**Smart Mirror:**

**• Widget Generation**

function mw\_\_WIDGET\_\_\_GEN() {

    let html = `

    <div class="mw\_time mws">

        <iframe src="/widget/WIDGET\_NAME/WIDGET.HTML ">

    </div>`;

    return html;

}

**mw\_WIDGET\_GEN()**

This is a generalized function that is used to create + add all the widgets to the mirror.

The **WIDGET** keyword on the codesnippet will be replaced with the real name of the widget.

**For example:**

function mw\_time\_\_\_GEN() {

    let html = `

    <div class="mw\_time mws">

        <iframe src="/widget/time/clock.html">

    </div>`;

    return html;

}

This is the function for the time widget.

**• Back end controller**

onScreen.on('value', function(data) {

    var v = data.val();

    let mw\_left = v.mw\_left;

    let mw\_right = v.mw\_right;

    $('.mw\_shell').html('');

    for (var i in mw\_left) {

        if (mw\_left[i]) {

            console.log(`${i}\_\_\_GEN()`);

            $('.mw\_left').append(eval(`${i}\_\_\_GEN()`));

        }

    }

    for (var i in mw\_right) {

        if (mw\_right[i]) {

            console.log(`${i}\_\_\_GEN()`);

            $('.mw\_right').append(eval(`${i}\_\_\_GEN()`));

        }

    }

});

This function basically checks if there is any change on the mirror from the app section and then updates the iframes with the newly added widgets

**Home Automation:**

**On NodeMCU**

**• The main controller in the Arduino**

server.on("/APPLIANCE/1/state", []() {

digitalWrite(APPLIANCE PIN, STATE);

Serial.println("APPLIANCE #1 is OFF");

server.send(200, "text/plain", " APPLIANCE #1 OFF");

});

This function listens to certain requests from our **Speculo web app**

Digital Write takes 2 arguments, which is the PIN and the STATE of that device

For example this is to turn on the **first light** in a specific room

server.on("/light/1/off", []() {

digitalWrite(16, LOW);

Serial.println("Light #1 is OFF");

server.send(200, "text/plain", "Light #1 OFF");

});

**• Showing the status of the AC:**

server.on("/ac/1/on", []() {

lcd.setCursor(0, 1);

// ac\_stat = "1"

lcd.print("AC:ON ");

Serial.println("AC is ON");

server.send(200, "text/plain", "AC");

});

**• Showing IP Address**

IP = WiFi.localIP().toString();

int sumDot = 0; int x;

for (x = 0; x < IP.length(); x++) {

if (IP[x] == '.') {

sumDot++;

if (sumDot == 3) {

break;

}

}

};

String host\_IP = "";

for (x + 1; x < IP.length(); x++) {

host\_IP += IP[x];

}

**On The Web App [MAIN]**

**MIRROR SECTION**

**• Generating mirror Widget icons**

function mw\_option\_constructor(id) {

    $(`#${id}`).slideUp();

    let i = $(`#${id}`).attr('data-icon');

    let name = $(`#${id}`).attr('data-name');

    let html = `

        <div id="mw\_${id}" data-id="${id}" data-icon="fa fa-list" class="mw\_options">

            <i class="${i}"></i>

            <span>${name}</span>

        </div>

    `;

    return html;

}

• **Adding new widgets to the mirror through the app**

$(document).ready(function() {

    onScreen.once('value', function(data) {

        var \_o = data.val();

        console.log(\_o);

        var mw\_left = \_o.mw\_left;

        var mw\_right = \_o.mw\_right;

        for (var i in mw\_right) {

            if (mw\_right[i]) {

                $(`#${i}`).hide();

                $('#mw\_right\_list').append(mw\_list\_constructor(i));

            }

        }

        for (var i in mw\_left) {

            if (mw\_left[i]) {

                $(`#${i}`).hide();

                $('#mw\_left\_list').append(mw\_list\_constructor(i));

            }

        }

    });

});

**• Get temperature data from NodeMCU**

function handleTempData() {

    request(`${IP\_OF\_NODEMCU}/getTemprature`, { json: true }, (err, res, body) => {

        if (err) {

            return console.log(err);

        }

        let tempObj = {

            temprature: body,

            time: new Date().getTime()

        };

        console.log(tempObj);

        fs.readFile('./datastore/db.json', 'utf8', function(err, data) {

            let obj = JSON.parse(data);

            obj['rijinmk']['rooms']['bedroom']['temperature'].push(tempObj);

            fs.writeFileSync('./datastore/db.json', JSON.stringify(obj));

        });

    });

}

setInterval(handleTempData, 1000 \* 60 \* 30);

**• Controlling Status of devices connected to NodeMCU from app**

// TURN ON AND OFF THE APPLIANCES (CHECKBOX)

$('.switch input[type=checkbox]').on('change', function() {

    var stat = $(this).is(':checked') ? 'on' : 'off';

    var link = $(this).attr('data-href');

    $.get(IP\_OF\_NODEMCU + link + stat);

    // database.ref('/WANCommand').set({ command: link + stat });

    console.log(IP\_OF\_NODEMCU + link + stat);

});

**• Scheduling data on the AC**

// SET AC TIME

$('#set\_ac\_time').click(function() {

    var startTime = $('#start\_time').val();

    var endTime = $('#end\_time').val();

    var URL = IP\_OF\_NODEMCU + '/ac/1/sch?time=' + startTime + '-' + endTime;

    // database.ref('/WANCommand').set({ command: '/ac/1/sch?time=' + startTime + '-' + endTime });

    console.log(URL);

    $.get(URL);

});