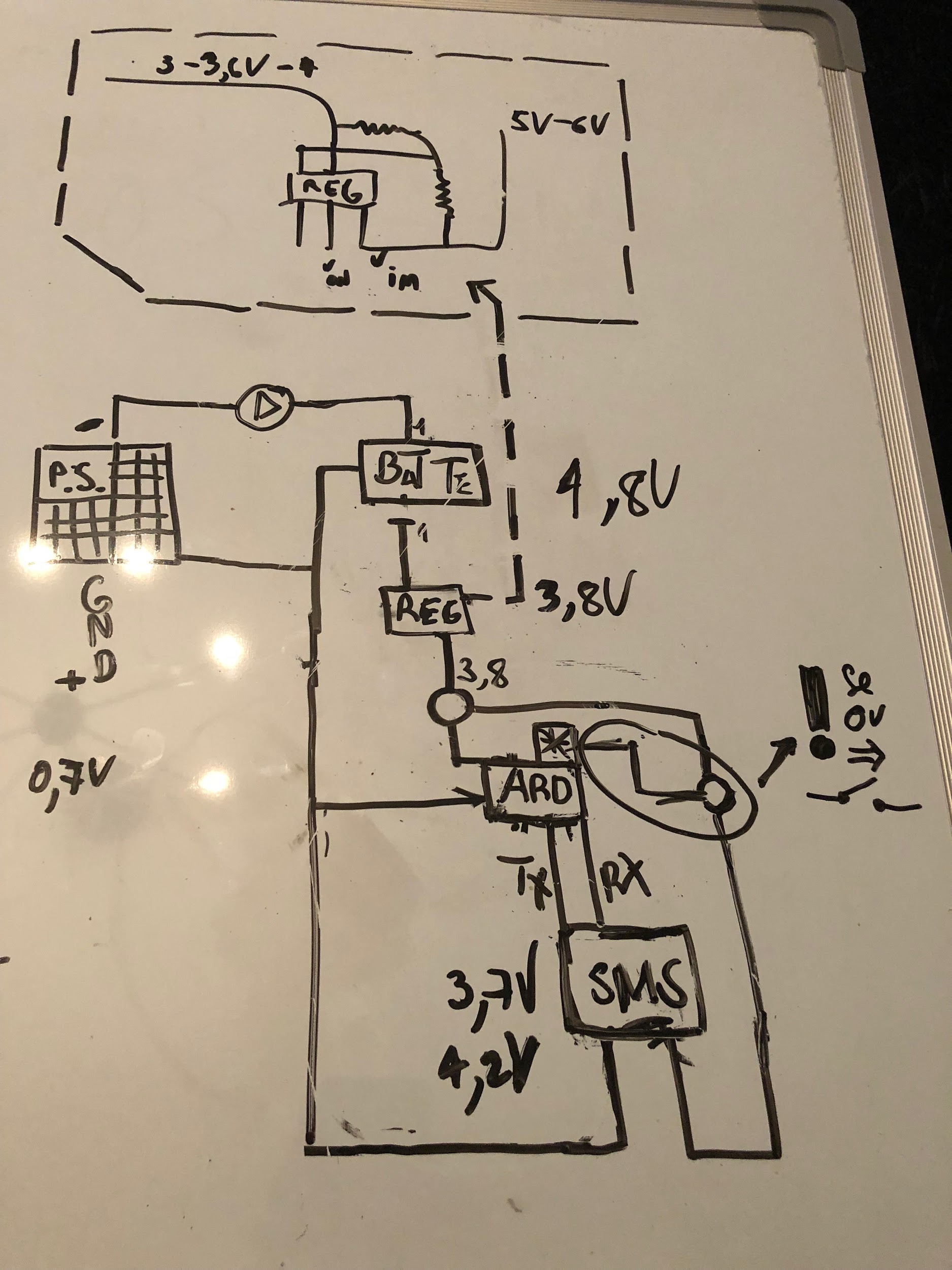
Projeto Mochu: Relatório

Circuito elétrico:

Alimentação:

O circuito será alimentado por 3-4 pilhas NiMH de 2500mAh, 1,2V. As pilhas são mais estáveis, menos perigosas, mais resistentes e menos poluentes que as pilhas de Li-Ion. A curva de descarga/carga de uma pilha de NiMH é bastante próxima da pilha teórica ideal, isto, porque a pilha não sofre uma variação acentuada de voltagem, ao longo do tempo de funcionamento. O Painel Solar ainda não foi escolhido. Entre o painel e as baterias deverá estar um diodo para impedir que a corrente elétrica flutue no sentido contrário ao pretendido (sentido pretendido: painel -> pilha).

Após o conjunto das pilhas, deverá ser montado um sistema de queima, onde um cabo de sacrifício se irá queimar.

Convém que a voltagem entrada do arduino não seja muito diferente da voltagem de operação, pois o regulador de tensão deste pode queimar os outros componentes da placa. O sensor de temperatura consegue funcionar com a corrente fornecida pelo arduino, não sendo necessário ligá-lo às baterias.

Consumo energético:

O arduino, em termos de programação, funciona por repetição contínua de um script, que por sua vezes, fisicamente. se representa por ciclos. Neste projeto, um ciclo é igual ao tempo de medição de voltagem, mais o tempo de sono. Existe, ainda, um outro ciclo que ocorrerá durante um incêndio, mas que agora pouco interessa. Voltando ao ciclo inicial, este demora entre 8,5-9 segundos, dos quais 0,5 - 1s são para fazer as respectivas medições e cálculos. Os restantes segundos são os que o arduino se encontra num modo de energia de consumo muito baixo. O consumo médio por ciclo do arduino é dado por:

Sendo: Cm - Consumo em média por ciclo (mA)

Con/off - Consumo ligado ou a dormir (mA)

Ton/off - Tempo ligado ou a dormir (s)

Após obtido o consumo médio por ciclo, é possível calcular o tempo que demora o arduino a descarregar as pilhas. O tempo de descarga da bateria depende da temperatura, variações e intensidade da corrente elétrica, descarregamento interno da pilha e variações de voltagem. Sem a utilização de um step-up converter (regulador de voltagem), o tempo de descarga obtém-se através da seguinte fórmula:

Sendo: Td - Tempo de descarga (h)

Cb - Capacidade da bateria ou pilhas (mAh)

ef - eficiência da pilha (%)

Nesta situação, é necessário entre 3 a 4 pilhas, sendo que estas devem ser montadas todas em série, porque o arduino necessita de pelo menos 3,3v (ou 5v) e o GSM de 3,8v. A fórmula acima não toma em consideração as mudanças de voltagem, pois essas não existem, significativamente. Segundo os nossos cálculos, o tempo de descarga será entre 90 a 105 dias, isto sem qualquer recarga do painel solar.

O step-up converter aumenta o consumo de corrente elétrica, mas esse consumo pode ter pouca relevância, visto que, desta forma, apenas duas pilhas têm de estar montadas em série. Como as pilhas ao longo do seu funcionamento perdem voltagem para calcular mais corretamente a potência de uma pilha, é necessário utilizar integrais. As restantes podem ser montadas em paralelo, aumentando assim o tempo de descarga.

Para se obter o tempo de descarga, é antes necessário calcular a potência consumida pelo step-up:

Sendo: Vin/out - Voltagem de entrada e de saída (V)

I in/out - Corrente de entrada e saída(mA)

efc - eficiência do step-up (%)

I out = Cm

E por fim, o tempo de descarga é:

Como é de esperar, os valores obtidos, nestas equações, serão sempre superiores aos obtidos experimentalmente. Existe, também, a lei de Peukert que expressa, aproximadamente, o tempo de descarga de uma bateria alcalina. A utilização desta lei para outro tipo de pilhas não é aconselhável, principalmente nas pilhas de Li-Ion, devido à temperatura de funcionamento que invalida a previsão da lei de Peukert. O fabricante de uma pilha disponibiliza, normalmente, as propriedades destas e alguns gráficos de como se comportam, os quais podem ajudar a prever a duração de uma pilha.

Recarga da Bateria:

Para aumentar o tempo de funcionamento do dispositivo, será utilizado um painel solar. O circuito de recarga é bastante simples, é necessário apenas um painel, um diodo e pilhas, isto porque as pilhas NiMH não geram muitos problemas ao serem recarregadas, desde que se respeitam as condições para um carregamento eficaz e seguro. Uma f condições é que a corrente de recarga deve ser, por norma:

Sendo: Cr - Corrente de recarga;

Cm - Corrente máxima fornecida pela bateria;

a - Constante experimental.

Se , então evita-se o sobre aquecimento da pilha.

Existe um problema que é impossível evitar, sem manutenção. Baterias ligadas em séries forçam a corrente a passar por cada uma delas. Se uma dessas baterias estiver descarregada e se for forçada a fornecer voltagem, então ocorre inversão de polaridade da mesma, impedindo a passagem de corrente num circuito DC.

A voltagem do painel solar tem de ser superior à voltagem das baterias em séries.

Sistema de queima:

O cabo de queima deve estar conectado ao arduino (ou à bateria, neste caso tem de existir um portão lógico para poupar energia) e no meio deste cabo tem de estar um fusível. A medição da voltagem é feita neste cabo. Este cabo existe para diminuir o consumo energético do arduino (caso contrário, os sensores teriam de estar sempre a funcionar), e para garantir que algo fisicamente se danificou (o software poderia ter um erro de cálculo e desencadear todo o processo de comunicação).

Sensor de Temperatura:

O sensor de temperatura é bastante fácil de montar e de comunicar. O única problema poderá ser que este leia temperaturas perigosas, mas que não sejam consequência de um incêndio, por exemplo: se a armadura for pintada de preto, durante o verão o material desta pode aquecer bastante chegando até aos 70 ou 90 Cº, as quais já são temperaturas relativamente perigosas. O sensor tem uma leitura máxima de 125 ºC. Ainda que o cabo de queima se sacrifique, o arduino tem de ainda verificar se a temperatura é aceitável para se fazer uma comunicação. A temperatura limite a escolher tem de respeitar o tempo necessário para o arduino conseguir fazer os cálculos e a comunicação.

GSM:

O módulo de GSM é o que permite a comunicação entre o arduino e as respectivas autoridades (Bombeiros, etc...). O módulo tem de ser ligado ao arduino e alimentado.

A alimentação deste módulo revela-se complicada, porque o arduino não tem a capacidade de fornecer uma corrente pico de 2A. Poderia se, facilmente, ligar o GSM às baterias e este funcionaria sem problemas, mas irá consumir corrente quando não é necessário, pois o módulo liga-se automaticamente assim que recebe corrente. A resolução a este problema, poderá ser a utilização de um circuito lógico que impeça a passagem de corrente, até que se verifique uma certa condição. Outro método seria fornecer corrente através do arduino e condensadores em paralelo, e assim já seria possível desligar o módulo quando não é necessário, porque neste caso o arduino controla a corrente que o GSM consome.

Neste momento, é difícil de prever teoricamente qual o melhor método, por falta de conhecimento na área de eletrônica.

De resto, o GSM revela-se mais eficaz que uma antena feita por nós (mais eficaz em alcance, consumo e espaço ocupado).

Código:

*#include<SoftwareSerial.h>*

*#include <LowPower.h>*

*#include<DallasTemperature.h>*

*#include <OneWire.h>*

*float voltagem;//voltagem a variavel, em float (numero real com 8 casas decimais), precisao*

*float temp = 0;//temperatura do sensor*

*int aviso = 0;//aviso a variavel, em int (numero real com duas casas ou quatro casas decimais), começa em 0*

*void setup() { //codigo que so corre uma vez, no inicio do programa*

*Serial.begin(9600);//inicio de comunicaçao com o computador a 9600 bits, se necessario*

*}*

*int warning(){ //funçao de aviso (por causa de ruidos de voltagem, por vezes o circuito pode ir abaixo de 1volts)*

*//se a voltagem se manter abaixo de 1 volt por mais de 4 mediçoes entao o sensores ativam se*

*if (voltagem < 1) { //se a voltagem for menor que 1 volt*

*aviso++;//então adiciona um aviso*

*Serial.println(aviso);*

*if(aviso > 4){ // e se o aviso for maior q 4, entao corre os sensores*

*sensor\_temp();//funçao dos sensor temperatura*

*Serial.println("4");*

*}*

*} else {//se a voltagem for maior q 1 entao os avisos voltam a zero*

*Serial.println("vivo");*

*Serial.println(aviso);*

*aviso = 0;*

*return aviso;*

*}*

*}*

*int sensor\_temp(){//funçao do sensor temperatura. é chamado pela função warning*

*#define ONE\_WIRE\_BUS 2 // Data wire is conntec to the Arduino digital pin 2, as chamadas sao feitas aqui para poupar energia*

*OneWire oneWire(ONE\_WIRE\_BUS);// Setup a oneWire instance to communicate with any OneWire devices*

*DallasTemperature sensors(&oneWire); // Pass our oneWire reference to Dallas Temperature sensor*

*sensors.begin();*

*// Call sensors.requestTemperatures() to issue a global temperature and Requests to all devices on the bus*

*// Why "byIndex"? You can have more than one IC on the same bus. 0*

*refers to the first IC on the wire*

*sensors.requestTemperatures(); //pedida da leitura dos sensores para a libraria dallas*

*temp = sensors.getTempCByIndex(0);// é escolhido o primeiro sensor na linha one wire*

*Serial.print("Celsius temperature: ");*

*Serial.print(temp);*

*if(temp > 90){ //se a temperatura for maior q 90oC, chama a funçao do GSM*

*gsm\_sms();*

*}*

*}*

*void gsm\_sms(){*

*#define SIM800\_TX 8*

*#define SIM800\_RX 9*

*SoftwareSerial sim800(SIM800\_TX,SIM800\_RX);*

*sim800.begin(9600);*

*delay(1000);*

*sim800.write("AT+CMGF=1\r");*

*delay(1000);*

*sim800.write("AT+CMGS=\"918671154\"\r");*

*delay(1000);*

*sim800.write("Data: Local: Temp:");*

*delay(1000);*

*sim800.write(0x1A);*

*Serial.println("sms grrehhhehehethheheehr");*

*}*

*/\*int low\_power(int sleep\_time){*

*for(int low\_temp = 0; low\_temp < sleep\_time; low\_temp++){//dorme*

*durante o tempo escolhido*

*LowPower.powerDown(SLEEP\_2S, ADC\_OFF, BOD\_OFF);*

*Serial.println(low\_temp);*

*}*

*} \*/*

*int volt\_med(){*

*int sensorValue = analogRead(A0);//le a porta A0*

*voltagem = sensorValue \* (5.0 / 1023.0);//a porta le entre 0-1023 valores. Esta é a conversao para volts*

*Serial.println(voltagem);//mostra o valor que le, ctrl + shitf + m*

*}*

*void loop() { //codigo que corre repetidamente*

*volt\_med();//funçao de medição de voltagem*

*warning();//chama a funçao warning*

*//low\_power(3);//dorme durante um tempo*

*delay(500);//espera 0.5s para fazer a proxima medição*

Armadura:

Impacto Ambiental:

Área de propagação do dispositivo:

