EDUDEMOS

3-in-1 Demonstrator:

Solar Energy, Wind Energy, and Water Collection in a Single Educational Project

EDUDEMOS

The “EduDemoS” project is funded by the European Union. Views and opinionsexpressed are however those of the author(s) only and do not necessarily reflect thoseof the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

1. ASSEMBLY OF THE 3-IN-1 DEMONSTRATOR

The assembly of the 3x1 Demonstrator will be carried out modularly, addressing each section separately in the corresponding sections of this guide. This approach aims to simplify the process and enhance understanding, ensuring that the assembly is clear and accessible for those undertaking it. Each step is designed to guarantee a functional and effective outcome, even for individuals with no prior experience in assembling demonstrators.

One of the main goals of this project is to promote the autonomy of educational institutions, enabling them to create their own demonstrators if they have 3D printers on-site. For this reason, we have made the necessary printing files available for the fabrication of the components, which can be downloaded via the following link: 3D Printing Files. With this tool, we aim to empower educational institutions to independently and efficiently integrate these demonstrators into their activities.

1.1. List of Materials

- 1 x **ESP32 DEVKIT V1** with 30 pins
- 2 x **Mini breadboards** with 170 tie-points
- 1 x **DHT11 module**, maximum width 14 mm, with or without built-in LED
- 10/12* x **Male-to-male jumper wires**, 10 cm
- 5 x **Male-to-male jumper wires**, 20 cm
- 3 x **Male-to-female jumper wires**, 10 cm
- 10 x **Male-to-female jumper wires**, 20 cm
- 5 x **1k ohm resistors**
- 1 x **Capacitor**, **100nF** or **1uF**
- 3 x Colored **LEDs** (preferably Red, Green, and Blue)
- 1 x **Solar panel**, maximum dimensions: 45 mm x 60 mm
- 1 x **Water level sensor**, 6 cm x 2 cm
- 1 x 5V **DC motor**, maximum dimensions: 28 mm diameter and 45 mm length
- 1 x **SG-90 Servo motor**
- 1 x **Voltage detector**
- 2 x **Photoresistors** standalone GL5528
- 1 x **6203 bearing**, either ZZ or 2RS type: 40 mm outer diameter, 17 mm inner diameter, and 12 mm height
- 3 x **Wooden dowels**: 8 mm diameter and 40 mm length
- 1 x **Motor gear**: 2 mm inner diameter, 10 teeth, and 5 mm height
- 1 x **Data transfer cable USB to microUSB**
- 1 x **Aluminum tape**, 5 cm wide
- 1 x **Double sided tape**, 3 cm de wide

* If the Motor DC doesn't come with wires, you will need 12 male-to-male jump wires of 10 cm.

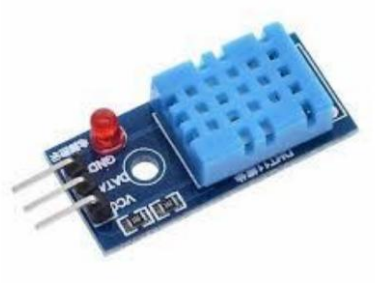
Tools you might need:

- **Soldering iron**: if the Motor DC or the Solar Panel don't come with wires.
- **Flat head screw driver 2 mm**: for the voltaje sensor.

1. 2. Assembly of the Water and Solar Demonstrator

1. 2. 1. Electronic Assembly (Water Component)

We will begin by assembling the water collector. To complete this assembly, you will need the following materials:



1 x DHT-11 Sensor

Responsible for measuring ambient temperature and humidity.

3 x 1 kΩ Resistors

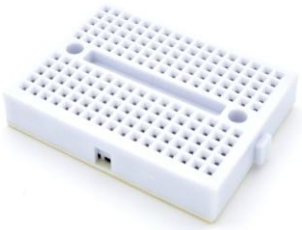
Used to limit the current flowing through LED diodes.

3 x Colored LEDs (Preferably Red, Green, and Blue)

Used as visual indicators of the measured temperature range.

1 x Dupont Cables (Male-Male) 10cm

Necessary for making connections between the different components on the breadboard.



2 x Mini Breadboard

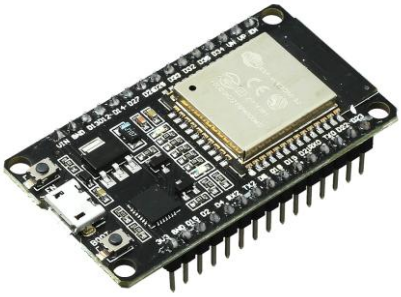
Component necessary to connect the microcontroller board to the different components.

6 x Dupont Cables (Male-Female) 20cm

Necessary for making connections between the different components on the breadboard.

1 x Water level sensor

Used to measure the water level in the tank.

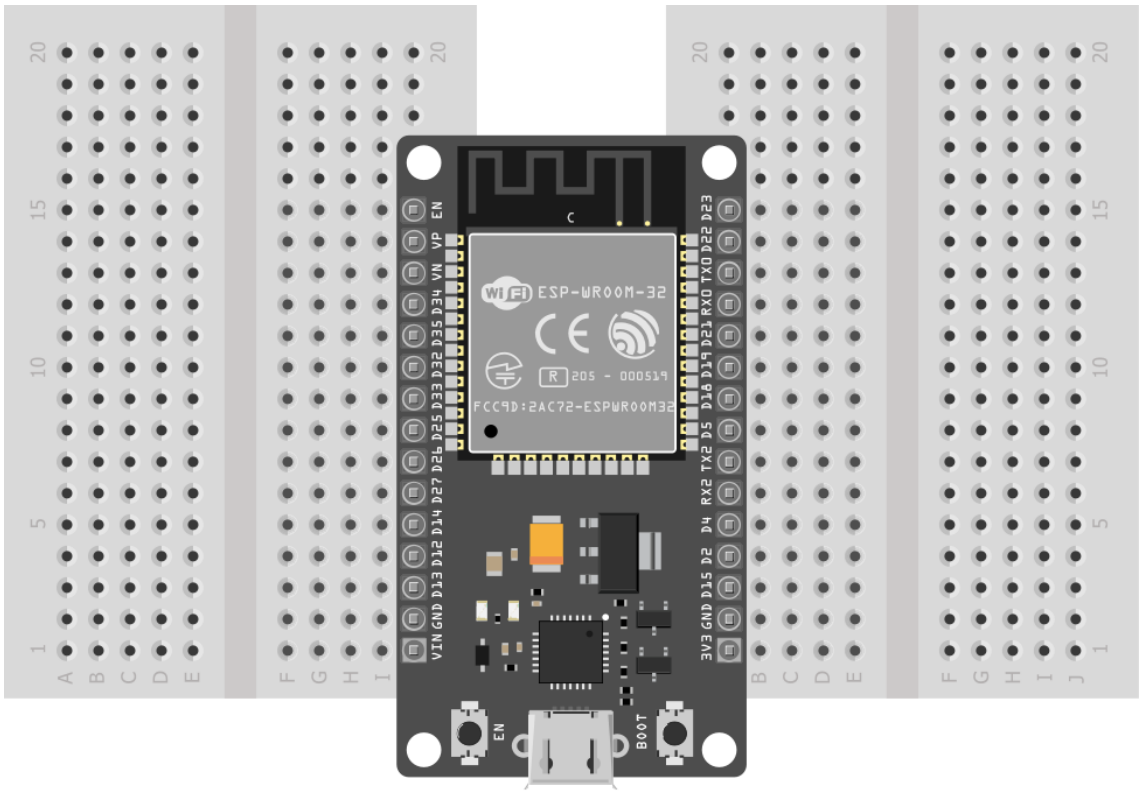


1 x ESP32 DEVKIT V1 (30 pins)

Responsible for executing the programming that controls the operation of the electronics.

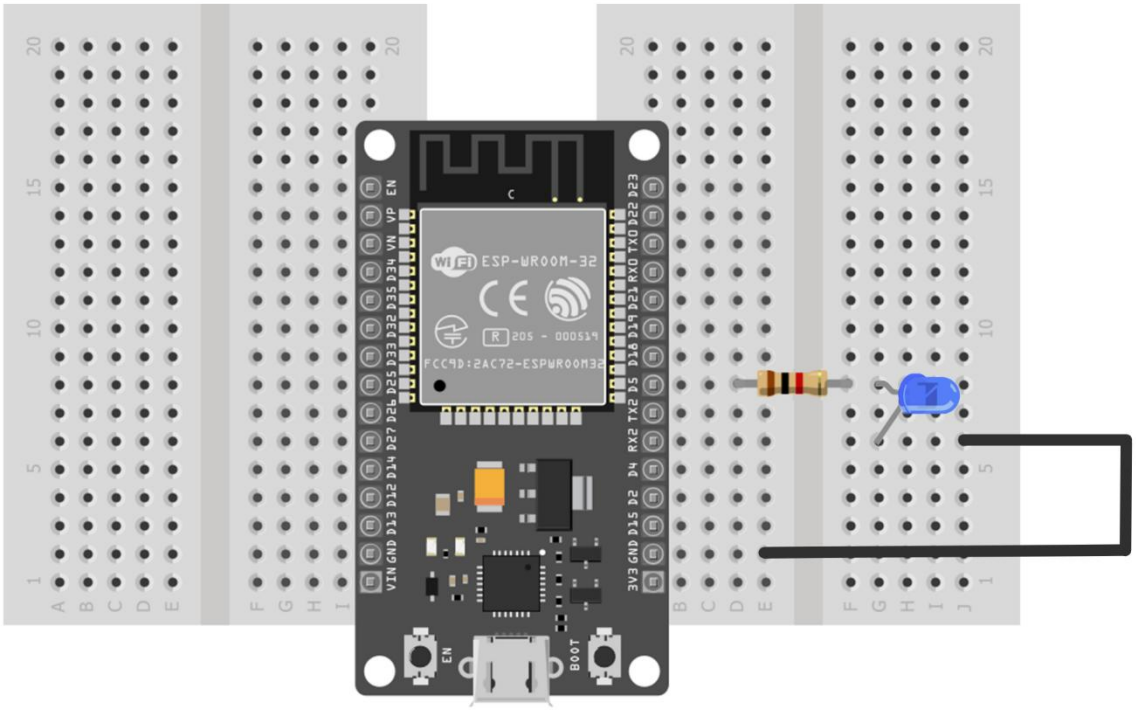
The objective of the assembly is to obtain the values of ambient temperature and humidity, and based on the temperature value, to light up a specific LED as indicators of danger for high or low temperatures. In this way, if the temperature is below 20°C, the Blue LED will light up; if it is above 25°C, the Red LED will light up; and if it is between those values (between 20 and 25°C), the Green LED will light up.

The first step is to place the ESP32 board on two small breadboards to have enough space to connect the components we will need later. Below, you can see the result:



Once we have the microcontroller placed on the two breadboards, we will start connecting the colored LEDs with their current-limiting resistors. We will begin by placing the Blue LED and one of the 1 kΩ resistors in series. We will connect the positive of the LED to **pin D5** and use any GND pin to connect the negative of the circuit. Below, you can see the indicated assembly.

Note: *The LED diode indicates the polarity of the connection at a glance by the length of its leads. The longer lead corresponds to the anode, or positive, and the shorter one is the cathode, or negative. In the diagram, the longer lead is the one connected in series with the resistor.*

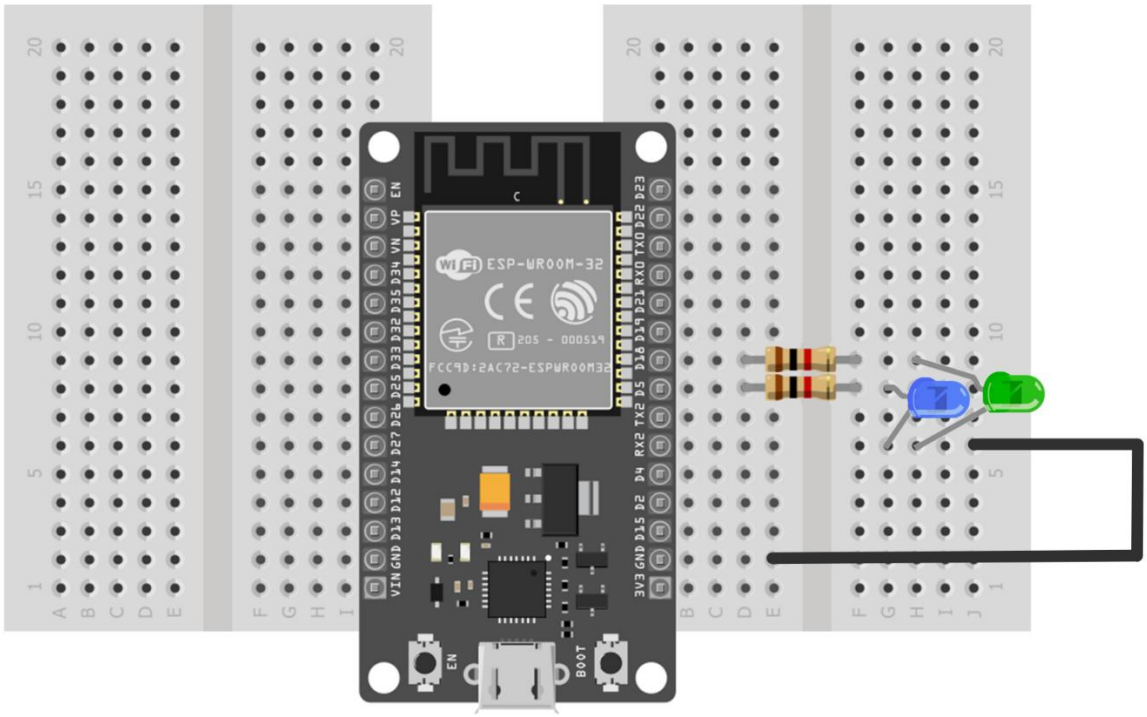


To verify the correct functioning of the assembly, we can upload the following code to the ESP32:

```
#define ledPin1Cold 5 // Define the pin where the Blue LED is connected

// Initialization block
void setup() {

    // We configure the operating mode of the LED pin as an output
    pinMode(ledPin1Cold, OUTPUT);

}

// Main loop of the code
void loop() {

    digitalWrite(ledPin1Cold, LOW); // We turn off the LED
    delay(1000); // We wait 1 second = 1000 milliseconds
    digitalWrite(ledPin1Cold, HIGH); // We turn on the LED
    delay(1000); // We wait 1 second

}
```

This small program makes the Blue LED blink, changing its state between on and off every second. If it does not work correctly, check that the LED is oriented properly (remember that the longer lead is the positive terminal, while the shorter one is the negative) and connected appropriately with the current-limiting resistor.

The next step in the assembly will be to add a second LED. This time, we will place and connect the Green LED to **pin D18** with its resistor. You can see the indicated assembly schematic in the following image.

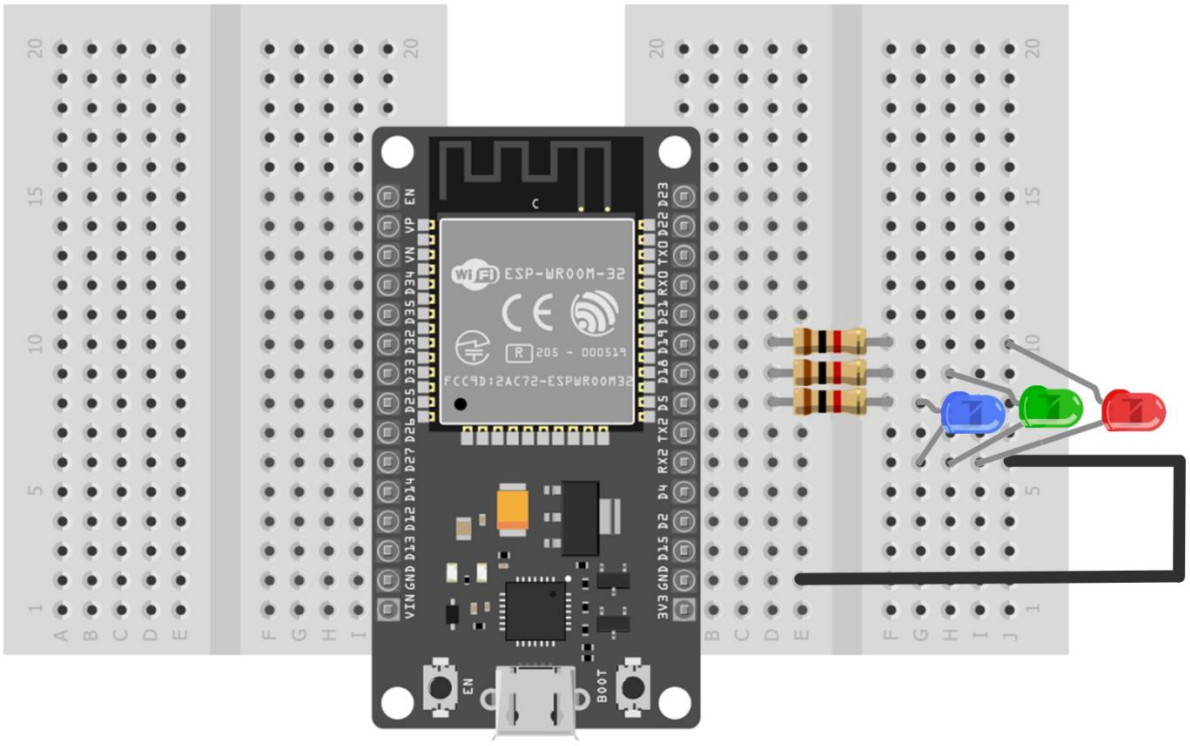


Let's verify that the assembly is correct again. We will use the code below to check that the new LED is also properly placed.

```
#define ledPin1Cold 5 // Define the pin where the Blue LED is connected
#define ledPin2Good 18 // Define the pin where the Green LED is connected

// Initialization block
void setup() {

    // We configure the operating mode of the LEDs pins as outputs
    pinMode(ledPin1Cold, OUTPUT);
    pinMode(ledPin2Good, OUTPUT);

}

// Main loop of the code
void loop() {

    digitalWrite(ledPin1Cold, LOW); // We turn off the Blue LED
    digitalWrite(ledPin2Good, LOW); // We turn off the Green LED
    delay(1000); // We wait 1 second = 1000 milliseconds
    digitalWrite(ledPin1Cold, HIGH); // We turn on the Blue LED
    digitalWrite(ledPin2Good, HIGH); // We turn on the Green LED
    delay(1000); // We wait 1 second

}
```

As you can see, the objective of the code remains the same: to turn both LEDs on and off, changing their state every second. If the Blue and Green LEDs are changing states correctly, we can consider the current assembly valid and proceed to the next step.

To complete the assembly of the colored LEDs needed to fulfill the established objective for this part of the Demonstrator, we need to connect the last LED, which is Red, to **pin D19**. Proceed to assemble the LED diode and the corresponding last resistor, as shown in the following image.

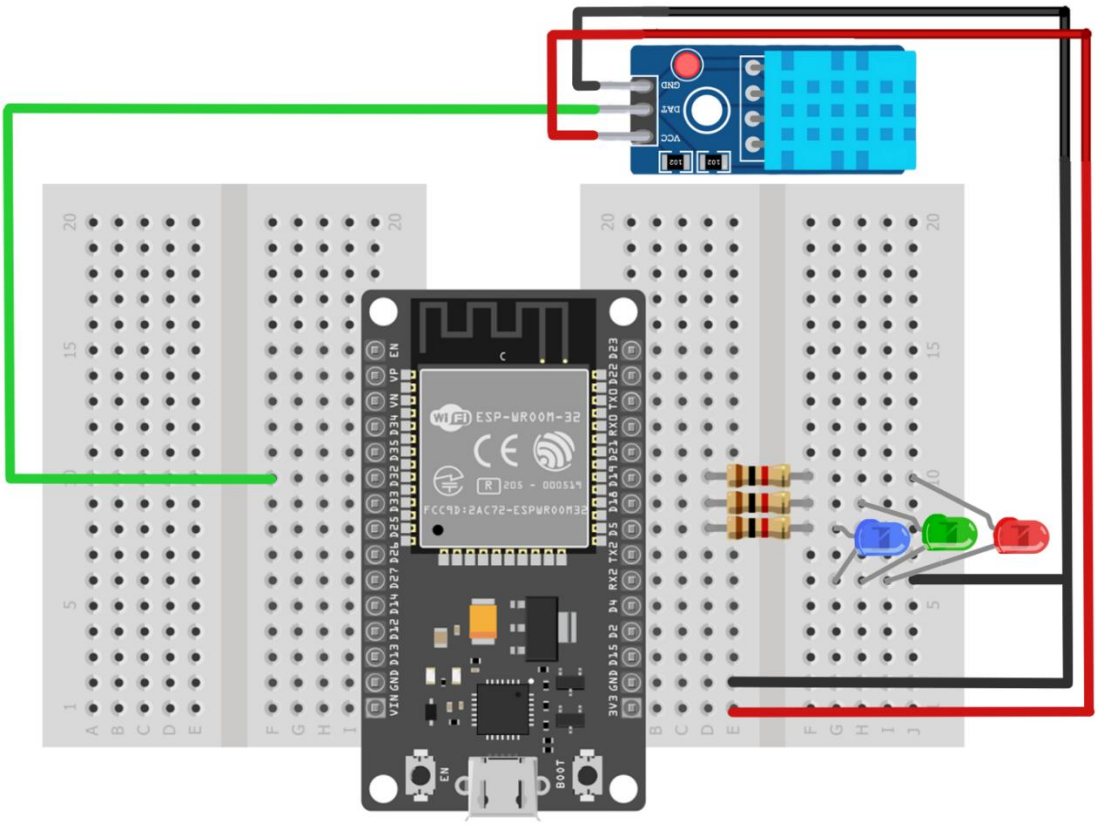


We will check that it is correct by using the same programming as in the previous cases, generating a blink in all three colored LEDs. The code for this is as follows:

```
#define ledPin1Cold 5 // Define the pin where the Blue LED is connected
#define ledPin2Good 18 // Define the pin where the Green LED is connected
#define ledPin3Heat 19 // Define the pin where the Red LED is connected

// Initialization block
void setup() {

    // We configure the operating mode of the LEDs pins as outputs
    pinMode(ledPin1Cold, OUTPUT);
    pinMode(ledPin2Good, OUTPUT);
    pinMode(ledPin3Heat, OUTPUT);

}

// Main loop of the code
void loop() {

    digitalWrite(ledPin1Cold, LOW); // We turn off the Blue LED
    digitalWrite(ledPin2Good, LOW); // We turn off the Green LED
    digitalWrite(ledPin3Heat, LOW); // We turn off the Red LED
    delay(1000); // We wait 1 second = 1000 milliseconds
```

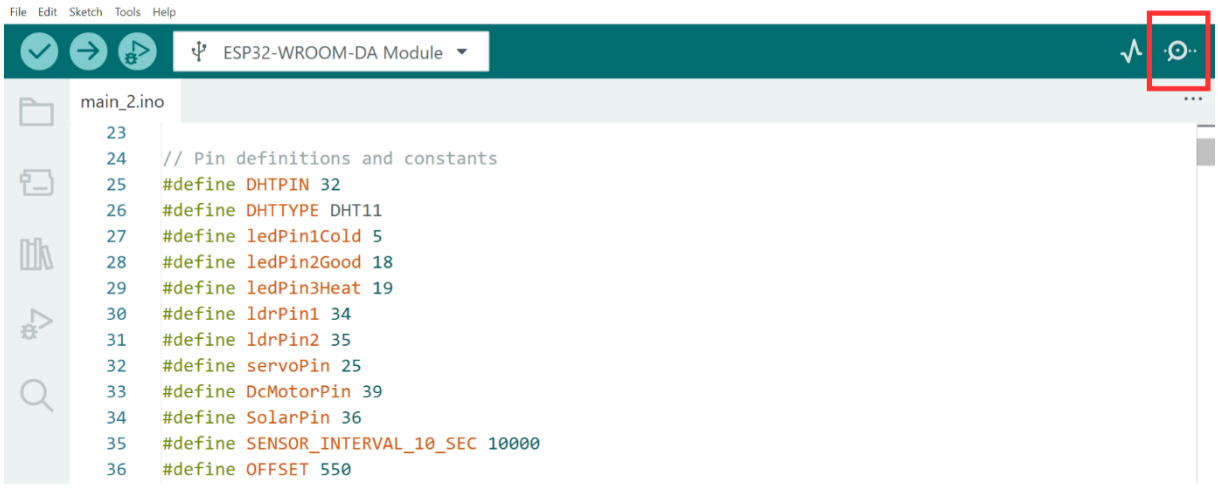
```
    digitalWrite(ledPin1Cold, HIGH); // We turn on the Blue LED
    digitalWrite(ledPin2Good, HIGH); // We turn on the Green LED
    digitalWrite(ledPin3Heat, HIGH); // We turn on the Red LED
    delay(1000); // We wait 1 second

}
```

The next step is to add and connect the DHT11 sensor to **pin D32**, either as an individual component or on a module. Below is the assembly completed so far along with the connections for the DHT11 sensor.



Note: *Make sure that the GND and VCC connections of the DHT11 sensor match the actual pinout of the component, as they may differ from what is shown in the diagram.*

We will now check the correct functioning of the DHT-11 sensor. Below is a test code for the entire assembly completed for this part of the project. Upload the program, and you will see that, in addition to the colored LEDs blinking as they have done so far, the ESP32 board will send the obtained measurements of ambient temperature (in ºC) and relative humidity (in %) to the Serial Monitor every couple of seconds (you can see the button to open it in the following image).

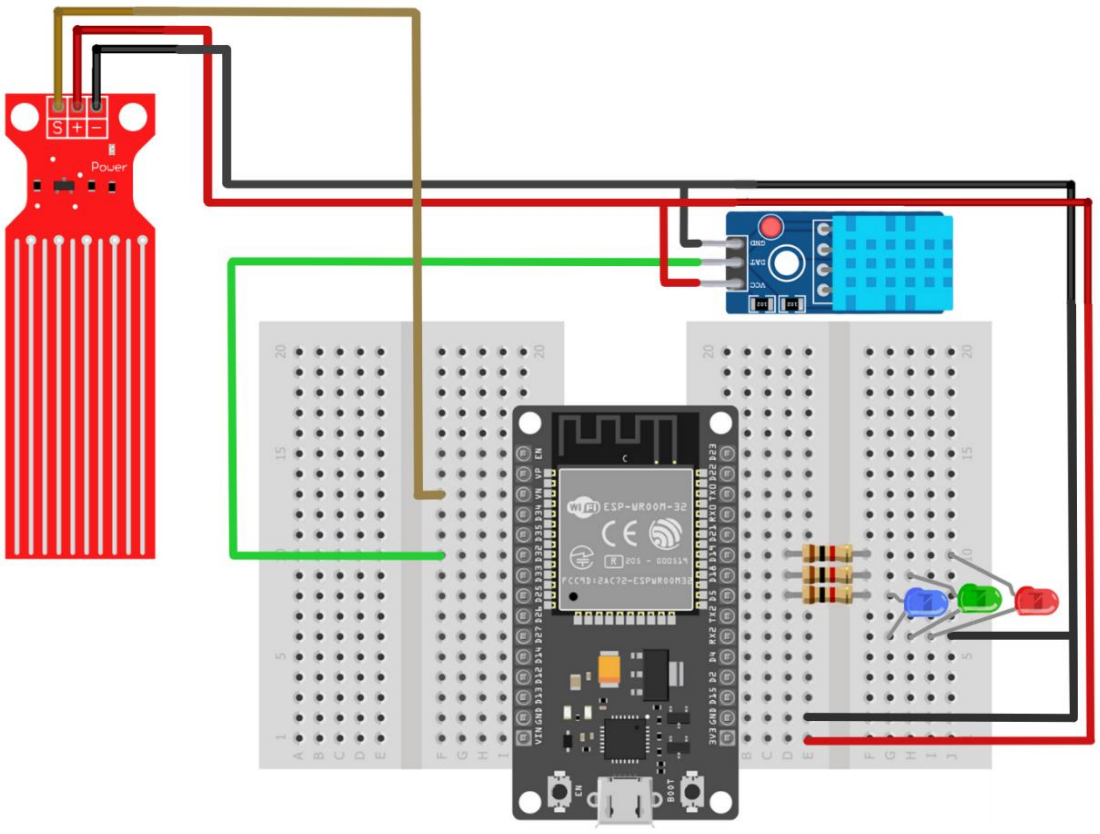



```
#include <DHT.h> // We include the necessary library to control DHT.

#define ledPin1Cold 5 // Define the pin where the Blue LED is connected
#define ledPin2Good 18 // Define the pin where the Green LED is connected
#define ledPin3Heat 19 // Define the pin where the Red LED is connected
#define DHTPIN 32 // Define the pin to which the DHT is connected
#define DHTTYPE DHT11 // Define the type of DHT sensor to use

// Define the type of DHT sensor to be usedCreate an object of type DHT with the corresponding
configuration
DHT myDHT11(DHTPIN, DHTTYPE);

// Initialization block
void setup() {

    // We configure the operating mode of the LEDs pins as outputs
    pinMode(ledPin1Cold, OUTPUT);
    pinMode(ledPin2Good, OUTPUT);
    pinMode(ledPin3Heat, OUTPUT);

    // Initialize the Serial monitor
    Serial.begin(115200);

    // Initialize the DHT sensor
```

```
    myDHT11.begin();

}

// Main loop of the code
void loop() {

    digitalWrite(ledPin1Cold, LOW); // We turn off the Blue LED
    digitalWrite(ledPin2Good, LOW); // We turn off the Green LED
    digitalWrite(ledPin3Heat, LOW); // We turn off the Red LED
    delay(1000); // We wait 1 second = 1000 milliseconds
    digitalWrite(ledPin1Cold, HIGH); // We turn on the Blue LED
    digitalWrite(ledPin2Good, HIGH); // We turn on the Green LED
    digitalWrite(ledPin3Heat, HIGH); // We turn on the Red LED
    delay(1000); // We wait 1 second

    //  We obtain the DHT sensor measurements
    float temperature = myDHT11.readTemperature();
    float humidity = myDHT11.readHumidity();

    // We show the values per monitor Serial
    Serial.print("Temperature: ");
    Serial.println(temperature);
    Serial.print("Humidity: ");
    Serial.println(humidity);

}
```

With the monitor open, make sure that the dropdown located furthest to the right on the Monitor is set to "115200". Now you will be able to see the data sent by the ESP32 from the DHT-11 constantly and verify that the assembly up to this point is functioning correctly.

The next component to be added is the water level sensor module. Connect the sensor to **pin VN**. In the following image you can see how the assembly will look like:



Now we are going to verify the operation of the water level sensor by constantly reading and displaying the measured value of the sensor on the serial monitor:

```
//Water sensor calibration
const int WaterSensorPin = 39; // Analog Value from VN PIN

void setup() {
Serial.begin(115200);
}
void loop() {
  int sensorValue; // Variable to store sensor values
  sensorValue = analogRead(WaterSensorPin); // Read the value from the sensor
  Serial.println(sensorValue); // Print value in serial monitor
  delay(100);
}
```

1. 2. 2. Electronic Assembly (Solar Component)

We will begin by assembling the solar component. To complete this assembly, you will need the following materials:

1 x SG-90 Servo Motor

Used to move the solar demonstrator and orient it toward sunlight.

2 x Photoresistors

Used to measure the level of received light intensity.

2 x 1 kΩ Resistors

Used only with the independent LDRs to assemble a voltage divider controlled by light intensity.

9 x Dupont Cable (Male-Male) 10 cm

Necessary for making connections between the different components on the breadboard.

5 x Dupont cable (Male-Male) 20cm

Necessary for making connections between the different components on the breadboard.

3 x Dupont cable (Male-Female) 10cm

Necessary for making connections between the different components on the breadboard.

4 x Dupont cable (Male-Female) 20cm

Necessary for making connections between the different components on the breadboard.

1 x Voltage sensor

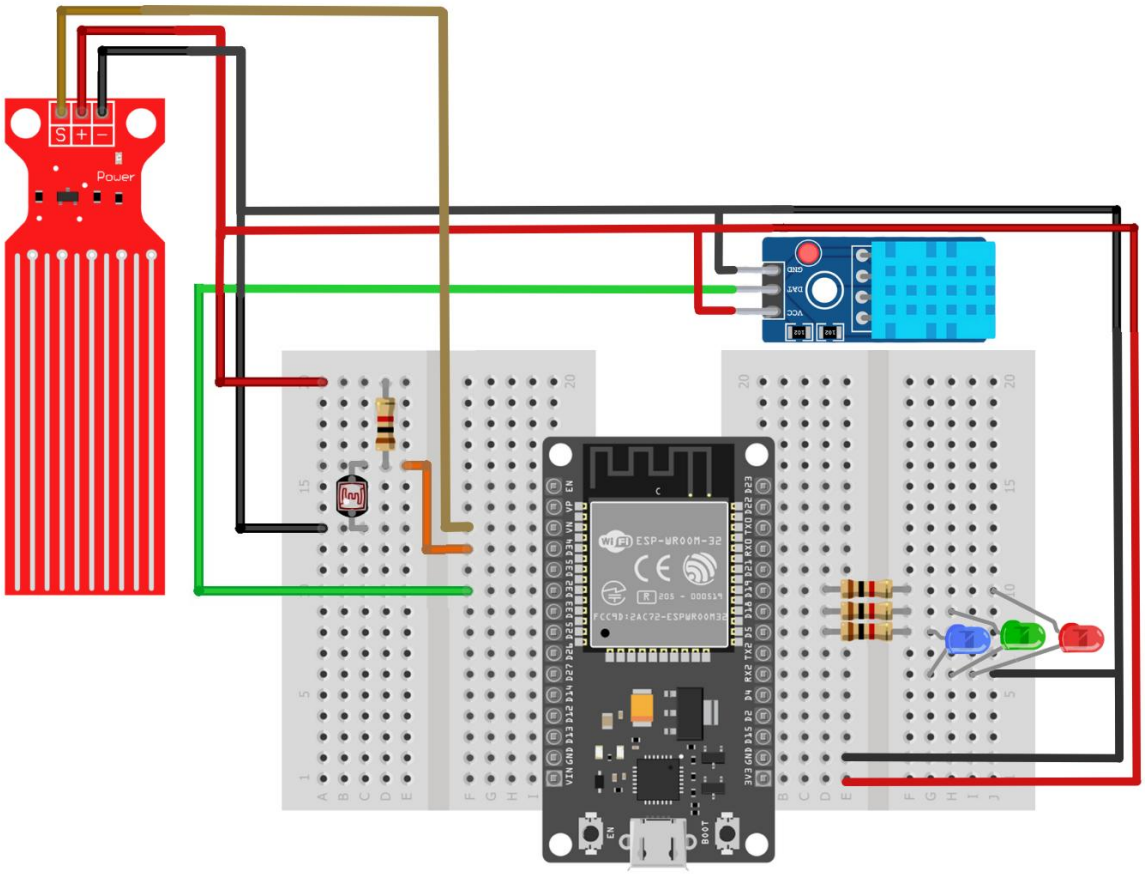
Used alongside the solar panel to detect sunlight.

1 x Solar Panel

Used to detect sunlight.

The objective of the assembly will be to determine the orientation of the incidence of sunlight with the LDRs and rotate the Demonstrator to maximize the amount of sunlight hitting the panel.

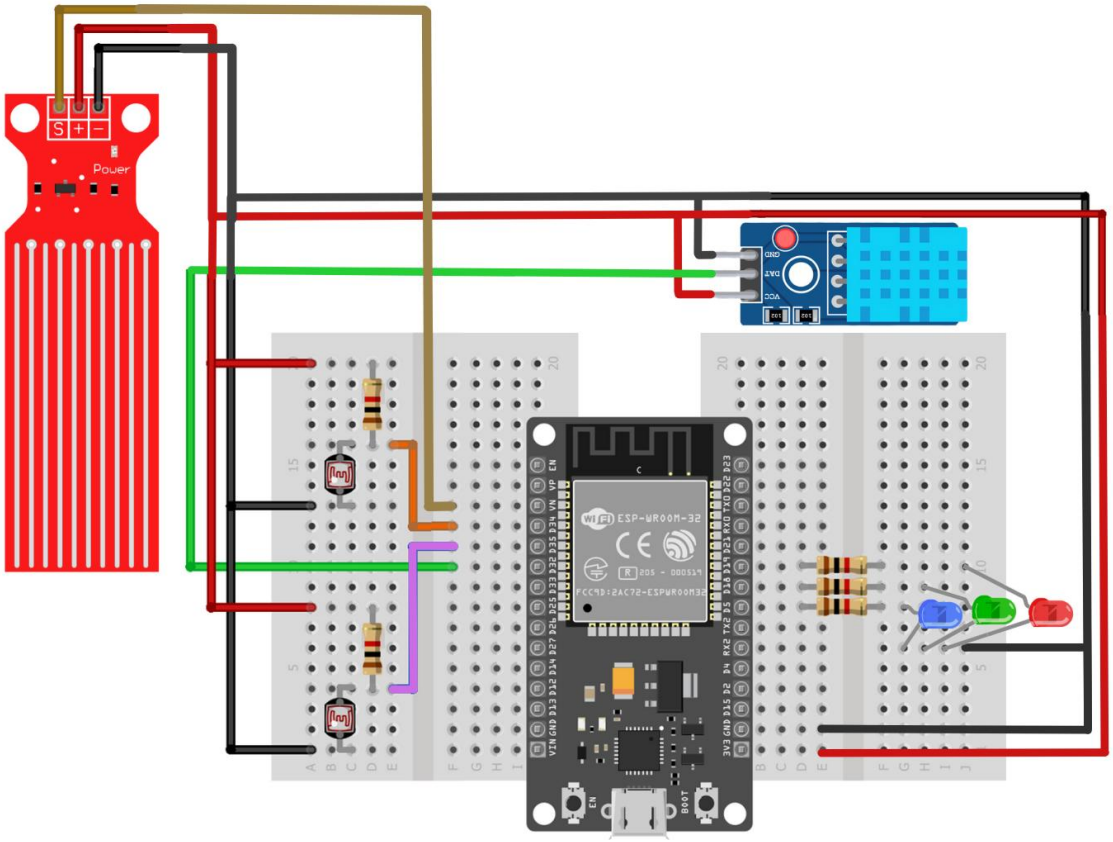
The assembly of the solar component will begin from the last assembly seen in the previous section. In this assembly, an LDR and a resistor will be added in series, forming a voltage divider controlled by light intensity. To obtain the value from the sensor, the digital output will be connected to **pin D34** of the ESP32. The assembly can be seen below:



To check the functionality of the new elements added to the assembly, we will use a code that only reads the LDR value to light up the Red LED with greater or lesser intensity based on the brightness measured by the photoresistor. You can see the code to upload below:

```
#define ldrPin1 34 // Define the pin to which the LDR is connected

// Initialization block
void setup() {

    // Initialize the serial monitor
    Serial.begin(115200);

}

// Main loop of the code
void loop() {

    // We take the reading of the pin where the LDR is located.
    int reading_ldr_1 = analogRead(ldrPin1);
    // Display the value on the Serial monitor
    Serial.print("Luminosity 1: ");
    Serial.println(reading_ldr_1);

}
```

Next, the second LDR with its resistor should be added. To obtain the reading value from this sensor, the digital output will be used, as in the previous case, but this one will be connected to **pin D35**. Below, you can see the assembly with both LDRs for the current part of the Demonstrator.

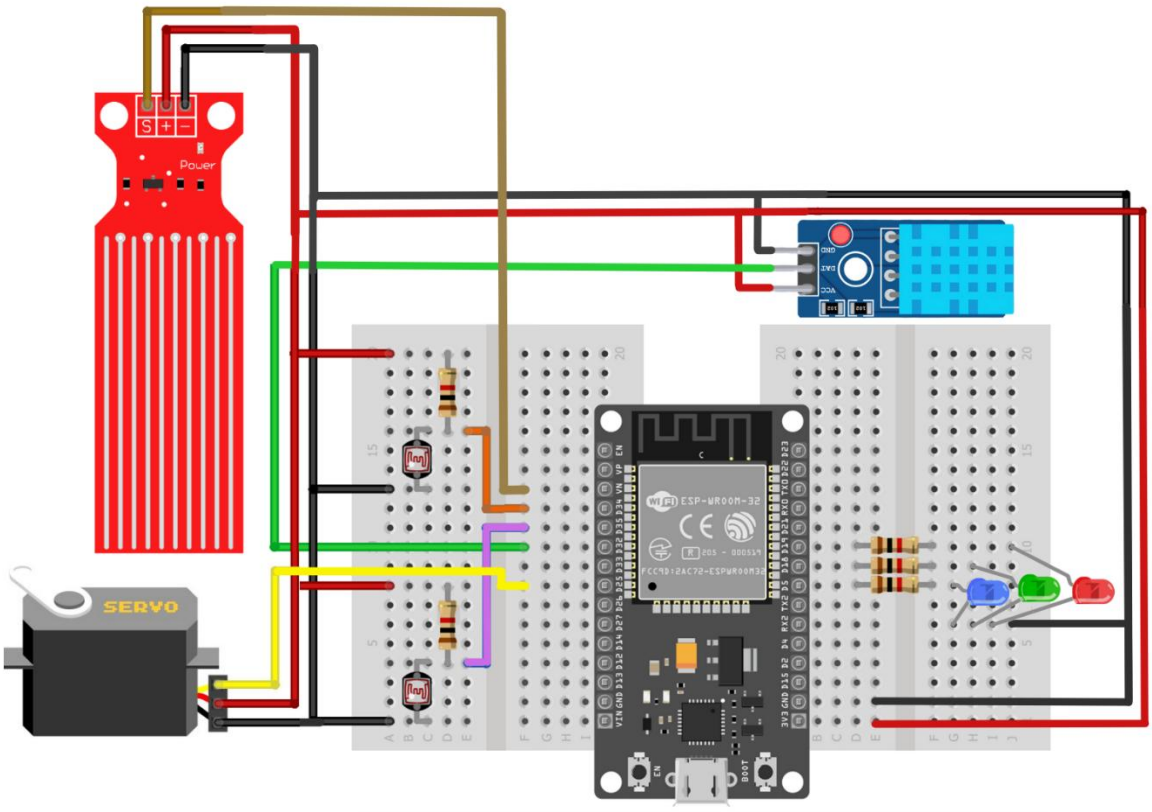


Let's verify that the connections have been made correctly using the following code, which is very similar to the previous one but displays the measurement values from both photoresistors.

```
#define ldrPin1 34 // Define the pin to which the first LDR is connected
#define ldrPin2 35 // Define the pin to which the second LDR is connected

// Initialization block
void setup() {

    // Initialize the serial monitor
    Serial.begin(115200);

}

// Main loop of the code
void loop() {

    // We take the readings of the pins where the LDRs are located
    int reading_ldr_1 = analogRead(ldrPin1);
```

```
    int reading_ldr_2 = analogRead(ldrPin2);
    // We display the first value on the Serial monitor
    Serial.print("Luminosity 1: ");
    Serial.println(reading_ldr_1);
    // We display the second value on the Serial monitor
    Serial.print("Luminosity 2: ");
    Serial.println(reading_ldr_2);

}
```

The next component to be added to the assembly is a servo motor. This will be connected to both power rails (the red wire to the positive and the black wire to the negative), and the yellow wire (which is the control signal) will be connected to **pin D25**.

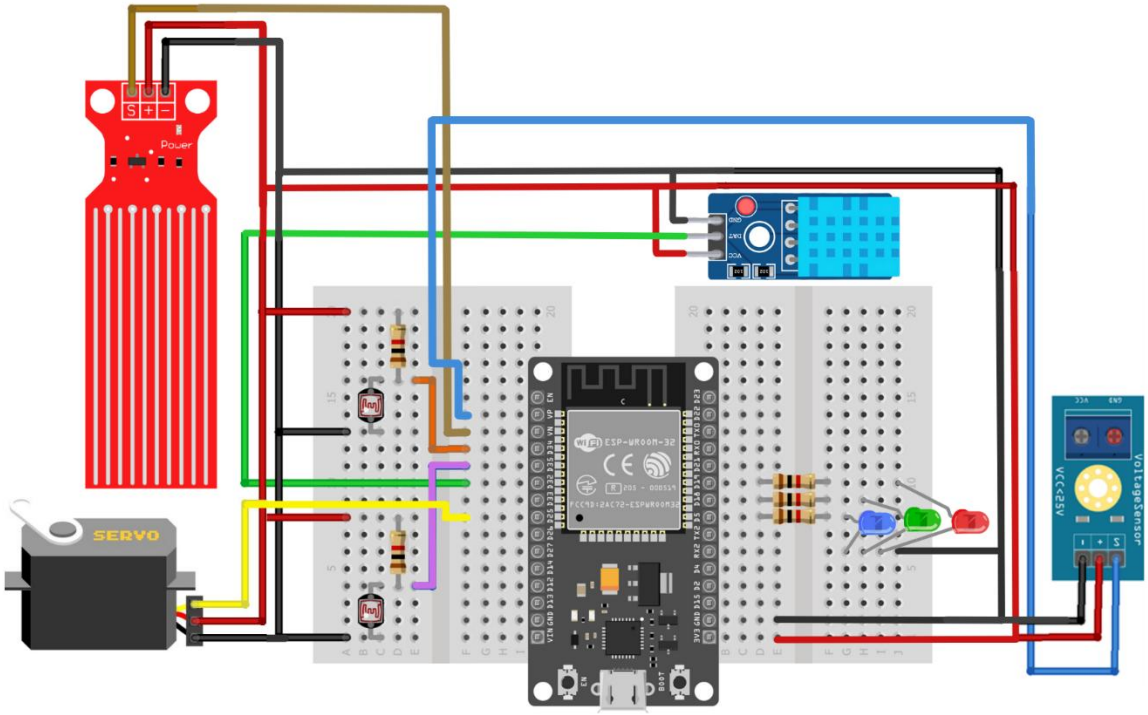


We will check the operation of the servo motor by uploading the following code to the ESP32, which sweeps the servo between the positions of 0°, 90°, and 180° in a loop repeatedly.

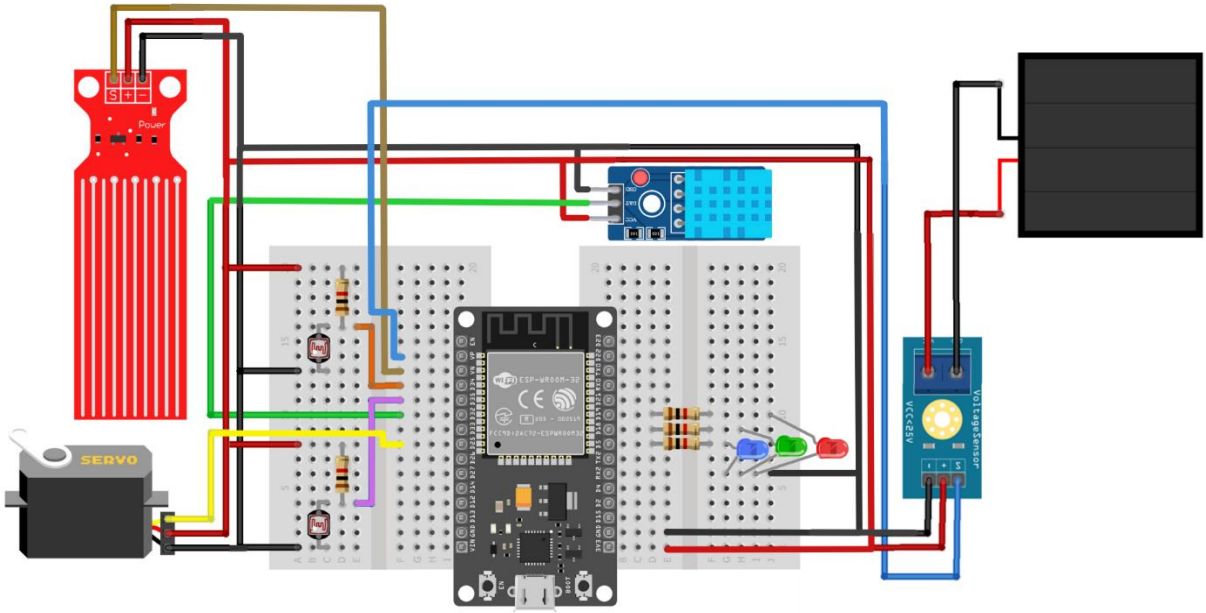
```
#include <ESP32Servo.h> // Include the necessary library to control the servomotor

#define servoPin 25 // Define the pin where the servomotor is connected

// We create an object of type Servo to control it
Servo myServo;

// Initialization block
void setup() {

    // We associate the servo we created to the pin where it is connected
    myServo.attach(servoPin);
    // We place the servo in an initial position of 0°
    myServo.write(0);

}

// Main loop of the code
void loop() {

    myServo.write(0); // We move the servomotor to 0°
    delay(1000); // We wait 1 second = 1000 milliseconds
    myServo.write(90); // We move the servomotor to 90°
    delay(1000); // We wait 1 second
    myServo.write(180); // We move the servomotor to 180°
    delay(1000); // We wait 1 second

}
```

Note: Attach the helix to the servo. Screws are not needed to secure it.

The last component we will connect to the ESP32 will be a voltage detector. This module should be powered through both power rails (Vcc and GND). The signal line will be connected to **pin VP**.



The final step to complete the electronic assembly is to connect the solar panel to the voltage detector. Below, you can see how the complete assembly will look.



We will check the functionality of the complete solar detector by using the following code, which simply reads the signal from the voltage detector module and turns on the Blue LED if it detects voltage.

```
#define ledPin1Cold 5 // Define the pin where the Blue LED is connected
#define detectorPin 36 // Define the pin where the Voltage Detector is connected

// Initialization block
void setup() {

    // We configure the operating mode of the LED pin as an output
    pinMode(ledPin1Cold, OUTPUT);

}

// Main loop of the code
void loop() {

    // We take the reading of the Detector Module
    bool tension_detectada = digitalRead(detectorPin);
    // We check if there is tension or not
    if(tension_detectada) { // Yes, there is tension

        digitalWrite(ledPin1Cold, HIGH); // We turn on the Blue LED

    } else { // No tension

        digitalWrite(ledPin1Cold, LOW); // We turn off the Blue LED

    }

}
```

Below is a test code to verify that all the sensors are getting readings and working correctly:

```
//include the necessary libraries
#include <ESP32Servo.h>
#include "DHT.h"

// Pin definitions and constants
#define ldrPin1 34 // set ldr 1 Analog input pin of East ldr as an integer
#define ldrPin2 35 // set ldr 2 Analog input pin of West ldr as an integer
#define DHTPIN 32
#define DHTTYPE DHT11
```

```
#define ledPin1Cold 5
#define ledPin2Good 18
#define ledPin3Heat 19
#define SolarPin 36

// Global variables
DHT dht(DHTPIN, DHTTYPE);
Servo myservo;
int valldr1 = 0 ;
int valldr2 = 0 ;
int averageLdrValue = 0;
int servopos = 90; // initial position of the Horizontal movement controlling servo motor
int tolerance = 20; // allowable tolerance setting - so solar servo motor isn't constantly in motion
float lastMicrovolts = 0;
const int WaterSensorPin = 39; // Analog Value from VN PIN
int sensorValue; // Variable to store sensor values

void setup(){
Serial.begin(115200);
analogReadResolution(12); // ESP32 has a 12-bit ADC
analogSetAttenuation(ADC_11db); // Set attenuation for higher sensitivity
pinMode(4, INPUT);

dht.begin(); // Start the DHT sensor

pinMode(ledPin1Cold, OUTPUT);
pinMode(ledPin2Good, OUTPUT);
pinMode(ledPin3Heat, OUTPUT);

myservo.attach(25); // attaches the servo on digital pin 2 to the horizontal movement servo motor

pinMode(ldrPin1, INPUT); //set East ldr pin as an input
pinMode(ldrPin2, INPUT); //set West ldr pin as an input
myservo.write(servopos); // write the starting position of the horizontal movement servo motor

delay(1000); // 1 second delay to allow the solar panel to move to its staring position before commencing solar tracking
}

void loop(){
```

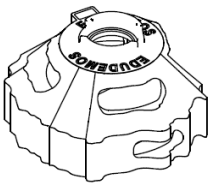
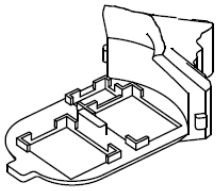
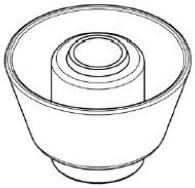
```
  Serial.println("----------------------------- Measured Solar (V) ------------------
----------  ");
   int solarVoltagedV = analogRead ( SolarPin); //dV
 int solarVoltage = solarVoltagedV * 10;

  Serial.print("Voltage of Solar Panel: ");
  Serial.println(solarVoltage);

  // Calculate voltage
  float voltage = (solarVoltage / 4095.0) * 3.3;  // For 12-bit ADC (0-4095)
  Serial.print("Measured voltage (V): ");
  Serial.println(voltage);  // Display the measured voltage

  // Convert to microvolts
  float microvoltsSolar = voltage * 1000000.0;
  Serial.print("Input voltage (µV): ");
  Serial.println(microvoltsSolar);

  // Apply a minimum threshold to filter noise
  const float threshold = 10.0;  // Threshold in microvolts
  if (microvoltsSolar < threshold) {
    microvoltsSolar = 0;
  }

  // Check if the microvolts value has changed
  if (microvoltsSolar != lastMicrovolts) {
    Serial.print("Send voltage: ");
    Serial.print(microvoltsSolar, 2);  // Display with 2 decimal places
    Serial.println(" µV");

    // Update the last microvolts value
    lastMicrovolts = microvoltsSolar;
  }

  Serial.println("----------------------- Measured Temperature / Humidity -----------
------------  ");
   // Read values from DHT sensor
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // Check if readings are valid
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
```

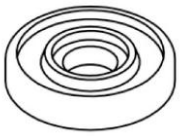
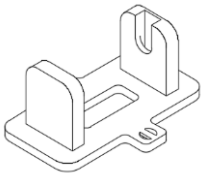
```
  }

  // Print sensor values
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" °C, Humidity: ");
  Serial.print(h);
  Serial.print(" %");

  // Example LED control based on temperature
  if (t < 20) {
    digitalWrite(ledPin1Cold, HIGH);
    digitalWrite(ledPin2Good, LOW);
    digitalWrite(ledPin3Heat, LOW);
    Serial.println("Cold is HIGH");

  } else if (t >= 20 && t < 25) {
    digitalWrite(ledPin1Cold, LOW);
    digitalWrite(ledPin2Good, HIGH);
    digitalWrite(ledPin3Heat, LOW);
    Serial.println("Good is HIGH");
  } else {
    digitalWrite(ledPin1Cold, LOW);
    digitalWrite(ledPin2Good, LOW);
    digitalWrite(ledPin3Heat, HIGH);
    Serial.println("Heat is HIGH");
  }

  Serial.println("----------------------------- Measured Water Level ----------------
-------------  ");
  int sensorValue; // Variable to store sensor values
  sensorValue = analogRead(WaterSensorPin); // Read the value from the soil moisture
sensor
  //Check watersensor calibration first
  if (sensorValue >= 2000) { //These value represents de max value given
    sensorValue = 1980; //This value would return a 99% value
  }
  float waterlevel = (sensorValue*100)/2000 ; //Calculation to return the %

  Serial.print("Water level: "); // Print value in serial monitor
  Serial.print(waterlevel); // Print value in serial monitor
  Serial.println(" %"); // Print value in serial monitor
```

```
  Serial.println("---------------------------- Measured LDR and Servo --------------
---------------  ");
    // Read LDR values
  valldr1 = analogRead(ldrPin1); // read the value of ldr 1
  valldr2 = analogRead(ldrPin2); // read the value of ldr 2
  // Invert readings so low values mean darkness and high values mean light

  averageLdrValue = (valldr1 + valldr2) / 2;

  // Print readings for debugging
  Serial.print("LDR 1: ");
  Serial.print(valldr1);
  Serial.print(" | LDR 2: ");
  Serial.println(valldr2);
  Serial.print("Average LDR: ");
  Serial.println(averageLdrValue);

if((abs(valldr1 - valldr2) <= tolerance) || (abs(valldr2 - valldr1) <= tolerance)) {
//no servo motor horizontal movement will take place if the ldr value is within the
allowable tolerance
}else {
if(valldr1 > valldr2) // if ldr1 senses more light than ldr2
{
servopos = servopos+10; // decrement the 90 degree position of the horizontal servo
motor - this will move the panel position Eastward
}
if(valldr1 < valldr2) // if ldr2 senses more light than ldr1
{
servopos = servopos-10; // increment the 90 degree position of the horizontal motor -
this will move the panel position Westward
 }
}

if(servopos > 180) {servopos = 180;} // reset the horizontal position of the motor to
180 if it tries to move past this point
if(servopos < 0) {servopos = 0;} // reset the horizontal position of the motor to 0 if
it tries to move past this point
myservo.write(servopos); // write the starting position to the horizontal motor

  // Print the servo angle
  Serial.print("Servo angle: ");
  Serial.print(servopos);
  delay(5000);
}
```

1. 2. 3. Mechanical Assembly of the Water and Solar Demonstrator

The elements that make up this demonstrator are as follows:

Element	Quantify	Description	Image
Base	1	It is the upper part that secures the moving components, such as the blades and the motor shaft. It provides stability and allows the system to operate smoothly, facilitating the correct transfer of energy from the blades to the generator.	
Drawer	1	It is the base that connects the different sections of the device, providing structural support and ensuring proper alignment between the motor, the blades, and other essential components for its operation.	
Water container	1	This part is capable of storing the water that falls into it.	
Leaf	8	A sheet that allows liquids to enter the water collection container.	
Small disc	3	It is a static component that does not move. Its primary function is to act as part of the structural support of the device, helping to maintain proper alignment and	

		stability of the elements.	
Static disc	2	A disc that remains fixed to cover and maintain the stability of the structure.	
Solar spin	1	A part that allows for the movement of the solar panel for proper orientation.	
Solar disc	1	A disc that connects the solar rotator with an installed servo motor and joins it with the rest of the structure.	

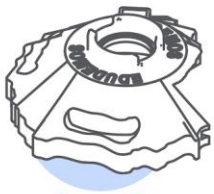
Next, the assembly of the demonstrator is shown:

Note: *The aluminum tape is used on the leaves' surface to optimise the water collection. The double sided tape is used on the back of the solar panel to stick it on the "solar spin" part.*

The water level sensor is placed in some slots inside the "water container" part.

When placing the electronic components into the mechanical assembly, it is necessary to use additional wires to extend the connections of the LDRs and the solar panel.

BASE



1 x1

2 x1

AGUA

3 x1

4 x8

5 x2

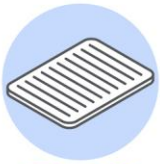
6 x2

SOL

7 x1

8 x1

ELECTRÓNICA

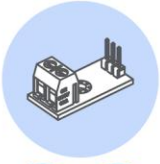


a x1

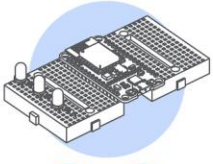
b x2

c x1

d x1

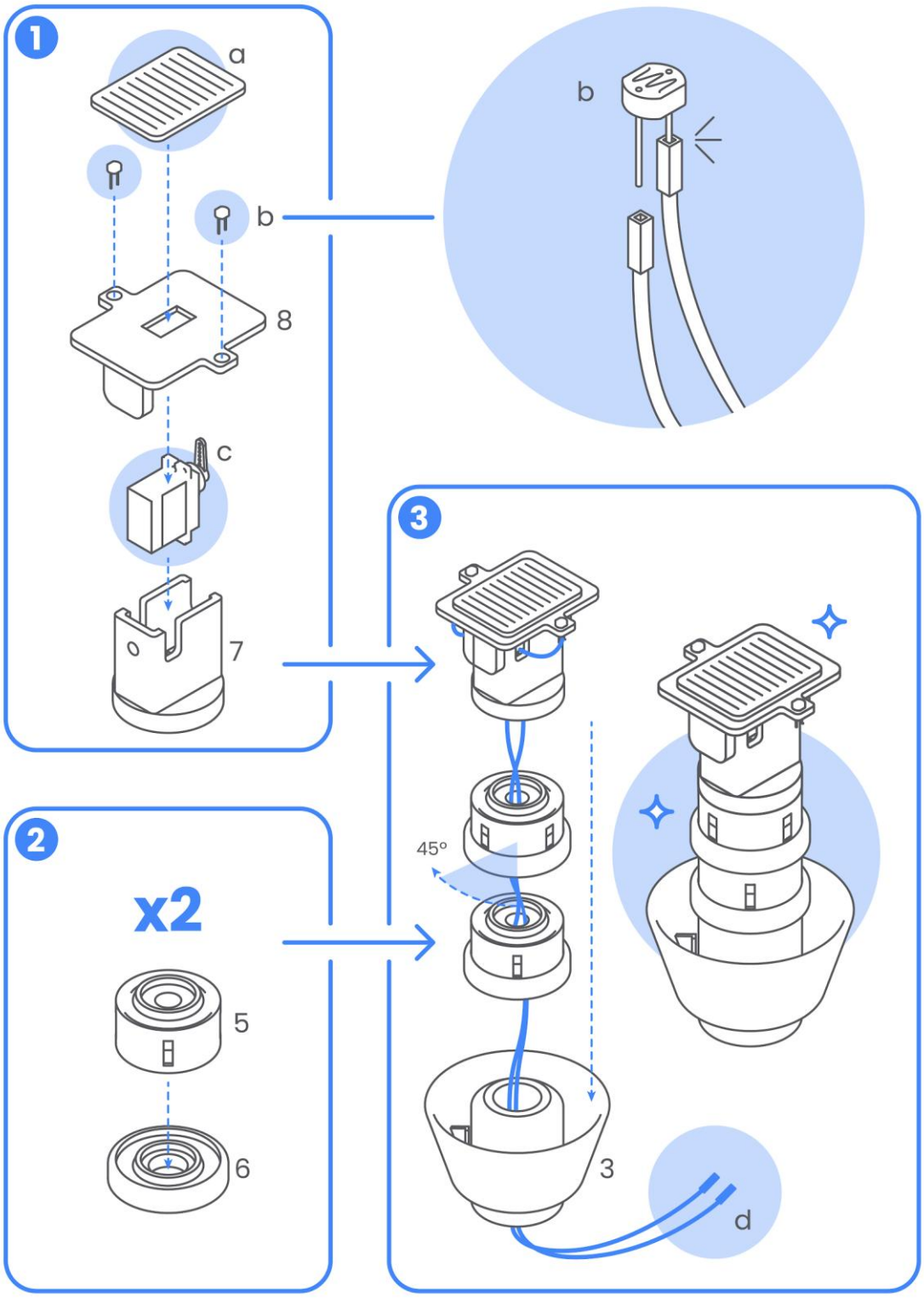


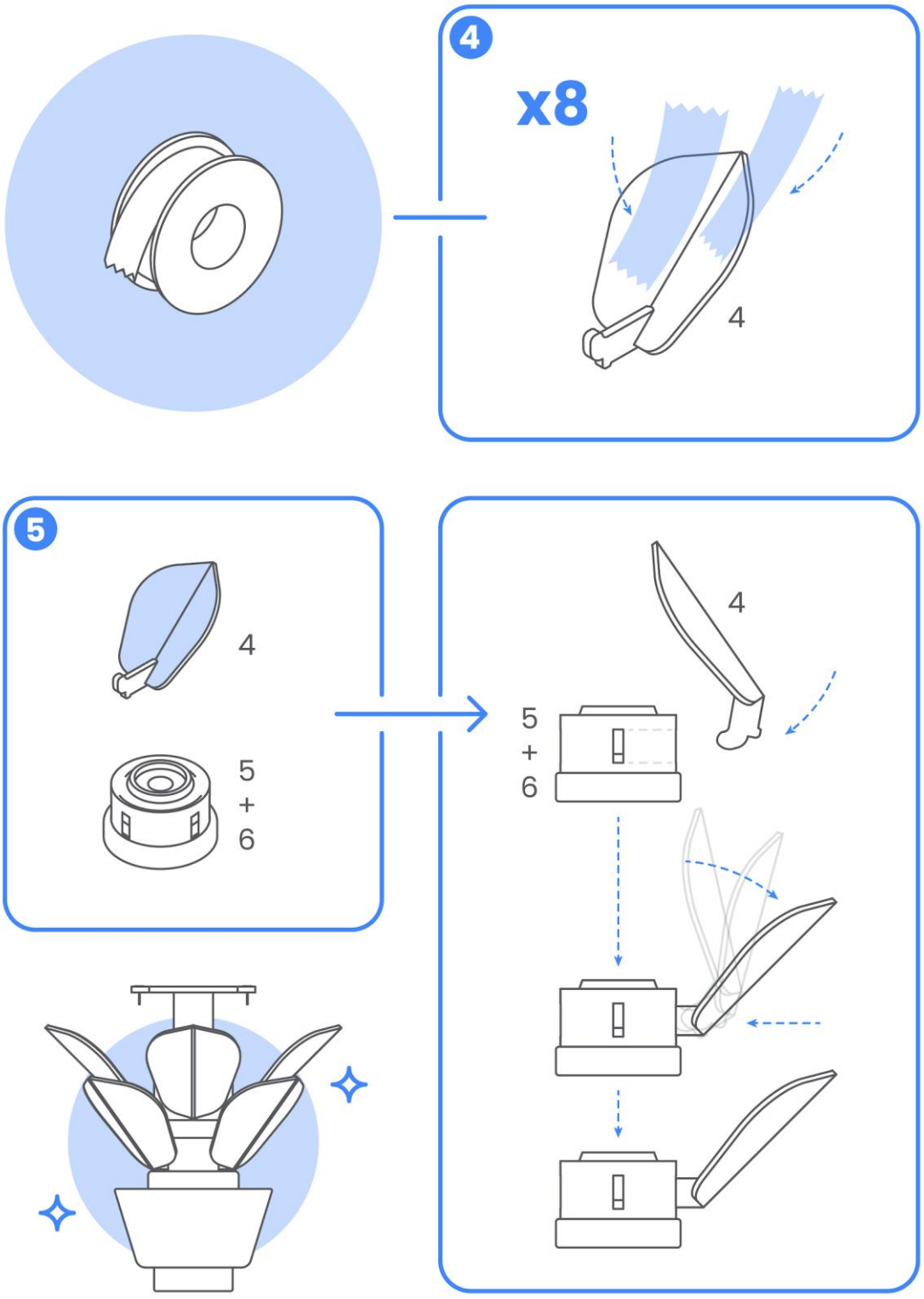
e x1



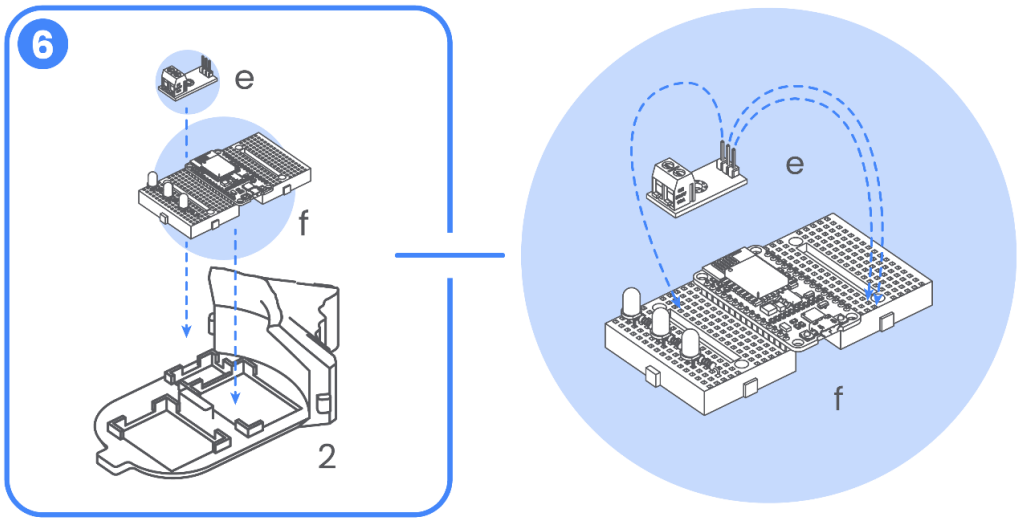
f x1

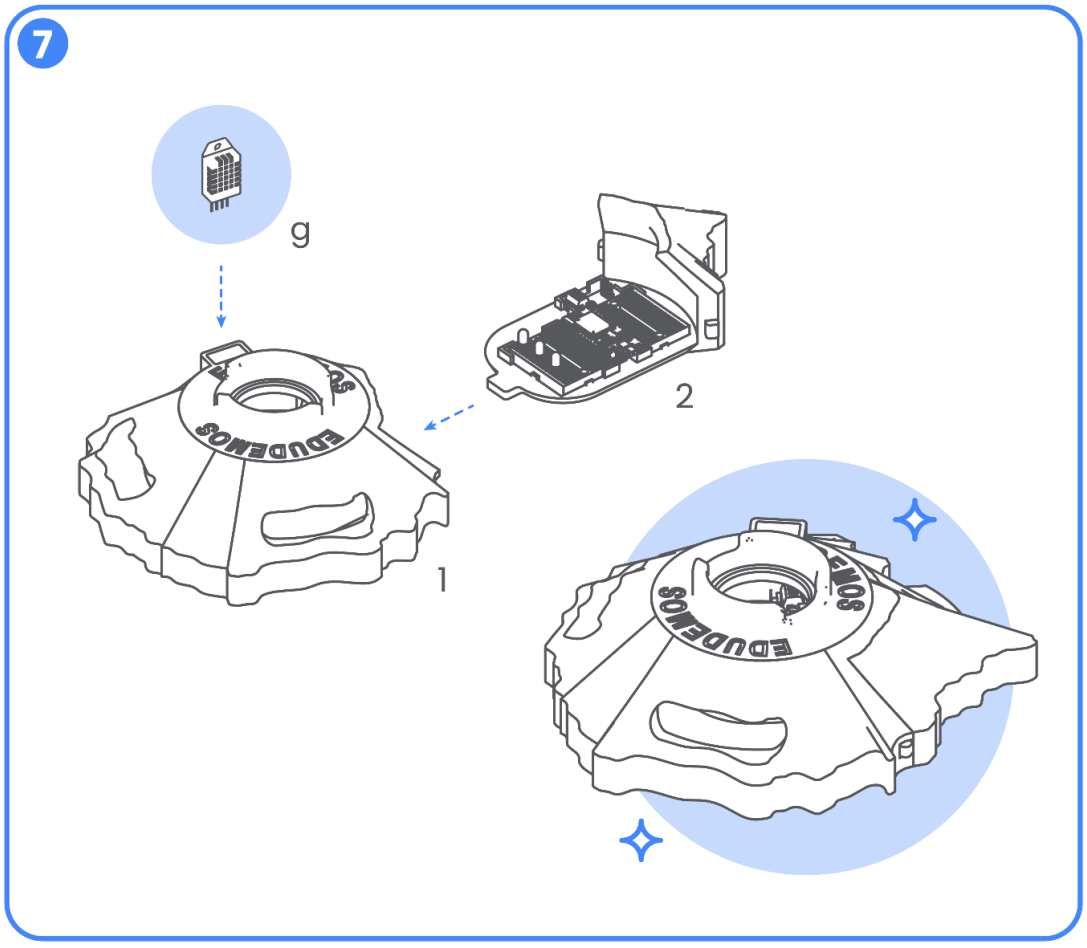
g x1


1
a
b
8
c
7
b
2
x2
5
6
3
45°
3
d

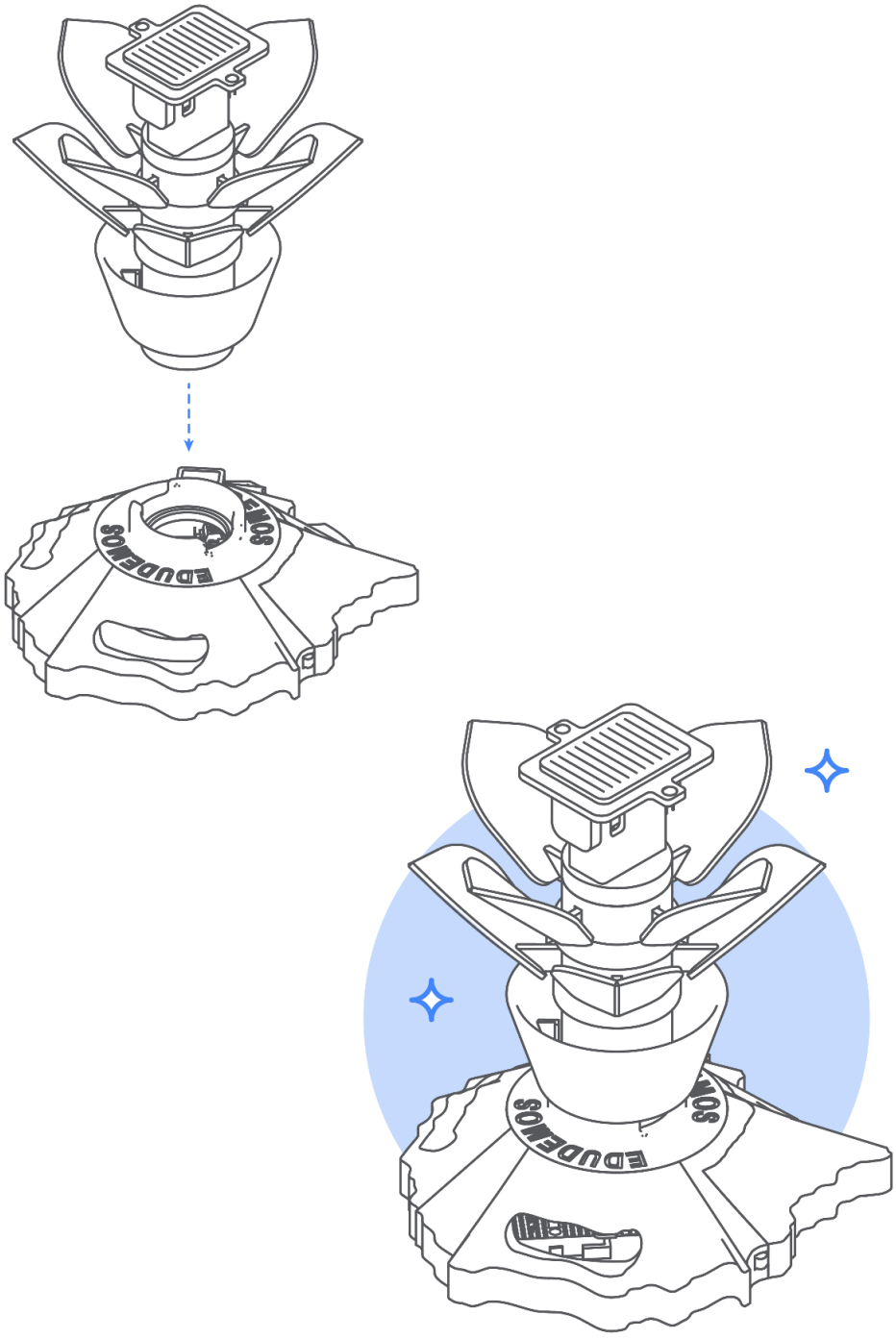


Funded by the European Union
Gerda Stetter Stiftung
Technik macht Spaß!


6
e
f
2
e
f


7
g
2
1

8



Funded by the European Union
Gerda Stetter Stiftung
Technik macht Spaß!
FUNDACIÓN SERGIO ALONSO
Finnova
gbs sg.ch

1. 3. Assembly of the Wind Demonstrator

1. 3. 1. Electronic Assembly

We will start by performing the assembly of the Wind Demonstrator. To perform this complete assembly you will need the following material:

1 x 5V DC Motor

Used to generate voltage with wind power.

1 x 100nF / 1uF Capacitor

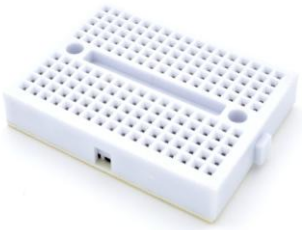
Used to eliminate unwanted noise generated by the DC motor.

2 x 1 kΩ Resistors

Used in the voltage divider that allows reading the voltage generated by the DC motor.

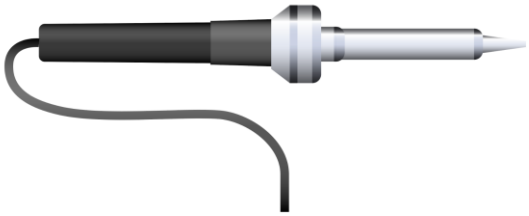
6 x Dupont Cable (Male-Male) 10 cm

Necessary for making connections between the different components on the breadboard.



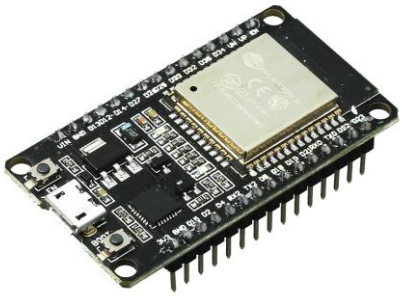
2 x Mini Breadboard

Component necessary to connect the microcontroller board to the different components.



Soldering iron

Necessary if the motor doesn't have wires included

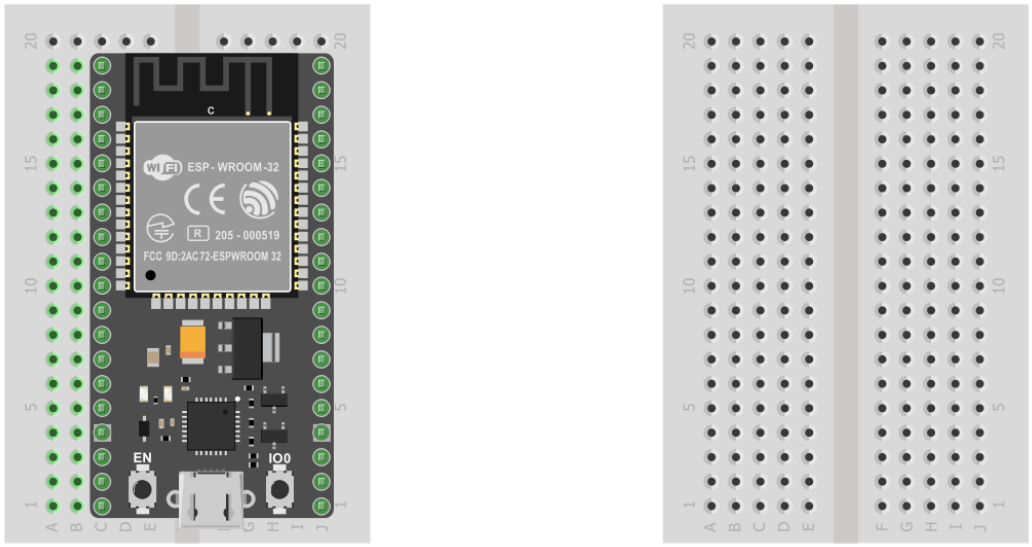


1 x ESP32 DEVKIT V1 (30 pins)

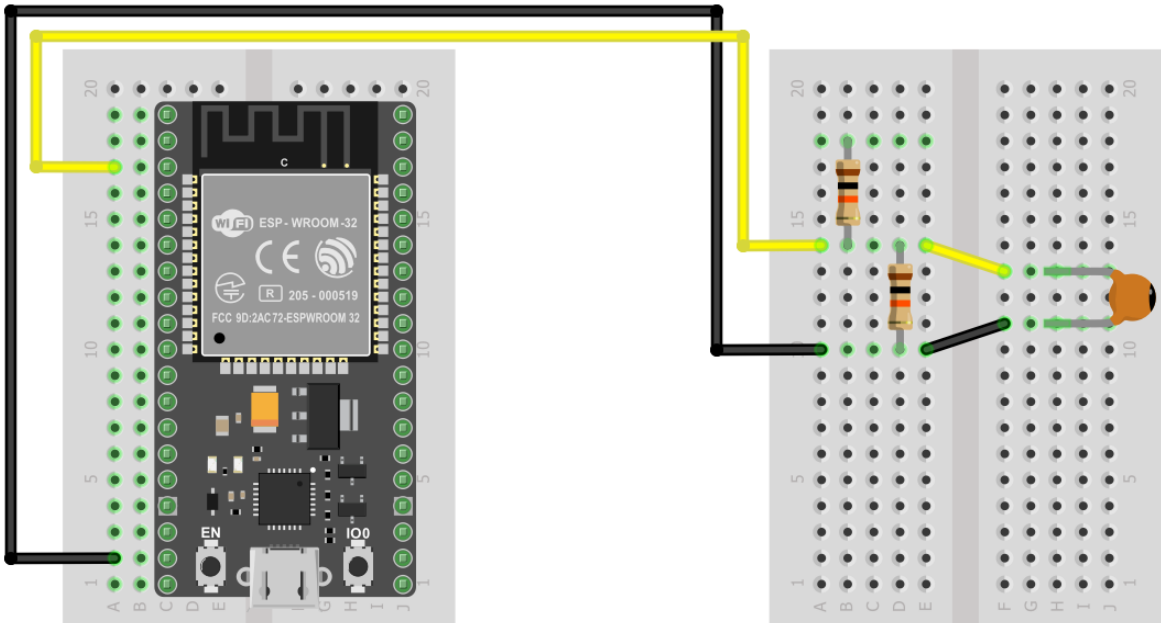
Responsible for executing the programming that controls the operation of the electronics.

The objective of the setup will be to measure the amount of voltage generated by the motor, which will be attached to a mechanical structure that will rotate due to wind power. To measure the voltage obtained from the movement of the blades of the Demonstrator using the ESP32, we need to assemble a voltage divider to reduce the maximum voltage level (5V) to one suitable for the ESP32 (which operates at 3.3V). This way, we will be able to determine how much voltage is being generated by the wind.

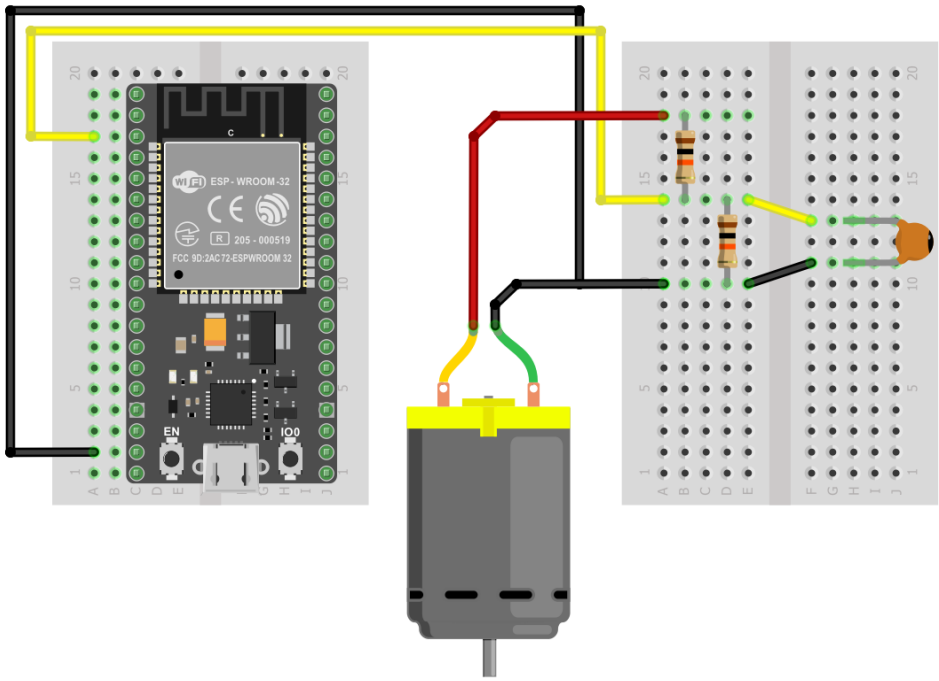
For this new setup, it will be necessary to create a circuit from scratch. We will begin by placing the ESP32 on two mini breadboards. The corresponding setup is shown below.



Next, a voltage divider will be assembled. The midpoint of this divider will be connected to **pin VN**. Additionally, a capacitor must be connected between the terminal of this pin and GND, which will act as a filter for unwanted variations in the measurement, as shown in the image below.



The final step in the electronic assembly of this demonstrator is to connect the motor to the previous setup. One of its terminals should be connected to GND, and the other terminal should be connected to the input of the voltage divider assembled earlier. The completed setup is presented below.

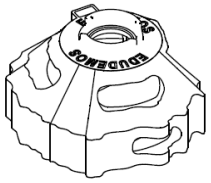
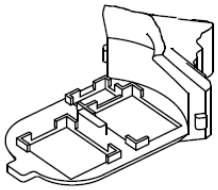
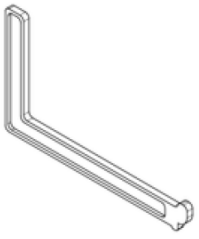


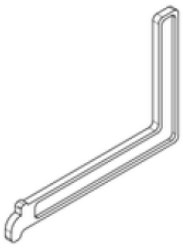
Let's check that the setup of this Demonstrator works correctly by using the following code snippet, where we only read the output of the voltage divider connected to the ESP32. By rotating the motor's rotor, we will see how the measurement gives different values through the Serial Monitor.

```
// Initialization block
void setup() {
    // We initialize the Serial Monitor
    Serial.begin(115200);
}
// Main loop of the code
void loop() {
    // We take the measurement from the pin connected to the output of the voltage
    //divider
    int medicion = analogRead(39);
    // We display the measurement on the Serial Monitor
    Serial.println(medicion);
}
```

1. 3. 2. Mechanical Assembly

The components of this demonstrator are as follows:

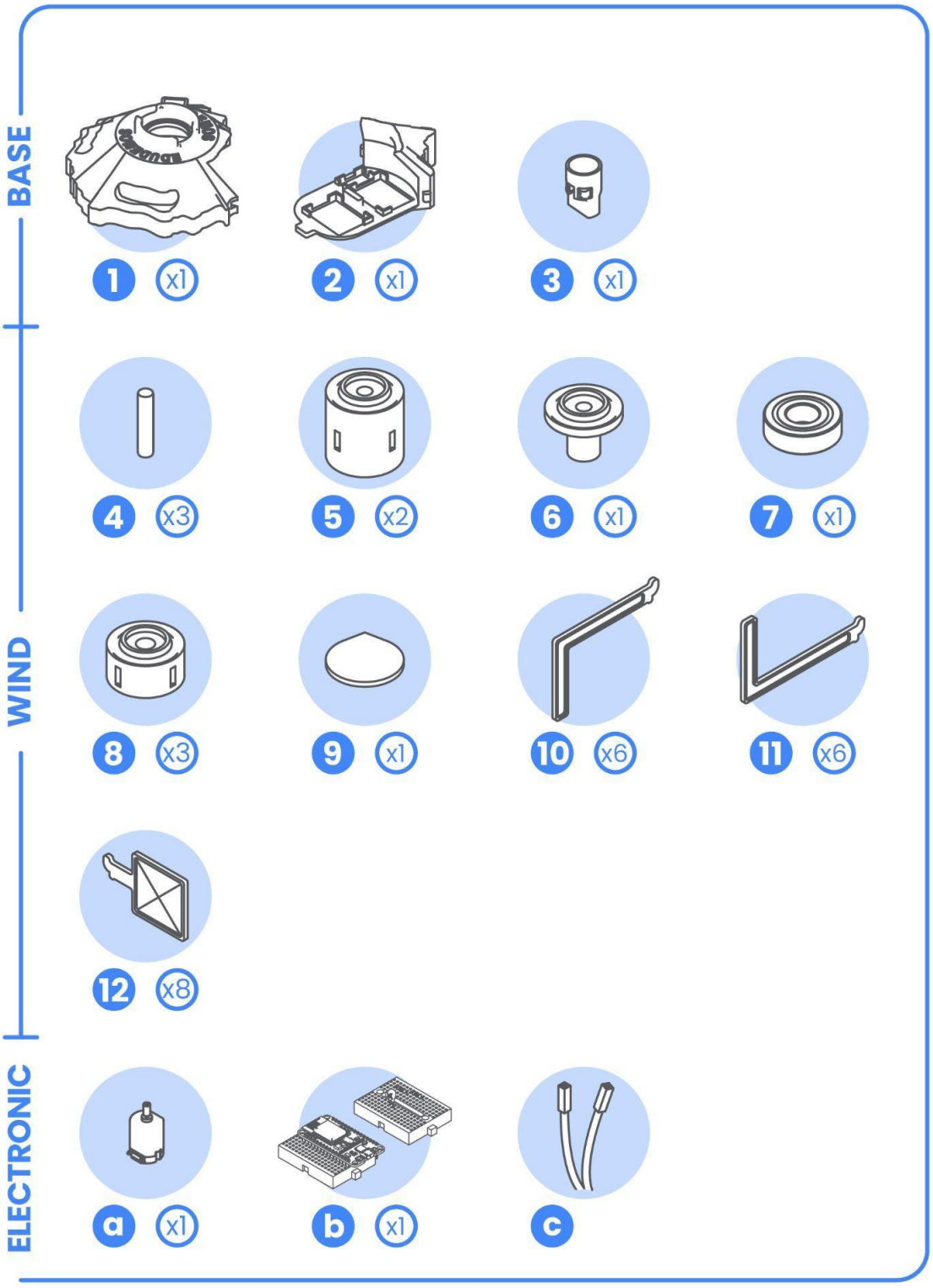
Elemento	Cantidad	Descripción	Imagen
Base	1	It is the top section that secures the moving components, such as the blades and the motor shaft. It provides stability and allows the system to operate smoothly, facilitating the proper transfer of energy from the blades to the generator.	
Drawer	1	It is the base that connects the different sections of the device, providing structural support and ensuring proper alignment between the motor, blades, and other essential components for its operation.	
Motor socket	1	This component will support and stabilize the motor, whose shaft will be connected to the bearing disc. This design will allow the connection of output pins, which will emit signals indicating the presence of movement.	
Big L blade	6	It is a fixed blade that does not move. It is designed to be part of the static structure of the demonstrator, contributing to the overall stability and balance of the system.	

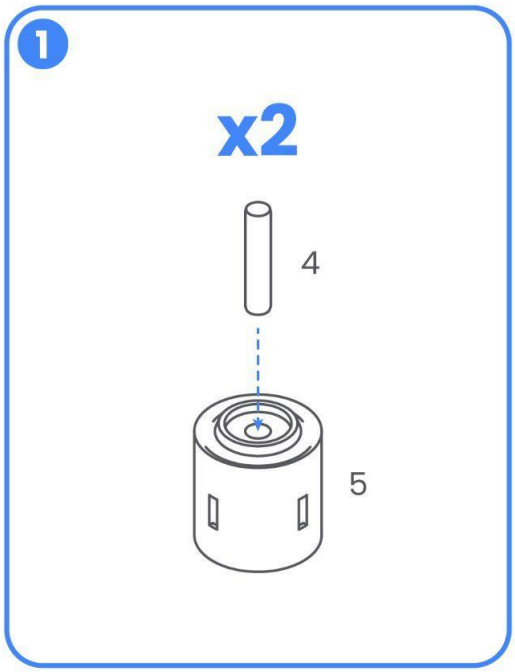
Small L blade	6	It is a fixed blade that does not move, smaller than the previous piece. It is designed to be part of the static structure of the demonstrator, contributing to the overall stability and balance of the system.	
Square blade	8	It is one of the key moving blades that rotates when driven by the wind. Its square design efficiently captures the wind's kinetic energy and converts it into rotational movement.	
Big disc	2	It is a key component that serves as a structural support for the moving blades. This disk is designed to rotate when the blades capture the wind, transferring the rotational movement to the system's central shaft.	
Small disc	3	It is a static component that does not move. Its main function is to act as part of the device's structural support, helping to maintain the proper alignment and stability of the moving elements, such as the blades and large disks.	
Bearing disc	1	It is a crucial component used to reduce friction between the system's moving parts. This disk acts as a rotational support, allowing components like the Large Disk and the blades to spin smoothly and efficiently around the central shaft.	

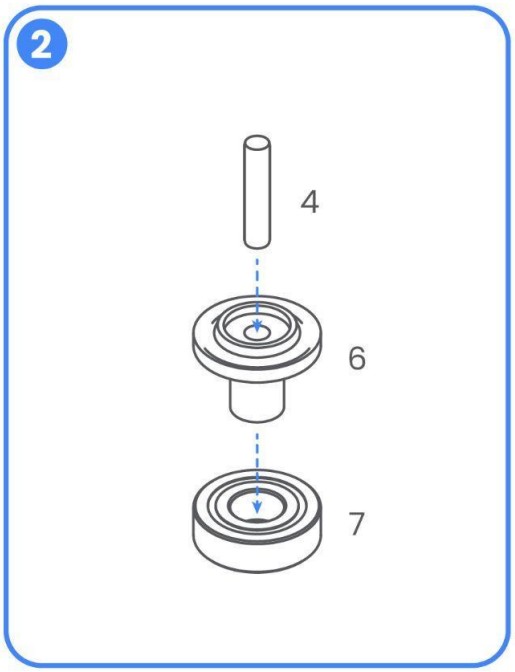
Top	1	It is a structural component placed at the top of the assembly, ensuring that all internal elements are protected and properly aligned.	
Bearing	1	It is a mechanical component that allows rotating elements, such as the disks and blades, to move with minimal friction.	
Wood joints	3	It is a structural component used to connect and secure the different wooden parts of the device.	

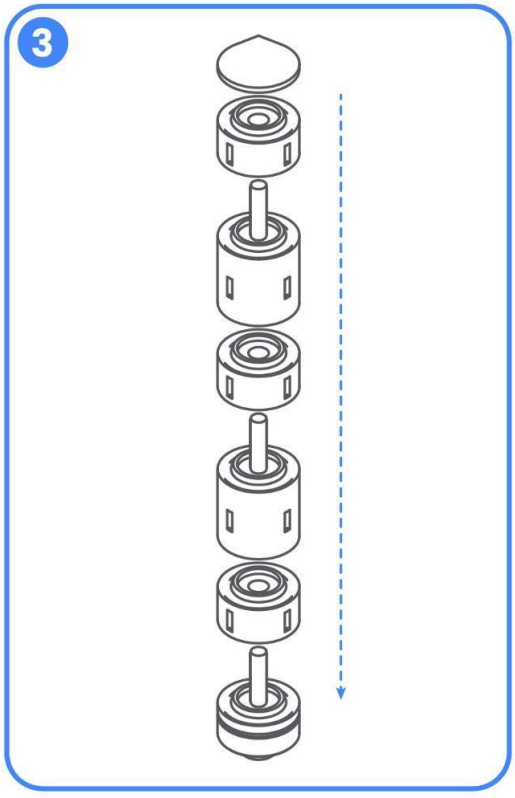
The following shows how the demonstrator assembly is carried out:

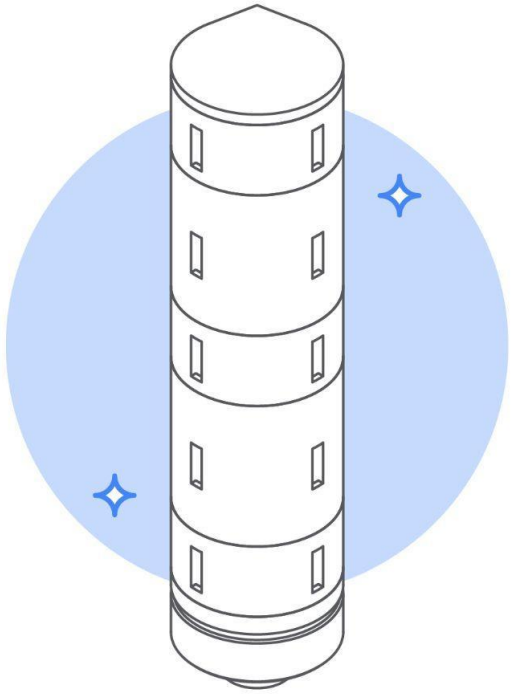
Note: It is necessary to insert the 10-tooth gear onto the DC motor shaft before starting the assembly of the demonstrator.

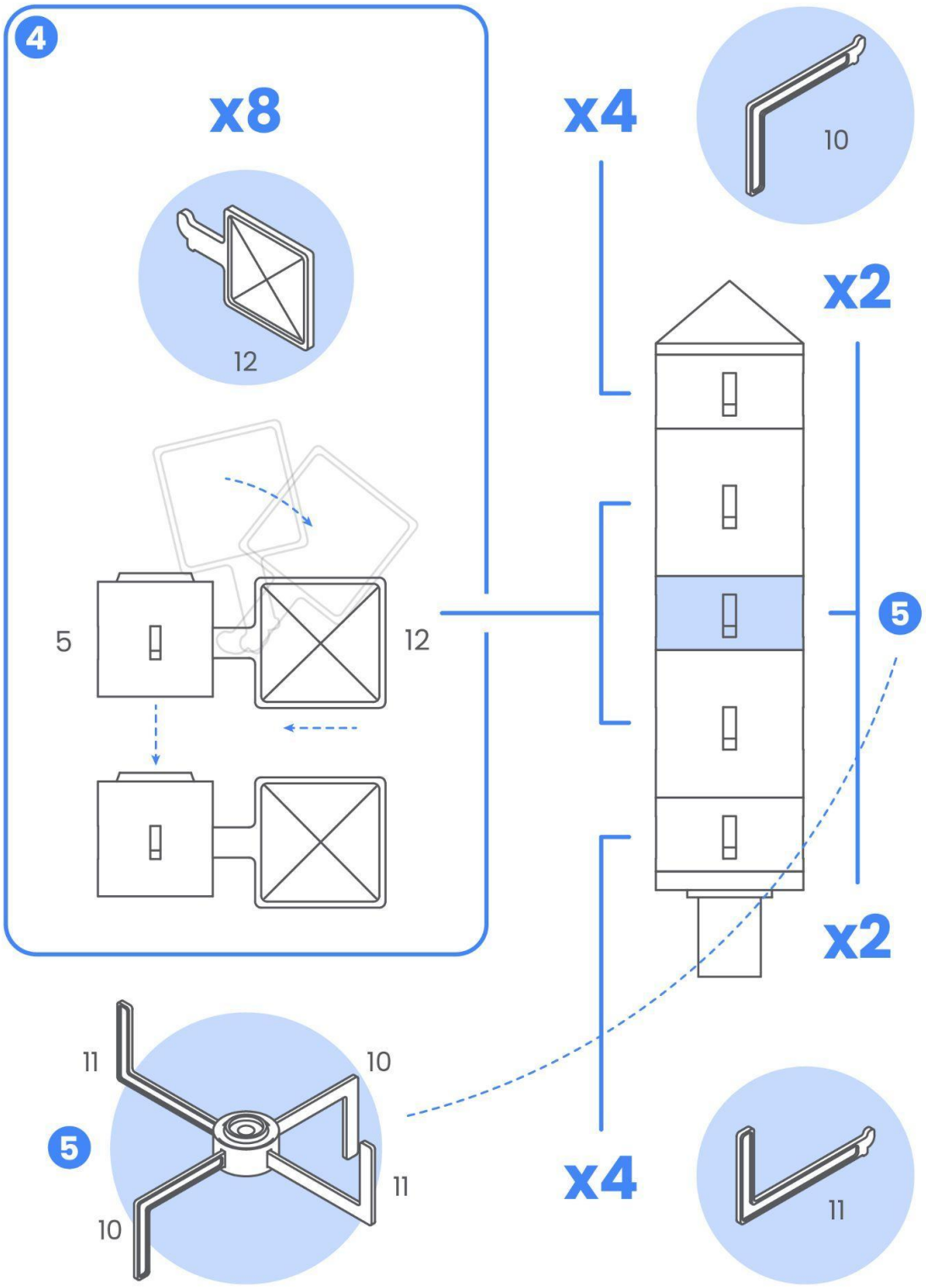

BASE
WIND
ELECTRONIC
1 x1
2 x1
3 x1
4 x3
5 x2
6 x1
7 x1
8 x3
9 x1
10 x6
11 x6
12 x8
a x1
b x1
c

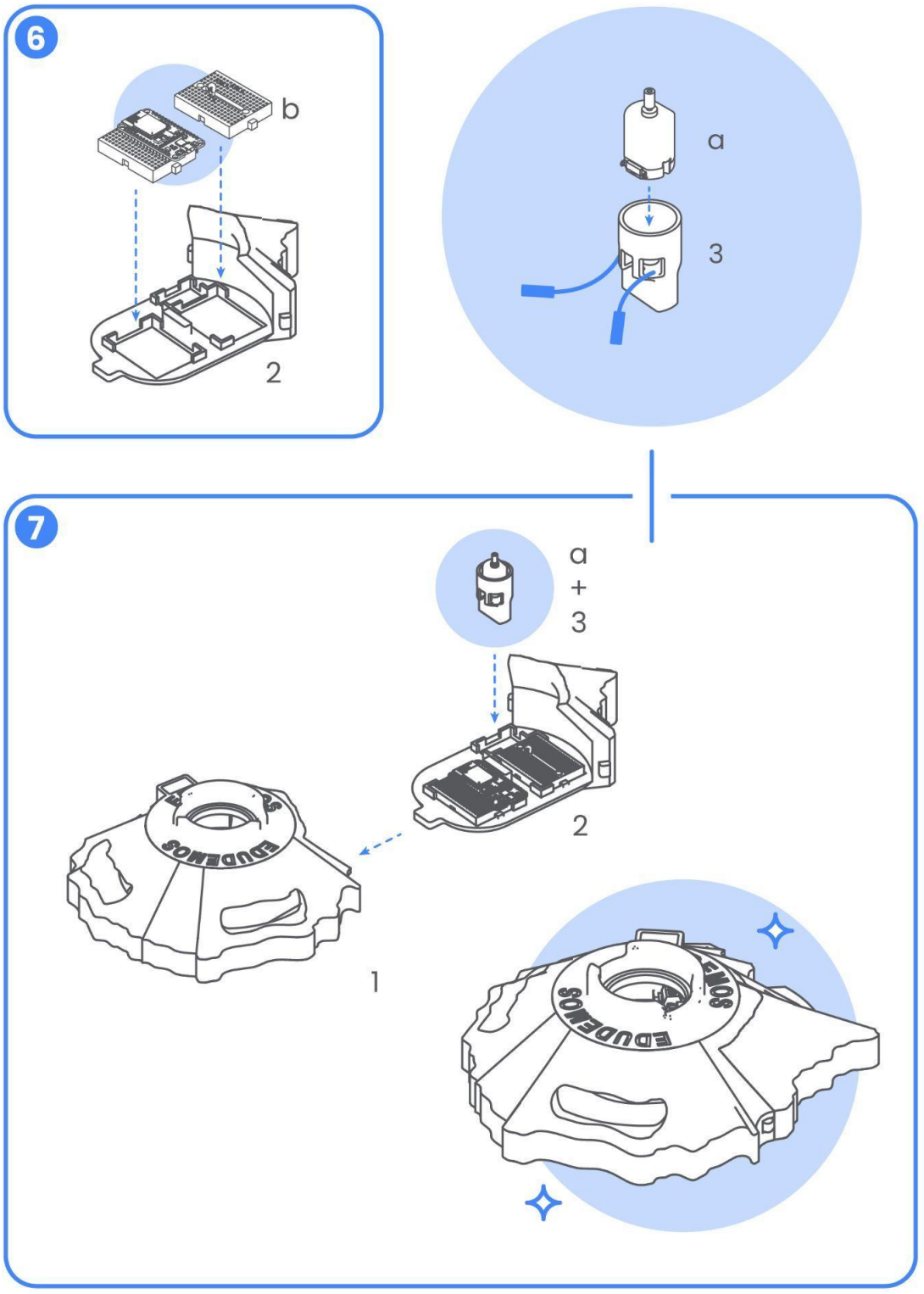

1
x2
4
5


2
4
6
7

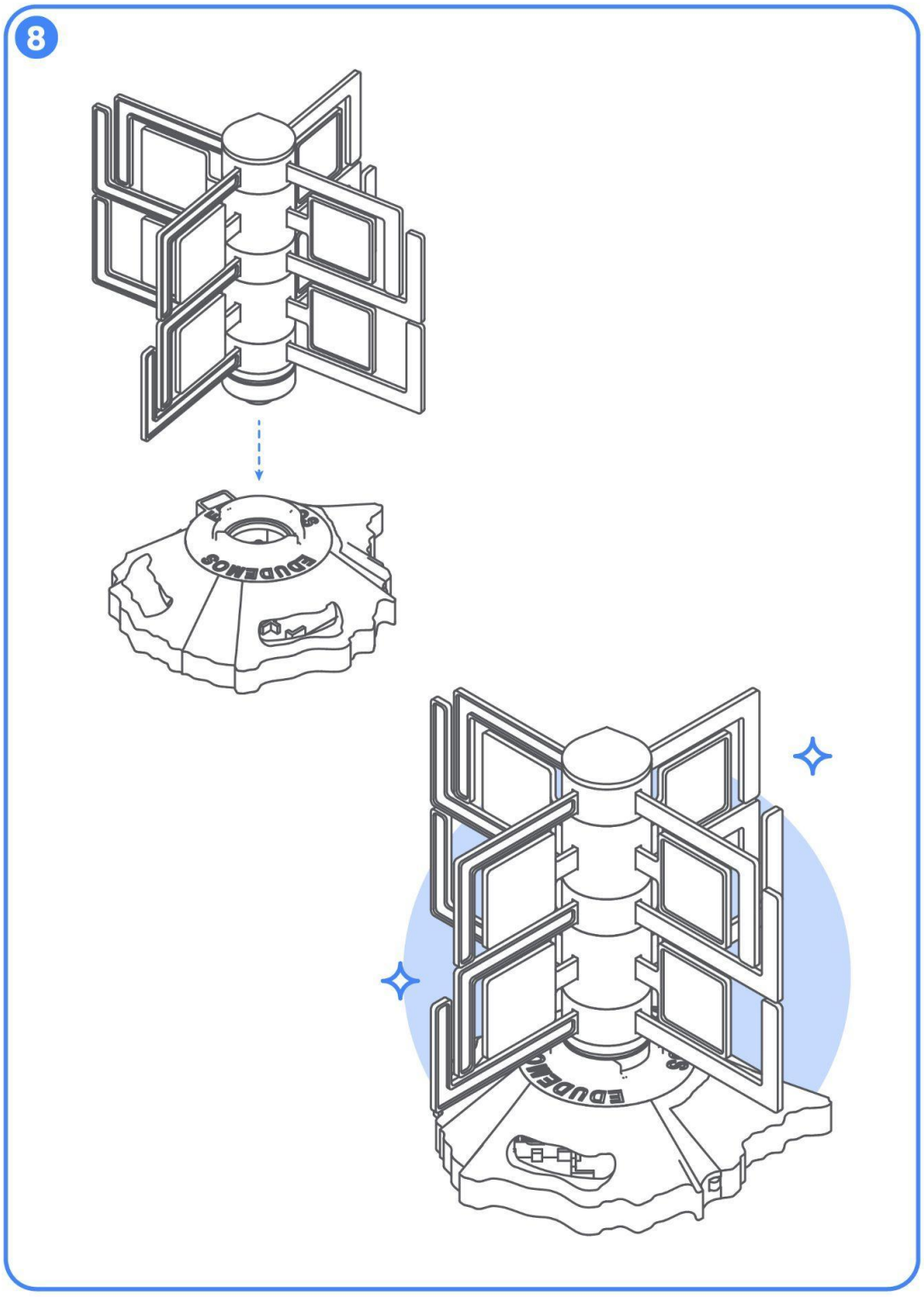

3




4
x8
12
5
12
x4
10
x2
5
x2
11
10
5
10
11
x4
11



Funded by the European Union
Gerda Stetter Stiftung
Technik macht Spaß!
FUNDACIÓN SERGIO ALONSO
Finnova
gbs sg.ch



Funded by the European Union
Gerda Stetter Stiftung
Technik macht Spaß!
FUNDACIÓN SERGIO ALONSO
Finnova
gbs sg.ch

EDUDEMOS

EDUcating through Sustainable DEMOnstrators

Funded by
the European Union

info@edudemos.eu

@edudemos

@edudemos

www.edudemos.eu