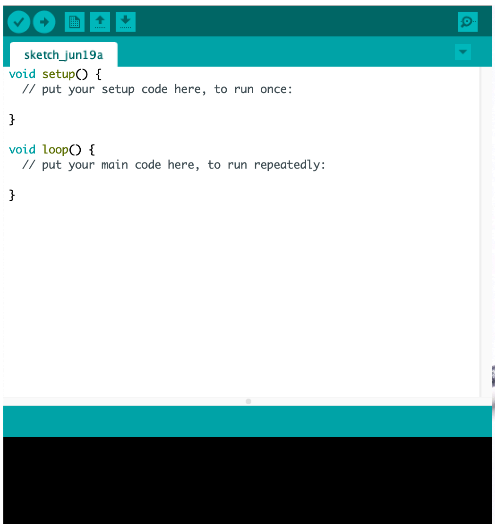
Smart Fridge

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 fridge. For instance, you will turn ON and OFF the fridge and control the its internal temperature. The circuit is of course just a simulation of real smart fridge and you are not expected to create the final product. The project will also get you get familiar with additional electronic components such as switches, and potentiometers.

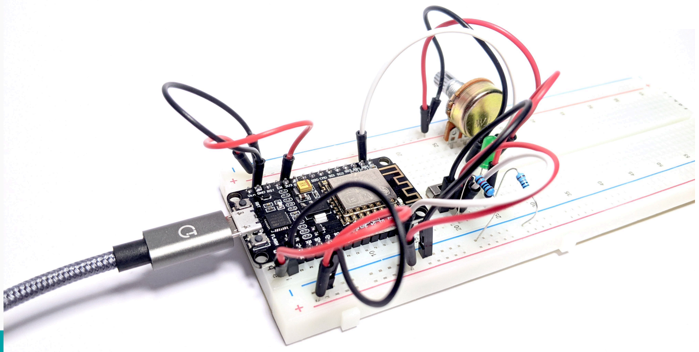


```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

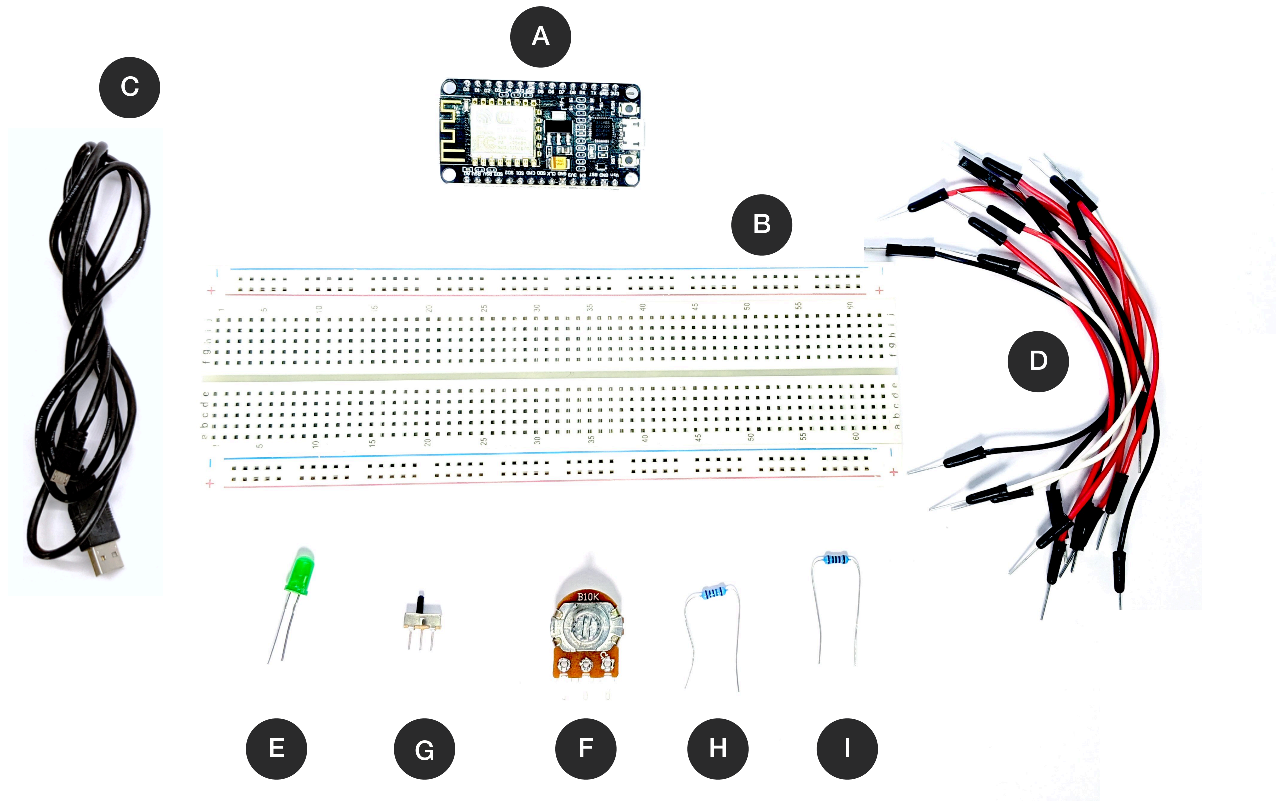


Project objectives:

- Expand your knowledge on digital and analog Input/Output pins
- Explore electric components such as switches, and potentiometers
- Use the Serial Monitor to print out values

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	12	Jumper Wires	Conductive cables frequently used with a breadboard to connect two points in a circuit
E	1	2V Green LED	A semiconductor light source that emits light when current flows through it
F	1	Potentiometer	A three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider
G	1	Switch	An electrical component that can disconnect or connect the conducting path in an electrical circuit
H	1	220 Ohm Resistors	A passive two-terminal electrical component that implements electrical resistance as a circuit element
I	1	10 Kilo Ohm Resistor	A passive two-terminal electrical component that implements electrical resistance as a circuit element



Step One

Introduction and Theory

You should now be familiar with the concepts of digital and analog Input/Output pins. In the smart chair exercise you encountered some interesting components such as the capacitive touch sensor, the RGB LED, and the photoresistor. This exercise introduces you to two new components: a switch and a potentiometer.

As a process to exploring these two new components, you will build the first part of a circuit to simulate a smart fridge. The idea here is to create a circuit that allows you to read the values of both the switch component and the potentiometer component. The switch will allow you to control whether the fridge is ON, and the potentiometer will allow you to control and set the fridge temperature. A green LED will signal when the fridge is ON.

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

- **LEDs:** An LED is a component which produces light by passing the electric current through a semiconducting material called diode. An LED has a shorter lead (Cathode) and longer lead (Anode). This is to distinguish how the component should be wired up in the circuit. The shorter lead must be connected to ground (GND) and the longer lead to source (3.3V). Wiring the LED in the opposite direction will not produce any light. You will use the LED to signal the status of the fridge (ON or OFF).

- **Resistors:** The purpose of a resistor is to limit the current through electrical components. Some components require less current than the source can output and resistors are designed to limit the current trough such components. Resistors are often used with LEDs to balance the current flow and allow the correct current load. Differently from LEDs, resistors can be wired up independently from their direction. You will notice coloured lines on the resistor bodies. The colour pattern is used to identify the value of such resistors and measured in Ohms. The higher the value, the stronger the current limitation. You will use the resistors to limit the current on the LED and to read the correct value from the switch.

- **Switch:** Similarly to a push button, a switch is a component that either allows current to flow or not to flow. When switched, a small metal spring inside makes contact with two wires, allowing electricity to flow. As opposed to a push button, you do not need to keep it pressed to allow current to flow. The switch has three leads. When it is ON, current is allowed between the middle lead and one of the external leads. You will use the switch to turn ON and OFF the fridge.

- **Potentiometer:** Similarly to a photoresistor, a potentiometer is a component with a variable resistor where its resistance is controlled manually using a rotating pole. The potentiometer has three leads. The external ones are the source (VCC) and ground (GND). The middle pin is the variable resistance. You will use the potentiometer to set the fridge temperature.

Step Two
Wiring up the Smart Fridge Circuit

The circuit assembly for this exercise requires you to wire up 1 green LED, 2 resistors, 1 switch, and 1 potentiometer.

The middle lead of the potentiometer should be connected to the A0 input/output analog pin. This will allow you to read a range of values which will represent the fridge temperature. The outer leads should be connected to ground (GND) and source (VCC).

The middle lead of the switch component should be connected to the digital pin D1. This will read the value of the switch. The outer leads should be connected to ground (GND) and source (VCC). Additionally, you want to add a 10K resistor between the middle lead of the switch and ground (GND). The resistor will act as a pull-down resistor, giving you the correct LOW read when the switch is OFF.

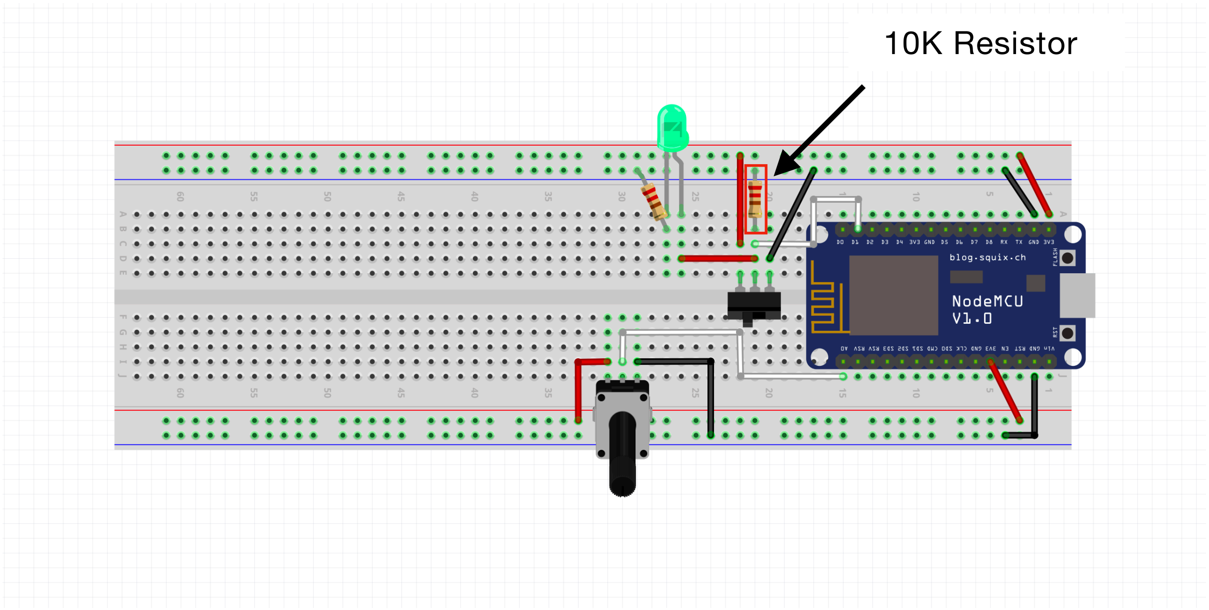
Finally, the green LED should be connected between the middle lead of the switch (longer lead) and ground (GND) with a 100ohm resistor to limit its current. This will light up when the switch is ON.

Here a quick recap of the circuit wiring:

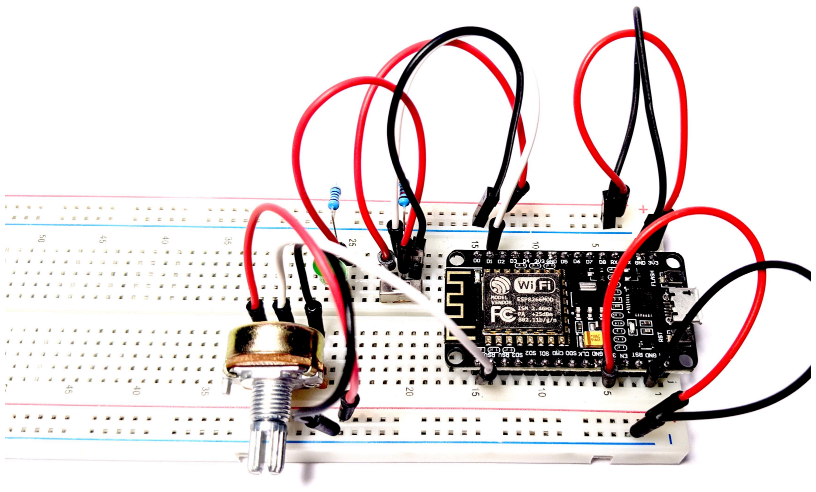
- **Potentiometer:** The middle lead should be wired up to pin A0. The outer leads go to ground (GND) and source (VCC).
- **Switch:** The middle lead of the switch component should be wired up to pin D1. The outer leads go to ground (GND) and source (VCC). An additional 10K resistor should be added between the middle lead of the switch and ground (GND).
- **LED:** The longer lead should be wired up to the middle lead of the switch. The shorter lead should be wired up to ground (GND). An additional 10ohm resistor should be added between the LED shorter lead and ground (GND).

It is now your turn to assembly the circuit. Please try to use only one side of the larger breadboard to build this circuit. We will reserve the other half for upcoming projects.

The diagram below shows you how the circuit should be assembled:



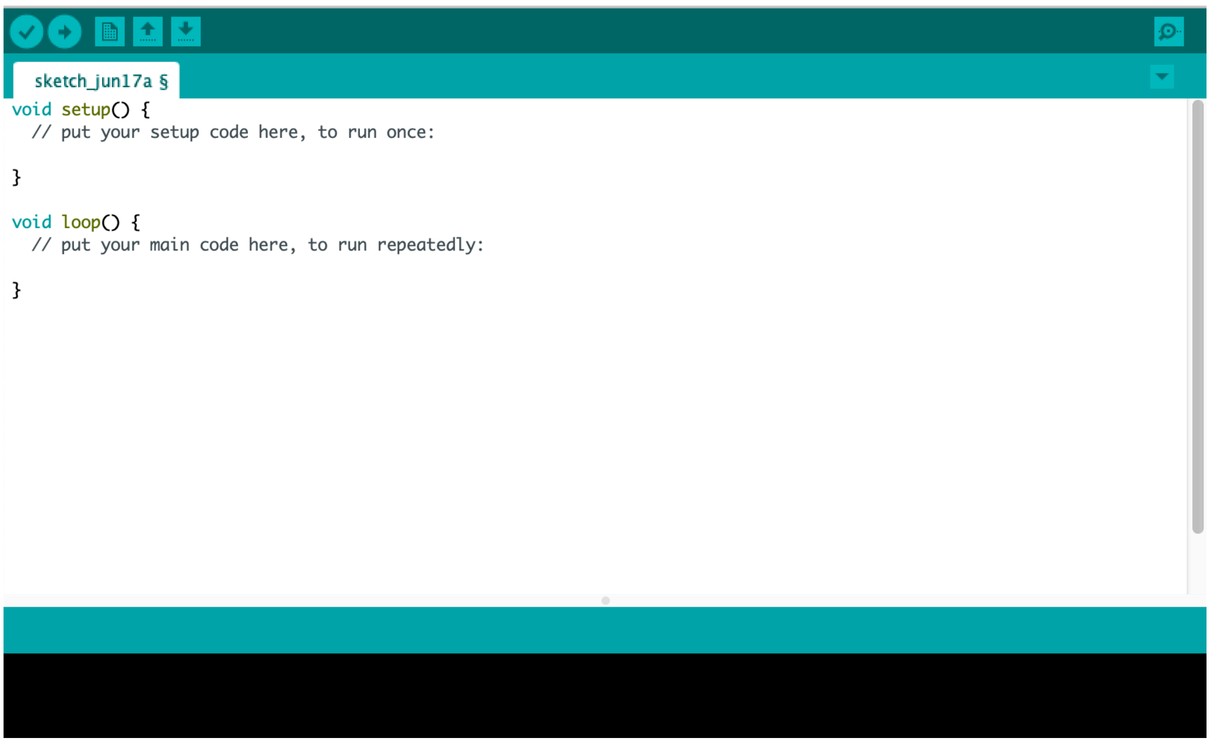
Furthermore, see below a picture of the circuit assembled:



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:



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

}

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

}
```

Feel free to save the sketch and rename it to something sensible:
smart_fridge_part1 for instance.

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

- Use the switch to turn ON the fridge. The green LED should light up when the fridge is ON.
- Read the value from the potentiometer, convert it to a temperature range, and output it on the Serial monitor.

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

```
// Initialise the switch pin
// The variable to store the switch value
const int switch_pin = D1;
int switch_value;

// Initialise the potentiometer pin
// The value coming from the potentiometer
const int potentiometer_pin  = A0;
int poteValue;

// Initialise the minimum and maximum temperature of the fridge
int minTemp = -2;
int maxTemp = 6;
```

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

Here we initialise all the variables needed to replicate the smart fridge behaviour. **switch_pin** is a reference to the digital pin D1. **switch_value** stores the value of **switch_pin**. **potentiometer_pin** is a reference to the analog pin A0, the pin to read the value of the potentiometer component. **poteValue** stores the value of **potentiometer_pin**. Finally, **minTemp** and **maxTemp** are variables to determine the temperature range of the fridge.

Next the **setup()** function:

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

  // Set the switch and potentiometer pins to INPUT
  // you want to read the state here
  pinMode(switch_pin, INPUT);
  pinMode(potentiometer_pin, INPUT);

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

}
```

Here we want to set both the switch pin and the potentiometer pin mode to INPUT (pinMode) as we want to read a signal from both components. 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 components. We can now use some utility functions to code the fridge functionalities.

First we want to create a function which allows us to read the value of the switch and determine whether the fridge is ON or OFF.

The **fridgeOn()** function:

```
// Utility function to check whether the fridge is ON
bool fridgeOn(){

  // read the switch pin value
  switch_value = digitalRead(switch_pin);

  // check the status and return either true or false
  if(switch_value == 0){
    return false;
  }
  return true;

}
```

Type the above code after the loop() function.

The **fridgeOn()** utility function checks the status of the switch digital pin and returns either **true**, if the switch is ON, or **false** if the switch is OFF.

Next, we want to read the value from the potentiometer and map it to a sensible temperature range.

The **fridgeTemperature()** function:

```
// Utility function to set the fridge temperature
int fridgeTemperature(){

  // Read the value of the potentiometer (0--1023)
  poteValue = analogRead(potentiometer_pin);

  // Map the potentiometer value in a range of minTemp - maxTemp
  poteValue = map(poteValue, 0, 1023, minTemp, maxTemp);

  return poteValue;
}
```

Type the above code after the fridgeOn() function.

The **fridgeTemperature()** utility function reads the value from the analog pin A0, the pin connected to the potentiometer, and stores it inside the **poteValue** variable. The **map()** function is a built-in function which takes the value read by the potentiometer (0 - 1023) and converts it into our desired range (-2 — 6). This means that when the potentiometer reads a value of 0, we map it to -2, our minimum desired temperature. The same happens when the potentiometer reads a value of 1023, we map it to 6, our maximum desired temperature. Finally, the function returns the mapped temperature value.

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 fridge behaviours. Remember, we want to first check if the fridge is ON. If it is, we want to use the serial monitor to read the temperature value.

The **loop()** function:

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

  // Only execute if the fridge is ON
  if (fridgeOn()){

    // Print the fridge temperature on the serial monitor
    Serial.println(fridgeTemperature());
  }

}
```

The **loop()** function uses all our utility functions to replicate the smart fridge behaviours. First, it checks if the fridge in ON (the switch component). If the fridge is ON, the mapped temperature is printed in 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.

When you turn on the switch, you should see the green LED light up. Furthermore, the serial monitor should output the fridge temperature in the range of -2 and 6. Rotating the potentiometer pole should output a different temperature inside the serial monitor.

Step Five
Additional Tasks

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

Meantime, try the following:

- Can you use the serial monitor to print “Fridge ON” when the switch is ON and ‘Fridge OFF’ when the switch is OFF?
- Play around with the temperature range and the map() function. Change the minTemp and the maxTemp values. Does the output of the serial monitor change?