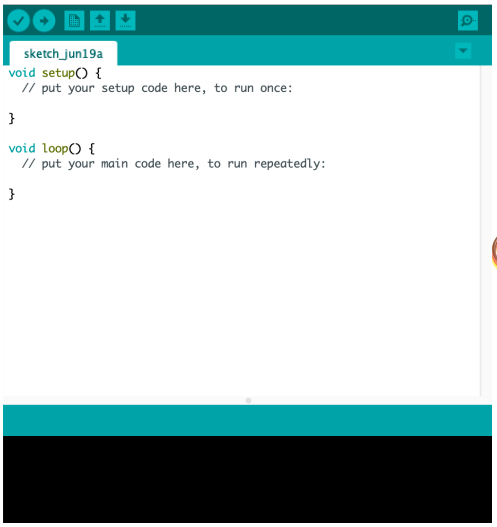
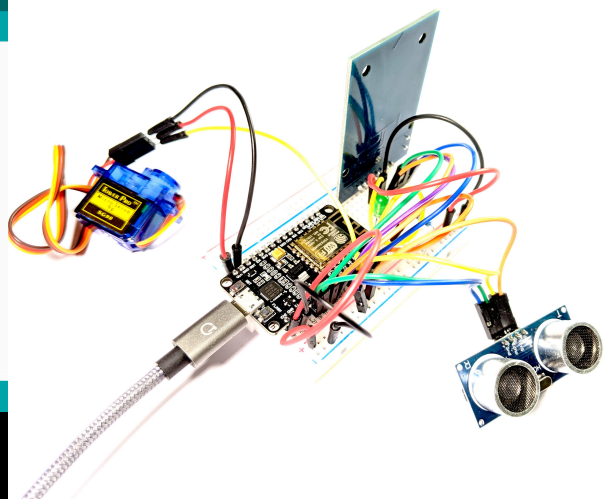
Smart Door

Part seven: The RDIF Card Reader

Project description:

In this project, you will build on the previous smart door exercise and add one more component to your smart door circuit. You will use the RDIF card reader component as one additional way control the door opening and closing mechanism. This is a continuation of the sixth part of the smart door exercise so please refer back to that part if you get lost.



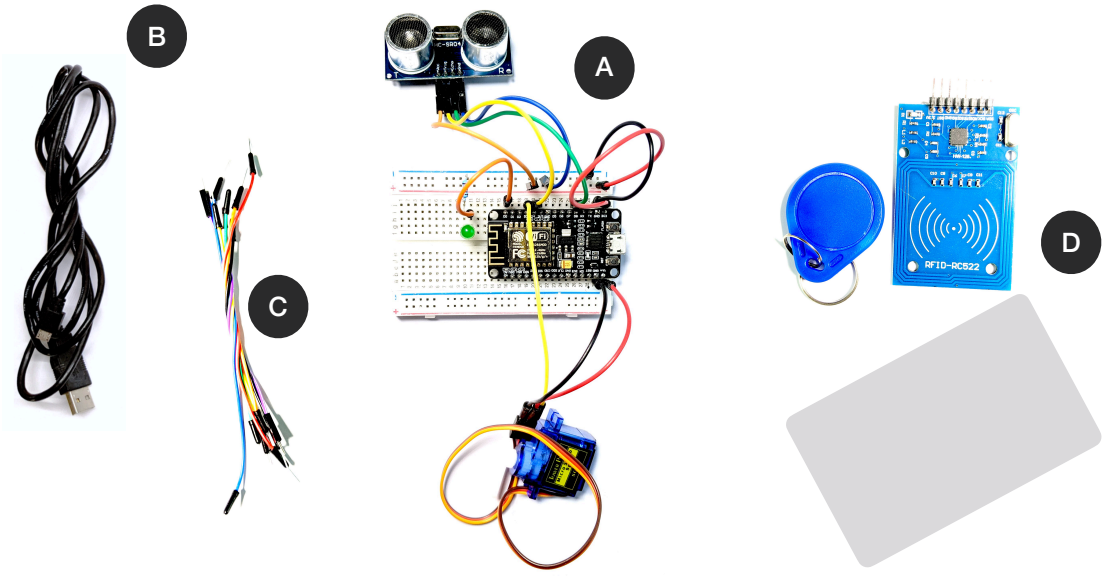


Project objectives:

- Get familiar with the RDIF card reader component
- Add the sensor to the smart door circuit
- Use the sensor to detect a valid card access

Project components:

Component Reference	Component Quantity	Component Name	Component Description
A	1	Smart Door Part 6	The smart door circuit which features the distance sensor, the green led, and the servo motor components.
B	1	Micro USB Cable	A USB cable to power and upload instructions to a microcontroller
C	7	Jumper Wires	Conductive cables frequently used with a breadboard to connect two points in a circuit
D	1	RC522 RFID Module	A radio-frequency identification module which uses electromagnetic fields to identify tags attached to objects.



Step One
Introduction and Theory

In the sixth part of the smart door exercise, you experimented with automation and made the servo motor component react to the distance sensor. Such an automation, allowed you to automatically control the door opening and closing mechanism. This exercise introduces you to one new component: the RC522 RFID module.

As a process to exploring the new component, you will build on the latest version of the smart door circuit. The idea here is to add and alternative way to control the door opening and closing mechanism using a radio-frequency identification module.

You will need both the circuit and the code for the smart door part six exercise to complete this project.

Let's briefly discuss the additional component for the circuit:

- **RC522 RFID module:** The RC522 module **RFID** consists of two main components, a tag attached to an object to be identified, and a reader which identifies the attached tag. Tags usually do not have a battery and are powered by an electromagnetic field when they come is close proximity with the reader. This module consists of 8 leads: the GND (ground), the VCC (3.3V), the RST (Input for reset), SDA (Serial Data), and the SCK (Serial Clock), MOSI (Master Out Slave In), MISO (Master In Slave Out), and IRQ (Alert signal for proximity).

The RC522 is essentially a module that allows you to read the information encoded on a tag/card when such objects are placed in close proximity to the reader. The reader then generates an electromagnetic field which powers up the tag/card. Finally, the latter will respond by sending its stored information back to the reader in the form of another radio signal.

Step Two

Adding the component to the Smart Door Circuit

The circuit assembly for this exercise requires you to wire up one RC522 RFID module component.

You will add the RC522 component to your existing smart door circuit that you have built so far. Refer back to the sixth part of the smart door exercise to view the latest version of the circuit.

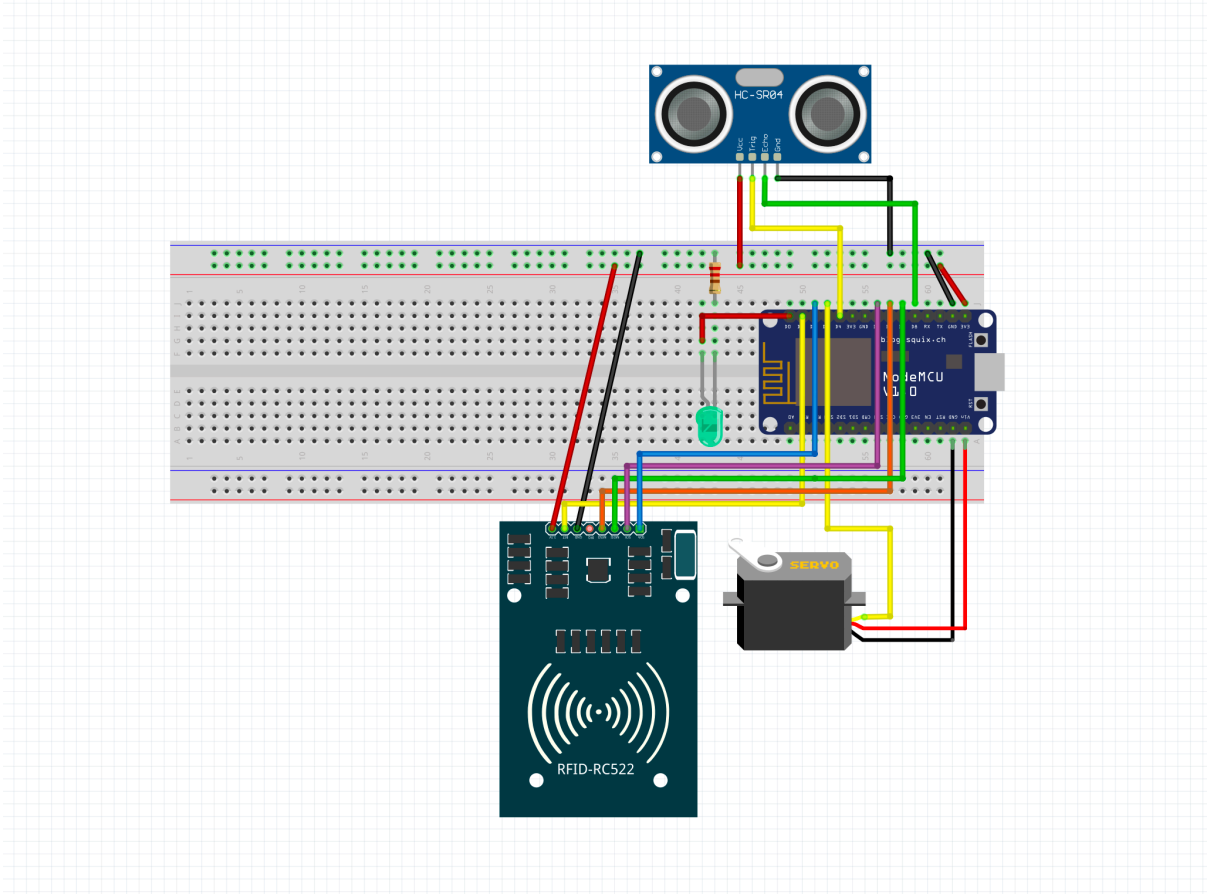
Connect the SDA and SCK leads respectively to digital pins D2 and D5 of your ESP board. The MOSI and MISO leads should be connected respectively to digital pins D7 and D6. The RST lead should be connected to digital spin D1. Finally, you should also connect the VCC lead to source (3.3V) and the GND lead to ground.

Here a quick recap of the circuit wiring:

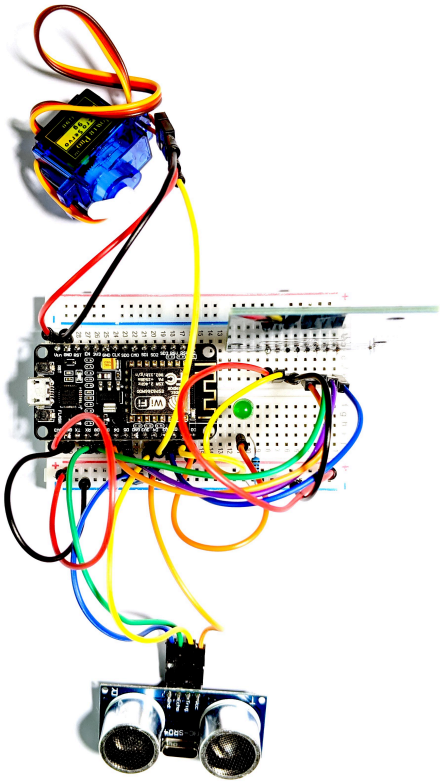
- **RC522:** The SDA and SCK leads should be connected to D2 and D5. The MOSI and MISO leads should be connected to D7 and D6. The RST lead should be connected to digital spin D1. The GND lead goes to ground and the VCC lead goes to source (3.3V).

It is now your turn to assembly the circuit. This exercise uses the latest version of the smart door circuit as a starting point so please refer back to the smart door part 6 exercise. Also, use different colour wires for the connections if you do not have wires with the same colours left in your kit.

The diagram below shows you how the circuit should be assembled. It combines the circuit components for the sixth part of the smart door circuit with the additional RC522 module component:



Furthermore, see below a picture of the circuit assembled:



Step Three
Writing the Code

Now that you have assembled the circuit, it is time to add the code for the RC522 module component. Go ahead and create a new empty sketch from the Arduino IDE.

You should see an empty sketch like the following:

Feel free to save the sketch and rename it to something sensible:
smart_door_part7 for instance.

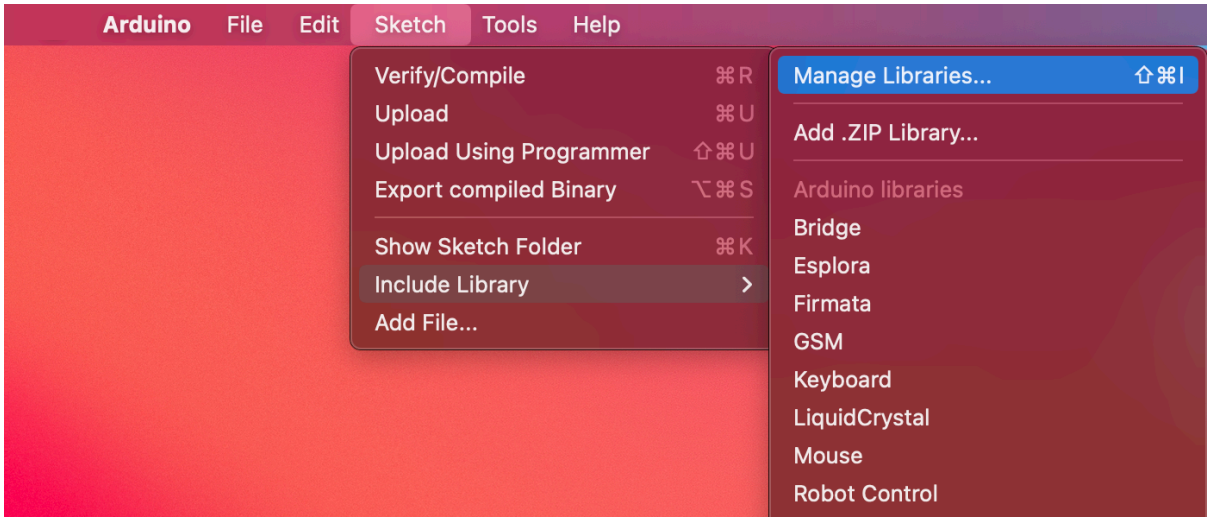
Now copy and paste the code that you wrote for the sixth part of the smart door exercise. You should have the code with the **ledControl(),** the **distanceCentimeter(),** the **jsonDistanceSensor(),** the **openDoor()**, the **setMinDistance()**, and the **get_json()** utility functions. You should also have an additional function **get_index()** and the web server functionality for the smart door dashboard.

There is one core functionality that we want to add to the program here:

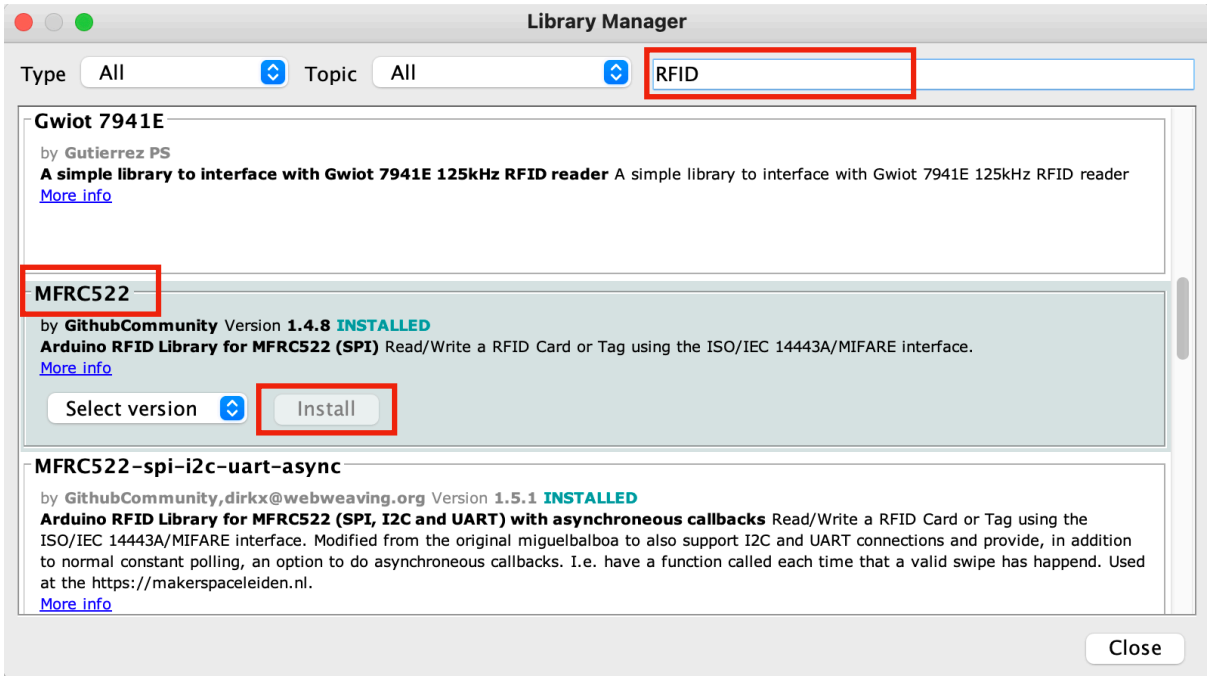
- Use the RC522 module to detect the card/tag ID and use it for authentication. You will then use the authentication credentials to open the door as part of this exercise additional tasks.

Let's begin by adding the **MFRC522** library. Such library will facilitate the MFRC522 RFID component control.

With your Arduino sketch opened, click on **Sketch -> Include Library -> Manage Libraries...** like shown below:



Next, search for "RFID" and install the latest version of the **MFRC522** library as shown below:



Now that the library has been installed, let's include it in the sketch.
Add the following two lines of code just below the Servo library (Servo.h):

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ArduinoJson.h>
#include <Servo.h>
#include <SPI.h>
#include <MFRC522.h>
```

Let's now initialise some new variables and the the MFRC22 component just before the setup function:

```
// The RFID reset and data pins
const int RST_PIN = D1;
const int SS_PIN = D2;

// String to store the card identifier
// bool variable to check for authentication
String cardId = "";
bool authenticated = false;

// Create MFRC522 instance with data and reset pins
MFRC522 mfrc522(SS_PIN, RST_PIN);
```

Type the above code just before the setup() function.

cardId is a variable to store the unique card identifier and **authenticated** is a variable to check for authentication (whether such cardId is valid or not).

Here we also initialise the **MFRC22** object using the MFRC22 library. The constructor takes two arguments: the reference to the SDA pin (**RST_PIN**) and the reference to the RST pin (**SS_PIN**).

Next we want to add some code to the setup() function:

```
// Init SPI bus and MFRC522
SPI.begin();
mfrc522.PCD_Init();

// Optional delay. Some board do need more time after init to be ready
delay(4);
```

Type the above code at the end, but inside, the setup function.

SPI.begin() initialises the bus for detecting the component
mfrc522.PCD_Init() initialises the RFID component
delay(4) is a tiny delay to make sure that the component is ready

At this point, we have configured the MFRC22 component and initialised it. We can now use some additional utility functions to first retrieve the unique identifier of our white card/tag and successively use it for authentication.

The **getCardId()** function:

```
// Obtain the card ID from the card reader
String getCardId(){

  // Connstruct the card ID String
  if (mfrc522.PICC_ReadCardSerial()) {
    String cardIdDetected;
    for (byte i = 0; i < 4; i++) {
      cardIdDetected+= mfrc522.uid.uidByte[i];
    }

    // Print the card ID and retur its value
    Serial.println(cardIdDetected);
    return cardIdDetected;
  }
}
```

Type the above code at the end of the sketch, together with the already existing utility functions.

The **getCardId()** utility function allows you to retrieve the unique identifier of your card/tag when you put it in proximity of the receiver. It first checks that a card/tag has been detected (**mfrc522.PICC_ReadCardSerial**). If a card/tag was detected, the function creates a new string and assigns to it the value retrieved from the receiver. This is done inside a for loop (4 times) as the receiver sends four bytes of information containing the unique identifier. The unique identifier is then printed on the Serial Monitor and returned from the function as a String.

Now that the **getCardId()** utility function in place, it is time to call such function in the **loop()** function and check the Serial Monitor for the card/tag identifier.

The **loop()** function:

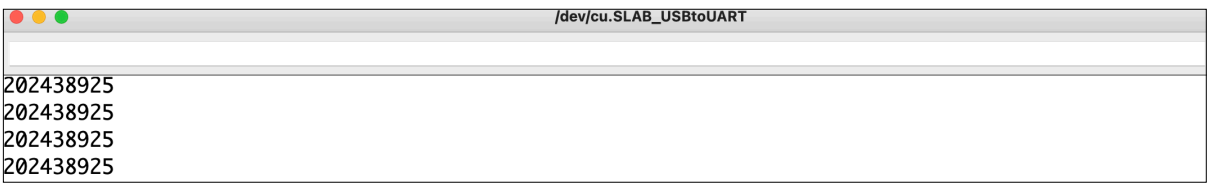
```
// put your main code here, to run repeatedly:
void loop() {

  // This will keep the server and serial monitor available
  //Serial.println("Server is running");

  //Handling of incoming client requests
  server.handleClient();

  // Prints the distance on the Serial Monitor
  distanceCentimeter();

  // Call the ledControl function
  // The function should light up the led according to the ultrosound distance
  ledControl();

  // Call the openDoor function to control the door
  openDoor();

  // Reset the loop if no new card present on the sensor/reader.
  // This saves the entire process when idle.
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }

  // Call the function to obtain the card unique ID
  cardId = getCardId();
}
```

Add the call to the **getCardId()** function inside the loop() function as shown above and assign its returned value to the variable **cardID**.

The ‘if’ statement checks if there is a card/tag present. If not, it resets the **loop ()** function and saves the call to the **getCardId()** function.

Hurray, you have now completed the code to detect your card/tag ID.
Go ahead, compile your code, and upload it to your ESP board.

Open the Serial Monitor, place the white card in from of the MFRC22 receiver, and make a note of the string ID printed out. You should see something like the image below (comment out other serial prints if needed):



Great, you now have your card unique identifier.
In my case it was **202438925**. You can now use it to verify your authentication when the same card/tag is positioned in the proximity of the receiver.

Let's write another utility function to check for valid authentication.

The **autenticated()** function:

```
// Compare the cardId with the detected ID
void autenticated(String id){

  if(id == "202438925"){ // Authenticated
    authenticated = true;
    Serial.print("User authenticated");
  }else{
    authenticated = false; // Refused
    Serial.print("User not authenticated");
  }

}
```

The **autenticated()** utility function simply compares the retrieved ID from the card reader with the unique identifier of such card. If the two matches, authentication is successful and "User authenticated" is printed on the Serial Monitor.

Note: change "202438925" with your card/tag unique identifier.

Finally, let's add the authentication function logic to the **loop()** function:

```
// Call the function to obtain the card unique ID
cardId = getCardId();

// Use the ID to authenticate
autenticated(cardId);
```

Add the **authenticated(cardId)** function call just below the **getCardId()** function call but still inside the **loop()** function.

Hurray, you have now completed the code to detect your card/tag ID and use it for authentication. Go ahead, compile your code, and upload it to your ESP board.

Open the Serial Monitor, place the white card in from of the MFRC22 receiver, and check for the “User authenticated” message.

Now that you can correctly authenticate yourself using the card/tag, you can automate the door to open/close when such authentication happens. You can explore this functionality as part of the additional tasks for this exercise.

Step Four
Additional Tasks

Your task is to use the new RFID component to automate the door opening and closing mechanism. You should have the code so that you can configure it in two modes: “distance” or “rfid”. This will allow you to specify the way in which you would like the door to be automated, either via the distance sensor or via the RFID sensor. You should modify the code inside the **openDoor()** utility function so that it first checks the automation mode, and then automates the door opening and closing mechanism accordingly. The “rfid” mode should use the **authenticated** variable to either open or close the door.

You are expected to attempt the following:

- Add a String global “doorMode” variable to keep track of the automation status: either “distance” or “rfid”.
- Change the “doorStatus” variable name to “doorStatusDistance” and add one additional String global variable and name it “doorStatusRfid”. Initialise the latter to “closed”. This will allow you to differentiate the status of the door based on the automation mode.
- Modify the code inside the **openDoor()** function with the following pseudocode:

1) Check which automation mode you are using for the door (either “distance” or “rfid”).

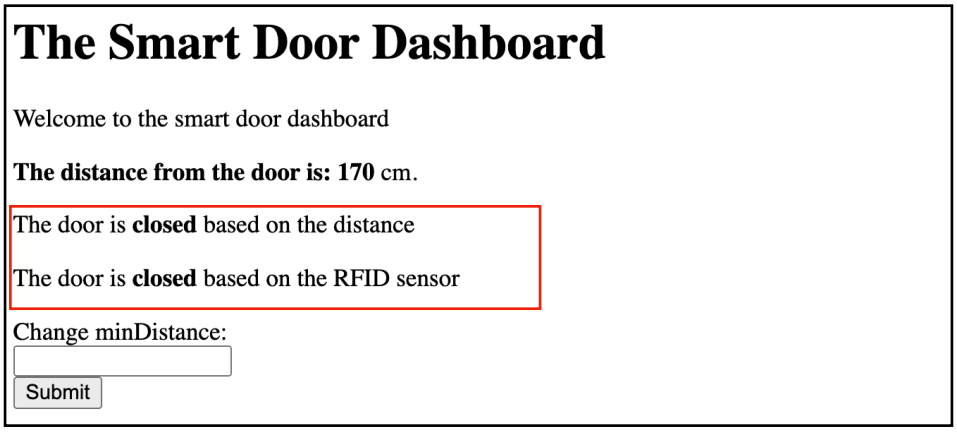
2) If the mode is “distance”, use the existing code to determine the door opening and closing mechanism. Make sure you update the name of “doorStatus” to “doorStatusDistance”.

3) If the mode is “rfid”, check if your are authenticated or not.

4) If “authenticated” is true, open the door, wait for one second, set the “doorStatusRfid” variable to “open”, and deny the authentication (authenticated = false). This will only open the door for one second after the RFID module detects the card.

5) If not “authenticated”, close the door and set the “doorStatusRfid” variable to “closed”.

- Change the **get_index()** function to replicate the following dashboard:



The dashboard should signal the door status according to the automation mode.

The above additional task determines the end of the smart door circuit project.

In the following course exercises you will learn how multiple ESP nodes (smart chair, smart fridge, smart door) can talk to each other and send/receive information.