AFF IoT Board GUIDE

Version 1.0.2

AFF IoT Guide

Your guide to the AFF IoT Board

Welcome to the *AFF IoT Guide*

This booklet contains information required to explore the circuits of the AFF IoT KIT.

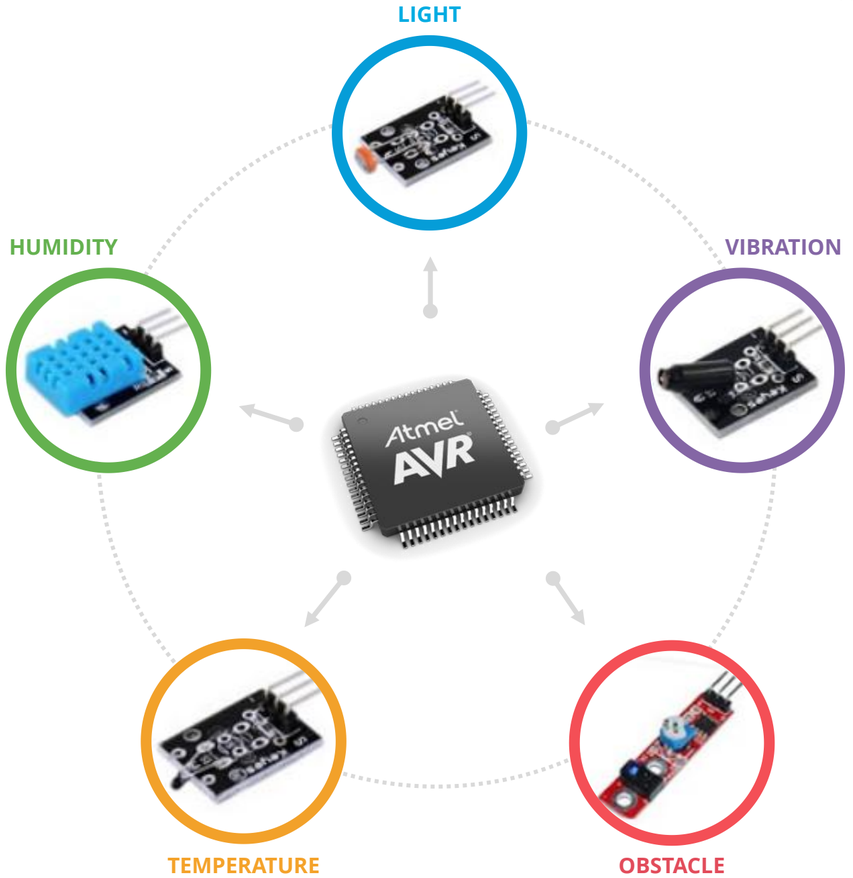
When done with this booklet you'll know how to start creating your own projects and experiments.

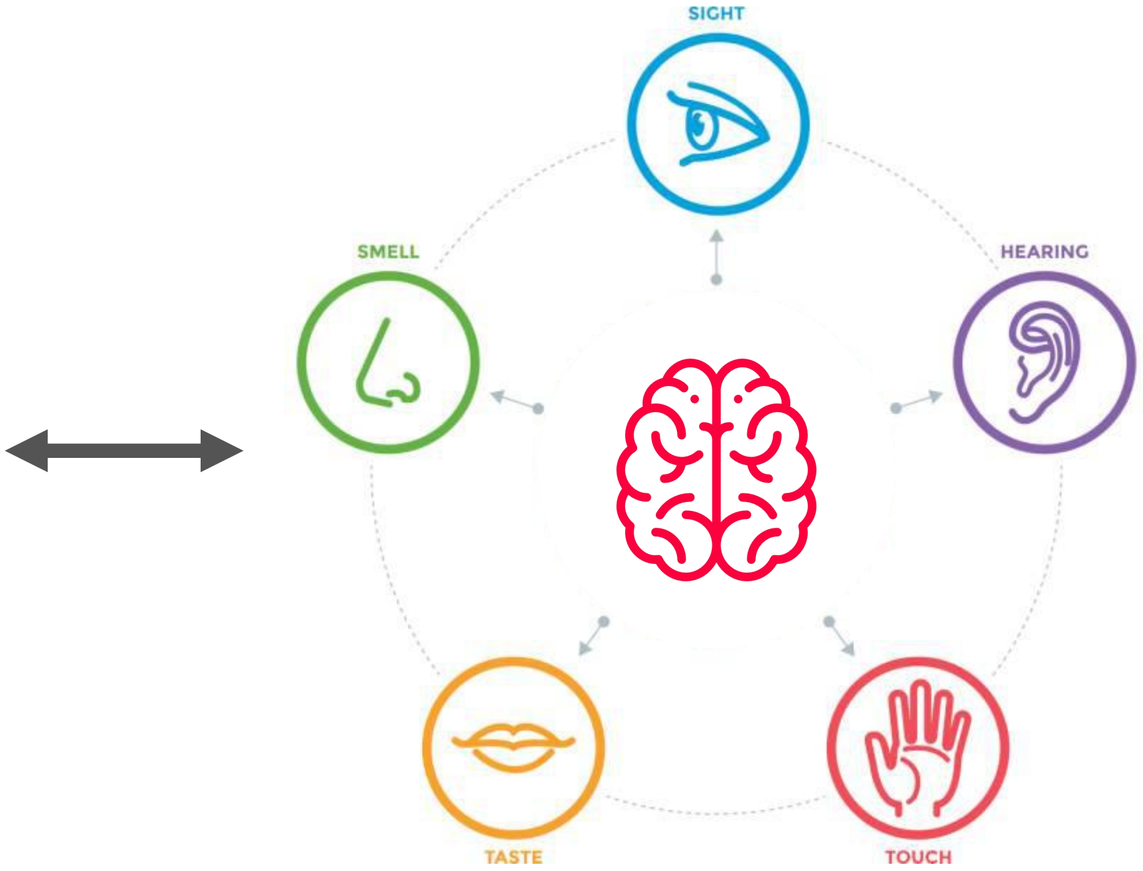
LET'S BEGIN!

Table of Contents

Fundamental concept

Human and microcontroller analogy





A **microcontroller** is a computer presented in a single integrated circuit, dedicated to perform and execute some tasks. It contains memory, programmable input/output peripherals as well as brain.
It gathers data from environment using sensors; processes the collected data according to programmer's needs, and then performs actions accordingly (control a motor, turn light on, etc.)

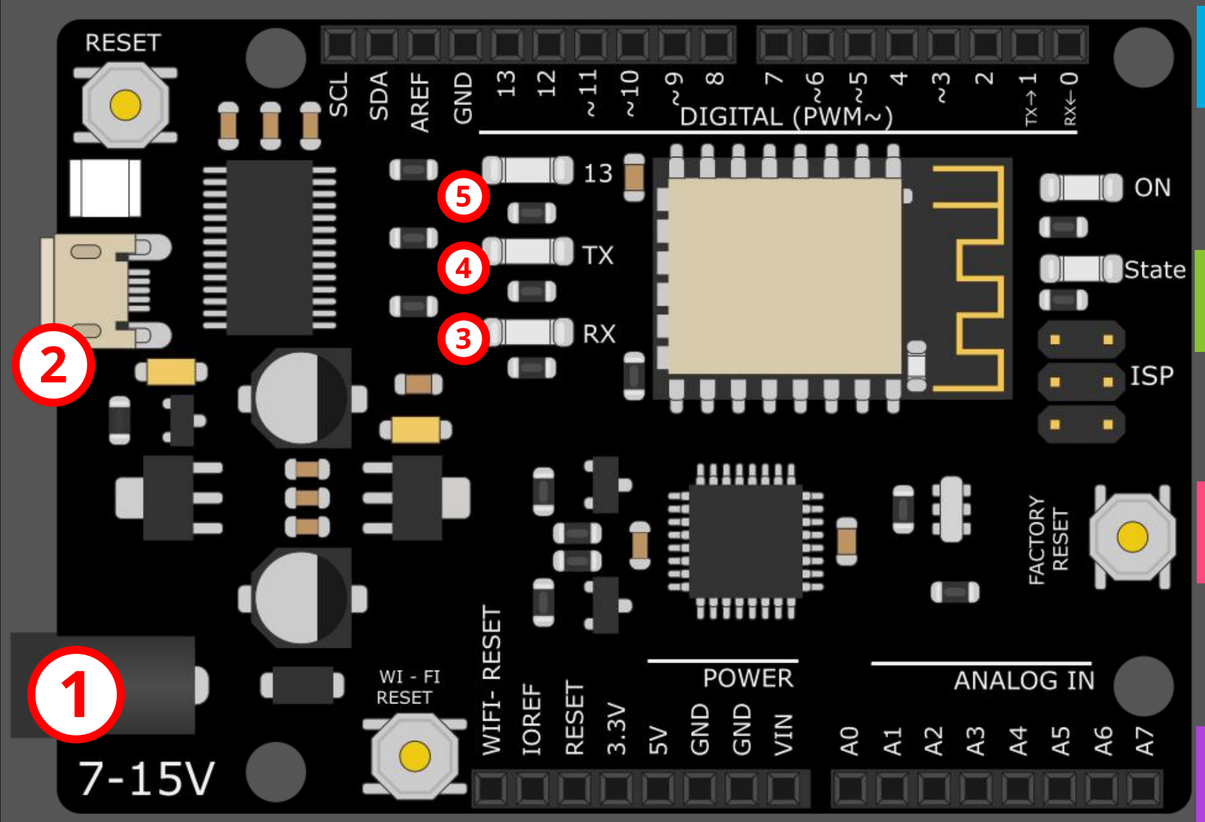
The **brain** is an organ that serves as the center of the nervous system.
It gathers data from environment using the sense organs; processes the collected data according to current needs along with similar past circumstances, and then sends orders to perform actions accordingly (walk, talk, etc.)

What is AFF IoT Board?

- The Internet of Things (IoT) consists of things that are connected to the Internet, to be controlled/monitored at anytime, anywhere. Technically, it consists of integrating sensors and devices that are connected to the Internet over fixed or wireless networks
- IoT devices are part of the larger concept: home automation. It includes lighting, heating, air conditioning, media and security systems. Long term benefits include energy savings by automatically ensuring lights and electronics are turned off when not used.
- AFF IoT Board is an Arduino-compatible microcontroller board that includes embedded Wi-Fi Module.

AFF **IoT Board**

Components



1

Power In (Barrel Jack)

Provides power to the board, works with either a 9V or 12V adaptor or battery. Power supply ranges from 7V to 15V.

2

USB Port

Provides power and connects the board with computer via USB.

3-4

LEDs (RX: Receiving / TX: Transmitting)

These LEDs show when the microcontroller is receiving/transmitting data bits from/to the computer.

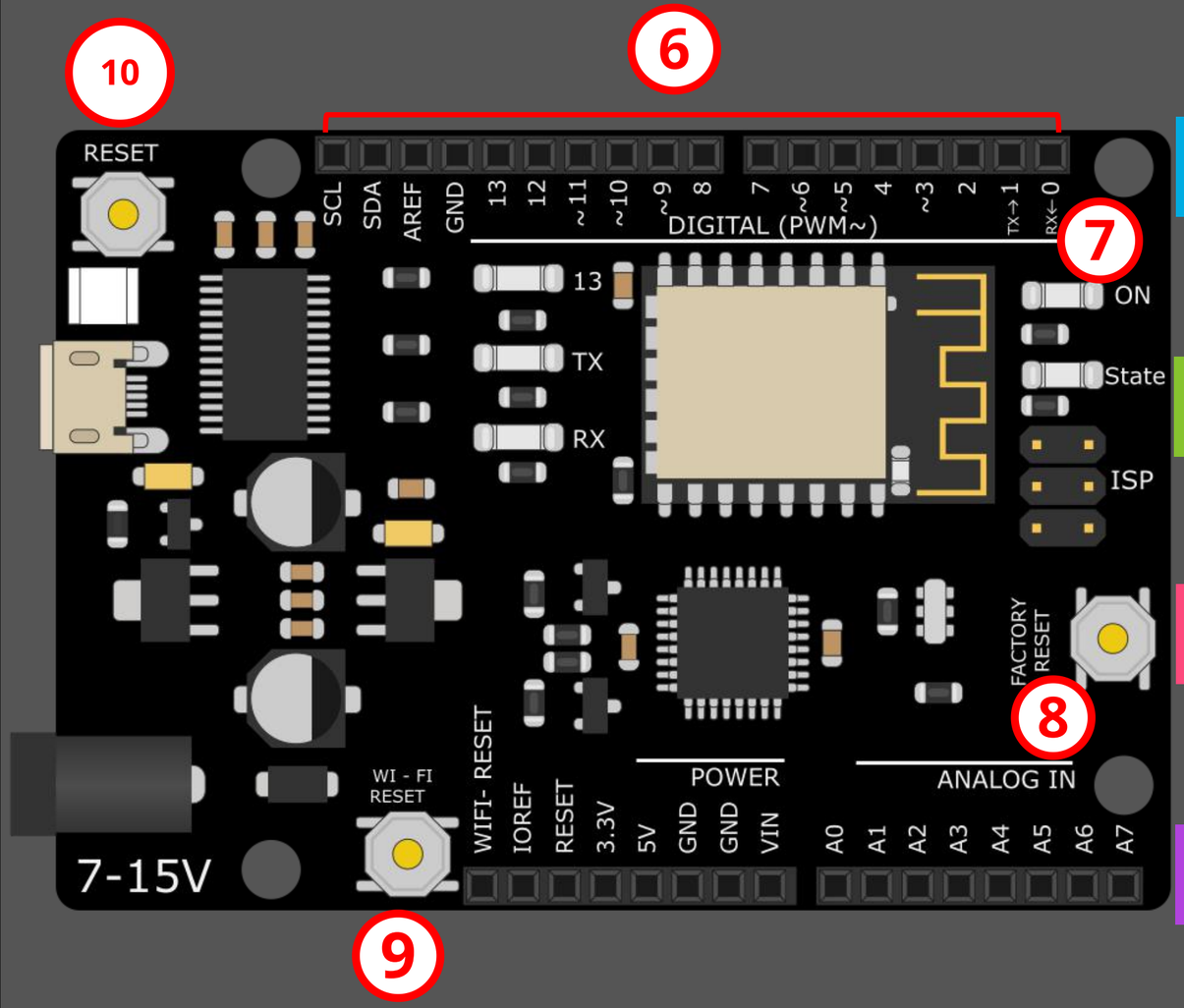
5

LED 13 (Built-in LED)

This LED is linked to the sketch to show if the program is running properly. It is connected via resistor to the digital pin 13.

AFF **IoT Board**

Components



6 Pins (Ground, Digital, Rx, Tx)

Various pins can be used for digital inputs, outputs, and serial communication.

7 LEDs (Red and Blue)

The Red one is a simple power indicator LED. The Blue one indicates the status of the board (configured or not)

8 “FACTORY RESET” Button

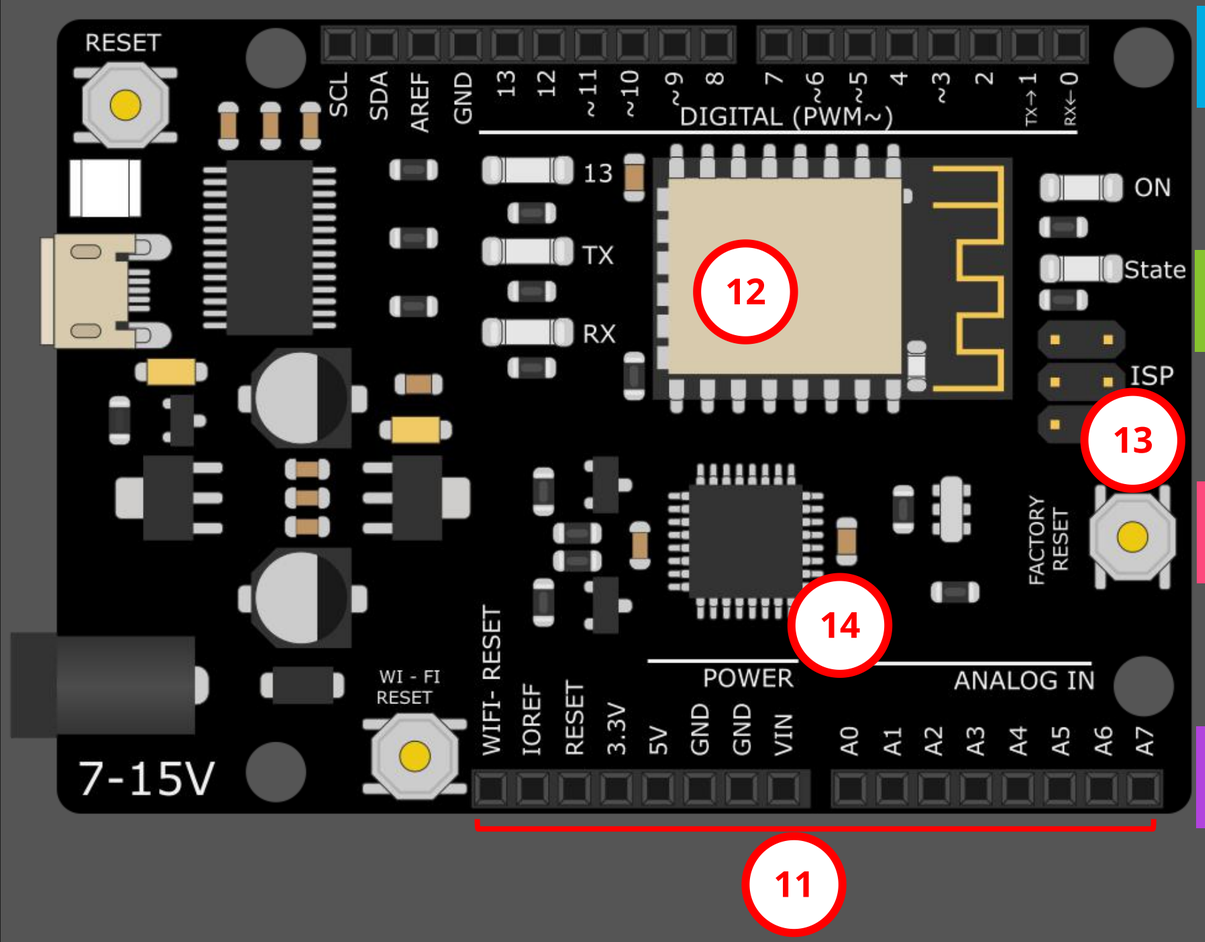
Resets the board, it deletes the Wi-Fi and account data. (press for 5 seconds to reset)

9-10 “Wi-Fi RESET” Button & “RESET” Button

The first one restarts the Board’s Wi-Fi module and reconnects to the server, and the second restarts the microcontroller (restarts the code) (One Press).

AFF **IoT Board**

Components



11 Pins (“ANALOG IN”, “POWER”)

Various pins can be used for analog inputs, outputs (3.3V and 5V), and ground.

12 ESP Wi-Fi Module

Microcontroller with integrated TCP/IP protocol that gives the AFF IoT Board access to the Internet.

13 ICSP Pins (In Circuit Serial Programming)

These pins are used to program the board via programming tool kit.

14 AVR Microcontroller

This is the microcontroller of the AFF board, similar to the one used in the traditional Arduino UNO board.

Download **Arduino IDE***

Access the internet

1

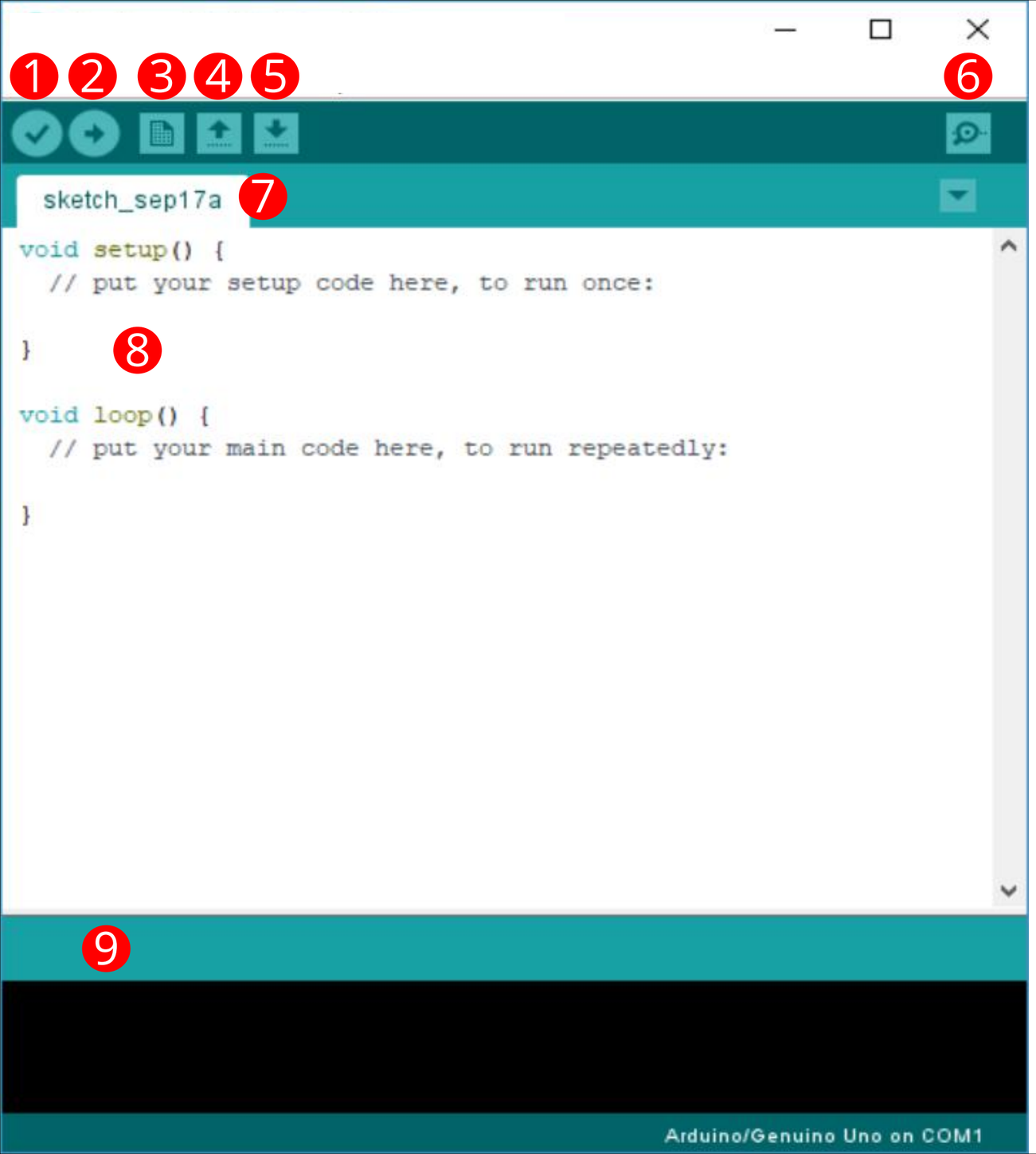
Download the newest version of the Arduino software from www.arduino.cc (it's free!). Type the following URL into the address bar of the browser: https://www.arduino.cc/en/Main/Software

2

Download

Choose the appropriate Operating System (Windows, MAC or Linux) installation package.

* IDE: Integrated Development Environment



About **Arduino IDE**

Open the Arduino IDE software on your computer and you will see a page similar to the one at the left.

About Arduino IDE

1

Verify

Compiles and verifies the code. It catches errors such as missing semi-colons or parenthesis.

2

Upload

Sends the code to the AFF IoT Board. When clicking it, the two yellow LEDs (RX and TX LEDs) on the board blink rapidly.

3

New

Opens up a new code window tab which contains only two functions: the `setup()` and `loop()` which are explained later.

4

Open

Opens up an existing sketch.

5

Save

Saves the current active sketch.

6

Serial Monitor

Opens a window that displays any serial information the AFF Board is transmitting. It is very useful for debugging.

7

Sketch Name

Displays the name of the sketch.

8

Code Area

Dedicated area for composing the sketch's code.

9

Message Area

Notifies the programmer about error(s) in the code and once the code is verified.

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

}
```

`setup()` function

This function is used to setup all configurations for pins and interfaces and initial values for used variables.

This function runs once upon powering up the board and when pressing “RESET” button.

`loop()` function

All lines of code written in this function will run repeatedly as long as the board is powered.

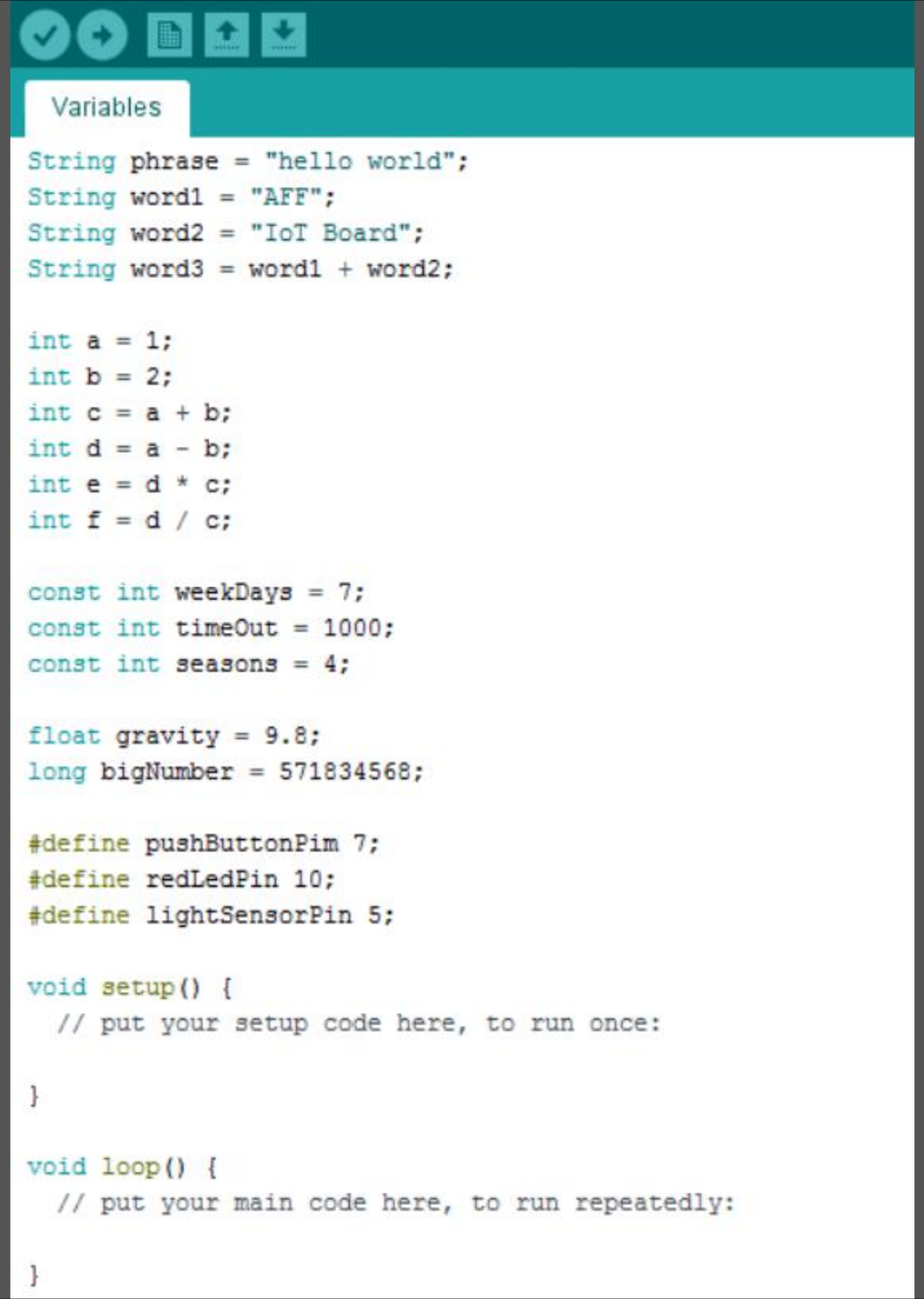
This function is used to execute the code infinitely.

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

}
```

Arduino essential functions

```
// this is for single line comments
// it’s good to put a description at
// the top and before anything ‘tricky’
```

```
/* this is for multi-line comments
   Like this…
   And this…. */
```



Learn programming and coding basics

In the next section all necessary programming concepts needed to program the AFF IoT board are explained with some illustrative examples.

Integer Variables

- A variable is like "bucket", it holds numbers or other values temporarily.

- Any variable can be assigned with a particular value using the equal operator “=“.

- All arithmetic operations can be used and the result of it can be stored at another variable.

```
int a = 1;

int b = 2;

int c = a + b;

int d = a - b;

int e = d * c;

int f = d / c;
```

String Variables

- This type of variables is intended to store characters which compose a word or phrase.

- Same as integer type, String variables can be combined together to compose another String using the plus operator “+”.

```
String phrase = "hello world";

String word1 = "AFF";

String word2 = "IoT Board";

String word3 = word1 + word2;
```

Constant Variables

- This type of variables can be assigned once and cannot be changed anywhere in the code.

- It can be declared by adding the “const” word before its type.

- Trying to assign another value to a constant variable cause a compiling error.

```
const int weekDays = 7;

const int timeOut = 1000;

const int seasons = 4;
```

Defined Variables

- It is similar to constant variables.

- It is usually used to assign pin numbers to the physical components.

- No need to declare its type, it can be assigned directly

- It is defined without equal operator “=“

```
#define pushButtonPim 7;

#define redLedPin 10;

#define lightSensorPin 5;
```

Decimal & long Variables

- Decimal variable is declared by using the type named “float”.

- Long variable is similar to the integer variable but it can store higher values.

- Long variable range is defined between:

-2147483647 & +2147483646

```
float gravity = 9.8;

long bigNumber = 571834568;
```

General concept

Functions and Libraries

Functions

```
void function(){
  // read sensors
  // test conditions
  // perform actions
}

int add(int a, int b){
  int c = a + b;
  return c;
}
```

- A function is a group of statements that together perform a task.

- A function can take some parameters (**a** and **b** in the example) perform some actions (addition in the example) and return the value (the sum of **a** and **b** which is **c**).

- It can also take no parameters and has no return value (declared as void function) such as print function.

Libraries

```
#include <Servo.h>

#include <EEPROM.h>

#include <NeoSWSerial.h>
```

- Libraries are files written in C or C++ (.c, .cpp) which provide your sketches with extra functionality (e.g. the ability to control a servo motor, or read an encoder, etc.). These statements make the public **functions** and constants **defined** by the **library** available to your sketch.

- To use an existing library in a sketch simply go to the Sketch menu, choose "Import Library", and pick from the libraries available. This will insert an **#include** statement at the top of the sketch for each header (.h) file in the library's folder.

Control Structure

The most important statement!

```
if (condition)
{
  //statement(s)
}

if (x > 120) digitalWrite(LEDpin, HIGH);

if (x > 120)
  digitalWrite(LEDpin, HIGH);

if (x > 120) {
  digitalWrite(LEDpin, HIGH);
}

if (x > 120) {
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}
```

The if statement checks for a condition and executes the proceeding statement or set of statements if the condition is 'true'.

The brackets may be omitted after an if statement. If this is done, the next line (defined by the semicolon) becomes the only conditional statement.

The statements being evaluated inside the parentheses require the use of one or more operators shown below.

Comparison Operators:

`x == y (x is equal to y)`

`x != y (x is not equal to y)`

`x <  y (x is less than y)`

`x >  y (x is greater than y)`

`x <= y (x is less than or equal to y)`

`x >= y (x is greater than or equal to y)`

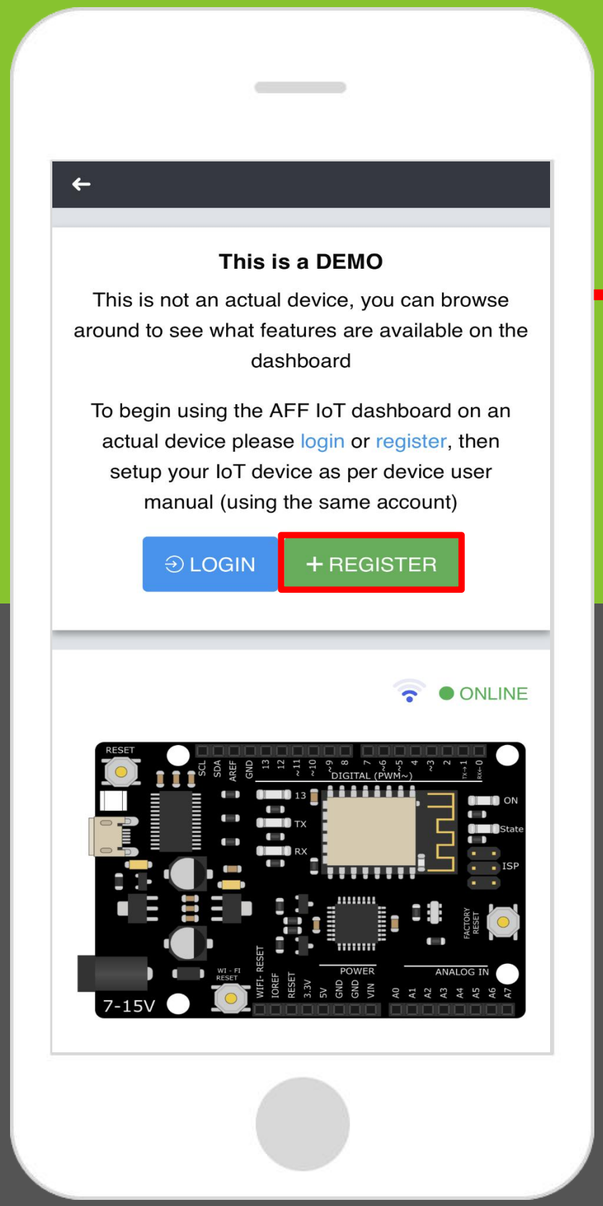
Beware of accidentally using the single equal sign (e.g. **`if(x = 10)`**). The single equal sign is the assignment operator, and sets x to 10 (puts the value 10 into the variable x). Instead use the double equal sign (e.g. **`if(x == 10)`**), which is the comparison operator, and tests whether x is equal to 10 or not. The latter statement is only true if x equals 10, but the former statement will always be true.

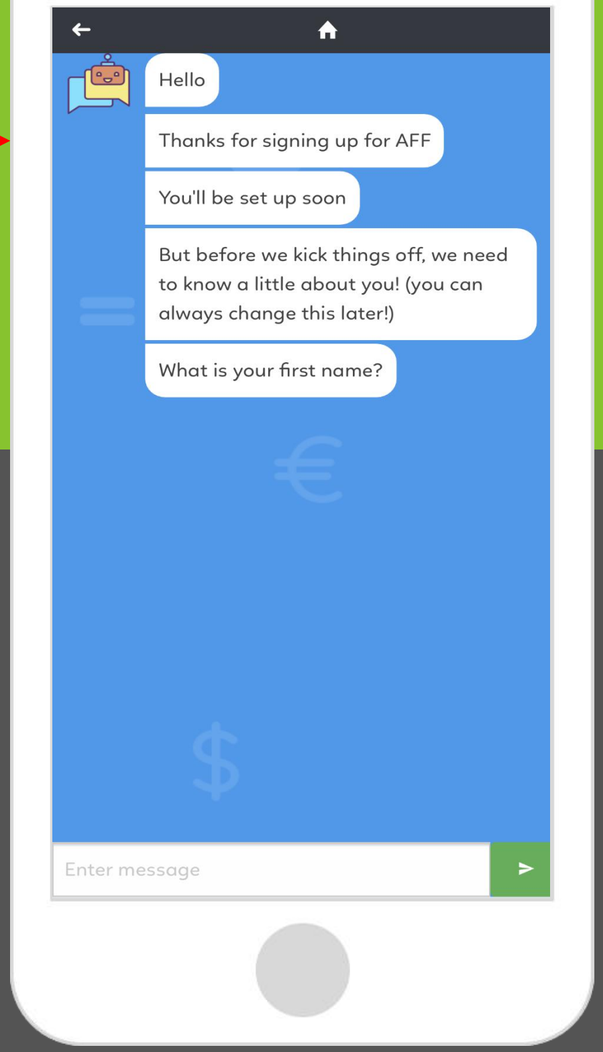
Register at AFF IoT app

(available on Android and iOS)

After downloading the app (AFF IoT), create new account. Creating an account is a simple 10 steps procedure and takes less than 2 minutes. If you don't have a smartphone, you can use the browser on the PC such as Chrome, Firefox, etc.

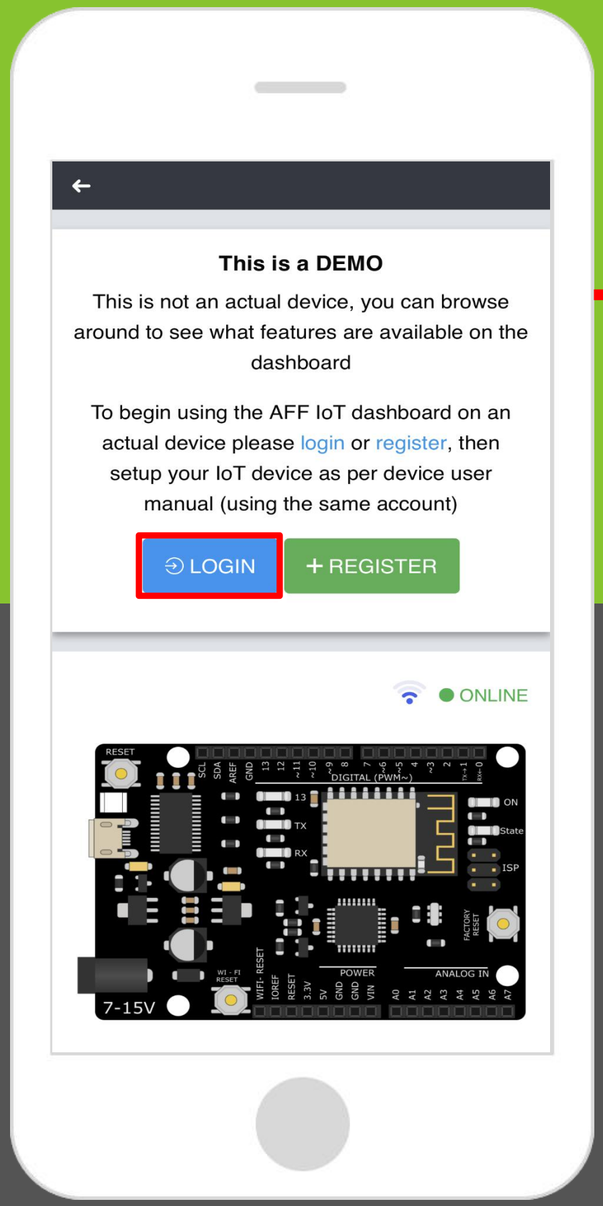
* If you face any problem while creating new account, or while logging in, please drop us an email to ***support@affcity.com***

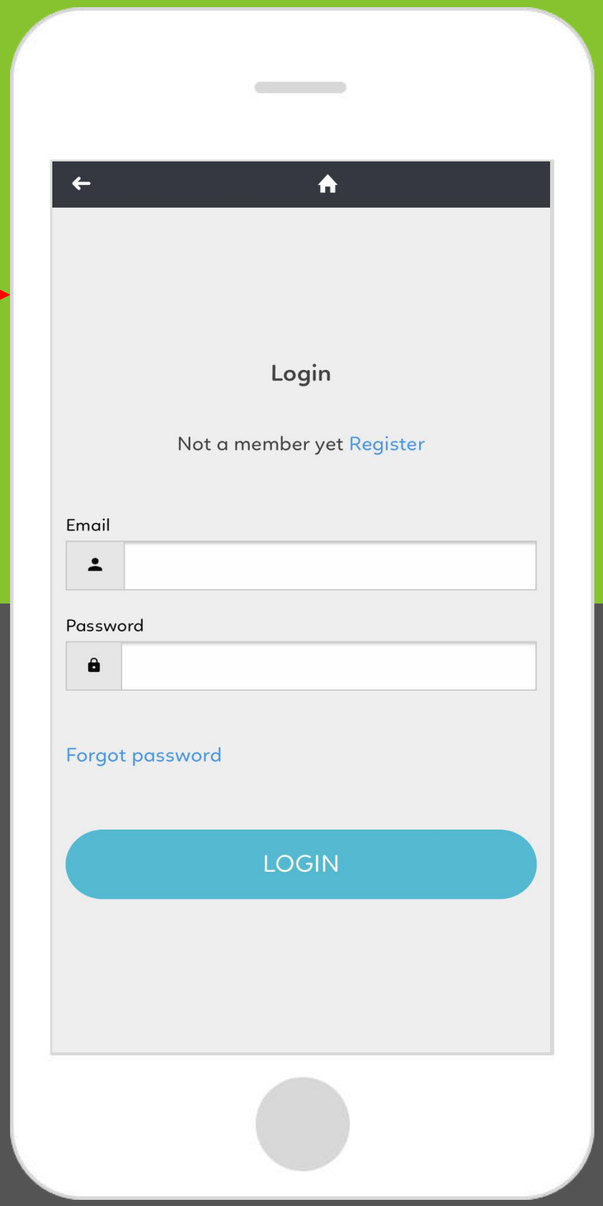




AFF IoT App

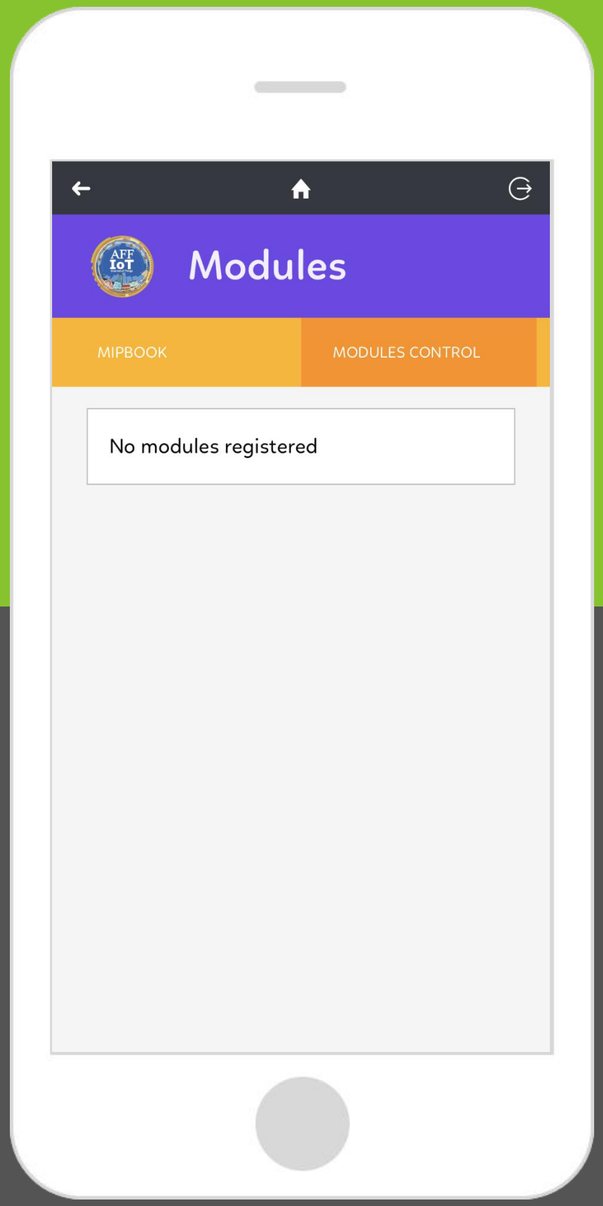
1. Create an account by clicking on the green button “**REGISTER**”.
2. Complete the required information.
3. Then Login using your email and password.





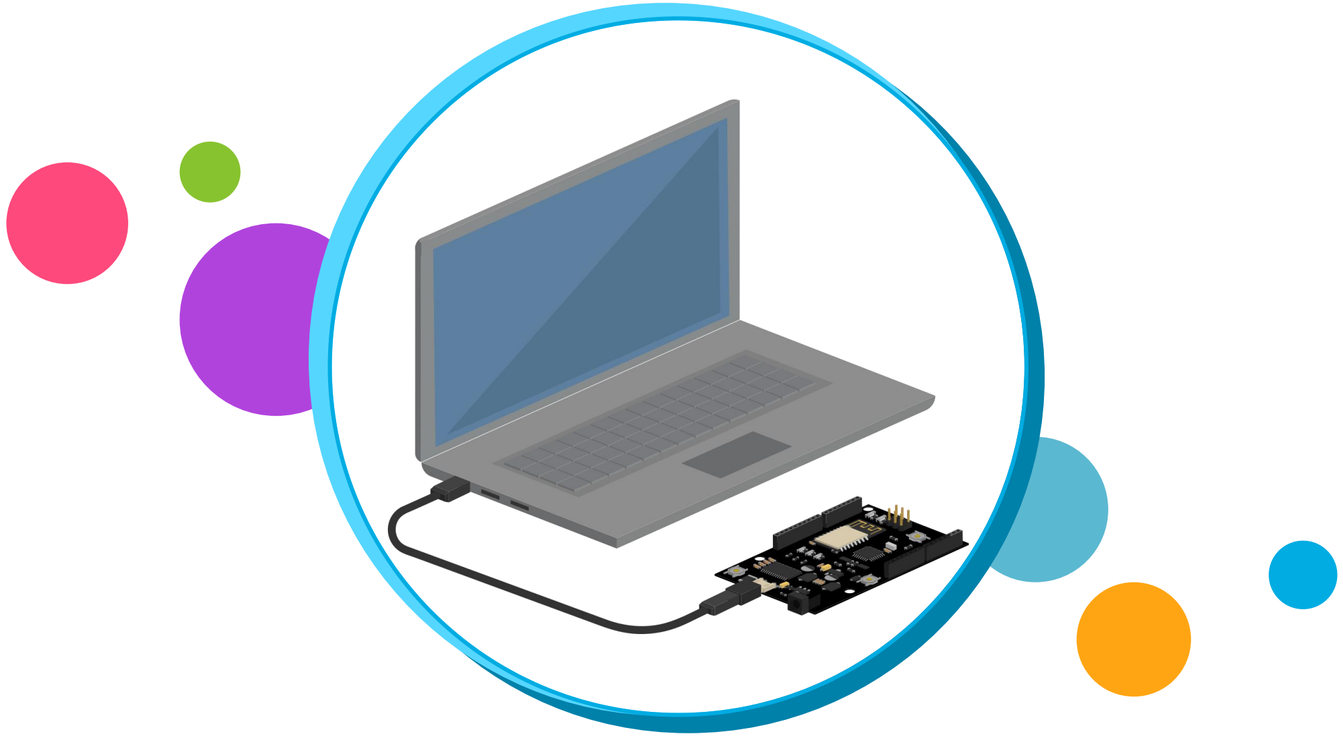
AFF IoT App

1. If you have an account, click on blue button **“LOGIN”**.
2. Then Login using your Email and Password.



AFF IoT App

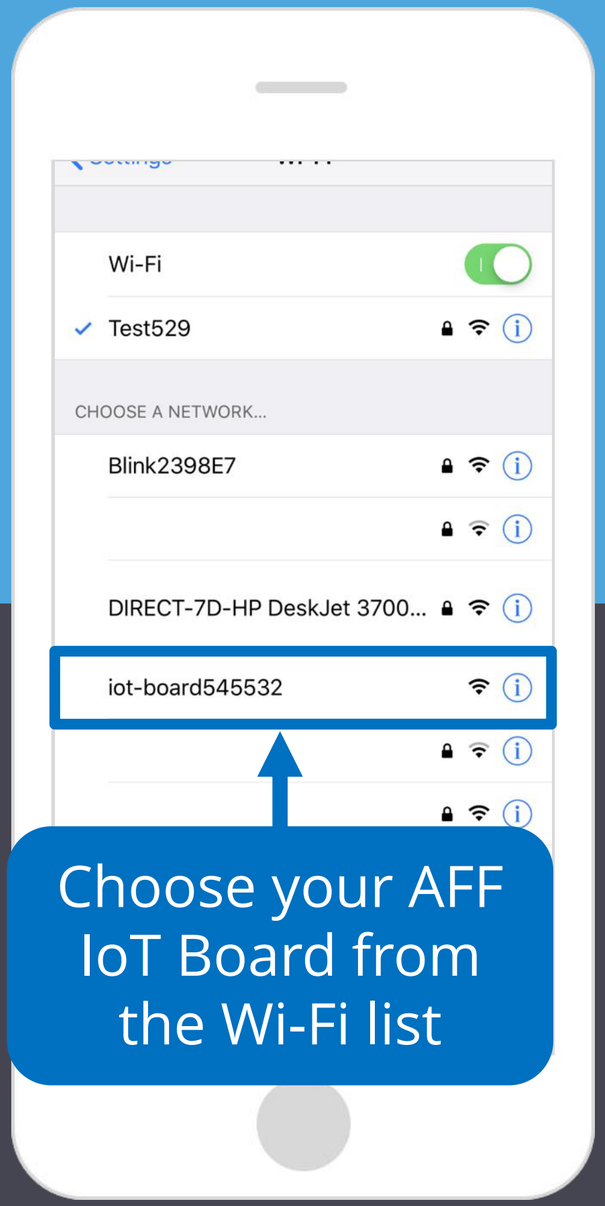
Next, register the AFF IoT Board (explained in next section). Then everything will be ready to build projects!



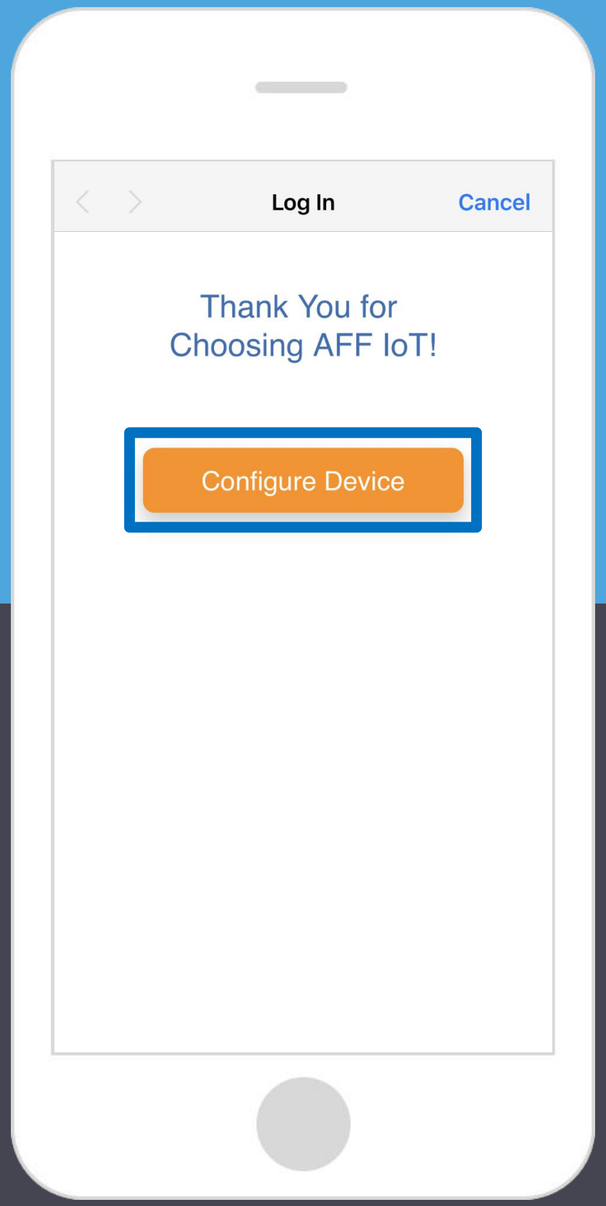
Connect AFF IoT Board to the Computer

Use micro USB cable to connect the AFF IoT Board to any of the computer’s USB ports.

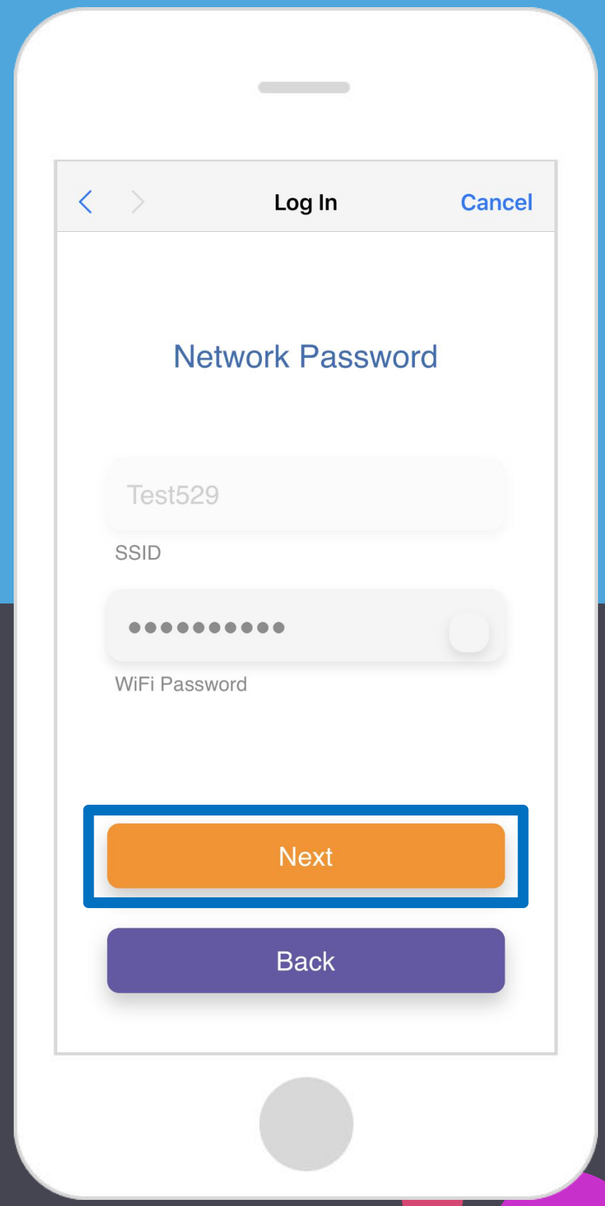
Then open the Wi-Fi page on your smartphone or PC.

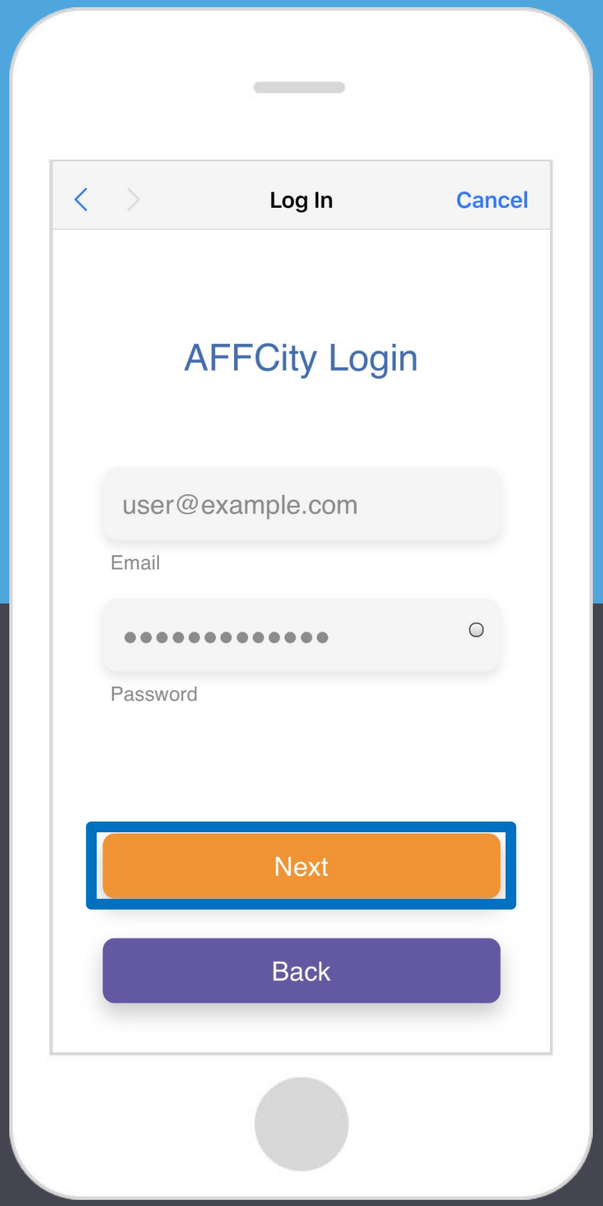


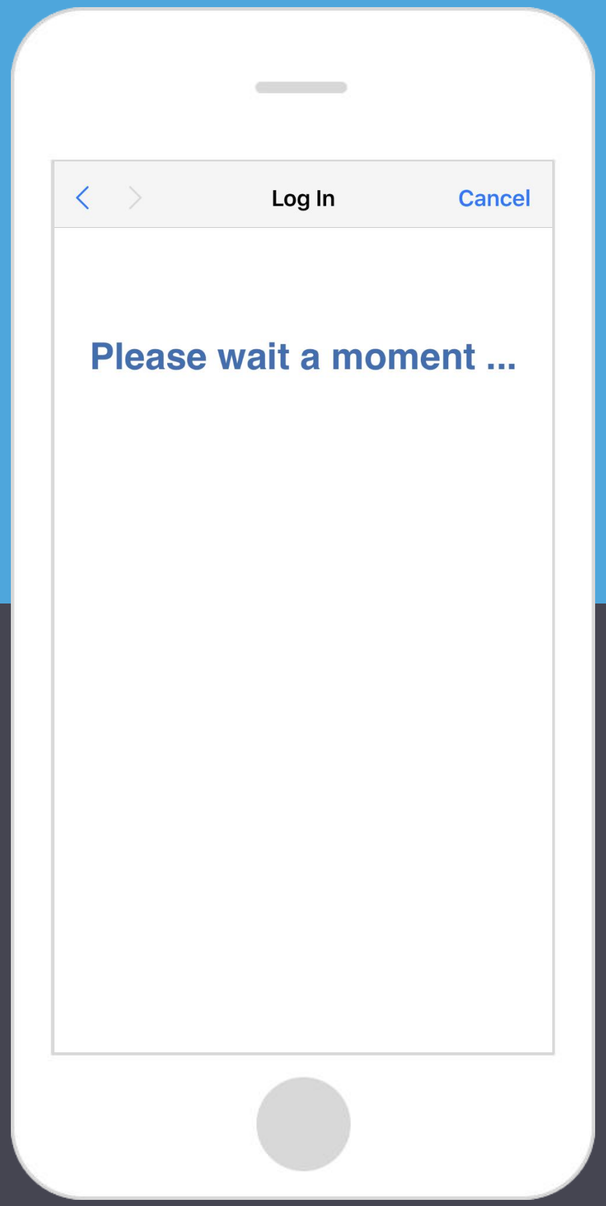
Choose your AFF IoT Board from the Wi-Fi list

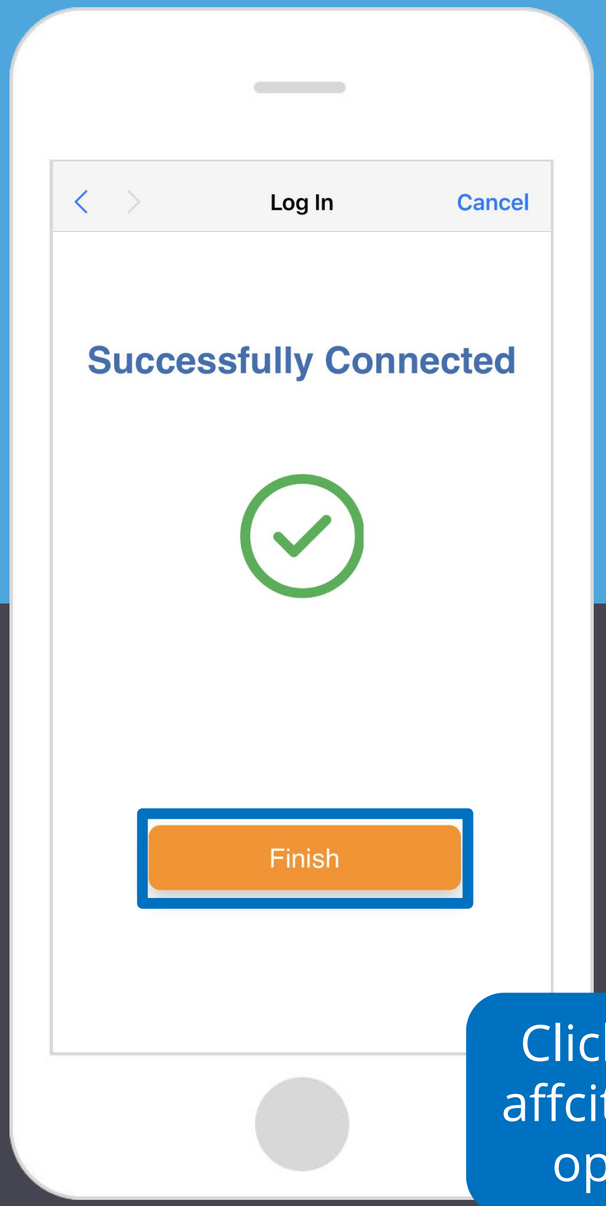


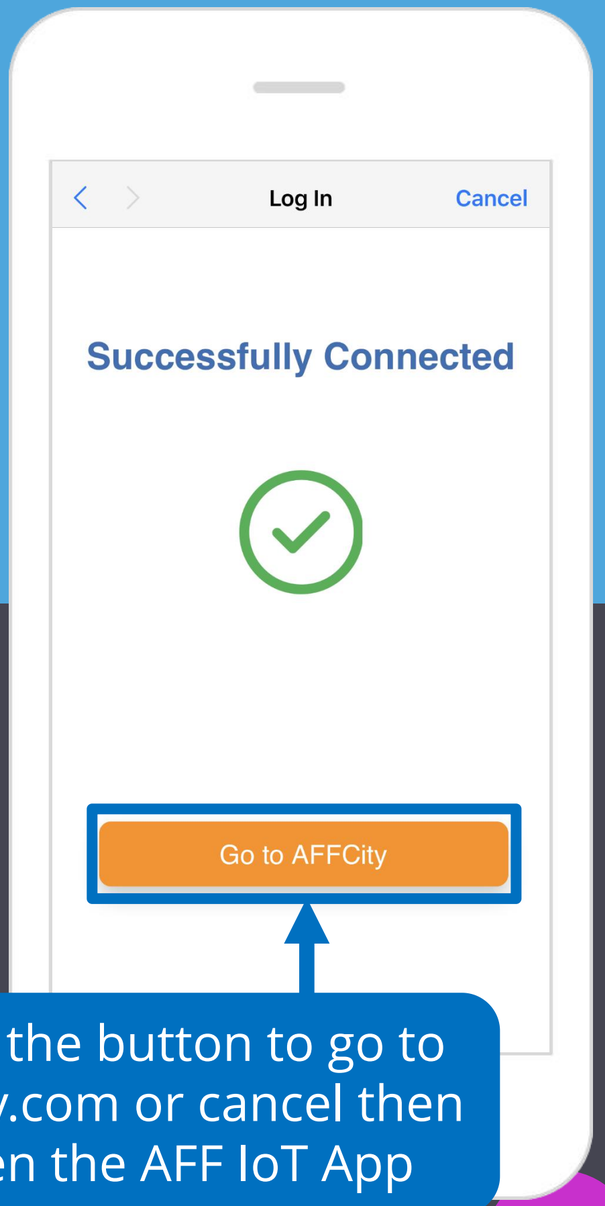
If your network doesn't listed, click "Scan Networks"



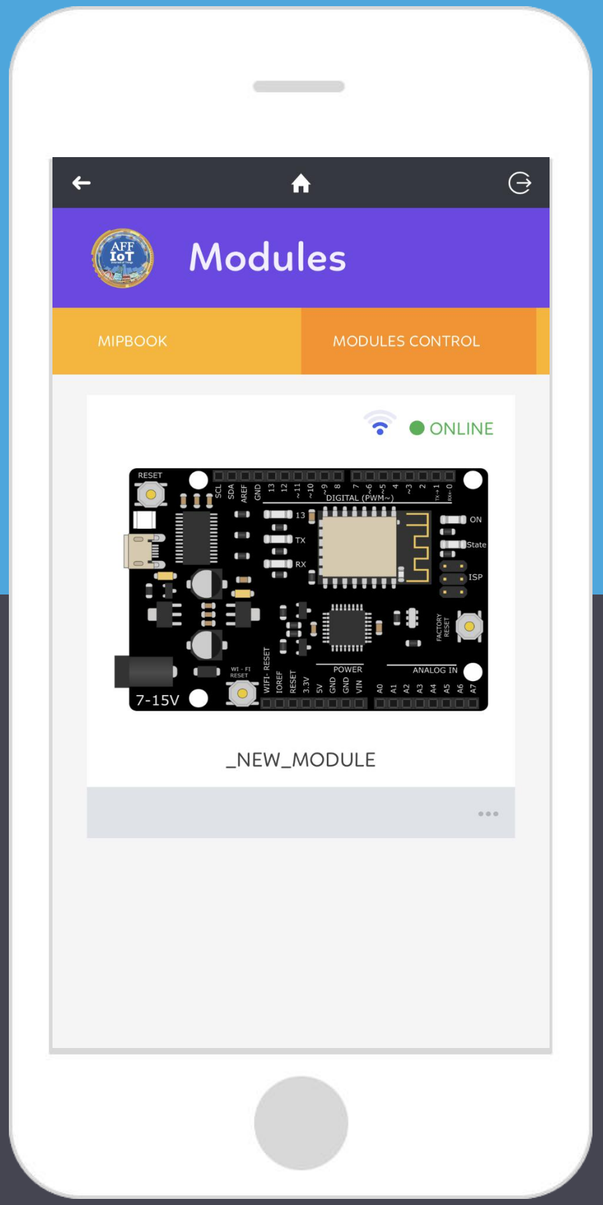








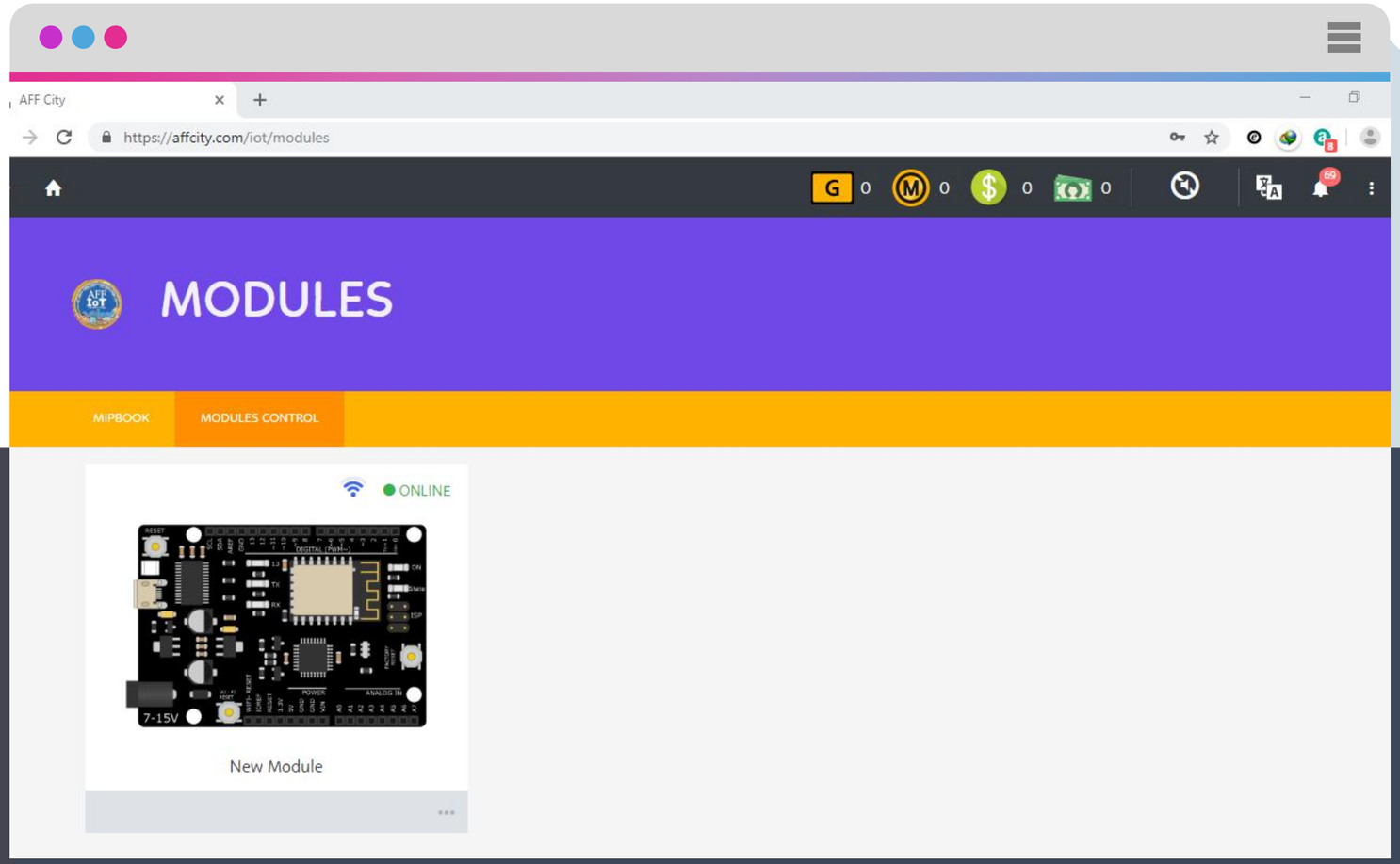
Click the button to go to affcity.com or cancel then open the AFF IoT App



Finally...

Open the AFF IoT App and you should see the AFF IoT board connected the Internet.

It is named by default NEW MODULE, and can be edited.



For PC users

The procedure is the same except instead of using the AFF IoT App, you should go to https://affcity.com/iot/modules.

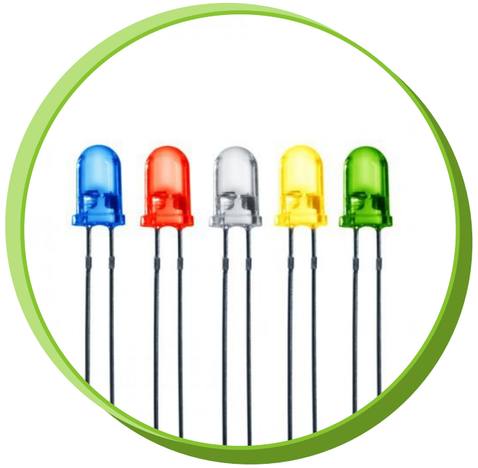
Here you can continue same as described using the App.

Build 12 projects

AFF IoT Kit **includes:**

Jumper wires x20

Various Colors



LEDs x27

Red x5, Green x5, Blue x5,
Yellow x5, White x5
and RGB x2

Pushbutton x9

Squared shape pushbutton

Potentiometer x4

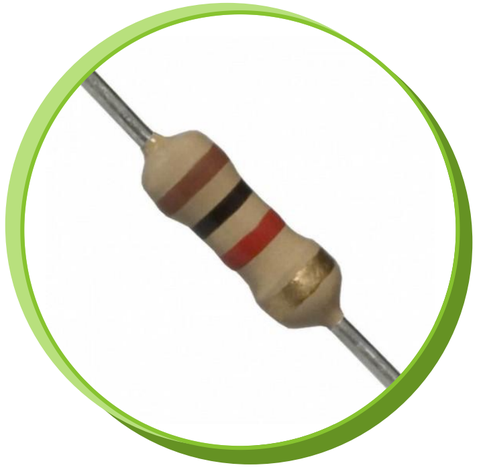
Variable resistor between

0Ω and 5kΩ

AFF IoT Kit **includes:**

10KΩ Resistor x6

0.5W 5%

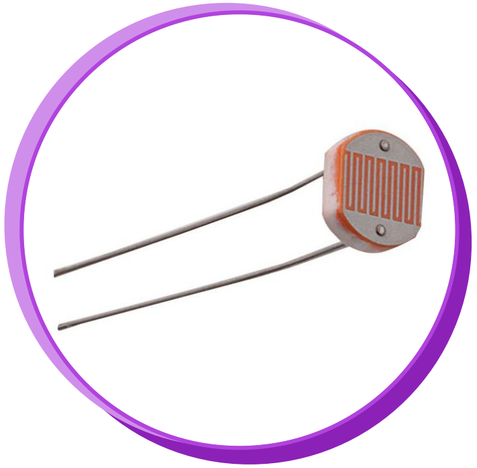


1KΩ Resistor x4

0.5W 5%

330Ω Resistors x25

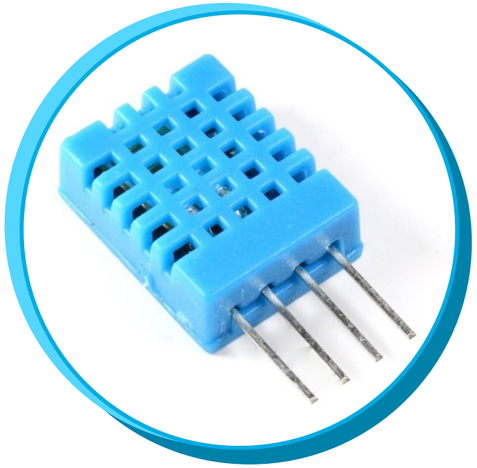
0.25W 5%



Photocell x2

Light Dependent Resistor (LDR)

AFF IoT Kit **includes:**



DHT11 x1

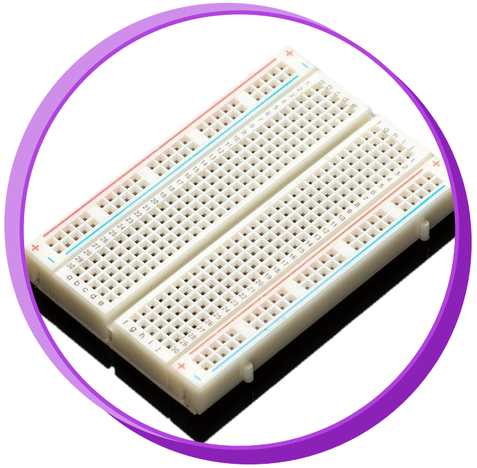
Humidity and Temperature sensor

Thermistor x3

1KΩ at 25°C

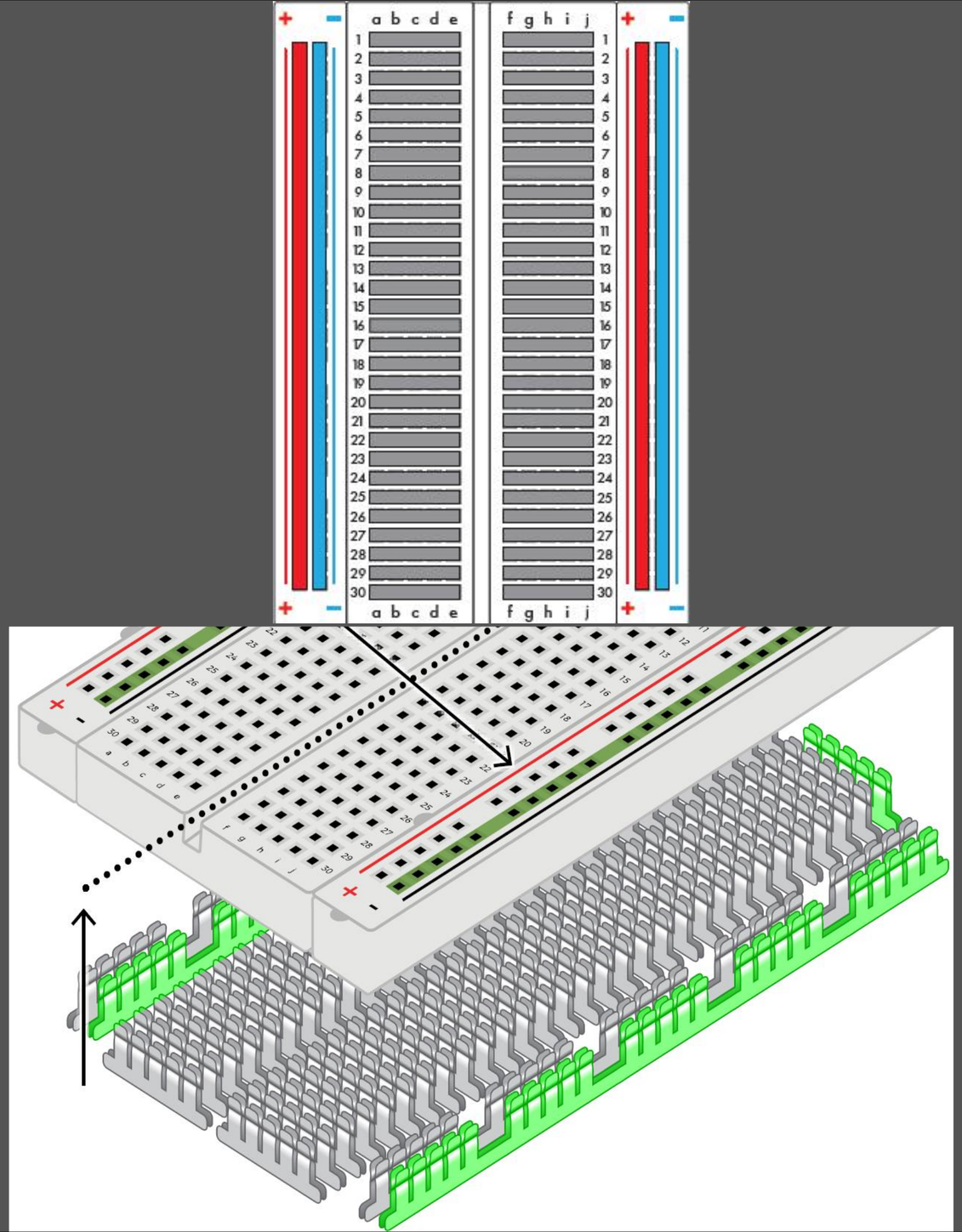
Piezo Buzzer x2

Polarized Piezo buzzer



Breadboard x1

400 pins solderless breadboard



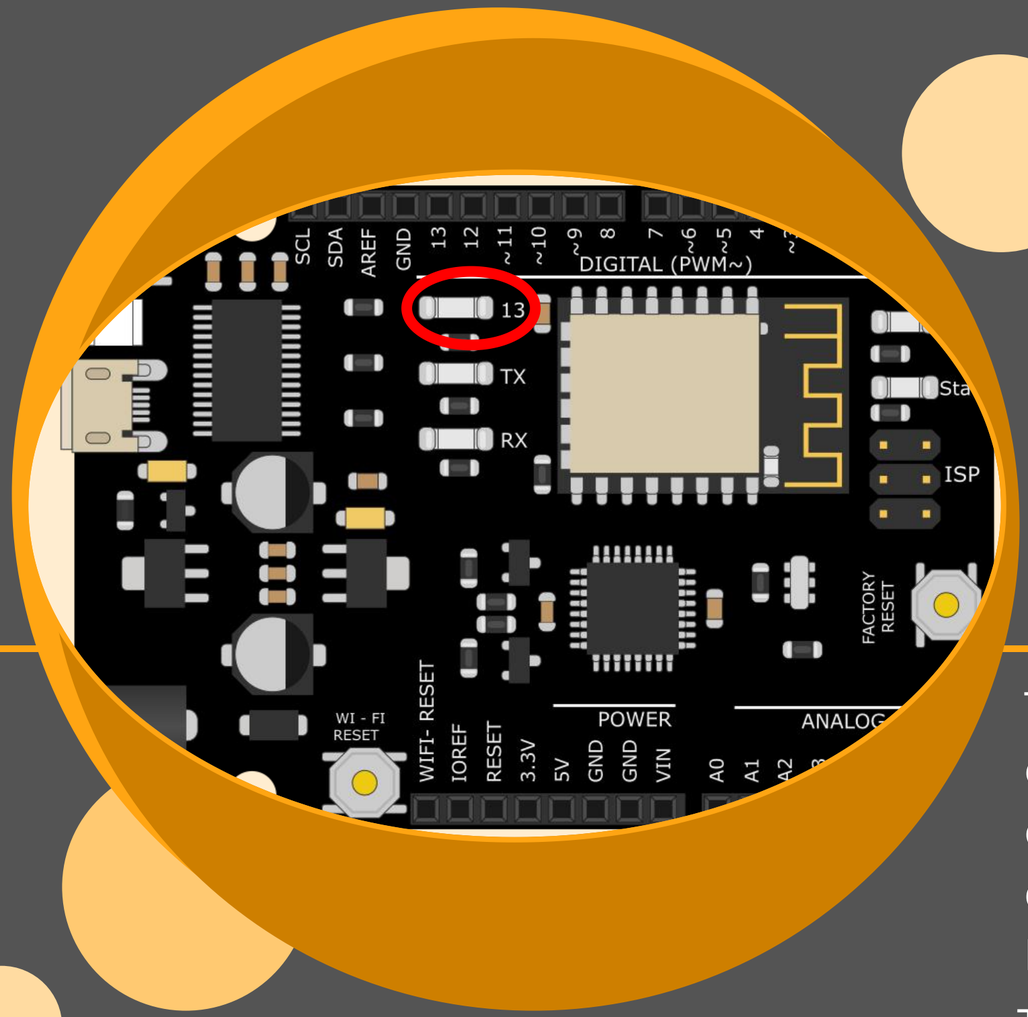
Breadboard

A breadboard is a construction base for prototyping of electronics.

+ VCC:
Each + sign runs power anywhere in the vertical column.

— GND:
Each - sign runs ground anywhere in the vertical column.

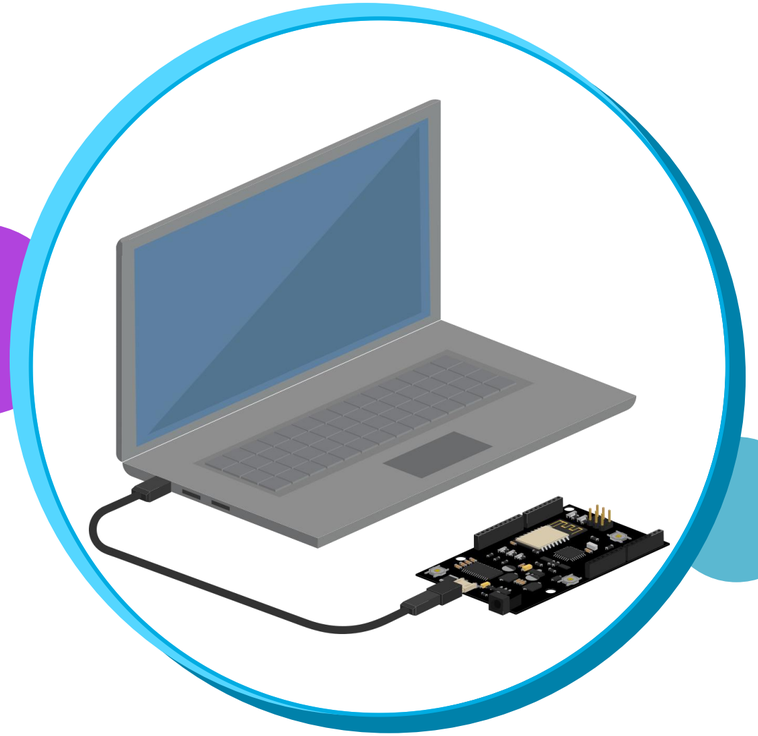
— Horizontal Rows: (a-e & f-j)
Each of these rows numbered from 1 to 30 are comprised of five horizontal sockets. Components placed in the same row will be connected together.



1 Blinking Built-in LED

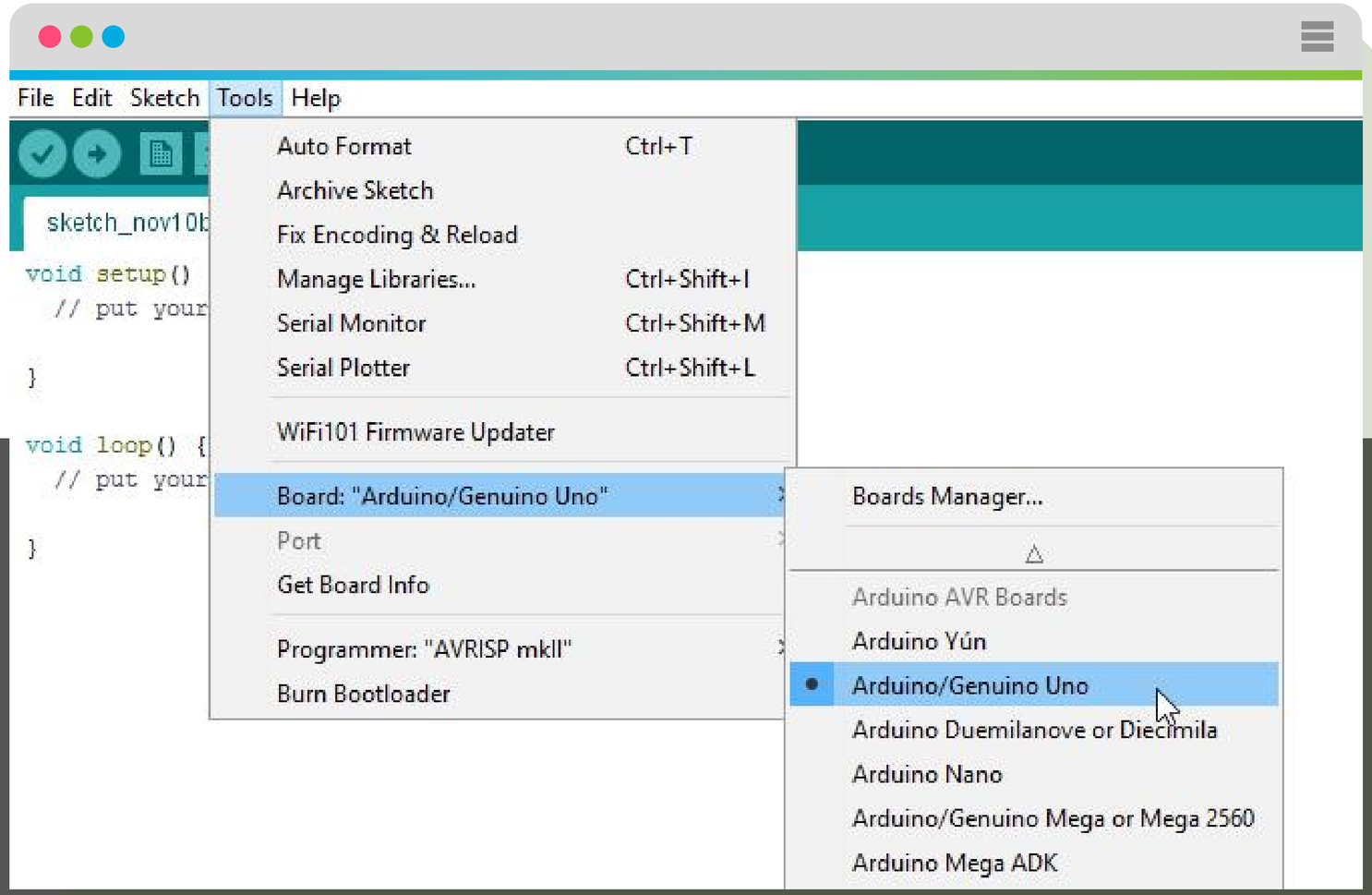
The first program every programmer learns consists in writing enough code to make their code show the sentence "Hello World!" on a screen. The blinking LED is the "Hello World!" of physical computing. In this tutorial the needed component is only the AFF IoT board and the micro USB cable.

This tutorial consist of turning the built-in LED (LED13) ON for one second, then OFF for one second, repeatedly.



Connect AFF IoT Board to the Computer

Use micro USB cable to connect the AFF IoT Board to any of the computer’s USB ports.

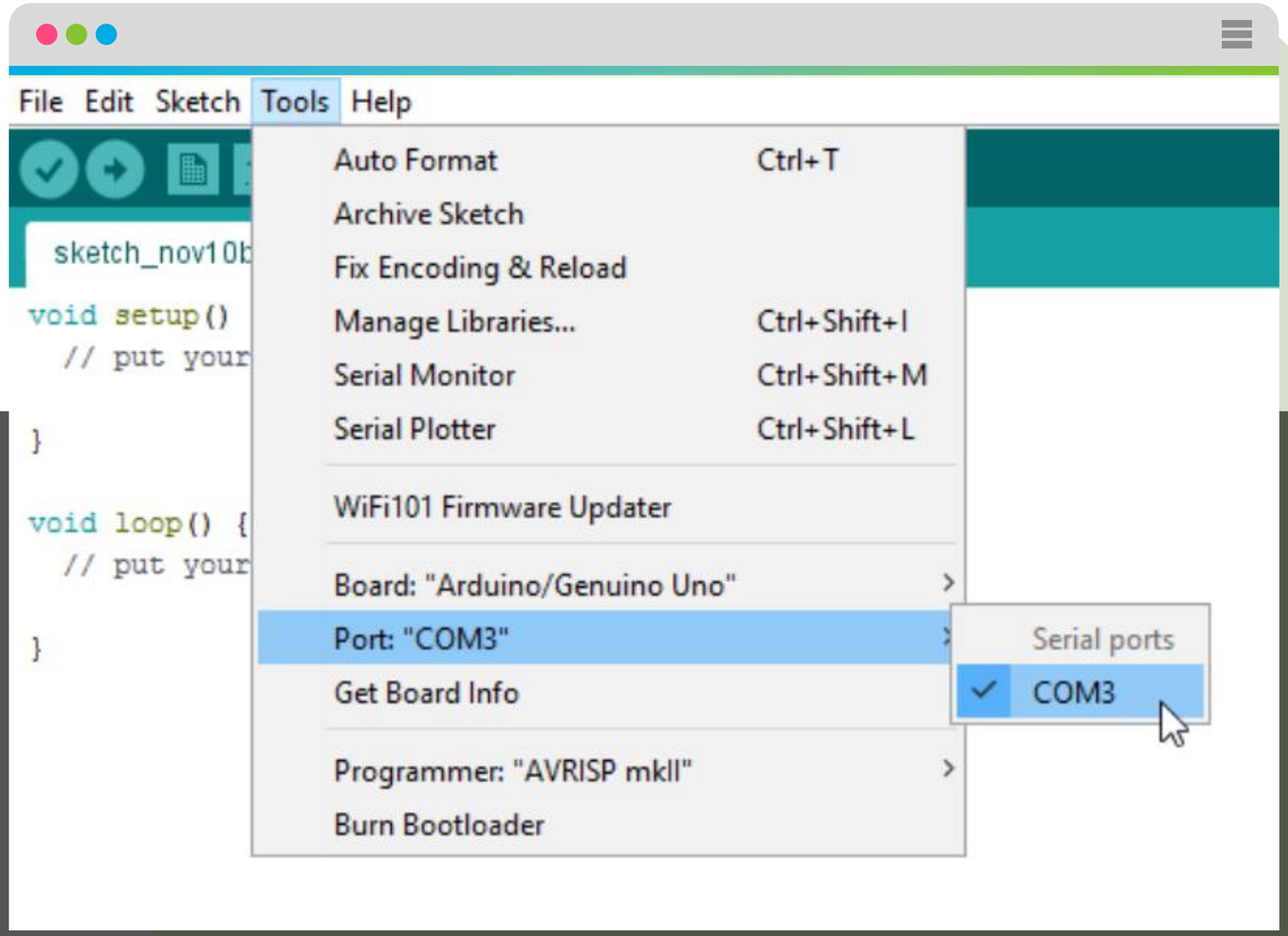


Select board: Arduino/Genuino Uno

Note:

AFF IoT Board is not listed in the Arduino Software.

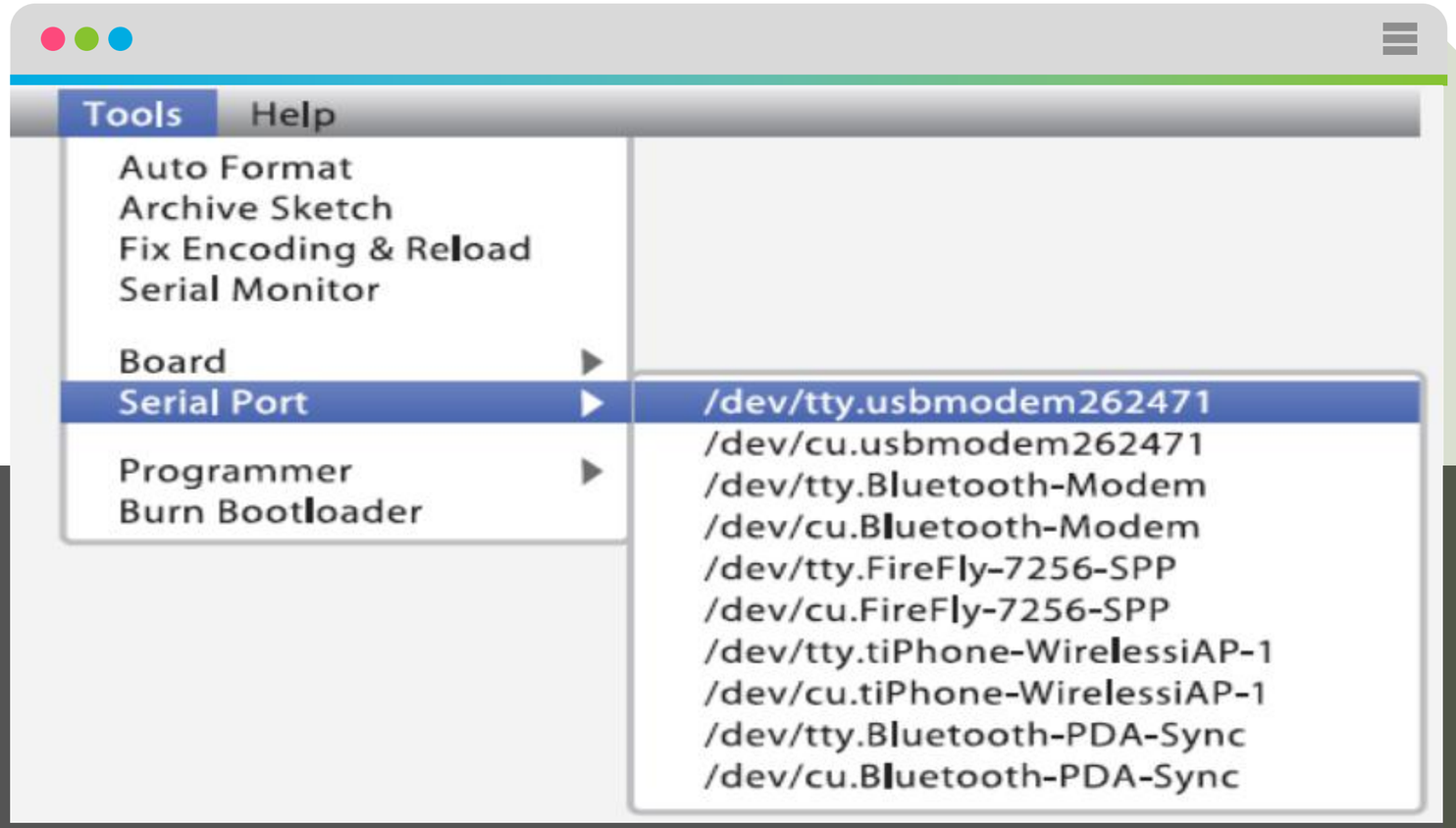
Select “Arduino/Genuino Uno” instead.



Select the serial device: (Windows)

Select the serial device of the Board from the Tools → Serial Port menu. To find out which port, you can disconnect the Board and re-open the menu; the entry that disappears should be the Board. Reconnect the board and select that serial port. Please note that the number of serial port may differs from the example used.

Note: If you cannot see the board please refer to the following link: https://learn.sparkfun.com/tutorials/how-to-install-ftdi-drivers/windows---in-depth

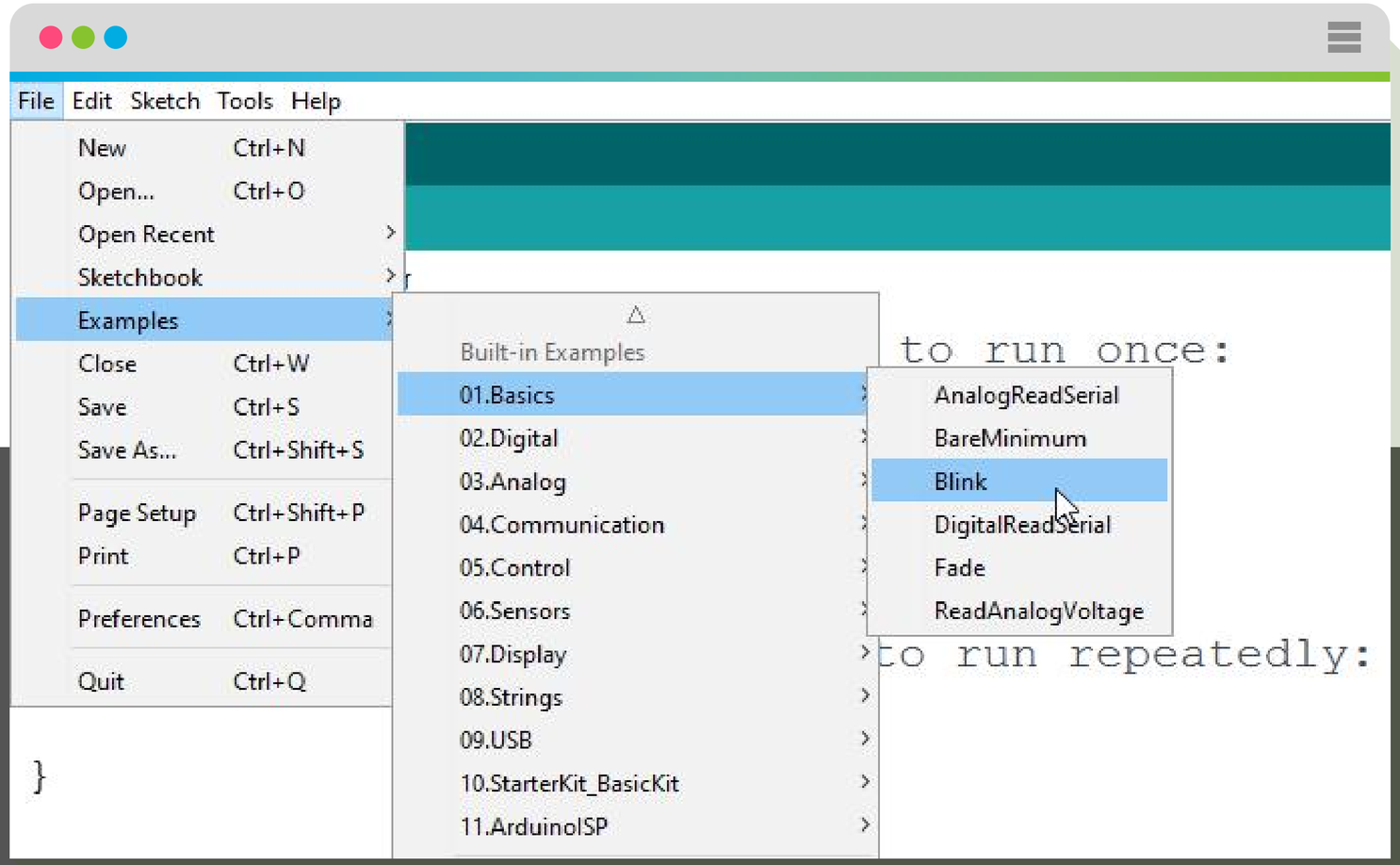


Select the serial device: (for Mac)

Select the serial device of the Board from the Tools → Serial Port menu. This should be something like

/dev/tty.usbmodem or

/dev/tty.usbserial



Open the Blink example

After connecting the AFF IoT board to the computer, open the Blink example listed in the written examples in the Arduino IDE.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Blink example

This example is the alphabet of the microcontroller world!

Code to note..

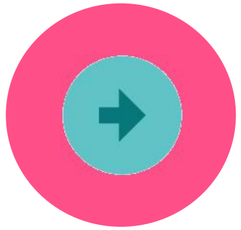
`pinMode(LED_BUILTIN, OUTPUT)`: Before you can use one of the board's pins, the pin has to be specified as an `INPUT` (to read a sensor data) or `OUTPUT` (control an actuator). We use a built-in "function" called `pinMode()` to set the pin13 (which LED13 is connected to) as `OUTPUT`.

`digitalWrite(LED_BUILTIN, HIGH)`: A digital output pin is either set to be `HIGH` (output 5 volts), or `LOW` (output 0 volts). We should choose which pin to be set in the first parameter (LED_BUILTIN in the example), and the value desired (HIGH in the example).

`delay(1000)`: It pauses the program for the amount of time (in milliseconds) specified as parameter. (There are **`1000`** milliseconds in a second).

Finally..

This compiles your code. The IDE changes it from text into instructions the microcontroller can understand.



This sends the instructions via the USB cable to the microcontroller chip on the Board. The Board will then begin running the code automatically.

What you **should see**

You should see your LED turn on for one second then turn off for one second and so on. If it isn't, make sure you have uploaded the code correctly to the board and there is no compiling error or see the troubleshooting tips below.

Troubleshooting

LED13 Not Lighting Up?
Make sure the board is plugged into the PC correctly

Program Not uploading?
It happens sometimes, the most likely cause is choosing wrong serial port, you can change this in Tools → Port

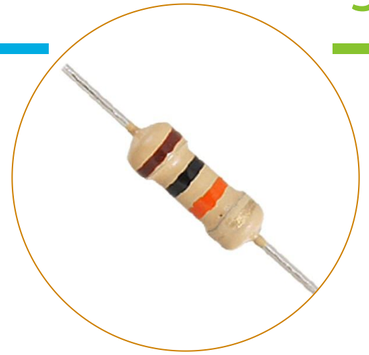
Still Not uploading?
The common solution for almost all electronic devices is to turn it OFF then turn it ON

2 Controlling LED with Pushbuttons

LED x1

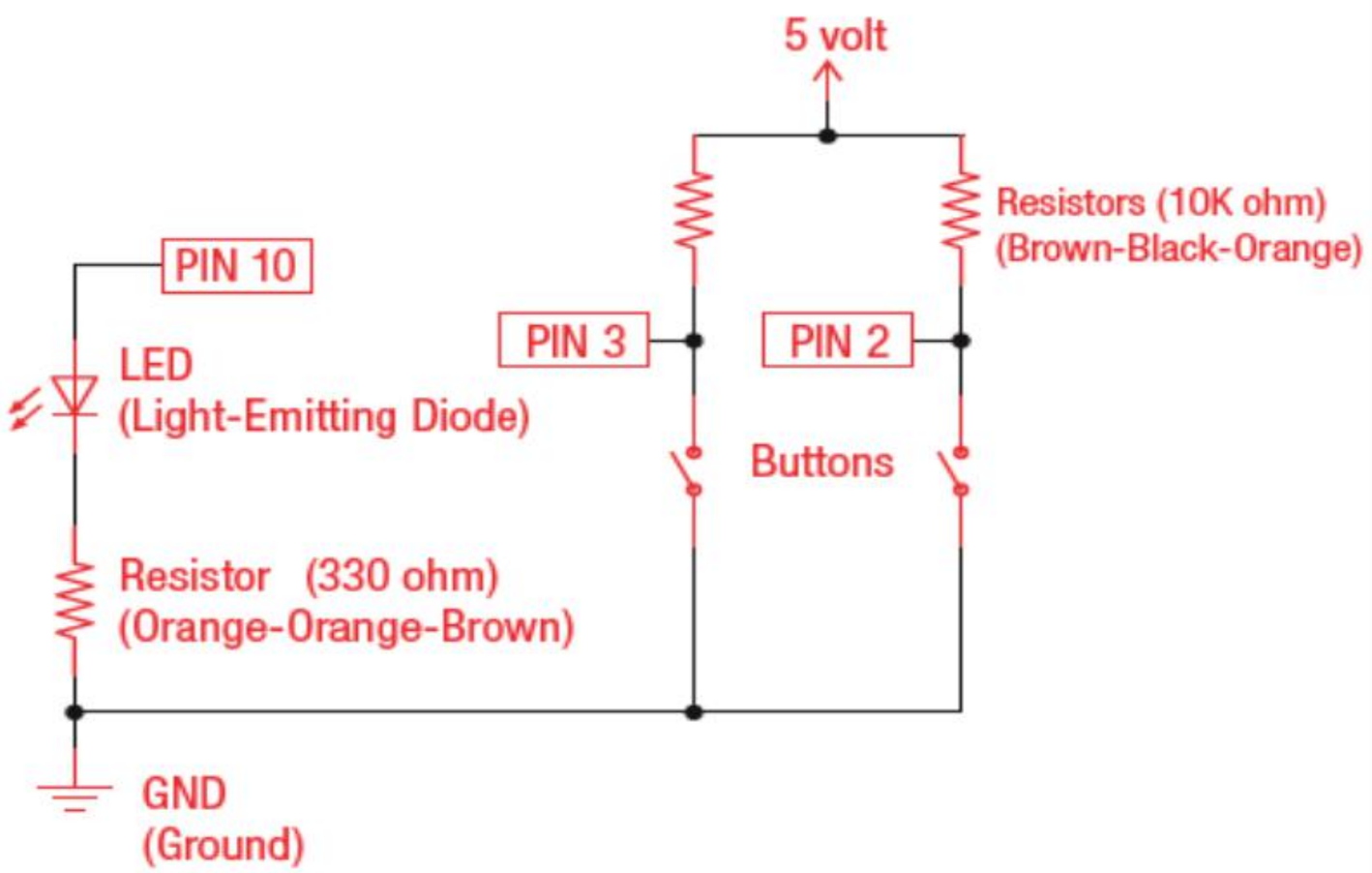
330Ω Resistor x1

Pushbutton x2



10KΩ Resistor x2

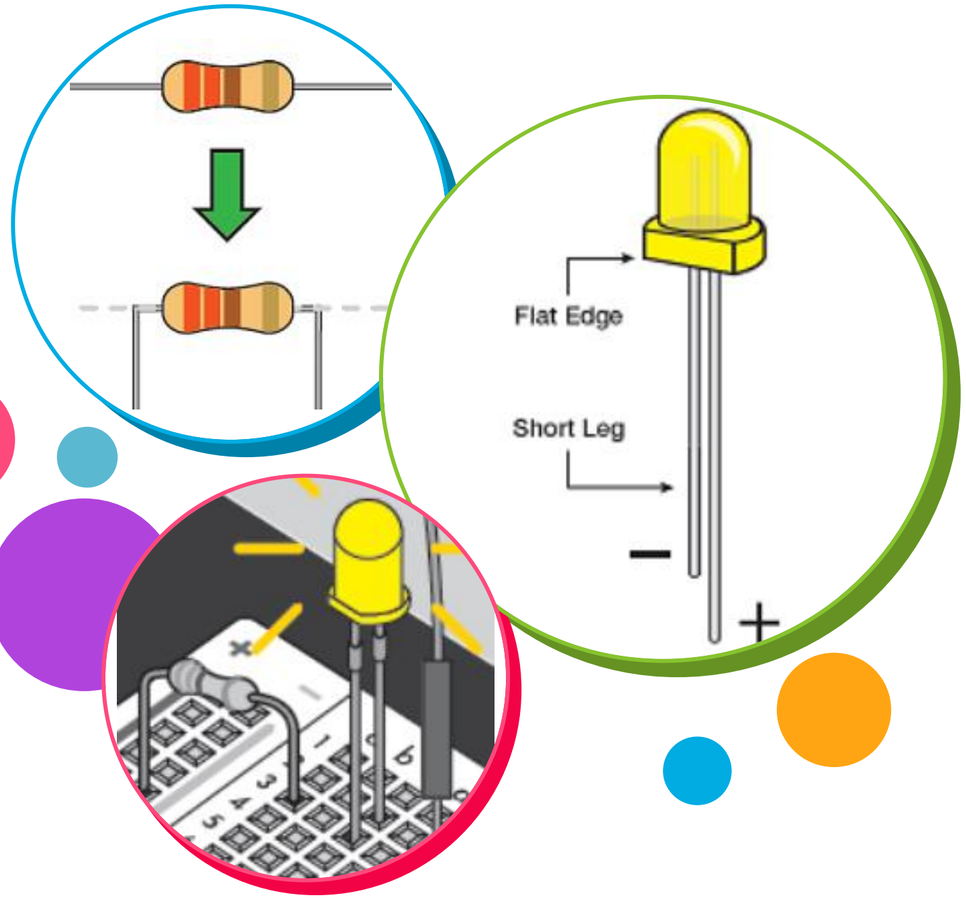
Jumper wires x7



Pushbutton

One of the most common and simple inputs is a push button. The way a push button works with board is that when the button is pushed, the voltage goes LOW. The board reads this and reacts accordingly. In this circuit, a pull-up resistor will be used to keep the voltage HIGH at 5V when the button is not pressed.

Be careful!



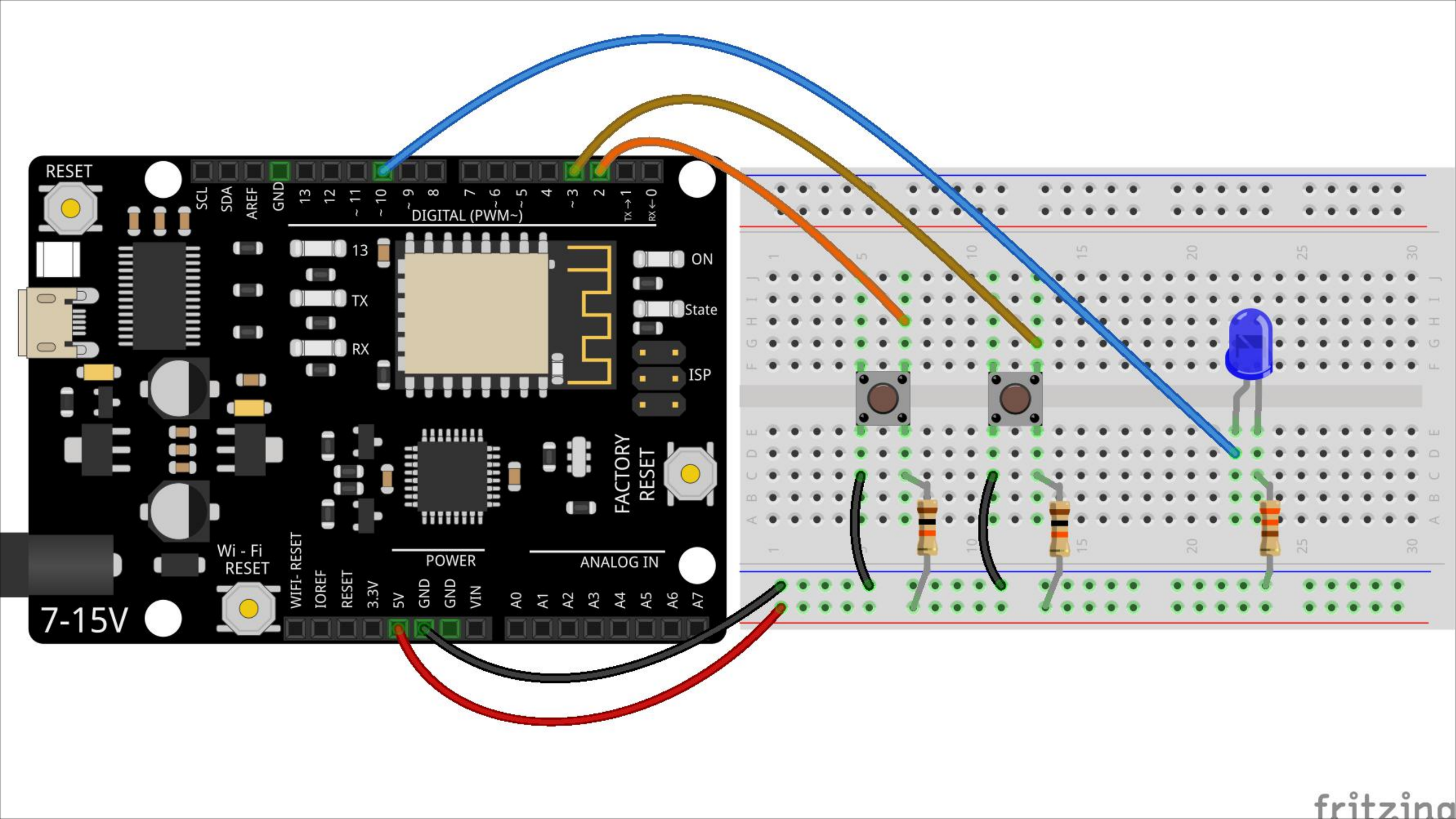
LED:

The short leg, marked with flat side, goes into the negative position (-).

330Ω Resistor:

The color banding should read orange-orange-brown-gold.

The component legs can go in either hole..


RESET
SCL
SDA
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TX → 1
RX ← 0
DIGITAL (PWM~)
13
TX
RX
ON
State
ISP
FACTORY RESET
Wi - Fi RESET
WIFI- RESET
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
A6
A7
7-15V
fritzing

==	EQUIVALENCE	A == B is true if A and B are the SAME.
!=	DIFFERENCE	A != B is true if A and B are NOT THE SAME.
&&	AND	A && B is true if BOTH A and B are TRUE
||	OR	A || B is true if A or B or BOTH are TRUE
!	NOT	!A is TRUE if A is FALSE !A is FALSE if A is TRUE

How to use LOGIC

You could turn on a heater if it gets too cold, a fan if it gets too hot, water the plants if they get too dry, etc. In order to make such decisions, there are set of logic operations. It can be combined to build complex `if()` statements.

For example:

```
if ((mode == heat) && ((temperature < threshold) || (time == "night")))
        digitalWrite(HEATER, HIGH);
```

AFF IoT Board folder > Circuits > Using_Pushbuttons

Using_Pushbuttons

```
#define Button1Pin  2
#define Button2Pin  3
#define BlueLedPin  10

void setup() {
  // Set up the pushbutton pins to be an input:
  pinMode(Button1Pin, INPUT); // configure the pin connected to the pushbutton 1 to be an input
  pinMode(Button2Pin, INPUT); // configure the pin connected to the pushbutton 2 to be an input
  pinMode(BlueLedPin, OUTPUT); // configure the pin connected to the Blue Led to be an output
}

void loop() {

  int Button1State, Button2State;  // variables to hold the pushbutton states
  Button1State = digitalRead(Button1Pin); // read the state of Button1 (LOW if pressed and HIGH if released)
  Button2State = digitalRead(Button2Pin); // read the state of Button2 (LOW if pressed and HIGH if released)

  if (((Button1State == LOW) || (Button2State == LOW))  // if we're pushing button 1 OR button 2
      && !                                               // AND we're NOT
      ((Button1State == LOW) && (Button2State == LOW))) // pushing button 1 AND button 2 simultaneously
                                                         // then...
  {
    digitalWrite(BlueLedPin, HIGH);  // turn the Blue LED on
  }
  else
  {
    digitalWrite(BlueLedPin, LOW);  // turn the Blue LED off
  }
}
```

Open your sketch: Using_Pushbuttons

Code to note..

`pinMode(Button1Pin, INPUT)`: The digital pins can be used as inputs as well as outputs. It's specified as `INPUT` in order to read the state of the button.

`Button1State = digitalRead(Button1Pin)`: To read a digital input, we use the `digitalRead()` function. It will return `HIGH` (or 1) if there's 5V present at the pin, or `LOW` (or 0) if there's 0V present at the pin.

`if (Button1State == LOW)`: Because we've connected the button to `GND`, it will read `LOW` when it's being pressed. The "equivalence" operator ("`==`") is used to see if the button is pressed.

`else`: the `else` statement means that if the condition in the `if` statement is `false`, the code in `else` brackets are executed.

What you **should see**

You should see the LED turn on if you press either button, and off if you press both buttons (see the code). If it isn't working, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting tips below.

Troubleshooting

LED Not Turning On?
The pushbutton is square, and because of this it is easy to put it in the wrong way. Give it a 90 degree rotation and see if it starts working.

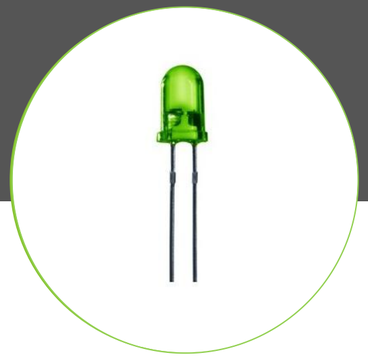
Still not lighting up?
LEDs will only work in one direction. Try taking it out and twisting it 180 degrees (don't worry, installing it the opposite way doesn't damage the LED).

Real World Application

Buttons in most video game controllers.

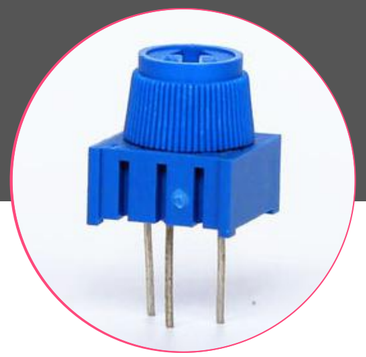
3 Dimming LED using Potentiometer

Jumper wires x7



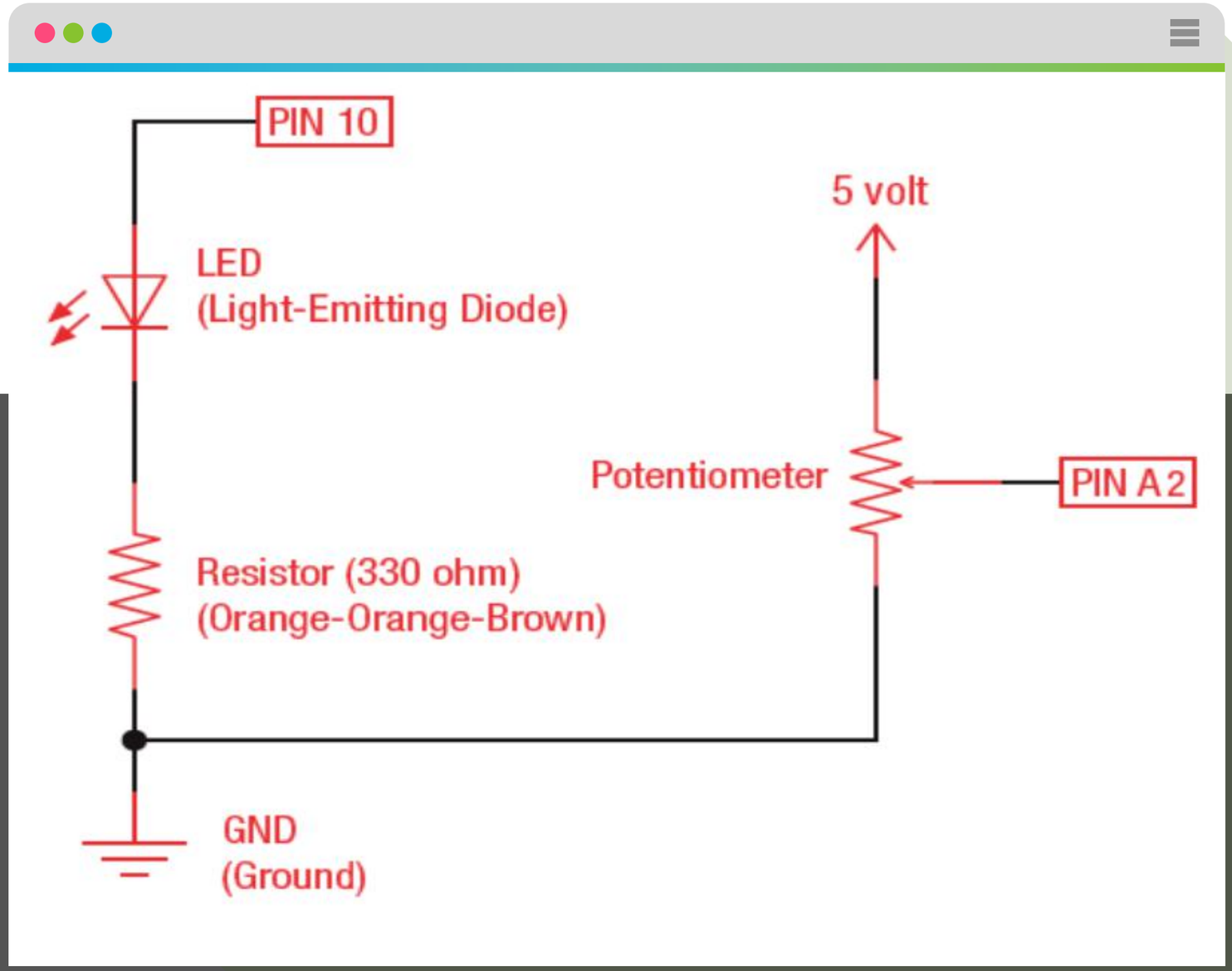
Green LED x1

330Ω Resistor x1



Potentiometer x1

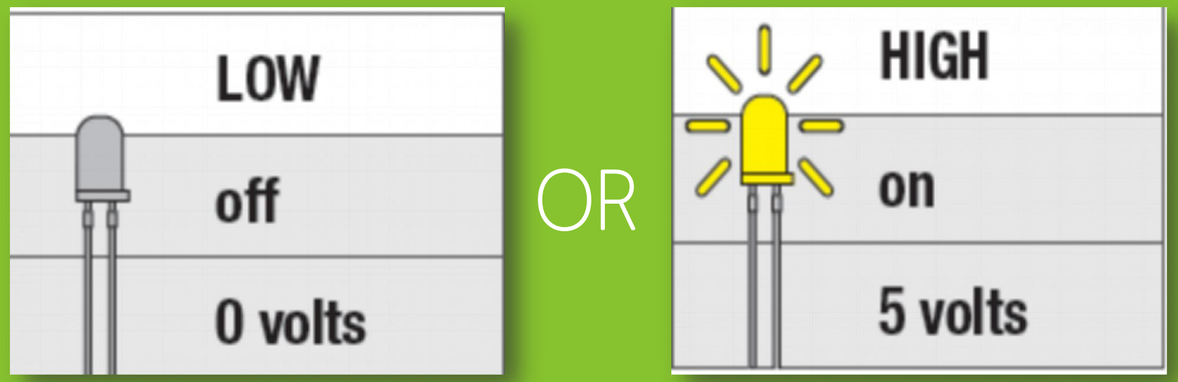
Note that analog pins A0 and A1 are reserved for communication with the Wi-Fi module, so please don't use them, otherwise the board won't work properly !

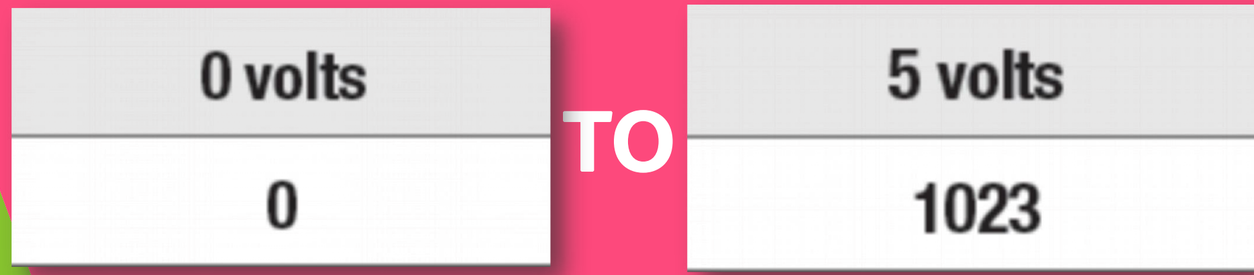


Potentiometer
(variable resistor)

The potentiometer has 3 pins, 2 outer-pins and 1 inner. When it's connected with 5 volts across its two outer pins, the middle pin outputs a voltage between 0V and 5V, depending on the position of the knob on the potentiometer. A potentiometer is a demonstration of a variable voltage divider circuit. In this circuit, you'll learn how to use a potentiometer to control the brightness of an LED.

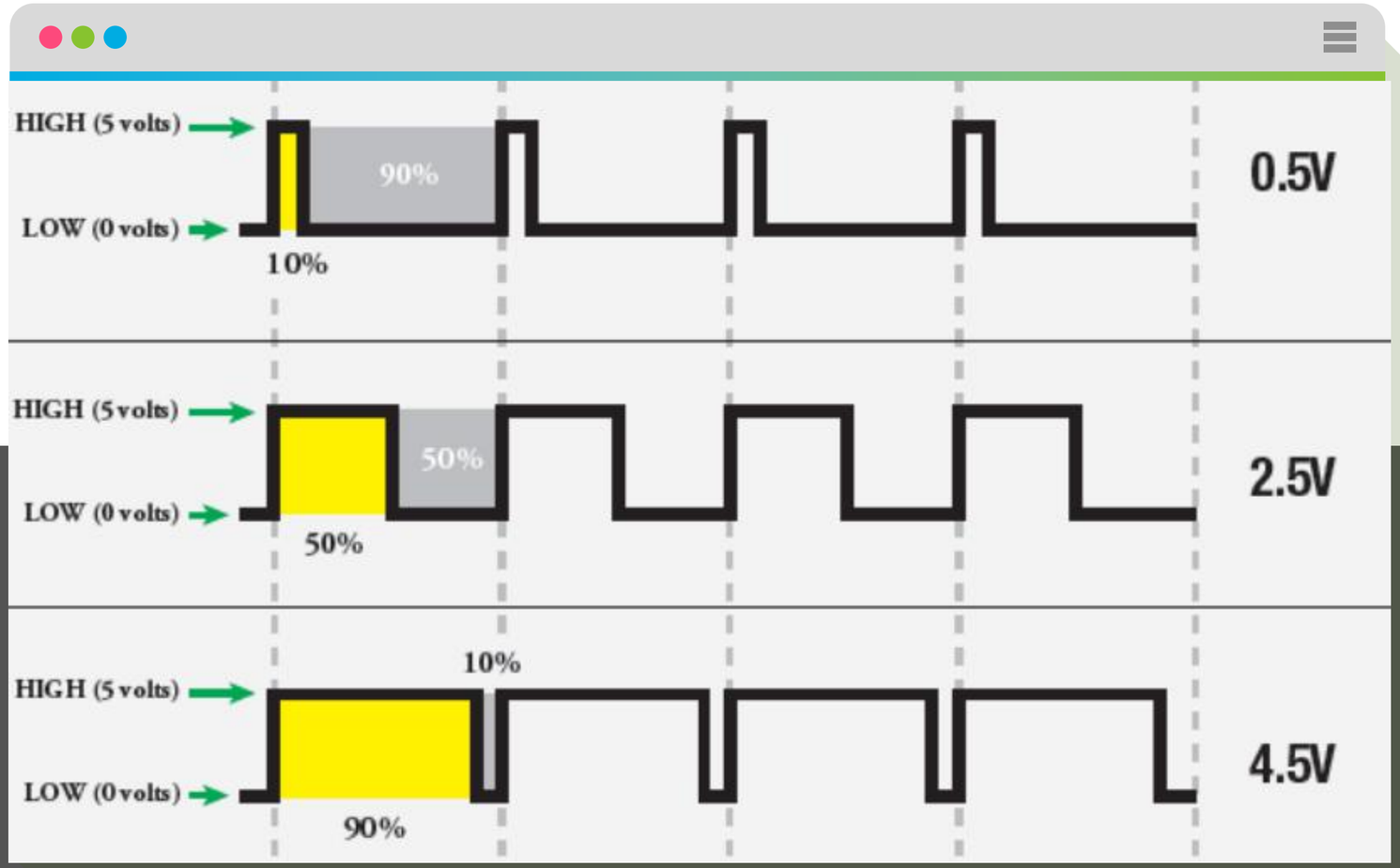
Digital or **Analog**





Many devices, such as LEDs and push-buttons, have only two possible states: “on” and “off”, or as they're known as "HIGH" (5 volts) and "LOW" (0 volts). The digital pins of a board are useful at transmitting/receiving signals to and from the outside world, and can even do tricks like simulated dimming (by blinking on and off really fast), and serial communications (transferring data to another device by encoding it as patterns of voltage HIGH and LOW).

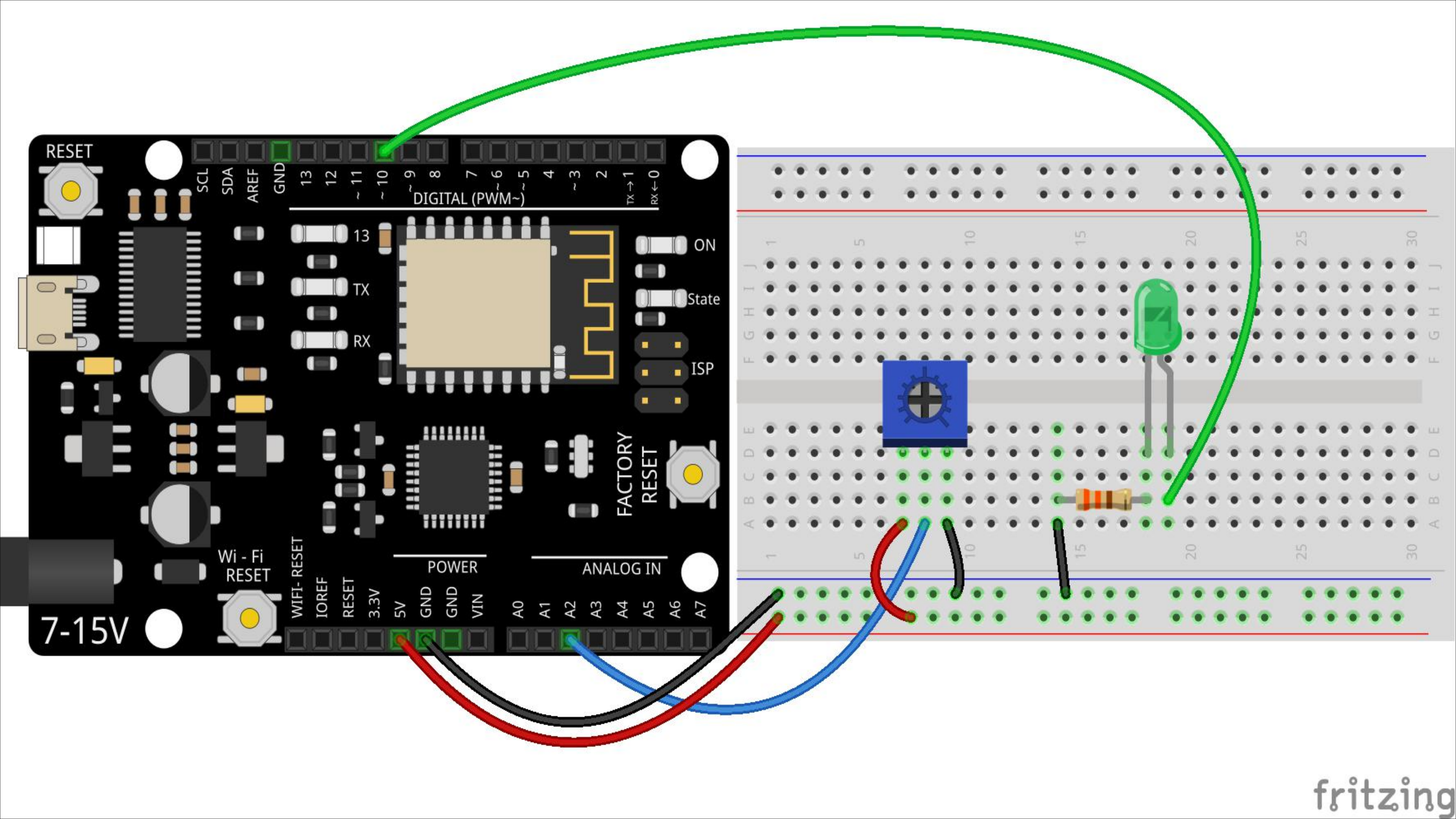
But there are also a lot of things out there that aren't just "on" or "off". Temperature levels, light intensity, etc. All have a continuous range of values between “HIGH” and “LOW”. For these, the board offers six analog inputs that translate an input voltage into a value that ranges from 0 (0 volts) to 1023 (5 volts). The analog pins are perfect for measuring all those "real world" values.



Is there a way for the IoT-Board to output analog voltages?

The AFF IoT Board does not have a true analog voltage output. But, because the Board is so fast, it can fake it using something called **PWM ("Pulse-Width Modulation")**. The pins on the Board with "~" next to them are PWM/Analog out compatible. The microcontroller is so fast that it can blink a pin on and off almost 1000 times/s.

PWM goes one step further by varying the amount of time that the blinking pin spends "HIGH" vs. the time it spends "LOW". If it spends most of its time "HIGH", a LED connected to that pin will appear bright. If it spends most of its time "LOW", the LED will look dim. Because the pin is blinking much faster than your eye can detect, the Board creates the illusion of a "true" analog output.


RESET
SCL
SDA
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TX → 1
RX ← 0
DIGITAL (PWM~)
13
TX
RX
ON
State
ISP
FACTORY RESET
Wi - Fi RESET
WIFI- RESET
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
A6
A7
ANALOG IN
7-15V
fritzing

AFF IoT Board folder > Circuits > Dimming_LED_using_Potentiometer

Dimming_LED_using_Potentiometer

```
#define GreenLedPin       10
#define PotentiometerPin  A2

void setup() {
  // put your setup code here, to run once:
  pinMode(GreenLedPin, OUTPUT); // configure the Green Led pin to be an output
}

void loop() {
  // put your main code here, to run repeatedly:
  int analogValue = analogRead(PotentiometerPin); // read the actual converted value (between 0 and 1023)
  // Note the analog pin can be read without declaring it as input

  analogValue = map(analogValue, 0, 1023, 0, 255); // transform the scale from 0 and 1023 to 0 and 255
  // becuase the analogWrite() function take parameter from 0 (0%) to 255 (100%) of duty cycle

  analogWrite(GreenLedPin, analogValue); // generate the PWM duty cycle according the needed intensity
}
```

Open your sketch:

Dimming_LED_using_Potentiometer

Code to note..

`int analogValue;` Variables must be declared before using them; here we're declaring a variable called analogValue, of type "int" (integer). Don't forget that variable names are case-sensitive!

`analogValue = analogRead(PotentiometerPin);` `analogRead()` function is used to read the value at an analog input pin. `analogRead()` takes one parameter, here the analog pin ("`PotentiometerPin`"), and returns a value ("`analogValue`") between `0` (`0` volts) and `1023` (5 volts).

`analogWrite(GreenLedPin, analogValue);` `analogWrite` function is used to generate a PWM signal in the `GreenLedPin` with duty cycle equal to `analogValue` (duty cycle is the percentage of High voltage (5V) over a period).

Code to note..

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

value: the variable to be mapped
fromLow: the lower bound of the value's current range
fromHigh: the upper bound of the value's current range
toLow: the lower bound of the value's desired range
toHigh: the upper bound of the value's desired range

```
analogValue = map(analogValue, 0, 1023, 0, 255)
```

Reading an analog signal using `analogRead()`, returns a number from `0` to `1023`. But to drive a PWM pin using `analogWrite()`, it requires a number from `0` to `255`. so it should "squeeze" the larger range into the smaller range using the `map()` function.

What you **should see**

You should see the LED becomes brighter or darker in accordance with the position of your potentiometer. If it isn't working, make sure you have assembled the circuit correctly and verified and uploaded the code to your board, or see the troubleshooting below:

Troubleshooting

Not Working
Make sure you haven't accidentally connected the resistive element in the potentiometer, to digital pin 2 rather than analog pin A2. (the row of pins beneath the power pins).

Sporadically Working
This is most likely due to a slightly dodgy connection with the potentiometer pins. Try pressing the potentiometer.

LED Not Lighting Up?
LEDs will only work in only one direction. Try taking it out and rotate it 180 degrees (don't worry, installing it the opposite way doesn't damage the LED).

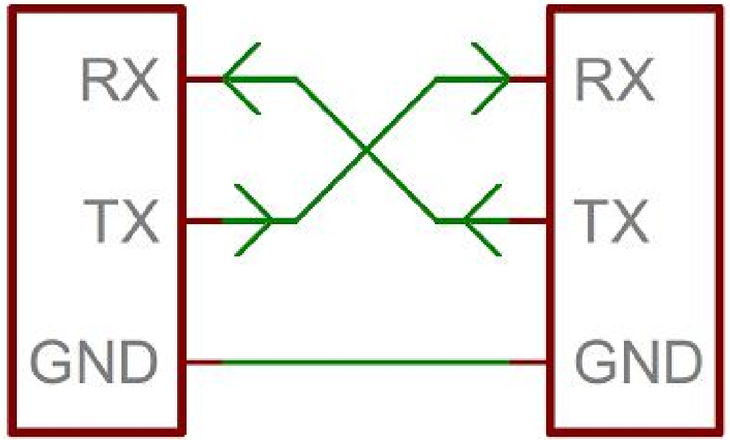
Real World Application

Most traditional volume knobs employ a potentiometer.

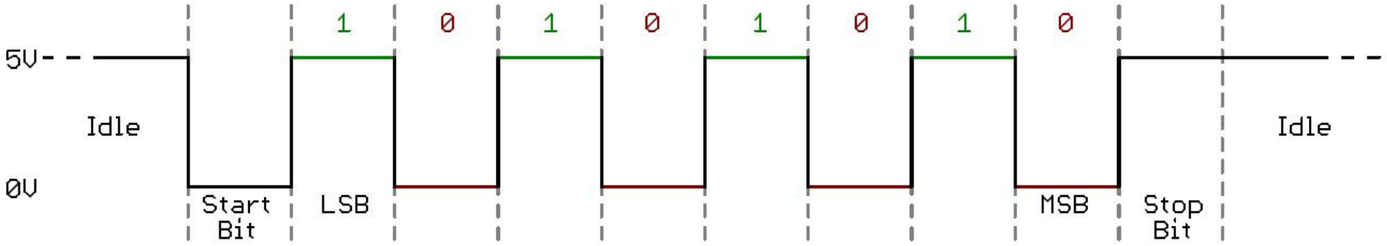
4 Using Serial Monitor

One of the most important things in Arduino IDE is the “Serial Monitor”.

Serial Communication



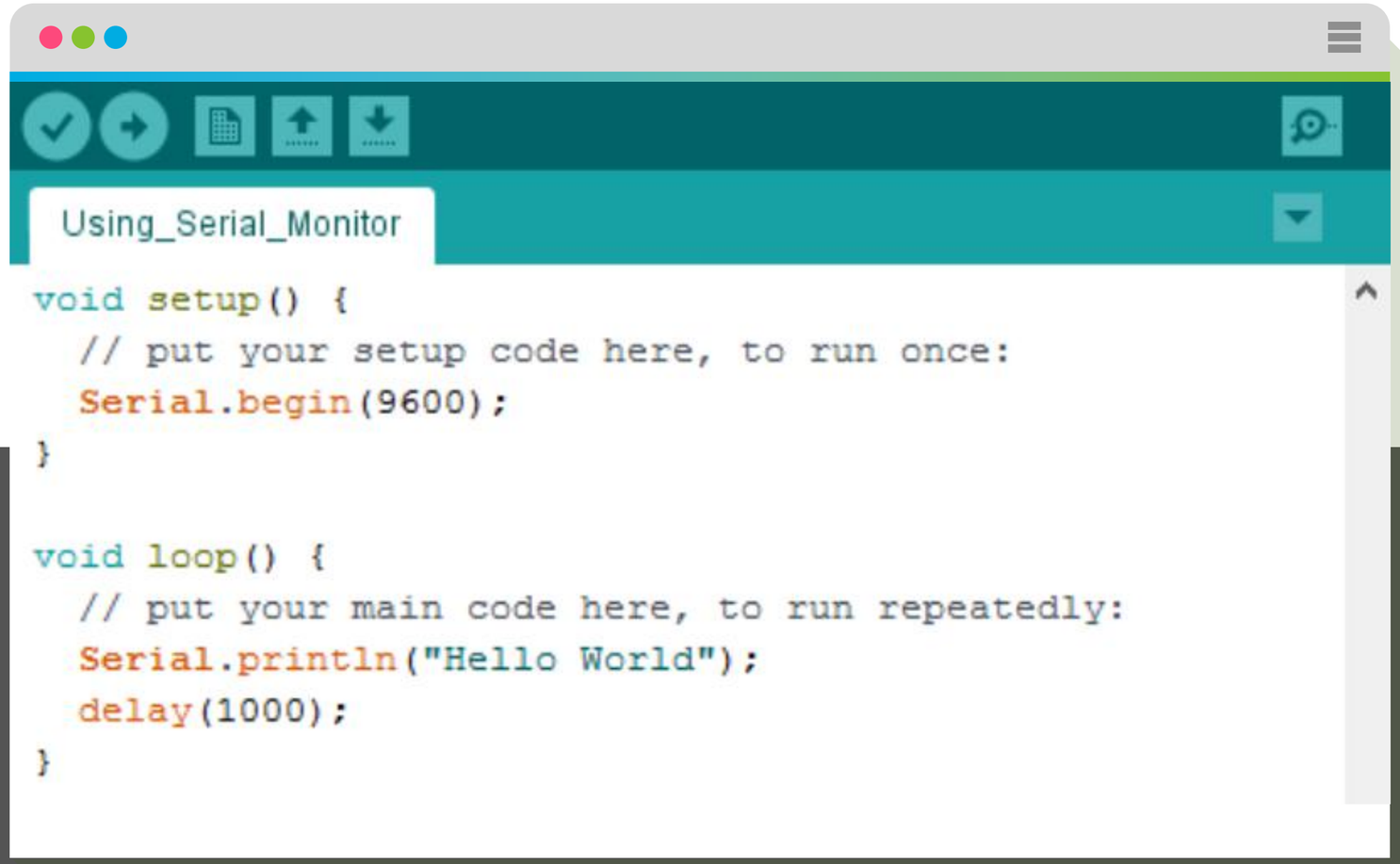
A serial bus consists of just two wires - one for sending data and another for receiving. As such, serial devices should have two serial pins: the receiver, **RX**, and the transmitter, **TX**.



In serial communication the data are sent one after other, until the end of all bytes.

The data are sent from the sender to receiver like pulling tennis balls from the container tube.

AFF IoT Board folder > Circuits > Using_Serial_Monitor

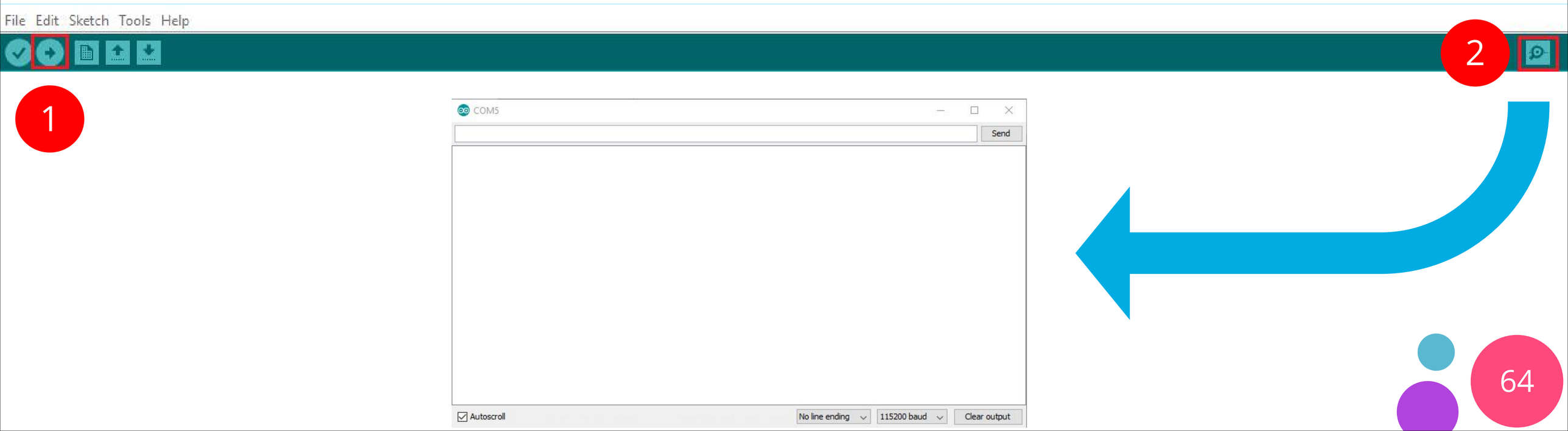


Open your sketch:

Using_Serial_Monitor

Serial Monitor

This example uses the Arduino IDE's Serial monitor. To open it, first upload the program then click the button which looks like a magnifier in a square. In order for the serial monitor to operate correctly it must be set to the same baud rate (speed in bits per second) as the code running. The code runs at 9600 baud rate; if the baud rate setting is not 9600, please change it to 9600.



Code to note..

`Serial.begin(9600)`: Before using the serial monitor, you must call `Serial.begin()` to initialize it. `9600` is the "baud rate", or communications speed. When two devices are communicating with each other, both must be set to the same speed. The serial monitor is used to debug the code also.

`Serial.println("Hello World")`: The `Serial.println()` command prints out anything written within the bracket, including variables of all types then moves to the next line.

However, `Serial.print()` prints everything on the same line.

for more info, visit http://arduino.cc/en/serial/print

What you **should see**

You should see a “Hello World” phrase printed on the Serial monitor every one second.

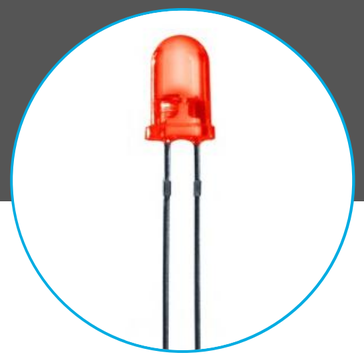
Troubleshooting

Nothing is Displayed
Try to exit the Serial monitor then start it again.

Gibberish is Displayed
This happens because the serial monitor is receiving data at a different speed than expected. Click the pull-down box that reads “xxxx baud" and change it to "9600 baud".

Nothing happen when opening Serial Monitor
Try to find the Serial monitor window in the windows taskbar by hovering the mouse over the Arduino icon then click on the Serial monitor tab.

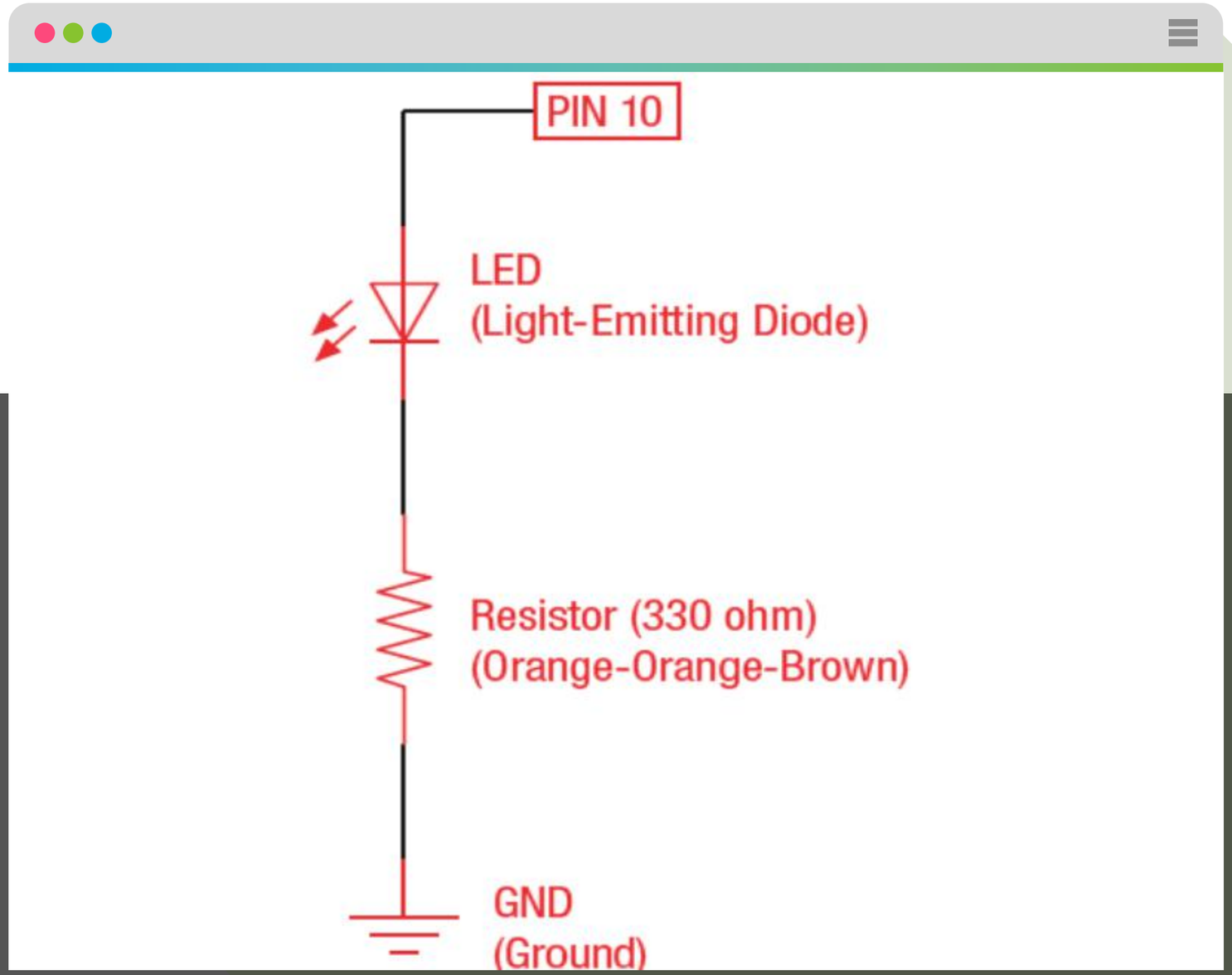
5 Controlling an LED (via “Internet”)



Red LED x1

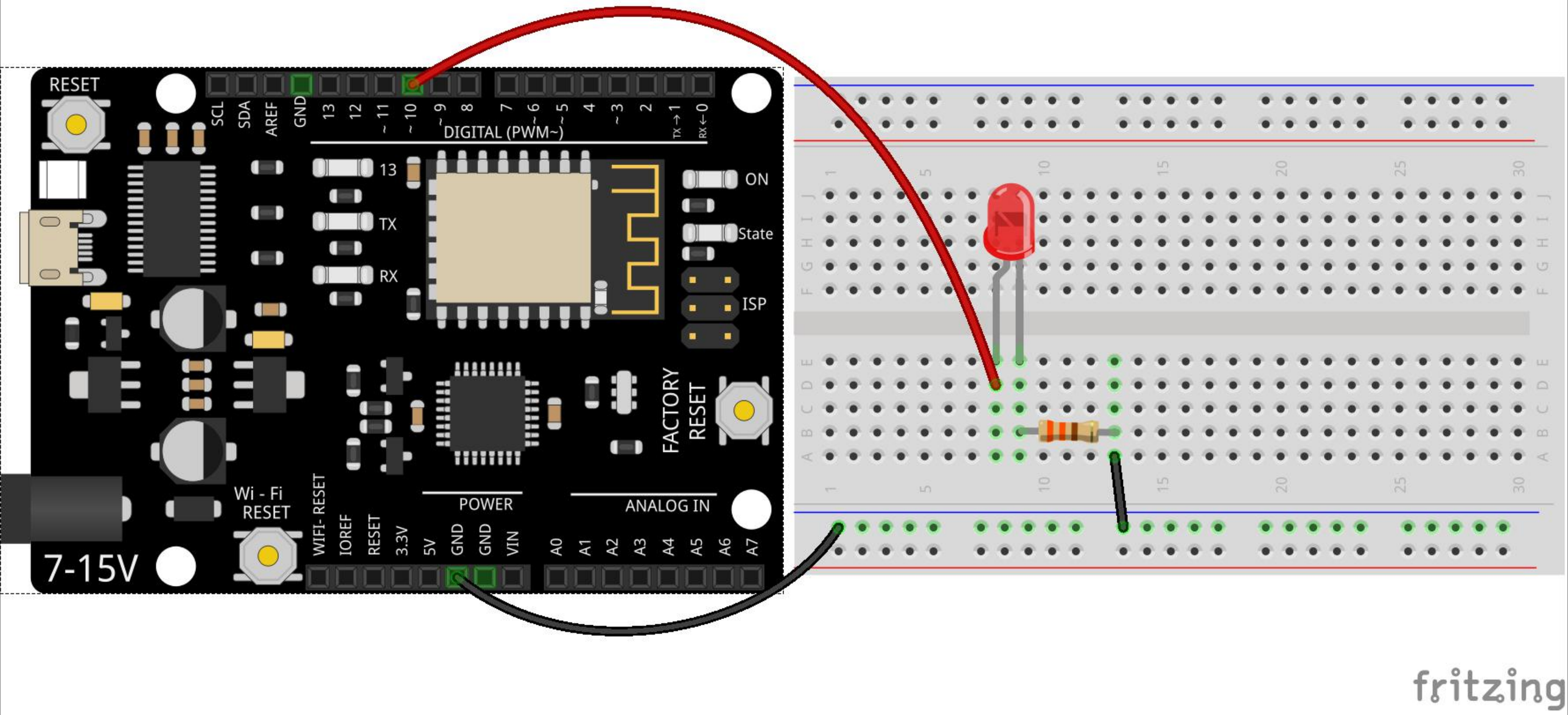
330Ω Resistor x1

Jumper wires x3

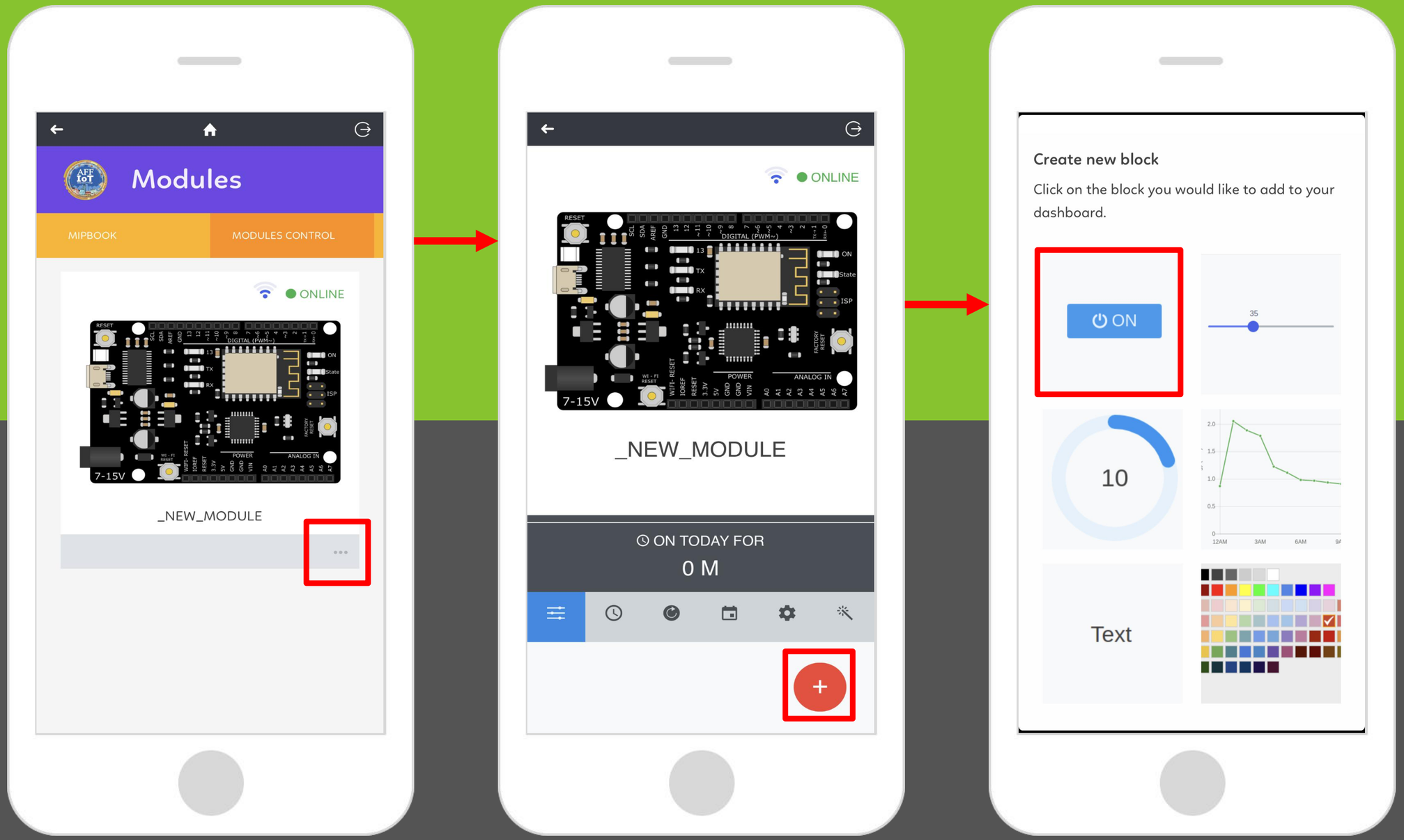


Control an LED

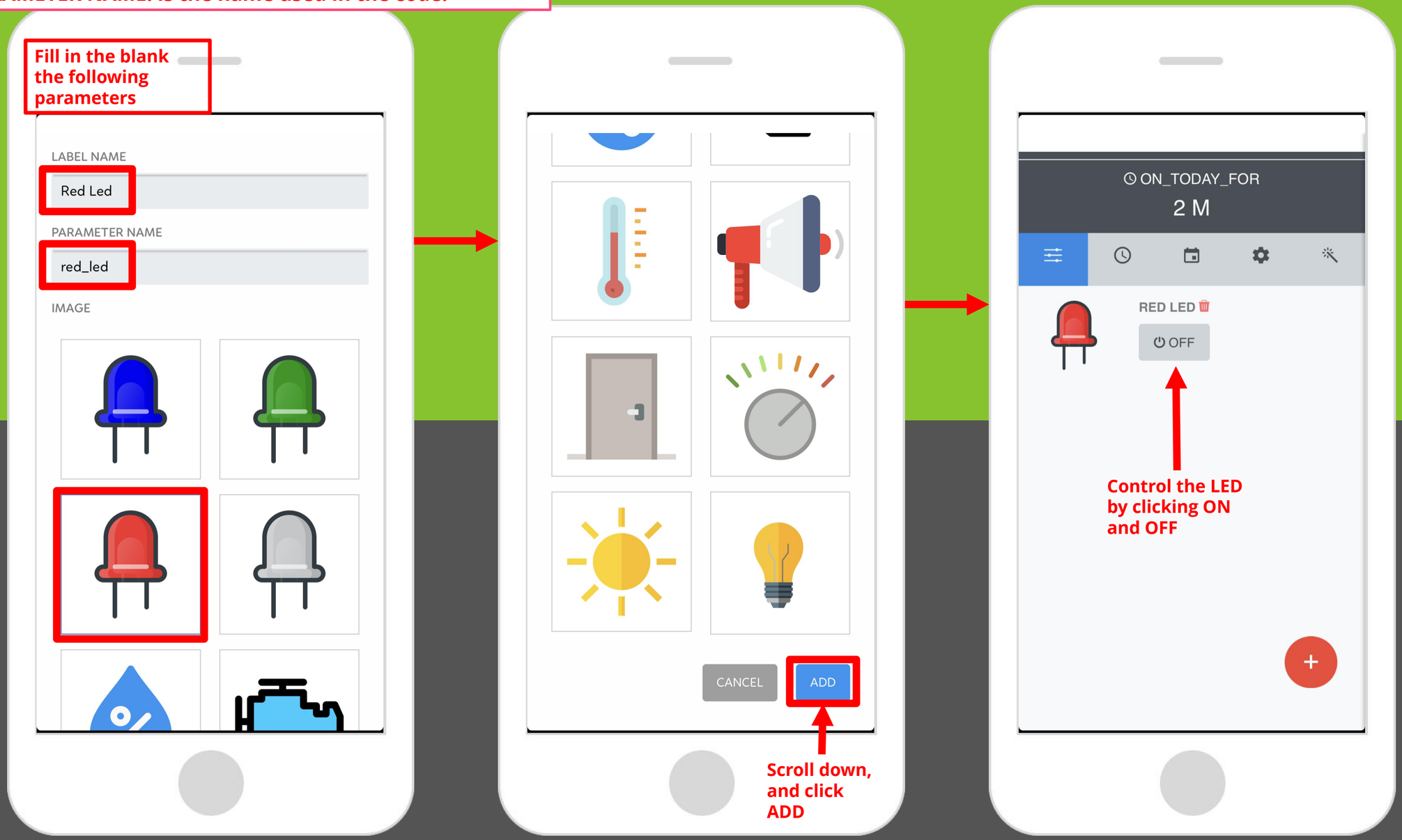
LEDs (light-emitting diodes) are small, but powerful light source that are used in many different applications. First we will work on blinking an LED. It is simple and basic function, yet, it is the basis for further complex functions.


RESET
SCL
SDA
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TX → 1
RX ← 0
DIGITAL (PWM~)
13
TX
RX
ON
State
ISP
FACTORY RESET
Wi - Fi RESET
WIFI- RESET
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
A6
A7
ANALOG IN
7-15V
A B C D E F G H I J
1 5 10 15 20 25 30

fritzing

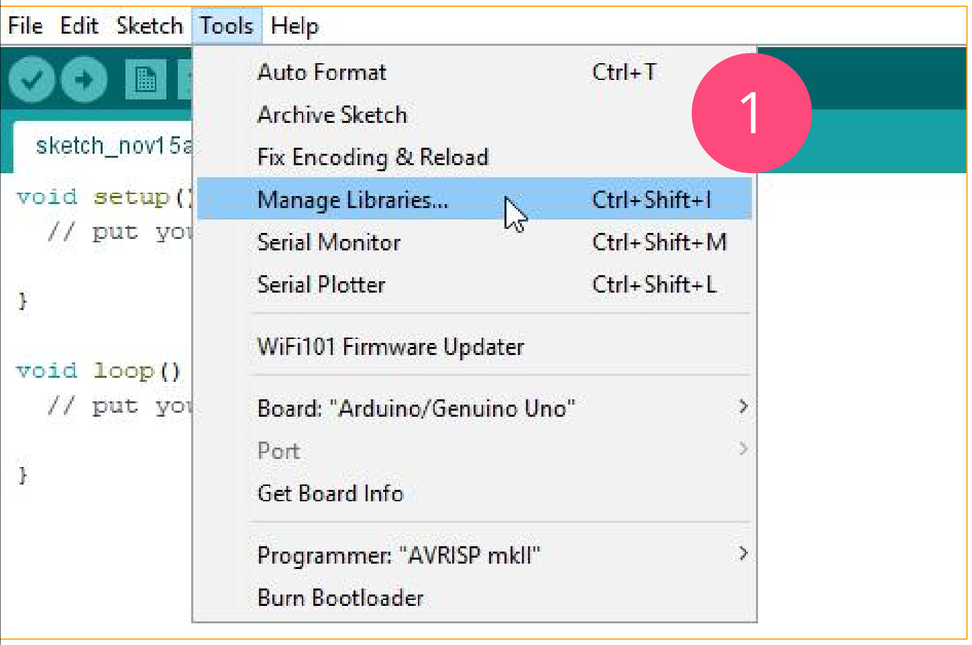

Modules
MIPBOOK
MODULES CONTROL
ONLINE
_NEW_MODULE
ONLINE
_NEW_MODULE
ON TODAY FOR
0 M
Create new block
Click on the block you would like to add to your dashboard.
ON
35
10
Text

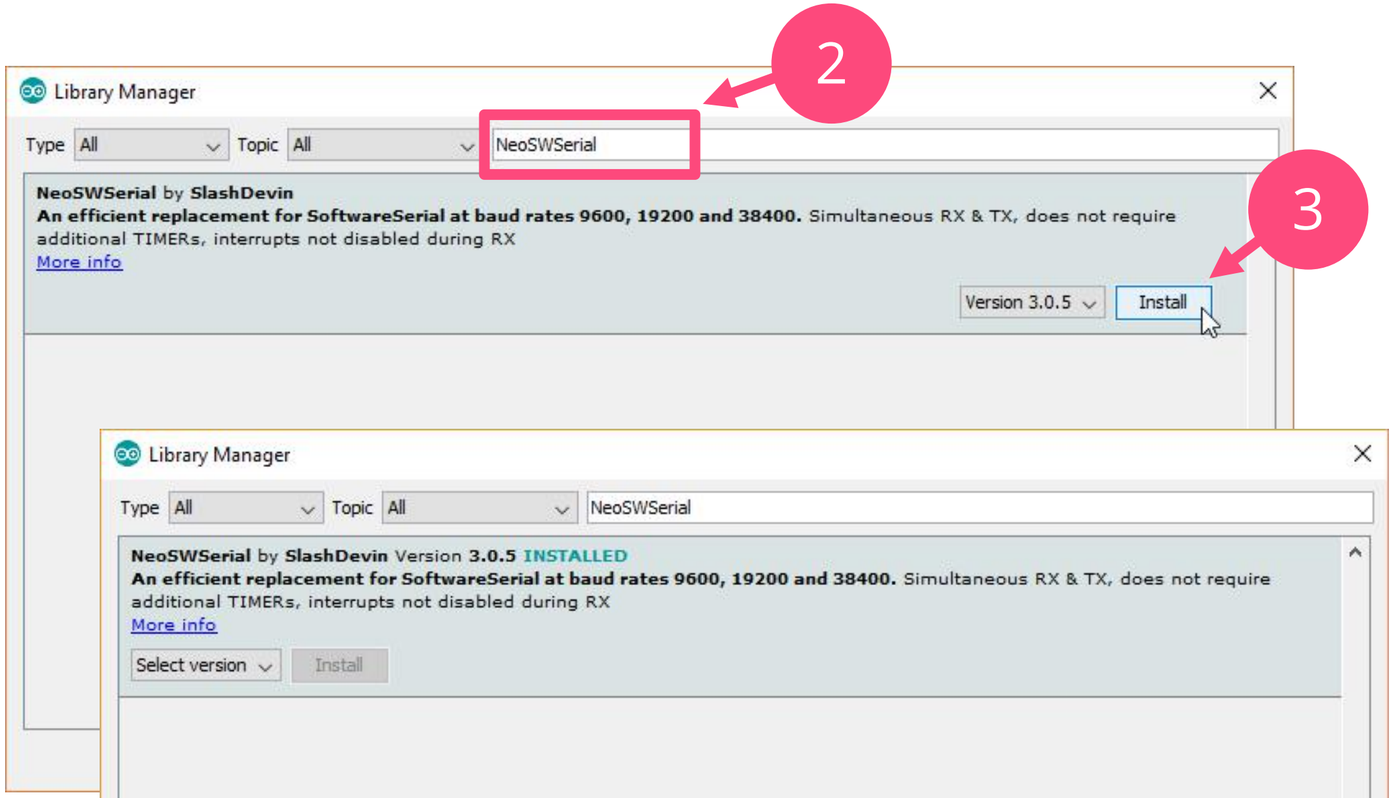
LABEL NAME: is the name that appears in the application.
PARAMETER NAME: is the name used in the code.

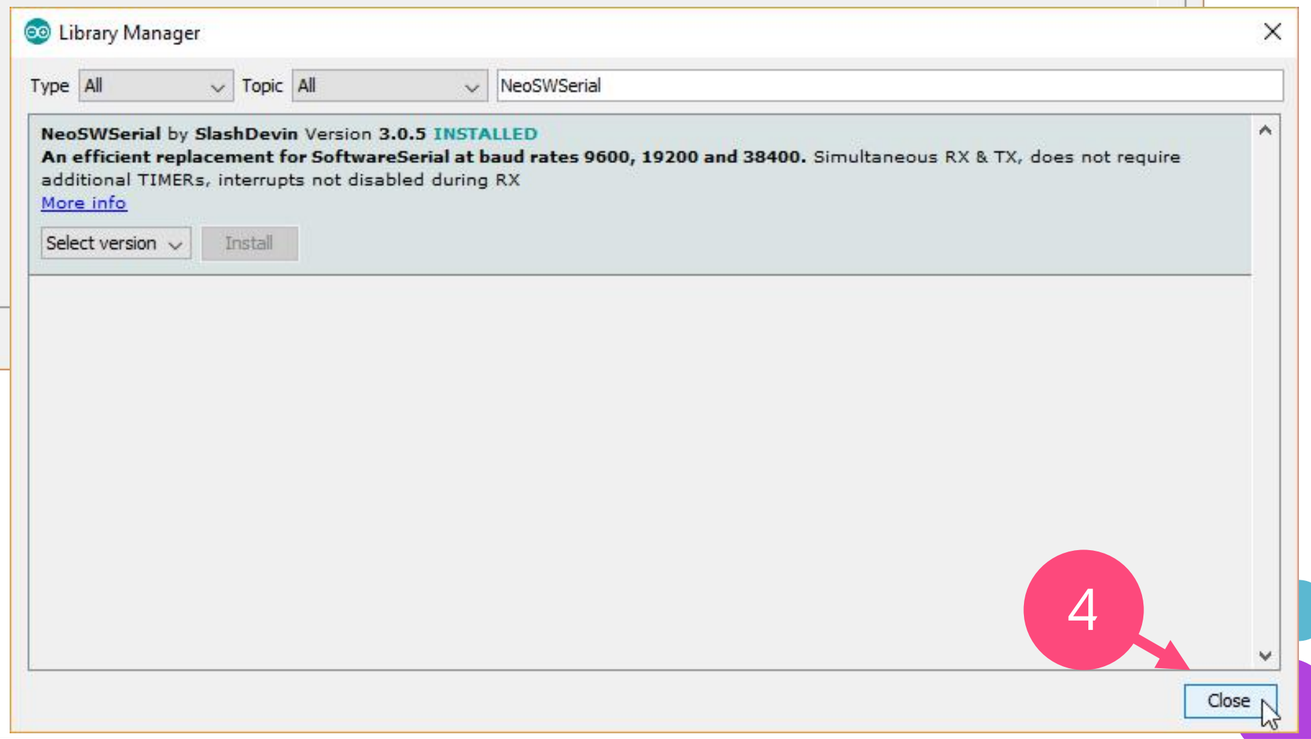


Before Start!

Instructions to add a new library (mandatory to communicate with the Wi-Fi module)







1) Open Arduino IDE and Click Tools, Manage Libraries.
2) Search for “NeoSWSerial”.
3) Then click Install.
4) Once download finish, click Close.

AFF IoT Board folder > Circuits > Controlling_an_LED

This is where you will find the Arduino code for each circuit

Controlling_an_LED

```
/*Start of mandatory lines of codes in each sketch*/
#define RX A0 // define the Receive pin (RX) to communicate with the WiFi module
#define TX A1 // define the Transmit pin (TX) to communicate with the WiFi module
#include <NeoSWSerial.h> // including the library to use the Software Serial rather than the Hardware Serial (Serial)
NeoSWSerial WiFiModule(RX, TX); //initialize the variable to use in communication with the WiFi module
/*End of mandatory lines of code*/

#define RedLedPin   10

void setup() {
  // put your setup code here, to run once:
  WiFiModule.begin(19200); // begin the communication between the WiFi module and the microcontroller on the board
  pinMode(RedLedPin, OUTPUT); // configure the pin connected to the Red Led to be an output
}

void loop() {
  // put your main code here, to run repeatedly:
  if (WiFiModule.available() > 0) // if the WiFi module receive data from the server
  {
    String Command = WiFiModule.readStringUntil('\n'); // read the command sent from the WiFi module to the microcontroller

    if (Command.indexOf("red_led=1") >= 0) // if the received command contains "red_led=1" turn on the LED
    {
      digitalWrite(RedLedPin, HIGH); // turn on the LED
    }
    if (Command.indexOf("red_led=0") >= 0) // if the received command contains "red_led=0" turn it off
    {
      digitalWrite(RedLedPin, LOW); // turn off the LED
    }
  }
}
//(for more info about indexOf function see https://www.arduino.cc/reference/en/language/variables/data-types/string/functions/indexof/)
```

Open your sketch: Controlling_an_LED

Code to note..

`#include <NeoSWSerial.h>:` This line of code consists of importing the Software Serial library, this library consists of making any two pins to be the transmit pin (Tx) and receive pin (Rx) of a serial communication between the board and another device (Wi-Fi module in AFF IoT board).

The function of this library is similar to the normal hardware `Serial`.

`NeoSWSerial WiFiModule(Rx, Tx):` Here a variable of type "`NeoSWSerial`" named "`WiFiModule`" is declared. It assigns the two pins `A0` and `A1` to be the transmit and receive pins for the serial communication with the Wi-Fi module

Code to note..

`WiFiModule.available()`: this function tests if the Wi-Fi module has received a command from the internet and returns the number of bytes received.

Note: if there is no data, the return value of this function is 0.

`String command = WiFimodule.readStringUntil('\n');` here a `String` variable named `command` is declared, and it will be assigned by the command text received from the Wi-Fi module. `readStringUntil('\n')` means that the board continue receiving data until the character `'\n'` is received (`'\n'` is the representation of the ENTER key on the keyboard).

Note: always use the this line of code same as written to get the command sent by the Wi-Fi module.

`if(command.indexOf("red_led") >= 0)`: the function `indexOf()` tests if the string between parenthesis (as parameter) exists in the caller string (command string in the example) and returns the index where it exits. If the parameter string doesn't exist in the caller string, the function returns -1.

What you **should see**

You should see your LED turn on and off when you click on the toggle button at the dashboard in the AFF IoT application. If it isn't, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting tips below.

Troubleshooting

LED Not Lighting Up?
LEDs will only work in one direction. Try taking it out and twisting it 180 degrees (don’t worry, installing it the opposite way doesn’t damage the LED).

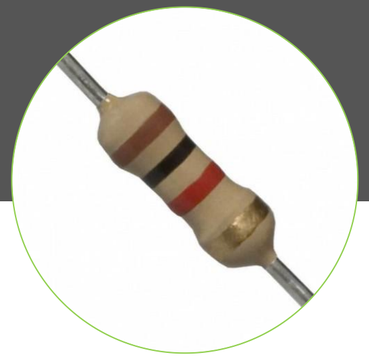
Program Not uploading?
It happens sometimes, the most likely cause is choosing wrong serial port, you can change this in Tools → Serial Port

Real World Application

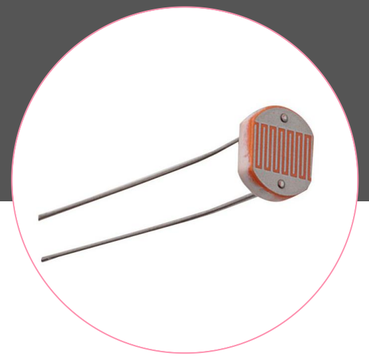
On/Off indicator in various electronic devices

6 Reading Light Intensity (via “Internet”)

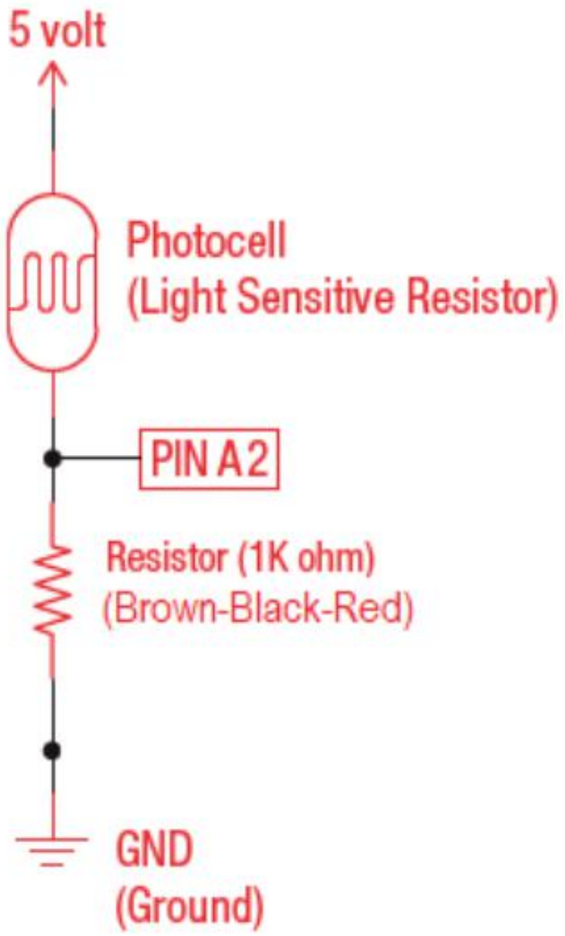
Jumper wires x5



1KΩ Resistor x1



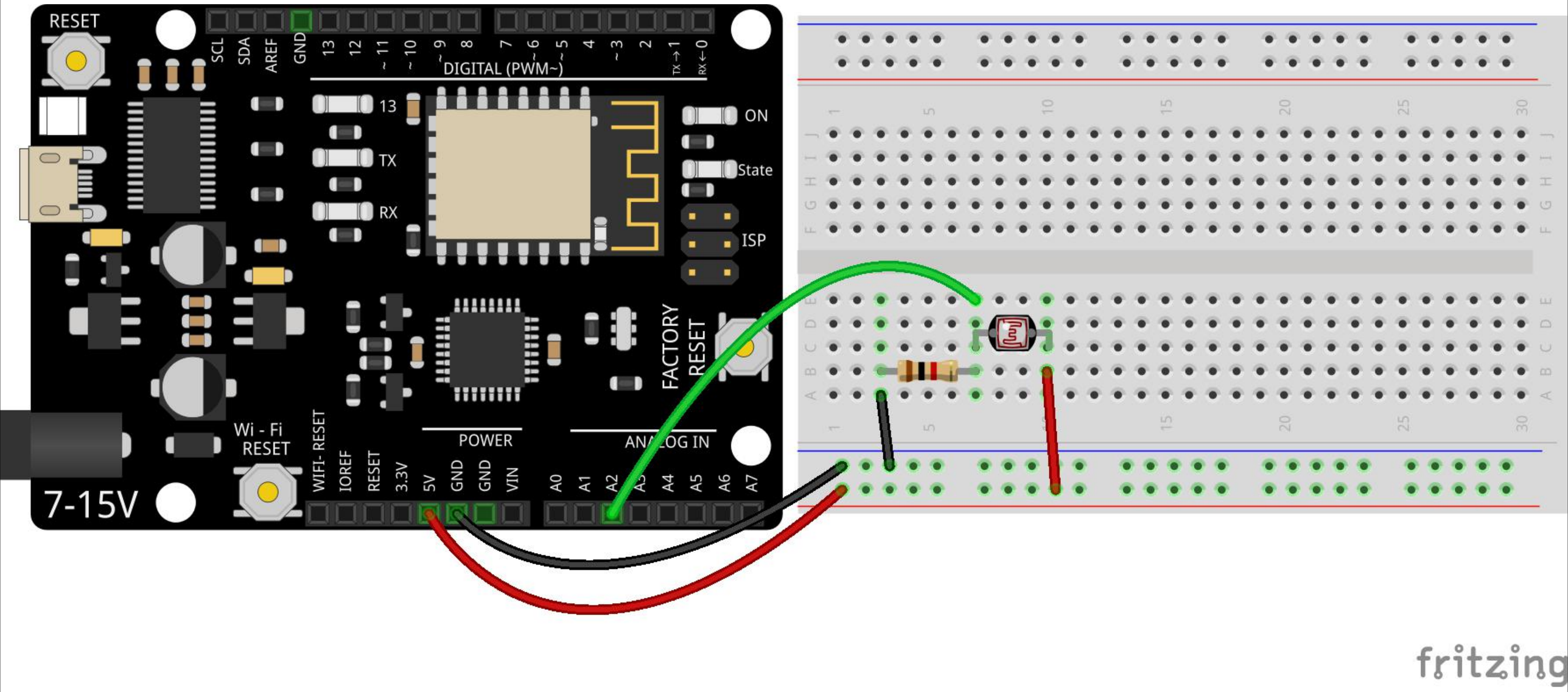
Photocell x1

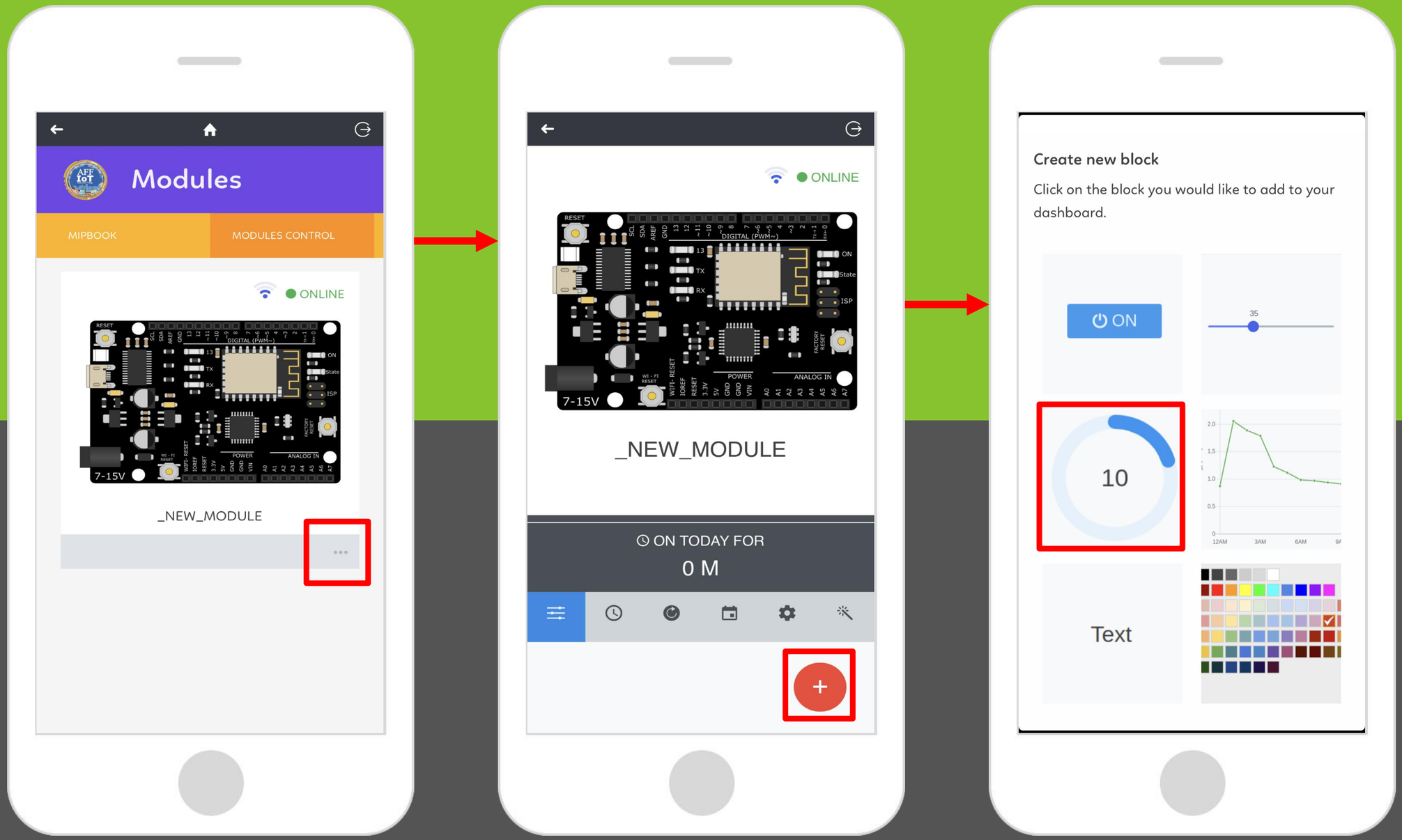


Photocell

A photocell is a resistor that its value varies with the change of light intensity. When bright, the resistor decreases, and the voltage (in the example circuit) increases.

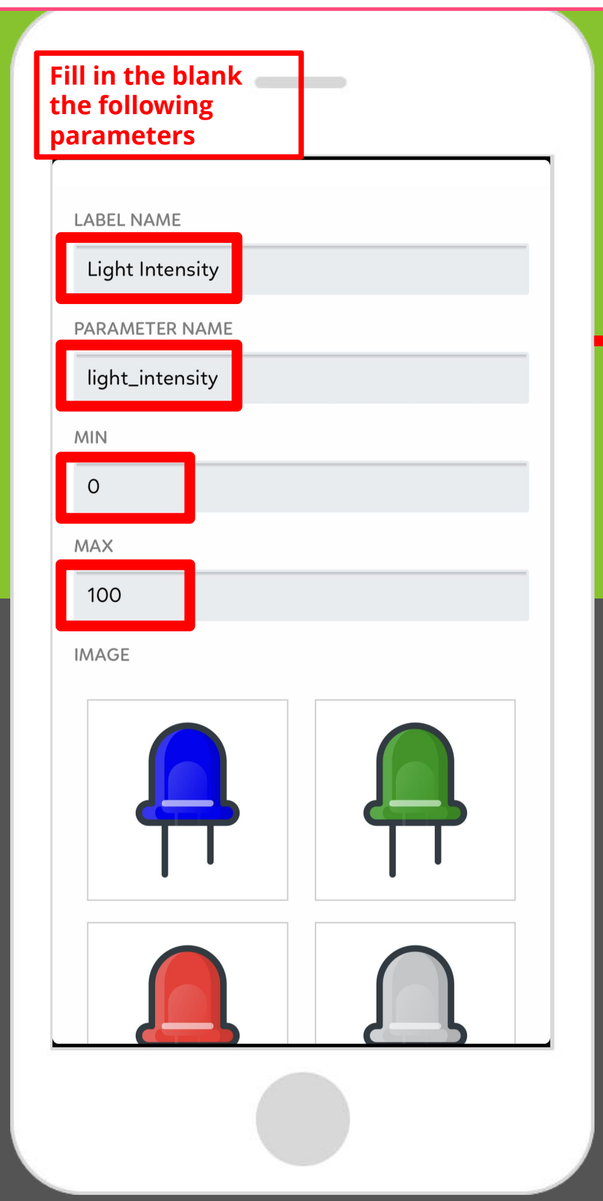
When it is dark, the resistor increases, and the voltage (in the example circuit) decreases.

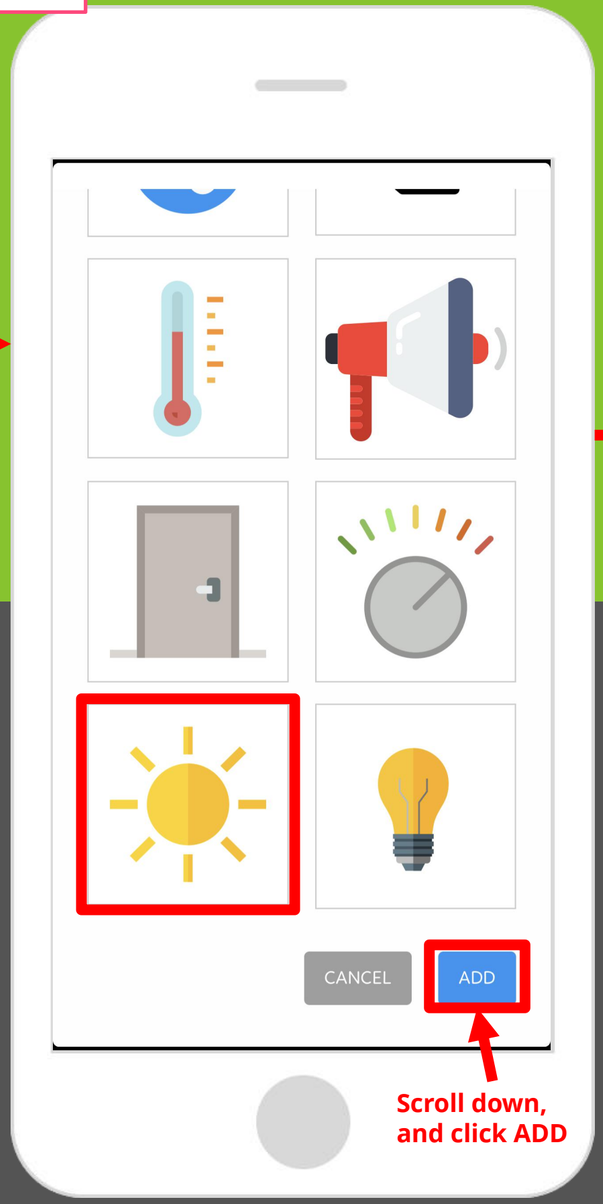

RESET
SCL
SDA
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TX → 1
RX ← 0
DIGITAL (PWM~)
13
TX
RX
ON
State
ISP
FACTORY RESET
Wi - Fi RESET
WIFI- RESET
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
A6
A7
ANALOG IN
7-15V
fritzing


Modules
MIPBOOK
MODULES CONTROL
ONLINE
_NEW_MODULE
ONLINE
_NEW_MODULE
ON TODAY FOR
0 M
Create new block
Click on the block you would like to add to your dashboard.
ON
10
Text

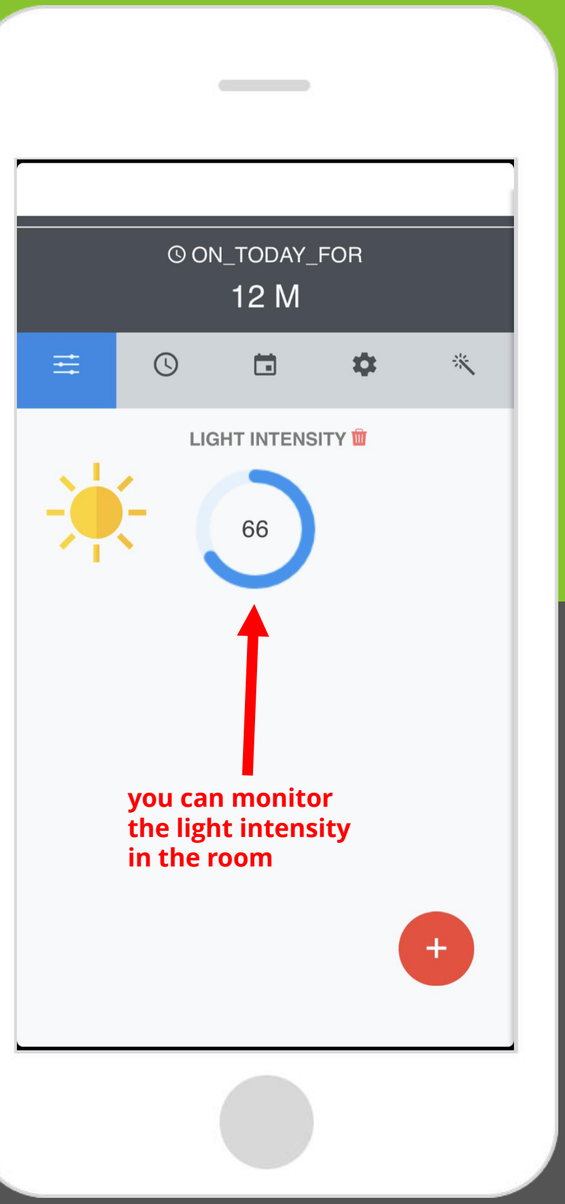
LABEL NAME: is the name that appears in the application.
PARAMETER NAME: is the name used in the code.

Fill in the blank the following parameters





Scroll down, and click ADD



you can monitor the light intensity in the room

AFF IoT Board folder > Circuits > Reading_Light_Intensity

Reading_Light_Intensity

```
/*Start of mandatory lines of codes in each sketch*/
#define RX A0 // define the Receive pin (RX) to communicate with the WiFi module
#define TX A1 // define the Transmit pin (TX) to communicate with the WiFi module
#include <NeoSWSerial.h> // including the library to use the Software Serial rather than the Hardware Serial (Serial)
NeoSWSerial WiFiModule(RX, TX); //initialize the variable to use in communication with the WiFi module
/*End of mandatory lines of code*/

#define LightSensorPin   A2

void setup() {
  // put your setup code here, to run once:
  WiFiModule.begin(19200); // begin the communication between the WiFi module and the microcontroller on the board
}

void loop() {
  // put your main code here, to run repeatedly:
  int lightIntensity; // declare an integer to read the voltage

  lightIntensity = analogRead(LightSensorPin); // read the actual converted value (between 0 and 1023)
  lightIntensity = map(lightIntensity, 0, 1023, 0, 100); // transform the scale from 0 and 1023 to 0% and 100%

  WiFiModule.println("light_intensity=" + String(lightIntensity)); // send the light intensity value to the server

  delay(3000); // wait for 3 second (3000ms = 3s)
}
```

Open your sketch:
Reading_Light_Intensity

What you **should see**

You should see the gauge value increase or decrease in accordance with how much light the photocell is reading. If it isn't working, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting tips below.

Troubleshooting

It isn't responding to changes in light
Given that the spacing of the wires on the photocell is not standard, it is easy to misplace it on the breadboard. Double check it's in the right place.

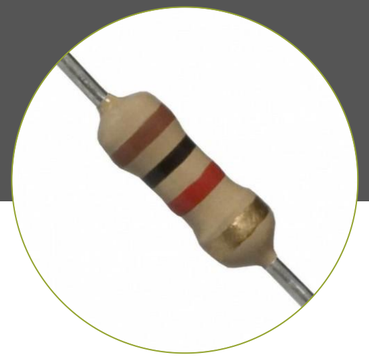
Still not quite working
You may be in a room which is either too bright or dark.
Try turning the lights on or off to see if this helps. Or if you have a flashlight near by, give that a try.

Real World Application

A street lamp uses light sensor to detect when to turn light ON

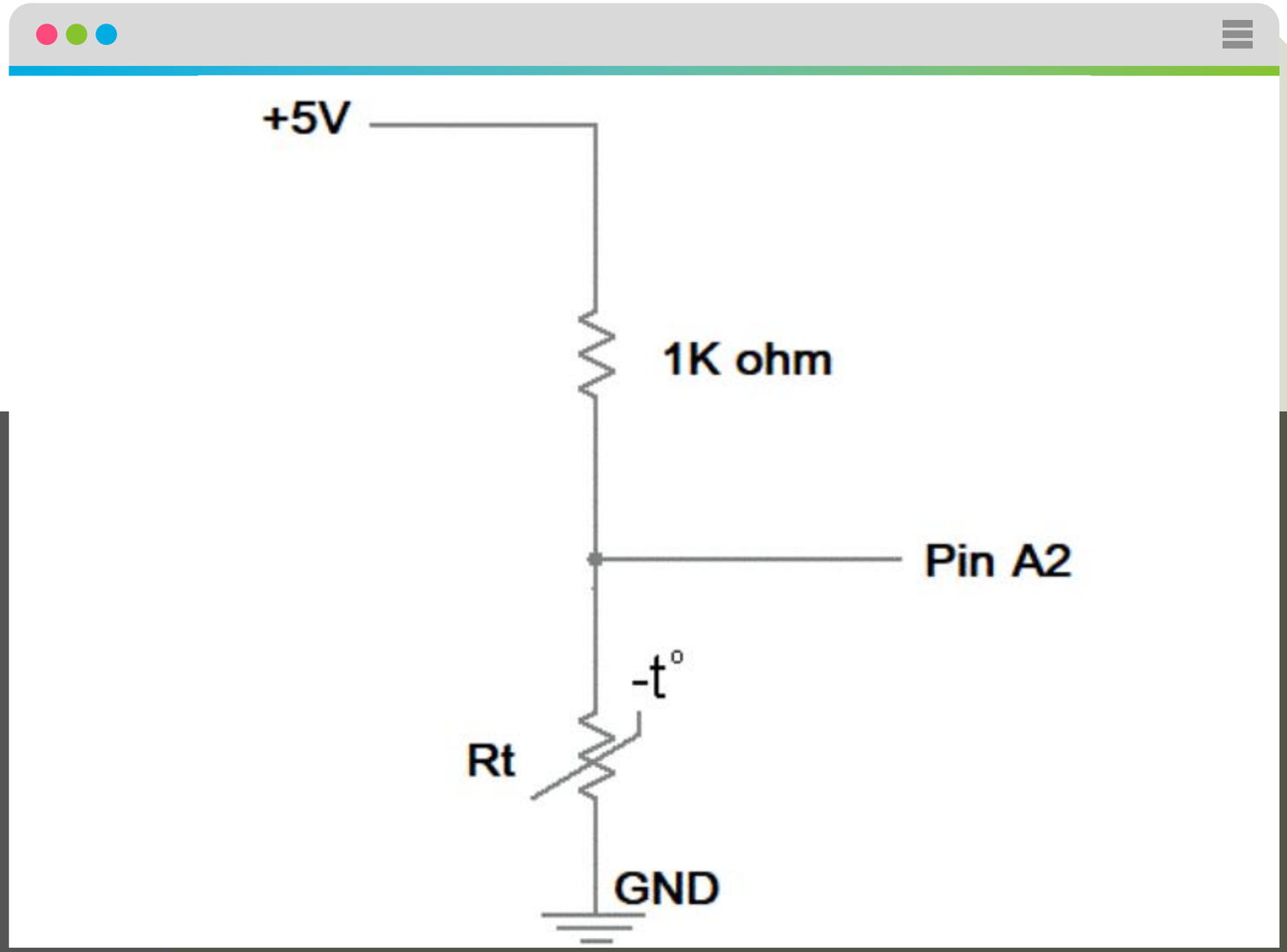
7 Monitoring Temperature (via “Internet”)

Thermistor x1



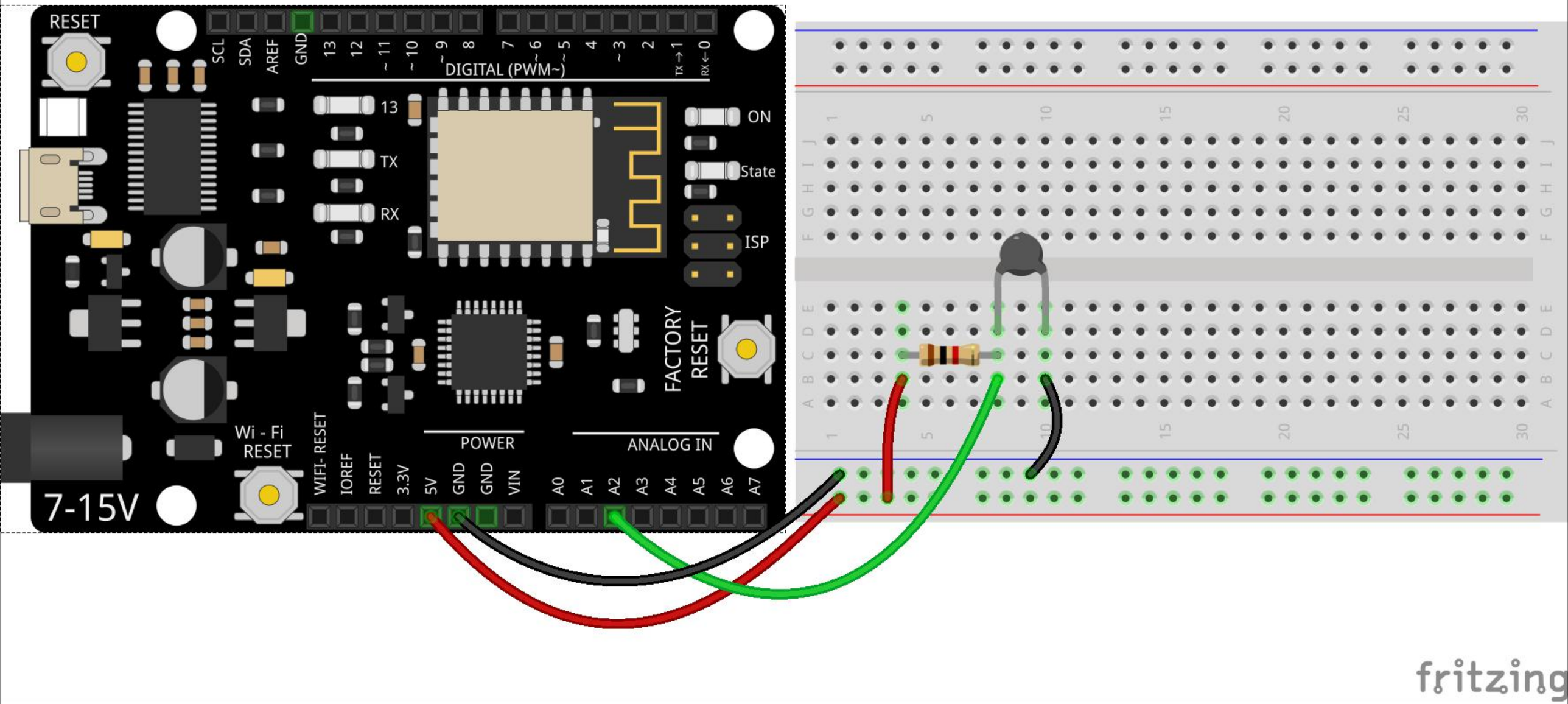
1KΩ resistor x1

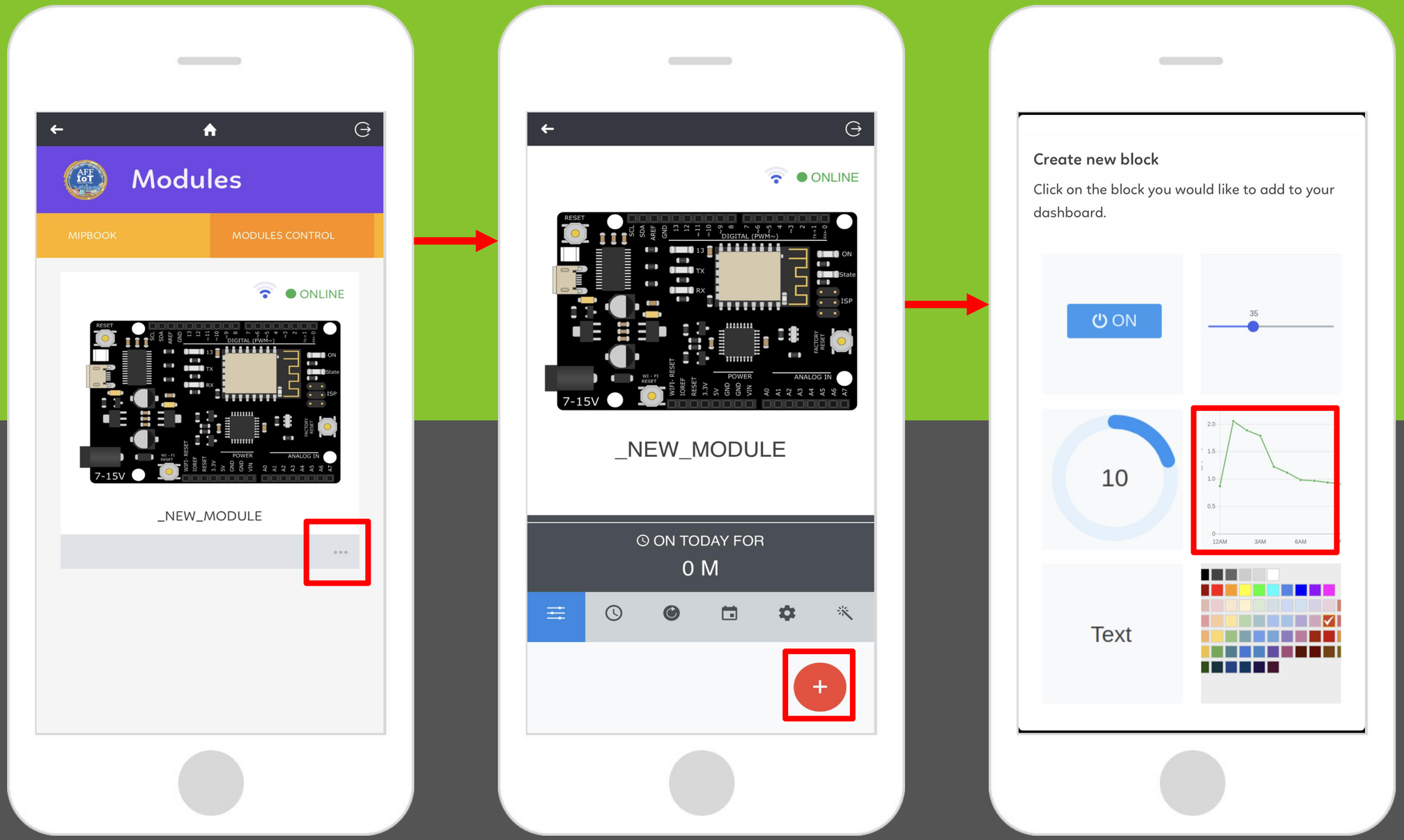
Jumper wires x5



Thermistor

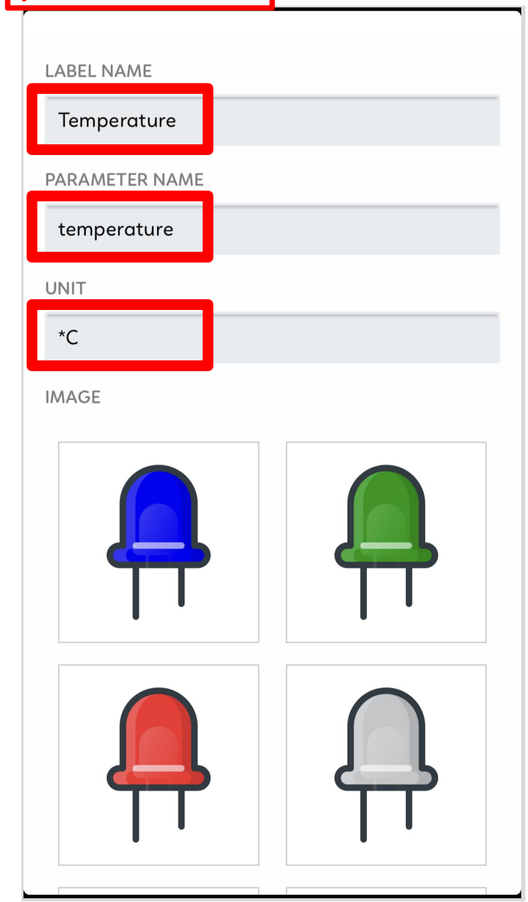
A **thermistor** is a type of resistor whose resistance is dependent on temperature, more sensitive than standard resistors. The word is a portmanteau of *thermal* and *resistor*. Thermistors are widely used as temperature sensors (Negative Temperature Coefficient or **NTC** type typically), self-regulating heating elements (Positive Temperature Coefficient or **PTC** type typically). With **NTC thermistors**, resistance ***decreases*** as temperature increases.

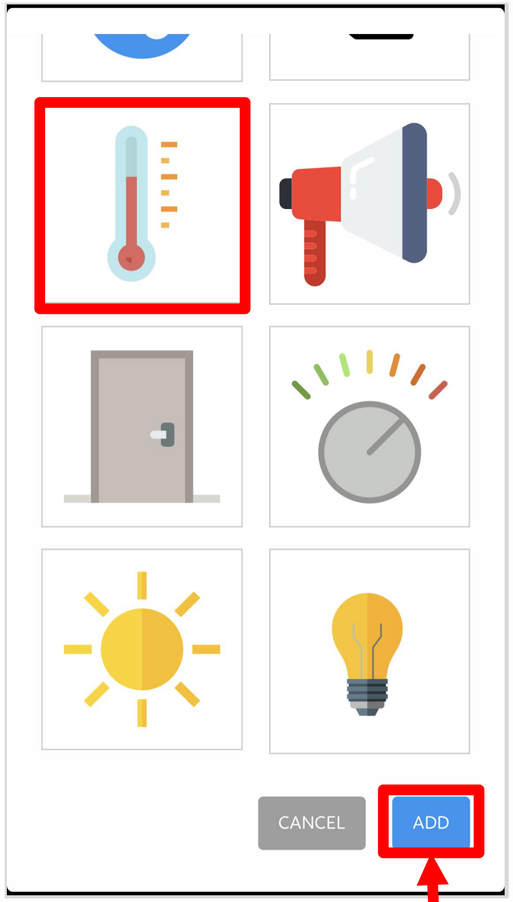

RESET
SCL
SDA
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TX → 1
RX ← 0
DIGITAL (PWM~)
13
TX
RX
ON
State
ISP
FACTORY
RESET
Wi - Fi
RESET
WIFI- RESET
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
A6
A7
7-15V


Modules
MIPBOOK
MODULES CONTROL
ONLINE
_NEW_MODULE
ONLINE
_NEW_MODULE
ON TODAY FOR
0 M
Create new block
Click on the block you would like to add to your dashboard.
ON
35
10
Text

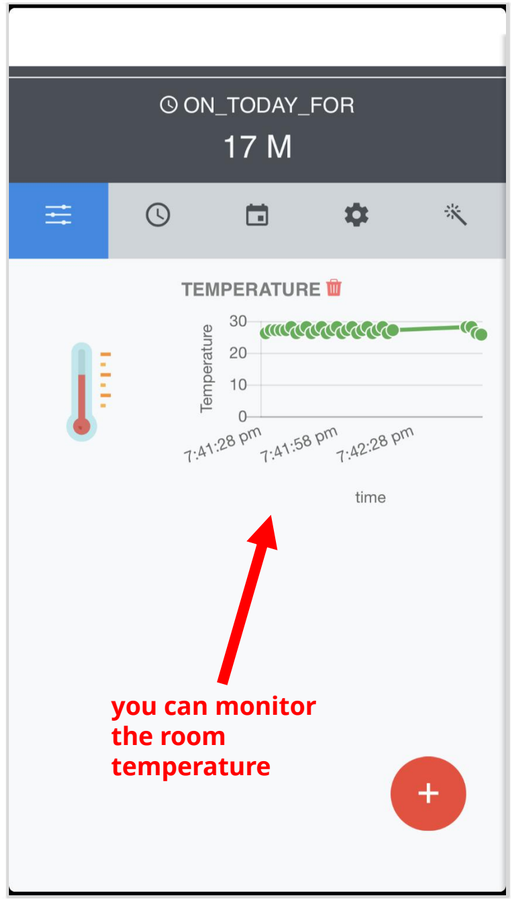
LABEL NAME: is the name that appears in the application.
PARAMETER NAME: is the name used in the code.

Fill in the blank the following parameters





Scroll down, and click ADD



you can monitor the room temperature

AFF IoT Board > Circuits > Monitoring_Temperature

Monitoring_Temperature

```
/*Start of mandatory lines of codes in each sketch*/
#define RX A0 // define the Receive pin (RX) to communicate with the WiFi module
#define TX A1 // define the Transmit pin (TX) to communicate with the WiFi module
#include <NeoSWSerial.h> // including the library to use the Software Serial rather than the Hardware Serial (Serial)
NeoSWSerial WiFiModule(RX, TX); //initialize the variable to use in communication with the WiFi module
/*End of mandatory lines of code*/

#define TemperatureSensorPin   A2

void setup() {
  // put your setup code here, to run once:
  WiFiModule.begin(19200); // begin the communication between the WiFi module and the microcontroller on the board
  Serial.begin(9600); // begin the communication between the board and the PC
}

void loop() {
  // put your main code here, to run repeatedly:
  float Voltage; // declare a decimal variable to store the read voltage
  int Temperature; // declare an integer variable to calculate the temperature
  Voltage = analogRead(TemperatureSensorPin) * 0.0048828125; // voltage = analog_value * (5/1024);

  Temperature = -21.231 * (Voltage - 3.765);
  // this equation is the linearized form of the temperature equation between 0 and 50 degrees (specific to the thermistor in the kit)

  Serial.print("temperature: "); // print on the serial monitor "Temperature: " and stay on the same line
  Serial.println(Temperature); // print on the serial monitor the actual temperature then go to the next line

  WiFiModule.println("temperature=" + String(Temperature)); // send the Temperature value to the server

  delay(3000); // wait for 3 second (3000ms = 3s)
}
```

Open your sketch: Monitoring_Temperature

What you **should see**

You should be able to read the temperature your temperature sensor is detecting, on the serial monitor in the Arduino IDE and on your app. If it isn't working, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting tips.

Troubleshooting

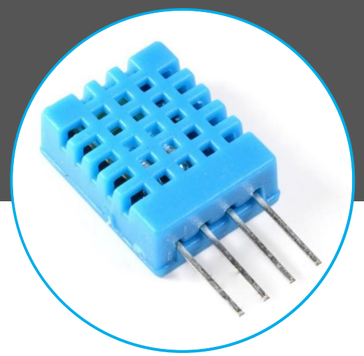
Gibberish is Displayed
This happens because the serial monitor is receiving data at a different speed than expected. Click the pull-down box that reads "*** baud" and change it to "9600 baud".

Temperature Value is Unchanging
Try rubbing the sensor with your fingers to heat it up or pressing a bag of ice close to it to cool it down.

Real World Application

Climate control systems use a temperature sensor to monitor and maintain cooling/heating settings.

8 Measuring Humidity & Temperature (via “Internet”)

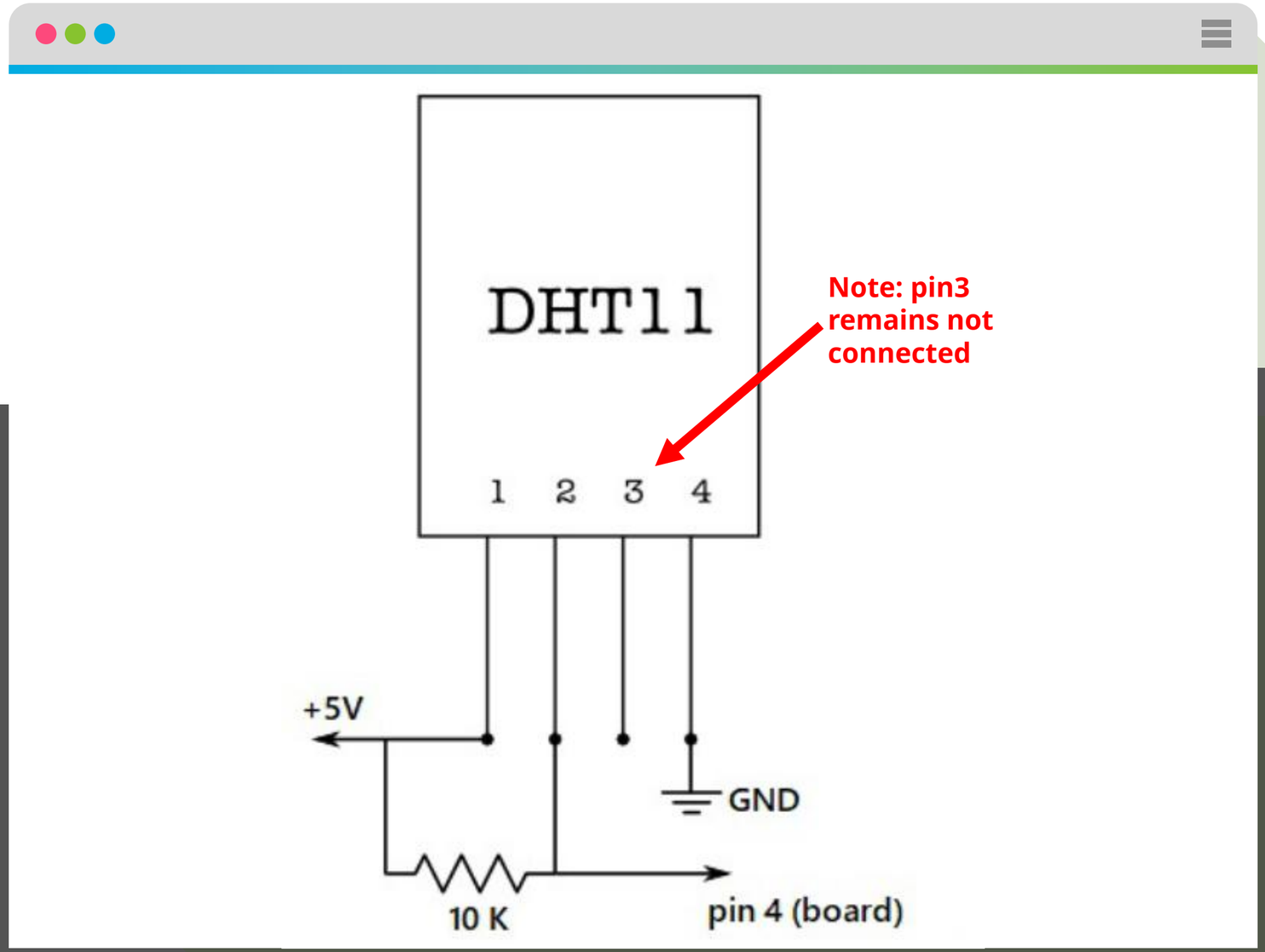


DHT11 x1

“Humidity and Temperature sensor”

10KΩ resistor x1

Jumper wires x6

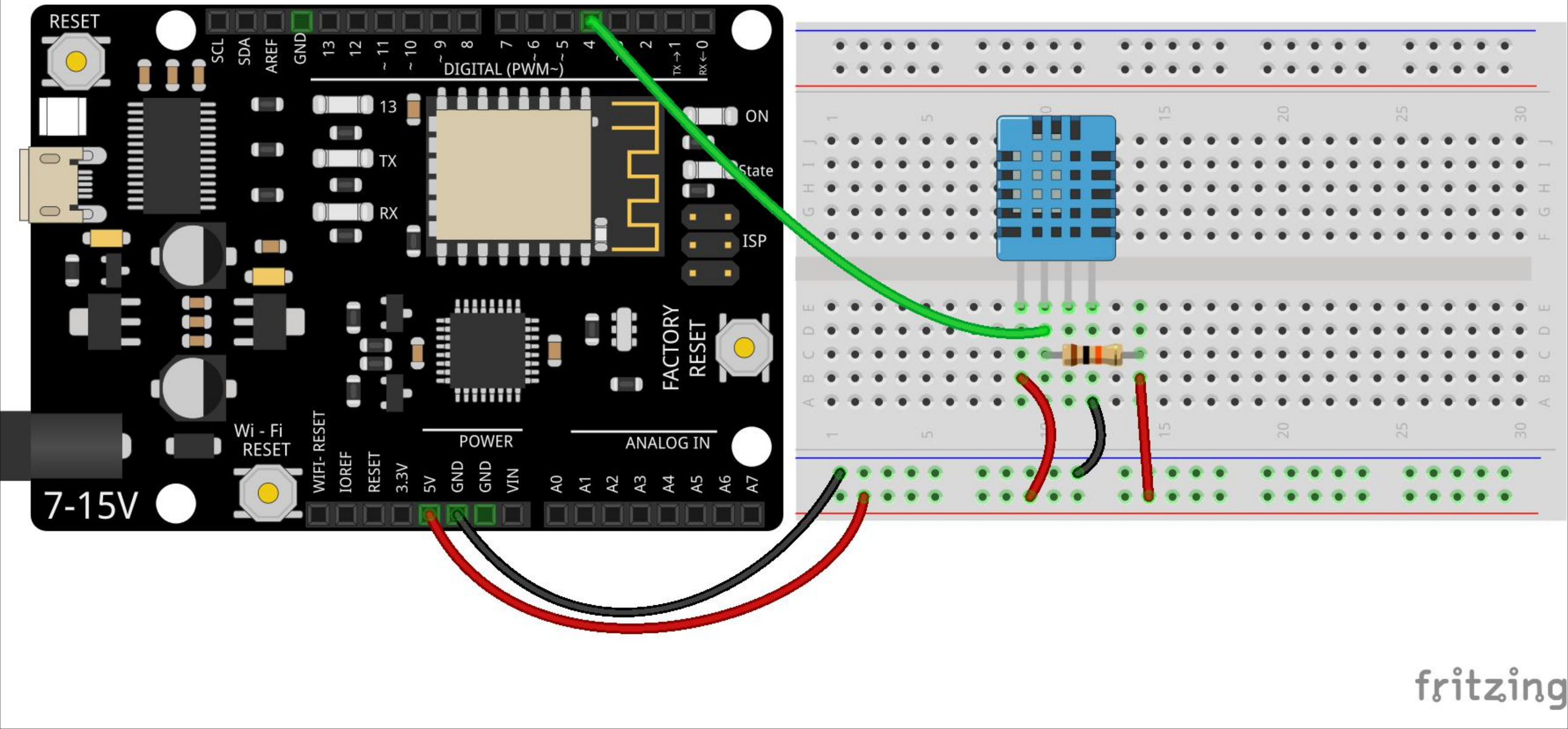


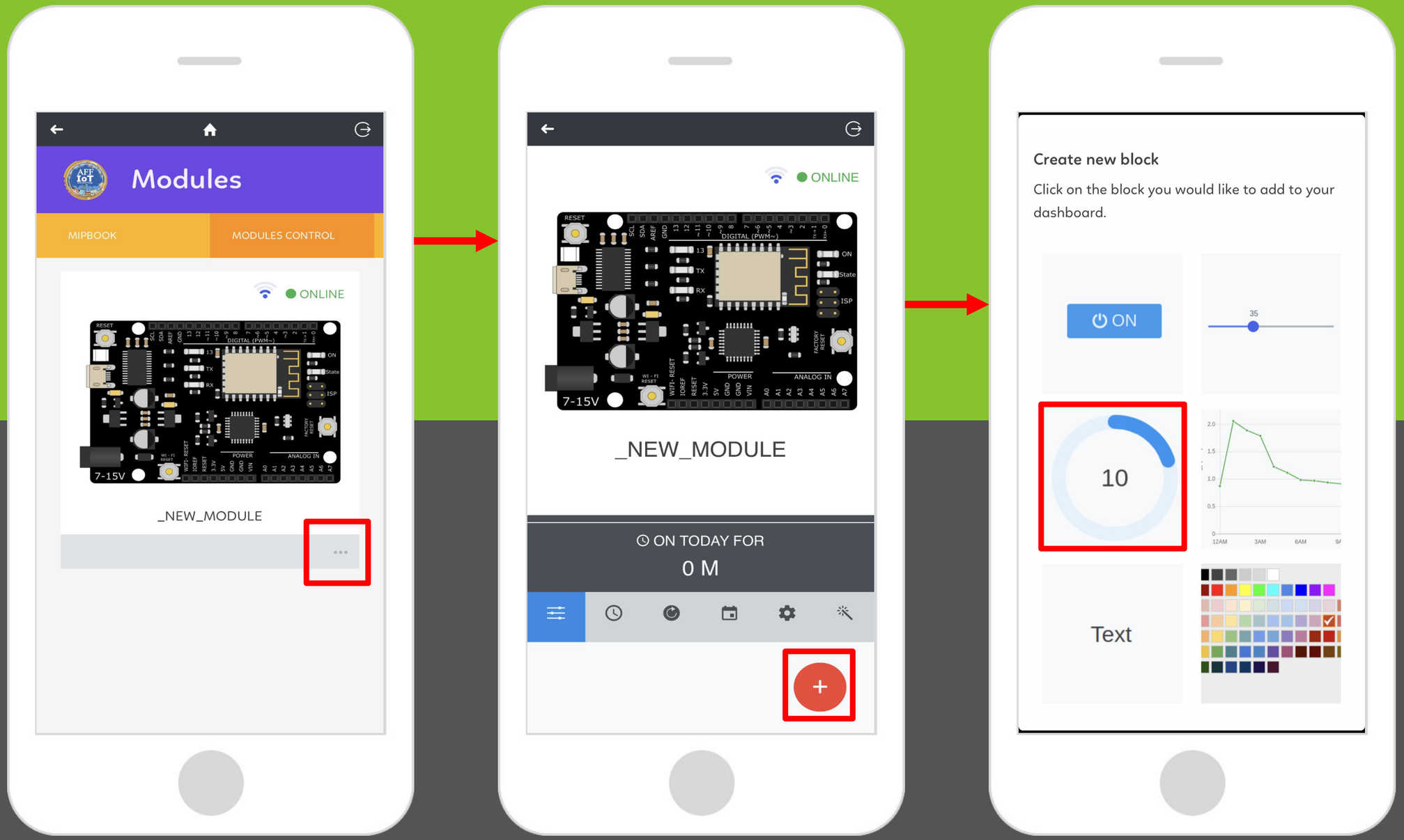
DHT11 sensor

DHT11 Temperature and Humidity Sensor

Features a temperature and humidity sensor complex with a calibrated digital signal output.

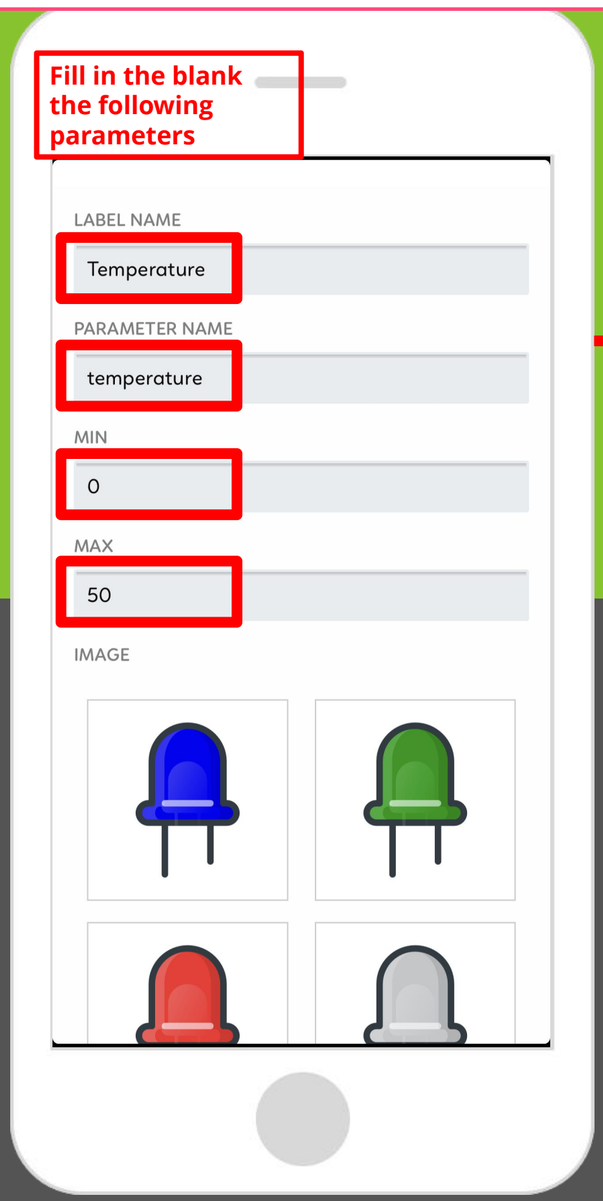
The sensor includes humidity measurement component and an NTC temperature measurement component.

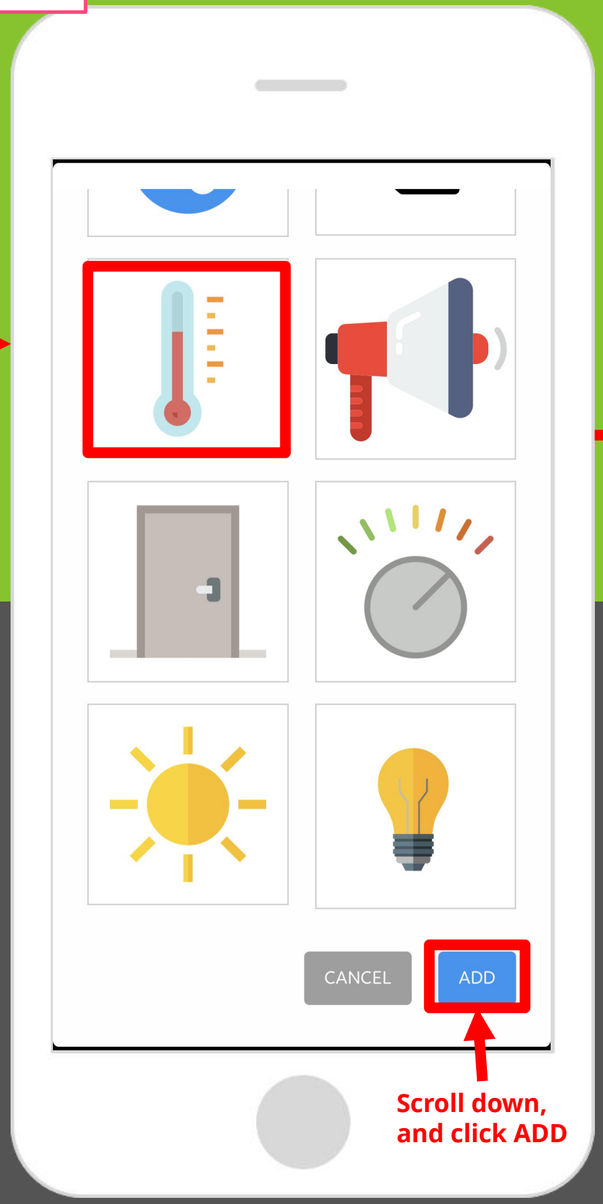

RESET
SCL
SDA
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
2
TX → 1
RX ← 0
DIGITAL (PWM~)
13
TX
RX
ON
State
ISP
FACTORY RESET
Wi - Fi RESET
WIFI- RESET
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
A6
A7
7-15V
fritzing


Modules
MIPBOOK
MODULES CONTROL
ONLINE
_NEW_MODULE
ONLINE
_NEW_MODULE
ON TODAY FOR
0 M
Create new block
Click on the block you would like to add to your dashboard.
ON
35
10
Text

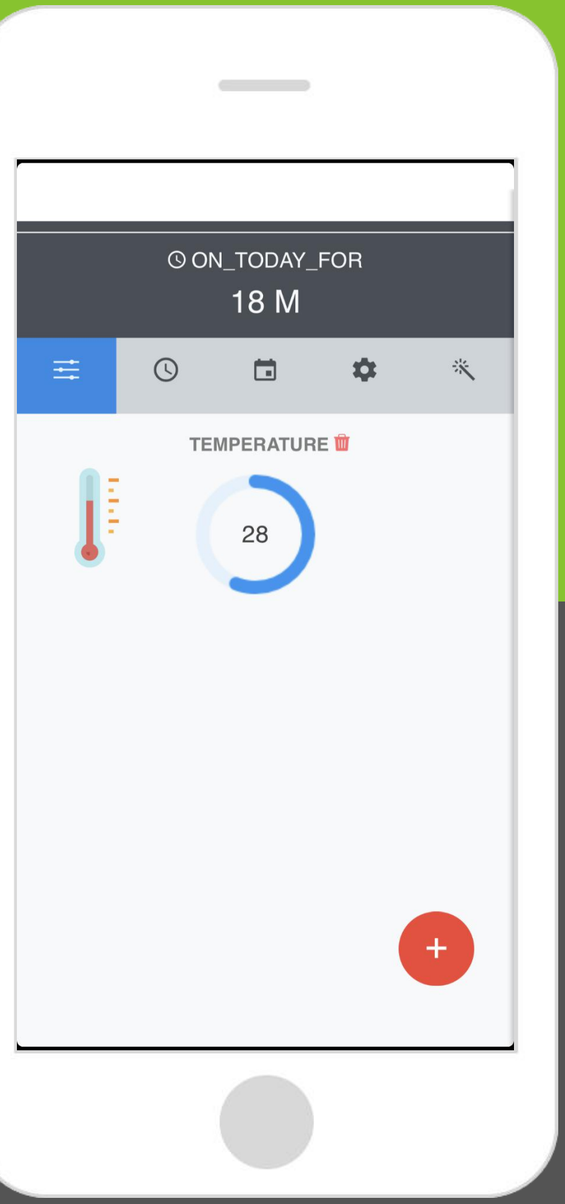
LABEL NAME: is the name that appears in the application.
PARAMETER NAME: is the name used in the code.

Fill in the blank the following parameters



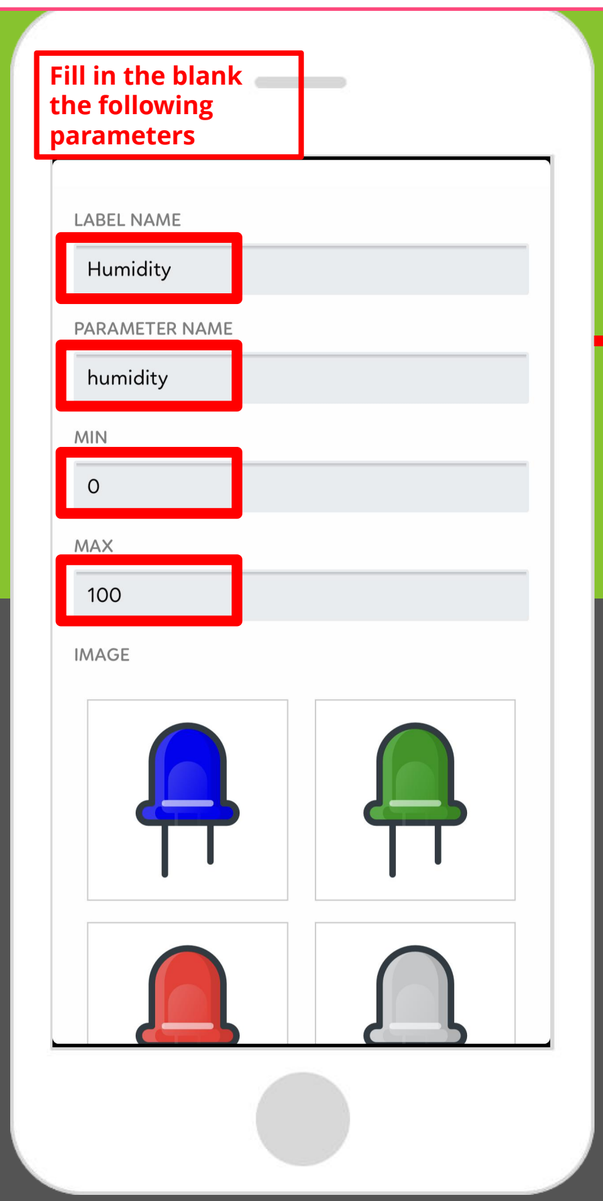


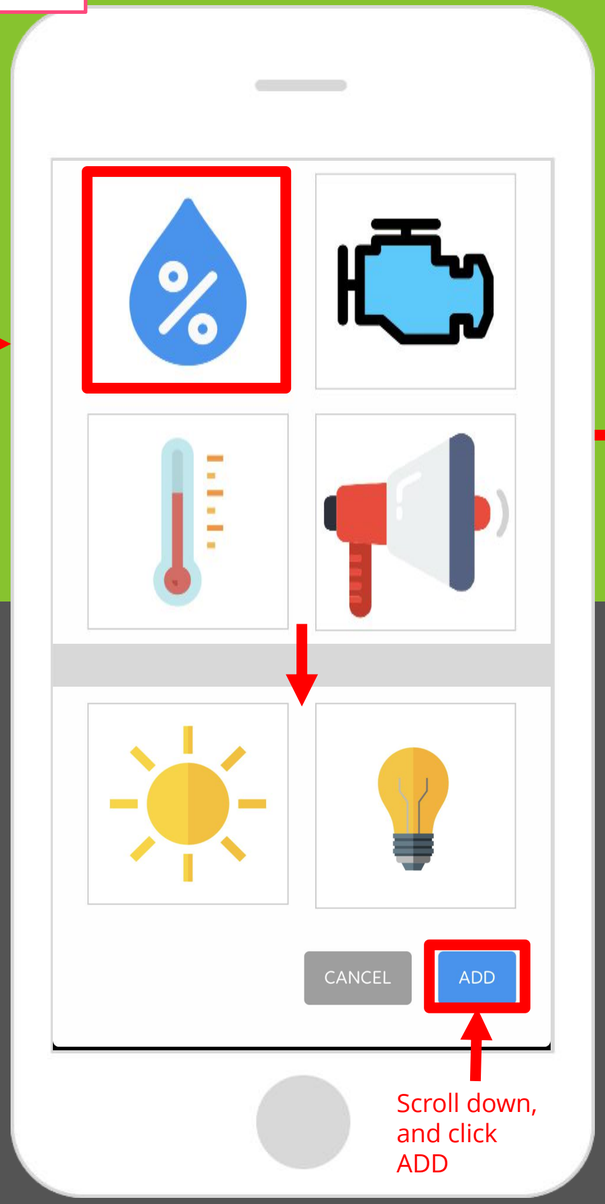
Scroll down, and click ADD



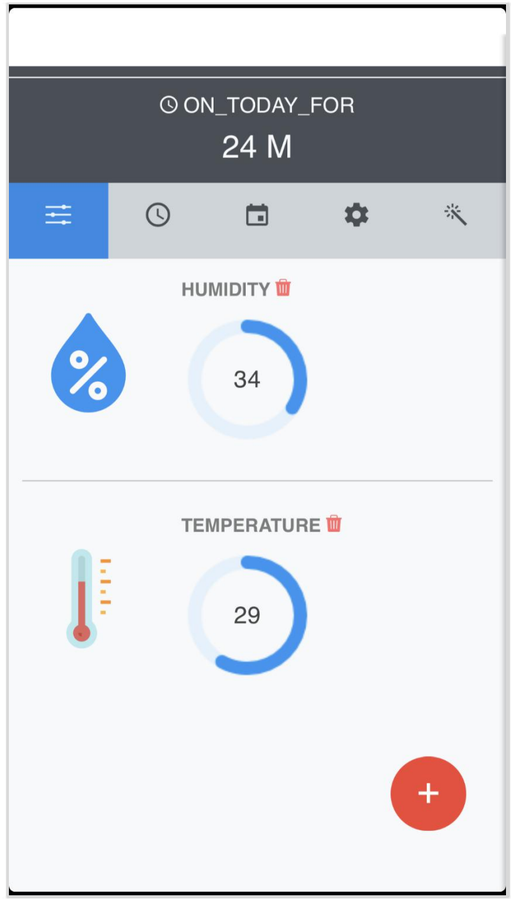
LABEL NAME: is the name that appears in the application.
PARAMETER NAME: is the name used in the code.

Fill in the blank the following parameters





Scroll down, and click ADD

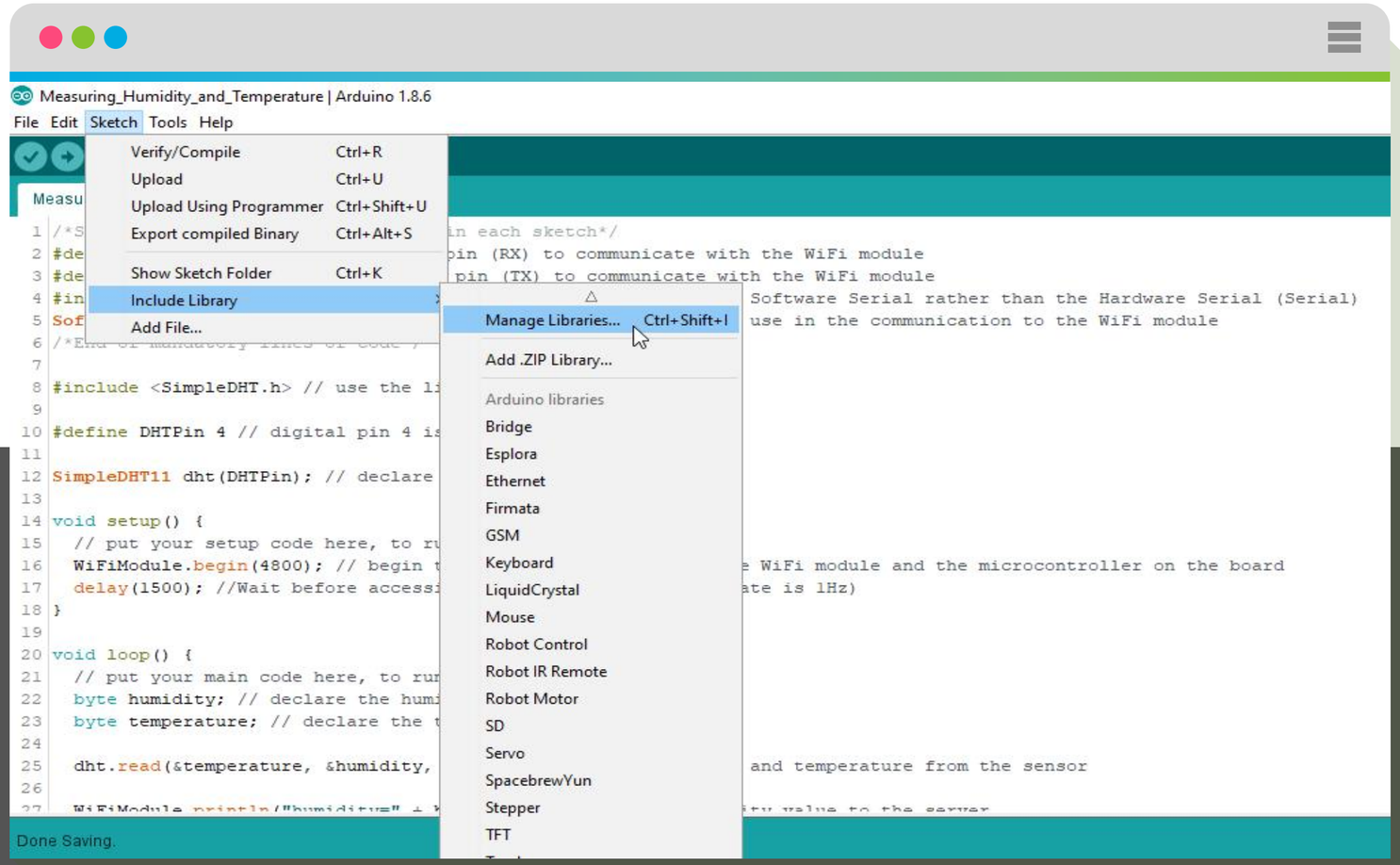


AFF IoT Board > Circuits > Measuring_Humidity_and_Temperature

Measuring_Humidity_and_Temperature

```
/*Start of mandatory lines of codes in each sketch*/
#define RX A0 // define the Receive pin (RX) to communicate with the WiFi module
#define TX A1 // define the Transmit pin (TX) to communicate with the WiFi module
#include <NeoSWSerial.h> // including the library to use the Software Serial rather than the Hardware Serial (Serial)
NeoSWSerial WiFiModule(RX, TX); //initialize the variable to use in communication with the WiFi module
/*End of mandatory lines of code*/

#include <SimpleDHT.h> // use the library functions in the code

#define DHTPin 4 // digital pin 4 is connected to the sensor

SimpleDHT11 dht(DHTPin); // declare the DHT11 instance

void setup() {
  // put your setup code here, to run once:
  WiFiModule.begin(19200); // begin the communication between the WiFi module and the microcontroller on the board
  delay(1500); //Wait before accessing Sensor (DHT11 sampling rate is 1Hz)
}

void loop() {
  // put your main code here, to run repeatedly:
  byte humidity; // declare the humidity variable
  byte temperature; // declare the temperature variable

  dht.read(&temperature, &humidity, NULL); // read both humudity and temperature from the sensor

  String data = "";
  data += "humidity=" + String(humidity); //example: humidity=70
  data += ","; // all parameters should be comma delimited
  data += "temperature=" + String(temperature); //example: temperature=27
  //data = "humidity=70,temperature=27"

  WiFiModule.println(data);// send the Humidity and Temperature values to the server

  delay(3000);//Wait 3 seconds before accessing sensor again.
}
```

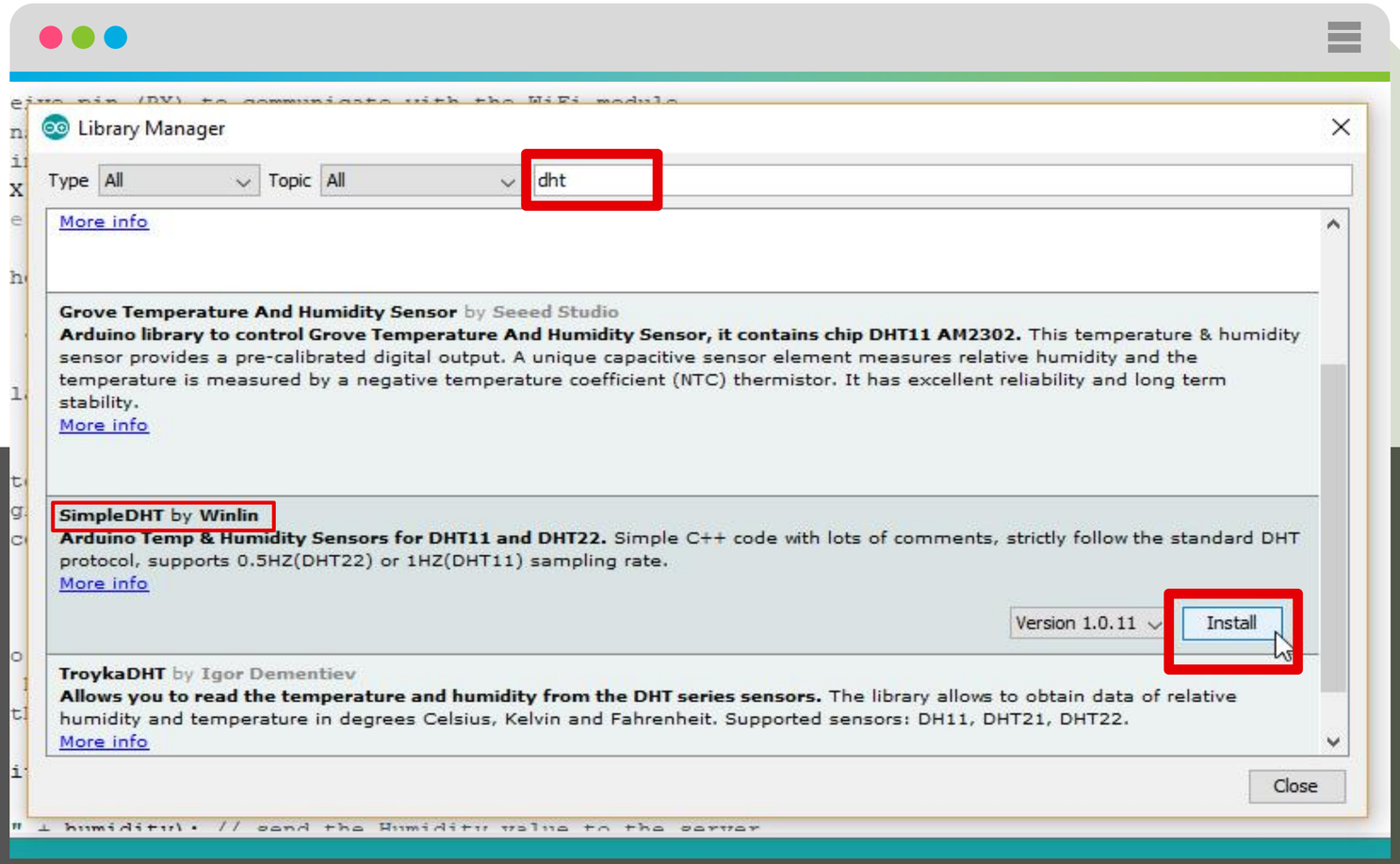
Open your sketch: Measuring_Humidity_and_Temperature



Before Compiling the code

Download the DHT library

Sketch > Include Library > Manage Libraries



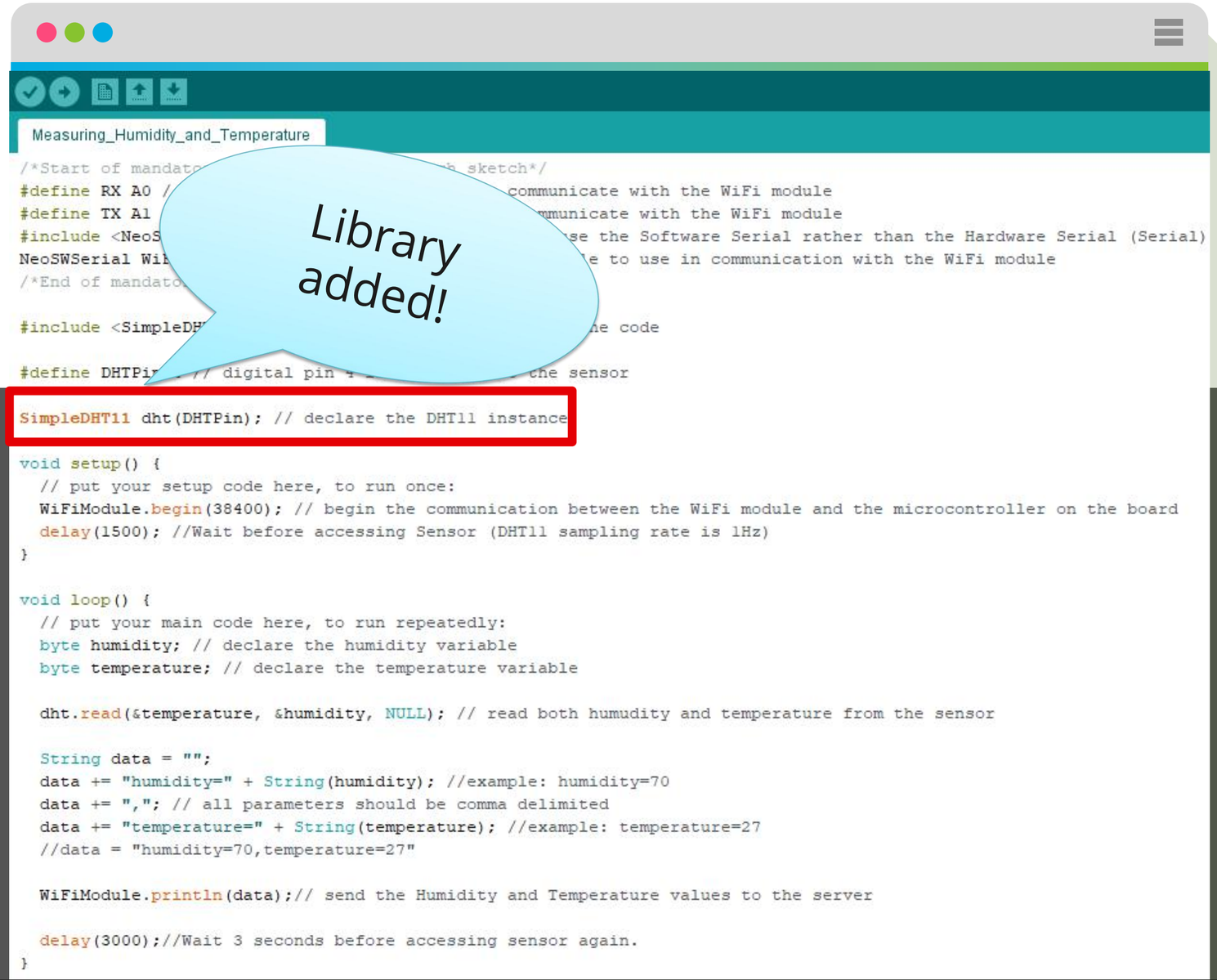
Before Compiling the code

Download the DHT library

Search for “dht”, select the newest version of “SimpleDHT library” and click “Install”.

Before Compiling the code

Download the DHT library



After Clicking “Close” button, we can see that the Arduino IDE recognizes the type of SimpleDHT and changes its color to orange.

Code to note..

```
dht.read(&temperature, &humidity, NULL):
```

This line of code calls the read function to get the humidity and temperature from the DHT11 sensor, and puts the values into the variables ‘temperature’ and ‘humidity’.

What you **should see**

You should see real humidity and temperature values. If it isn't working, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting tips below.

Troubleshooting

Nothing seems to happen
This program has no outward indication it is working. To see the results you must open the AFF IoT application.

Values are unchanging
Try rubbing the sensor with your fingers to heat it up or pressing a bag of ice close to it to cool it down.

Real World Application

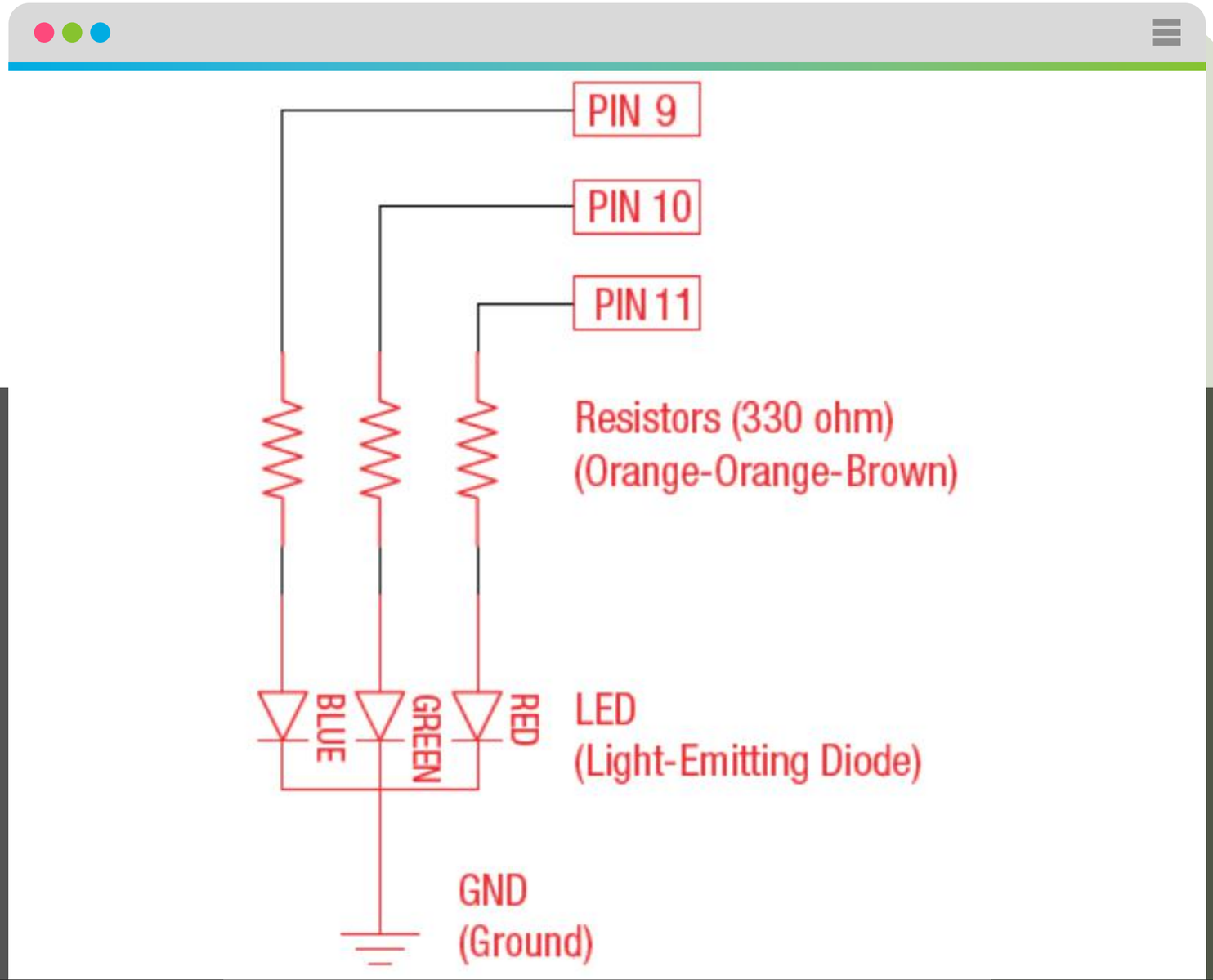
Measure the humidity and temperature at storage rooms
to prevent damage of stored items

9 Adjusting RGB LED color (via “Internet”)

RGB LED x1

330Ω Resistor x3

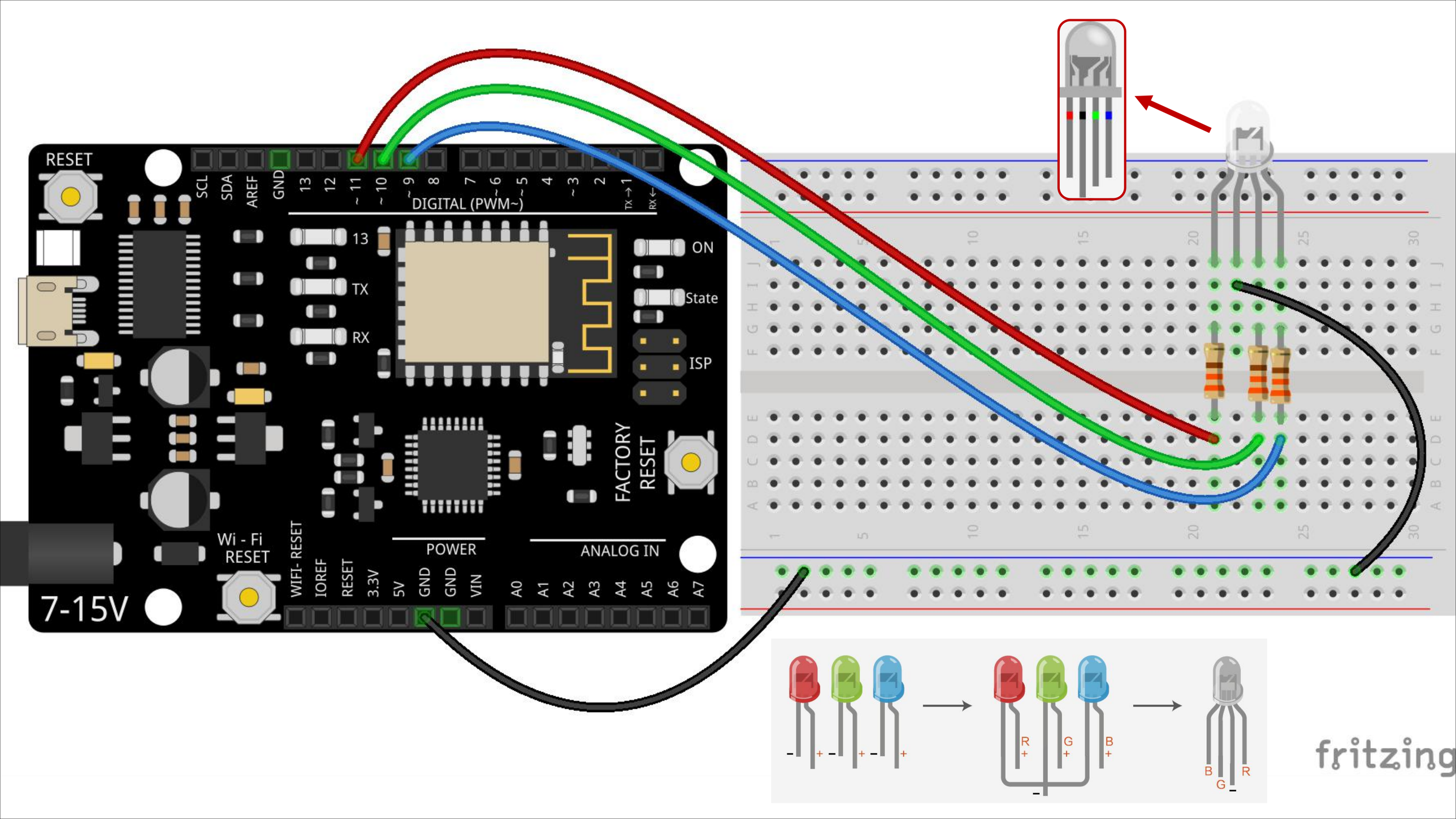
Jumper wires x5

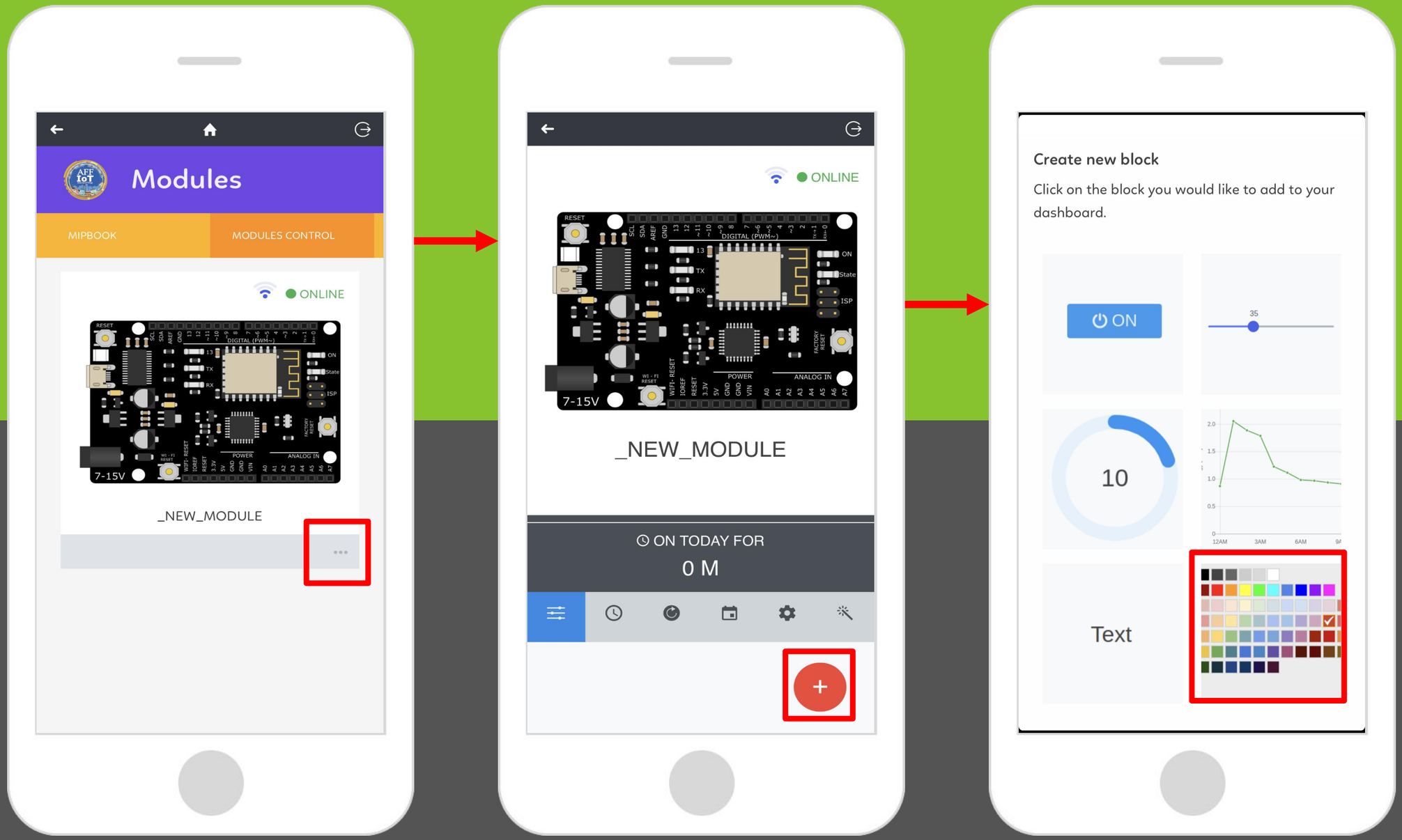


RGB LED

RGB, or red-green-blue, LEDs have three different color-emitting diodes that can be combined to create all sorts of colors. In this circuit, you'll learn how to use an RGB LED to create unique color combinations.

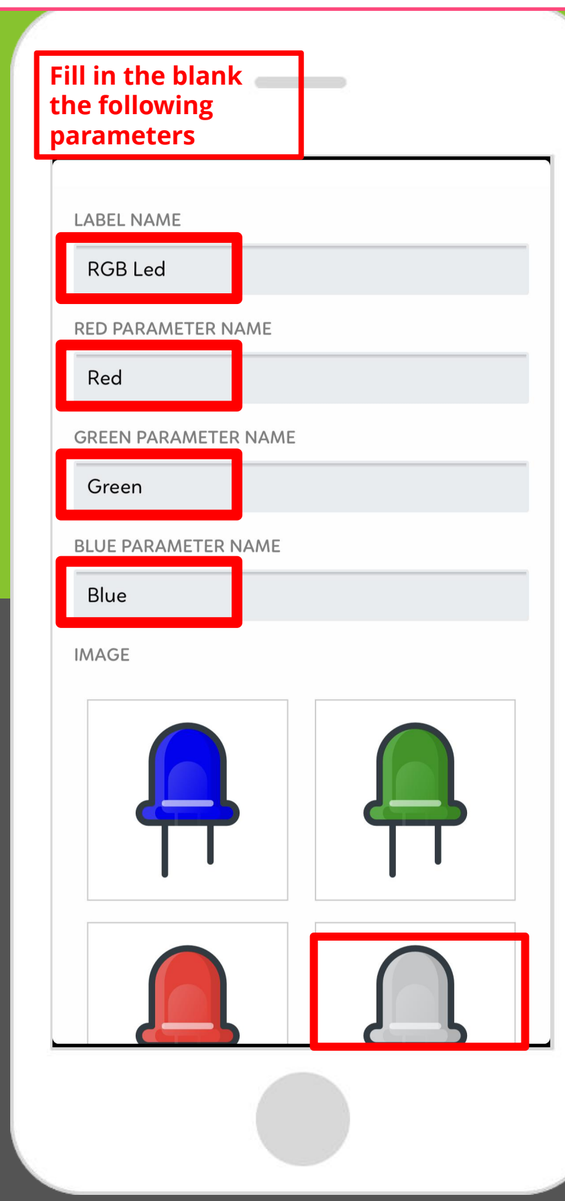
Depending on how bright each diode is, nearly any color is possible!

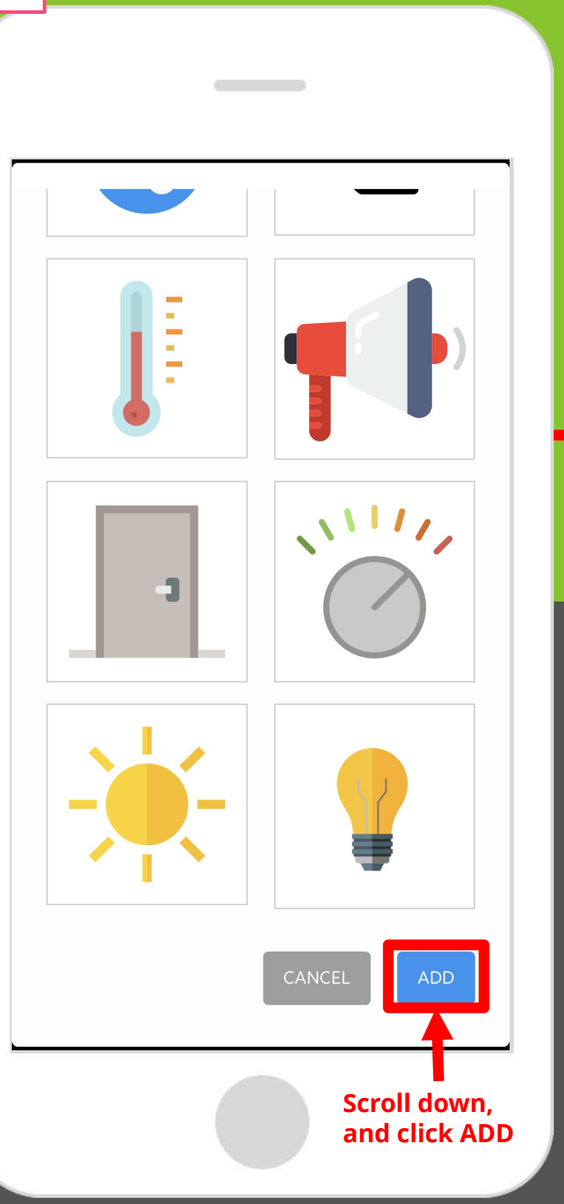

RESET
SCL
SDA
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TX → 1
RX ←
DIGITAL (PWM~)
13
TX
RX
ON
State
ISP
FACTORY RESET
Wi - Fi RESET
WIFI- RESET
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
A6
A7
7-15V
R
G
B
B
G
R
fritzing


Modules
MIPBOOK
MODULES CONTROL
ONLINE
_NEW_MODULE
ONLINE
_NEW_MODULE
ON TODAY FOR
0 M
Create new block
Click on the block you would like to add to your dashboard.
ON
35
10
Text

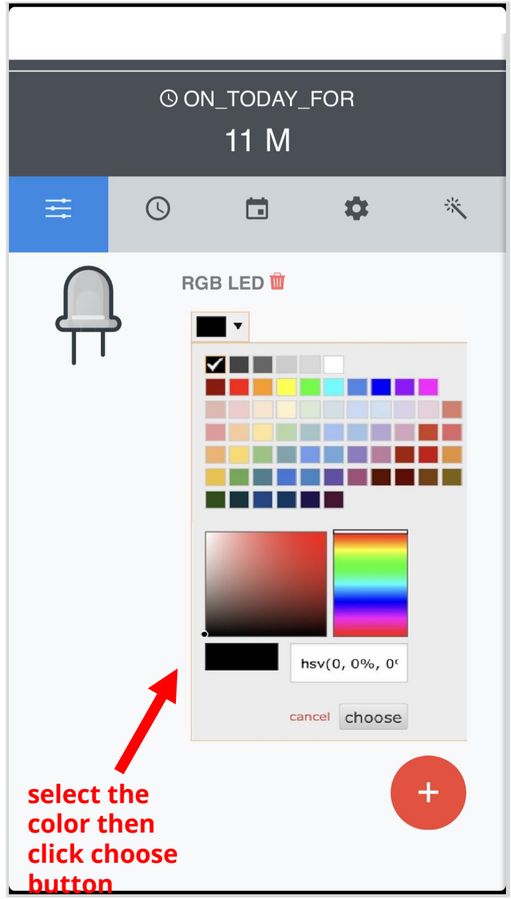
LABEL NAME: is the name that appears in the application.
PARAMETER NAME: is the name used in the code.

Fill in the blank the following parameters





Scroll down, and click ADD



select the color then click choose button

AFF IoT Board folder > Circuits > Adjusting_RGB_LED_Color

Adjusting_RGB_LED_Color

```
/*Start of mandatory lines of codes in each sketch*/
#define RX A0 // define the Receive pin (RX) to communicate with the WiFi module
#define TX A1 // define the Transmit pin (TX) to communicate with the WiFi module
#include <NeoSWSerial.h> // including the library to use the Software Serial rather than the Hardware Serial (Serial)
NeoSWSerial WiFiModule(RX, TX); //initialize the variable to use in communication with the WiFi module
/*End of mandatory lines of code*/

#define RedLedPin    11
#define GreenLedPin  10
#define BlueLedPin   9

String RedStringValue = "";
String GreenStringValue = "";
String BlueStringValue = "";

int RedValue = 0;
int GreenValue = 0;
int BlueValue = 0;

void setup() {
  // put your setup code here, to run once:
  WiFiModule.begin(19200); // begin the communication between the WiFi module and the microcontroller on the board
  pinMode(RedLedPin, OUTPUT); // configure the pin connected to the Red Led to be an output
  pinMode(GreenLedPin, OUTPUT); // configure the pin connected to the Green Led to be an output
  pinMode(BlueLedPin, OUTPUT); // configure the pin connected to the Blue Led to be an output
  Serial.begin(9600);
}
```

1

Open your sketch:
Adjusting_RGB_LED_Color

AFF IoT Board folder > Circuits > Adjusting_RGB_LED_Color

```
void loop() {
  // put your main code here, to run repeatedly:
  if (WiFiModule.available() > 0) // if the WiFi module receive data from the server
  {
    String Command = WiFiModule.readStringUntil('\n'); // read the command sent from the WiFi module to the microcontroller
    Serial.println(Command); // print the command on the Serial monitor for debugging
    if (Command.indexOf("Red=") >= 0) // if the received command contains "Red=" so we should change the Red intensity
    {
      String RedString = Command.substring(0, Command.indexOf(',')+1); // extract the command for "Red" parameter
      RedStringValue = Command.substring(Command.indexOf('=') + 1); // extract the Red intensity string value
      RedValue = RedStringValue.toInt(); // transform the string to an integer value
      analogWrite(RedLedPin, RedValue); // generate the PWM duty cycle according to the needed intensity
      Command.replace(RedString, ""); // remove the executed command
      Serial.println(Command); // print the command on the Serial monitor for debugging
    }
    if (Command.indexOf("Green=") >= 0) // if the received command contains "Green=" so we should change the Green intensity
    {
      String GreenString = Command.substring(0, Command.indexOf(',')+1); // extract the command for "Green" parameter
      GreenStringValue = Command.substring(Command.indexOf('=') + 1); // extract the Green intensity string value
      GreenValue = GreenStringValue.toInt(); // transform the string to an integer value
      analogWrite(GreenLedPin, GreenValue); // generate the PWM duty cycle according to the needed intensity
      Command.replace(GreenString, ""); // remove the executed command
      Serial.println(Command); // print the command on the Serial monitor for debugging
    }
    if (Command.indexOf("Blue=") >= 0) // if the received command contains "Blue=" so we should change the Blue intensity
    {
      String BlueString = Command.substring(0, Command.indexOf(',')+1); // extract the command for "Blue" parameter
      BlueStringValue = Command.substring(Command.indexOf('=') + 1); // extract the Blue intensity string value
      BlueValue = BlueStringValue.toInt(); // transform the string to an integer value
      analogWrite(BlueLedPin, BlueValue); // generate the PWM duty cycle according to the needed intensity
      Command.replace(BlueString, ""); // remove the executed command
      Serial.println(Command); // print the command on the Serial monitor for debugging
    }
  }
}
//(for more info about indexOf function see https://www.arduino.cc/reference/en/language/variables/data-types/string/functions/indexof/)
//(for more info about substring function see https://www.arduino.cc/en/Tutorial/StringSubstring)
```

2

Open your sketch:

Adjusting_RGB_LED_Color

Code to note..

`RedValue = RedStringValue.toInt()`: the function `toInt()` converts a valid String to an integer. The input String should start with an integer number. If the String contains non-integer numbers, the function will stop performing the conversion.

`Command.substring(index)`: with only one parameter, it looks for a given substring from the assigned index to the end of the string (`Command` in this example).

What you **should see**

You should see your LED turn on, but this time in new, crazy colors! If it isn't, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting tips below.

Troubleshooting

Seeing Red
The red diode within the RGB LED may be a bit brighter than the other two. To make your colors more balanced, use a higher Ohm resistor for the red one. Or adjust in code.

```
analogWrite(RedLedPin, RedValue);
```

To `analogWrite(RedLedPin, RedValue/3);`

LED Remains Dark or Shows Incorrect Color
With the four pins of the LED so close together, it's sometimes easy to misplace one. Double check each pin where is inserted.

Real World Application

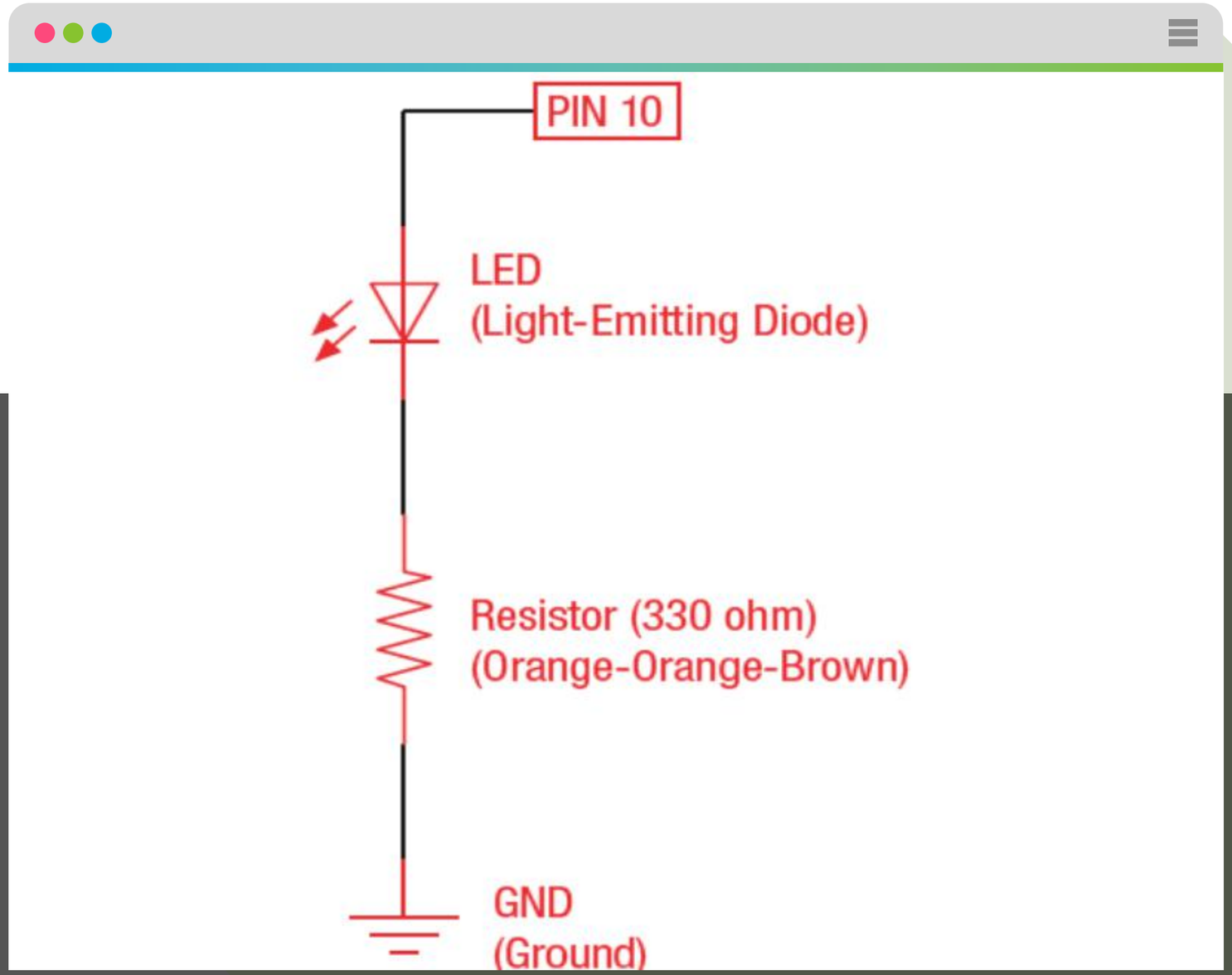
Many electronic devices such as videogame consoles use RGB LEDs to show different colors within the same light source.

10 Scheduling controls

White LED x1

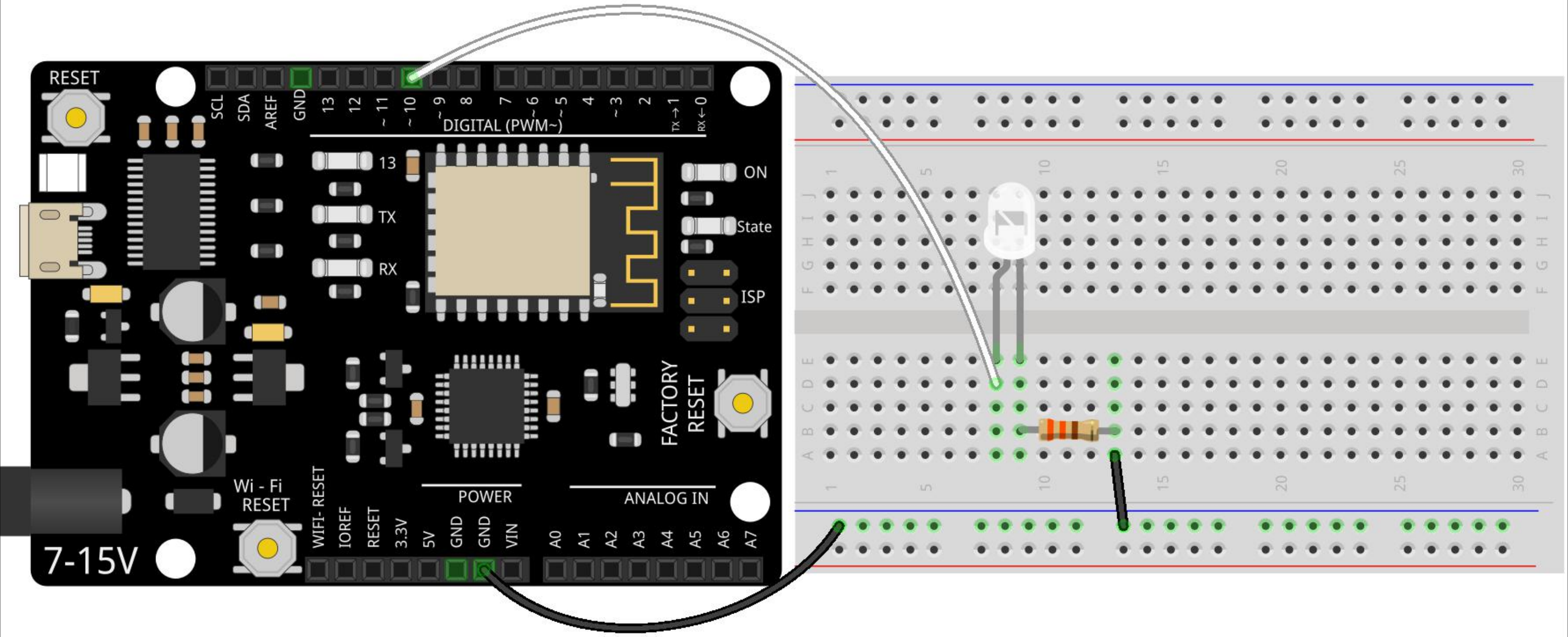
330Ω Resistor x1

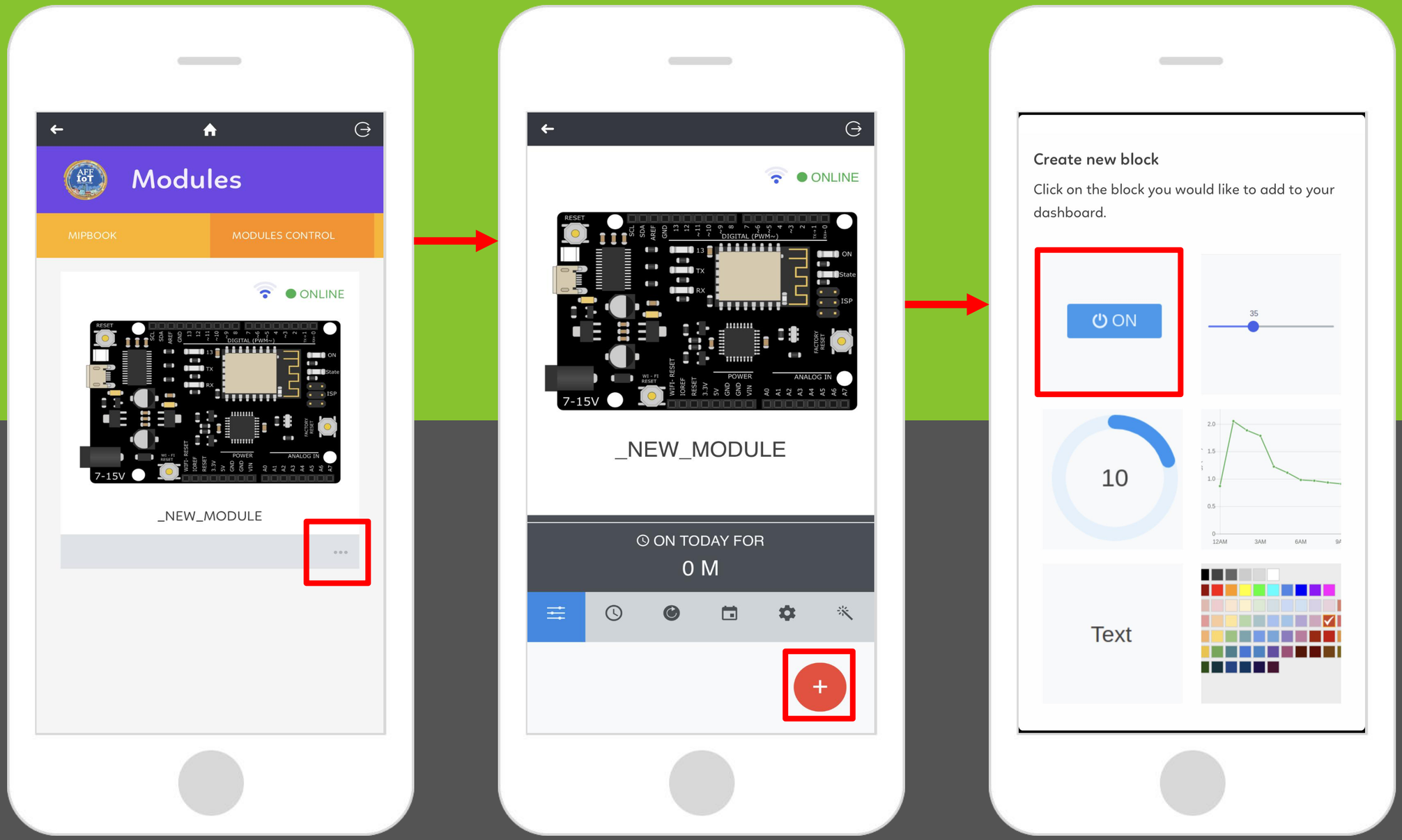
Jumper wires x3



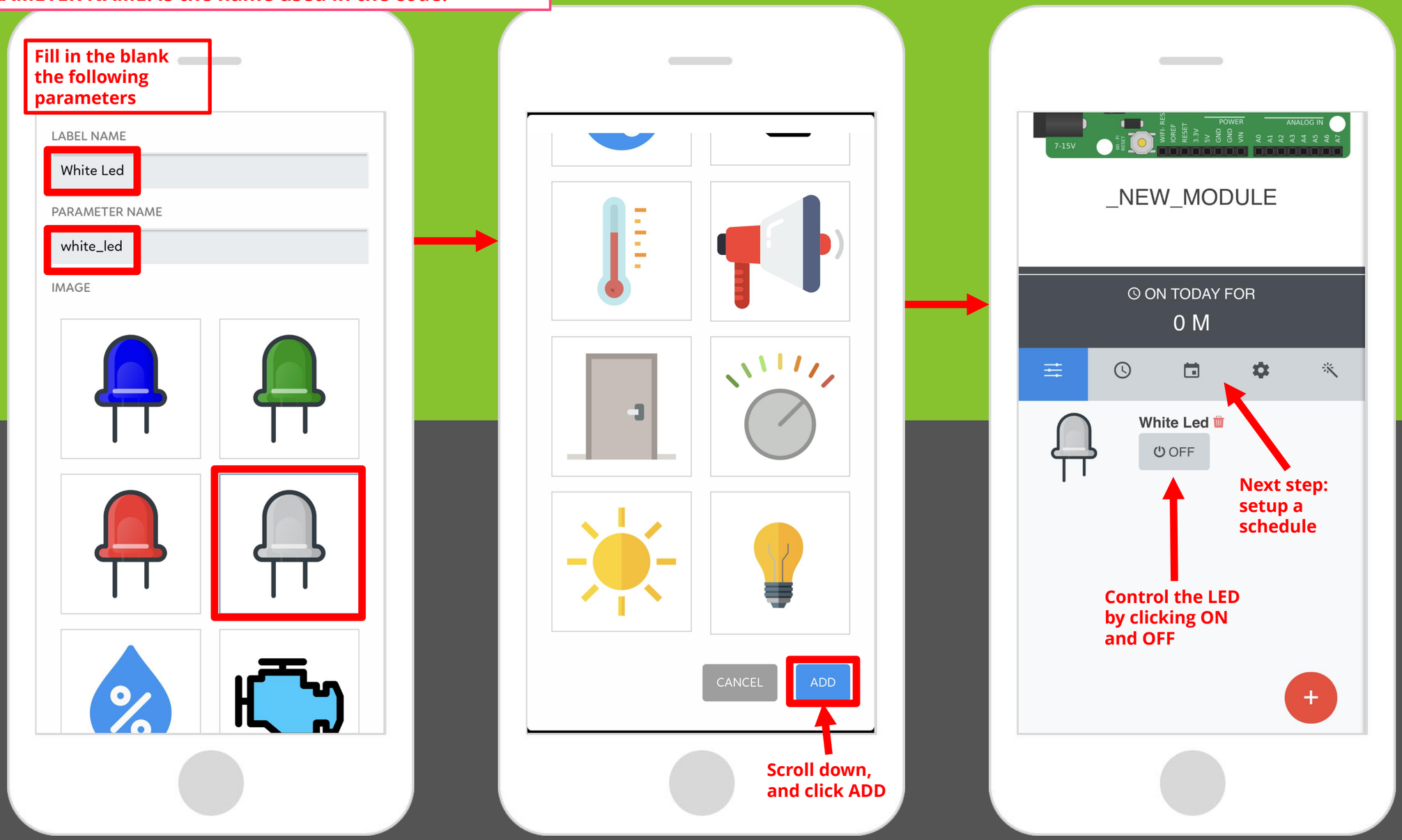
Turn ON/OFF an LED

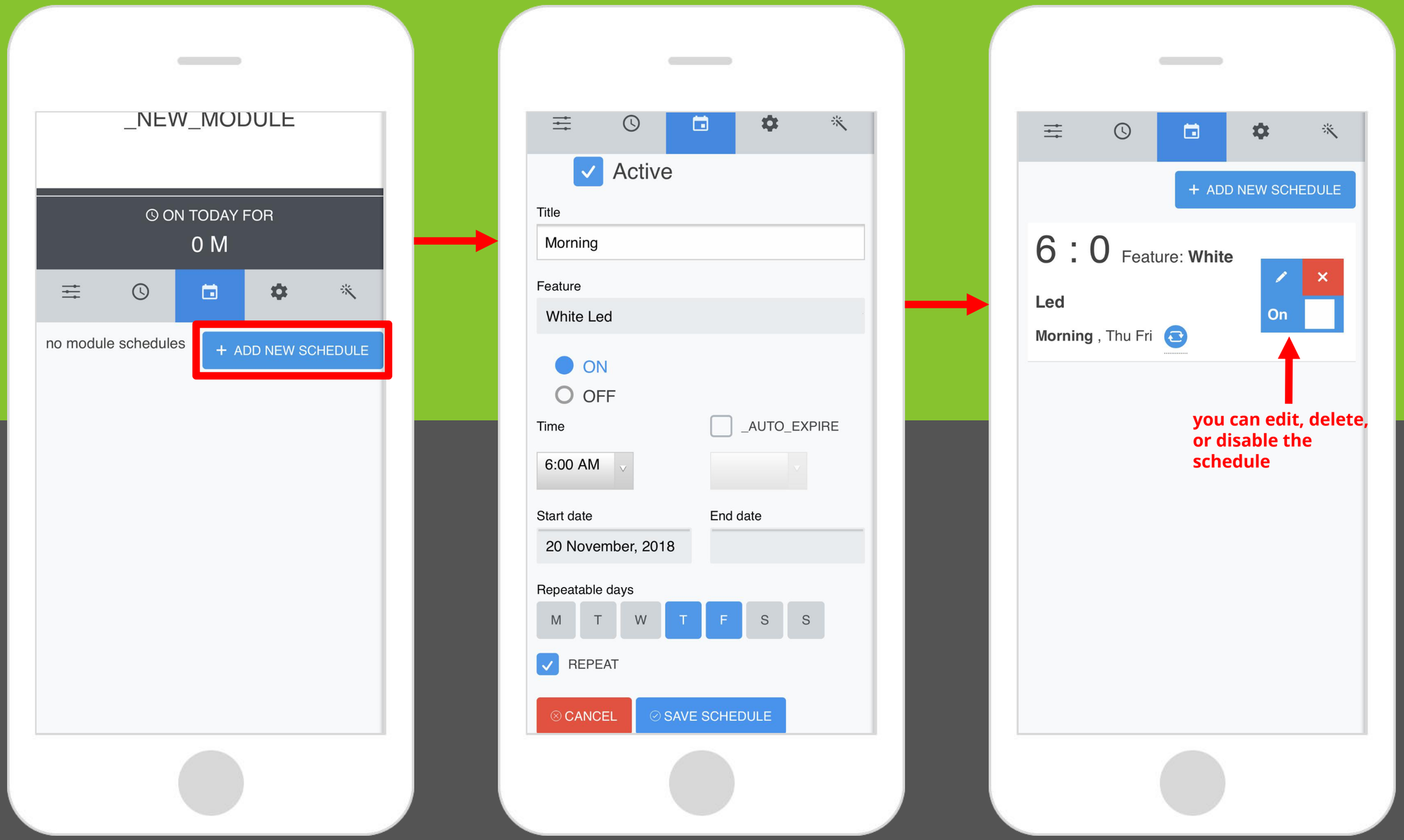
LEDs (light-emitting diodes) are small, but powerful light source that are used in many different applications. First we will work on controlling an LED. It is simple and basic function, yet, it is the basis for further complex functions.

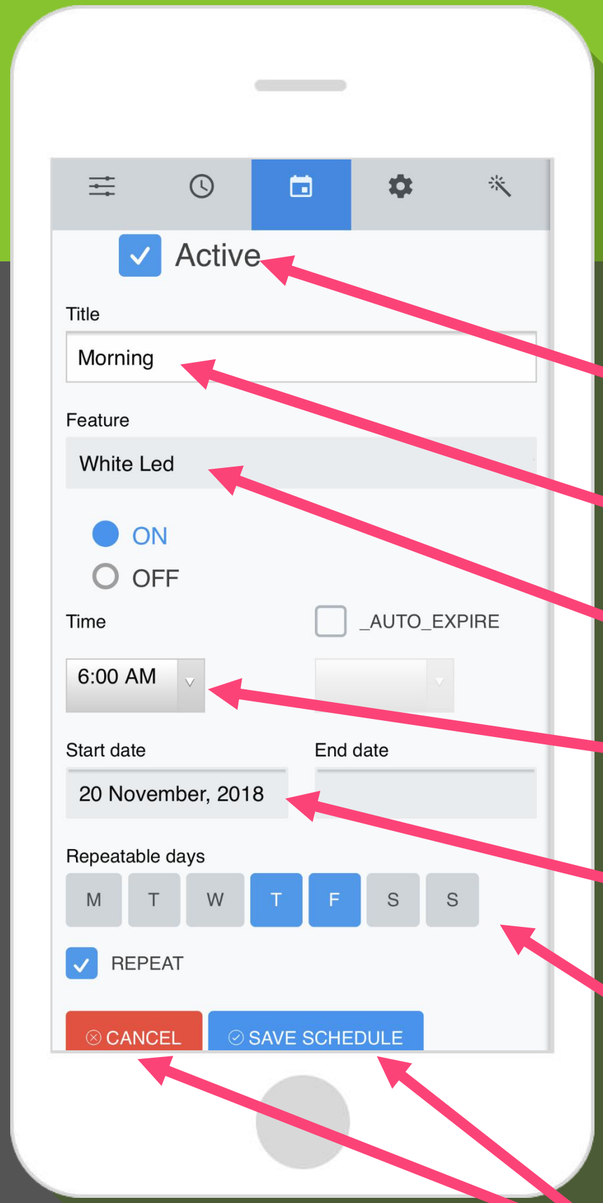

RESET
SCL
SDA
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TX → 1
RX ← 0
DIGITAL (PWM~)
13
TX
RX
ON
State
ISP
FACTORY
RESET
Wi - Fi
RESET
WIFI- RESET
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
A6
A7
7-15V


Modules
MIPBOOK
MODULES CONTROL
ONLINE
_NEW_MODULE
ONLINE
_NEW_MODULE
ON TODAY FOR
0 M
Create new block
Click on the block you would like to add to your dashboard.
ON
35
10
Text

LABEL NAME: is the name that appears in the application.
PARAMETER NAME: is the name used in the code.




_NEW_MODULE
ON TODAY FOR
0 M
no module schedules
+ ADD NEW SCHEDULE
Active
Title
Morning
Feature
White Led
ON
OFF
Time
_AUTO_EXPIRE
6:00 AM
Start date
End date
20 November, 2018
Repeatable days
M T W T F S S
REPEAT
CANCEL
SAVE SCHEDULE
+ ADD NEW SCHEDULE
6 : 0 Feature: White
Led
Morning , Thu Fri
On
you can edit, delete, or disable the schedule



Explanation

- Active: select the state of the schedule (Active or not).

- Title: is the title of the created schedule.

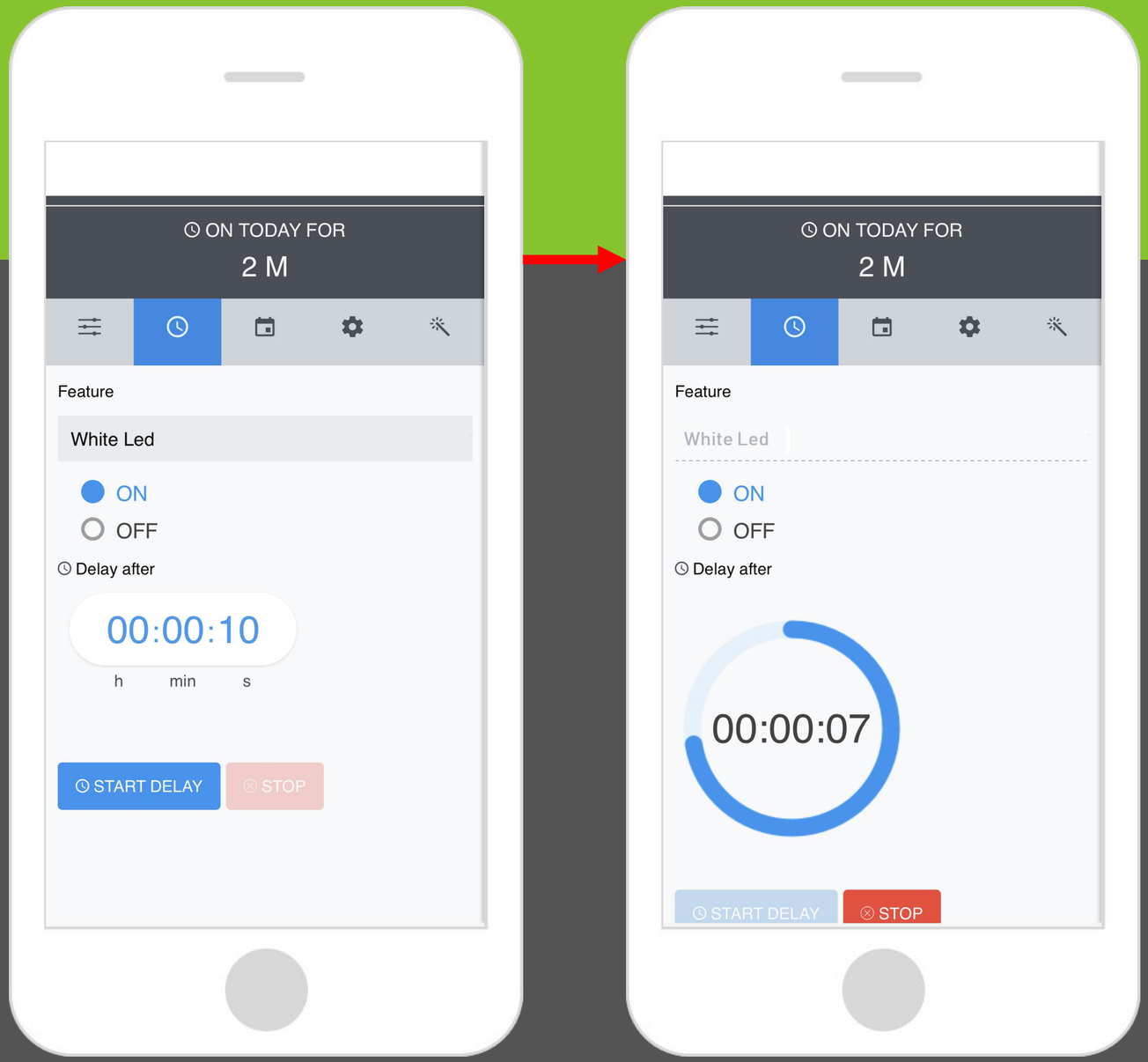
- Feature: is the control to be scheduled (Set ON or OFF).

- Time: is the time when the action will be executed.

- Start date: is the date when the schedule will be activated.

- Repeatable days: if the REPEAT checkbox is checked, you can choose what are days the schedule is active.

- Save Schedule or Cancel it.



Timer

The timer option consists of choosing a control and setting a delay time. After this delay the control will be executed.

You can also stop the timer by clicking on STOP button, and the delayed action will be discarded.

AFF IoT Board folder > Circuits > Examples > Scheduling_Controls

Scheduling_Controls

```
/*Start of mandatory lines of codes in each sketch*/
#define RX A0 // define the Receive pin (RX) to communicate with the WiFi module
#define TX A1 // define the Transmit pin (TX) to communicate with the WiFi module
#include <NeoSWSerial.h> // including the library to use the Software Serial rather than the Hardware Serial (Serial)
NeoSWSerial WiFiModule(RX, TX); //initialize the variable to use in communication with the WiFi module
/*End of mandatory lines of code*/

#define WhiteLedPin   10

void setup() {
  // put your setup code here, to run once:
  WiFiModule.begin(19200); // begin the communication between the WiFi module and the microcontroller on the board
  pinMode(WhiteLedPin, OUTPUT); // configure the pin connected to the White Led to be an output
}

void loop() {
  // put your main code here, to run repeatedly:
  if (WiFiModule.available() > 0) // if the WiFi module receive data from the server
  {
    String Command = WiFiModule.readStringUntil('\n'); // read the command sent from the WiFi module to the microcontroller

    if (Command.indexOf("white_led=1") >= 0) // if the received command contains "white_led=1" turn on the LED
    {
      digitalWrite(WhiteLedPin, HIGH); // turn on the LED
    }
    if (Command.indexOf("white_led=0") >= 0) // if the received command contains "white_led=0" turn it off
    {
      digitalWrite(WhiteLedPin, LOW); // turn off the LED
    }
  }
}
//(for more info about indexOf function see https://www.arduino.cc/reference/en/language/variables/data-types/string/functions/indexof/)
```

Open your sketch:
Scheduling_Controls

This sketch will work same as the Controlling LED example (project 5) but with change in LED name.

What you **should see**

You should see the LED turns ON after 10 seconds. If you select the OFF option, you should see the LED turns OFF after the delay time.

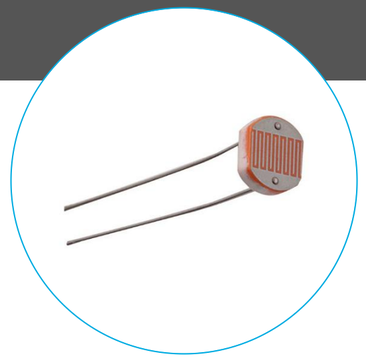
Troubleshooting

LED Not Lighting Up?
LEDs will only work in one direction. Try taking it out and twisting it 180 degrees (don't worry, installing it the opposite way doesn't damage the LED).

Real World Application

Most IoT devices are built with integrated timers

11 Simulating automated lighting system



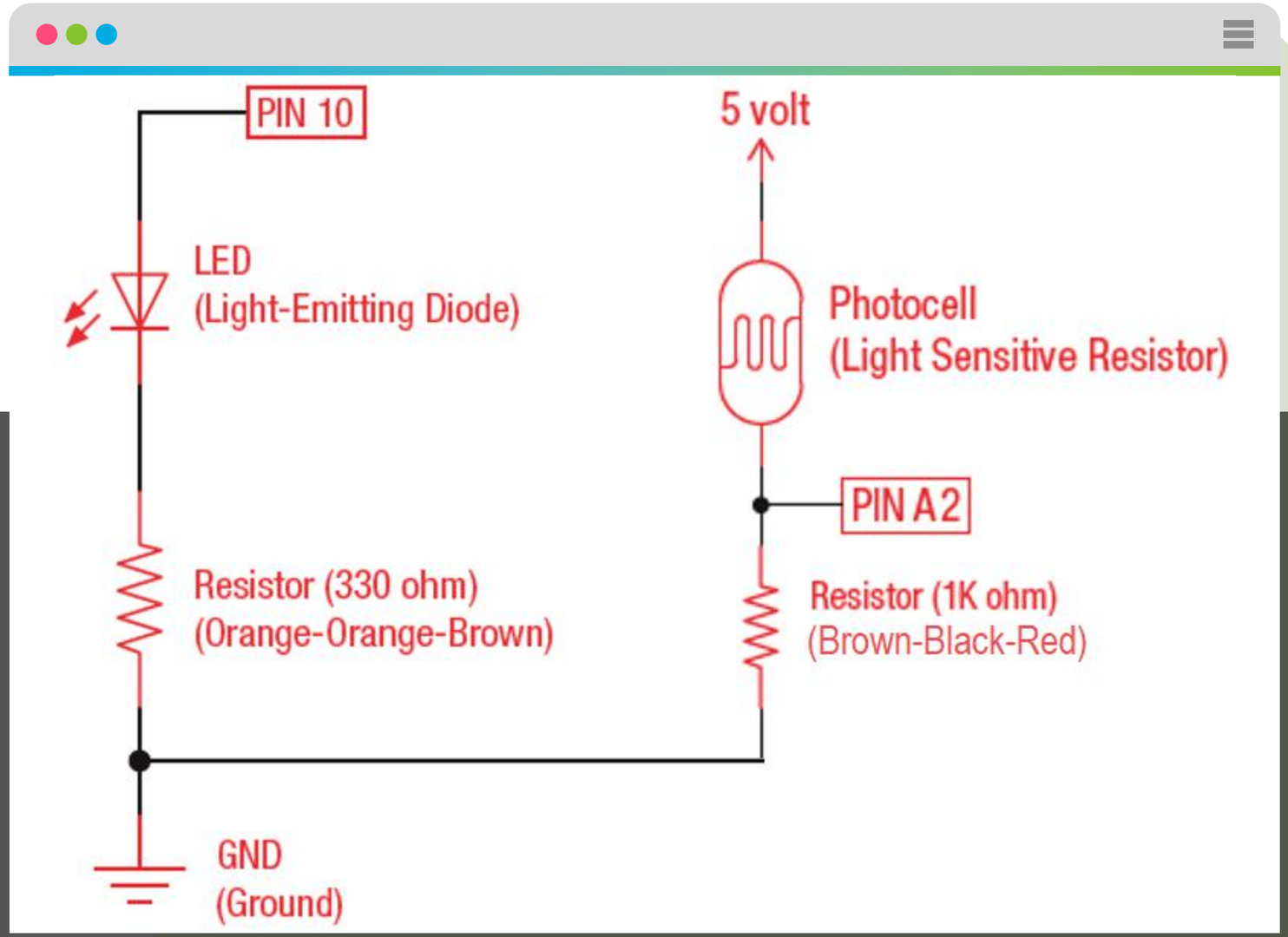
Photocell x1

Yellow LED x1

1KΩ Resistor x1

330Ω Resistor x1

Jumper wires x5

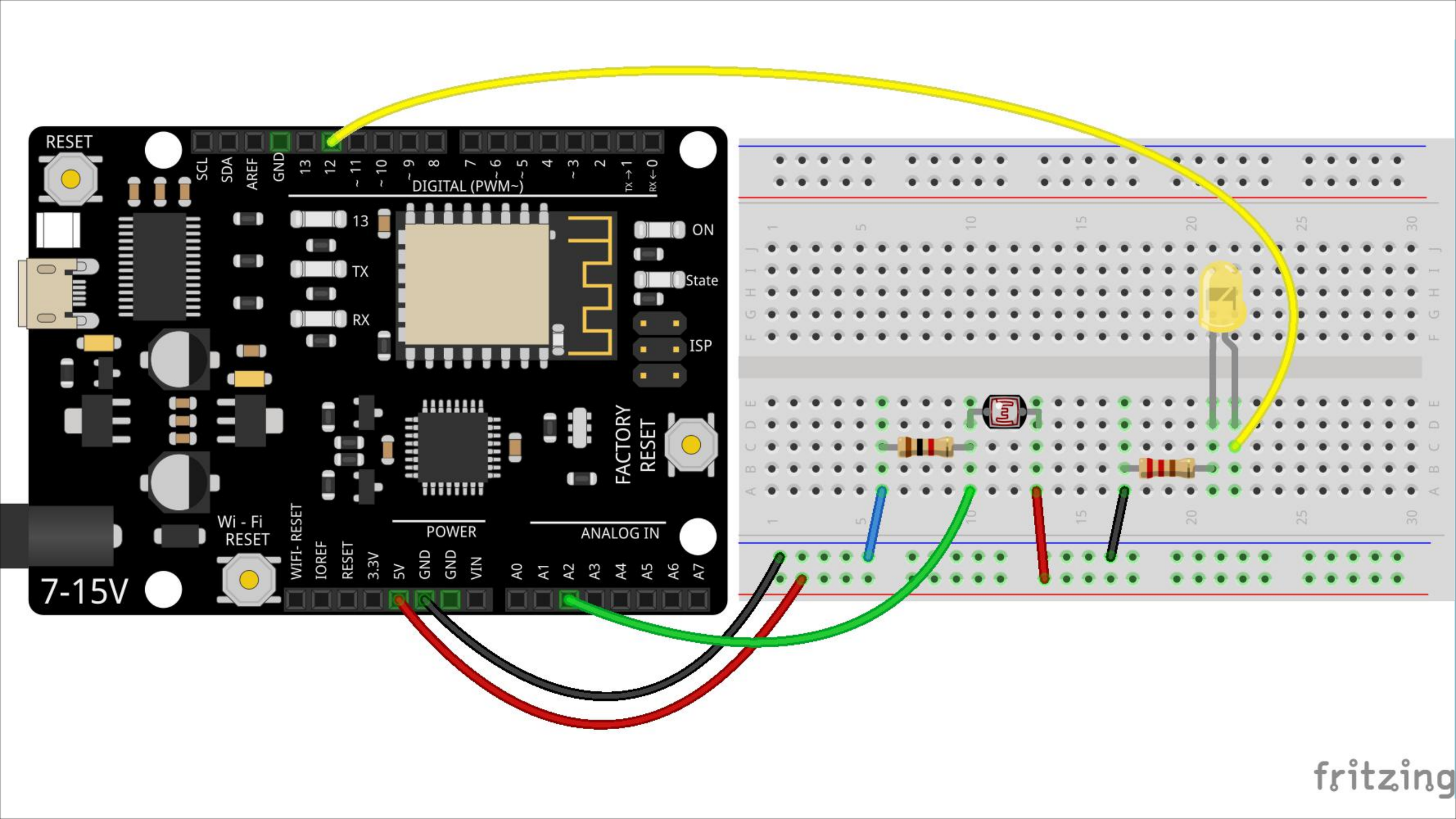


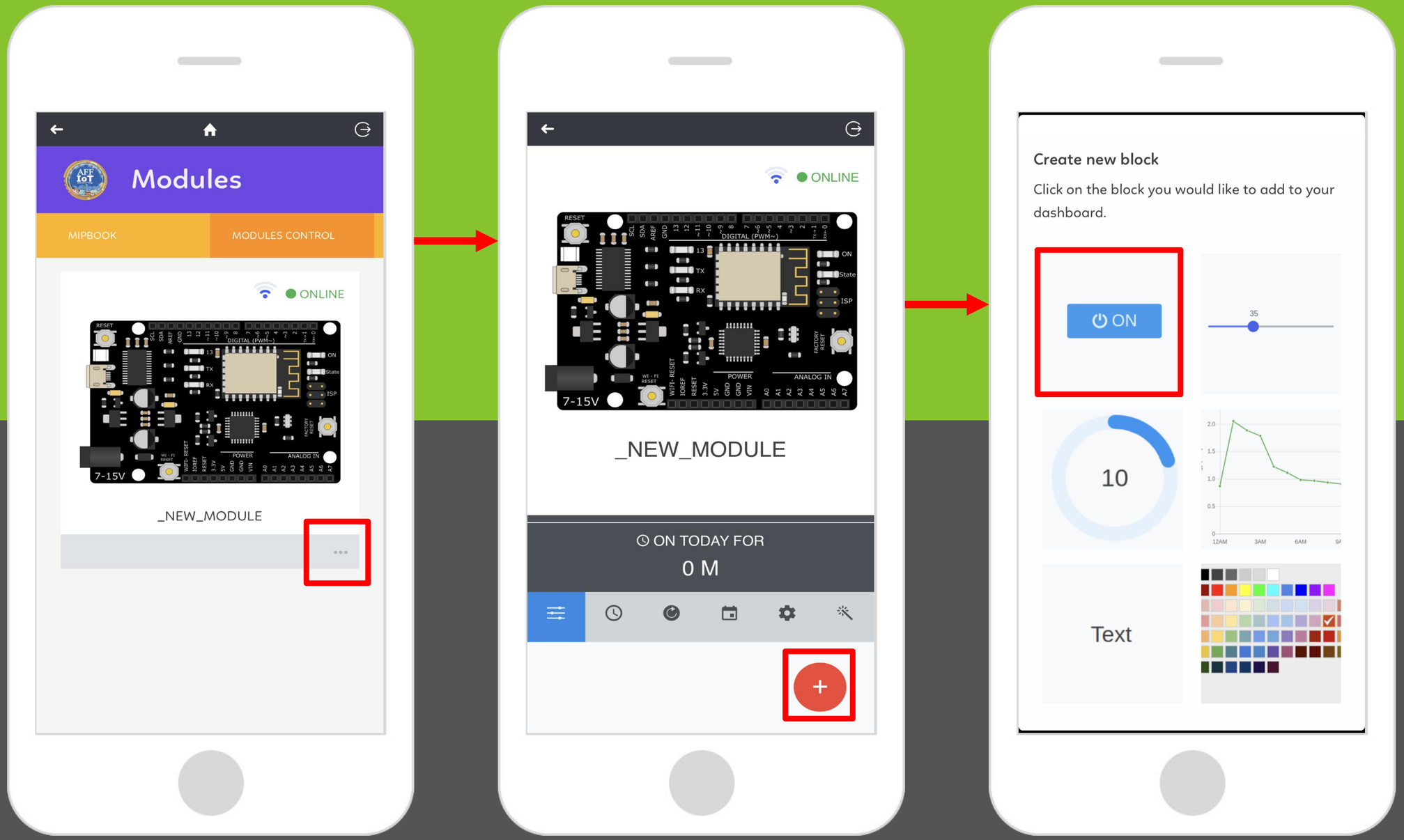
Simulating real system

A simulation is an imitation of a real-world operation.

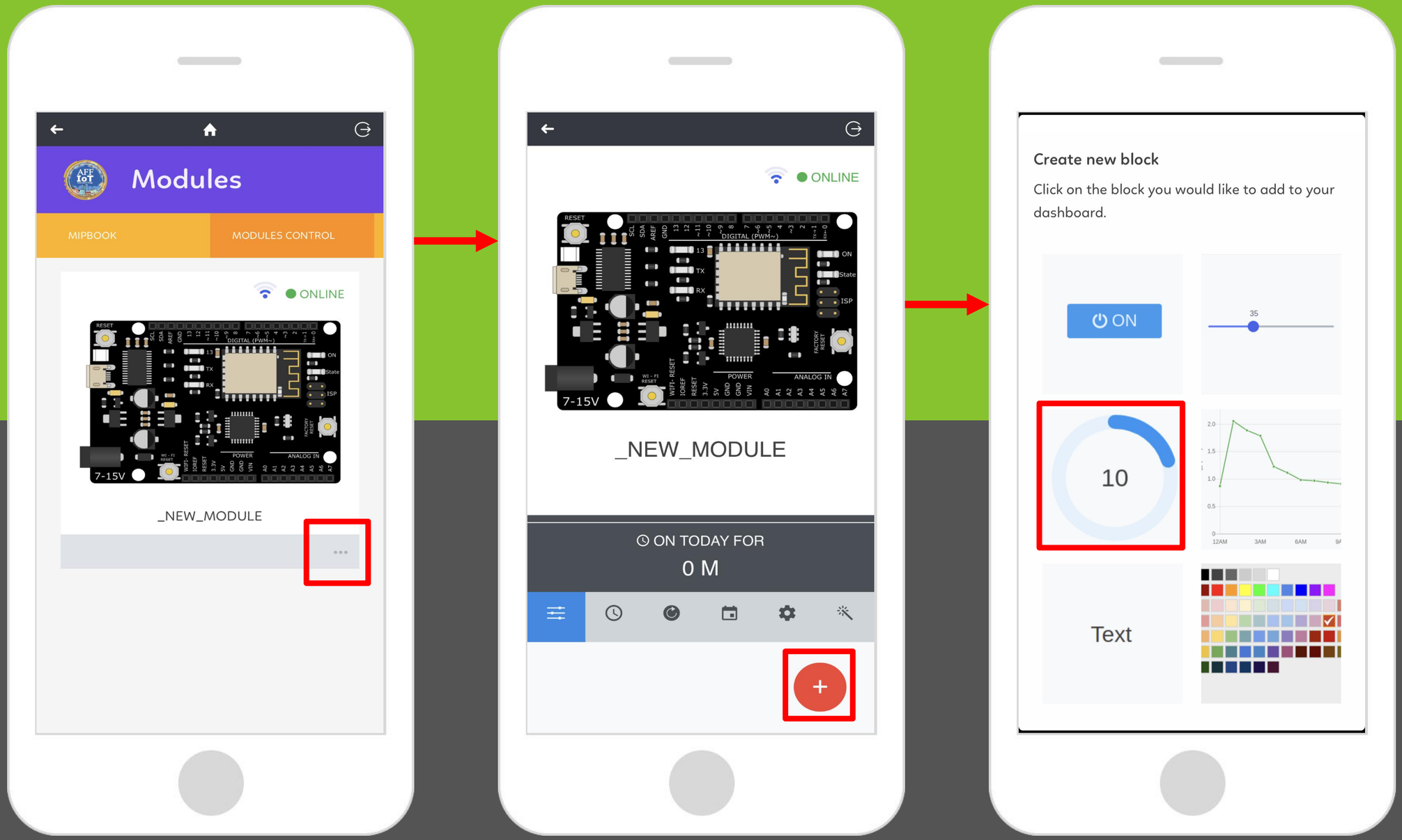
So in real-world, instead of LED, a light bulb is used.

LED is used instead of light bulb for safety reasons while applying several trials.

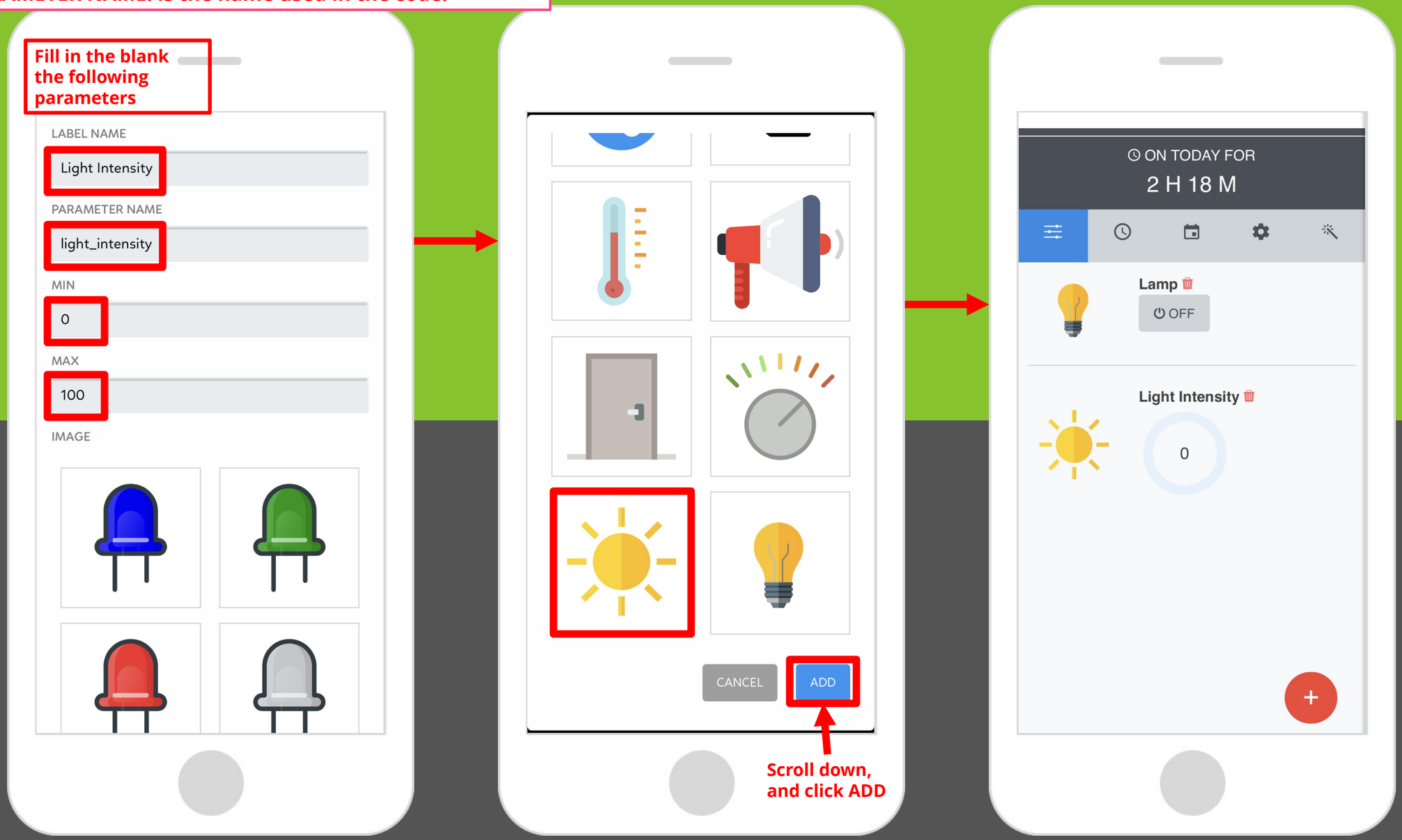

RESET
SCL
SDA
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TX → 1
RX ← 0
DIGITAL (PWM~)
13
TX
RX
ON
State
ISP
FACTORY RESET
Wi - Fi RESET
WIFI- RESET
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
A6
A7
ANALOG IN
7-15V
fritzing

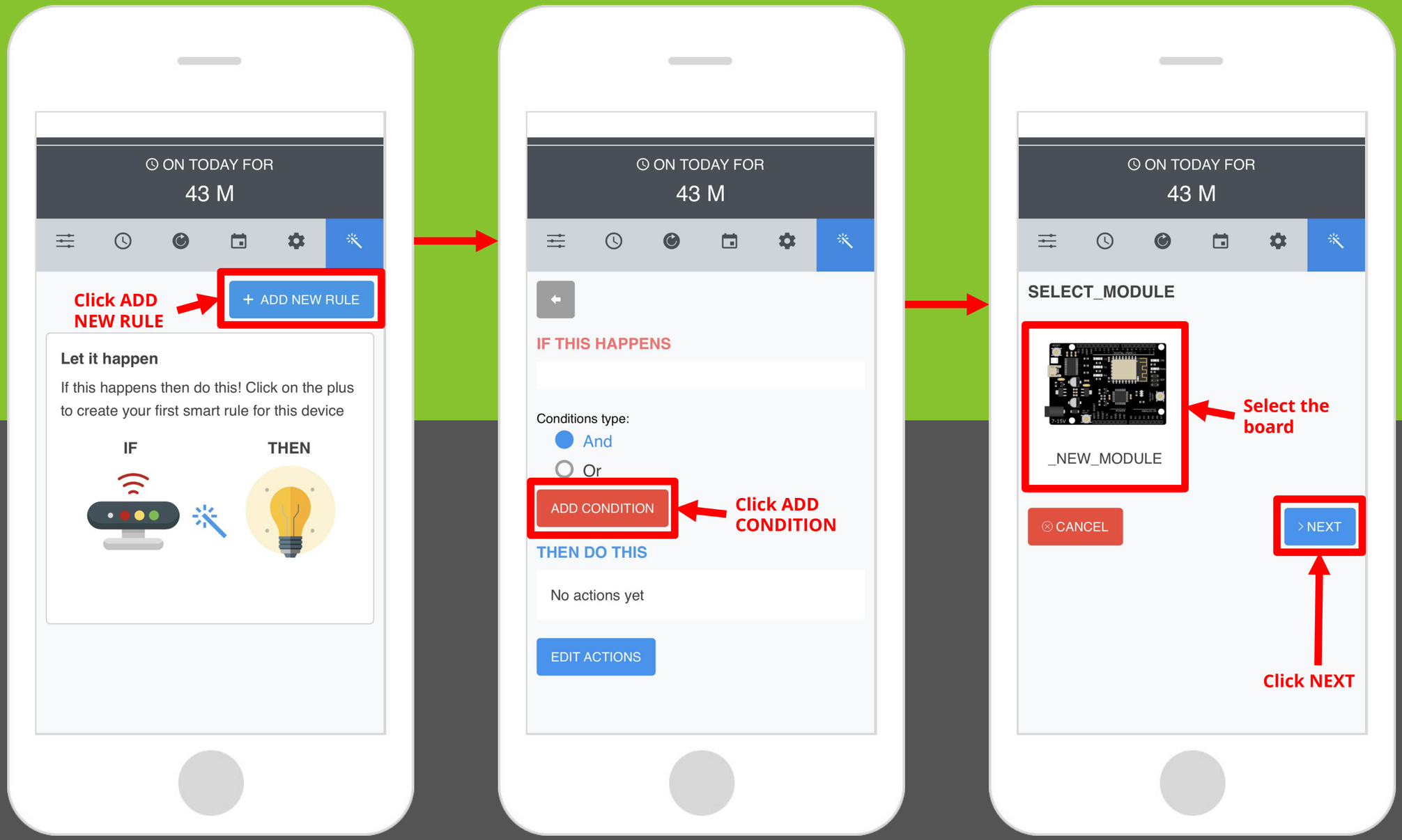

Modules
MIPBOOK
MODULES CONTROL
ONLINE
_NEW_MODULE
ONLINE
_NEW_MODULE
ON TODAY FOR
0 M
Create new block
Click on the block you would like to add to your dashboard.
ON
35
10
Text

LABEL NAME: is the name that appears in the application.
PARAMETER NAME: is the name used in the code.


Modules
MIPBOOK
MODULES CONTROL
ONLINE
_NEW_MODULE
ONLINE
_NEW_MODULE
ON TODAY FOR
0 M
Create new block
Click on the block you would like to add to your dashboard.
ON
35
10
Text

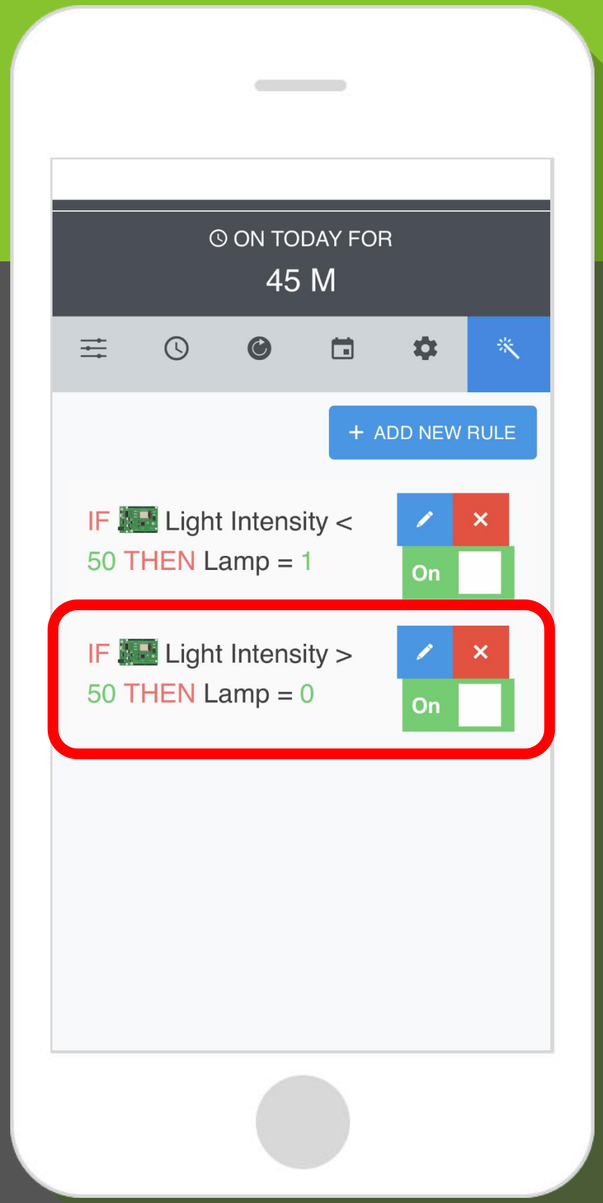
LABEL NAME: is the name that appears in the application.
PARAMETER NAME: is the name used in the code.




ON TODAY FOR
43 M
Click ADD NEW RULE
+ ADD NEW RULE
Let it happen
If this happens then do this! Click on the plus to create your first smart rule for this device
IF
THEN
ON TODAY FOR
43 M
IF THIS HAPPENS
Conditions type:
And
Or
ADD CONDITION
Click ADD CONDITION
THEN DO THIS
No actions yet
EDIT ACTIONS
ON TODAY FOR
43 M
SELECT_MODULE
_NEW_MODULE
Select the board
CANCEL
NEXT
Click NEXT


ON TODAY FOR
43 M
IF
ONLINE
Lamp
Light Intensity
Select Light Intensity
IS
=
>
<
50
CANCEL
BACK
FINISH
ON TODAY FOR
43 M
IF THIS HAPPENS
Light Intensity < 50
Conditions type:
And
Or
ADD CONDITION
THEN DO THIS
EDIT ACTIONS
Click EDIT ACTIONS
43 M
THEN
1
FEATURE 1
CANCEL
FINISH
3
Done
2
Lamp


ON TODAY FOR
43 M
THEN
Turn lamp ON
FEATURE 1
Lamp
ON
OFF
FEATURE 2
CANCEL
FINISH
Click FINISH
ON TODAY FOR
45 M
Click BACK
to return
IF THIS HAPPENS
Light Intensity < 50
Conditions type:
And
Or
ADD CONDITION
THEN DO THIS
Lamp 1
EDIT ACTIONS
ON TODAY FOR
45 M
+ ADD NEW RULE
IF Light Intensity <
50 THEN Lamp = 1
On
You can edit, delete,
or deactivate the
rule
Now when the
light intensity is
lower than 50%
then the yellow
LED (lamp in real
life) turns ON



Finally

Repeat the process but instead select the light intensity to be lower grater 50% and the action is to turn OFF.

You can adjust the light intensity percentage to fit your room light.

AFF IoT Board folder > Circuits > Examples > Automated_Lighting_System

Automated_Lighting_System

```
/*Start of mandatory lines of codes in each sketch*/
#define RX A0 // define the Receive pin (RX) to communicate with the WiFi module
#define TX A1 // define the Transmit pin (TX) to communicate with the WiFi module
#include <NeoSWSerial.h> // including the library to use the Software Serial rather than the Hardware Serial (Serial)
NeoSWSerial WiFiModule(RX, TX); //initialize the variable to use in communication with the WiFi module
/*End of mandatory lines of code*/

#define LightSensorPin   A2
#define YellowLedPin     10

void setup() {
  // put your setup code here, to run once:
  WiFiModule.begin(19200); // begin the communication between the WiFi module and the microcontroller on the board
}

void loop() {
  // put your main code here, to run repeatedly:
  if (WiFiModule.available() > 0) // if the WiFi module receive data from the server
  {
    String Command = WiFiModule.readStringUntil('\n'); // read the command sent from the WiFi module to the microcontroller

    if (Command.indexOf("lamp=1") >= 0) // if the received command contains "lamp=1" turn on the LED
    {
      digitalWrite(YellowLedPin, HIGH); // turn on the LED
    }
    if (Command.indexOf("lamp=0") >= 0) // if the received command contains "lamp=0" turn it off
    {
      digitalWrite(YellowLedPin, LOW); // turn off the LED
    }
  }

  int lightIntensity; // declare an integer to read the voltage

  lightIntensity = analogRead(LightSensorPin); // read the actual converted value (between 0 and 1023)
  lightIntensity = map(lightIntensity, 0, 1023, 0, 100); // transform the scale from 0 and 1023 to 0% and 100%

  WiFiModule.println("light_intensity=" + String(lightIntensity)); // send the light intensity value to the server

  delay(3000); // wait for 3 second (3000ms = 3s)
}
```

Open your sketch:

Automated_Lighting_System

What you **should see**

You should see the yellow LED turns ON when the brightness in the room decreases, and OFF when the brightness increases.

Troubleshooting

Refer to the troubleshootings for the Controlling_LED and the Reading_Light_Intensity projects.

Real World Application

Recently, most lamps come with integrated light sensor. They turn ON when the light is lower than preset threshold, and OFF otherwise.

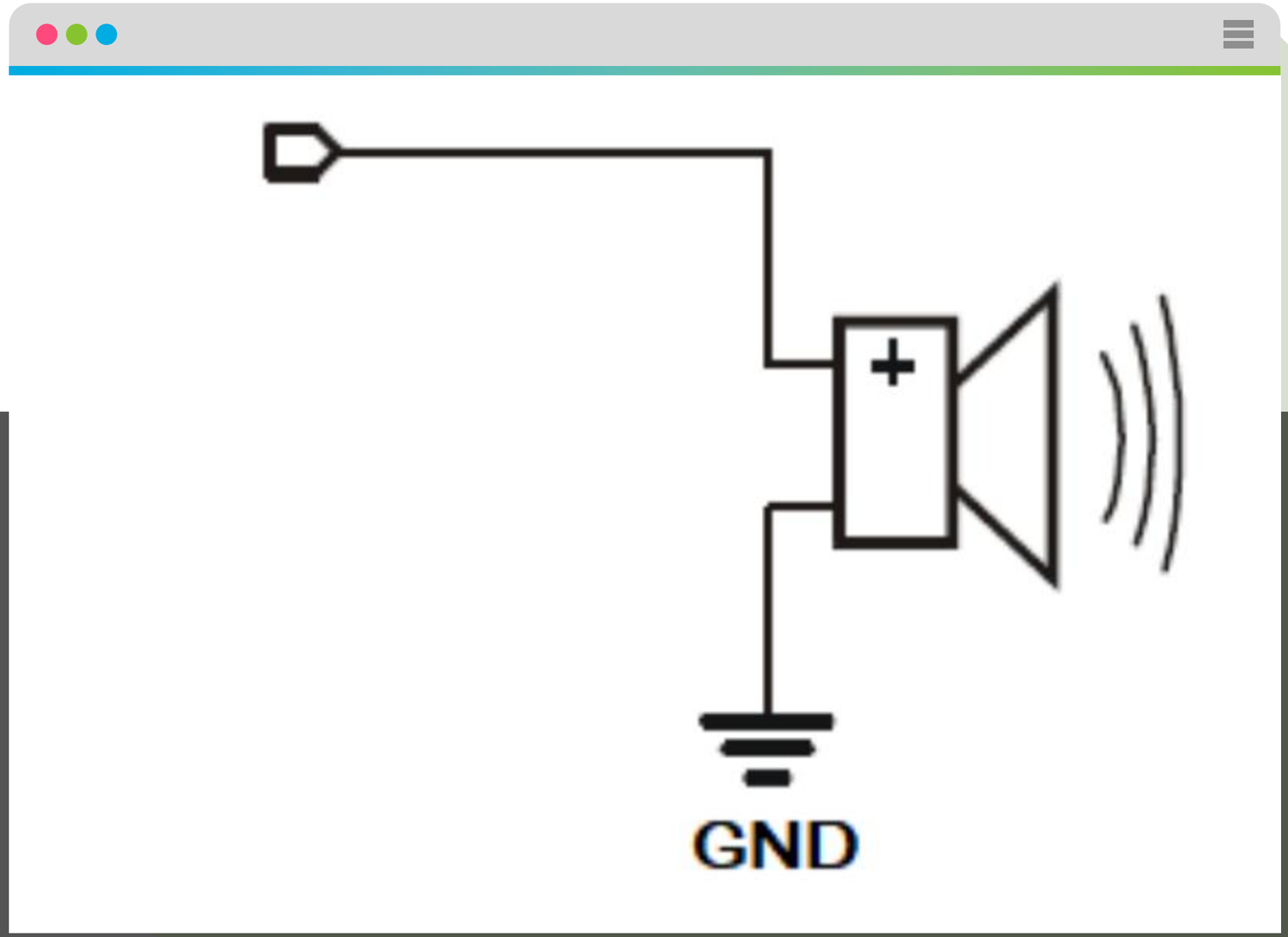
12 Building overheat alarm circuit

Piezo buzzer x1

Thermistor x1

1KΩ Resistor x1

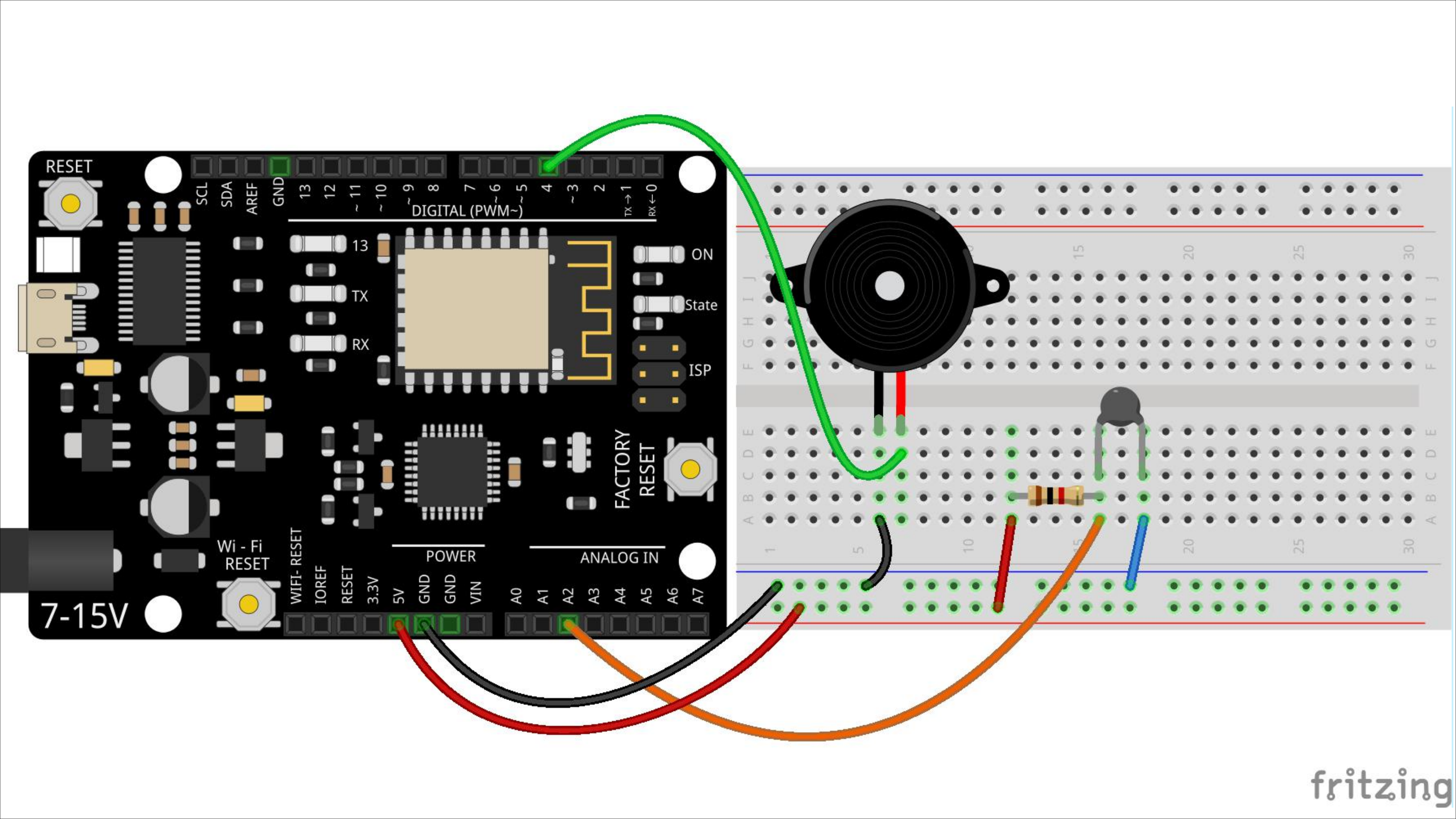
Jumper wires x7

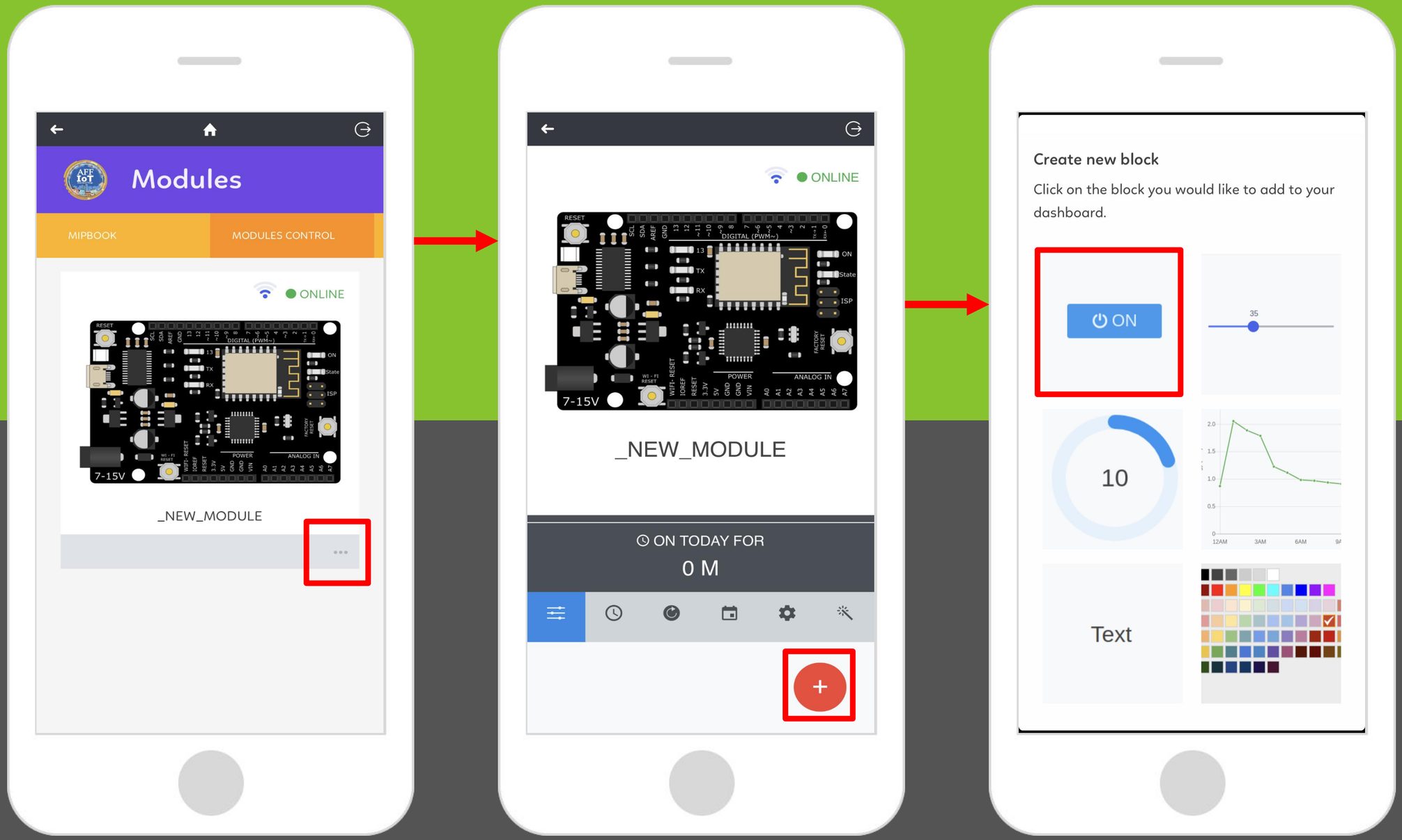


Piezo Buzzer

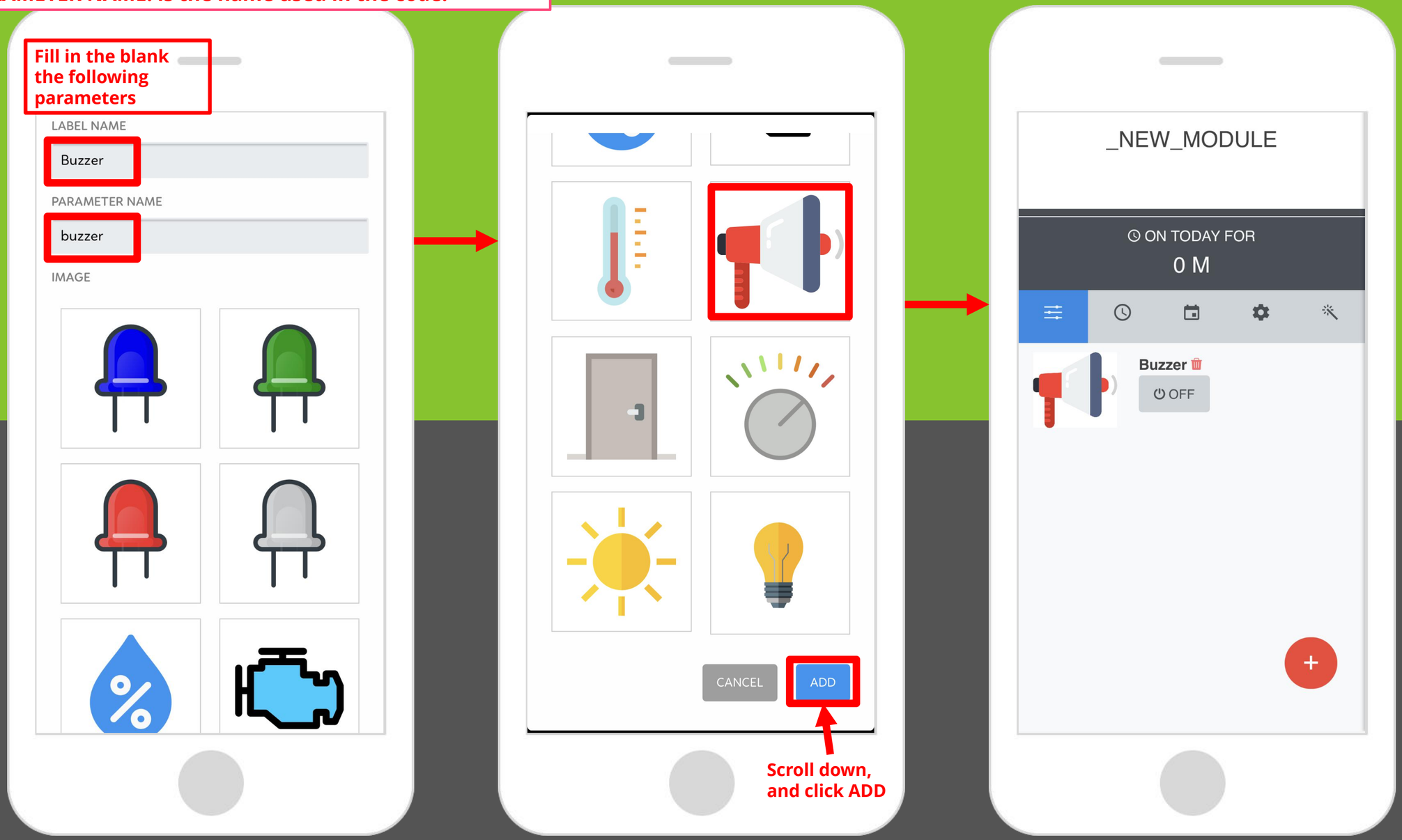
We'll be using a buzzer that makes a “tone" sound when applying voltage to it !

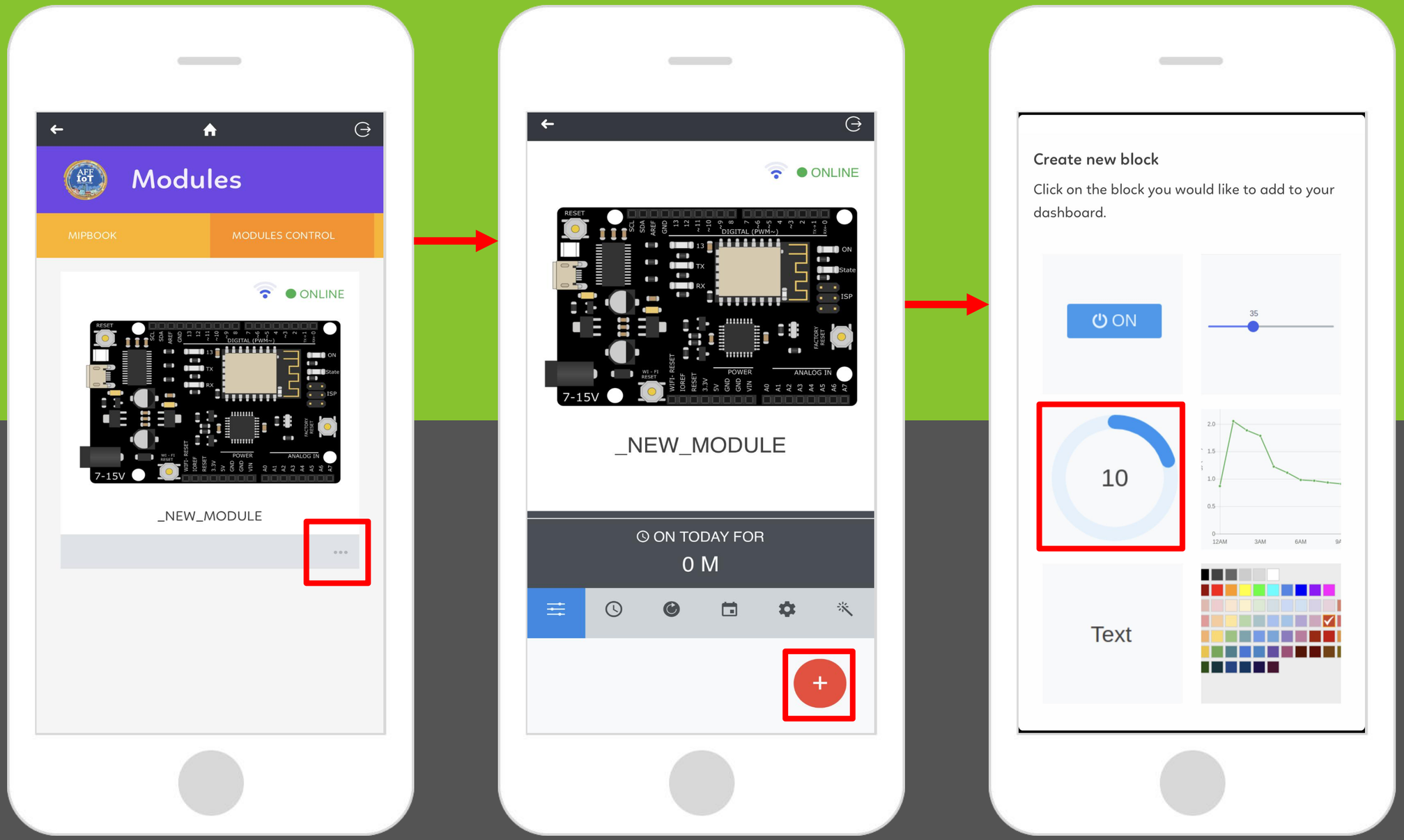
This tutorial is the introduction to the Alarm systems.


RESET
SCL
SDA
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TX → 1
RX ← 0
DIGITAL (PWM~)
13
TX
RX
ON
State
ISP
FACTORY RESET
Wi - Fi RESET
WIFI- RESET
IOREF
RESET
3.3V
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
A6
A7
ANALOG IN
7-15V
fritzing

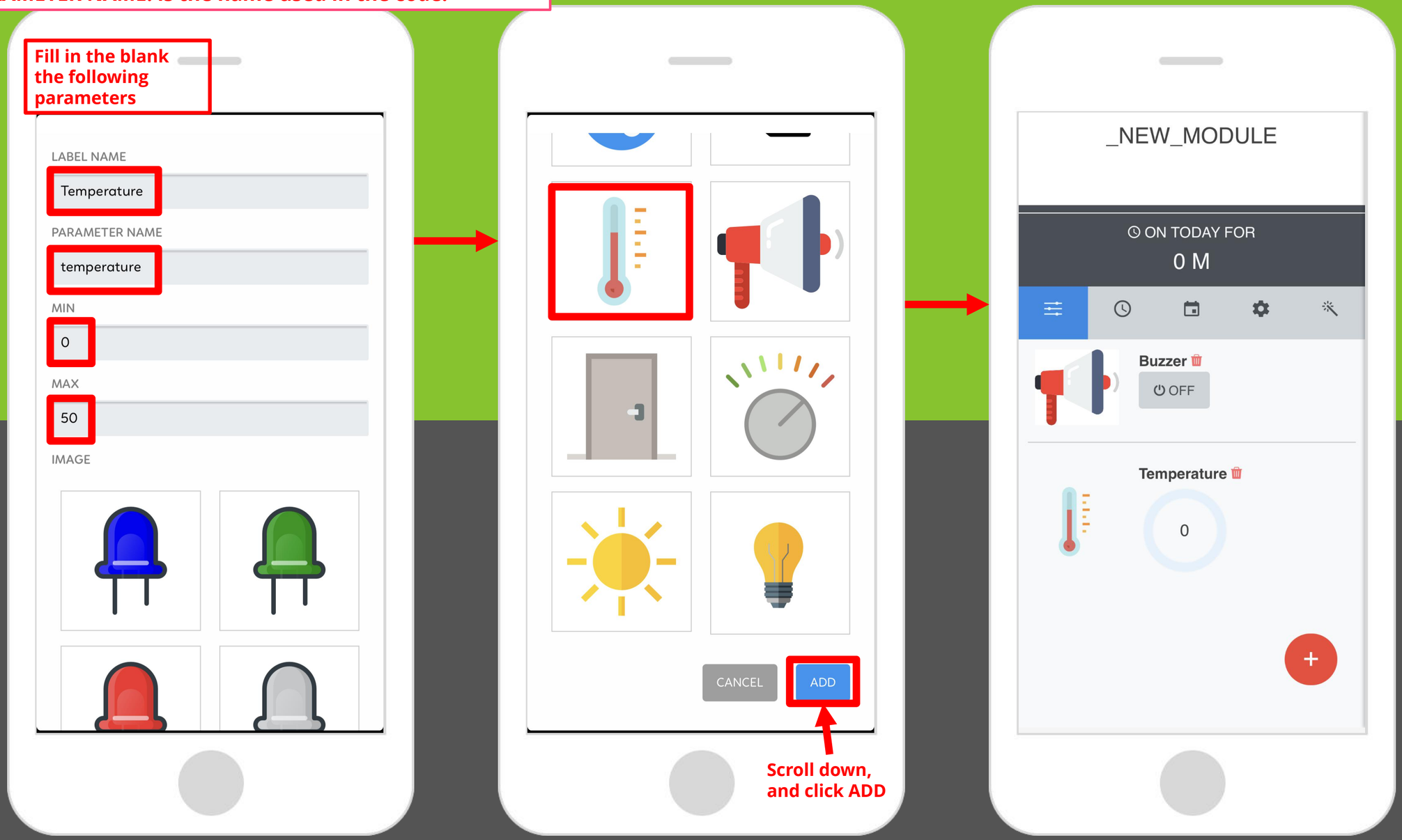

Modules
MIPBOOK
MODULES CONTROL
ONLINE
_NEW_MODULE
ON TODAY FOR
0 M
Create new block
Click on the block you would like to add to your dashboard.
ON
35
10
Text

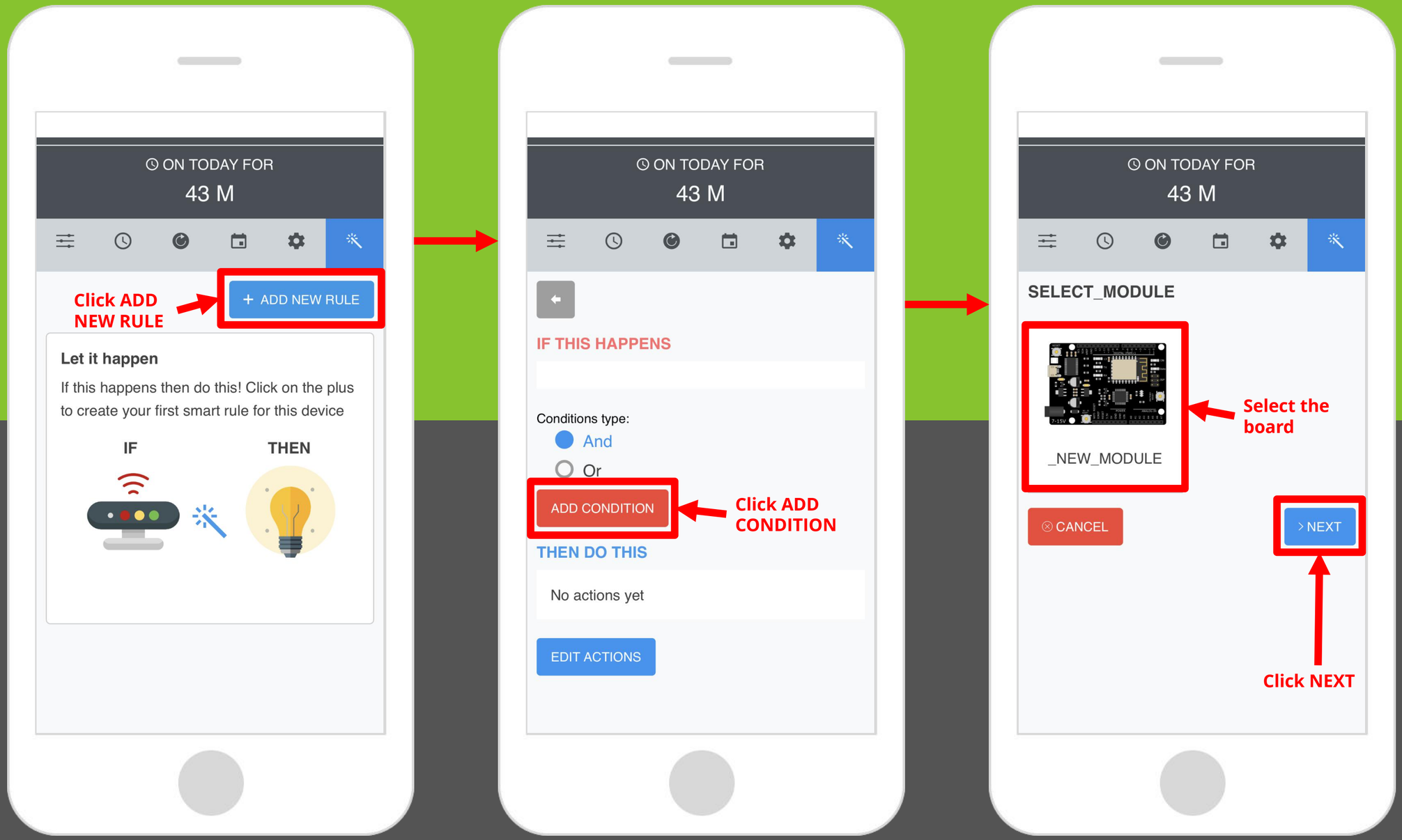
LABEL NAME: is the name that appears in the application.
PARAMETER NAME: is the name used in the code.

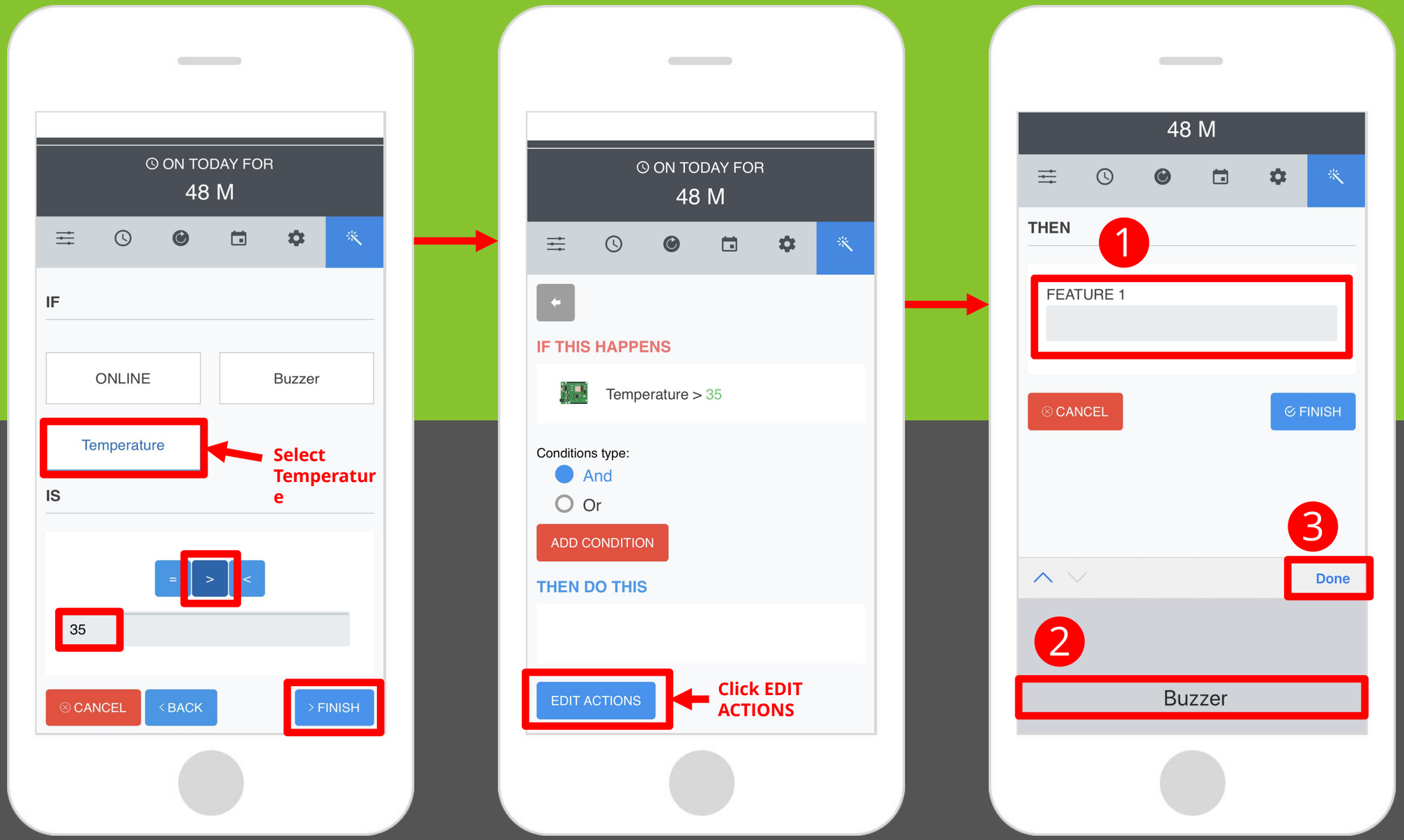



Modules
MIPBOOK
MODULES CONTROL
ONLINE
_NEW_MODULE
ONLINE
_NEW_MODULE
ON TODAY FOR
0 M
Create new block
Click on the block you would like to add to your dashboard.
ON
35
10
Text

LABEL NAME: is the name that appears in the application.
PARAMETER NAME: is the name used in the code.




ON TODAY FOR
43 M
Click ADD NEW RULE
+ ADD NEW RULE
Let it happen
If this happens then do this! Click on the plus to create your first smart rule for this device
IF
THEN
ON TODAY FOR
43 M
IF THIS HAPPENS
Conditions type:
And
Or
ADD CONDITION
Click ADD CONDITION
THEN DO THIS
No actions yet
EDIT ACTIONS
ON TODAY FOR
43 M
SELECT_MODULE
_NEW_MODULE
Select the board
CANCEL
NEXT
Click NEXT


ON TODAY FOR
48 M
IF
ONLINE
Buzzer
Temperature
Select Temperature
IS
=
>
<
35
CANCEL
BACK
FINISH
ON TODAY FOR
48 M
IF THIS HAPPENS
Temperature > 35
Conditions type:
And
Or
ADD CONDITION
THEN DO THIS
EDIT ACTIONS
Click EDIT ACTIONS
48 M
THEN
1
FEATURE 1
CANCEL
FINISH
3
Done
2
Buzzer


ON TODAY FOR
48 M
IF THIS HAPPENS
Temperature > 35
Conditions type:
And
Or
ADD CONDITION
THEN DO THIS
Buzzer =1
EDIT ACTIONS
ON TODAY FOR
48 M
+ ADD NEW RULE
IF Temperature >
35 THEN Buzzer = 1
On
You can edit or
delete the rule

Open AFF IoT Board// Circuits > Examples >Overheat_Alarm

Overheat_Alarm

```
/*Start of mandatory lines of codes in each sketch*/
#define RX A0 // define the Receive pin (RX) to communicate with the WiFi module
#define TX A1 // define the Transmit pin (TX) to communicate with the WiFi module
#include <NeoSWSerial.h> // including the library to use the Software Serial rather than the Hardware Serial (Serial)
NeoSWSerial WiFiModule(RX, TX); //initialize the variable to use in communication with the WiFi module
/*End of mandatory lines of code*/

#define BuzzerPin   4
#define TemperatureSensorPin A2


void setup() {
  // put your setup code here, to run once:
  WiFiModule.begin(19200); // begin the communication between the WiFi module and the microcontroller on the board
  pinMode(BuzzerPin, OUTPUT); // configure the pin connected to the Buzzer to be an output
}
void loop() {
  // put your main code here, to run repeatedly:
  float Voltage; // declare a decimal variable to store the read voltage
  int Temperature; // declare an integer variable to calculate the temperature
  Voltage = analogRead(TemperatureSensorPin) * 0.0048828125; // voltage = analog_value * (5/1024);

  Temperature = -21.231 * (Voltage - 3.765);
  // this equation is the linearized form of the temperature equation between 0 and 50 degrees (specific to the thermistor in the kit)

  WiFiModule.println("temperature=" + String(Temperature)); // send the Temperature value to the server

  delay(3000); // wait for 3 second (3000ms = 3s)

  if (WiFiModule.available() > 0) // if the WiFi module receive data from the server
  {
    String Command = WiFiModule.readStringUntil('\n'); // read the command sent from the WiFi module to the microcontroller
    if (Command.indexOf("buzzer=1") >= 0) // if the received command contains "buzzer=1" trigger on the Buzzer
    {
      digitalWrite(BuzzerPin, HIGH); // turn on the LED
    }
    if (Command.indexOf("buzzer=0") >= 0) // if the received command contains "buzzer=0" trigger it off
    {
      digitalWrite(BuzzerPin, LOW); // turn off the LED
    }
  }
}
```

Open your sketch: Overheat_Alarm

In this tutorial, we assumed that a temperature sensor is installed in the room. When the temperature rise to reach the 35^{0}C the buzzer should produce an alarm sound.

This code is a combination of the code to control the buzzer and the code to stream the temperature.

What you **should see**

You should see nothing! But you should hear a sound when you approach a lighter toward the sensor. If it isn't working, make sure you have assembled the circuit correctly and verified and uploaded the code to your board or see the troubleshooting tips below.

Troubleshooting

No Sound
Given the size and shape of the piezo buzzer it is easy to miss the right holes on the breadboard. Double check its placement.

Still No Sound !
This Piezo is polarized, so take care that the pin with label “+” should be connected to the digital pin, and the other one to GND.

Real World Application

All security systems use buzzers (or siren) upon triggering an alarm

Credit

All circuit diagrams are created with Fritzing (an open-source hardware initiative that makes electronics accessible as a creative material for anyone).

Grateful thanks for the Sparkfun Inventor’s Guide for their clarification images.