Arduino Controlled Battery Charger

I. Summary

In this project we will learn to use an Arduino and an attached charging circuit to control the charging of NiMH rechargeable batteries. Rechargeable batteries are a great way to power your portable electronics. They can save a lot of money when properly recycled. But it's much more fun to build one for yourself. So, here is how to build an Arduino controlled battery charger. Each type of battery uses a different chemical process to make it work. As a result, each type of battery needs to be charged differently. We are going to focus on the most common type of AA rechargeable battery, Nickel-Metal Hydride (NiMH).

II. Objectives

- To teach students how to construct an arduino controlled battery charger.

III. Industry-Based Applications

The industry focuses on durability and reliability. Industrial batteries are bulkier than those used in consumer products, but achieve a longer service life. Batteries are electro-chemical devices that convert higher-level active materials into an alternate state during discharge. The speed of such transactions determines the load on a battery. Also referred to as concentration polarization, the nickel and lithium-based batteries are superior to lead-based batteries in reaction speed. This attribute reflects in good load characteristics. Rechargeable batteries include all types of portable devices automobile starters, portable consumer devices, light vehicles (such as motorized wheelchairs, golf carts, electric bicycles, and electric forklifts), tools, uninterruptible power supplies, and battery storage power stations.

Project Methodology

There are many different ways to charge a NiMH battery. The method that is used will depend mostly on how fast you want to charge your battery. The Charge rate (or C-rate) is measured relative to the capacity of the battery. If your battery has a capacity of 2500mAh and you charge it with a current of 2500 mA, then you are charging it at a rate of 1C and If you charge it with a current of 250 mA, then you are charging it at a rate of C/10.

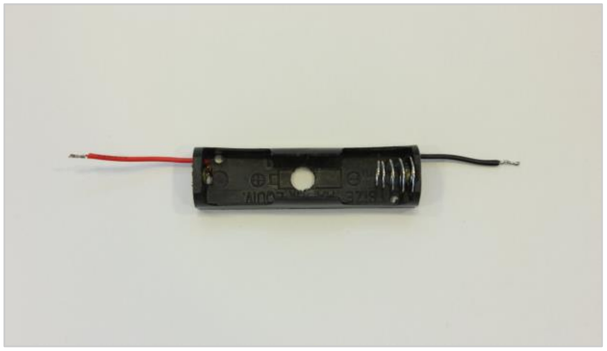
When charging your battery quickly (at a rate higher C/10), you need to carefully monitor the battery’s voltage and temperature to make sure that you don’t overcharge it. This can seriously damage your battery. Because of this, slow charging methods are generally considered to be safer and will help maximize battery life.

The circuit design for this charger is a basic Arduino controlled power supply. The circuit is powered by a 5-volt regulated voltage source such as an AC adapter or an ATX computer power supply, we will be using an AC/DC power supply. Most USB ports would not be appropriate for this project because of the current limitations. The 5V source charges the battery through a 10 ohm power resistor and a power MOSFET.

Components:

- 1x Arduino Microcontroller

- 1x AA Battery Holder

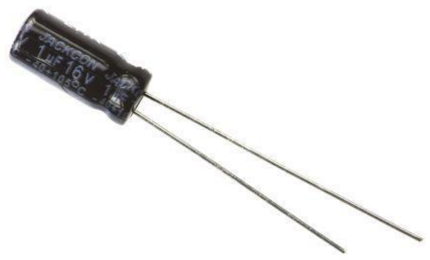


- 1x NiMH AA Battery

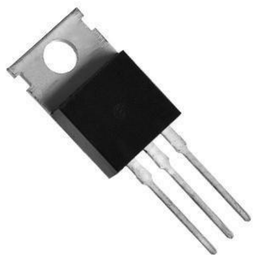
- 1x 10 ohm Power Resistor (rated for at least 5 watts)

- 1x 1 Mohm resistor

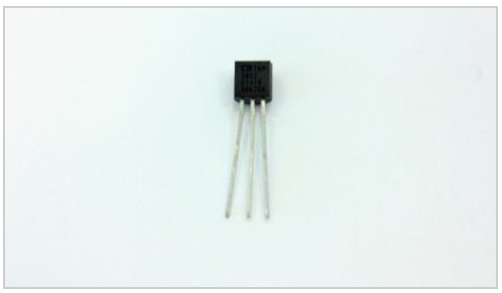
- 1x 1 µF Capacitor



- 1x IRF510 MOSFET

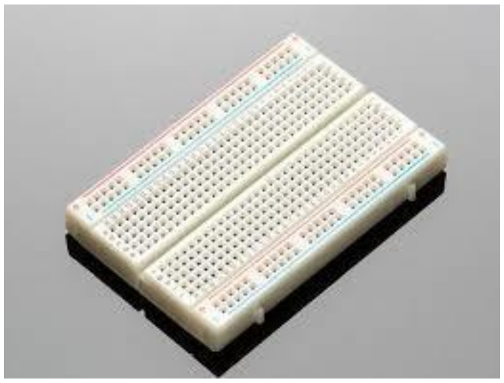


- 1x TMP36 Temperature Sensor



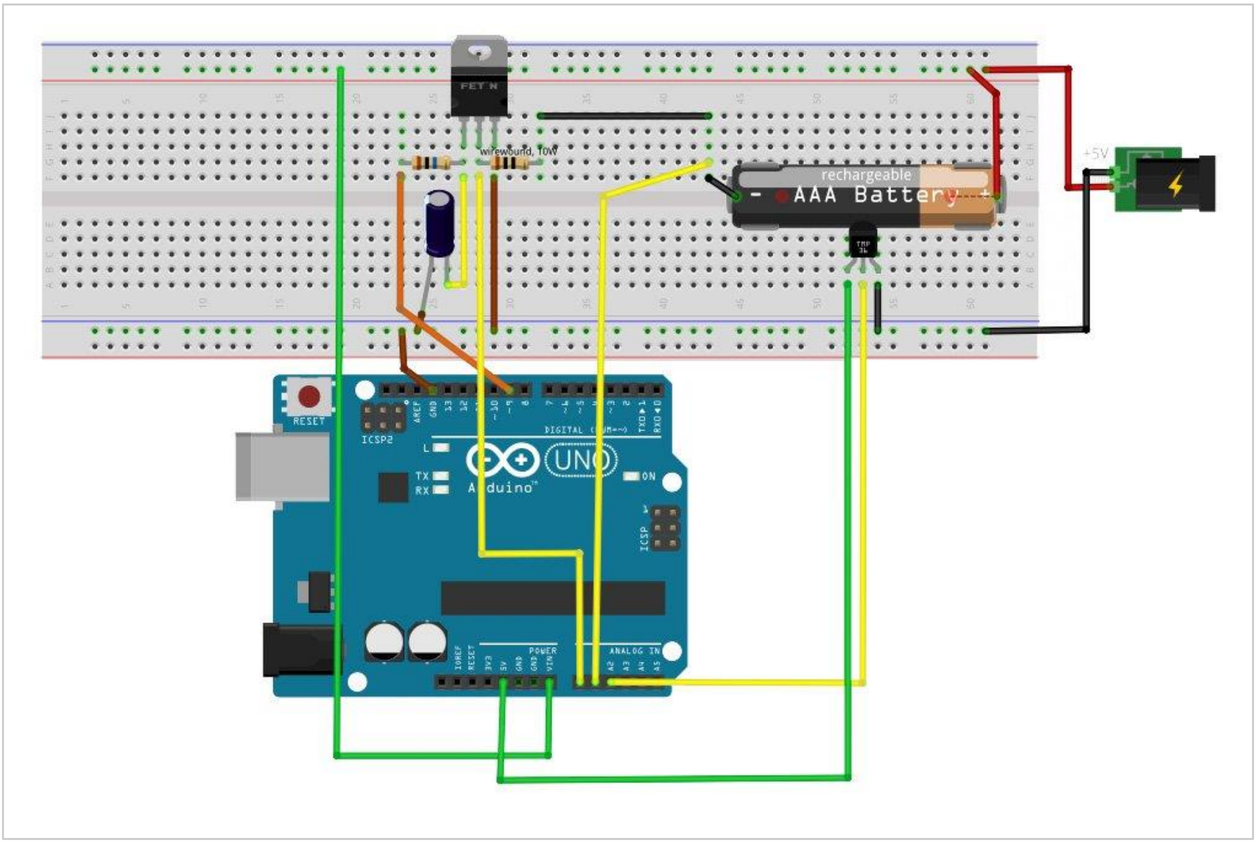
- 1x 5V Regulated Power Supply

- 1x Prototyping Breadboard

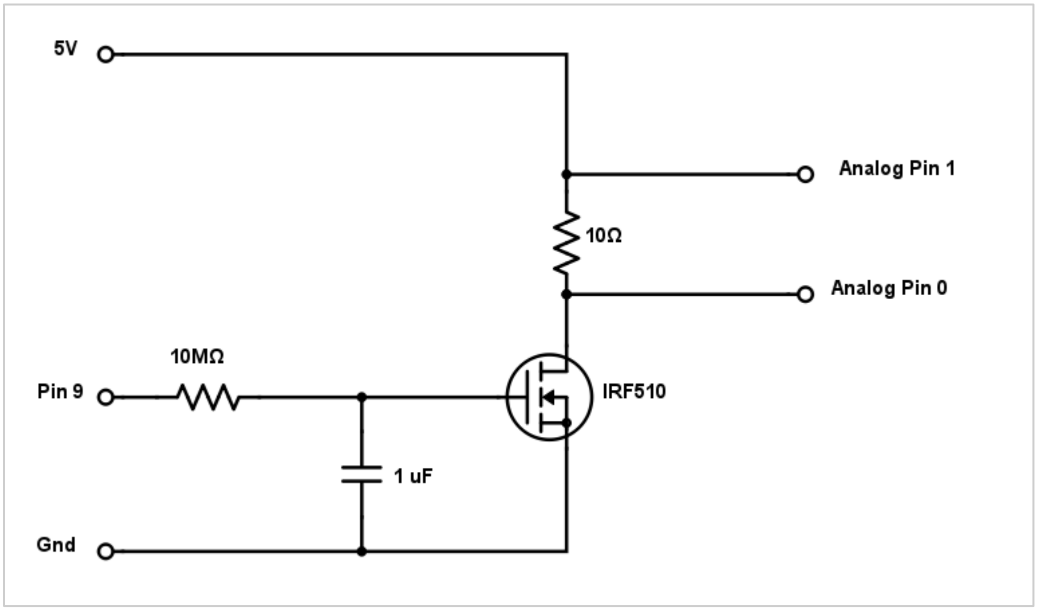


- Jumper Wires

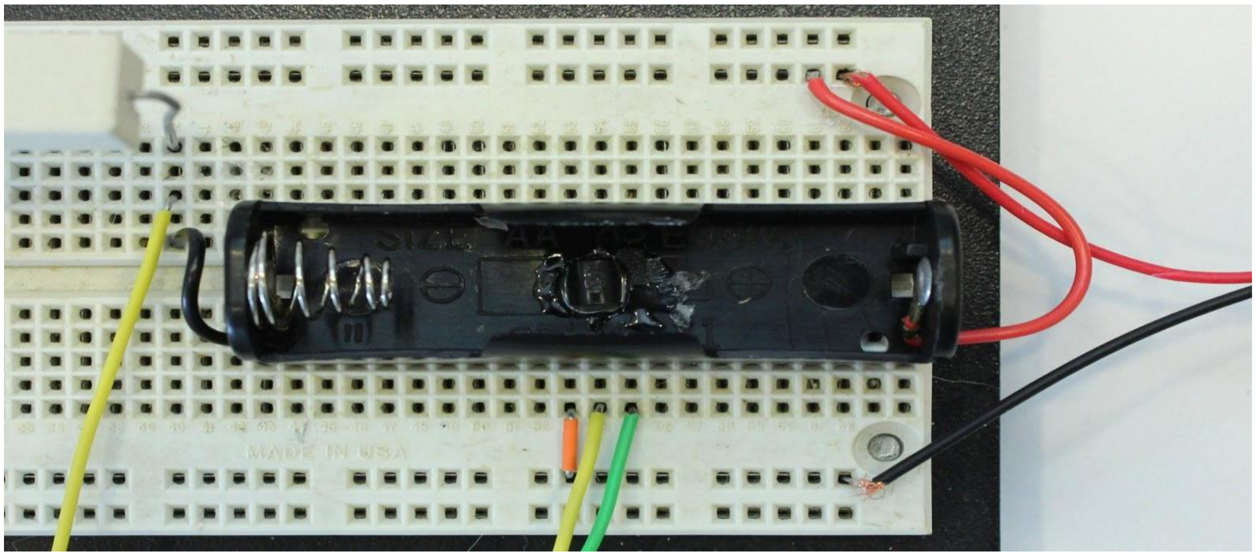
Project Procedures:



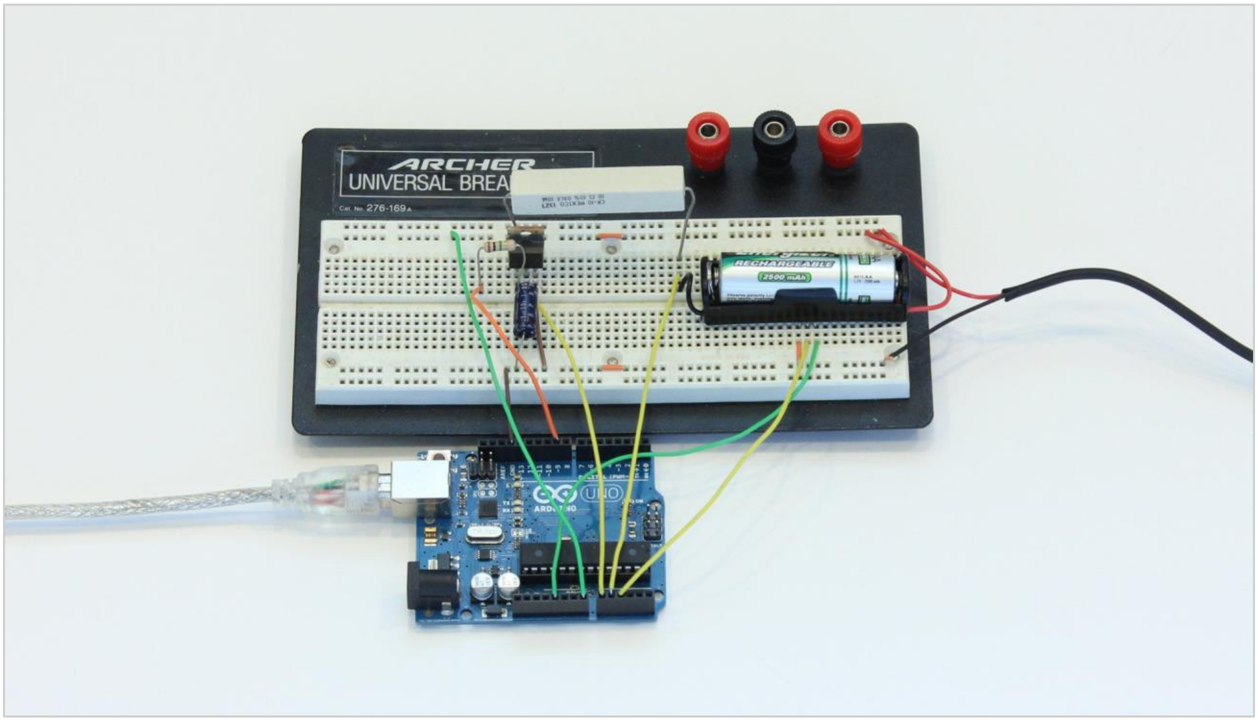
The MOSFET sets how much current is allowed to flow into the battery. The resistor is included as an easy way to monitor the current. This is done by connecting each terminal to analog input pins on the Arduino and measuring the voltage on each side. The MOSFET is controlled by a PWM output pin on the Arduino connected with wires. The pulses of the pulse width modulation signal are smoothed out into a steady voltage signal by a 1M resistor and a 1 µF capacitor. This circuit allows the Arduino to monitor and control the current flowing into the battery.



The AA battery holder after being placed on the breadboard.



As an extra precaution, I included a TMP36 temperature sensor to monitor the temperature of a battery. This sensor outputs a signal voltage that directly corresponds to the temperature. So it doesn't require calibration or balancing like a thermistor does. The sensor is mounted in place by drilling a hole in the back of the battery housing and gluing the sensor in so that it sits against the side of the battery when installed. The pins of the sensor are then connected to 5V, GND, and an analog input pin on the Arduino.



After all connection has been made the project should look similar to what to the figure above.

IV. Appendix

Code

```
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 hour
```

```
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 - 500mV) times 100)
```

```
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
}
```

V. References

1. Create an Arduino Controlled Battery Charger - Projects. (n.d.). Retrieved May 10, 2020, from https://www.allaboutcircuits.com/projects/create-an-arduino-controlled-battery-charger/
2. Charge Batteries with an Arduino. (n.d.). Retrieved May 14, 2020, from https://www.digikey.com/en/maker/projects/charge-batteries-with-an-arduino/b9d6ff74ecea437da8126d097d680d56
3. Industrial Applications. (n.d.). Retrieved from https://batteryuniversity.com/learn/archive/industrial_applications