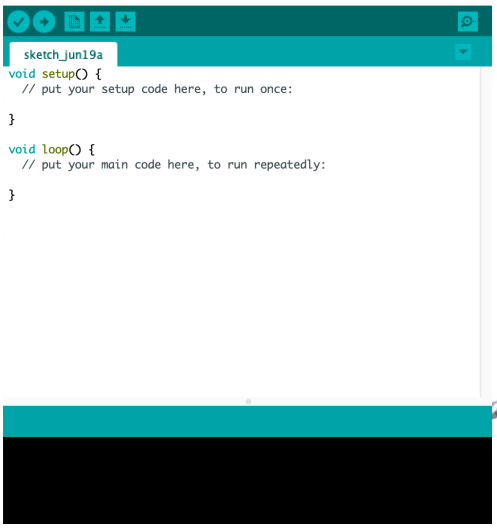
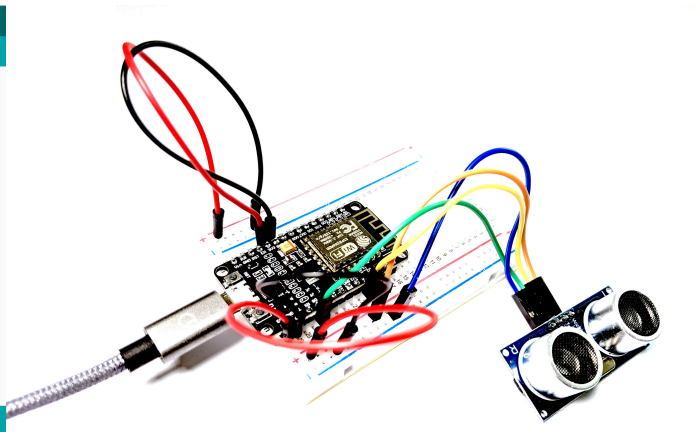
Smart Door

Part one: The circuit and first components

Project description:

In this project, you will expand your knowledge on **digital** and **analog** Input/Output pins by wiring up a circuit to simulate some of the functionalities of a smart door. For instance, the smart door will be aware of your distance from it. The circuit is of course just a simulation of real smart door and you are not expected to create the final product. This initial part of the project will get you familiar with the HC-SR04 ultrasonic sensor to measure your distance from the door.



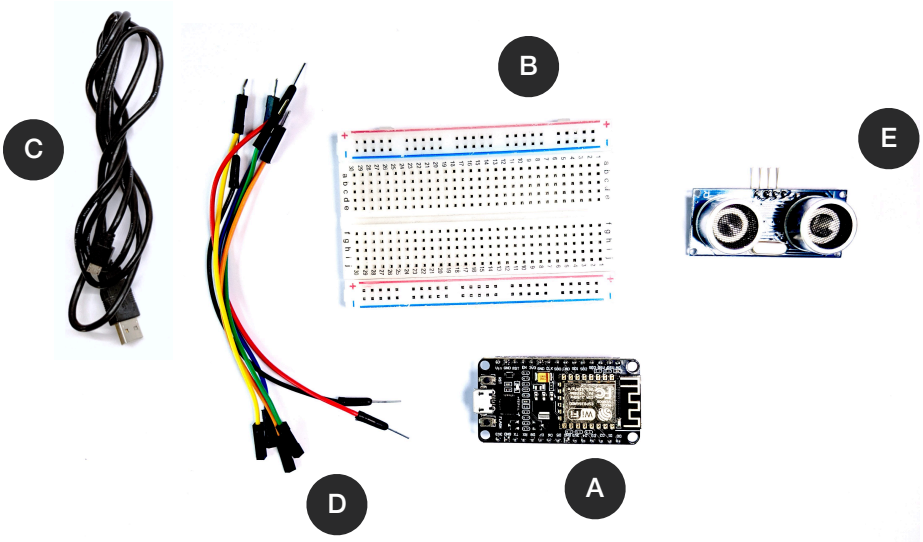


Project objectives:

- Expand your knowledge on digital and analog Input/Output pins
- Explore electric components such as the HC-SR04 ultrasonic sensor
- Use the Serial Monitor to print out your distance from the door

Project components:

Component Reference	Component Quantity	Component Name	Component Description
A	1	ESP8266 WiFi Module	A low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability
B	1	Breadboard	A rectangular plastic board with conductive rails for fast circuit prototyping. **Use the remaining half of the smart fridge circuit board for this exercise.**
C	1	Micro USB Cable	A USB cable to power and upload instructions to a microcontroller
D	6	Jumper Wires	Conductive cables frequently used with a breadboard to connect two points in a circuit
E	1	HC-SR04 ultrasonic sensor	An electrical component which uses sonar to determine the distance of an object. It offers stable readings from 2 to 400 cm



Step One
Introduction and Theory

In the smart fridge exercise you encountered some interesting components such as the switch, the potentiometer, the buzzer, the LCD screen, and the temperature and humidity sensor. In this exercise you will begin to work on a smart door circuit and explore a new component: the HC-SR04 ultrasonic sensor.

As a process to exploring the new component, you will build the first part of the circuit to simulate a smart door. The idea here is to create a simulation of a smart door which detects your distance from it. The HC-SR04 component is an electrical sensor that can be used to calculate the distance of the closest object placed in front of it.

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

- **HC-SR04 ultrasonic sensor:** The HC-SR04 emits an ultrasound at 40 000 Hz (not audible by humans) which travels through the air and if there is an object or obstacle on its path it will bounce back to the module. You can then use the travel time and speed of the sound to calculate the distance. The component has four pins: the GND (ground), the VCC (3.3V), the Trig (used to send out the ultrasound signal), the Echo (used to receive the signal).

Step Two
Wiring up the Smart Door Circuit

The circuit assembly for this exercise requires you to wire up one HC-SR04 ultrasonic sensor component.

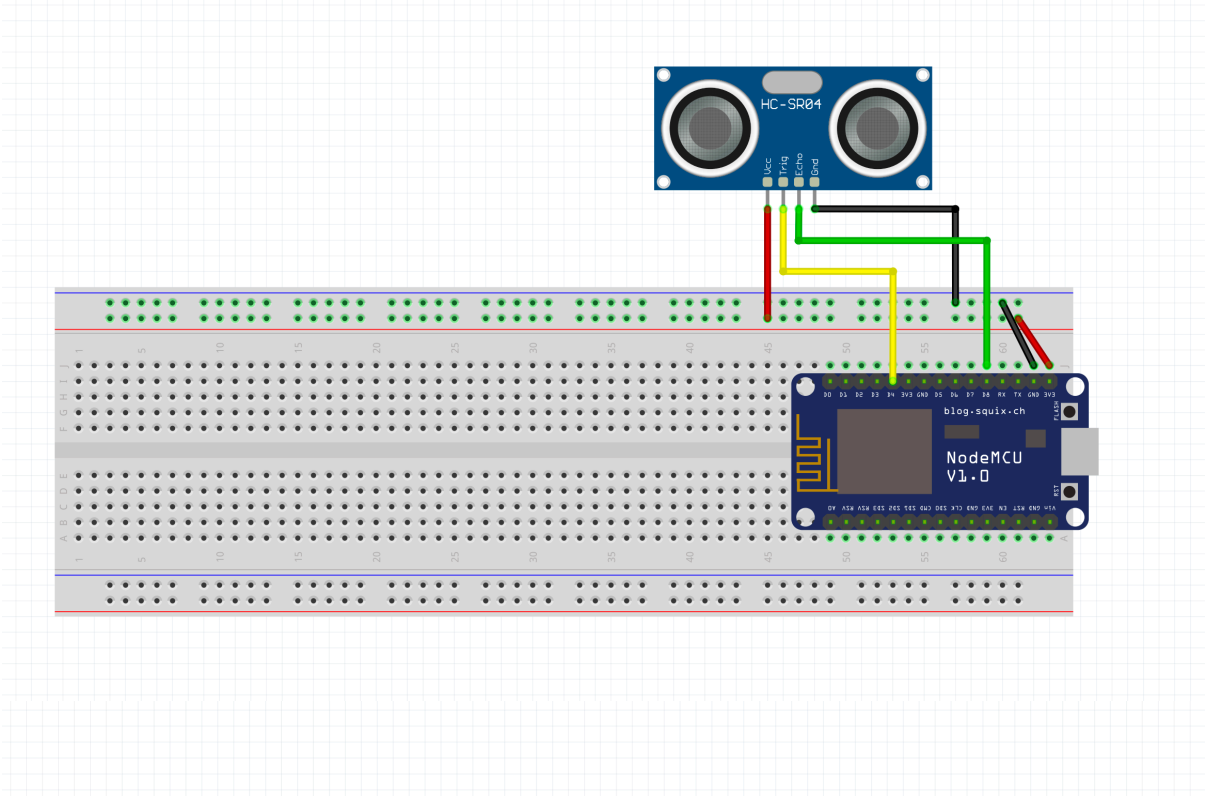
The Trig lead of the ultrasonic sensor should be connected to the digital pin D4 of your ESP board. The Echo lead of the ultrasonic sensor should be connected to digital pin D8 of your ESP board. The remaining two leads are GND and VCC and should be connected to ground (GND) and source (VCC) of your ESP board.

Here a quick recap of the circuit wiring:

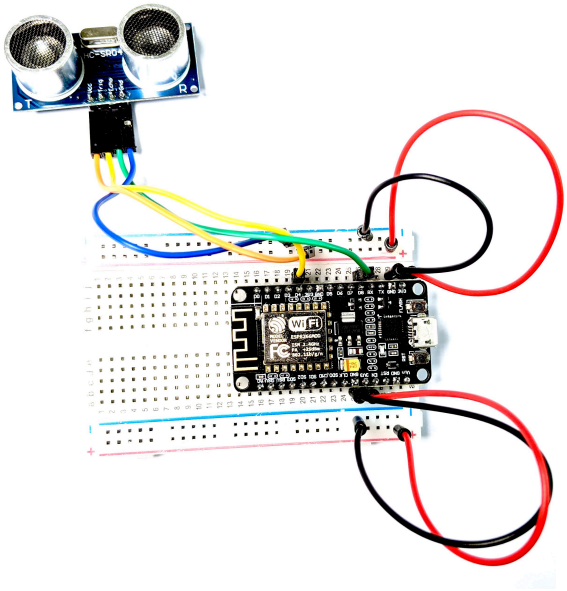
- **HC-SR04:** The Trig lead should be wired up to pin D4. The Echo lead should be wired up to pin D8. The remaining two leads should be connected to ground (GND) and source (3.3V).

It is now your turn to assembly the circuit. Please try to use the remaining side of the breadboard that you used for the smart fridge 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:



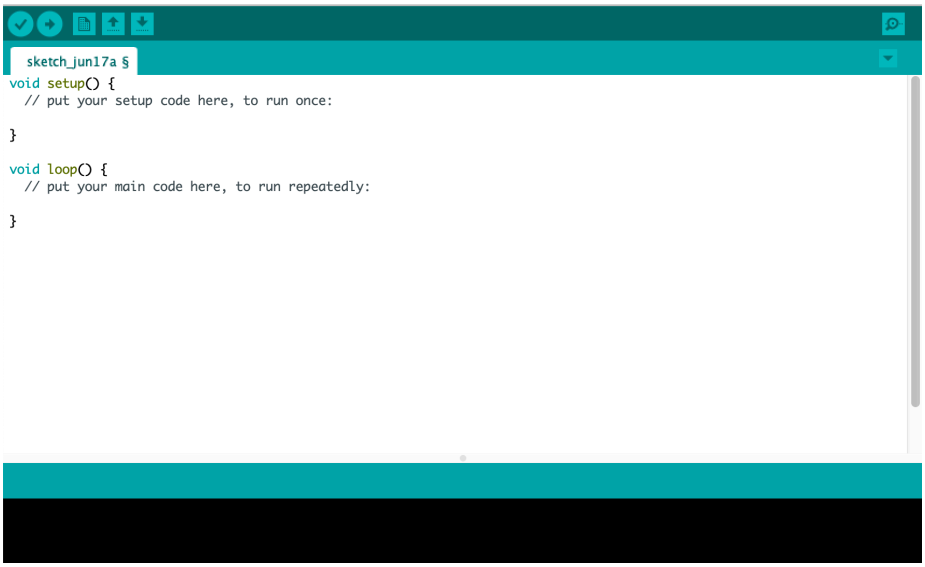
Furthermore, see below a picture of the circuit assembled:



Step Three
Writing the Code

Now that you have assembled the circuit, it is time to write the code for the smart door behaviour. 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_part1** for instance.

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

- Read the value from the ultrasound sensor, convert the reading into centimetres, end print the obtained distance value to the Serial Monitor.

Let's begin by initialising some variables just before the setup function:

```
// Trigger Pin of Ultrasonic Sensor and  Echo Pin of Ultrasonic Sensor
const int trigPin = D4;
const int echoPin = D8;

// Duration and distance variables
long duration = 0;
int distance= 0;
```

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

Here we initialise all the variables needed to replicate the smart door behaviour. **trigPin** is a reference to the digital pin D4. **echoPin** is a reference to the digital pin D8. **duration** will store the value read by the **echoPin**. Finally, **distance** will store the converted value of **echoPin** in centimetres.

Next the **setup()** function:

```
// put your setup code here, to run once:
void setup() {

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input

  Serial.begin(9600); // Starts the serial communication

}
```

Here we want to set the trigPin to OUTPUT (pinMode) as we want to send out an ultrasound signal. On the other hand, we want to set the echoPin to INPUT (pinMode) as we want to read the ultrasound signal that comes back. Then, we initialise the serial monitor to debug some of the variables.

pinMode: Set the pin mode (<pin number>, <mode>). The first argument is the pin reference and the second argument is the mode, either INPUT (read signal from digital pin) or OUTPUT (send signal to digital pin).

Serial.begin(9600): Starts the serial monitor on port 9600. You can use the serial monitor to print and debug your variables.

At this point, we have set the pin mode for our component. We can now use some utility functions to code the door functionalities.

First, we want to create a function which allows us to send out the ultrasound signal from the trigPin, read the time that it took for the ultrasound to come back to the echoPin, convert the time in distance (cm), and print the obtained value to the Serial Monitor.

The **distanceCentimeter()** function:

```
// Calculates the distance in cm
void distanceCentimeter() {

  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);

  // Clears the trigPin
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculating the distance in cm
  distance = (duration * 0.034)/2;

  // Prints distance to Serial Monitor
  Serial.print(distance);
  Serial.println(": Centimeters");

}
```

Type the above code after the loop() function.

The **distanceCentimeter()** utility function converts the time that the ultrasound takes to come back to the module and converts it into distance.

First, we clear the trigger pin by setting it to LOW for 2 microseconds.

Next, we send out the ultrasound signal (HIGH) for 10 microseconds before clearing again the trigger signal (LOW) for 2 microseconds.

Using the pulseIn() function, we read the travel time and store that value into the variable duration. The variable now contains the travel time in microseconds.

Finally, we convert the travel time into distance using the distance formula:

Space = Time * Speed

The time is stored inside the **duration** variable and the speed of sound is 0,034cm/us. Notice that we have to divide the equation by 2 as the value that you get from the Echo pin will be double that number because the sound wave needs to travel forward and bounce backward:

distance = (duration * 0.034) / 2

The last few lines of the function simply prints the obtained distance to the Serial Monitor.

Now that we have the utility function in place, it is time to put the code together in the **loop()** function to simulate the smart door behaviour.

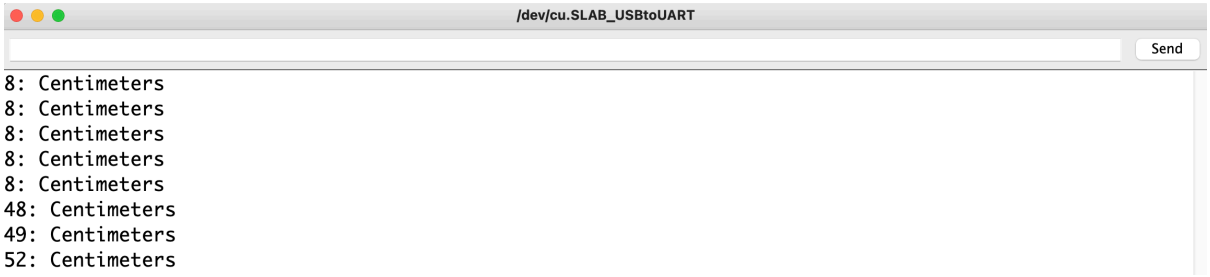
The **loop()** function:

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

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

}
```

The **loop()** function uses the utility function to print the distance on the Serial Monitor.

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

Now open the Serial Monitor and move your hand in front of the ultrasound sensor. You should see the distance printed in cm:



Step Five
Additional Tasks

In the next exercise, you will add more components to the smart door circuit.

Meantime, try the following:

- Can you convert the distance from cm to any other valid measure?
- What is the maximum distance range that the ultrasound sensor can detect?