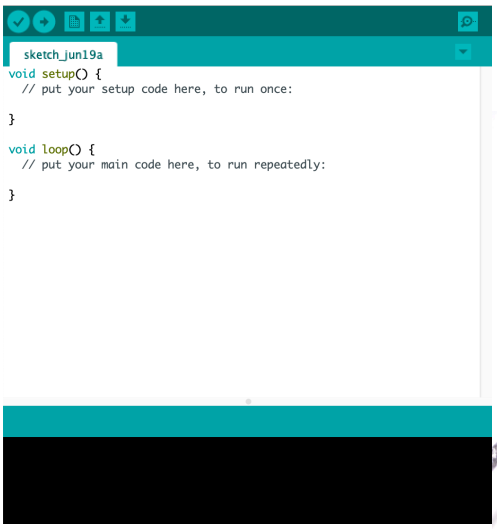
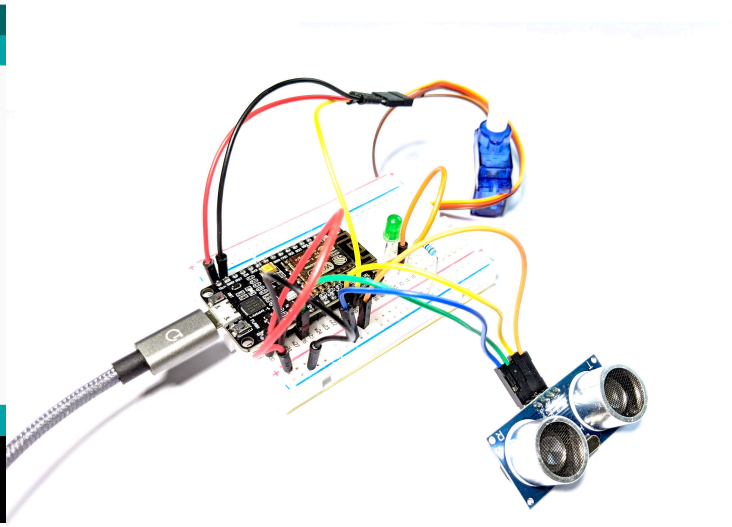
Smart Door

Part five: Adding the Servo Component

Project description:

In this project, you will put into practice some of the theory around motors and microcontrollers. You will build on the previous version of the smart door circuit, add a servo component, and understand how it operates. The circuit is of course just a simulation of a real smart door and you are not expected to create a realistic version. This part of the project will get you familiar with the servo sensor. In the next exercise, you will use the distance sensor to automate the door opening and closing mechanism.



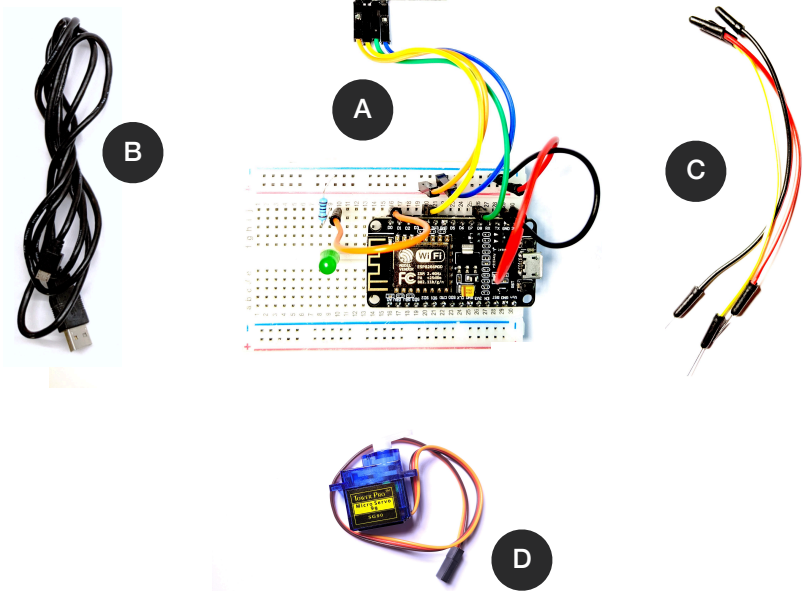


Project objectives:

- Expand your knowledge on digital and analog Input/Output pins
- Explore and write the code to control the servo component
- Use the Serial Monitor or the web server to print out the servo angle value

Project components:

Component Reference	Component Quantity	Component Name	Component Description
A	1	Smart Door Part 4	The smart door circuit which features the distance sensor, and the green led components.
B	1	Micro USB Cable	A USB cable to power and upload instructions to a microcontroller
C	3	Jumper Wires	Conductive cables frequently used with a breadboard to connect two points in a circuit
D	1	9g Micro Servo Motor	An electromechanical device that produces torque and velocity based on the supplied current and voltage.



Step One
Introduction and Theory

So far, in the smart door exercise you encountered the ultrasonic sensor component and used automation to change the brightness of a green led according to the distance from the door. In this exercise you will work with motor sensors and explore a new component: the 9g Micro Servo Motor.

As a process to exploring the new component, you will add the servo to the previous and latest version of the smart door circuit. The idea here is to create a simple loop animation to change the servo rotation angle. In the next exercise you will then automate the process with the ultrasonic sensor and open the door according to the distance value.

Let's briefly discuss the main component of the circuit:

- **9g Micro Servo Motor:** The micro servo is a self-contained electrical device used within motion control. It contains a brushless rotary electric motor which offers high torque rotation and accuracy. This component offers a rotating cycle of 180 degrees. The component has three pins: the GND (ground), the VCC (5V), and the control signal (used to control the servo rotation angle).

Step Two
Wiring up the Smart Door Circuit

The circuit assembly for this exercise requires you to wire up one 9g micro servo motor component.

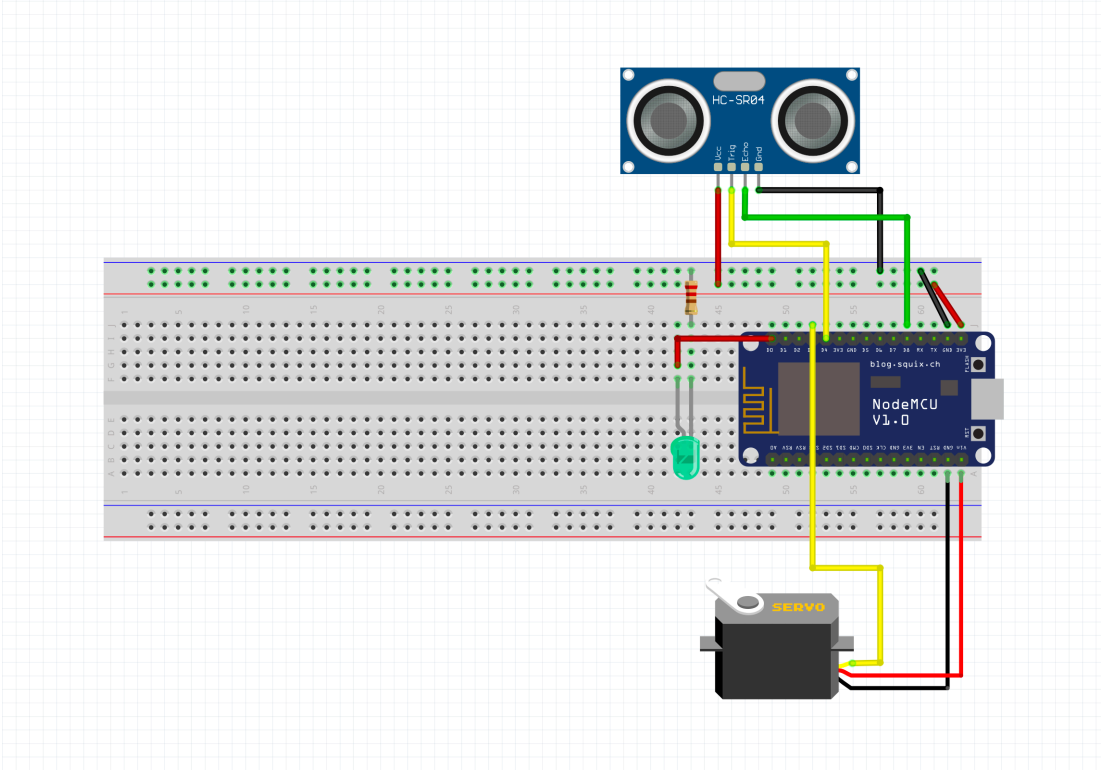
The control signal lead of the servo component should be connected to the digital pin D3 of your ESP board. The remaining two leads are GND and VCC and should be connected to ground (GND) and source (Vin) of your ESP board.

Here a quick recap of the circuit wiring:

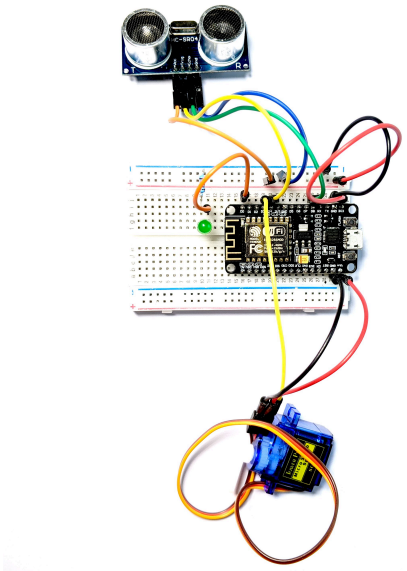
- **9g Micro Servo Motor:** The control signal lead should be wired up to pin D7. The remaining two leads should be connected to ground (GND) and source (Vin).

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 4 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:



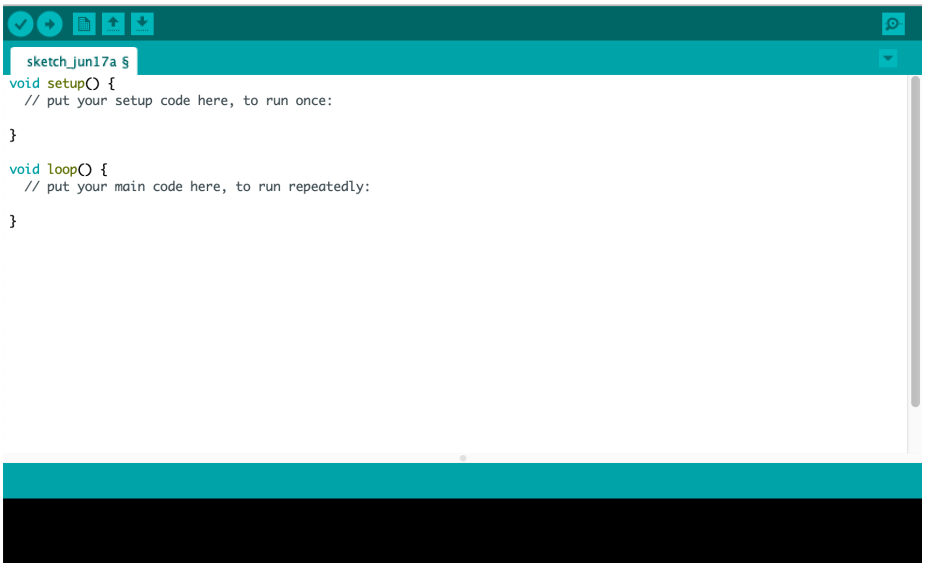
Furthermore, see below a picture of the circuit assembled:



Step Three
Writing the Code

Now that you have assembled the circuit, it is time to write the code for the smart door servo animation. 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_part5 for instance.

Now copy and paste the code that you wrote for the fourth part of the smart door exercise. You should have the code with the **ledControl(),** the **distanceCentimeter(),** the **jsonDistanceSensor(),** 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 program here:

- Create a simple animation to control the servo rotation angle

Let's begin by including the servo library in the sketch.
Add the following line of code just below the ArduinoJson library:

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

Let's now initialise the servo object just before the setup function:

```
// create servo object to control a servo
Servo myservo;
```

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

Here, we initialise the servo object using the Servo library.

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

```
// Attaches the servo on pin D3 to the servo object
myservo.attach(D3);
```

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

attach is a built in function which specifies the pin to which the servo is connected. In our case, the servo is connected to pin D3 of the ESP board.

At this point, we have configured the Servo library and initialised the servo object. Let's now add one utility function to control the servo rotation.

The **servoMovement()** utility function:

```
// Control the servo door movement
void servoMovement(){

   // Sets the servo position to 0 angle
   myservo.write(0);
   // Add small delay
   delay(500);

   // Sets the servo position to 45 angle
   myservo.write(45);
   // Add small delay
   delay(500);

   // Sets the servo position to 90 angle
   myservo.write(90);
   // Add small delay
   delay(500);

   // Sets the servo position to 180 angle
   myservo.write(180);
   // Add small delay
   delay(500);
}
```

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

The **servoMovement()** utility function creates a simple animation in rotating the servo at different angles. The **write** function is part of the Servo library and allows you to specify the rotation angle for your servo. You can pass to the function an argument value between 0 and 180 for the servo rotation. The **delay** function allows the servo to rotate to a specific angle before the next rotation angle is set.

Finally, we need to call the **servoMovement()** function inside the **loop()** function for the servo animation code to be executed.

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 servoMovement function for the servo animation
  servoMovement();

}
```

Add the **servoMovement()** function call at the end, but inside, the **loop()** function.

Hurray, you have successfully completed the code section of the circuit.
Go ahead, compile your code, and upload it to your ESP board.

Once the code has been uploaded, you should see the servo component rotating at different angles with a tiny pause in between rotations.

Step Five
Additional Tasks

In the next exercise, you will further automate the smart door circuit.

Meantime, try the following:

- Alter the servoMovement() function to create a different servo animation
- Create a global variable to store the servo rotation value
- Use either the Serial Monitor or the Web Server to display the rotation value