SOLARLAPME

MIT AKKU UND LICHTSENSOR

betreut von Martin Fischbock
von Christa Carstensen und Annkathrin Lüthans

WAS WAR UNSER ZIEL?

Lampe

Solarpanel

Akku

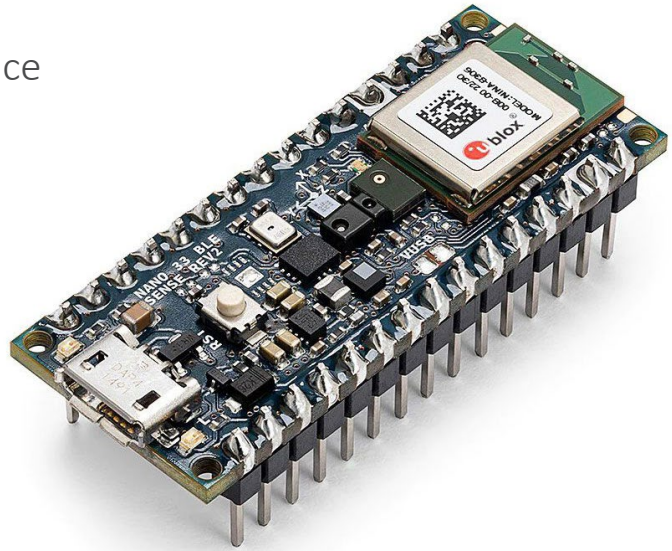
Lichtsensor

Funktionierender Schaltkreis, indem die **Lampe** über den **Akku**, der von dem **Solarpanel** mit Strom versorgt wird, geladen wird und ein **Lichtsensor** regelt, wann die Lampe an (wenn es zu x Prozent dunkel ist) und aus (wenn es zu x Prozent hell ist) geht.

ARDUINO

WAS HABEN WIR GENUTZ?

Arduino Nano 33	Board
Arduino IDE	Software zum Programmieren von Arduino-Mikrocontrollern
Arduino IoT Cloud	Visualisierung in einem Interface

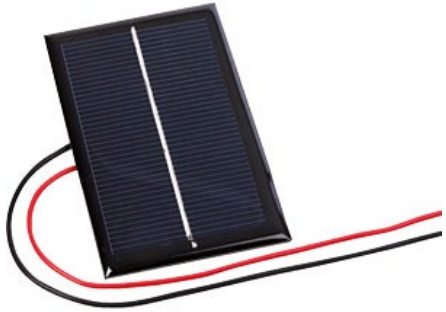


ARDUINO

ANALOGE TEILE

LED-Modul / Lampe

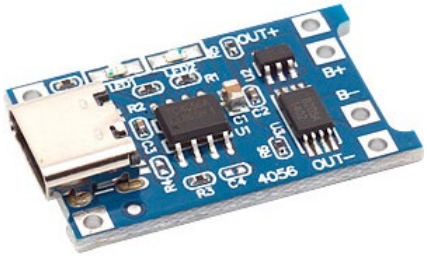
5V Solarzelle



Lichtsensor

3,7V Akku

Ladeplatine für 3,7V
Li-Akku



ARDUINO

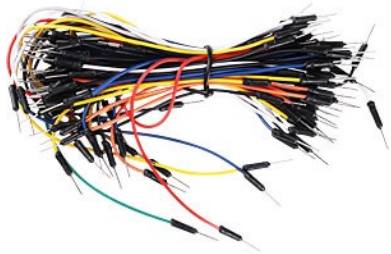
ANALOGE TEILE

Diode 1n4007

Wiederstände

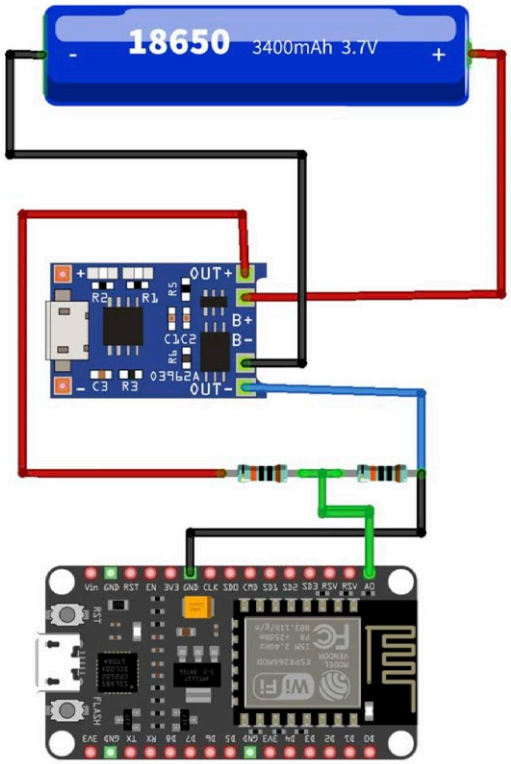
Relais V1.2

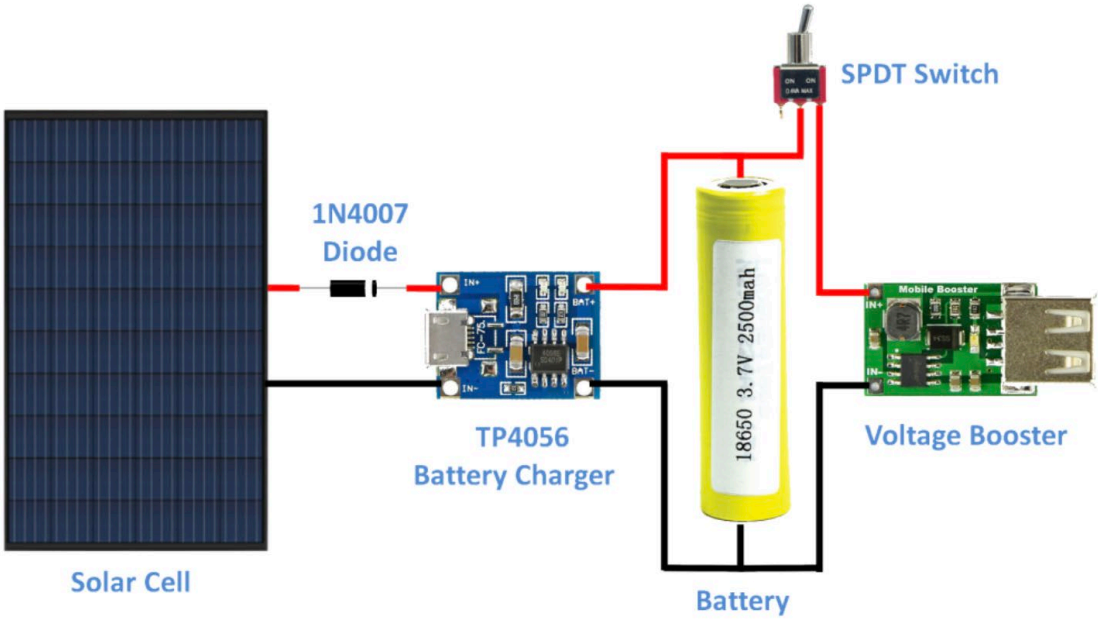
Verbindungskabel



ARDUINO

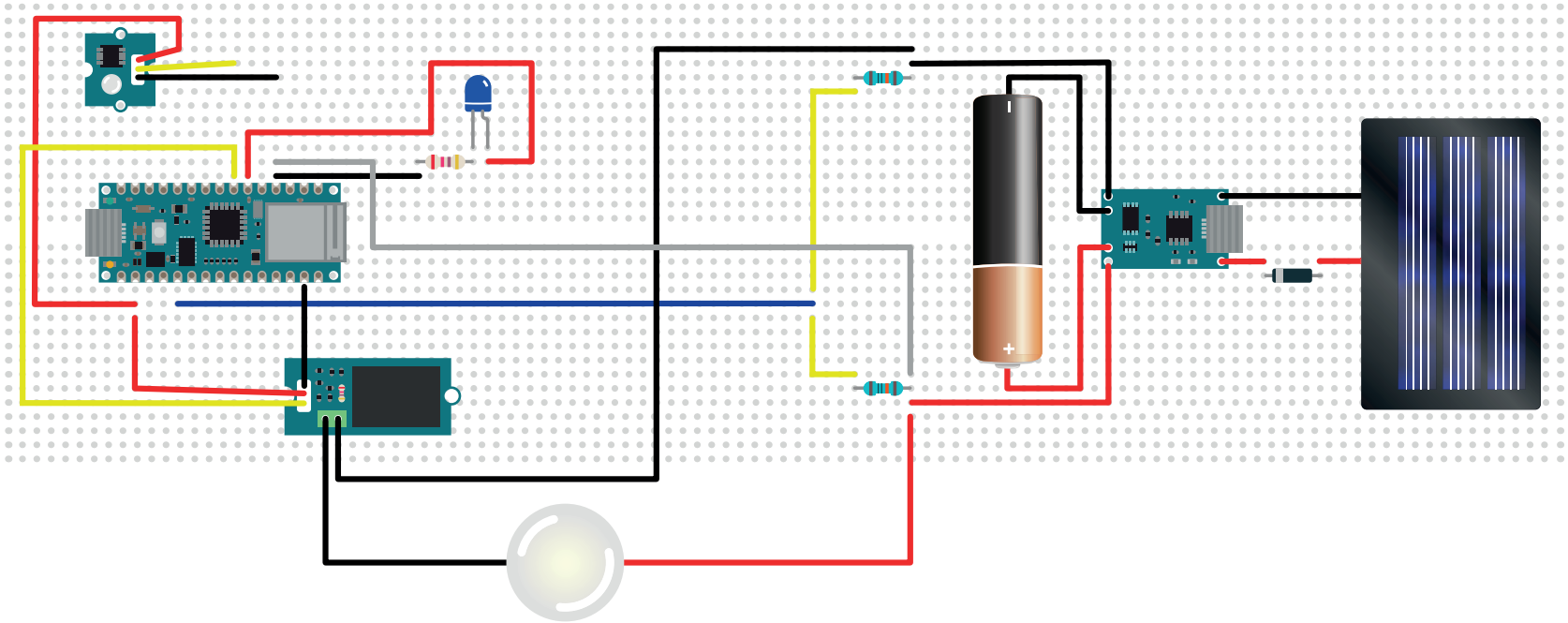
ANLEITUNGEN





UNSER AUFBAU

Skizze



CODE - Arduino IDE

```
/*
  Sketch generated by the Arduino IoT Cloud Thing “Untitled 2”
  https://create.arduino.cc/cloud/things/ad1e5f26-5c35-40dd-b6ff-51b3cab8e8be

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes are made to the Thing

  float voltage;
  int bat_percentage;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/

#include “arduino_secrets.h”

//enter your sensitive data in the secret tab
char ssid[] = SECRET_SSID;                    // your network SSID- add network name in arduino_secrets.h
char pass[] = SECRET_PASS;                    // your network password- add the network password in arduino_secrets.h

#include “thingProperties.h”

// define “status” varibale for network connection
int status = WL_IDLE_STATUS;
```

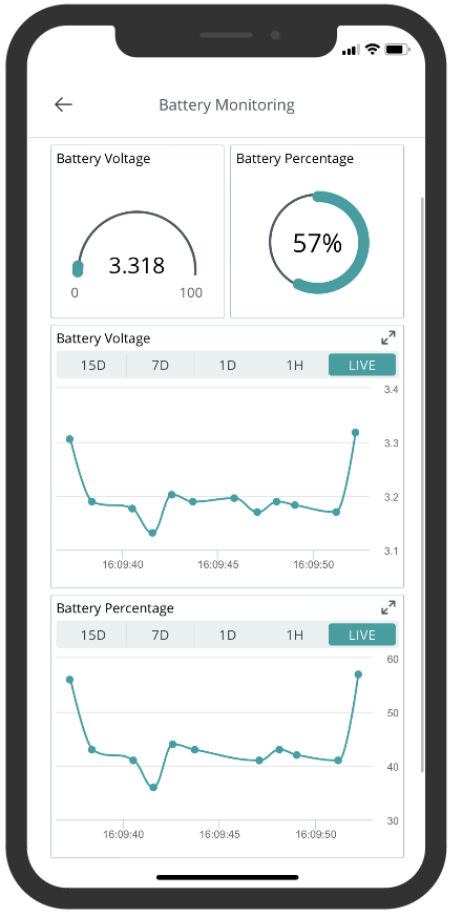
```
// constants for light sensor, LED and relay
const int LIGHT_SENSOR_PIN = A0;                // arduino pin connected to light sensor's pin
const int LED_PIN          = 3;                 // arduino pin (D3) connected to LED's pin
const int ANALOG_THRESHOLD = 500;               // threshold value of the measured light spectrum for switching on/off the LED/relay

// changing variables for light sensor, LED and relay
int analogValue; // variable transfering the light value

// constants & variables for battery monitor
int analogInPin  = A1;      // Analog input pin int sensorValue;
float calibration = 0.36;   // Check Battery voltage using multimeter & add/subtract the calibration value (to fix the tolerance of ±5% of the resistors)


void setup() {
 // Initialize serial and wait for port to open:
 Serial.begin(9600);
 // This delay gives the chance to wait for a serial monitor without blocking if none is found
 delay(1500);

 // attempt to connect to Wi-Fi network:
 while (status != WL_CONNECTED) {
  Serial.print("Attempting to connect to network: ");
  Serial.println(ssid);
  // connect to WPA/WPA2 network:
  status = WiFi.begin(ssid, pass);

  // wait 10 seconds for connection:
  delay(10000);
 }
```

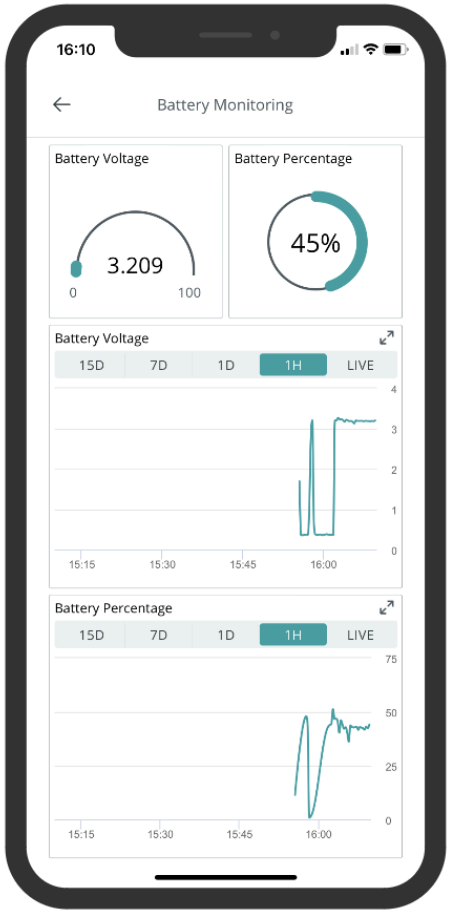
```
  // if connected, print out the data:
  Serial.println("You're connected to the network");
  Serial.println("----------------------------------------");

  // cloud variables defined in thingProperties.h
  initProperties();

  // connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
    The following function allows you to obtain more information
    related to the state of network and IoT Cloud connection and errors
    the higher number the more granular information you'll get.
    The default is 0 (only errors).
    Maximum is 4
  */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();


  pinMode(LED_PIN, OUTPUT);              // set arduino pin (LED_PIN = D3) to output mode for the LED
  pinMode(4, OUTPUT);                    // set arduino pin D4 to output mode for the relay

}

void loop() {
  ArduinoCloud.update();
  int analogValue = analogRead(A0);      // reads the input on analog pin A0 (value between 0 and 1023) for the light sensor

  Serial.print("Analog reading: ");      // print in serial monitor
  Serial.print(analogValue);             // print in serial monito: the raw analog reading
```

```
// interpretation of brightness/darkness
if (analogValue < ANALOG_THRESHOLD) {
  digitalWrite(LED_PIN, HIGH); // turn on LED
  digitalWrite(4, HIGH); // turn on relay
  delay(100);
} else {
    digitalWrite(LED_PIN, LOW);  // turn off LED
    digitalWrite(4, LOW); // turn off relay
}

// for serial monitor: threshholds qualitatively determined
if (analogValue < 10) {
  Serial.println("- Dark");
} else if (analogValue < 200) {
  Serial.println("- Dim");
} else if (analogValue < 500) {
  Serial.println("- Light");
} else if (analogValue < 800) {
  Serial.println("- Bright");
} else {
  Serial.println("- Very bright");
}

// battery monitor
ArduinoCloud.update();
sensorValue = analogRead(analogInPin);                  // A1 is defined as analogInPin
voltage = (((sensorValue * 3.3) / 1024) * 2 + calibration);      //multiply by two as voltage divider network is 100K & 100K resistor

bat_percentage = mapfloat(voltage, 2.8, 3.7, 0, 100);           //2.8V as battery cut off voltage & 3.7V as maximum voltage

 if (bat_percentage >= 100)
 {
   bat_percentage = 100;
 }
 if (bat_percentage <= 0)
 {
   bat_percentage = 1;
 }
```

```
  // battery monitoring printed in serial monitor
  Serial.print("Analog Value = ");
  Serial.print(sensorValue);
  Serial.print("\t Output Voltage = ");
  Serial.print(voltage);
  Serial.print("\t Battery Percentage = ");
  Serial.println(bat_percentage);
  delay(1000);
}

/*
  Since Voltage is READ_WRITE variable, onVoltageChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onVoltageChange()  {
}

/*
  Since BatPercentage is READ_WRITE variable, onBatPercentageChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onBatPercentageChange()  {
}

float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x- in_min) * (out_max- out_min) / (in_max- in_min) + out_min;
}
```

INTERFACE - Arduino Cloud





Darstellung in der Arduino App auf dem Handy

ARDUINO

ARDUINO