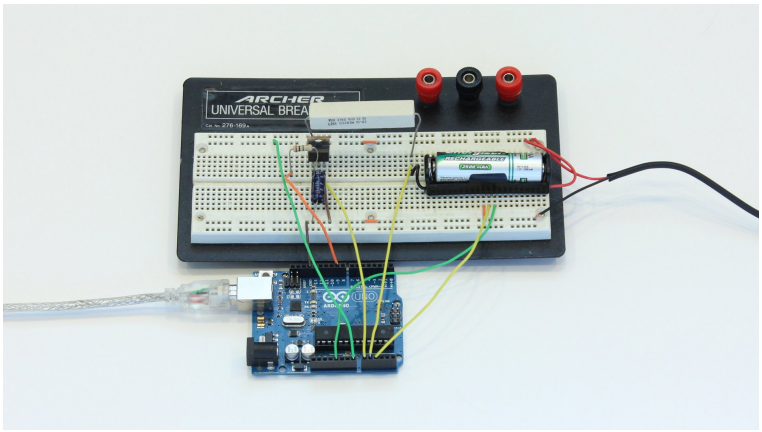
BLOC AUTONOME D'ECLAIRAGE DE SÉCURITÉ : BAES

Les blocs autonomes d'éclairage de sécurité (BAES), parfois appelés « Blocs de secours », sont des sources lumineuses d'évacuation destinées à éclairer et montrer l'emplacement des sorties dans différents types d'établissement lors d'évacuation d'urgence ou de défaillance de l'éclairage principal d'un bâtiment. Ils permettent de respecter la législation et les règles d'usages pour les locaux accueillant le public.

Ces blocs sont constitués d'un luminaire muni d'ampoules et d'une batterie, le tout permettant d'assurer un fonctionnement pendant une durée déterminée par les codes ou règlements locaux/nationaux. Cette durée est généralement d'une heure en Europe et de trente minutes en Amérique du Nord.

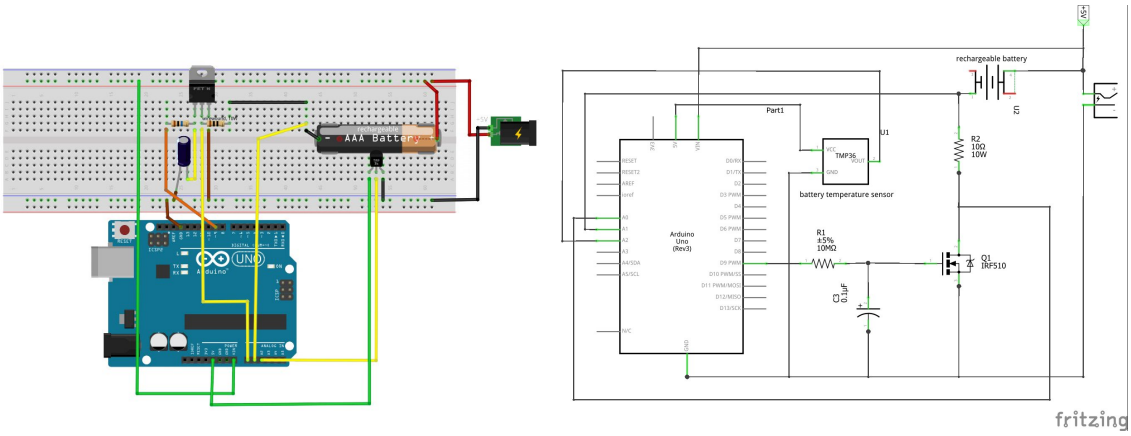
CHARGEUR DE BATTERIE



Composants

- Arduino Microcontroller
- 3 AA Battery Holder
- NiMH AA Battery
- 10 ohm Power Resistor (rated for at least 5 watts)
- 1 Mohm resistor
- 1 uF Capacitor
- IRF510 MOSFET
- TMP36 Temperature Sensor

- 5V Regulated Power Supply
- Prototyping Breadboard
- Jumper Wires



1 Réaliser le montage ci-dessus.
Constation professeur :

2 Implémenter le code ci-dessous.
Constation professeur :

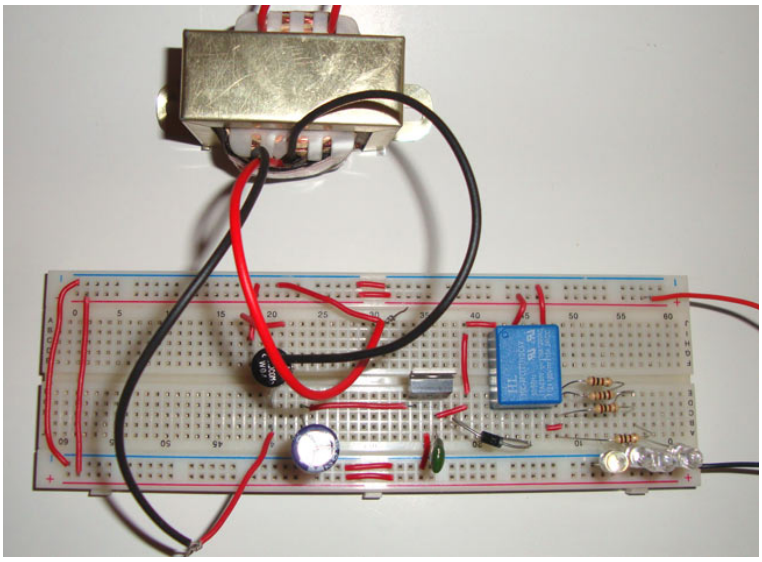
3 Tester le code ci-dessous.
Constation professeur :

```
int batteryCapacity = 2500;     //capacity rating of battery in mAh
float resistance = 10.0;     //measured resistance of the power resistor
int cutoffVoltage = 1600;     //maximum battery voltage (in mV) that should not be exceeded
float cutoffTemperatureC = 35;     //maximum battery temperature that should not be exceeded (in degrees C)
//float cutoffTemperatureF = 95;     //maximum battery temperature that should not be exceeded (in degrees F)
long cutoffTime = 46800000;     //maximum charge time of 13 hours that should not be exceeded

int outputPin = 9;     // Output signal wire connected to digital pin 9
int outputValue = 150;     //value of PWM output signal

int analogPinOne = 0;     //first voltage probe connected to analog pin 1
float valueProbeOne = 0;     //variable to store the value of analogPinOne
float voltageProbeOne = 0;     //calculated voltage at analogPinOne

int analogPinTwo = 1;     //second voltage probe connected to analog pin 2
float valueProbeTwo = 0;     //variable to store the value of analogPinTwo
float voltageProbeTwo = 0;     //calculated voltage at analogPinTwo

int analogPinThree = 2;     //third voltage probe connected to analog pin 2
float valueProbeThree = 0;     //variable to store the value of analogPinThree
float tmp36Voltage = 0;     //calculated voltage at analogPinThree
float temperatureC = 0;     //calculated temperature of probe in degrees C
//float temperatureF = 0;     //calculated temperature of probe in degrees F

float voltageDifference = 0;     //difference in voltage between analogPinOne and analogPinTwo
float batteryVoltage = 0;     //calculated voltage of battery
float current = 0;     //calculated current through the load (in mA)
float targetCurrent = batteryCapacity / 10;     //target output current (in mA) set at C/10 or 1/10 of the battery capacity per ho
float currentError = 0;     //difference between target current and actual current (in mA)



void setup()
{
  Serial.begin(9600);     //  setup serial
  pinMode(outputPin, OUTPUT);     // sets the pin as output
}
```

```
void loop()
{

  analogWrite(outputPin, outputValue);  //Write output value to output pin

  Serial.print("Output: ");     //display output values for monitoring with a computer
  Serial.println(outputValue);

  valueProbeOne = analogRead(analogPinOne);    // read the input value at probe one
  voltageProbeOne = (valueProbeOne*5000)/1023;     //calculate voltage at probe one in milliVolts
  Serial.print("Voltage Probe One (mV): ");     //display voltage at probe one
  Serial.println(voltageProbeOne);

  valueProbeTwo = analogRead(analogPinTwo);    // read the input value at probe two
  voltageProbeTwo = (valueProbeTwo*5000)/1023;     //calculate voltage at probe two in milliVolts
  Serial.print("Voltage Probe Two (mV): ");     //display voltage at probe two
  Serial.println(voltageProbeTwo);

  batteryVoltage = 5000 - voltageProbeTwo;     //calculate battery voltage
  Serial.print("Battery Voltage (mV): ");     //display battery voltage
  Serial.println(batteryVoltage);

  current = (voltageProbeTwo - voltageProbeOne) / resistance;     //calculate charge current
  Serial.print("Target Current (mA): ");     //display target current
  Serial.println(targetCurrent);
  Serial.print("Battery Current (mA): ");     //display actual current
  Serial.println(current);

  currentError = targetCurrent - current;     //difference between target current and measured current
  Serial.print("Current Error  (mA): ");     //display current error
  Serial.println(currentError);

  valueProbeThree = analogRead(analogPinThree);    // read the input value at probe three
  tmp36Voltage = valueProbeThree * 5.0;     // converting that reading to voltage
  tmp36Voltage /= 1024.0;

  temperatureC = (tmp36Voltage - 0.5) * 100 ;     //converting from 10 mv per degree wit 500 mV offset to degrees ((voltage - 500m
  Serial.print("Temperature (degrees C) ");     //display the temperature in degrees C
  Serial.println(temperatureC);

 /*
  temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;     //convert to Fahrenheit
  Serial.print("Temperature (degrees F) ");
  Serial.println(temperatureF);
 */

  Serial.println();     //extra spaces to make debugging data easier to read
  Serial.println();



  if(abs(currentError) > 10)     //if output error is large enough, adjust output
   {
    outputValue = outputValue + currentError / 10;

    if(outputValue < 1)    //output can never go below 0
     {
      outputValue = 0;
     }

    if(outputValue > 254)     //output can never go above 255
     {
      outputValue = 255;
     }

    analogWrite(outputPin, outputValue);     //write the new output value
   }


  if(temperatureC > cutoffTemperatureC)     //stop charging if the battery temperature exceeds the safety threshold
   {
    outputValue = 0;
```

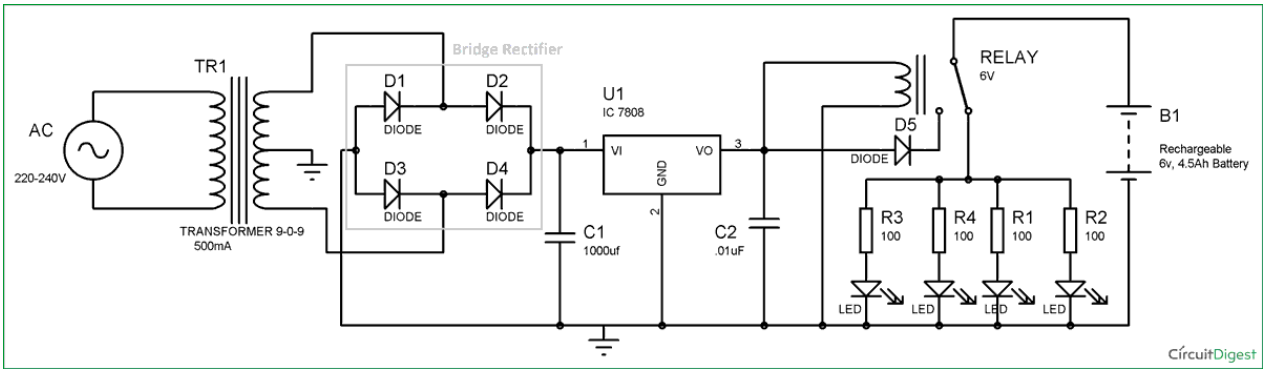
```
    Serial.print("Max Temperature Exceeded");
   }

  /*
  if(temperatureF > cutoffTemperatureF)     //stop charging if the battery temperature exceeds the safety threshold
   {
    outputValue = 0;
   }
   */

   if(batteryVoltage > cutoffVoltage)     //stop charging if the battery voltage exceeds the safety threshold
   {
    outputValue = 0;
    Serial.print("Max Voltage Exceeded");
   }

   if(millis() > cutoffTime)     //stop charging if the charge time threshold
   {
    outputValue = 0;
    Serial.print("Max Charge Time Exceeded");
   }

   delay(10000);     //delay 10 seconds before next iteration
}
```

ALIMENTATION D'URGENCE



Composants

- Transformer- 9-0-9 500mA
- Bridge rectifier
- Diode- 1N4007
- IC 7808 voltage regulator
- Capacitor 1000uF, 0.01uF
- Relay- 6v
- Resistors- 100 ohm
- LEDs- Ultra bright white LED
- Rechargeable 6v, 4.5Ah Battery



4 Connecter juste le transformateur du montage ci-dessus. Mesurer la tension.
Constation professeur :

5 Connecter le transformateur et le pont de diodes du montage ci-dessus. Mesurer la tension.
Constation professeur :

6 Connecter le transformateur, le pont de diodes et le regulateur du montage ci-dessus. Mesurer la tension.
Constation professeur :

7 Connecter le transformateur, le pont de diodes, le regulateur et le relai avec sa diode de protection du montage ci-dessus. Mesurer la tension.
Constation professeur :

BAES

8 Représenter un diagramme de fonctionnement combinant les deux circuits précédents et réalisant un BAES.
Constation professeur :

9 Connecter les deux parties du circuit et réaliser le BAES.
Constation professeur :