A 120-LED NeoPixel strip, powered the right way

Online guide, always up to date

Two meters of addressable LEDs with the FireBeetle 2 ESP32-E: a separate power supply, current at both ends, a capacitor, a resistor on the data line and a level shifter. Then a comet and a rainbow.

By **Ada Intelisano** · informaticait.it technical team

IN SHORT

- 120 LEDs all at white can draw up to 7.2 A (Adafruit's calculation): with a 5 V 4 A power supply the brightness is limited to 96 out of 255.
- Power goes in at both ends, so no LED is more than 1 m from a power point; the data goes in only at the start.
- A 1000 µF capacitor on the power supply, a 470 Ω resistor on Din close to the LEDs, all grounds in common.
- The FireBeetle works at 3.3 V: the Pixel Shifter brings the data signal up to 5 V.

Time: an afternoon | Level: three wires to solder | Before: guides no. 21 and 48

In guide no. 48, twelve addressable LEDs ran from the Arduino 5V pin. A real strip is another story: 120 LEDs over two meters, at full white, need far more current than a USB port can give, and the voltage drops along the strip. Here we wire it the right way: a separate power supply, current brought to both ends, a capacitor, a resistor on the data wire and a level shifter, because the FireBeetle 2 ESP32-E works at 3.3 V. At the end, two effects. The general rules are in the News article "NeoPixel addressable LEDs: power and wiring"; here we put them into practice.

1 The parts

- the **DFRobot FIT0613 strip**: 120 addressable RGB LEDs over 2 m, that is 60 per meter, at 5 V, 10 mm wide, with adhesive on the back;
- the **FireBeetle 2 ESP32-E** with its headers soldered and its USB-C cable;
- the **Adafruit Pixel Shifter**, which raises the data signal from 3.3 to 5 V, all with screw terminals;
- a **5 V 4 A power supply** with a 5.5 × 2.1 mm plug and the Adafruit **adapter** from DC jack to screw terminals;
- a **1000 µF electrolytic capacitor**, rated 6.3 V or more (we use a 10 V one), and a **470 Ω resistor**;
- two three-wire **lever connectors**, 4 m of 0.75 mm² **red and black cable**, three female-to-male jumper wires, heat-shrink tubing, a soldering iron and a multimeter.

New word: power injection
Bringing power to several points of the strip, not just the start. Data always goes in only at the first LED; power, Adafruit writes, can come in anywhere.

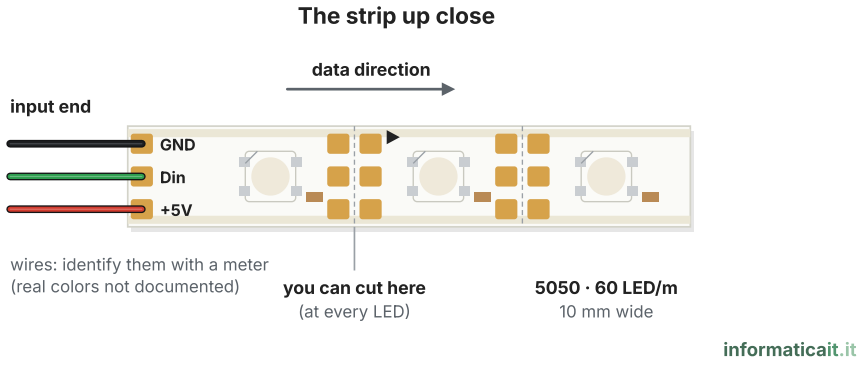
2 First, work out the current

Adafruit's calculation	Per LED	120 LEDs
Worst case: all white at full brightness	60 mA	**7.2 A**
Rule for normal animations	20 mA	2.4 A
Worst case with `setBrightness(96)` (our calculation)	–	about 2.7 A

The calculation comes from Adafruit's guide: the number of LEDs times 20, or times 60 if you want to be safe, divided by 1000, gives you the amps. For this strip DFRobot states a consumption that does not match Adafruit's calculation: we use the more cautious one.

The 4 A power supply is enough for animations, not for two meters of white at full brightness. That is why the sketches fix the brightness with `setBrightness(96)`: out of 255 that is 38%, and even in the worst case you stay around 2.7 A (our calculation, 7.2 × 96 ÷ 255, assuming the current goes down in proportion). If one day you want more light, get a power supply with more amps: as Adafruit puts it, “Extra Amps = good. Extra Volts = bad.”

3 Prepare the strip

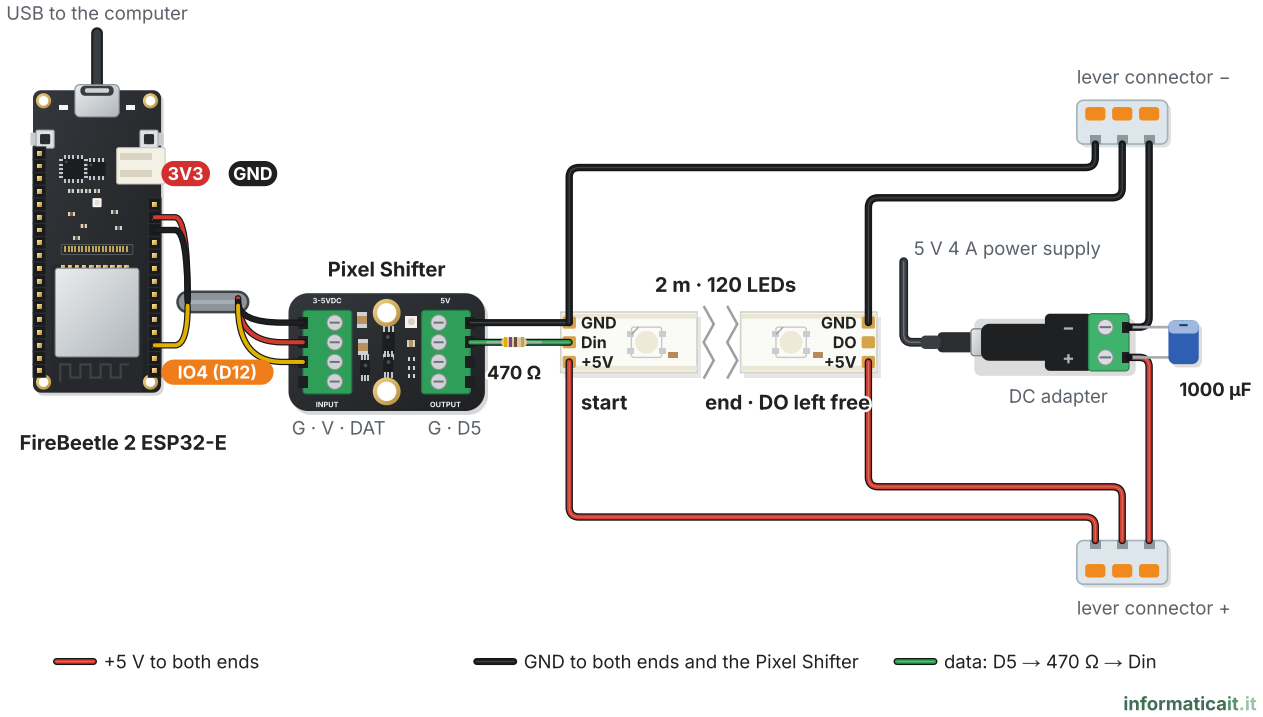


- **Find the start.** Adafruit warns that the connector does not always mark the input end: go by the printed triangles, which point in the direction of the data. The start is the end with **Din**.
- **Identify the wires.** At each end DFRobot fits a three-way connector and two loose wires, but does not say which wire is which. With the multimeter in continuity mode (guide no. 8), find the wire that goes to the +5V pad, the Din (or DO) one and the GND one, and mark them with a piece of tape. Adafruit says it again: always trust the labels printed on the strip, not the colors.
- **At the start**, cut off the connector, leaving the wires a few centimeters long: Din and GND go to the Pixel Shifter, and the connector's +5V wire gets insulated with heat-shrink. The two loose wires carry the power.

- **At the end** you only need the two loose wires, +5V and GND. DO stays free: nothing goes at the far end of the strip.
- The example in the DFRobot wiki powers the strip from the 5V pin of an Arduino UNO and counts 180 LEDs: don't follow it.

4 The wiring

The wiring: data at one end, power at both ends



informaticait.it

- **Data.** IO4 (D12) on the FireBeetle goes to **DAT** on the Pixel Shifter; **D5** goes to the strip's Din through the 470 Ω resistor. Adafruit wants the resistor at the LED end: solder it on the Din wire, close to the strip, and cover it with heat-shrink. Keep the data wire short: according to Adafruit, a meter or two is usually fine.
- **Pixel Shifter.** V to **3V3** and G to **GND** on the FireBeetle: Adafruit asks you to power it at the logic voltage of the board (it accepts 3 to 5 V), and it makes the 5 V for the data on its own. There is no +5 V on the output: the output G goes to the strip's GND, and that is how the grounds become one.
- **Power.** The + and − of the adapter go to the two lever connectors; from each connector one wire runs to the start and one to the end of the strip, red on +5V and black on GND. For the long run use the 0.75 mm² cable (our proposal: Adafruit does not give a wire size).
- **Capacitor.** Its two leads go into the same adapter terminals as the wires: the light stripe (−) on −.

5 Power up in the right order

1. Before connecting the strip, plug in the power supply and measure with the multimeter on the adapter terminals: the **+** must read +5 V. The adapter assumes the positive is in the center of the plug, and Qoltec does not state the polarity.
2. Unplug it and wire everything: first the GNDs, then +5 V, data last. That is the order Adafruit gives.
3. Power the strip **before** the FireBeetle: otherwise the LEDs try to draw power from the data wire.
4. Only now connect the FireBeetle to the computer with the USB cable.

6 The first test

```
// Strip test: the 120 LEDs light up one at a time, then all white.
// On the Serial Monitor (115200) the current estimated with the Adafruit rule.
#include <Adafruit_NeoPixel.h>

const int DATA_PIN = 4;      // IO4 = D12, goes to DAT of the Pixel Shifter
const int NUM_LED = 120;     // 2 meters at 60 LEDs per meter
const int BRIGHTNESS = 96;   // out of 255: keeps the current below 4 A (see the guide)

Adafruit_NeoPixel strip(NUM_LED, DATA_PIN, NEO_GRB + NEO_KHZ800);

// Adafruit counts up to 60 mA per LED at full white (red + green + blue): here we
// split it evenly, 20 mA per color at the value 255 (our estimate).
// getPixels() gives the values actually sent, already reduced by setBrightness.
float estimateAmps() {
  uint8_t *values = strip.getPixels();
  long sum = 0;
  for (int i = 0; i < NUM_LED * 3; i++) {
    sum += values[i];
  }
  return sum * 0.020 / 255.0;
}

void fillStrip(uint32_t color) {   // one LED at a time, from the first to the last
  for (int i = 0; i < NUM_LED; i++) {
    strip.setPixelColor(i, color);
    strip.show();
    delay(15);
  }
  Serial.printf("Estimated current: %.2f A\n", estimateAmps());
}

void setup() {
  Serial.begin(115200);
  strip.begin();
  strip.setBrightness(BRIGHTNESS);   // only once, here in setup
  strip.clear();
  strip.show();
```

```
}

void loop() {
  fillStrip(strip.Color(255, 0, 0));       // red
  fillStrip(strip.Color(0, 255, 0));       // green
  fillStrip(strip.Color(0, 0, 255));       // blue
  fillStrip(strip.Color(255, 255, 255));   // white: it needs the most current
  delay(5000);                             // do the LEDs at the far end stay white?
  strip.clear();
  strip.show();
  delay(1000);
}
```

`estimateAmps()` adds up the red, green and blue values actually sent to the strip, already reduced by `setBrightness`, and turns them into amps with Adafruit's calculation. It is an estimate, useful for staying within limits.

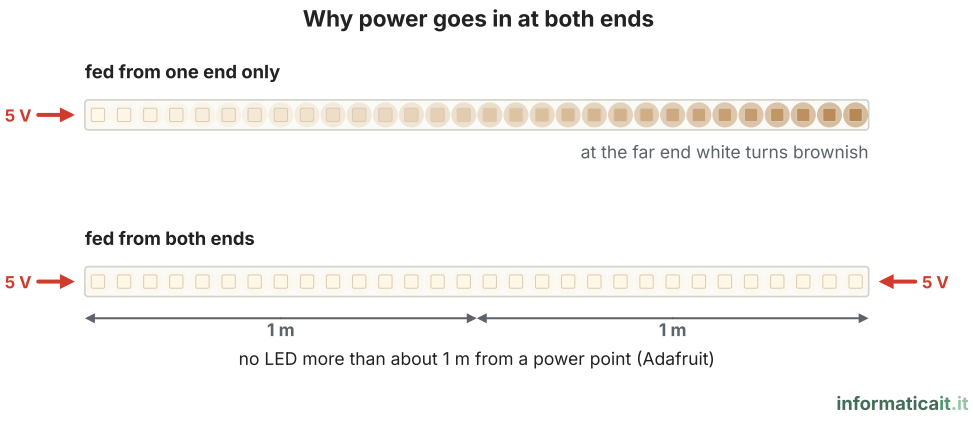
✓ What you should see

The LEDs light up one at a time all the way to the end: red, green, blue, then white. On the Serial Monitor, at 115200 baud, the current estimate: 0.90 A with a single color, 2.71 A with white.

If it doesn't work

- Nothing lights up: look at the small LED on the Pixel Shifter, which sits on the data line. If it lights up, the data is getting through: check Din (did you pick the right end?) and the GND between the Pixel Shifter and the strip.
- The first LED works and the others show random colors: the common ground is missing, or the data wire is too long.
- Red and green swapped: in the line that declares the strip, try `NEO_RGB` instead of `NEO_GRB`.
- With white, the last LEDs turn yellowish or brownish: the wires at the end are not carrying current. Check the lever connectors.

7 Why both ends



The copper in the strip has its own resistance: the farther the current travels, the more the voltage drops. Adafruit describes it like this: the farthest LEDs turn brownish, because blue

and green need more voltage than red. The rule it gives is to keep every LED within about 1 m of a power point. With two meters, feeding both ends, the farthest LED is right in the middle, one meter away.

To see it, with the white of the test, take the two end wires out of the lever connectors and compare the first and the last LED, then reconnect them.

8 Two effects: the comet and the rainbow

```
// Two effects on the 120 LEDs, taking turns every 15 seconds: comet and rainbow.
#include <Adafruit_NeoPixel.h>

const int DATA_PIN = 4;               // IO4 = D12, goes to DAT of the Pixel Shifter
const int NUM_LED = 120;
const int BRIGHTNESS = 96;            // the same as in the test: current below 4 A
const int TAIL = 20;                  // how many LEDs long the comet's tail is
const unsigned long DURATION = 15000; // milliseconds for each effect

Adafruit_NeoPixel strip(NUM_LED, DATA_PIN, NEO_GRB + NEO_KHZ800);

int head = 0;           // where the comet is
uint16_t hue = 0;       // color of the comet and starting point of the rainbow

// The comet: the head at full light and the tail behind it, darker and darker.
void comet() {
  strip.clear();
  for (int j = 0; j < TAIL; j++) {
    int i = head - j;
    if (i < 0) {
      i += NUM_LED;                   // the tail goes on from the start of the strip
    }
    uint8_t light = 255 - j * 255 / TAIL;
    strip.setPixelColor(i, strip.gamma32(strip.ColorHSV(hue, 255, light)));
  }
  strip.show();
  head = (head + 1) % NUM_LED;        // one LED further on
  hue += 256;                         // and a slightly different color
  delay(15);
}

// The rainbow: the whole color wheel spread over the 120 LEDs, moving at every step.
void rainbow() {
  for (int i = 0; i < NUM_LED; i++) {
    uint16_t t = hue + i * 65536L / NUM_LED;
    strip.setPixelColor(i, strip.gamma32(strip.ColorHSV(t)));
  }
  strip.show();
  hue += 256;
  delay(10);
}

void setup() {
  strip.begin();
  strip.setBrightness(BRIGHTNESS);
```

```
  strip.show();
}

void loop() {
  if ((millis() / DURATION) % 2 == 0) {
    comet();
  } else {
    rainbow();
  }
}
```

- **The comet**: at each step the strip goes dark, then twenty LEDs light up, the head and its tail, each one a little darker. The third number in `ColorHSV` is the light level, from 255 down to almost zero.
- **The rainbow**: the color wheel, 65536 hues, spread over 120 LEDs; each step shifts it by 256.
- `gamma32` makes the colors look more natural: Adafruit recommends always using it together with `ColorHSV`.
- `millis() / DURATION % 2` is 0 for 15 seconds, then 1, then 0 again: that is the switch between effects, without stopping the sketch.

✓ What you should see
The comet runs along the two meters, changing color; after 15 seconds the moving rainbow starts, then the comet comes back.

Try it yourself

- Add a third effect: the `theaterChase` from the library's strandtest example.
- Change `BRIGHTNESS` in the test sketch and watch the estimate: stay below the 4 A of the power supply.
- Cut the strip on a cut line and update `NUM_LED`.

✓ What you learned

- work out the current before choosing the power supply;
- data goes in at the start, power goes to both ends;
- capacitor on the power supply, resistor on Din at the LED end, common grounds;
- with a 3.3 V board, the data goes through a level shifter to 5 V.

Sources: DFRobot, FIT0613 strip: product page, wiki and photos · Adafruit, Pixel Shifter (6066): product page, Learn guide and Eagle files · Adafruit, “NeoPixel Überguide”; Adafruit NeoPixel library 1.15.5 and strandtest example · DFRobot, FireBeetle 2 ESP32-E: wiki and schematic; Espressif, arduino-esp32 · Qoltec, power supply 50778; Adafruit, adapter 368 · informaticait.it: article “NeoPixel addressable LEDs: power and wiring”; guides no. 8, 21 and 48