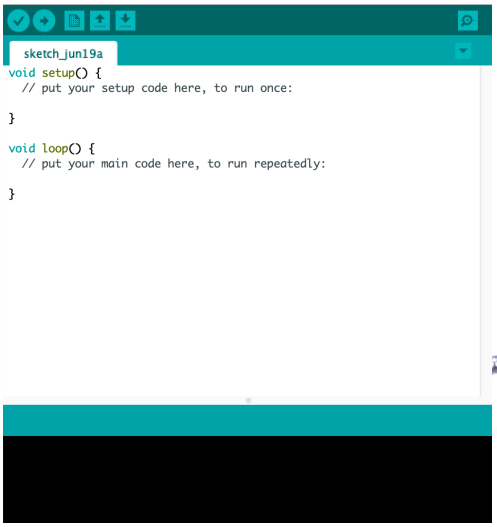
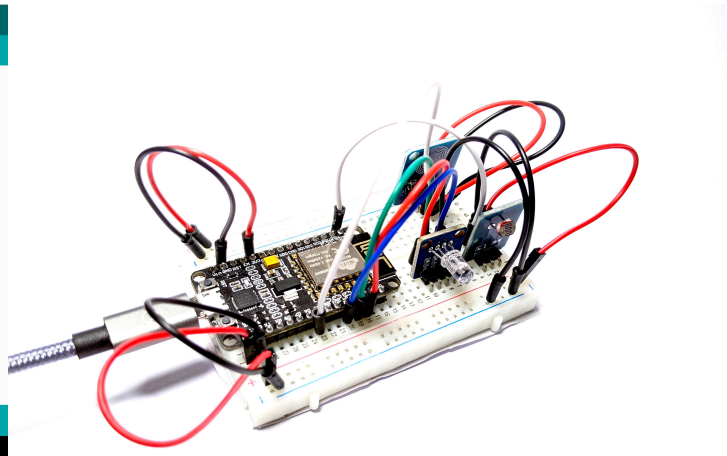
Smart Chair

Part one: The circuit and code

Project description:

In this project, you will explore both the **digital** and the **analog** Input/Output pins of the ESP8266 board by wiring up a circuit to simulate some of the functionalities of a smart chair. For instance, you will output the status of the chair (whether someone is sitting on it or not), and have the chair light up when it gets dark. The circuit is of course just a simulation of real smart chair behaviours and you are not expected to create the final product. The project will also get you get familiar with additional electronic components such as RGB LEDs, capacitive touch sensors, and photoresistors.



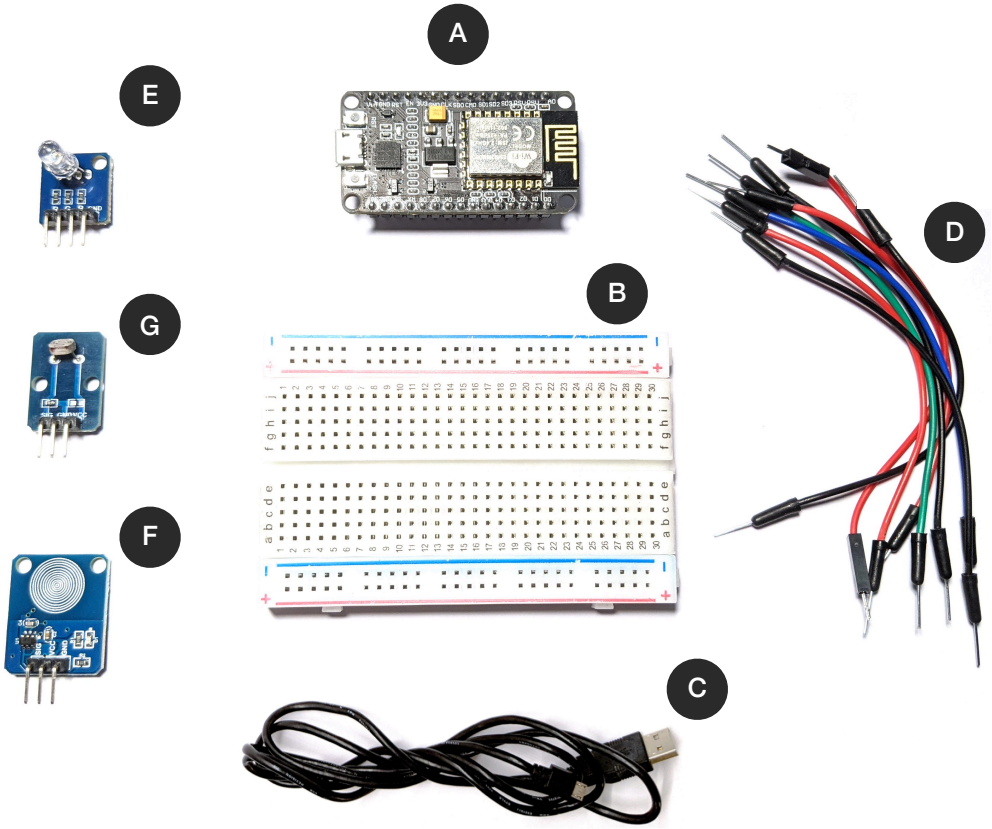


Project objectives:

- Understand the basics of digital and analog Input/Output pins
- Explore electric components such as RGB LEDs, capacitive touch sensors, and photoresistors.
- Write code to manipulate digital and analog Input/Output pins

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
C	1	Micro USB Cable	A USB cable to power and upload instructions to a microcontroller
D	14	Jumper Wires	Conductive cables frequently used with a breadboard to connect two points in a circuit
E	1	RGB LED module	A patch of three light sources that emits light when current flows through it
F	1	Touch switch sensor	A capacitance measuring circuit that when detects a change in capacitance generates a LOW or HIGH signal
G	1	Photoresistor sensor	A light-controlled variable resistor where its resistance decreases with increasing incident light intensity



Step One
Introduction and Theory

The aim of this project is to revisit the concept of digital Input/Output pins and introduce you to the concept of analog Input/Output pins. Just like a common Arduino, the ESP8266 board has analog input/output pins: the A0 pin to be precise. Similarly to digital pins, analog pins can be used to read and write a signal. The main difference is that such a signal is not either 0V or 3.3V, like it occurs with digital pins, but it can read or send a range of voltages between 0 and 3.3V. Such pins are ideal for reading signals which are not binary. On the ESP8266, the A0 analog pin reads signals in the range of 0 - 1023. For instance, you can use the A0 analog pin and a photoresistor to measure the amount of light in a room, amount that it is not binary (either a 1 or a 0) but that instead ranges between a minimum (usually 0 on the ESP board) and a maximum (usually 1023 on the ESP board).

As a process to refresh the concept of digital pins and establish some knowledge about analog pins, you will build a circuit to simulate a smart chair. The idea here is to create a circuit that tells you when someone is sitting on a chair. Furthermore, the chair should light up when it gets dark. You will use an RGB LED as the visual signal that shows whether someone is sitting on the chair, a touch switch sensor to identify whether someone is sitting on the chair, and a photoresistor sensor to detect a low light condition and to activate the chair lighting.

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

- **RGB LEDs:** An RGB LED component is a combination of three individual LEDs, respectively a red LED, a green LED, and blue LED. Combining together these three sources of light and varying their intensity allow to have one single component to produce multiple coloured lights. The component has usually four leads: the ground (GND), the red LED (R), the green LED (G), and the blue LED (B). You can use this component to signal if the chair is free (green light) or if it is taken (red light).
- **Touch switch sensor:** The touch sensor is a component which detects a change in capacitance , or the ability of a component to collect and store energy, and generates a LOW or HIGH signal. The component has three leads: the ground (GND), the source (VCC), and the signal (SIG). Touching the sensor with your skin does change the capacitance of the component. You can use this component to detect if someone sits on the chair.
- **Pothoresistor sensor:** The photoresistor sensor is a component with a variable resistor where its resistance decreases with increasing incident light intensity. A high intensity of light incident on the surface will cause a lower resistance, whereas a lower intensity of light will cause higher resistance. The component has three leads: the ground (GND), the source (VCC), and the signal (SIG). You can use this component to detect the amount of light in a room and eventually turn on the lights on the chair.

Step Two
Wiring up the Smart Chair Circuit

The circuit assembly for this exercise requires you to wire up 1 RGB LED module, 1 photoresistor sensor, and 1 touch switch sensor. If you are wondering about resistors for the RGB LED module, you are absolutely right. Fortunately, the RGB LED module comes with built in resistors and there is no need for you to use them in the circuit for this component.

The RGB LED module should be connected to individual digital input/output pins of your ESP board. This will allow you to control them individually when writing the code for the circuit. The digital pins for the RGB LED module are D2 (red), D3 (green), D4 (blue), as well as ground (GND).

The touch switch sensor should be connected to a digital input/output pins of your ESP board. This will allow you to read a LOW or HIGH value when you touch the component. The digital pins for the touch switch sensor component are D5 (SIG) as well as ground (GND) and source (VCC).

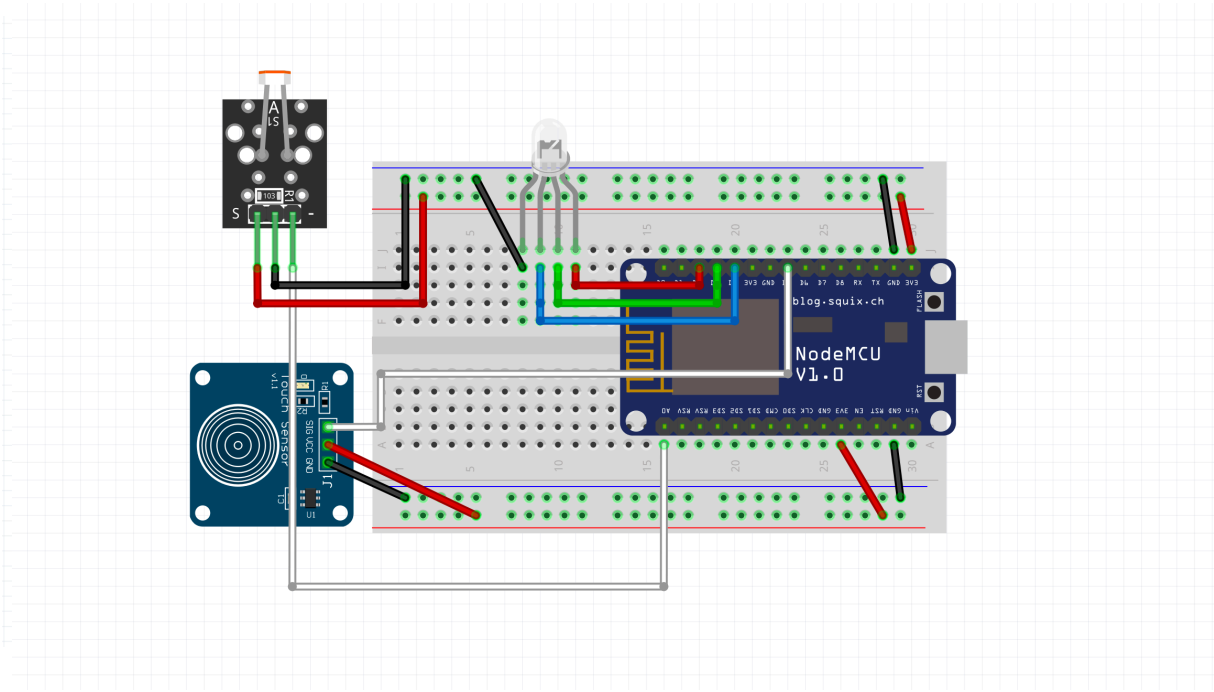
The photoresistor in the end, should be connected to the A0 input/output analog pin. This will allow you to read a range of values which will represent the amount of light. The digital pins for the photoresistor component are A0 (SIG) as well as ground (GND) and source (VCC).

Here a quick recap of the circuit wiring:

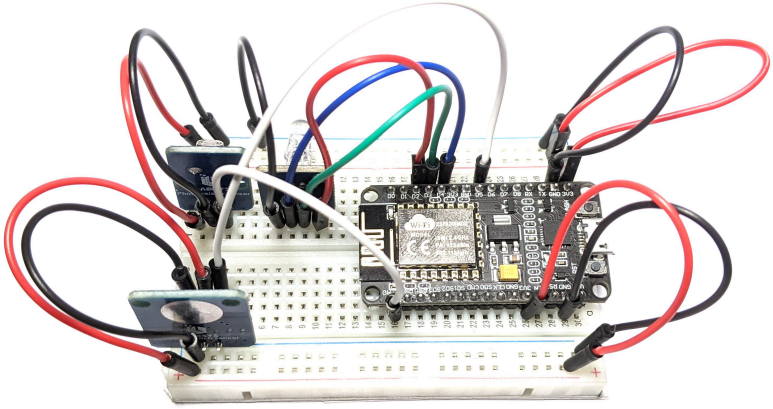
- **RGB LED:** It should be wired up to pins D3, D4, D5. The GND lead goes to ground and the VCC lead goes to 3.3V.
- **Touch switch sensor:** It should be wired up to pin D5. The GND lead goes to ground and the VCC lead goes to 3.3V.
- **Photoresistor:** It should be wired up to pin A0. The GND lead goes to ground and the VCC lead goes to 3.3V.

It is now your turn to assembly the circuit. Use the small breadboard to assemble this circuit.

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 chair 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_chair_part1 for instance.

There are three core functionalities that we want to program here:

- Turn the RGB LED red when you touch the capacitive sensor. Turn the RGB LED green when you release the capacitive sensor.
- Read the digital value from the capacitive touch sensor.
- Read the analog value from the photoresistor.

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

```
// Initialise the RGB pins
const int red_led_pin = D2;
const int green_led_pin = D3;
const int blue_led_pin = D4;

// Initialise the touch sensor pin
// The chair value coming from the touch sensor
const int touch_sensor = D5;
int chair_busy;

// Initialise the photoresistor pin
// The photoresisotr value coming from the photo sensor
// The photoresisotr night treshold
const int photo_sensor = A0;
int photoValue = 0;
int photoTreshold = 550;
```

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

Here we initialise all the variables needed to replicate the smart chair behaviour. **red_led_pin**, **green_led_pin,** and **blue_led_pin** are references to the digital pins D2, D3, and D4. **touch_sensor** is a reference to the digital pin D5, the pin to read the value of the touch sensor component. **chair_busy** stores the value of **touch_sensor**. **photo_sensor** is a reference to the analog pin A0, the pin to read the value of the photoresistor component. **photoValue** stores the value of **photo_sensor**. Finally, **photoTreshold** is our threshold to detect a low light condition.

Next the **setup()** function:

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

  // LEDs as OUTPUT
  pinMode(red_led_pin, OUTPUT);
  pinMode(green_led_pin, OUTPUT);
  pinMode(blue_led_pin, OUTPUT);

  // Touch sensor and photoresistors as INPUT
  pinMode(touch_sensor, INPUT);
  pinMode(photo_sensor, INPUT);

  // Start the serial to debug the values
  Serial.begin(9600);

}
```

Here we we want to set the RGB LED pins mode to OUTPUT (pinMode) as we want to send out a signal to either turn the red, green, and blue LEDs ON or OFF. On the other hand, we want both the photoresistor and the touch sensor pin mode to be INPUT as we want to read their values. Finally, 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 components. We can now use some utility functions to code the chair behaviours.

First we want to create a function which allows us to turn ON and OFF the RGB LED to signal if somebody is sitting on the chair.

The **rgbLed()** function:

```
// Utility function to control the RGB led
void rgbLed(int red_led_amount, int green_led_amount, int blue_light_amount) {

  analogWrite(red_led_pin, red_led_amount);
  analogWrite(green_led_pin, green_led_amount);
  analogWrite(blue_led_pin, blue_light_amount);

}
```

Type the above code after the loop() function.

The **rgbLed()** utility function takes three parameters: the amount of red (0-255), the amount of green (0-255), and the amount of blue (0-255). The function body uses **analogWrite()** to set the amount of light for each individual LED of the RGB LED module.

Wait, what? Are we sending analog signals to what we initialise as digital pins? The short answer is no. Digital pins can simulate analog signals by turning ON and OFF at fast intervals (Pulse-width modulation). The quicker is the ON/OFF cycle, the stronger is the output signal.

In the example of the RGB LED module, you can use the analogWrite on the individual LEDs (red, green, blue) of the component.
For instance, analogWrite(red_led_pin, 255) will set the red LED of the RGB LED module at maximum brightness. This is because the Pulse-width modulation (ON/OFF cycle) happens very fast.

analogWrite: Set the signal amount(<pin number>, <amount>). The first argument is the pin reference and the second argument is the amount, ranging from 0 (OFF) to 255 (ON).

You can then use the **rgbLed()** utility function to set the overall light of the RGB LED module component.

Next, we want to read the value from the touch sensor and turn on the correct RGB LED light. This will simulate somebody sitting on the chair. A red light should appear if the chair is busy and a green light should appear if the chair is free.

The **chairSignal()** function:

```
//  Utility function to control the chair availability
void chairSignal(){

  chair_busy = digitalRead(touch_sensor); // Read the value of the touch sensor (0--1)
  Serial.println(chair_busy);

  if(chair_busy == HIGH){
    rgbLed(255, 0, 0); // The chair is busy, RED
  }else{
   rgbLed(0, 255, 0); // The chair is free, GREEN
  }

}
```

Type the above code after the rgbLed() function.

The **chairSignal()** utility function monitors the value of the touch sensor. First, we read the value of the touch sensor component (digitalRead) and we store the value in the **chair_busy** variable. The serial monitor (Serial.println) is used to debug and check the value of the **chair_busy** variable. Next, we use the **chair_busy** variable to determine if the capacitive sensor is touched. If it is, it means that the chair is taken and we set the RGB LED to a full red light, if it is not, we set the RGB LED to a full green light, meaning that the chair is free.

The next step is to read the value from the photoresistor to determine a low light condition.

The **isNight()** function:

```
// Utility function to check if it is night
bool isNight (){

  photoValue = analogRead(photo_sensor); // Read the value of the photo resistor (0--1023)
  Serial.println(photoValue);

  if ( photoValue > photoTreshold){ // Not night
  return false;
  }
  return true; // It is night

}
```

Type the above code below the chairSignal() function.

The **isNight()** utility function monitors the level of light detected by the photoresistor. First, we read the value from the photoresistor (analogRead). The value read will range between 0 and 1023 depending on the amount of light detected (no light detected will give a value closer to 0). The serial monitor (Serial.println) is used to debug and check the value of the **photoValue** variable. Next, we compare the value obtained with our threshold value and return either true (it is dark), or false (it is not dark).

The last utility function simulates the chair light up behaviour.

The **lightShow()** function:

```
// Utility function to start the light show
void lightShow(){

   rgbLed(255, 0, 0);
   delay(500);
   rgbLed(0, 255, 0);
   delay(500);
   rgbLed(0, 0, 255);
   delay(500);
}
```

Type the above code below the isNight() function.

The **lightShow()** utility function simulates a lighting pattern behaviour. This function will be triggered on low light conditions to simulate the the chair light up behaviour. Here, we are simply changing the RGB LED module from red, to green, to blue with a small delay in between.

- rgbLed(255,0,0) —> full red, no green, no blue
- rgbLed(0,255,0) —> no red, full green, no blue
- rgbLed(0,0,255) —> no red, no green, full blue

Now that we have all the utility functions in place, it is time to put the code together in the **loop()** function to simulate the smart chair behaviours. Remember, we want to first check the light conditions. If it is dark, we want to start the lighting behaviour. Otherwise, we want check whether the chair is available or not and output the correct LED signal.

The **loop()** function:

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

  // Check the photoresistor threshold
  if(isNight()){
   // It is night, start the light show
   lightShow();
  }else{
   // It is not night, check for chair availability
   chairSignal();
  }

}
```

The **loop()** function uses all our utility functions to replicate the smart chair behaviours. First, it checks for low light conditions using the photoresistor. If a low light condition is detected, the chair lights up with a coloured pattern. Otherwise, the check for the chair availability is triggered.

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

Place the circuit in a lit environment. You should see the RGB LED turn green, signalling that the chair is available. If you touch the capacitive sensor, the RGB LED should turn red, signalling that the chair is now taken.

Now place the circuit in a low light condition. You should see the RGB LED changing colour from red, to green, to blue, signalling that the chair is now in the dark and it required some light.

Step Five
Additional Tasks

Now that you have successfully completed the smart chair exercise, try to alter the code to modify the lightning behaviour.

Try the following:

- Change the lightShow sequence. Can you make the lightning animation slower or faster? What about red first, then blue, then green?
- Can you adjust the threshold for the light sensitivity to fit your home environment? Hint: Use the serial monitor to check the photoresistor range value.
- Can you think of any other functionality that a smart chair could implement?