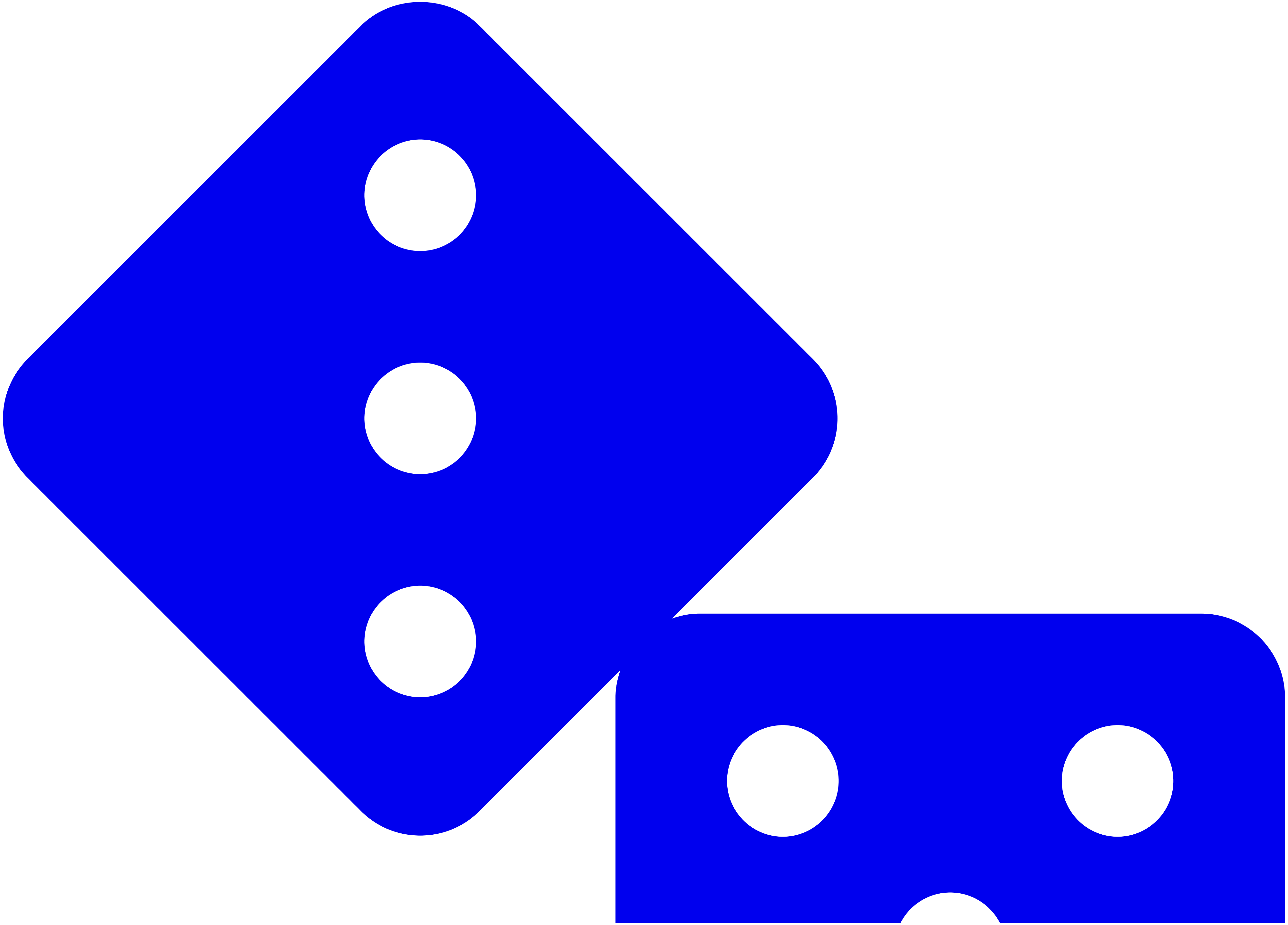


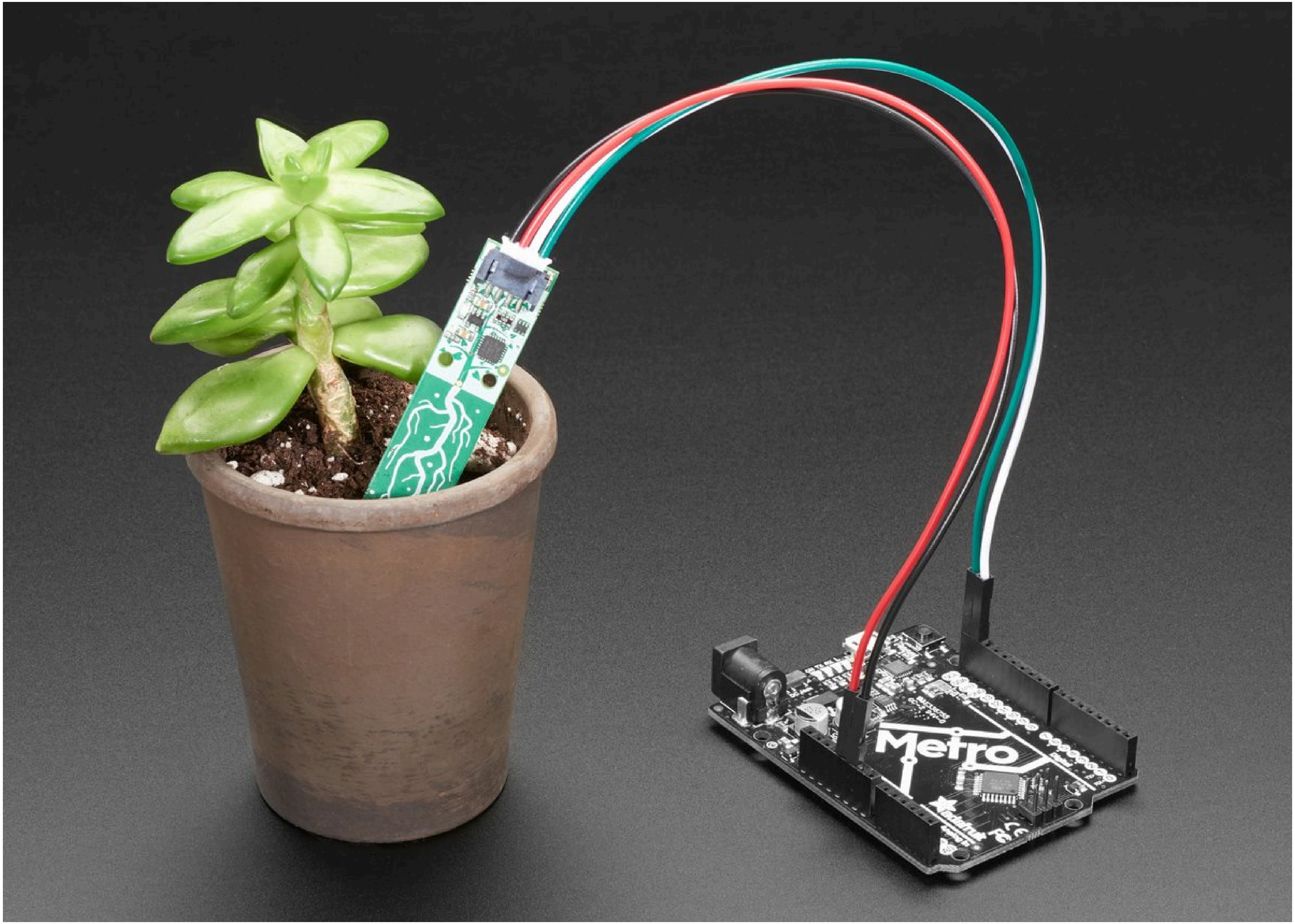
- 

- Shop
- Learn
- Blog
- Forums
- LIVE!
- AdaBox
- IO

Sign In
0

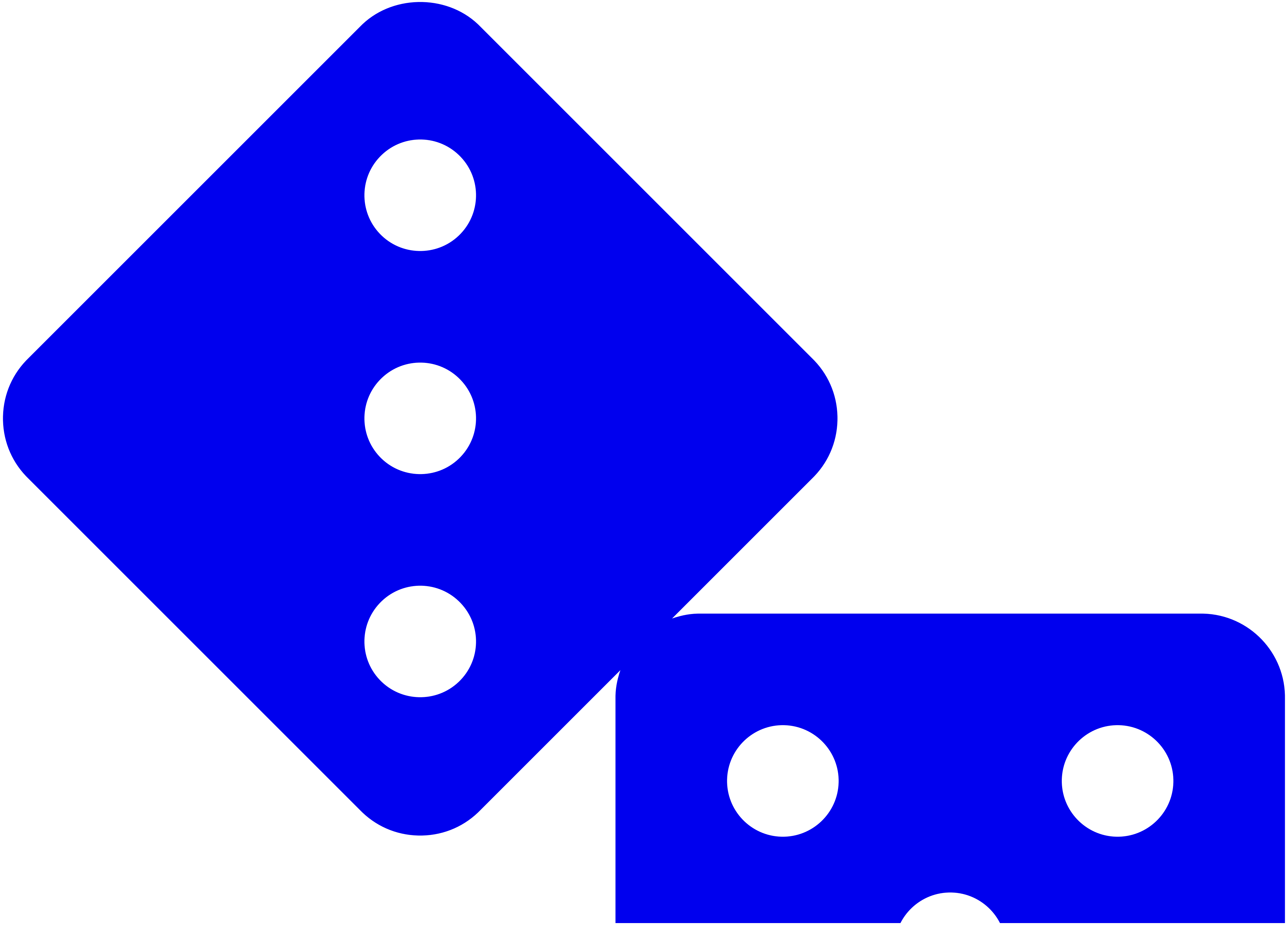
- Explore & Learn
  Learn Categories
  - 3D Printing
  - AdaBox
  - Adafruit Products
  - Arduino Compatibles
  - Breakout Boards
  - Circuit Playground
  - CircuitPython
  - CLUE

- Community Support
- Components
- Crickit
- Customer & Partner Projects
- Development Boards
- Educators
- EL Wire/Tape/Panel
- Feather
- Gaming
- Hacks
- Internet of Things - IOT
- LCDs & Displays
- LEDs
- Machine Learning
- MakeCode
- Maker Business
- micro:bit
- Microcontrollers
- Programming
- Raspberry Pi
- Robotics & CNC
- Sensors
- STEMMA
- Tools
- Trellis
- Wearables


Metro

Explore
Groups to get your gears turning
[Explore Groups]

- [New Guides]
- [Playground]



[Adafruit Data Logger Shield](https://learn.adafruit.com/adafruit-data-logger-shield) Code Walkthrough

### Adafruit Data Logger Shield

By Bill Earl
Adafruit's Data Logger Shield, now pre-assembled!

- Single page
- Download PDF

Feedback? Corrections?

# Code Walkthrough

Save Subscribe

×

### New Subscription

Please sign in to subscribe to this guide.

You will be redirected back to this guide once you sign in, and can then subscribe to this guide.

Close

## Introduction

This is a walkthrough of the Light and Temperature Logging sketch. Its long and detailed so we put it here for your perusal. We strongly suggest reading through it, the code is very versatile and our text descriptions should make it clear why everything is there!

Download the complete file here:

## Includes and Defines

Download File
Copy Code

```
#include "SD.h"
#include <Wire.h>
#include "RTClib.h"
```

OK this is the top of the file, where we include the three libraries we'll use: the **SD** library to talk to the card, the **Wire** library that helps the Arduino with i2c and the **RTClib** for chatting with the real time clock
Download File
Copy Code

```
// A simple data logger for the Arduino analog pins
#define LOG_INTERVAL  1000 // mills between entries
#define ECHO_TO_SERIAL   1 // echo data to serial port
#define WAIT_TO_START    0 // Wait for serial input in setup()


// the digital pins that connect to the LEDs
#define redLEDpin 3
#define greenLEDpin 4
```

```
// The analog pins that connect to the sensors
#define photocellPin 0           // analog 0
#define tempPin 1                // analog 1
```

Next are all the "defines" - the constants and tweakables.

- `LOG_INTERVAL` is how many milliseconds between sensor readings. 1000 is 1 second which is not a bad starting point
- `ECHO_TO_SERIAL` determines whether to send the stuff thats being written to the card also out to the Serial monitor. This makes the logger a little more sluggish and you may want the serial monitor for other stuff. On the other hand, its hella useful. We'll set this to **1** to keep it on. Setting it to **0** will turn it off
- `WAIT_TO_START` means that you have to send a character to the Arduino's Serial port to kick start the logging. If you have this on you basically can't have it run away from the computer so we'll keep it off (set to **0**) for now. If you want to turn it on, set this to **1**

The other defines are easier to understand, as they are just pin defines

- `redLEDpin` is whatever you connected to the Red LED on the logger shield
- `greenLEDpin` is whatever you connected to the Green LED on the logger shield
- `photocellPin` is the analog input that the CdS cell is wired to
- `tempPin` is the analog input that the TMP36 is wired to

# Objects and error()

Download File
Copy Code

```
RTC_DS1307 RTC; // define the Real Time Clock object


// for the data logging shield, we use digital pin 10 for the SD cs line
const int chipSelect = 10;

// the logging file
File logfile;

void error(char *str)
{
  Serial.print("error: ");
  Serial.println(str);

  // red LED indicates error
  digitalWrite(redLEDpin, HIGH);

  while(1);
}
```

Next up we've got all the objects for the RTC, and the SD card chip select pin. For all our shields we use **pin 10** for SD card chip select lines

Next is the `error()` function, which is just a shortcut for us, we use it when something Really Bad happened, like we couldn't write to the SD card or open it. It prints out the error to the Serial Monitor, turns on the red error LED, and then sits in a `while(1);` loop forever, also known as a **halt**

## Setup

Download File
Copy Code

```
void setup(void)
{
  Serial.begin(9600);
  Serial.println();

#if WAIT_TO_START
  Serial.println("Type any character to start");
  while (!Serial.available());
#endif //WAIT_TO_START
```

K now we are onto the code. We begin by initializing the Serial port at 9600 baud. If we set `WAIT_TO_START` to anything but **0**, the Arduino will wait until the user types something in. Otherwise it goes ahead to the next part

Download File
Copy Code

```
  // initialize the SD card
  Serial.print("Initializing SD card...");
  // make sure that the default chip select pin is set to
  // output, even if you don't use it:
  pinMode(10, OUTPUT);

  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    return;
  }
  Serial.println("card initialized.");

  // create a new file
  char filename[] = "LOGGER00.CSV";
  for (uint8_t i = 0; i < 100; i++) {
    filename[6] = i/10 + '0';
    filename[7] = i%10 + '0';
    if (! SD.exists(filename)) {
      // only open a new file if it doesn't exist
      logfile = SD.open(filename, FILE_WRITE);
      break;  // leave the loop!
    }
  }

  if (! logfile) {
    error("couldnt create file");
  }

  Serial.print("Logging to: ");
  Serial.println(filename);
```

Now the code starts to talk to the SD card, it tries to initialize the card and find a FAT16/FAT32 partition.

Next it will try to make a logfile. We do a little tricky thing here, we basically want the files to be called something like **LOGGERnn.csv** where **nn** is a number. By starting out trying to create **LOGGER00.CSV** and incrementing every time when the file already exists, until we get to **LOGGER99.csv**, we basically make a new file every time the Arduino starts up

To create a file, we use some Unix style command flags which you can see in the `logfile.open()` procedure. **FILE_WRITE** means to create the file and write data to it.

Assuming we managed to create a file successfully, we print out the name to the Serial port.
Download File
Copy Code

```
  Wire.begin();
  if (!RTC.begin()) {
    logfile.println("RTC failed");
#if ECHO_TO_SERIAL
    Serial.println("RTC failed");
#endif  //ECHO_TO_SERIAL
  }


  logfile.println("millis,time,light,temp");
#if ECHO_TO_SERIAL
  Serial.println("millis,time,light,temp");
#if ECHO_TO_SERIAL// attempt to write out the header to the file
  if (logfile.writeError || !logfile.sync()) {
    error("write header");
  }

  pinMode(redLEDpin, OUTPUT);
  pinMode(greenLEDpin, OUTPUT);

   // If you want to set the aref to something other than 5v
  //analogReference(EXTERNAL);
}
```

OK we're wrapping up here. Now we kick off the RTC by initializing the Wire library and poking the RTC to see if its alive.

Then we print the header. The header is the first line of the file and helps your spreadsheet or math program identify whats coming up next. The data is in CSV (comma separated value) format so the header is too: "millis,time,light,temp" the first item **millis** is milliseconds since the Arduino started, **time** is the time and date from the RTC, **light** is the data from the CdS cell and **temp** is the temperature read.

You'll notice that right after each call to **logfile.print()** we have #if ECHO_TO_SERIAL and a matching **Serial.print()** call followed by a #if ECHO_TO_SERIAL this is that debugging output we mentioned earlier. The **logfile.print()** call is what writes data to our file on the SD card, it works pretty much the same as the **Serial** version. If you set **ECHO_TO_SERIAL** to be **0** up top, you won't see the written data printed to the Serial terminal.

Finally, we set the two LED pins to be outputs so we can use them to communicate with the user. There is a commented-out line where we set the analog reference voltage. This code assumes that you will be using the 'default' reference which is the VCC voltage for the chip - on a classic Arduino this is 5.0V. You can get better precision sometimes by lowering the reference. However we're going to keep this simple for now! Later on, you may want to experiment with it.

## Main loop

Now we're onto the loop, the loop basically does the following over and over:

1. Wait until its time for the next reading (say once a second - depends on what we defined)
2. Ask for the current time and date froom the RTC
3. Log the time and date to the SD card
4. Read the photocell and temperature sensor
5. Log those readings to the SD card
6. Sync data to the card if its time

## Timestamping

Lets look at the first section:
Download File
Copy Code

```
void loop(void)
{
  DateTime now;

  // delay for the amount of time we want between readings
  delay((LOG_INTERVAL -1) - (millis() % LOG_INTERVAL));

  digitalWrite(greenLEDpin, HIGH);

  // log milliseconds since starting
  uint32_t m = millis();
  logfile.print(m);           // milliseconds since start
  logfile.print(", ");
#if ECHO_TO_SERIAL
  Serial.print(m);         // milliseconds since start
  Serial.print(", ");
#endif

  // fetch the time
  now = RTC.now();
  // log time
  logfile.print(now.get()); // seconds since 2000
  logfile.print(", ");
  logfile.print(now.year(), DEC);
  logfile.print("/");
  logfile.print(now.month(), DEC);
  logfile.print("/");
  logfile.print(now.day(), DEC);
  logfile.print(" ");
  logfile.print(now.hour(), DEC);
  logfile.print(":");
  logfile.print(now.minute(), DEC);
  logfile.print(":");
  logfile.print(now.second(), DEC);
#if ECHO_TO_SERIAL
  Serial.print(now.get()); // seconds since 2000
  Serial.print(", ");
  Serial.print(now.year(), DEC);
  Serial.print("/");
  Serial.print(now.month(), DEC);
```

```
  Serial.print("/");
  Serial.print(now.day(), DEC);
  Serial.print(" ");
  Serial.print(now.hour(), DEC);
  Serial.print(":");
  Serial.print(now.minute(), DEC);
  Serial.print(":");
  Serial.print(now.second(), DEC);
#endif //ECHO_TO_SERIAL
```

The first important thing is the **delay()** call, this is what makes the Arduino wait around until its time to take another reading. If you recall we **#defined** the delay between readings to be 1000 millseconds (1 second). By having more delay between readings we can use less power and not fill the card as fast. Its basically a tradeoff how often you want to read data but for basic long term logging, taking data every second or so will result in plenty of data!

Then we turn the green LED on, this is useful to tell us that yes we're reading/writing data now.

Next we call **millis**() to get the 'time since arduino turned on' and log that to the card. It can be handy to have - especially if you end up not using the RTC.

Then the familiar **RTC.now()** call to get a snapshot of the time. Once we have that, we write a timestamp (seconods since 2000) as well as the date in **YY/MM/DD HH:MM:SS** time format which can easily be recognized by a spreadsheet. We have both because the nice thing about a timestamp is that its going to montonically increase and the nice thing about printed out date is its human readable

## Log sensor data

Next is the sensor logging code

Download File
Copy Code

```
int photocellReading = analogRead(photocellPin);
  delay(10);
  int tempReading = analogRead(tempPin);

  // converting that reading to voltage, for 3.3v arduino use 3.3
  float voltage = (tempReading * 5.0) / 1024.0;
  float temperatureC = (voltage - 0.5) * 100.0 ;
  float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;

  logfile.print(", ");
  logfile.print(photocellReading);
  logfile.print(", ");
  logfile.println(temperatureF);
#if ECHO_TO_SERIAL
  Serial.print(", ");
  Serial.print(photocellReading);
  Serial.print(", ");
  Serial.println(temperatureF);
#endif //ECHO_TO_SERIAL

  digitalWrite(greenLEDpin, LOW);
}
```

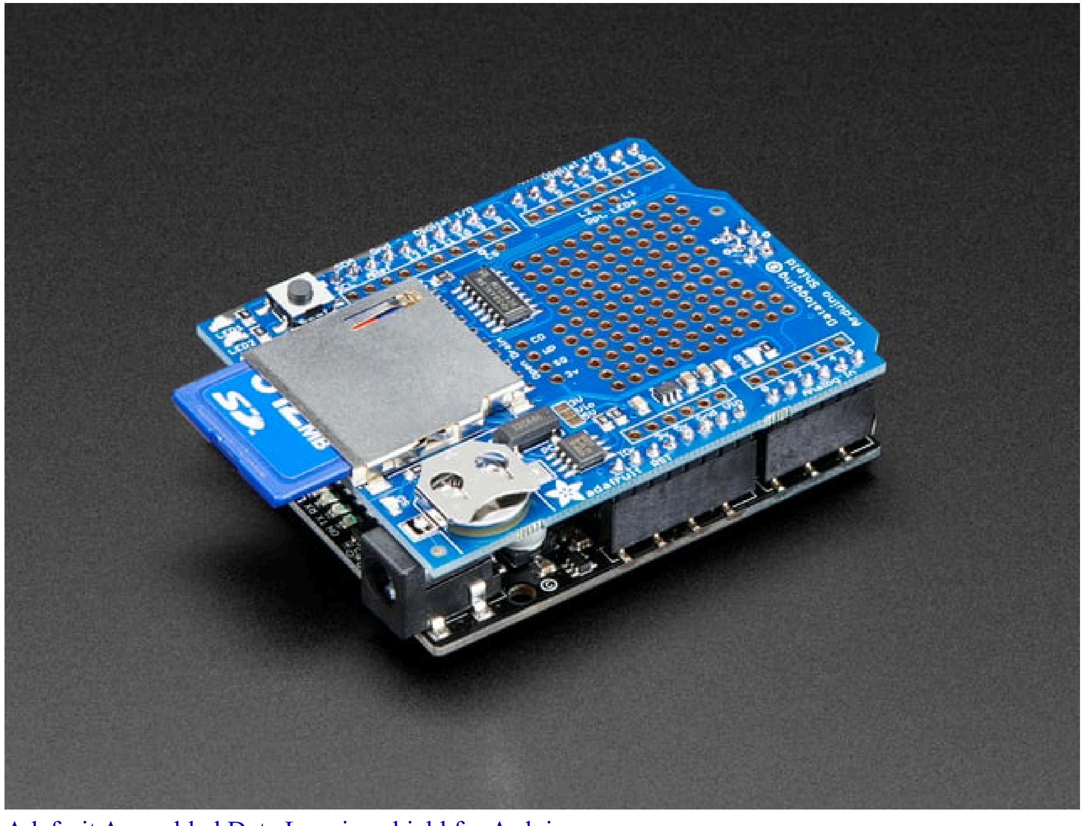
This code is pretty straight forward, the processing code is snagged from our earlier tutorial. Then we just **print()** it to the card with a comma seperating the two

We finish up by turning the green LED off
Use It! Downloads

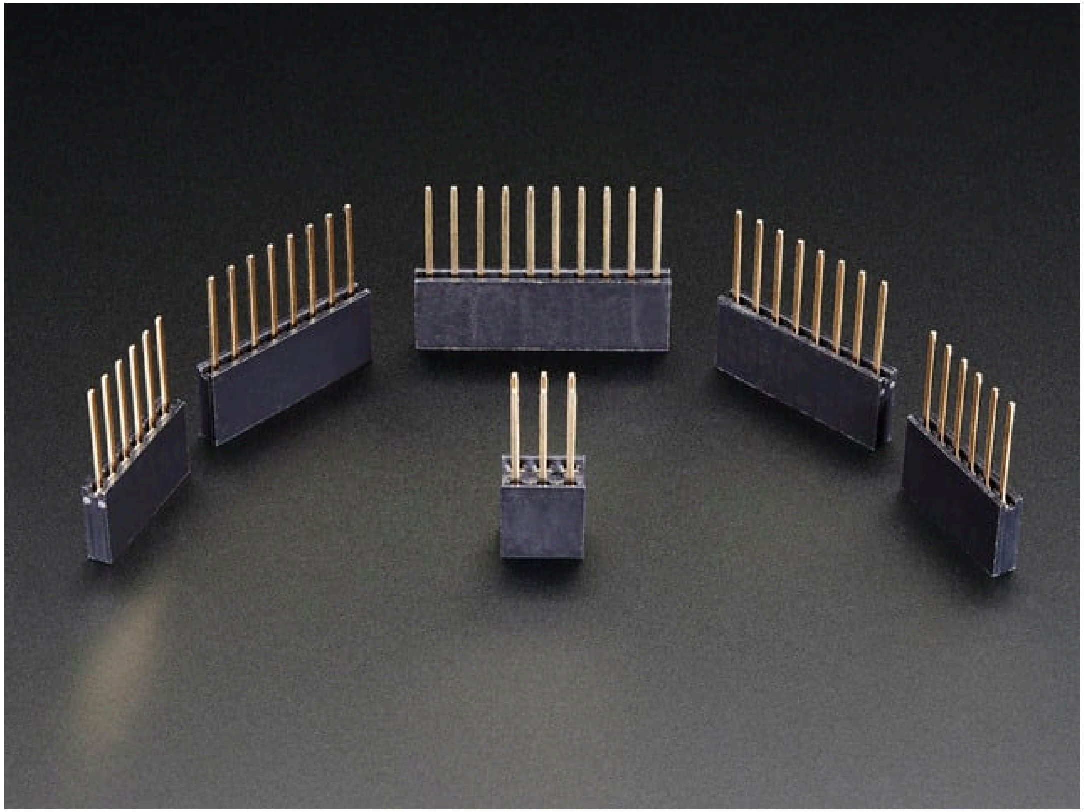
This guide was first published on Apr 12, 2013. It was last updated on Mar 20, 2024.

This page (Code Walkthrough) was last updated on Mar 30, 2013.

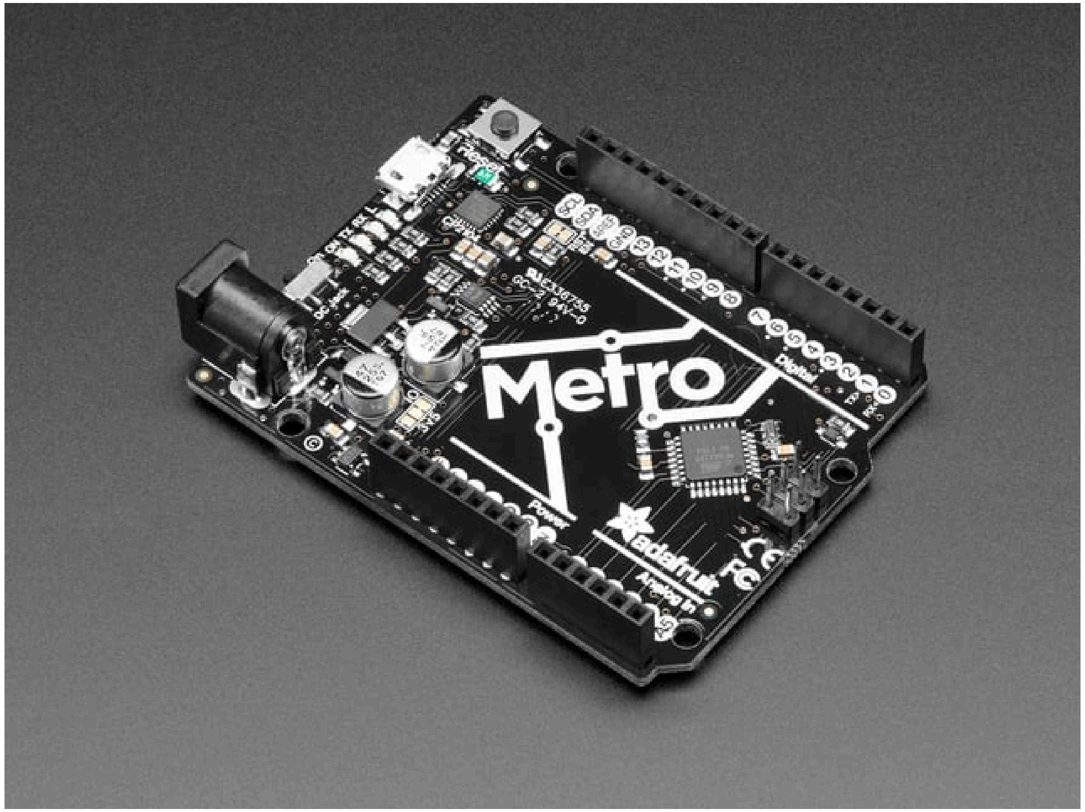
Text editor powered by tinymce.
Difficulty: Beginner
Guide Type: Project
Contributors: Bill Earl, lady ada
Categories: Adafruit Products
140 Saves
Featured Products view all



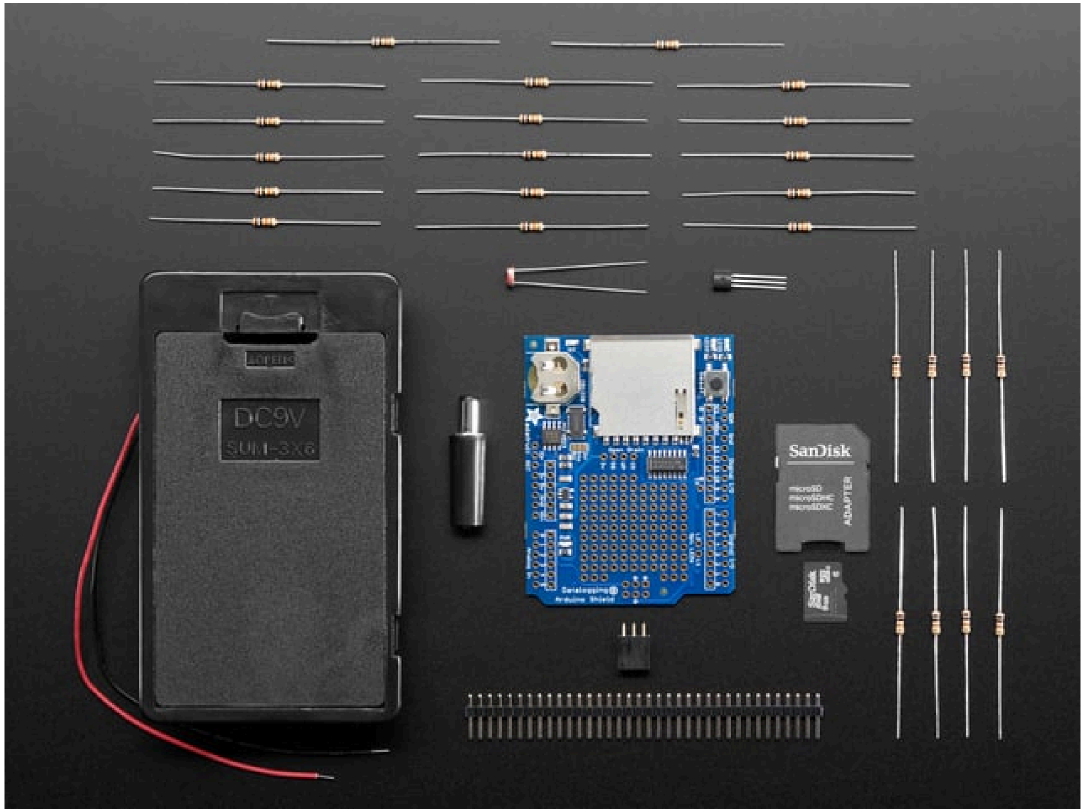
Adafruit Assembled Data Logging shield for Arduino
$13.95
Add to Cart



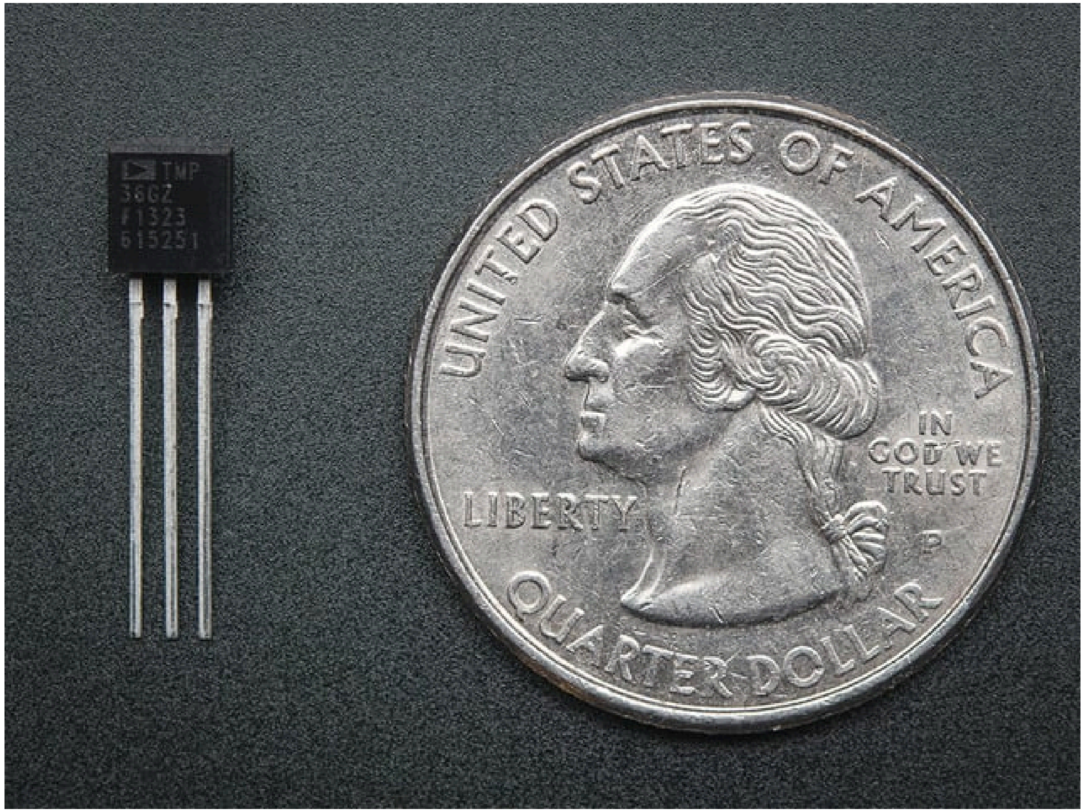
[Shield stacking headers for Arduino (R3 Compatible)]
$1.95
[Add to Cart]



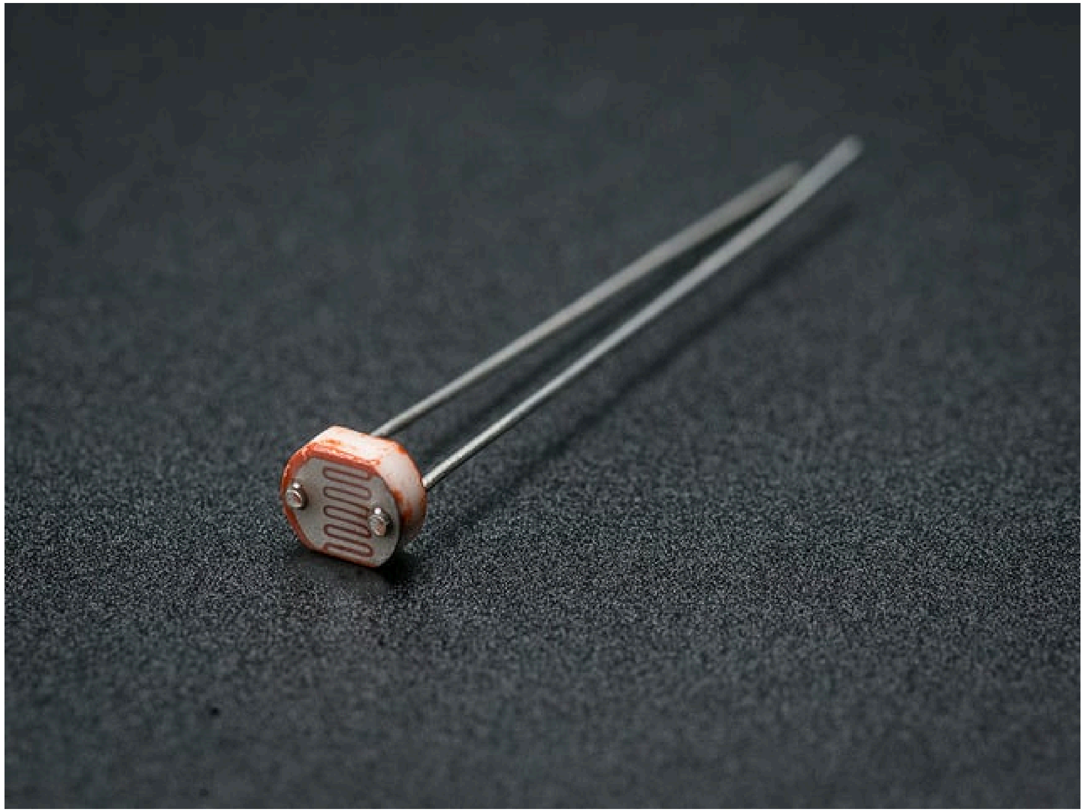
[Adafruit METRO 328 Fully Assembled - Arduino IDE compatible](https://www.adafruit.com/product/2488)
[Out of Stock](https://www.adafruit.com/product/2488)




[Light and temperature data-logger pack]
$27.50
[Add to Cart]




[TMP36 - Analog Temperature sensor]
$2.75
[Add to Cart]



[Photo cell (CdS photoresistor)](#)
[Out of Stock](#)


[6 x AA battery holder with 5.5mm/2.1mm plug]
$2.50
[Add to Cart]

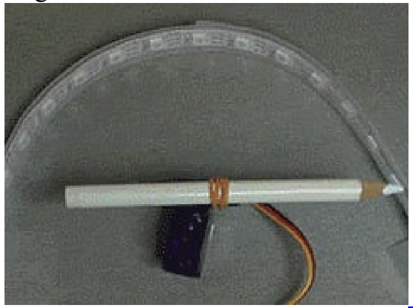
[9V battery holder with switch & 5.5mm/2.1mm plug](https://www.adafruit.com)
[Out of Stock](https://www.adafruit.com)




Related Guides

[Mini Thermal Receipt Printers]
By [Phillip Burgess]
102
Beginner
Updated
[Electronic Demon Costume]
[Electronic Demon Costume]
By [Phillip Burgess]
32
Beginner

[Smart Measuring Cup]
By [Tony DiCola]
17
Beginner

Arduino Lesson 4. Eight LEDs and a Shift Register
Arduino Lesson 4. Eight LEDs and a Shift Register
By Simon Monk
95
Beginner
2.8" TFT Touch Shield
2.8" TFT Touch Shield
By lady ada
20
Beginner
Portable Solar Charging Tracker
Portable Solar Charging Tracker
By lady ada
41
Intermediate
Adafruit Analog Accelerometer Breakouts
Adafruit Analog Accelerometer Breakouts
By Bill Earl
26
Intermediate
Arduino Lesson 5. The Serial Monitor
Arduino Lesson 5. The Serial Monitor
By Simon Monk
55
Beginner

Using NeoPixels and Servos Together
Using NeoPixels and Servos Together
By Phillip Burgess
67
Intermediate
Track Your Treats: Halloween Candy GPS Tracker
Track Your Treats: Halloween Candy GPS Tracker
By Tony DiCola
35
Beginner
RePaper eInk Development Board
RePaper eInk Development Board
By Bill Earl

5
Beginner
How to Choose a Microcontroller
How to Choose a Microcontroller
By mike stone
108
Beginner
Ladyada's Learn Arduino - Lesson #0
Ladyada's Learn Arduino - Lesson #0
By lady ada
196
Beginner
How to Build a Testing Jig
How to Build a Testing Jig
By Dano Wall
18
Intermediate
Trinket Audio Player
Trinket Audio Player
By Phillip Burgess
56
Beginner
×

**OUT OF STOCK NOTIFICATION**

YOUR NAME
YOUR EMAIL
NOTIFY ME
Search

# Search

**Categories**

No results for query

- «
- ‹
- 1
- ›
- »

- Contact Us
- Tech Support Forums

- FAQs
- Shipping & Returns
- Freebies
- Terms of Service
- Privacy & Legal
- Website Accessibility

- About Us
- Press
- Educators
- Distributors
- Jobs
- Gift Cards

"Collaborative production is simple: no one person can take credit for what gets created, and the project could not come into being without the participation of many"
Clay Shirky

A Minority and Woman-owned Business Enterprise (M/WBE)