Detector de raios para Arduino

Neste tutorial, vamos construir um detector de raios usando um Arduino Uno, alguns resistores e alguns fios de jumpers. A maioria dos detectores de relâmpago muitas vezes custam muito para o hobbyista normal, no entanto isso não significa que não se pode desfrutar da detecção de raios e da física por trás dele.

Neste tutorial, usando um circuito surpreendentemente simples, seremos capazes de detectar relâmpagos de cerca de 10-20 km de distância, o que é para dizer ao menos impressionante.

O objetivo é construir um circuito simples para detectar relâmpagos com um Arduino e produzir resultados significativos.

Quando um raio atinge, uma enorme quantidade de energia é liberada em diferentes formas. Os mais óbvios são a luz e o som, sendo este último um subproduto da taxa de aumento da temperatura das partículas imediatas que rodeiam o raio, o que então provoca o som. Mas isso não é tudo.

Os relâmpagos emitem grande quantidade de radiação eletromagnética na faixa VLF (Very Low Frequency) e LF (Low Frequency), normalmente variando de 3kHz a 300kHz. VLF e LF são semelhantes às ondas de luz, ondas WiFi e também as ondas do forno de microondas, mas com a diferença de operar em frequências mais baixas. Por exemplo, WiFi normalmente opera em torno de 2,4 GHz, que é 2,4 bilhões de oscilações por segundo. VLF e LF opera em frequências mais baixas, e com um Arduino podemos capturar frequências em torno de 7kHz. As vantagens em usar este tipo de radiação para a detecção do relâmpago é que normalmente nada se vê além de rajadas grandes como visto nos relâmpagos, em torno desta frequência; E sendo uma onda eletromagnética que viaja à velocidade da luz, o que significa que o sensor irá detectar relâmpagos como eles acontecem (alguns microssegundos depois).

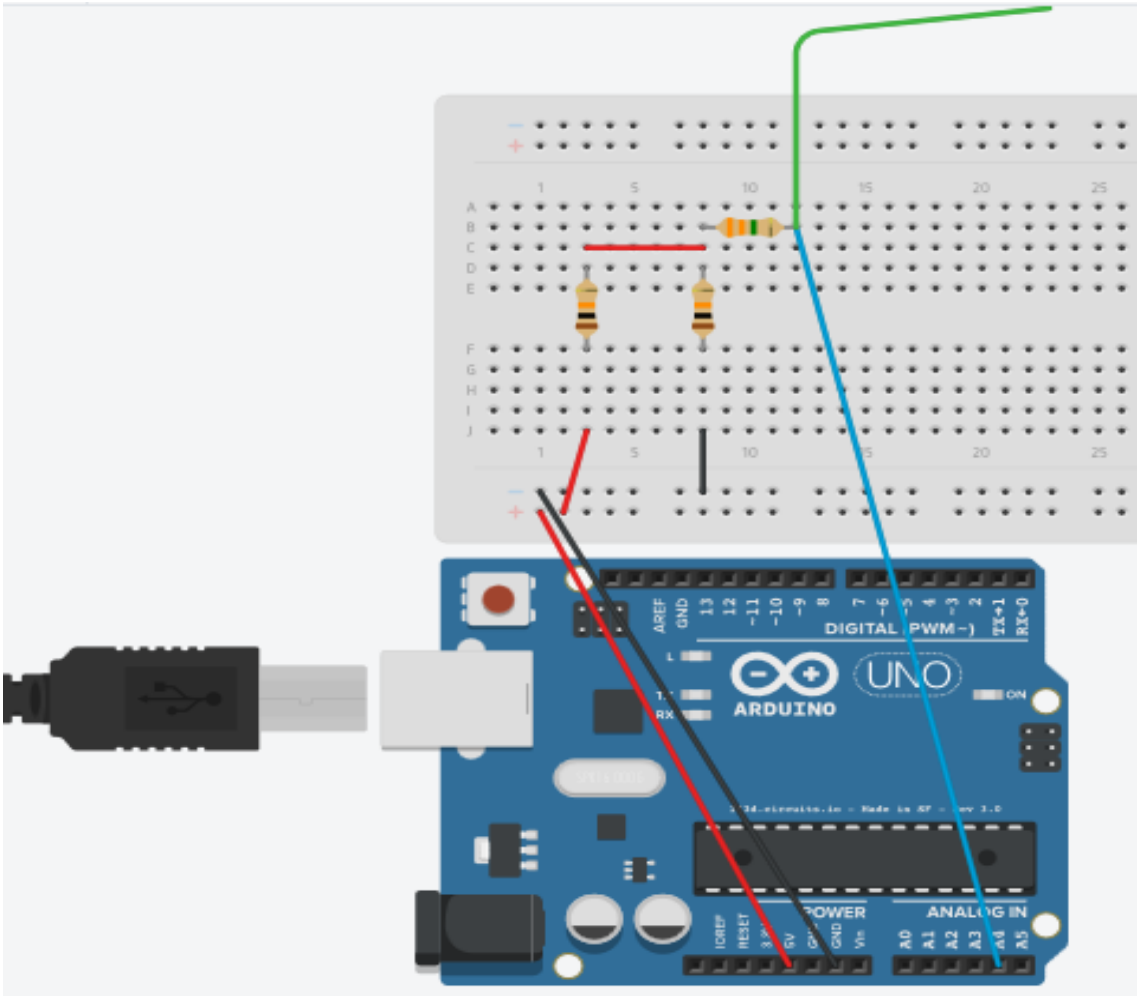
Nosso pequeno Arduino terá uma antena (uma espécie de), um pedaço de fio que vai pegar flutuações no espectro eletromagnético especificamente em torno do 7-9kHz. Estas flutuações induzem uma pequena tensão + ve ou -ve no fio. Podemos usar essas flutuações usando os pinos analógicos do Arduino.

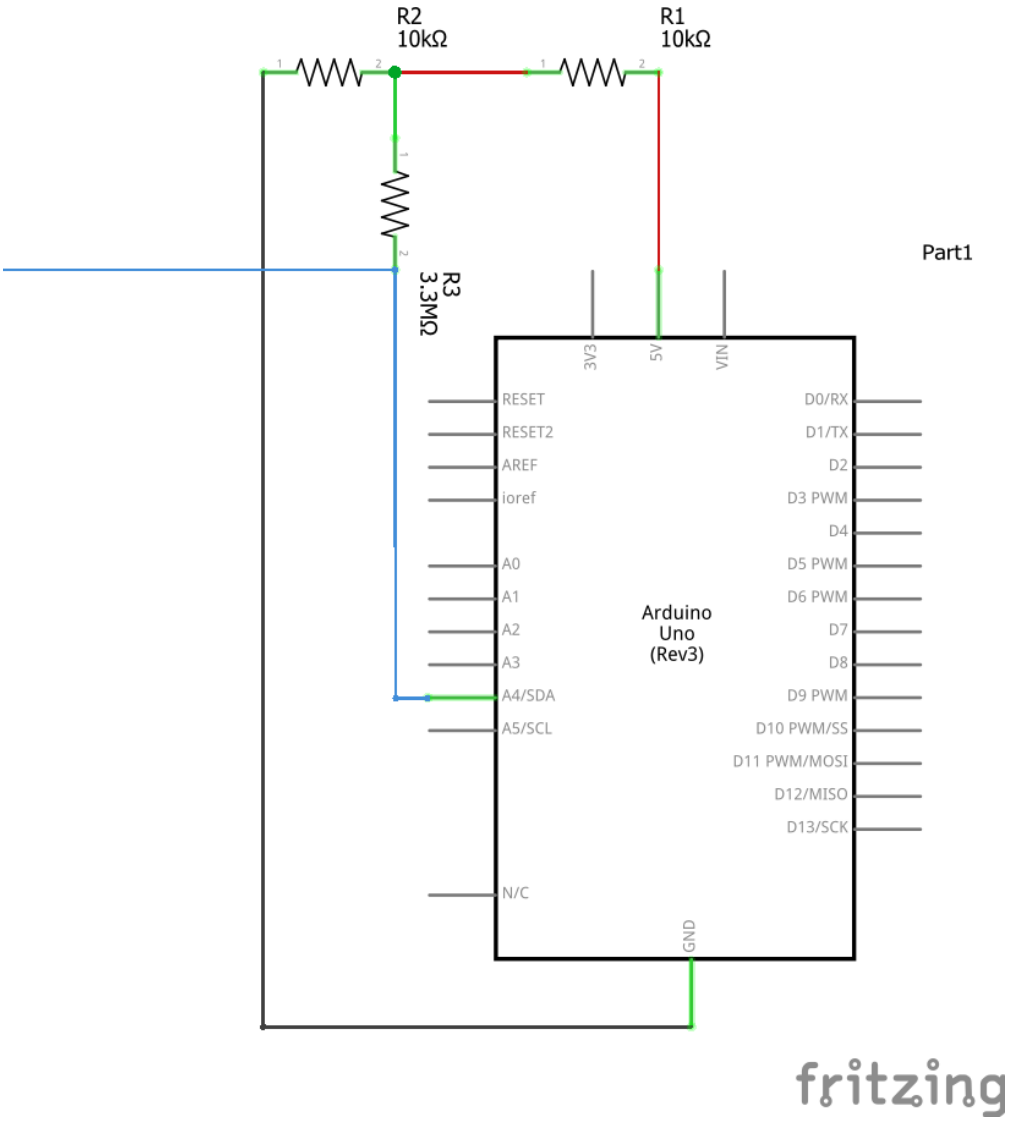
Pré-requisitos

- Resistor de 2x10k Ohms
- 1x 3,3M Ohms Resistor
- 4x Jumper Wires
- 1x Arduino (estou usando o Uno, mas qualquer outro funcionará, desde que ele possa operar em 16Mhz)
- Breadboard por simplicidade

Como você já deve saber, os pinos na placa Arduino permitem tensões entre 0v e 5v, qualquer coisa abaixo de 0v e acima de 5v não será lido, portanto, os dados serão perdidos. Mais importante ainda, tensões abaixo de 0v potencialmente danificar o pino. Isso criará um pequeno problema para nós porque as tensões produzidas no fio flutuam abaixo e acima de 0v. Para resolver este problema nós ajustamos a tensão do pino no meio da escala 5v, em 2.5v e este será realizado usando um truque pequeno, um divisor da tensão. Ao fazê-lo, vamos definir o pino para um constante 2,5v e as flutuações de tensão terá uma origem de 2,5v, portanto, nenhum dano ou perda de dados.

Diagrama de circuito do detector de raios





Detector de relâmpagos

O circuito é bastante direto, temos 2x 10k Ohm resistores em série de 5v (fio vermelho) para GND (fio preto), este é basicamente o divisor de tensão. Então um resistor de 3,3M Ohm (MegaOhm) é conectado entre o resistor de 2x 10k Ohm. Em série com o resistor de 3,3M Ohm anexar um fio para pino A4 (fio azul), isso vai nos dar exatamente 2.5v no pino

A4. Em seguida, anexar um fio que irá atuar como uma antena (fio verde) de cerca de 6-8 polegadas de comprimento. Isso deve ser conectado a partir de uma extremidade apenas como mostrado acima.

Esboço

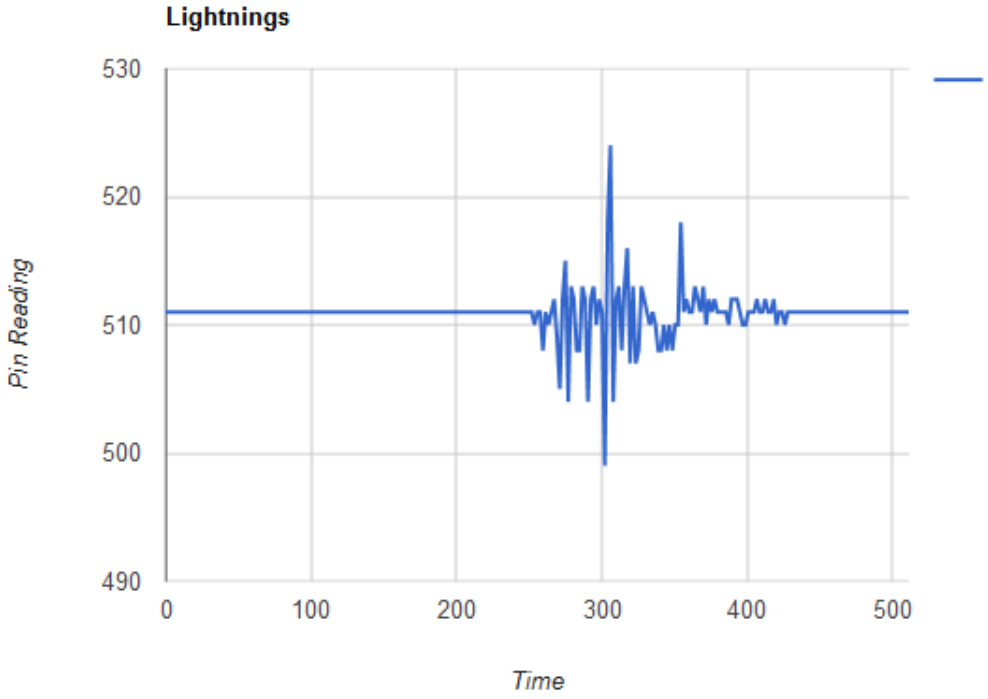
Aqui vem a parte mais difícil de explicar. Como mencionado acima, a frequência que precisamos pegar a partir dos relâmpagos, é de cerca de 7kHz e para ler uma onda semi-decente a taxa de amostragem tem que ser 4x tanto, dando-nos 4 leituras por comprimento de onda. Ou seja, 28.000 amostras por segundo.

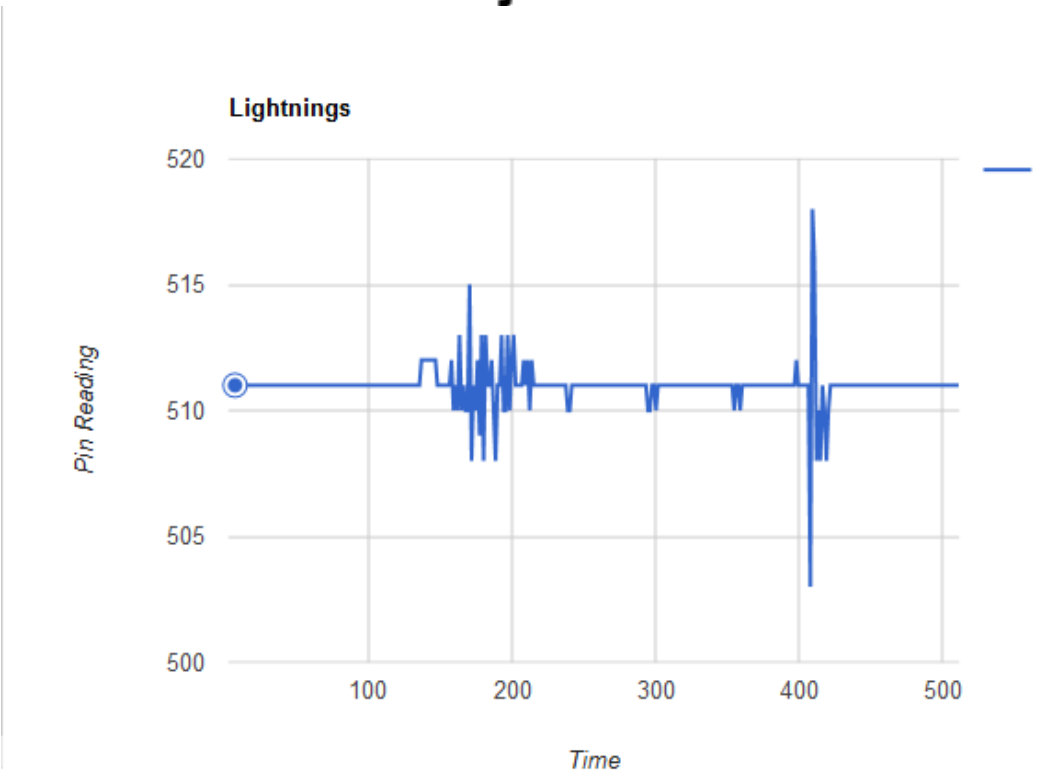
Os pinos analógicos do Arduino só podem nos dar 9.600 amostras por segundo. Com essa taxa de amostragem, só seremos capazes de capturar ondas a 2kHz ou um pouco mais, o que está longe de ser bom. Graças ao chip ATMEGA, ele pode ser configurado para acelerar o processo ADC por um determinado fator, mantendo uma boa resolução. Isso é chamado de pré-escalador, e pode ser configurado através de código. Há um número de fatores de divisão do preditor, mas usaremos o fator 16 que, em teoria, nos dará uma taxa de amostragem de 77kHz. Na prática, qualquer forma de cálculo irá diminuir esta taxa de amostragem, assim, eu só era capaz de obter cerca de 46kHz, que ainda é muito bom para este projeto.

Movendo-se para a frente, o esboço usa uma matriz de 512 bytes para armazenar válvulas de tensão do pino A4. Ele constantemente lê o valor do pino e escreve-o para o próximo local na matriz. Assim que um raio é detectado toda a matriz é enviada através da porta serial. Isso pode ser plotado no plotter gráfico no Arduino IDE ou talvez enviado para outro Arduino ou ESP8266 para publicar os dados on-line. É provavelmente melhor monitorá-lo através do IDE Arduino no início, por isso, se houver algumas falhas, eles podem ser então abordados lá.

Resultados

. A seguir estão alguns resultados.





Pegue o código fonte do Github: https://github.com/klauscam/Arduino-Lightning-Detector

Por favor, sinta-se livre para comentar abaixo se você precisar de mais esclarecimentos.

Atualização 2

Para usar o plotter serial IDE do Arduino, substitua esta função

```
voidsendData()

{

 // Serial.print(">>>");
```

```
//                                  Serial.println(batchStarted);


for (inti=0;i<data;i++){

Serial.println(storage[i]);

 }

 // Serial.print("<<<");

 // Serial.println(batchEnded);

 // Serial.println("END");


toSend=false;

}
```

Atualização 3

Note que o dispositivo é muito sensível às flutuações de EMF. Isso inclui fontes de alimentação CA. Coloque o dispositivo longe de qualquer fonte de alimentação CA e de qualquer tomada. Também observamos que ele se comporta anormal quando conectado a um laptop durante o carregamento. Recomendamos a utilização de um portátil com baterias apenas para um desempenho ótimo.

Fonte: http://runtimeprojects.com/2016/02/a-lightning-detector-for-arduino/