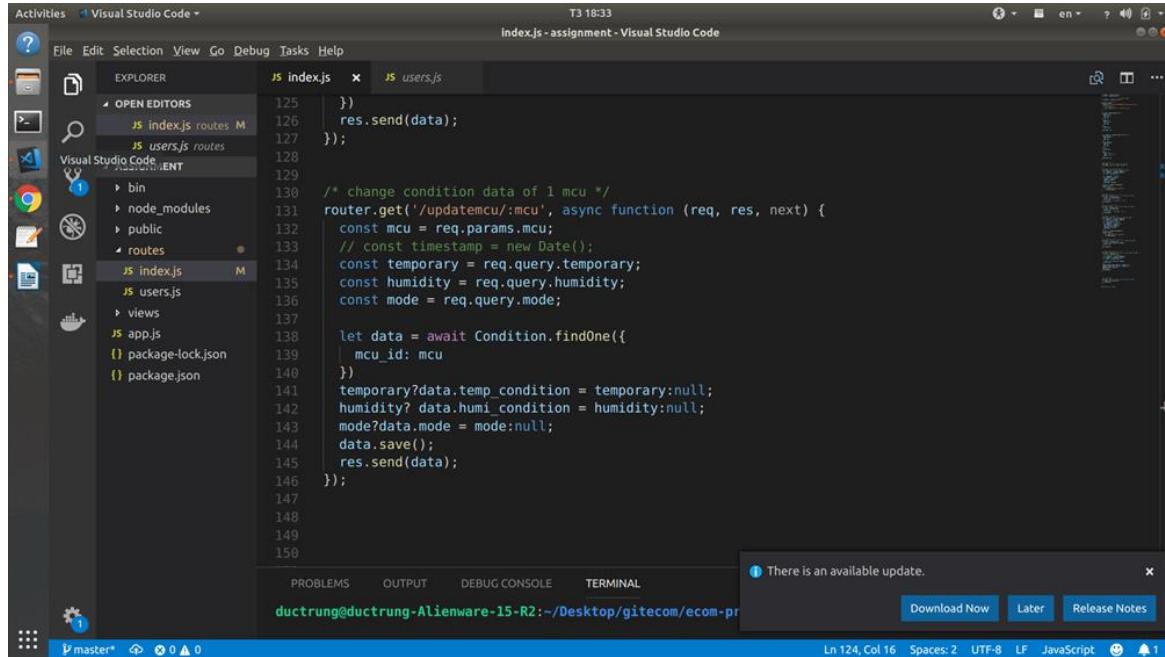
<https://iot-assignment.herokuapp.com/mcu/123>



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files like index.js, users.js, routes, and app.js.
- Editor:** Displays a portion of the index.js file containing code to handle a GET request for a specific mcu. It uses a Condition model to find the data for the given mcu_id.
- Bottom Status Bar:** Shows the command `ductrung@ductrung-Alienware-15-R2:~/Desktop/gitecom/ecom-pr` and status information like Ln 63, Col 28, Spaces: 2, UTF-8, LF, JavaScript.

- Api to change condition of one mcu which requests mcu uid, humidity condition, temporary condition. Example for this api:



```

T3 18:33
index.js - assignment - Visual Studio Code

File Edit Selection View Go Debug Tasks Help
OPEN EDITORS JS index.js x JS users.js
JS index.js routes M JS users.js routes M
Visual Studio Code RESUME ENT
bin node_modules public routes
JS index.js M JS users.js M
views app.js
package-lock.json package.json
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
ductrung@ductrung-Alienware-15-R2:~/Desktop/gitecom/ecom-pr
Ln 124, Col 16 Spaces: 2 UTF-8 LF JavaScript 1
Download Now Later Release Notes

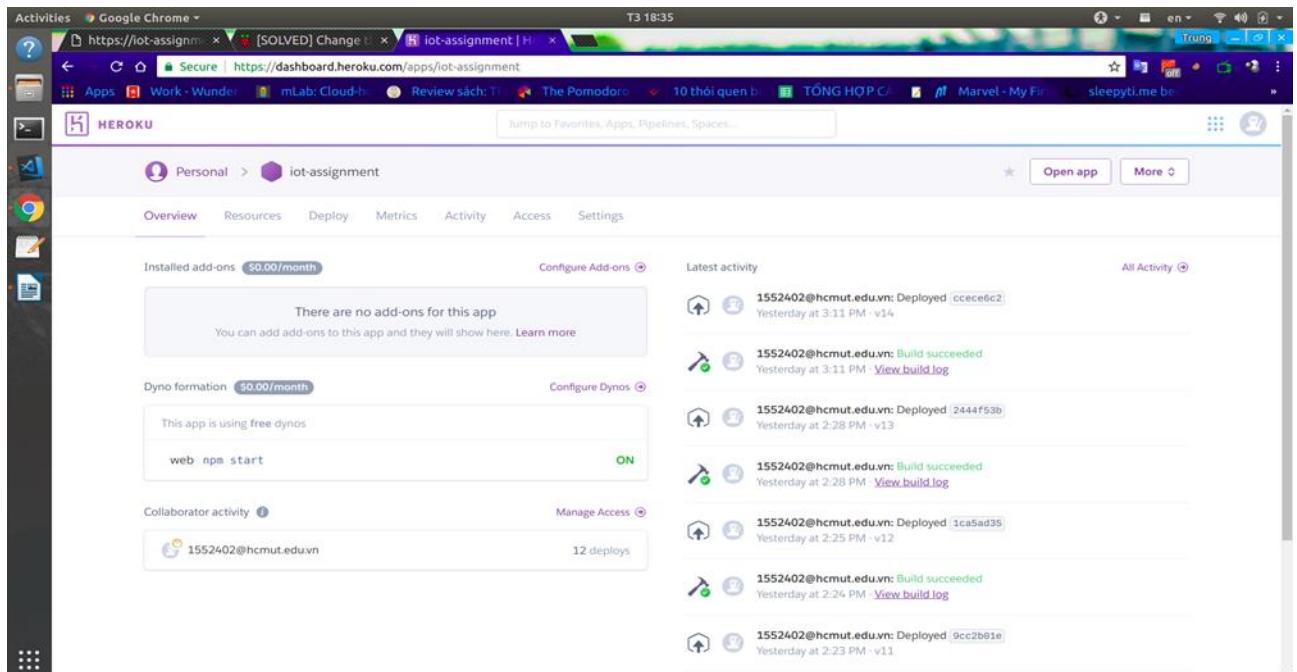
```

The screenshot shows the Visual Studio Code interface with two files open: index.js and users.js. The code in index.js includes a section for handling an update request to change the condition of a specific MCU. It uses Express.js routes and MongoDB queries to find and update the MCU's condition based on provided parameters.

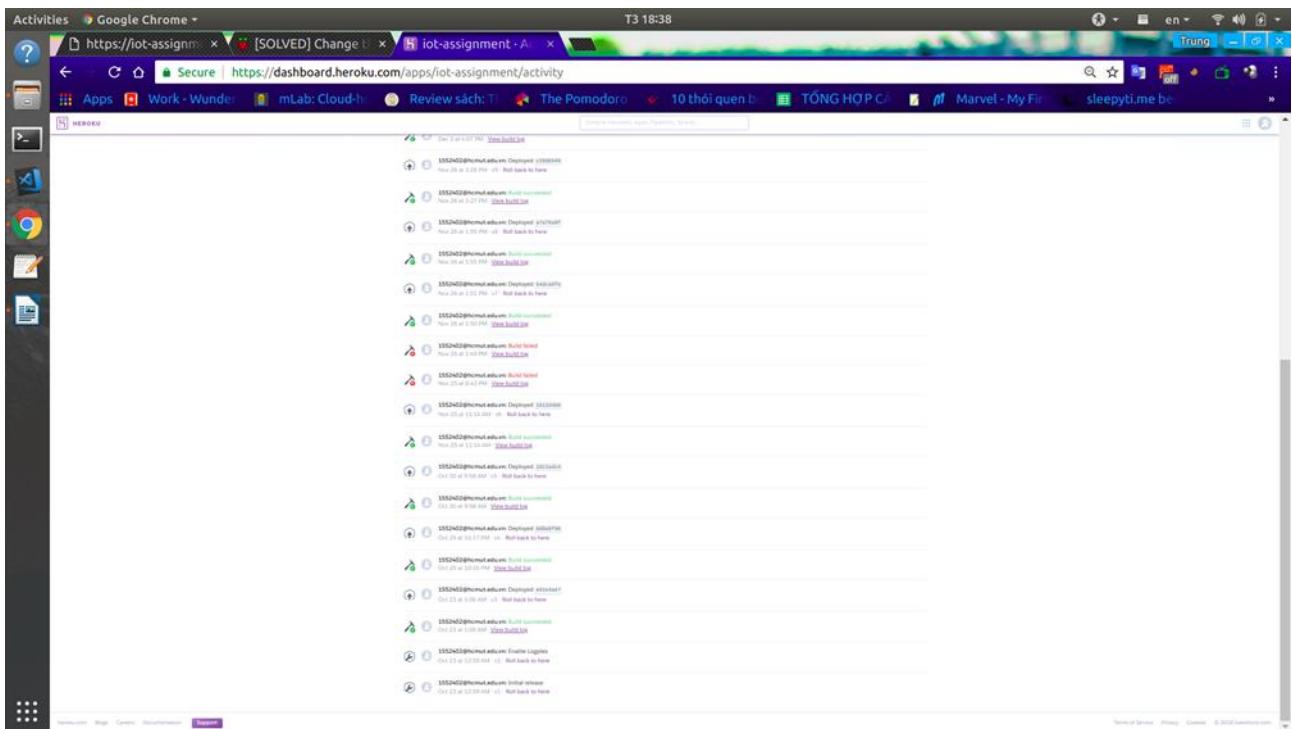
<https://iot-assignment.herokuapp.com/updatemcu/123?temporary=27&humidity=80>

3) Deploy back-end into a host site (api and database online)

- Heroku is free host site and its can be integrated with Nodejs so I can use it to update the apis which created in local and push them by GIT.



The screenshot shows the Heroku dashboard for the 'iot-assignment' application. It displays the deployment history with several successful deployments listed, each showing the user who deployed, the timestamp, and the deployment ID. The dashboard also shows sections for add-ons, dyno formation, and collaborator activity.



- For the database I use a free cloud database name MLAB which give me 500mb for storage.

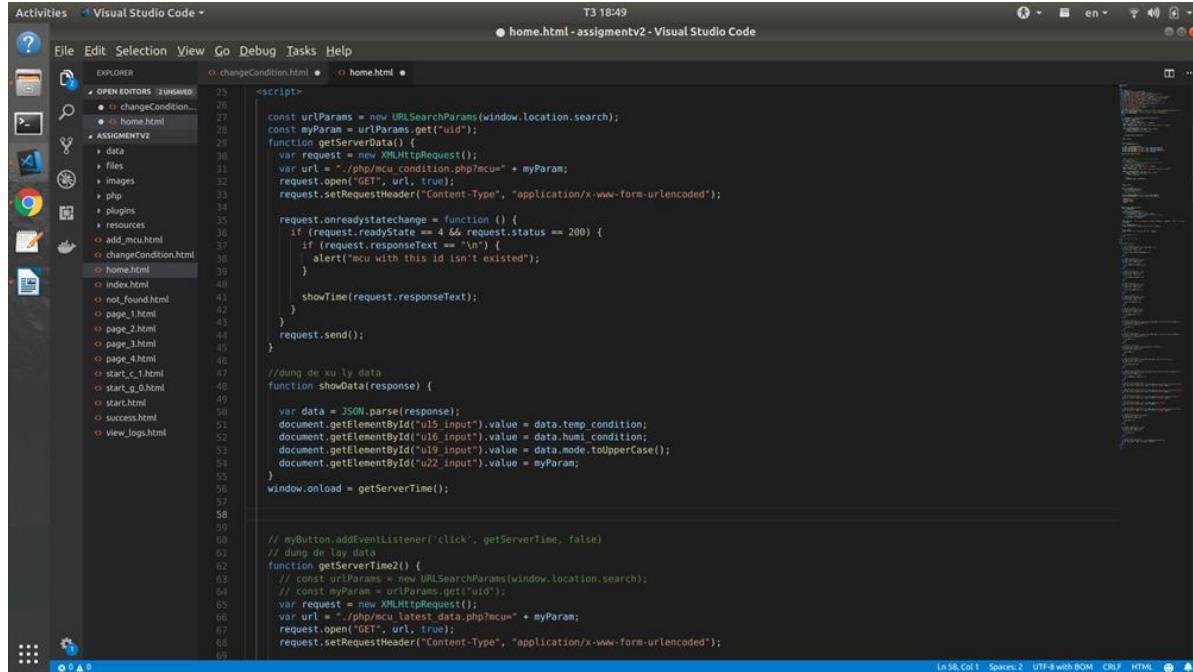
NAME	DOCUMENTS	CAPPED?	SIZE
apps	8	false	8.86 KB
conditions	5	false	9.16 KB

4) Write code in front-end pages to get data from server.

In this part, I usually use javascript in html page and sometimes use php for retrieving data from api (because of something 's not working when retrieving directly from api).

Here are some examples of functions using to access api:

Function to get condition:



```
const urlParams = new URLSearchParams(window.location.search);
const myParam = urlParams.get("uid");
function getServerData() {
    var request = new XMLHttpRequest();
    var url = "./php/mcu_condition.php?mcu=" + myParam;
    request.open("GET", url, true);
    request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

    request.onreadystatechange = function () {
        if (request.readyState == 4 && request.status == 200) {
            if (request.responseText == "n") {
                alert("mcu with this id isn't existed");
            }
            showTime(request.responseText);
        }
        request.send();
    }
}

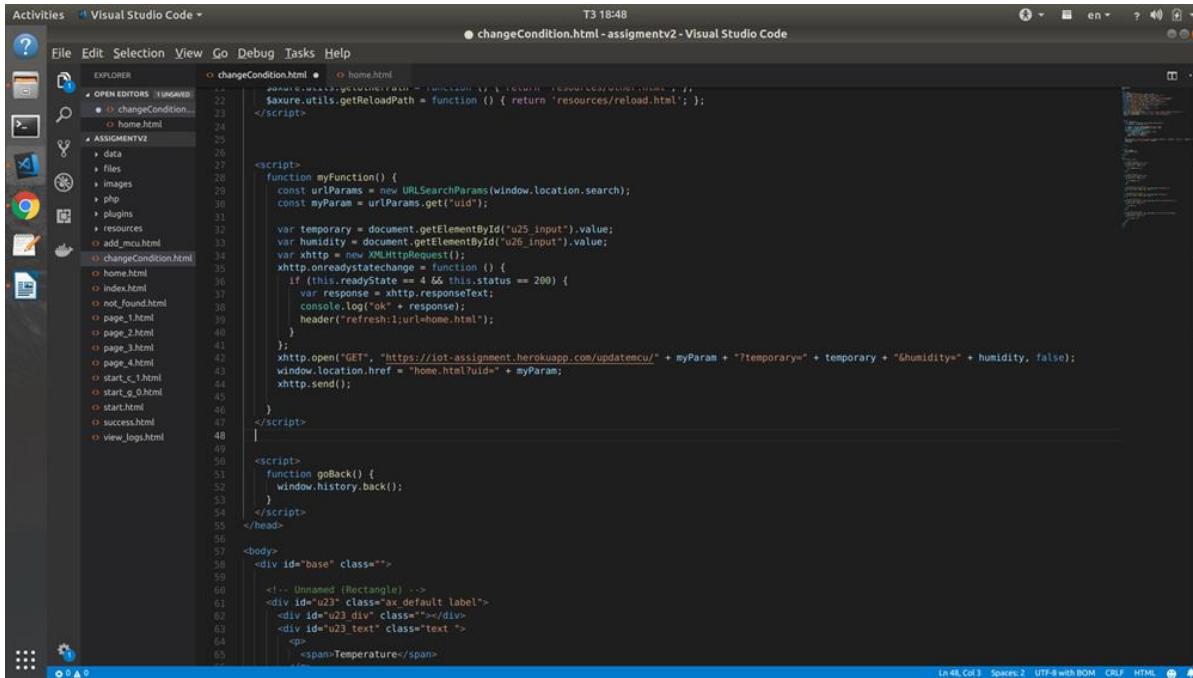
// dung de xu ly data
function showData(response) {

    var data = JSON.parse(response);
    document.getElementById("u15_input").value = data.temp_condition;
    document.getElementById("u19_input").value = data.humi_condition;
    document.getElementById("u22_input").value = myParam;
}

window.onload = getServerTime();

// dung de lay data
function getServerTime2() {
    // const urlParams = new URLSearchParams(window.location.search);
    // const myParam = urlParams.get("uid");
    var request = new XMLHttpRequest();
    var url = "./php/mcu/latest_data.php?mcu=" + myParam;
    request.open("GET", url, true);
    request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
}
```

Function to update condition:



```
$axure.utils.getReloadPath = function () { return 'resources/reload.html'; };

<script>
function myFunction() {
    const urlParams = new URLSearchParams(window.location.search);
    const myParam = urlParams.get("uid");

    var temporary = document.getElementById("u25_input").value;
    var humidity = document.getElementById("u26_input").value;
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            var response = xhttp.responseText;
            console.log("ok" + response);
            header("refresh:1:url=home.html");
        }
    }
    xhttp.open("GET", "https://iot-assignment.herokuapp.com/updatemcu/" + myParam + "?temporary=" + temporary + "&humidity=" + humidity, false);
    window.location.href = "home.html?uids=" + myParam;
    xhttp.send();
}
</script>

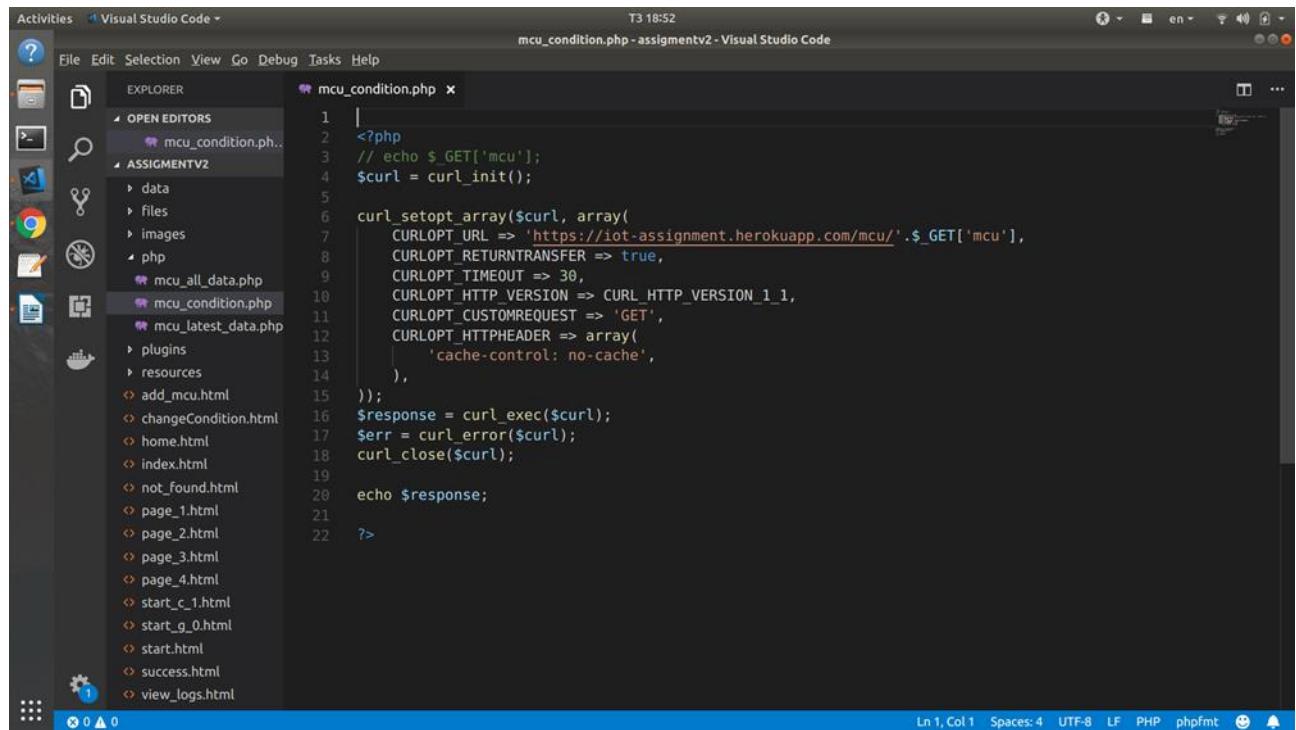
<script>
function goBack() {
    window.history.back();
}
</script>
</head>

<body>
<div id="base" class="">
    <!-- Unnamed (Rectangle) -->
    <div id="u23" class="ax_default label">
        <div id="u23_div" class=""></div>
        <div id="u23_text" class="text">
            <p>
                <span>Temperature</span>
            </p>
        </div>
    </div>

```

PHP file for accessing api:

- Get MCU condition



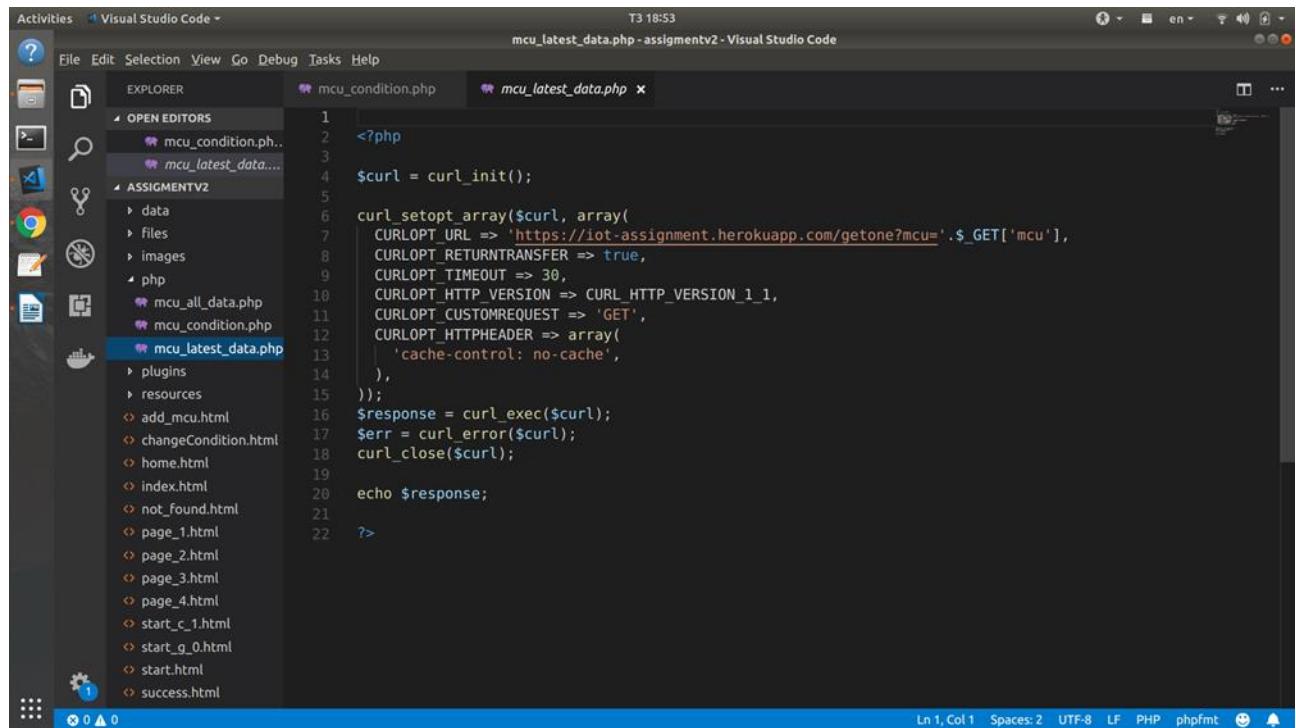
The screenshot shows the Visual Studio Code interface with the title bar "Activities Visual Studio Code" and "T3 18:52". The main area displays the code for "mcu_condition.php" located in the "ASSIGNMENTV2" folder. The code uses cURL to make a GET request to a specified URL and prints the response.

```
<?php
// echo $_GET['mcu'];
$curl = curl_init();

curl_setopt_array($curl, array(
    CURLOPT_URL => 'https://iot-assignment.herokuapp.com/mcu/' . $_GET['mcu'],
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_TIMEOUT => 30,
    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
    CURLOPT_CUSTOMREQUEST => 'GET',
    CURLOPT_HTTPHEADER => array(
        'cache-control: no-cache',
    ),
));
$response = curl_exec($curl);
$err = curl_error($curl);
curl_close($curl);

echo $response;
?>
```

- Get latest data of MCU



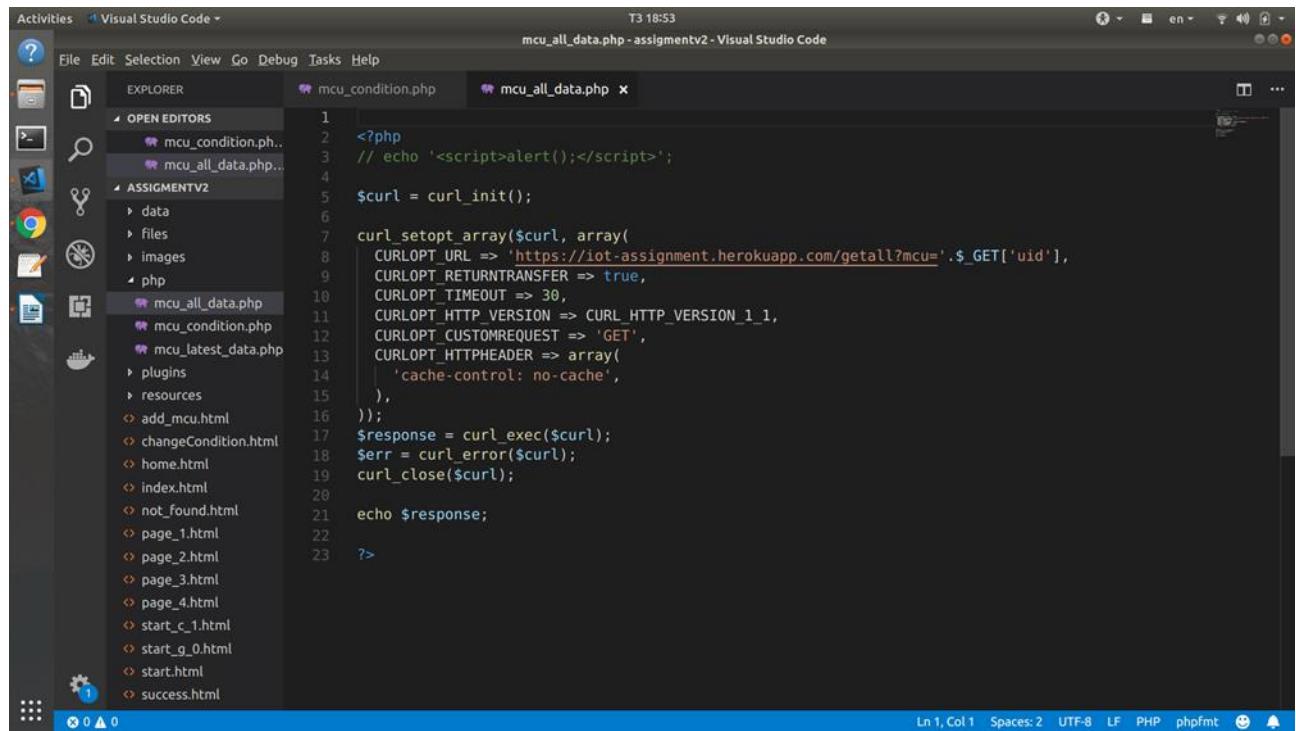
The screenshot shows the Visual Studio Code interface with the title bar "Activities Visual Studio Code" and "T3 18:53". The main area displays the code for "mcu_latest_data.php" located in the "ASSIGNMENTV2" folder. This file is identical to "mcu_condition.php" but includes an additional parameter in the cURL URL to specify the type of data to retrieve.

```
<?php
$curl = curl_init();

curl_setopt_array($curl, array(
    CURLOPT_URL => 'https://iot-assignment.herokuapp.com/getone?mcu=' . $_GET['mcu'],
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_TIMEOUT => 30,
    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
    CURLOPT_CUSTOMREQUEST => 'GET',
    CURLOPT_HTTPHEADER => array(
        'cache-control: no-cache',
    ),
));
$response = curl_exec($curl);
$err = curl_error($curl);
curl_close($curl);

echo $response;
?>
```

- Get all data of MCU for view logs



The screenshot shows the Visual Studio Code interface with a dark theme. In the Explorer sidebar, there are several files listed under 'ASSIGNMENTV2' and 'php'. The file 'mcu_all_data.php' is currently open in the editor. The code is a PHP script that uses cURL to fetch data from a specific URL. The URL contains a query parameter 'mcu=' followed by the value of the \$_GET['uid'] variable. The code includes comments explaining the use of curl_setopt_array and curl_exec.

```
<?php
// echo '<script>alert();</script>';

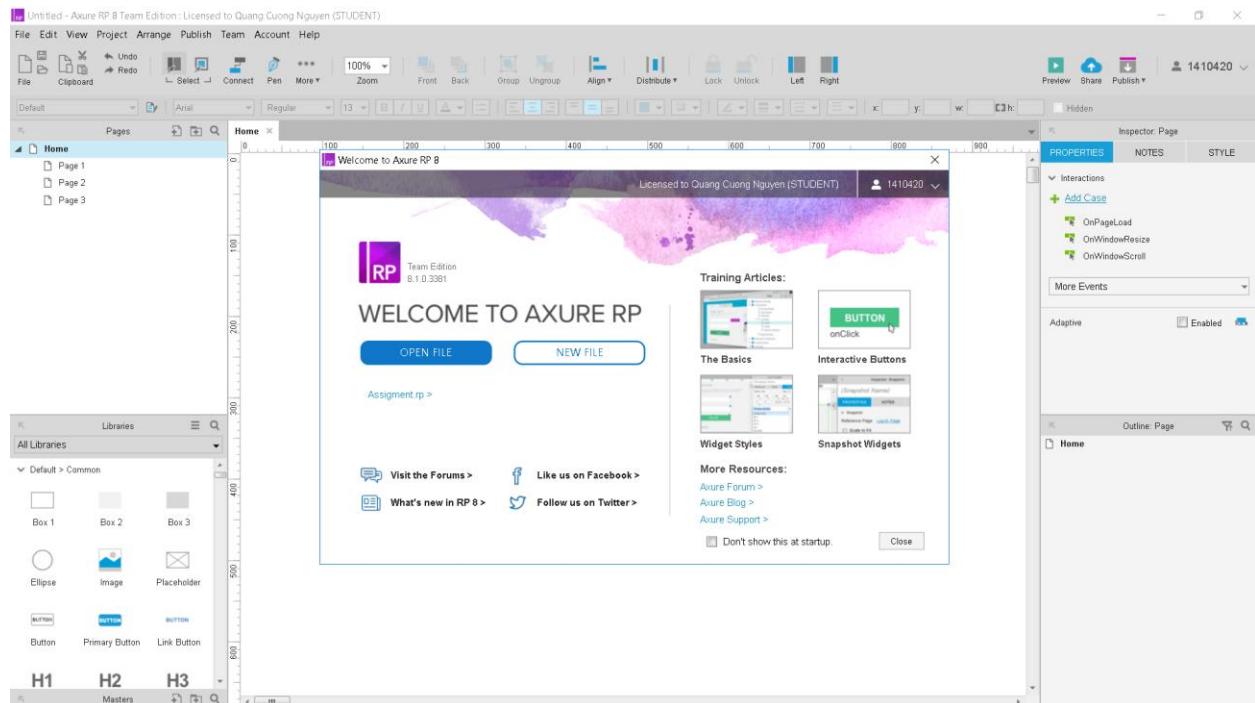
$curl = curl_init();

curl_setopt_array($curl, array(
    CURLOPT_URL => 'https://iot-assignment.herokuapp.com/getall?mcu='.$_GET['uid'],
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_TIMEOUT => 30,
    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
    CURLOPT_CUSTOMREQUEST => 'GET',
    CURLOPT_HTTPHEADER => array(
        'cache-control: no-cache',
    ),
));
$response = curl_exec($curl);
$err = curl_error($curl);
curl_close($curl);

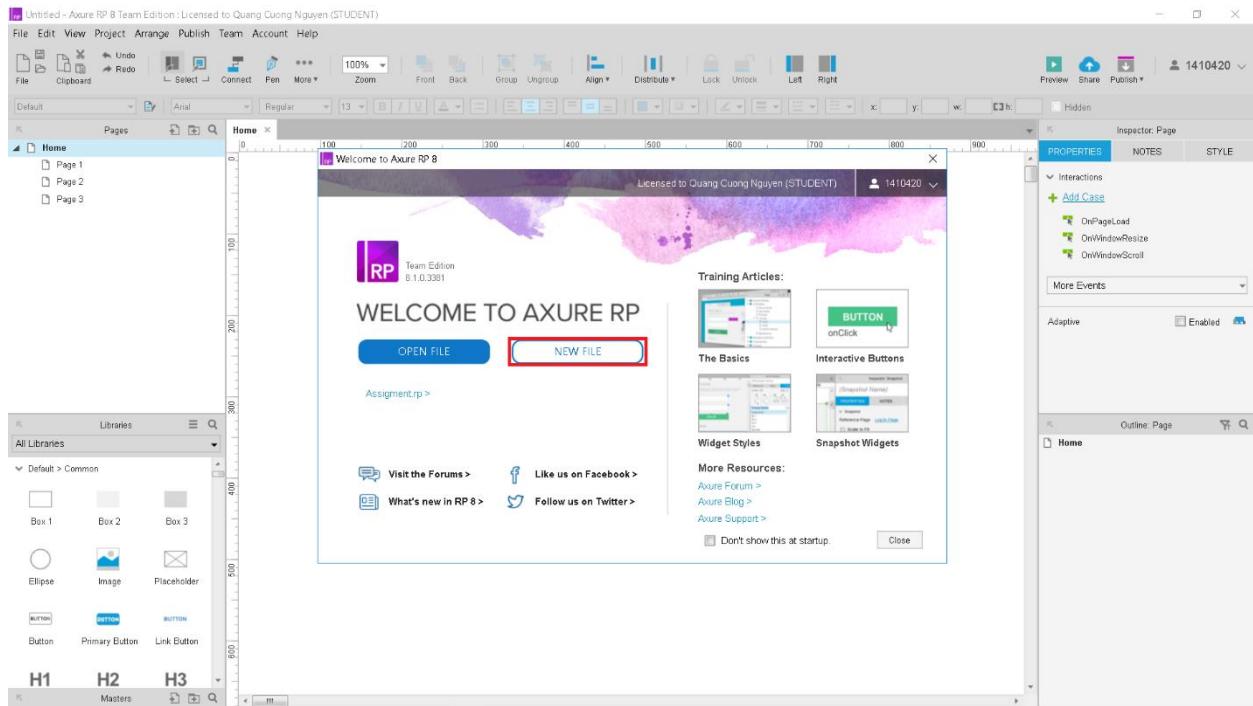
echo $response;
?>
```

3. UI

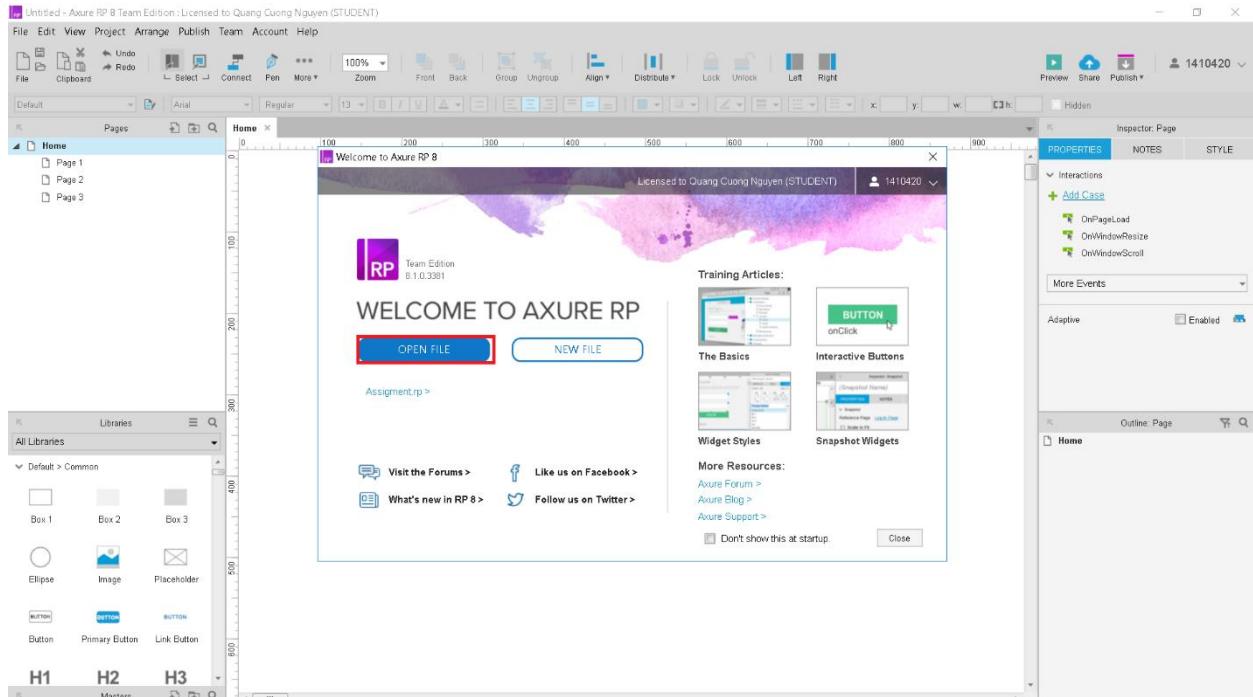
- Download Axure RP8 at <https://www.axure.com/download>
- Open Axure RP



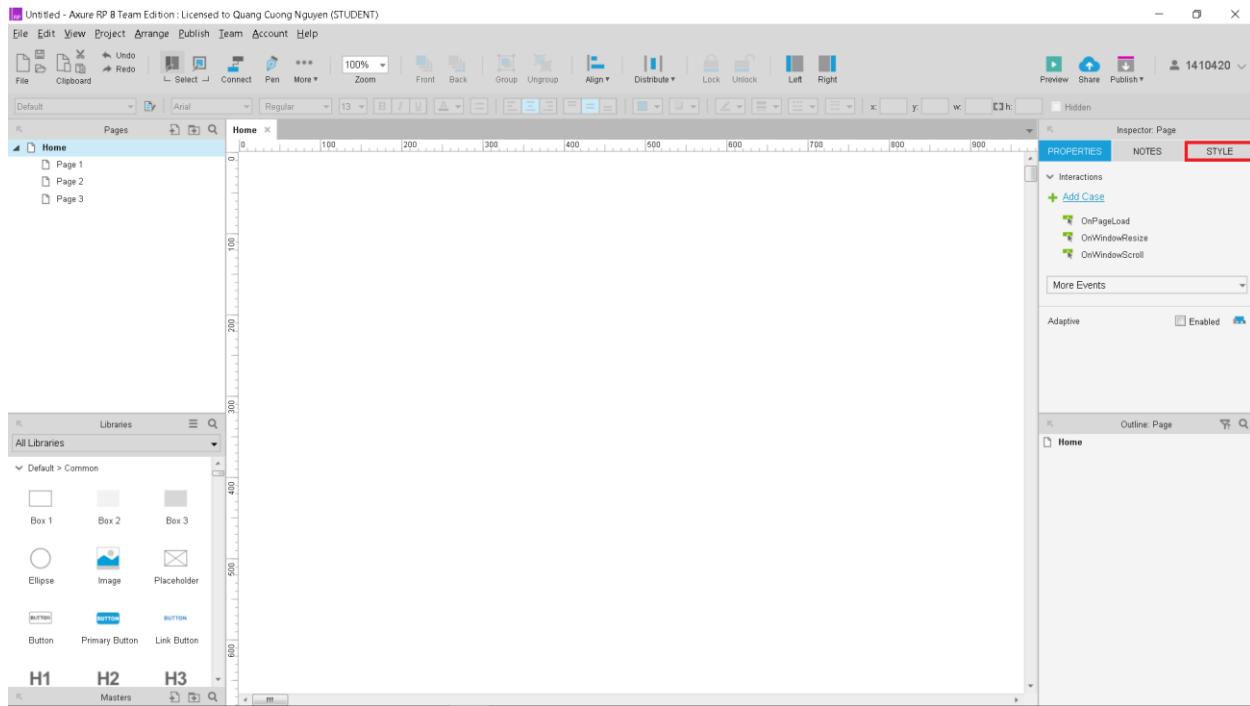
- Create new file



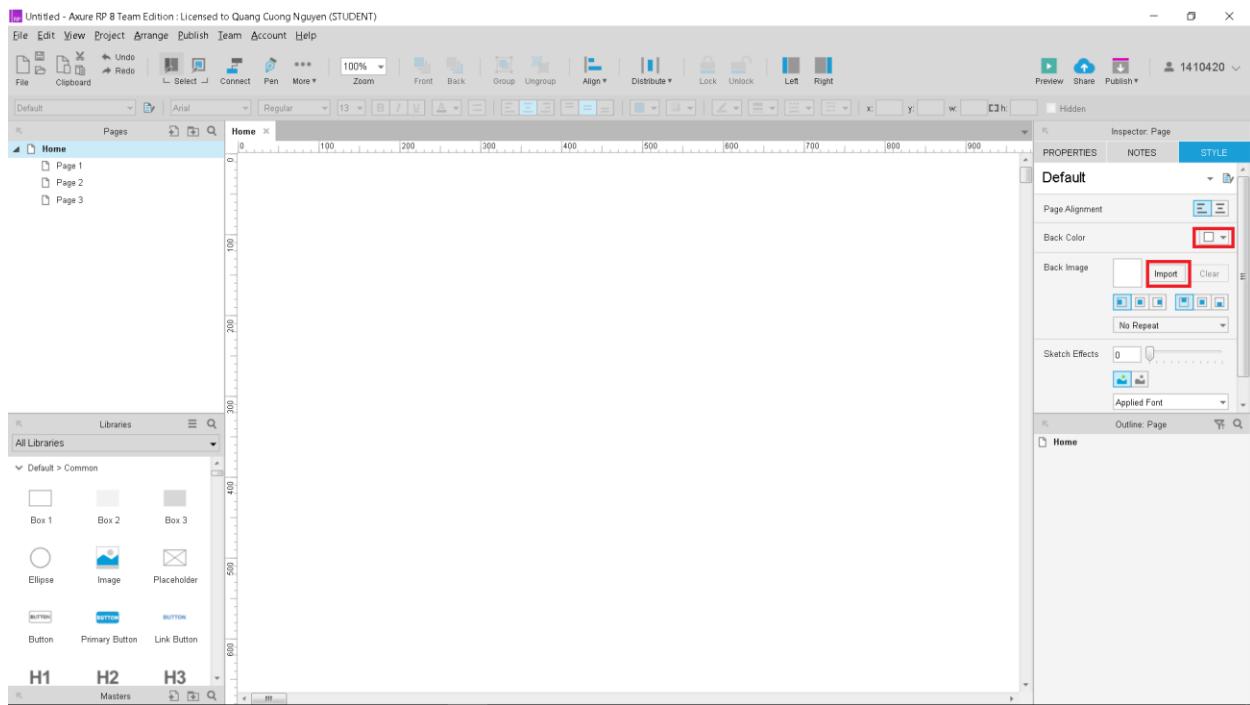
- Create new file



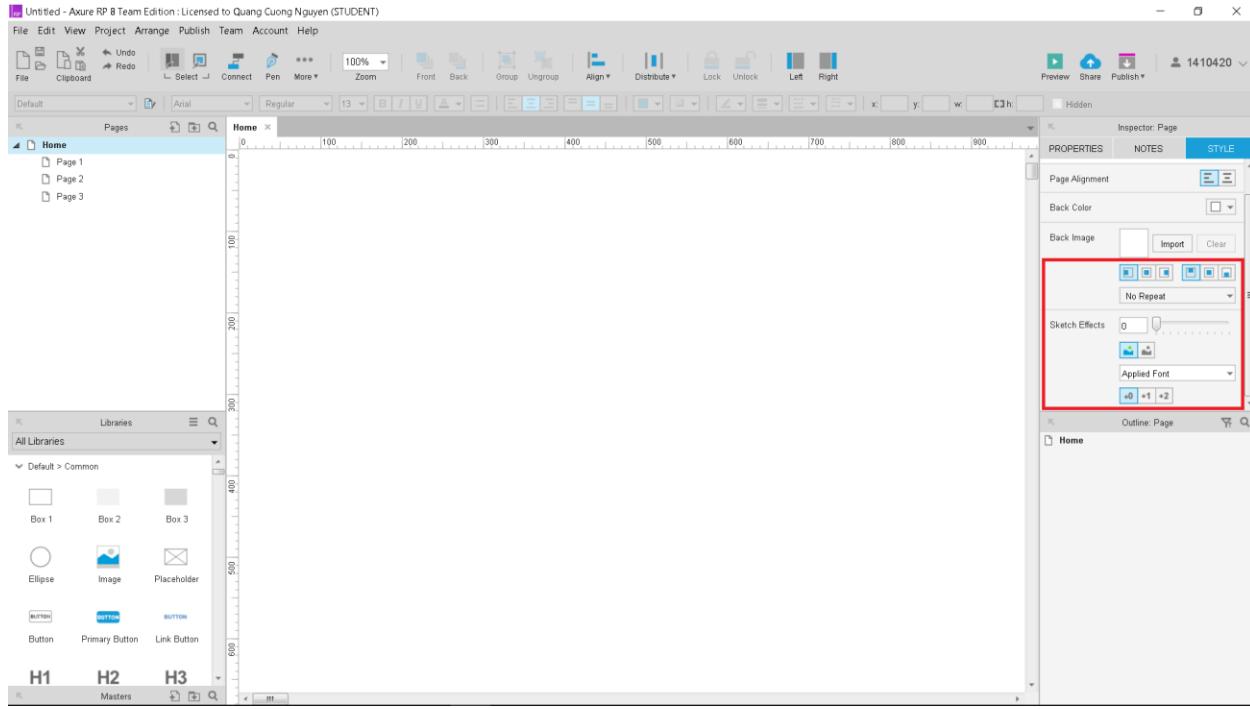
- Open existed file
- Design background



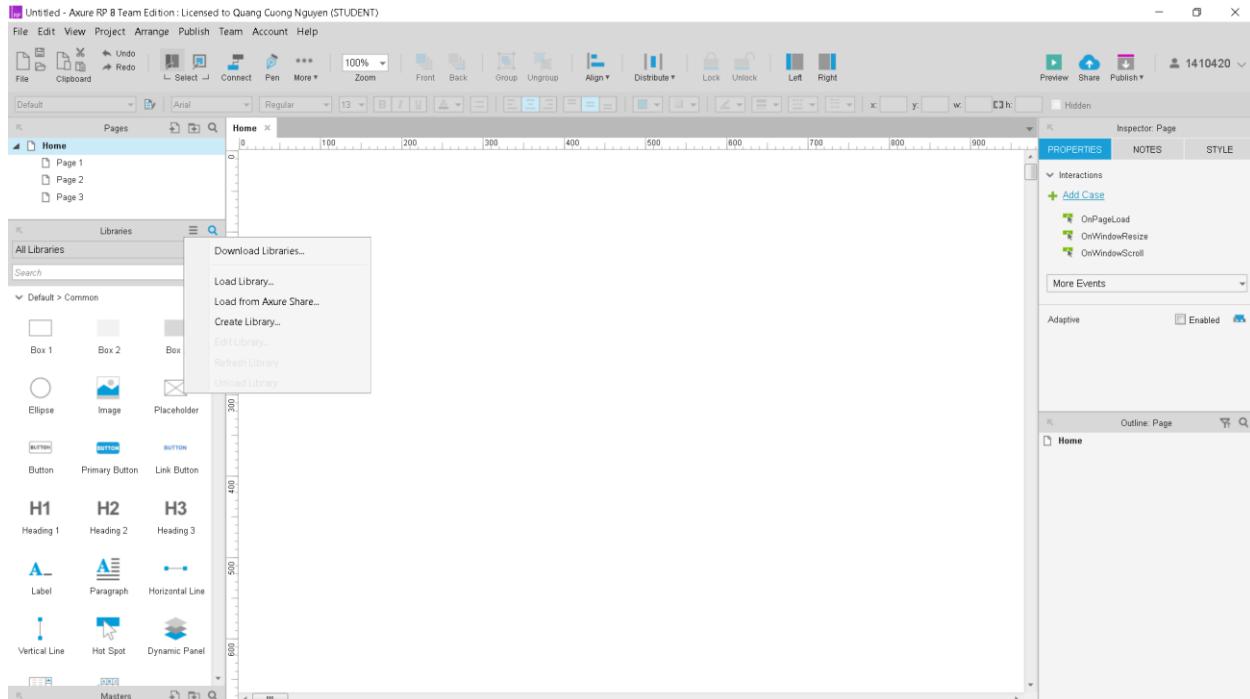
- Click Style



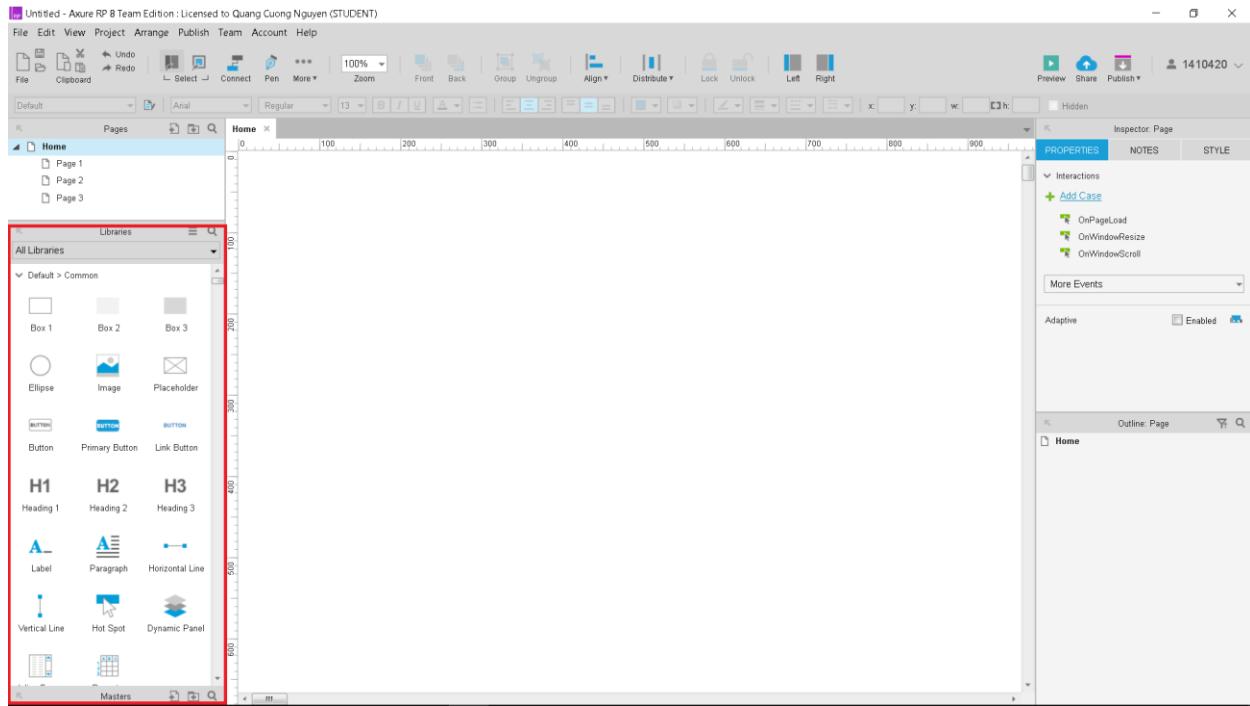
- Choose color or Import picture for background



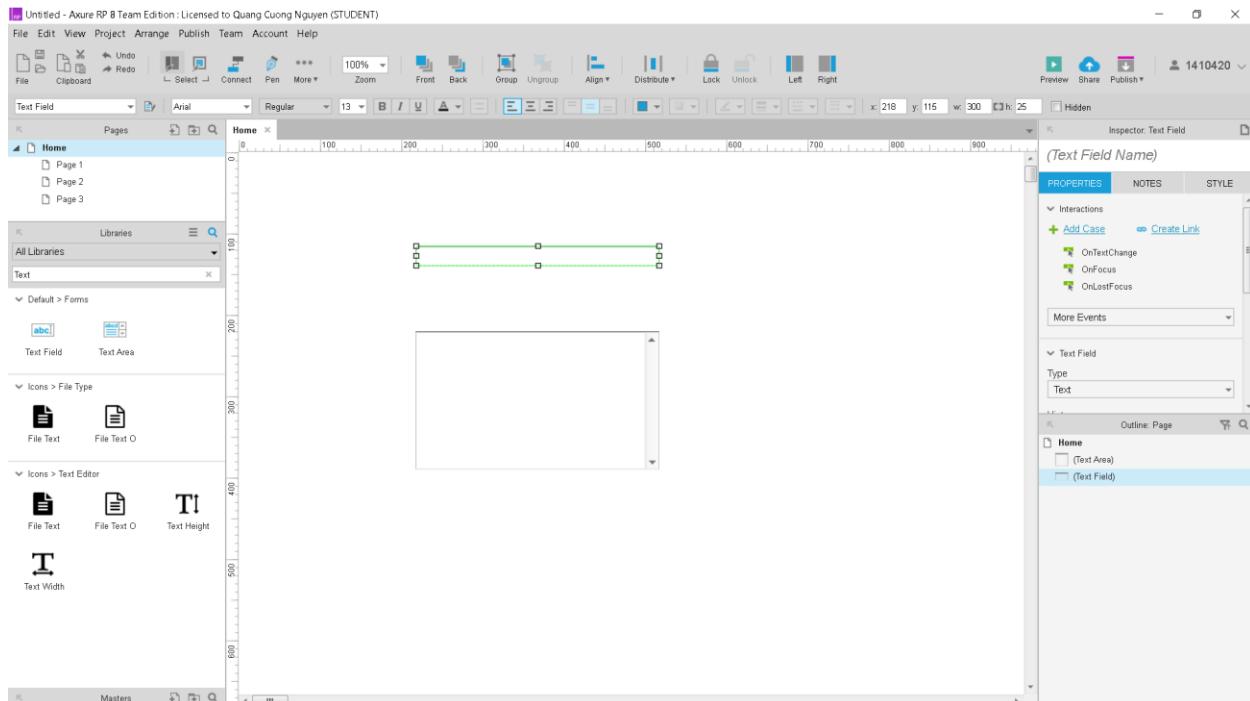
- Edit background
- Design text field / area



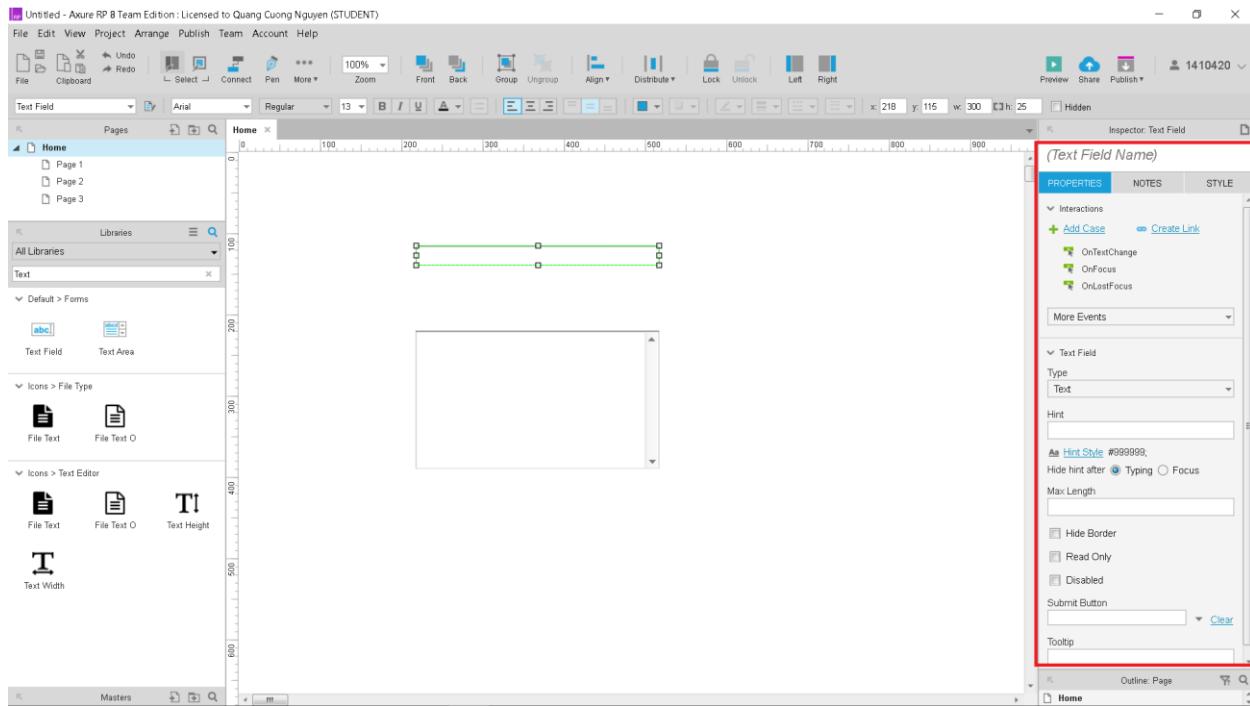
- Download and add more libraries



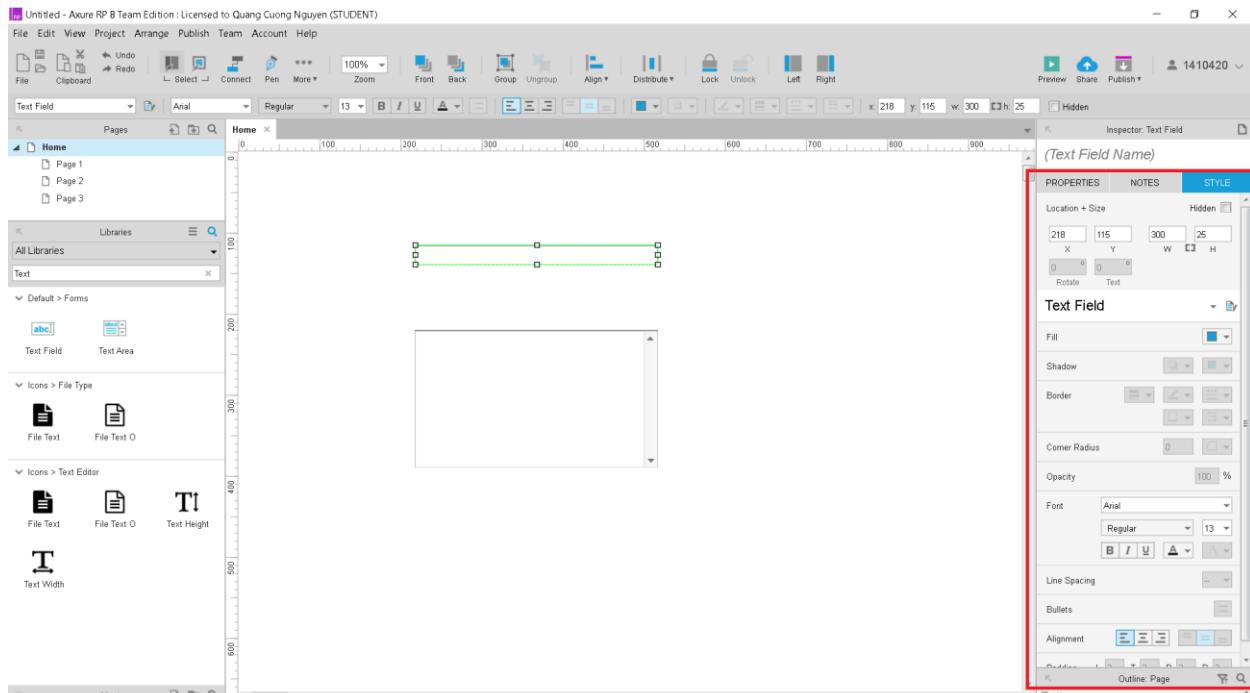
- Object for design website



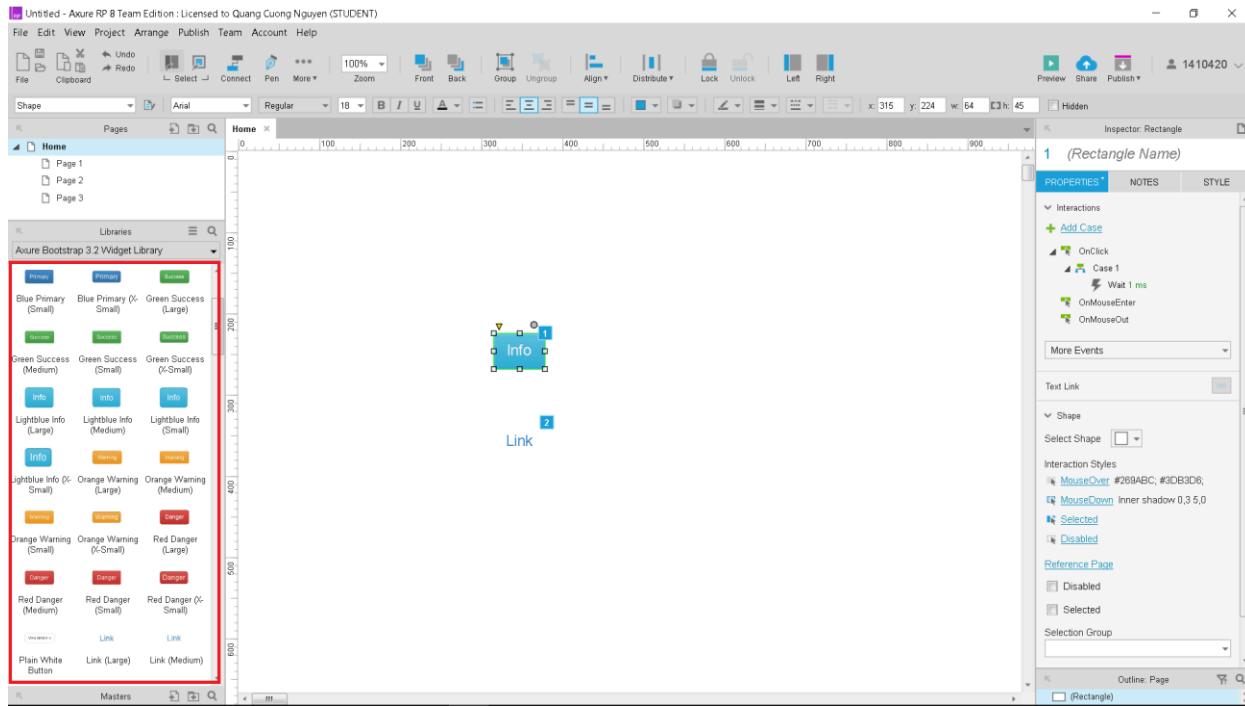
- Create text field and text area



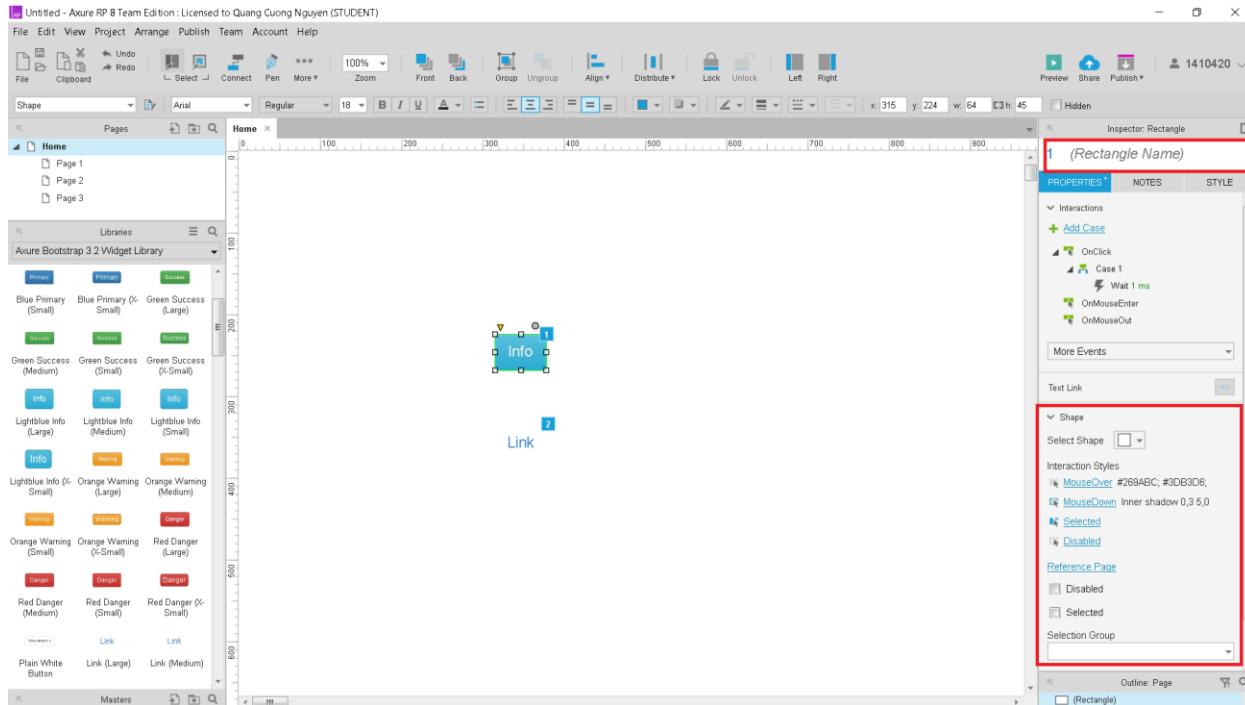
- Edit name and features



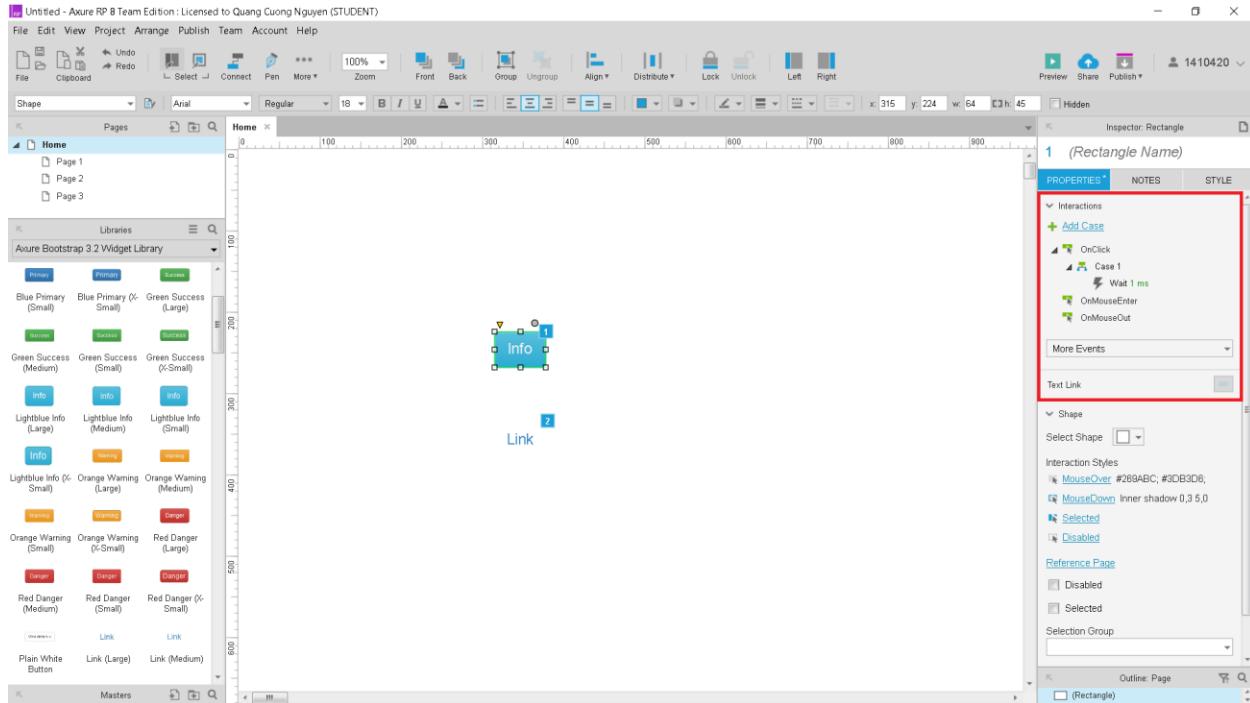
- Edit style (font, format...)
- Design button and link



- Some buttons and links

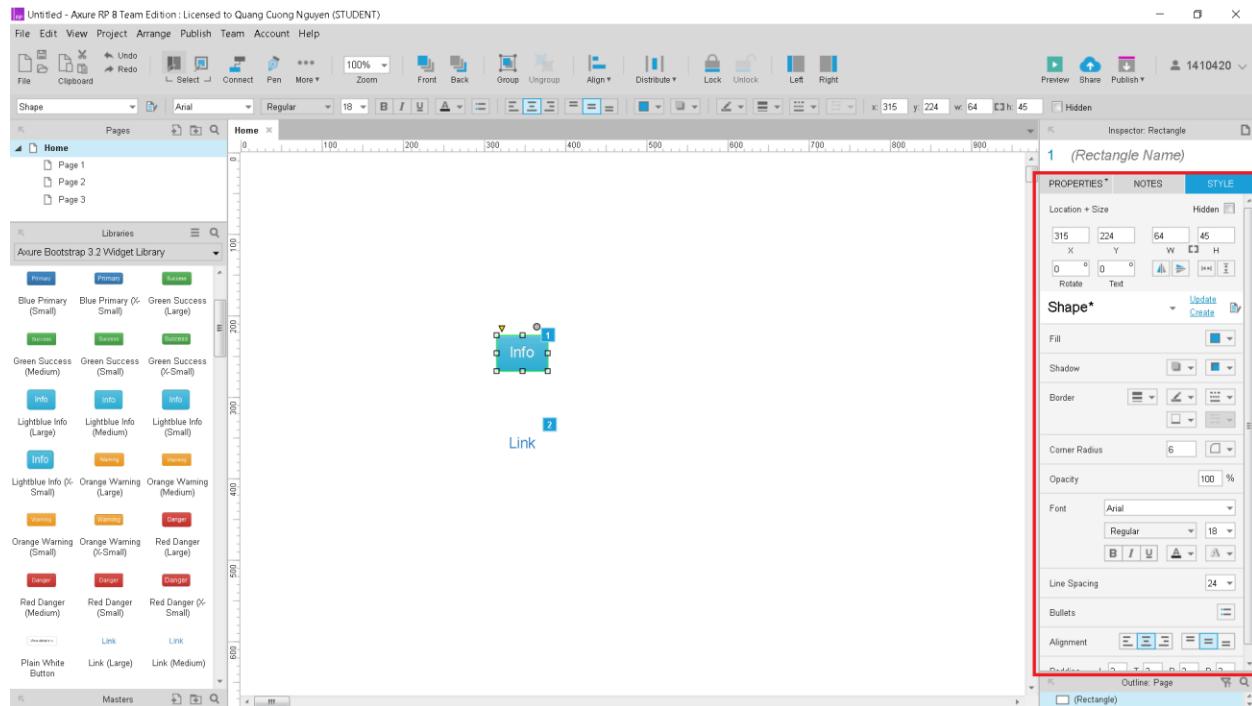


- Edit name and features

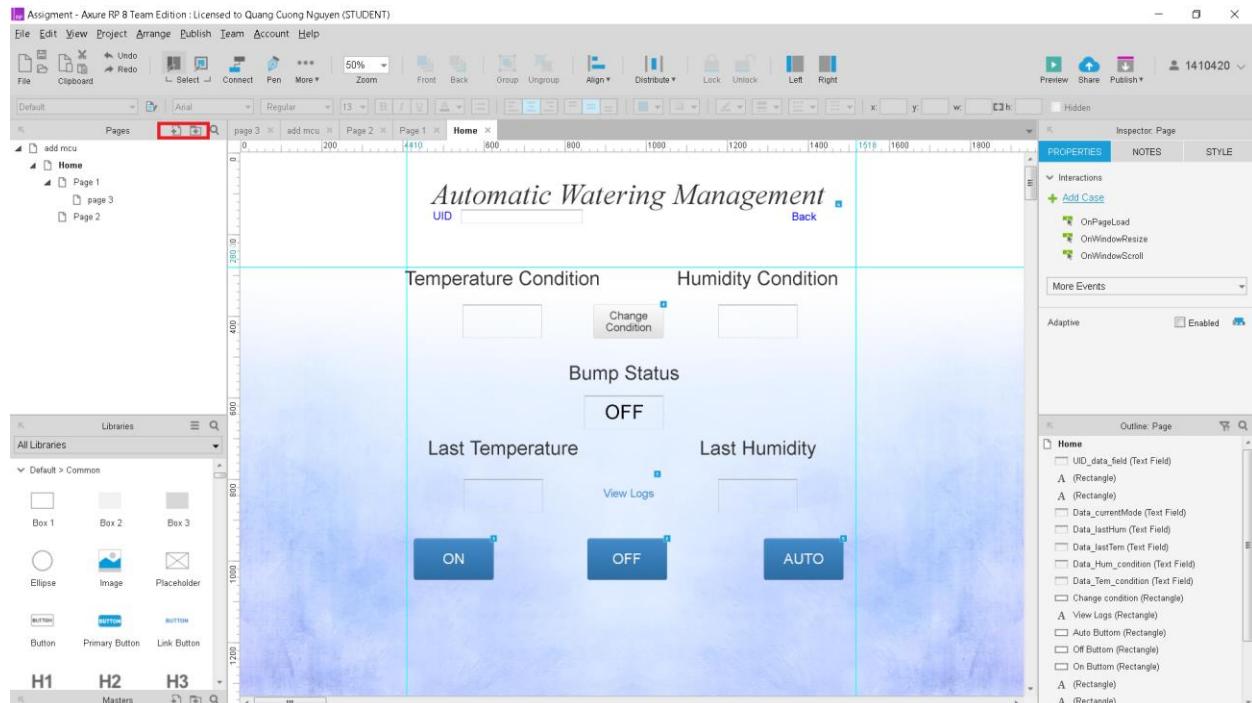


- Edit actions

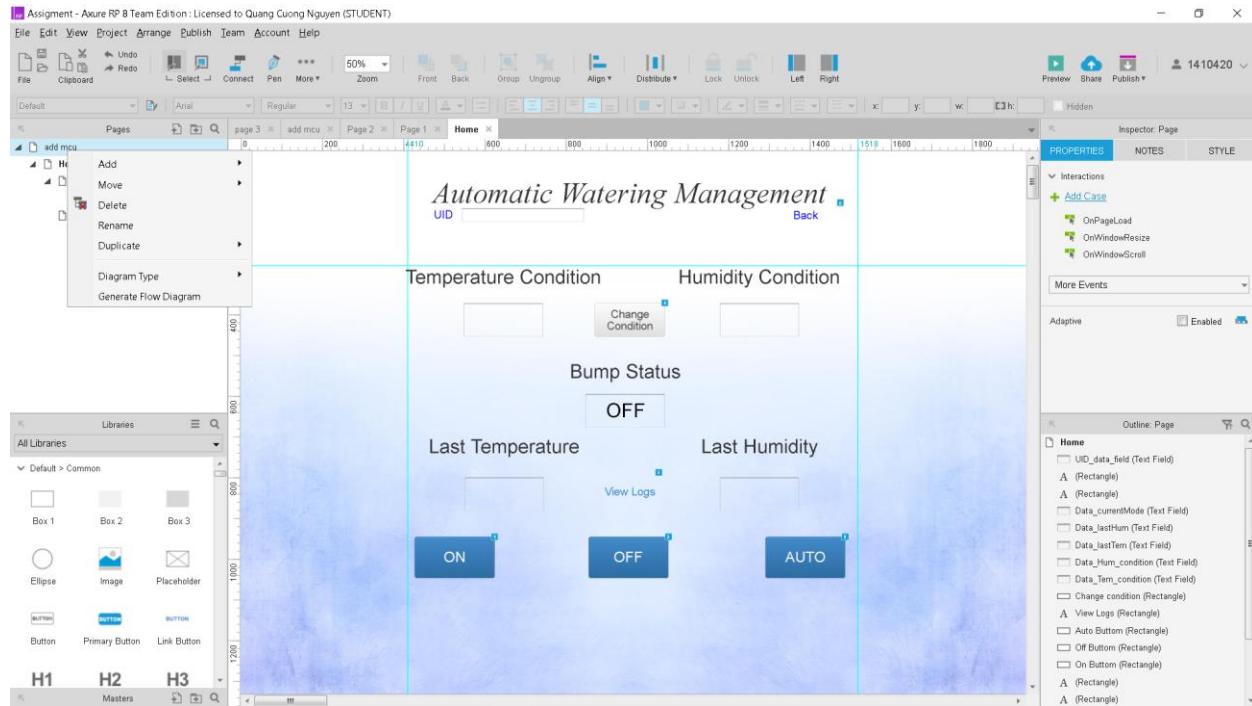
- Example: some action when you click on the button/link



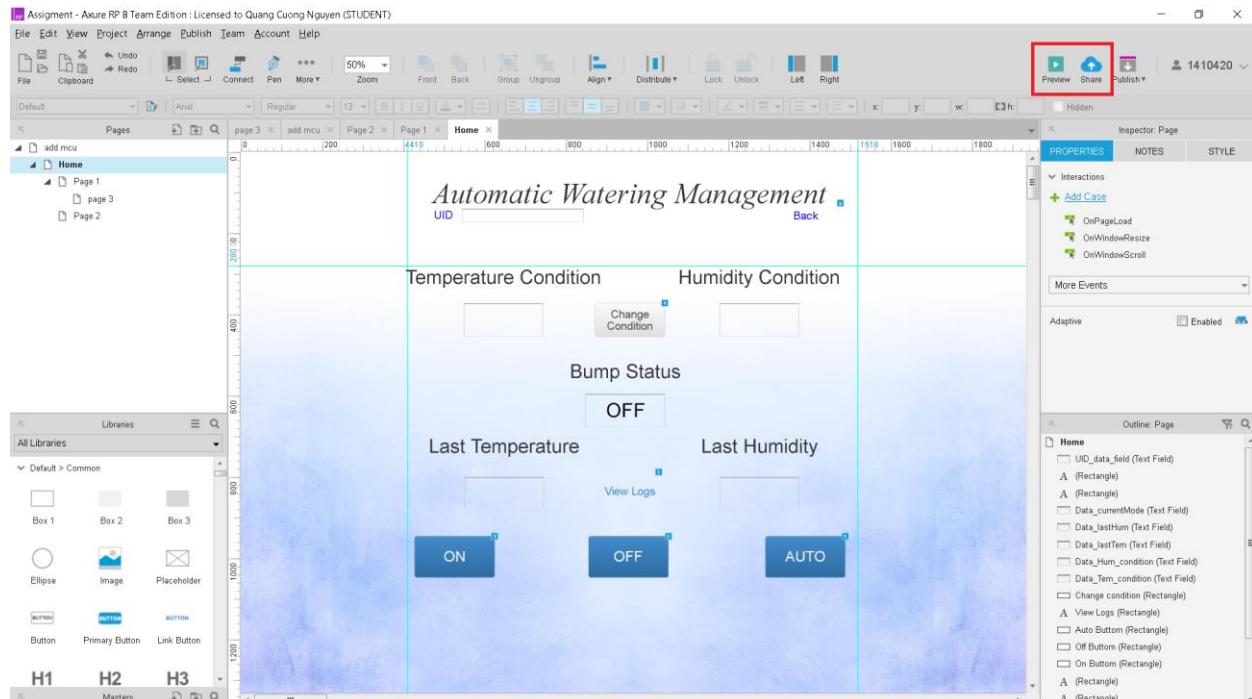
- Edit style
- Edit page



- Add page/folder



- Right click to edit page
- Preview and share



- Click to preview or share

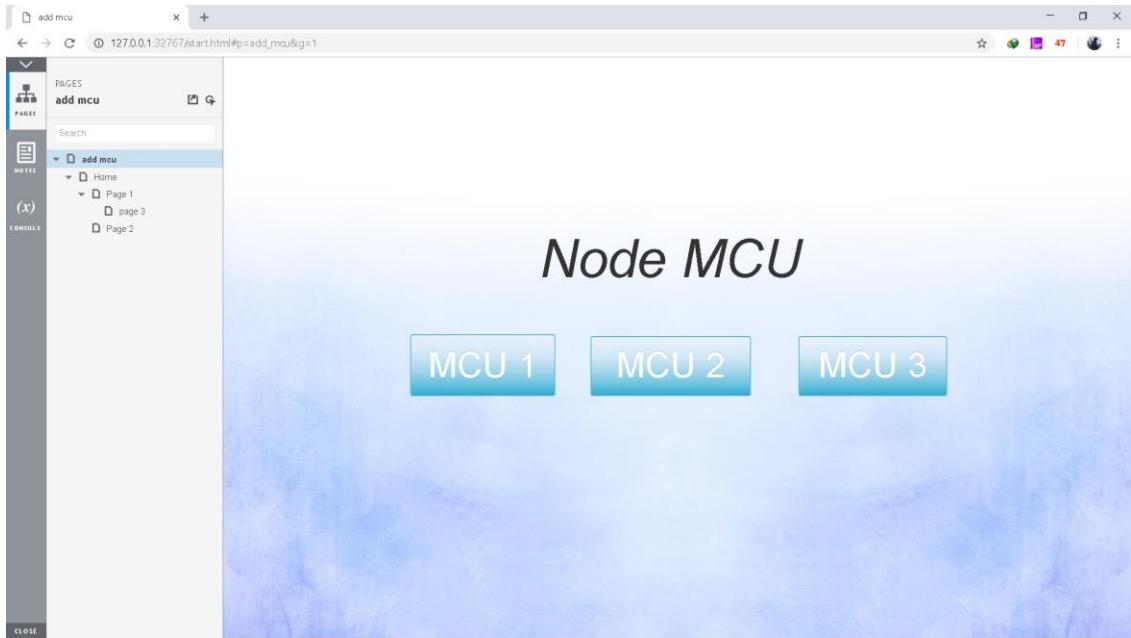
The image displays two screenshots of a web-based application interface for "Automatic Watering Management".

Main Dashboard (Top Screenshot):

- Header:** Automatic Watering Management, Back, UID [input field]
- Left Sidebar:** PAGES, Home, Search, add mcu, Home, Page 1, page 3, Page 2.
- Content Area:**
 - Temperature Condition:** [input field]
 - Humidity Condition:** [input field] Change Condition
 - Bump Status:** OFF
 - Last Temperature:** [input field]
 - Last Humidity:** [input field]

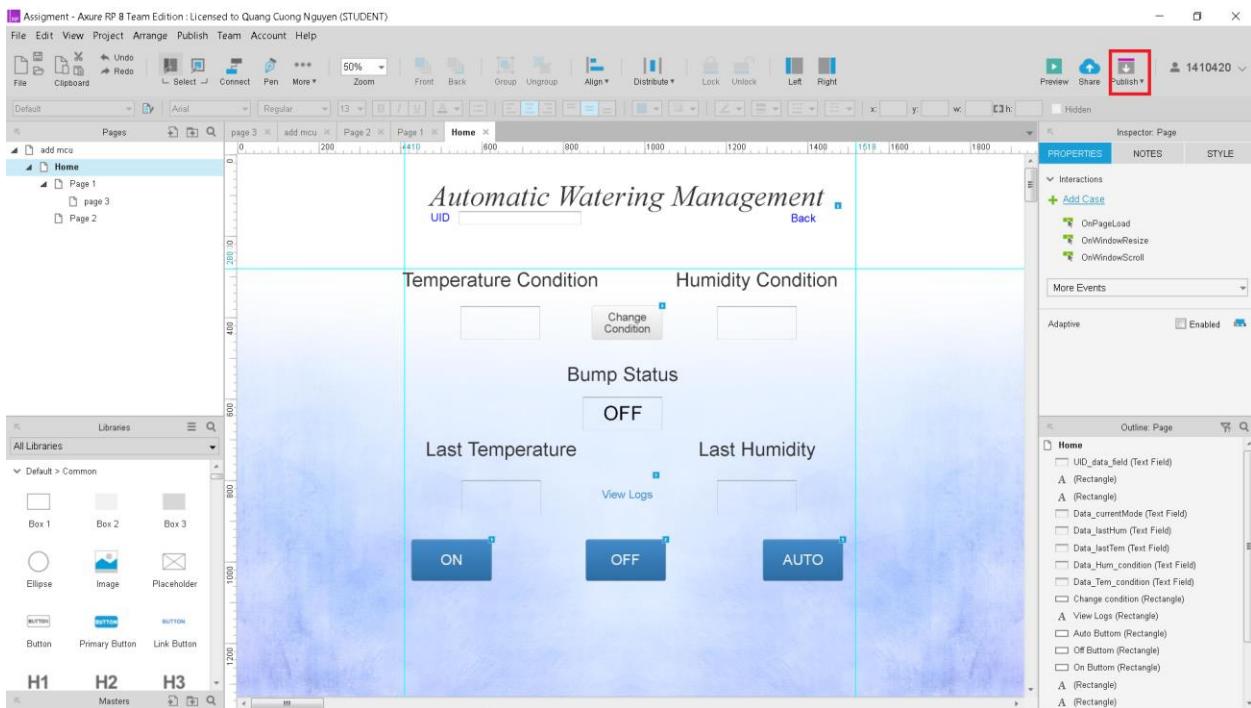
Modal Dialog (Bottom Screenshot):

- Header:** Page 1, 127.0.0.1:32767/start.html?p=page_1&g=1
- Left Sidebar:** PAGES, Page 1, Search, add mcu, add Home, Page 1, page 3, Page 2.
- Content Area:**
 - Temperature:** [input field]
 - Humidity:** [input field]
 - Buttons:** OK (green), Cancel (green)

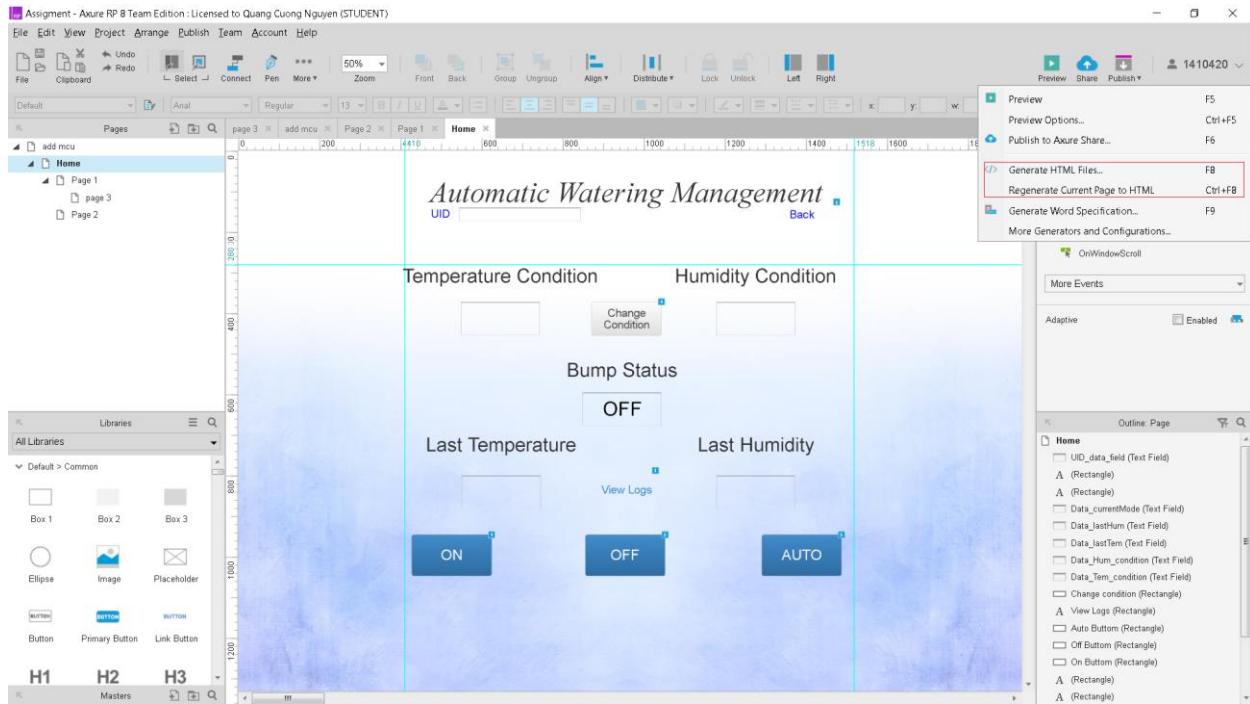


Example of preview

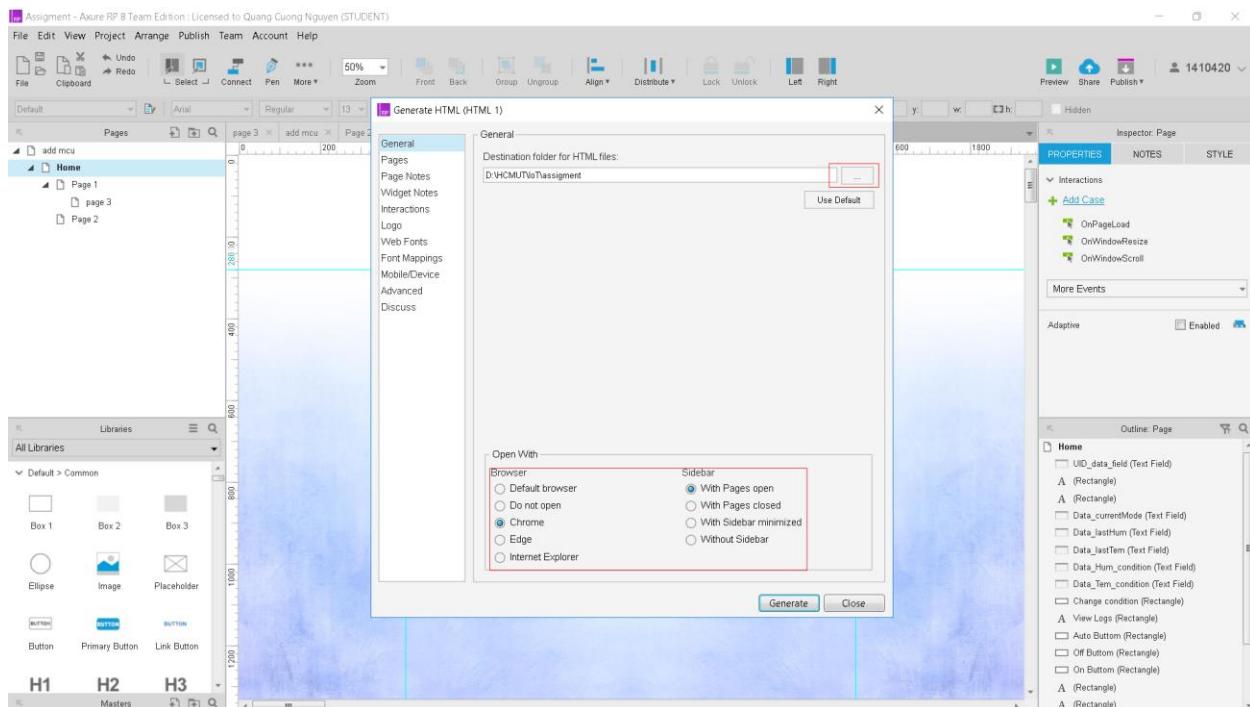
- Convert to html code



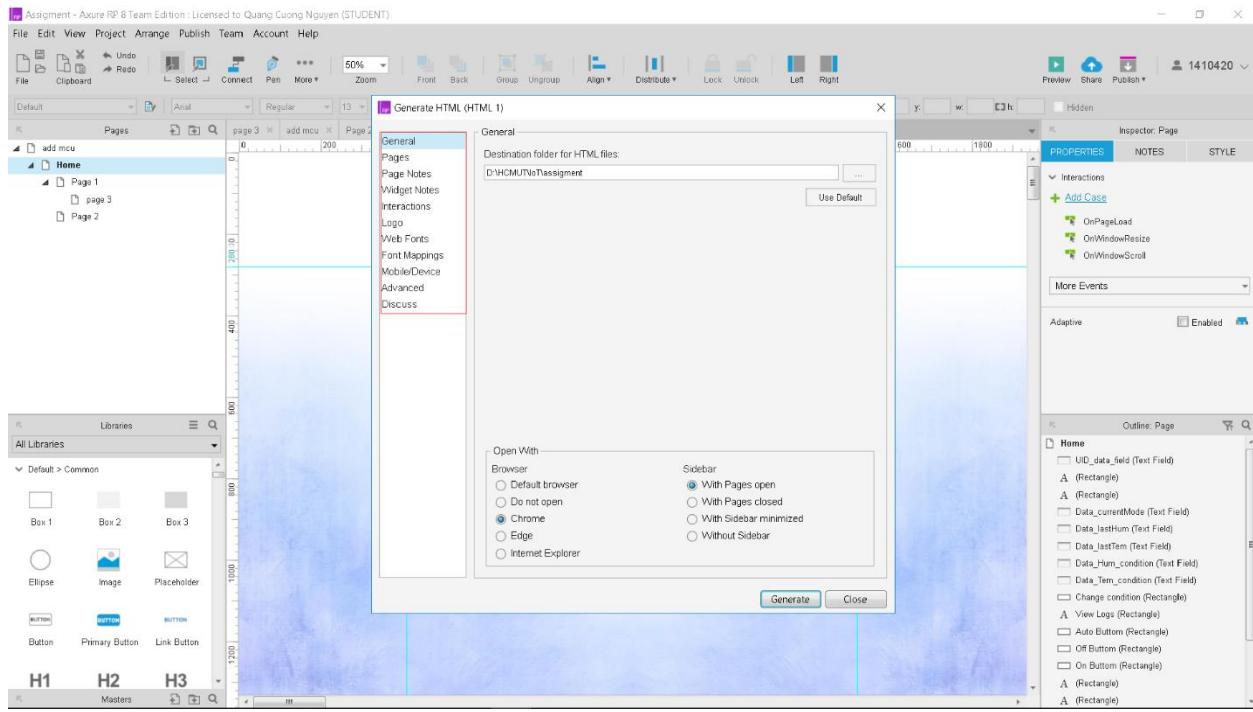
- Left click on Publish



- F8 for convert all pages, Ctrl+F8 for convert current page



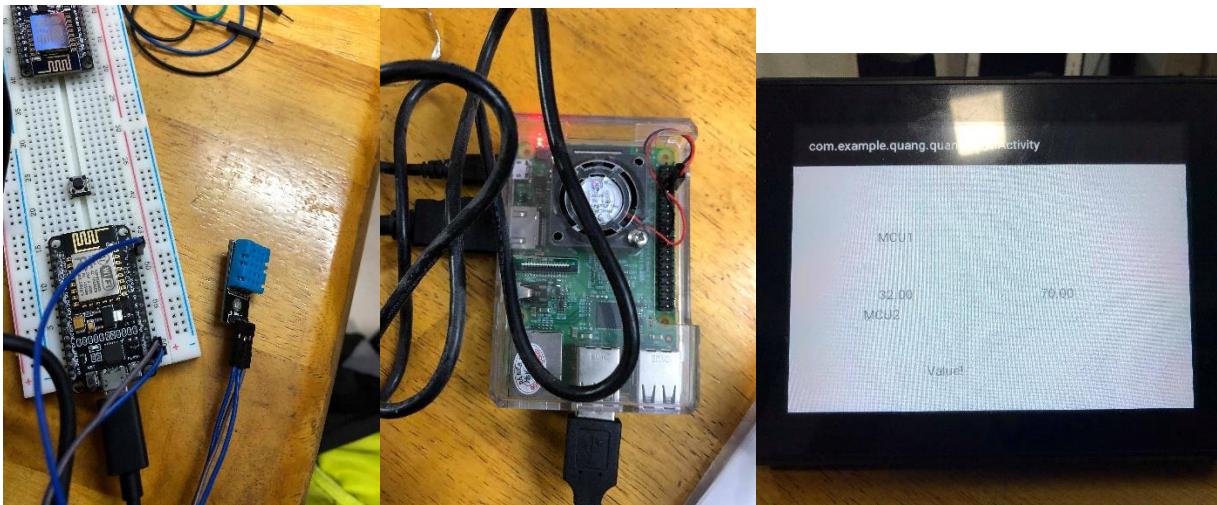
- Click on ‘...’ to choose place to save, edit some features at ‘open with’ box



- Edit more features by click on left column and click on Generate to finish

IV. DEMO

1. Hardware



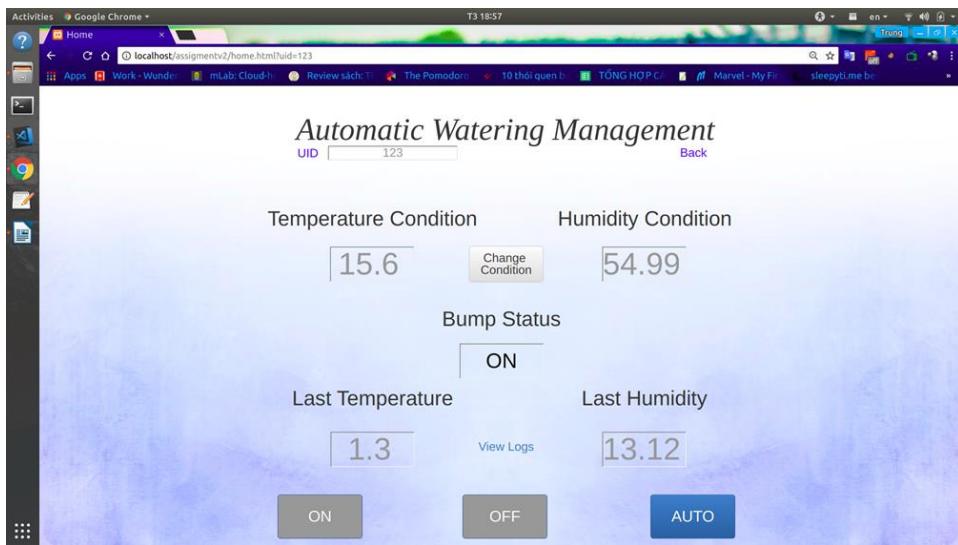
DHT-11 connects to node MCU esp8266.

Raspberry Pi runs Android Things.

Small UI for checking the data.

2. Server

In home screen with example mcu:

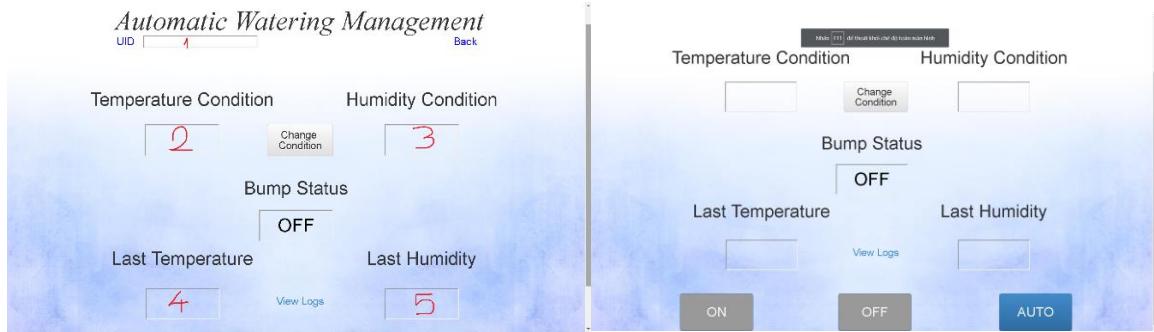


View logs example

Date/Time	Temparature	Humidity
18:17	1.3	13.12
15:48	1.3	13.12
15:48	1.3	12.12
15:20	13	50
15:5	30	60

And all buttons and functions in pages are working correctly.

3. Website UI



- UID (1) is ID of the MCU.
- Temperature condition (2) is number that the bump will automatically spray if temperature of land risk to.
- Humidity condition (3) is number that bump will automatically spray if humidity of land fall to.
- Last temperature (4) is temperature of land at the last time when DHT-11 got.
- Last humidity (5) is humidity of land at the last time when DHT-11 got.
- Bump status is the status of the bump (On/Off) at current.
- Change condition button, use for edit temperature condition and humidity condition. With this button, the user can edit temperature and humidity.
- View logs link, use to open the view logs page where the monthly data save at.
- On/Off/Auto button, these button use for change bump status, on/off button use to open/close the bump immediately. Auto button will control the bump base on humidity condition and temperature condition.