CPET - CENTRO DE PROFISSIONALIZAÇÃO E EDUCAÇÃO TÉCNICA

RICARDO CAMPOS

TELEMETRIA DE EQUIPAMENTO EM ÁREAS REMOTAS COM FOCO NA DIMINUIÇÃO DE CUSTOS DE IMPLANTAÇÃO E MANUTENÇÃO/OPERAÇÃO

SÂO JOSÉ -SC
2022

RICARDO CAMPOS

TELEMETRIA DE EQUIPAMENTO EM ÁREAS REMOTAS COM FOCO NA DIMINUIÇÃO DE CUSTOS DE IMPLANTAÇÃO E MANUTENÇÃO/OPERAÇÃO

Trabalho de Conclusão de Curso apresentado ao curso de Trabalho de Conclusão de Curso - ELETROTÉCNICA , área..., da Centro de Profissionalização e Educação Técnica, como requisito parcial para a Obtenção do grau de Bacharel em Trabalho de Conclusão de Curso - ELETROTÉCNICA .

SÂO JOSÉ -SC
2022

RICARDO CAMPOS

TELEMETRIA DE EQUIPAMENTO EM ÁREAS REMOTAS COM FOCO NA DIMINUIÇÃO DE CUSTOS DE IMPLANTAÇÃO E MANUTENÇÃO/OPERAÇÃO

Trabalho de Conclusão de Curso apresentado ao curso de Trabalho de Conclusão de Curso - ELETROTÉCNICA , área..., da Centro de Profissionalização e Educação Técnica, como requisito parcial para a Obtenção do grau de Bacharel em Trabalho de Conclusão de Curso - ELETROTÉCNICA .

BANCA EXAMINADORA

_____

Prof. Dr. .....

Universidade .....

_____

Prof. Dr. .....

Universidade .....

_____

Prof. Dr. .....

Universidade .....

Dedico este trabalho a minha mãe Geni Dias de Souza, família e amigos que sempre me incentivaram.

AGRADECIMENTOS

Agradeço aos meus professores e colegas por me ajudarem a desenvolver este trabalho.

É ótimo celebrar o sucesso, mas mais importante ainda é assimilar as lições trazidas pelos erros que cometemos". - Bill Gates

RESUMO

Hoje a automação tomou conta de diversos processos como de processamento de grandes volumes de dados a processos físicos industriais complexos, inclusive a automação é utilizada em larga escala para processos mais simples como alimentador automático de animais ou até um portão eletrônico, para alguns desses processos a comunicação é imprescindível, chamada popularmente de telemetria é utilizada para monitoramento e controle em tempo real. A telemetria é principalmente utilizada em processos de automação como controle de sistemas de saneamento, cofres inteligentes, sistemas em áreas remotas ou de difícil acesso, controle de frotas, monitoramento de nível de reservatórios e muitos outros. Para ambientes remotos ou de difícil acesso uma aplicação conhecida é em sistemas de monitoramento por imagens de regiões de mata, as câmeras PTZ são posicionadas em torres geralmente muito altas, embora algumas possuem entrada de energia AC outras constam apenas com um pequeno gerador eólico, painel solar e bateria, possuem uma conexão com a internet via fibra ótica ou rádio, esses equipamentos possuem um elevado custo de manutenção preventiva e corretiva, além disso eventualmente fica inoperante por problemas simples como, travamento da câmera ou do conversor de fibra ótica. O objetivo desse projeto é solucionar esses problemas com telemetria.

Palavras-chave: Telemetria. Automação. Monitoramento.

ABSTRACT

Today automation has taken over several processes such as processing large volumes of data to complex industrial physical processes, including automation is used on a large scale for simpler processes such as automatic animal feeder or even an electronic gate, for some of these processes the communication is essential, popularly called telemetry is used for real-time monitoring and control. Telemetry is mainly used in automation processes such as control of sanitation systems, smart safes, systems in remote or difficult to access areas, fleet control, reservoir level monitoring and many others. For remote environments or difficult access, a known application is in monitoring systems by images of forest regions, the PTZ cameras are positioned in towers that are usually very high, although some have AC power input, others only have a small wind generator, solar and battery, they have an internet connection via fiber optics or radio, these equipments have a high cost of preventive and corrective maintenance, in addition, it eventually becomes inoperative due to simple problems such as camera or fiber optic converter locking. The objective of this project is to solve these problems with telemetry.

Keywords: Telemetry. Automation. monitoring.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AC	Corrente alternada
DC	Corrente contínua
DHCP	Dynamic Host Configuration Protocol
GET	Métodos HTTP para solicitar dados
HTTP	Hyper Text Transfe Protocol
IN	Na eletrônica utilizamos para entrada de sinais
MHz	Mega Hertz
NA/NF	Normalmente Aberto e Normalmente fechado
OUT	Na eletrônica utilizamos para saída de sinais
PTZ	Pan, Tilt e Zoom.
RJ45	Conector para ethernet (jack registrado de 8 vias)
URL	Uniform Resource Locator

LISTA DE SÍMBOLOS

A	Ampere
R	Resistência
V	Volt
Hz	Hertz

SUMÁRIO

1 **INTRODUÇÃO**

Este documento detalha o projeto de fim de curso realizado pelo aluno Ricardo Campos no curso de Técnico de eletrotécnica, Trabalho de conclusão de curso de eletrotécnica na instituição de ensino CPET. O curso não exige estágio, porém atuei na área de automação industrial por mais de cinco anos e atualmente trabalho como desenvolvedor de software.

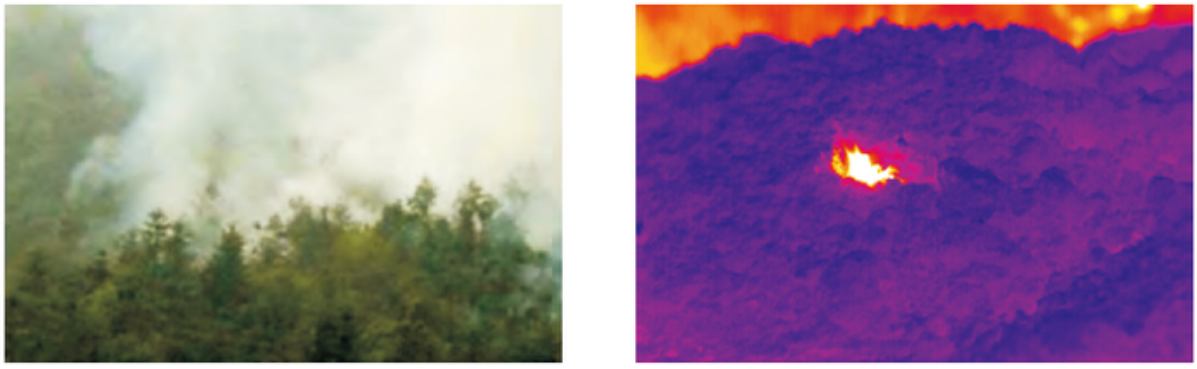
O trabalho apresentado abaixo é autoral incluindo hardware e software, tem como objetivo demonstrar conhecimento em algumas áreas abordados no curso, como grandezas elétricas demonstrada na construção dos circuitos, eletrônica digital demonstrada na utilização de circuitos digitais no projeto, eletrônica analógica demonstrada nos sensores de tensão e corrente desenvolvidos neste projeto, automação e controle demonstrada na funcionalidade final do projeto tendo em vista seu objetivo de automatizar processos, controlar remotamente o dispositivo via internet/intranet (protocolo tcp IP), protocolo de comunicação demonstrada na utilização de uma porta de rede ethernet e implementação funcional de protocolo TCP/IP via firmware, manutenção preventiva e corretiva demonstrada na análise de caso de uso no qual apresentamos no capítulo 2, um caso clássico de manutenção corretiva e preventiva que precisa ser melhorado com o auxílio da tecnologia, Software apresentamos um firmware que será demonstrado integralmente nos capítulos subsequentes, além de um software client, este não faria parte do projeto total, mas foi necessário para complementar o projeto. O projeto do hardware foi simulado no Software proteus e funcionou perfeitamente, assim como ambos os softwares, o firmware foi simulado diretamente no Proteus, já o software client foi simulado no modo desenvolvedor utilizando o framework node.js e React js.

Esse projeto vai ficar aberto ao público no repositório de software referenciado os capítulos finais, o com licença livre para que outros estudantes possam utilizar em seu projetos de estudo, visando auxiliar novos estudantes e entusiastas adicionei ao último capítulo desse projeto sugestões de melhoria em várias partes do projeto.

2 PROBLEMA E PROPOSTA DE SOLUÇÃO

Para esse projeto será analisado um problema de monitoramento e gerenciamento de equipamento em região remota ou de difícil acesso, nesse caso analisaremos um sistema de monitoramento de mata em ambiente de preservação ambiental com objetivo de detectar início de focos de incêndio para que possa ser combatido rapidamente, os satélites geralmente não detectaram focos iniciais de incêndio e ainda não possuem monitoramento constante, o sistema não é amplamente utilizado pois possui ainda elevado custo de implantação, já que utiliza uma câmera PTZ térmica, essa por sua vez é posicionada em torres geralmente muito altas, o que possibilita monitoramento térmico de grandes distâncias, a câmera térmica pode chegar na casa de dezenas de milhares de reais, sem contar o custo de implantação e manutenção, por outro lado o combate de um incêndio que já passou da fase inicial ou queimadas de grandes áreas de fauna chega em um valor incalculável. Uma alternativa com a popularização de drones e o aumento de sua autonomia, seria a utilização de câmeras mais baratas instaladas em drones autônomos, porém essa alternativa ainda precisa ser amadurecida. Com esse estudo de caso analisaremos de forma detalhada as dificuldades de manutenção e a dificuldade de interruptibilidade do sistema para o caso proposto.

Imagem 1 — Exemplo de visão térmica em área de mata



Fonte: yoseenir (2022).

2.1 O SISTEMA

O sistema de monitoramento utilizado hoje em algumas localidades é composto de uma torre em local estratégico, no alto da torre uma luz de ponto que acende na parte da noite, um painel solar, um gerador eólico pequeno, uma câmera ip térmica PTZ, uma caixa hermética onde equipamentos como: conversor de fibra óptica (alguns casos possuem conexão via rádio, a velocidade de conexão é de 1MB+), roteador, fonte, bateria, disjuntor e conversores necessários. Alguns casos possuem entrada AC 220v, então será previsto um monitoramento para entrada AC no nosso sistema, um relé fotoelétrico é utilizado para acender a lâmpada de ponto na parte da noite, por se tratar de uma câmera PTZ, ele possui visão de 360° que pode ter uma rota programada além de alarme ao encontrar focos de incêndio.

2.2 O PROBLEMA

Devido o local de instalação do sistema de difícil acesso e a diversidade de equipamentos utilizados, muitos problemas começam a surgir, sendo o mais comum o travamento do conversor de fibra óptica/roteador, embora o problema seja simples a logística para reiniciar o equipamento é muito complexa, o sistema fica inutilizado sem a comunicação com a câmera, geralmente por 15 dias ou mais, outro problema parecido é o travamento da câmera. Alguma torres possuem entrada de energia AC da concessionária, porém devido o ambiente mais distante, quedas de energia AC são constantes e, para estes casos o painel solar e gerador eólico são responsáveis por manter as baterias carregadas ou até mesmo manter o sistema por tempo indeterminado, funciona bem mesmo em locais onde não se tem entrada de energia da concessionária, por esse motivo é importante que o sistema de micro geração de energia esteja sempre com a manutenção preventiva em dia. Como analisamos esses diversos problemas são na sua maioria comuns, porém extremamente difíceis de resolver, pois o acesso é dificultado, além disso é uma manutenção muito cara e demorada, tudo isso acaba inviabilizando a utilização na maioria dos casos, pois o sistema tende a ficar longos períodos inoperante, o que acaba não justificando o alto custo de manutenção e aquisição.

2.3 PROPOSTA DE SOLUÇÃO

Para solucionar os problemas encontrados podemos utilizar a automação e telemetria, permitindo o monitoramento, controle e realização de atividades autônomas como watchdog que testa a conexão com internet e reinicia o sistema de

comunicação de forma autônoma para que o sistema não se mantenha travado, monitoramento de tensão e corrente elétrica gerada permitindo que se verifique a saúde dos equipamentos bem como a montagem de gráficos e relatórios de manutenção, vida útil e inclusive a maior incidência de geração de energia durante o ano, permitindo dimensionar melhor equipamentos futuros, verificação de disponibilidade de energia AC, Medição de tensão da bateria para verificar sua carga, acionamento da lâmpada de ponto, saídas de Relé controladas remotamente para reiniciar equipamentos através de comandos http, É possível fazer de forma relativamente simples e é assim que vamos explorar essa ideia no projeto proposto, precisamos disponibilizar quatro portas com saída de contato seco NA/NF, três portas com leitor de tensão DC até 75 V com proteção para sobre tensão, três portas com leitor de corrente DC até 20A, três portas de leitura digital (IN) e uma porta de corrente alternada para verificação de existência de energia elétrica da concessionária AC. Como o objetivo é simular o funcionamento dos circuitos eletrônicos via software Proteus, vamos utilizar componentes que permitem a simulação e que permitam a prototipação de forma manual, partindo desse princípio a primeira tarefa é o levantamento de recursos necessários no componente eletrônico que iremos projetar.

2.4 LEVANTAMENTO DE RECURSOS E ETAPAS

Primeiramente levantaremos grosseiramente alguns recursos para trabalhar nos circuitos em cima dessa base, vamos entender a quantidade de portas que precisaremos no microcontrolador, interface de comunicação e o sistema de potência que nada mais é que um conversor 12v para 5V.

2.4.1 **Interfaces e recursos do microcontrolador**

Para as entradas de leitura de tensão, temos que projetar um circuito aproveitando as portas analógicas do microcontrolador que possui resolução de 0 a 1024 na conversão analógico digital, e tensão máxima de 5V DC. Para as entradas de leitura de corrente também será utilizado entradas analógicas do microcontrolador, neste caso usaremos um componente adequado para leitura de até 20A. Para o sensor de energia elétrica AC, chamaremos de região de alta tensão embora a tensão esperada seja de 220V, como o objetivo é saber se existe energia apenas e não mediar a tensão, usaremos uma porta lógica do microcontrolador configurada como IN. Usaremos ainda 3 portas lógicas configuráveis como IN para

adicionar outras funções como um LDR para detectar a luminosidade do dia e acionar a lâmpada de ponto quando escurecer, um alarme e uma de reserva. para os relés usaremos quatro portas lógicas configuradas como OUT.

2.4.2 **Potência**

Precisaremos de um conversor 12V para 5V, o ideal seria separar os circuitos de potência em dois, entenderemos o porquê no decorrer do projeto, porém como objetivo será didático, alimentamos todo o circuito com apenas um sistema de potência de 5V e um de 3.3v..

2.4.3 **Comunicação**

Existem diversas formas de adicionar uma interface ethernet para conexão TCP IP, vamos utilizar a mais simples, embora apresente algumas desvantagens em relação as outras vai atender ao propósito e tornar o projeto mais didático, então precisaremos de um controlador ethernet e uma conexão femea RJ45.

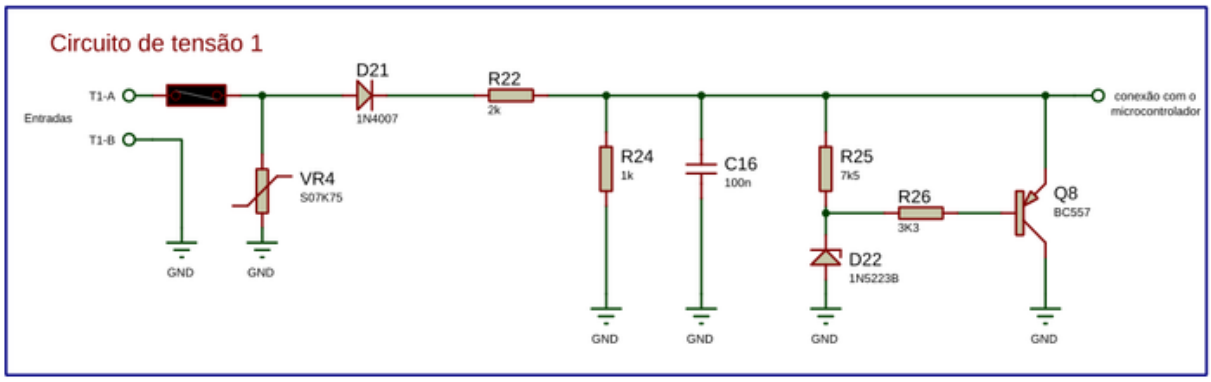
3 **ANÁLISE E PROJETO DO HARDWARE**

Conforme nosso levantamento de recursos vamos escolher um microcontrolador com memória suficiente para o nosso software, para isso precisaremos além dos recursos de entradas, saídas e regras de negócio, definir o Microchip Ethernet que será utilizado, bom, utilizaremos um enc28j60(capítulo 3 item 3.6) com base nisso utilizaremos um Atmega2560, este chip permitirá atender todas as nossas necessidades levantadas até aqui, além de disponibilizar uma grande quantidades de recursos adicionais para melhorias no projeto, como as melhorias que vou sugerir ao final desse projeto em capítulo 8.

3.1 CIRCUITO LEITOR DE TENSÃO

Primeiramente vamos definir uma tensão máxima, para adicionarmos um circuito de proteção, para que o microcontrolador não seja afetado em possíveis surtos, então vamos definir um range de 0 a 100V DC e colocaremos um limite de 75V para entrada máxima no circuito, adicionamos um varistor com pico máximo de 75V e um fusível compatível em paralelo:

Desenho 1 — circuito de tensão



Fonte: O autor (2022) Campos.

Na figura 2 podemos identificar as entradas (T1-A + e T2-B -), vamos utilizar também dois divisores de tensão para que o circuito leitor atinja no máximo 5V (quando a entrada for igual a 100V).

resolução do conversor analógico digital = 1024

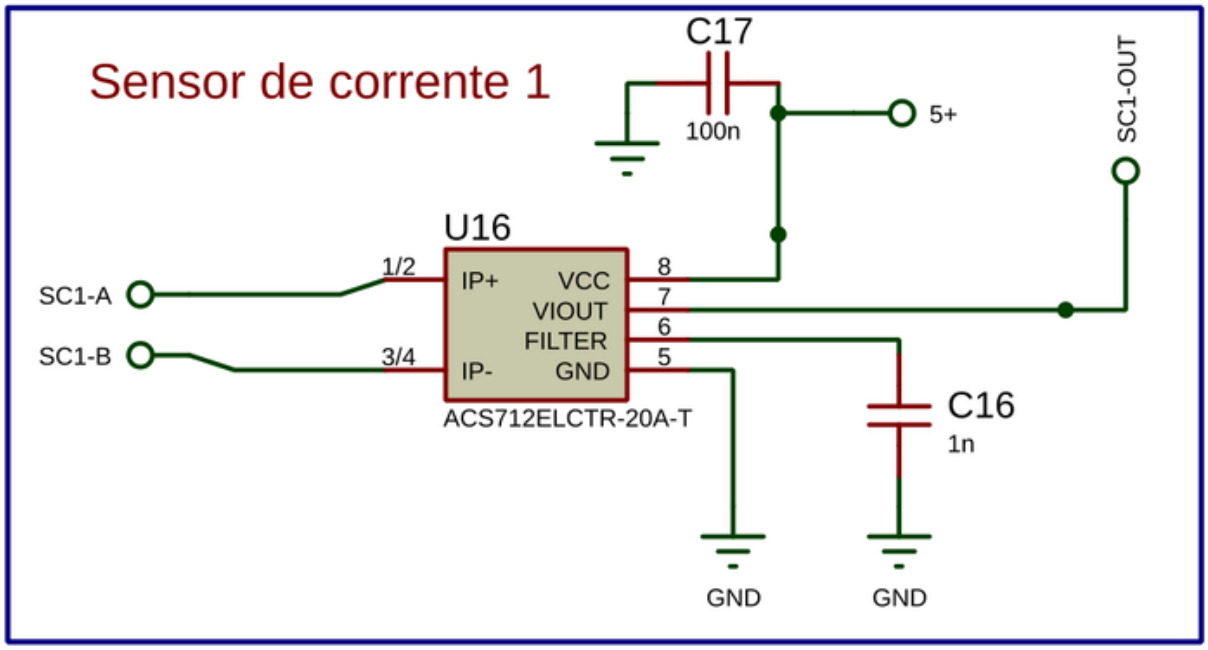
1024 = 100v

3.2 CIRCUITO LEITOR DE CORRENTE

Para o sensor de corrente usaremos um componente ACS712 de até 20A. O

circuito foi projetado conforme a recomendação no datasheet e trabalha em um range de 0 a 5V na saída.

Desenho 2 — Sensor de corrente



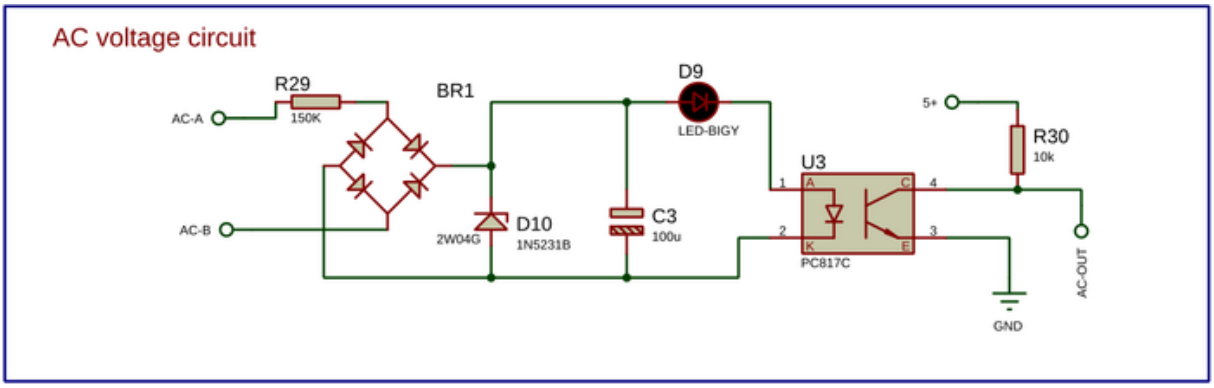
Fonte: O autor (2022) Campos.

A entradas para o circuito são SC1-A e SC1-B, não adicionamos proteção de sobretensão e sobrecorrente, porém pode ser adicionado, o pico máximo de 20A suportado pelo componente é uma boa alternativa tendo em vista a geração os sistemas já apresentados de micro geração de energia. O terminal SC1-OUT é conectado em uma porta analógica do microcontrolador.

3.3 CIRCUITO VERIFICADOR DE EXISTÊNCIA DE AC

Chamaremos essa parte do circuito de área de alta tensão devido a grande diferença entre os 5V DC e a entrada de tensão 220V AC, é um circuito simples porém isolado na placa devido ao uso de um optoacoplador.

Desenho 3 — Sensor verificador de AC



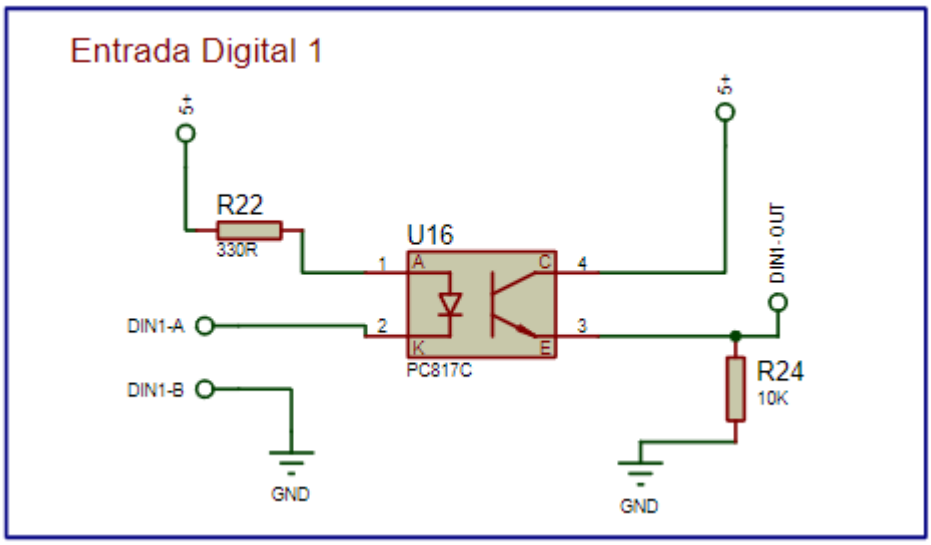
Fonte: O autor (2022) Campos.

A Corrente elétrica AC é transformada em DC através de uma ponte retificadora e a tensão é diminuída para que possamos utilizar diretamente no optoacoplador, o terminal AC-OUT é conectado diretamente na na porta do microcontrolador em uma porta lógica.

3.4 CIRCUITO LEITOR DE ENTRADAS DIGITAIS

Para entrada digital um circuito simples usando um optoacoplador, é possível notar que ambos os lados do acoplador utilizam a mesma fonte de alimentação 5V, essa alimentação (que não é funcional) foi feita desta forma pois possuímos apenas um circuito alimentador de 5V e um de 3.3V, essa é uma melhoria que será sugerida para ser feita ao final do projeto, um segundo circuito de 5V para fazer essas alimentações externas.

Figura 1 — Circuito de entrada digital

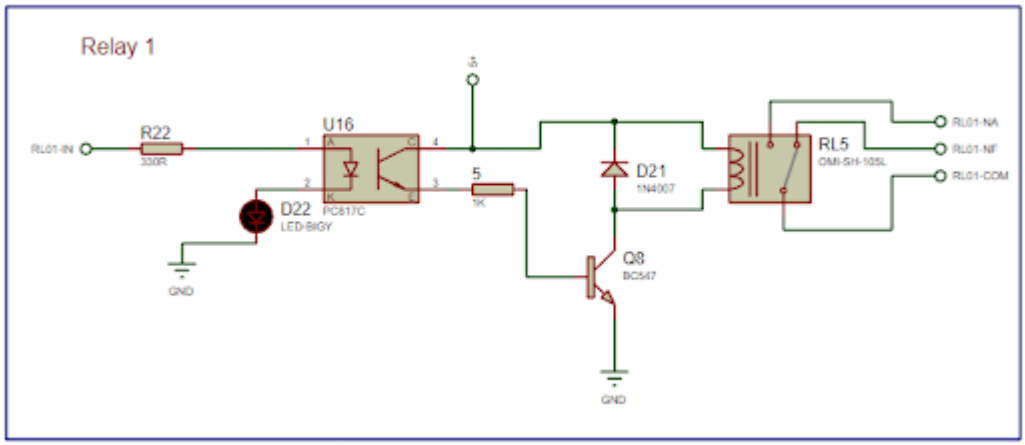


Fonte: O autor (2022) Campos.

3.5 CIRCUITO DE RELÉS

Este também é um circuito simples porém robusto e também utiliza um optoacoplador com saída transistorizada e diodo ligado em paralelo para proteção é ligado diretamente em uma porta digital do microcontrolador que é configurada como saída. Possui o mesmo problema do circuito anterior, pois utiliza o 5V comum na parte externa do circuito.

Desenho 4 — Circuito de reles

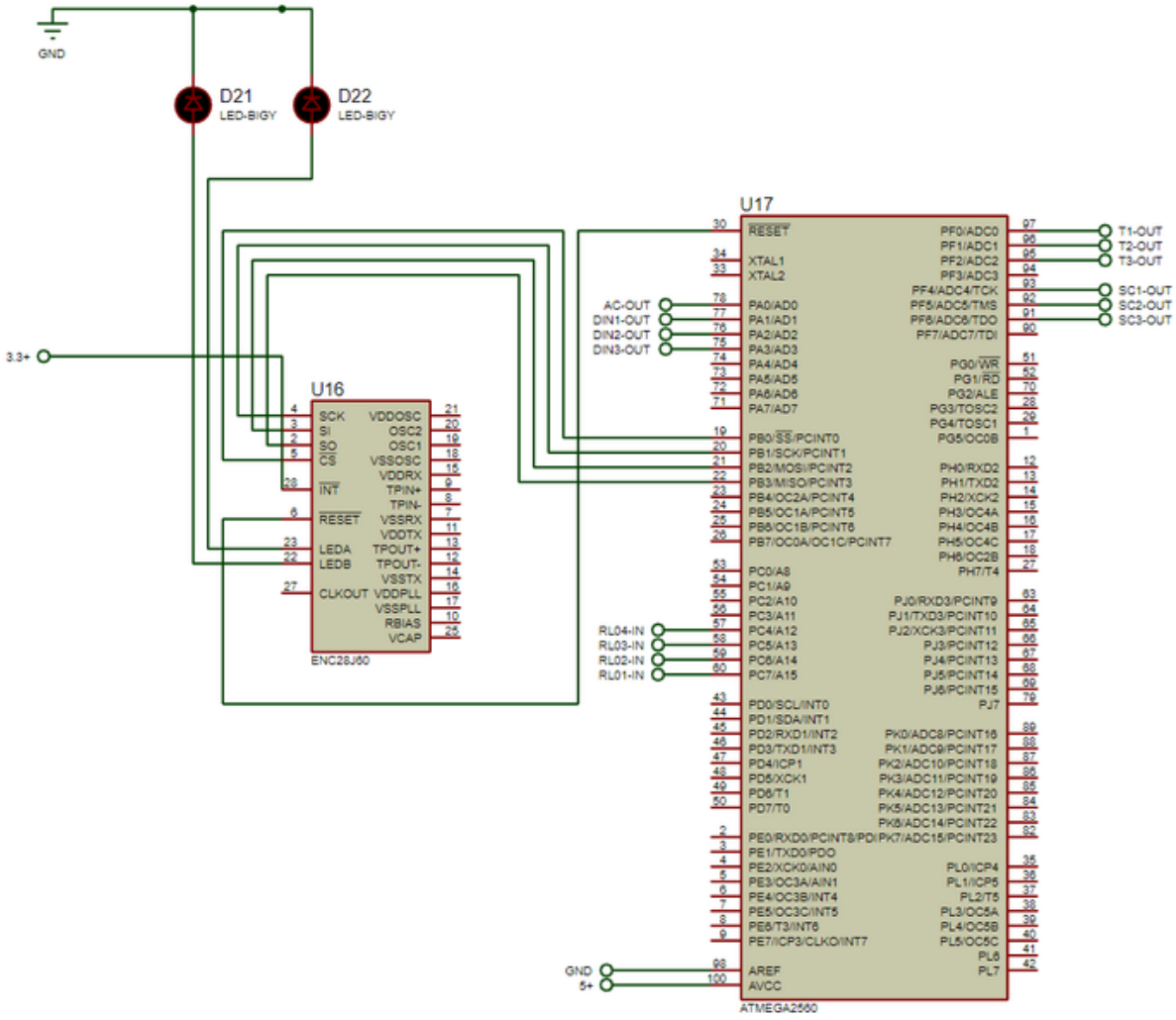


Fonte: O autor (2022) Campos.

3.6 ETHERNET

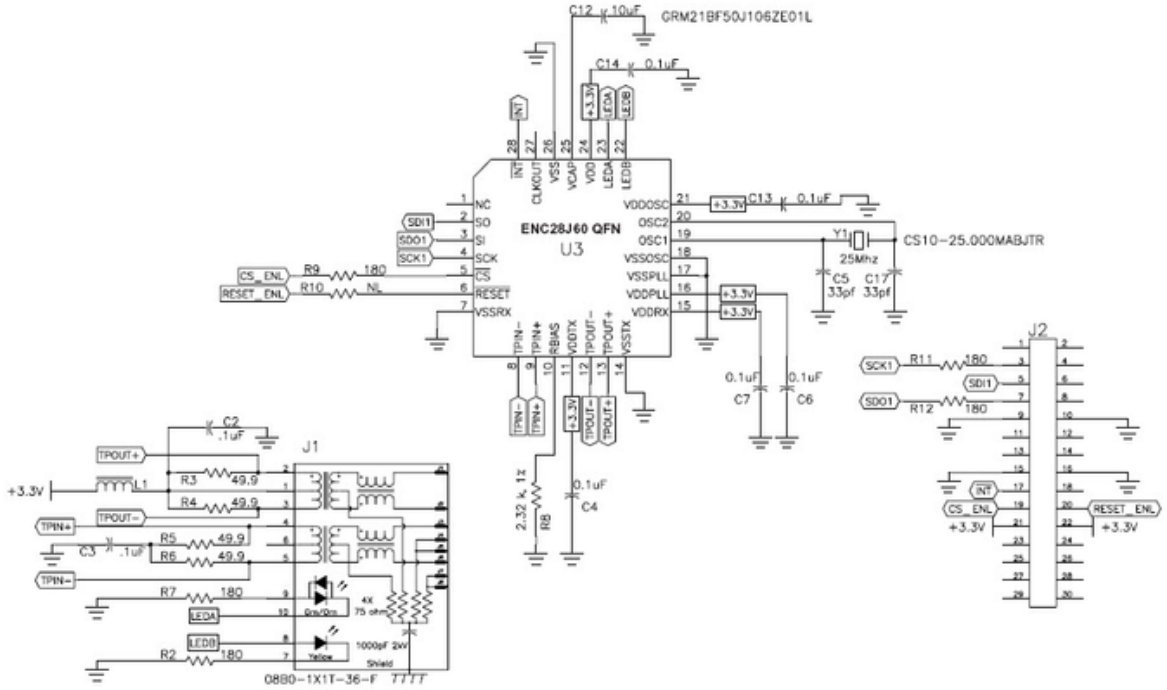
Esse é um circuito baseado no chip enc28j60, foi escolhido pela simplicidade porém a implementação do protocolo TCP/IP é em software e essa carga cai toda em cima do microcontrolador, porém para fins didáticos a implementação é muito simples, possível ser simulada conforme a Figura 7 e como estamos utilizando um chip Atmega podemos utilizar algumas bibliotecas do Arduino que são open source e podem ser utilizadas mesmo em projetos comerciais. Essa representação de circuito apresentada na Figura 7 omite o cristal de 25 Mhz, porta RJ45 e demais elementos conforme é possível observar na figura Figura 8, isso porque estamos simulando no proteus e esses componentes citados são adicionados por baixo dos panos de forma automática para simulação.

Desenho 5 — Circuito digital e ethernet



Fonte: O autor (2022) Campos.

Figura 2 — projeto de referência usando o chip ENC28J60 da Microchip Technology



Fonte: Microchip (p. 02).

3.7 POTÊNCIA

O circuito de potência é suficiente para a alimentação do sistema porém precisa de dissipadores robustos, por isso exige melhorias para aplicações práticas ou comerciais, a primeira melhoria seria o desenvolvimento de uma nova fonte de alimentação 5V para os circuitos externos mantendo essa para os circuitos internos, o circuito 3.3v é utilizado pelo controlador ethernet, ambos utilizam os reguladores LM.

4 **FIRMWARE**

A forma mais simples de desenvolver este firmware é utilizando um framework do Arduino para aproveitamos algumas funções que a ferramenta proporciona, deixando nosso código menos verboso e mais didático, o uso do framework do Arduino é totalmente compatível com o atmega2560 pois é o mesmo chip utilizado no Arduino mega.

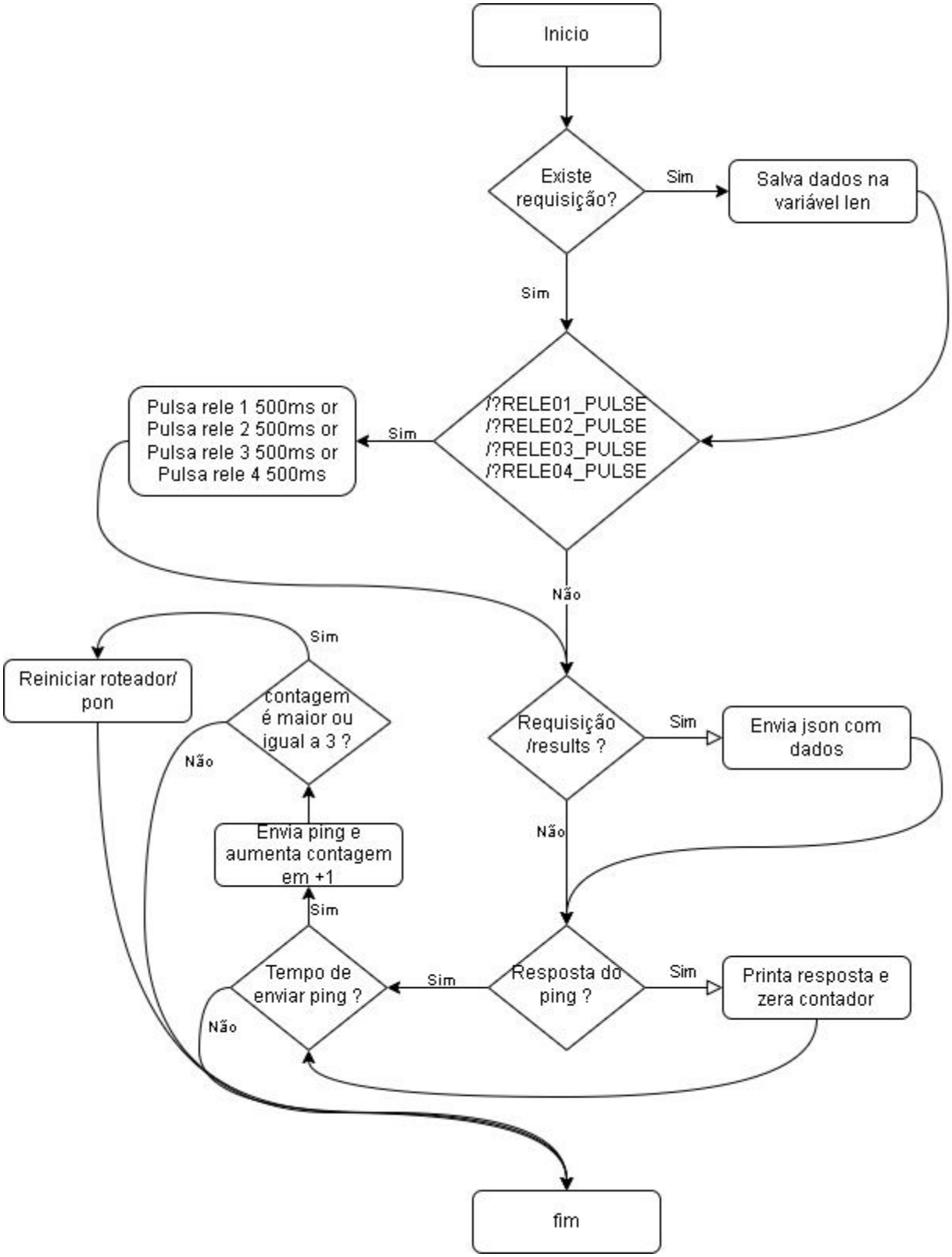
4.1 IDE E PLATAFORMA

O firmware apresentado abaixo foi desenvolvido na IDE VSCODE com a extensão PlataformIO, essa extensão permite o desenvolvimento do software, adição de framework e bibliotecas externas como o framework do Arduino por exemplo, a extensão em questão também permite descarregar os binários diretamente no chip. Como estamos simulando, utilizamos uma opção da extensão PlataformIO que apenas compila o código assim o arquivo “firmware.elf” é configurado no PROTEUS (software no qual projetamos nosso circuito e simulador).

4.2 FLUXO DE LOOP DE FIRMWARE

A figura 10 representa o fluxo de loop do servidor web que será implantado no firmware do Atmega 2560, inicialmente a biblioteca EtherCard cuida dos protocolos TCP cada vez que chamamos a função packetreceive(), a partir desse ponto é possível visualizar o fluxo de comparações, os nomes apresentados, são os paths a partir do IP, exemplo “/?RELE01_PULSE” a url ficaria http://192.168.0.200/?RELE01_PULSE, cada vez questão, o fluxo identifica e executa a ação.

Desenho 6 — Fluxo de loop do firmware



Fonte: O autor (2022) Campos.

4.3 CÓDIGO

para iniciar o código precisaremos carregar apenas uma biblioteca pois ao framework do arduino é configurado diretamente na extensão, na utilização de outra IDE, o framework deve ser adicionado via #include, a biblioteca a se adicionada é Ethercard que é uma biblioteca de código aberto, e pode ser encontrada em : https://github.com/njh/EtherCard, nosas extrutura inicial começa da seginte forma:

```
#include <EtherCard.h>
void setup () {
}
void loop () {
}
```

4.3.1 **Definições iniciais**

Inicialmente vamos definir algumas constantes e algumas variáveis, quando a plataforma é usada diretamente em um roteador que possui serviço de distribuição automática de ip's, a opção “STATIC” pode possuir valor fixo 0 (zero) isso permite que o hardware receba um ip aleatório definido pelo roteador, no nosso caso definimos como 1 (um), para utilizar IP fixo conforme as linhas 8 e 10 na figura 10, um MAC deve ser definido obrigatoriamente (linha 6 da figura 10). Para o mapeamento de portas utilizamos nomes que façam sentido com a sua utilização, um exemplo é "volts01" que representa leitura de tensão porta 1 do hardware, esse padrão segue para os leitores de corrente, portas de entradas e saídas digitais.

Imagem 2 — Declarações iniciais

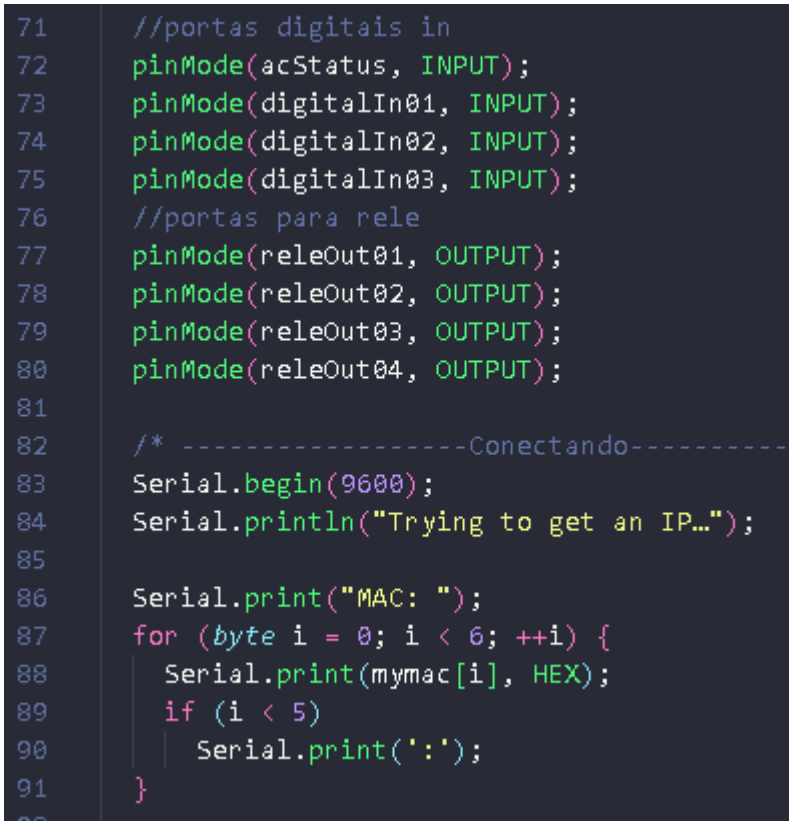
```
#include <EtherCard.h>

#define STATIC 1 // DHCP ( 1 = sim, 2 = não)

// mac
static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0x31 };
// ethernet  ip
static byte myip[] = { 192,168,1,200 };
// gateway ip
static byte gwip[] = { 192,168,1,1 };


int current01 = A0;
int current02 = A1;
int current03 = A2;

int volts01 = A4;
int volts02 = A5;
int volts03 = A6;

int acStatus = 22;    //verifica a existencia de ac

int digitalIn01 = 23; //entrada digital 01
int digitalIn02 = 24; //entrada digital 02
int digitalIn03 = 25; //entrada digital 03

int releOut01 = 30; // saida relé
int releOut02 = 31; // saida relé
int releOut03 = 32; // saida relé
int releOut04 = 33; // saida relé

byte Ethernet::buffer[2000];
static uint32_t timer;
```

Fonte: O autor (2022) Campos.

4.3.2 **Setup**

Para o setup primeiramente será definido os pinos digitais como saídas ou entradas, conforme o que foi definida no hardware e definido o baud rate da porta serial, utilizando o método begin que inicializa nas portas padrões 2 e 3(RXD0 e TXD0 respectivamente) do chip atmega 2560, a partir deste ponto, é feito print do MAC e iniciamos a busca por ip caso use DHCP.

Figura 3 — Declarações em setup()



```
71    //portas digitais in
72    pinMode(acStatus, INPUT);
73    pinMode(digitalIn01, INPUT);
74    pinMode(digitalIn02, INPUT);
75    pinMode(digitalIn03, INPUT);
76    //portas para rele
77    pinMode(releOut01, OUTPUT);
78    pinMode(releOut02, OUTPUT);
79    pinMode(releOut03, OUTPUT);
80    pinMode(releOut04, OUTPUT);
81
82    /* ------------------Conectando----------
83    Serial.begin(9600);
84    Serial.println("Trying to get an IP…");
85
86    Serial.print("MAC: ");
87    for (byte i = 0; i < 6; ++i) {
88      Serial.print(mymac[i], HEX);
89      if (i < 5)
90        Serial.print(':');
91    }
```

Fonte: Campos.

A busca por DHCP inicializa caso seja configurado, não é o caso, porém de qualquer forma o ip fixo é setado conforme os dados declarados no escopo deste arquivo (ver fig 10) , ao final do processo caso não haja falhas os dados da conexão serão printados caso esteja adicionado um terminal virtual na simulação do circuito (a partir da linha 121)

Figura 4 — Busca por DHCP e utilização de IP fixo

```
93    if (ether.begin(sizeof Ethernet::buffer, mymac) == 0)
94    {
95      Serial.println( "Failed to access Ethernet controller");
96    }
97    else
98    {
99      Serial.println("Ethernet controller access: OK");
100   }
101   ;
102
103   #if STATIC
104     Serial.println( "Getting static IP.");
105   if (!ether.staticSetup(myip, gwip)){
106     Serial.println( "could not get a static IP");
107     while (true){
108       blinkLedOrRele(1, 500, 100, false); // blink forever to indicate a problem
109     }
110   }
111   #else
112
113   Serial.println("Setting up DHCP");
114   if (!ether.dhcpSetup()){
115     Serial.println( "DHCP failed");
116     while (true){
117       blinkLedOrRele(1, 500, 100, false); // blink forever to indicate a problem
118     }
119   }
120   #endif
121   // printa no terminal serial o ip conectado.
122   ether.printIp("My IP: ", ether.myip);
123   ether.printIp("Netmask: ", ether.netmask);
124   ether.printIp("GW IP: ", ether.gwip);
125   ether.printIp("DNS IP: ", ether.dnsip);
126
```

Fonte: O autor (2022) Campos.

Ainda é preciso informar o IP do servidor remoto, este ip será usado para montar um watch-dog, forçando o reinício automático do roteador/pon por uma quantidade “n” vezes caso haja ausência de conexão no teste de ping, a biblioteca Ethercard possui métodos prontos para teste de ping.

Figura 5 — Declarações de IP do servidor remoto

```
127     // Definindo ip do servidor remoto
128     ether.parseIp(ether.hisip, "192.168.1.23");
129    // identificar outros servidores pingando nosso dispositivo
130     ether.registerPingCallback(gotPinged);
131     timer = -9999999; //cronometro iniciado
```

Fonte: O autor (2022) Campos.

4.3.3 **Loop**

Inicialmente adicionaremos variáveis importantes , as primeiras (linhas 137, 138 e 139) são para controle do ping, acumuladores de tentativa de conexão que são usados como parâmetro para ação automática de reiniciar o roteador/pon no lado do cliente, em seguida a variável “char_arr” é um ponteiro para um array de caracteres, esta variável será utilizada em uma função fora do loop, as variáveis “len” e “pos” são utilizadas para identificar informações relacionadas à requests recebidas, sendo “len” para o ping e “pos” para chamadas http como no exemplo abaixo (linha 257), o exemplo identifica requests do tipo GET e path /?results, ao receber essa request a função “server” é chamada, e passados todos os parâmetros de telemetria, a função envia no tipo json.

Imagem 3 — Declaração de variáveis e rota de request “results”

```
137   int pingControl = 0;
138   int pingReboot = 0;
139   bool tentativaDePing = true;
140   char* char_arr; // array de caracteres para pagina web
141
142   word len = ether.packetReceive();
143   word pos = ether.packetLoop(len);
144
145   // verifica comandos recibidos via tcpIp, tipo GET
146   resAction(pos);
147
148   // liga lux de ponto automaticamente
149   if(digitalRead(digitalIn01)){
150     digitalWrite(releOut02, HIGH);
151   }
152   if(!digitalRead(digitalIn01) && digitalRead(releOut02)){
153     digitalWrite(releOut02, LOW);
154   }
155
156   // envia os dados de telemetria
157   if(strstr((char *)Ethernet::buffer + pos, "GET /?results") != 0) {
158       server(
159         // leitura de sensor de voltage ate 75v
160         calculaCorrent(analogRead(current01)), calculaCorrent(analogRead(current02)), calculaCorrent(analogRead(current03)),
161         // leitura de sensor de corrente 20A // 12 - 24 volts DC
162         calculaTensao(analogRead(volts01)), calculaTensao(analogRead(volts02)), calculaTensao(analogRead(volts03)),
163         //leituras de entrdas digitais
164         digitalRead(digitalIn01), digitalRead(digitalIn01) ,digitalRead(digitalIn01),
165         //verificação do stado dos reles
166         digitalRead(releOut01),digitalRead(releOut02),digitalRead(releOut03),digitalRead(releOut04),
167         digitalRead(acStatus) // sensor de tensao alternada
168         ,char_arr // arrey de caracteres
169         );
170   }
```

Fonte: Campos.

A função abaixo é montar manualmente o header e body com json, isso facilita a leitura e descrição dos dados, já que usamos TypeScript no dashboard.

Figura 6 — Função que envia os dados de telemetria no formato Json

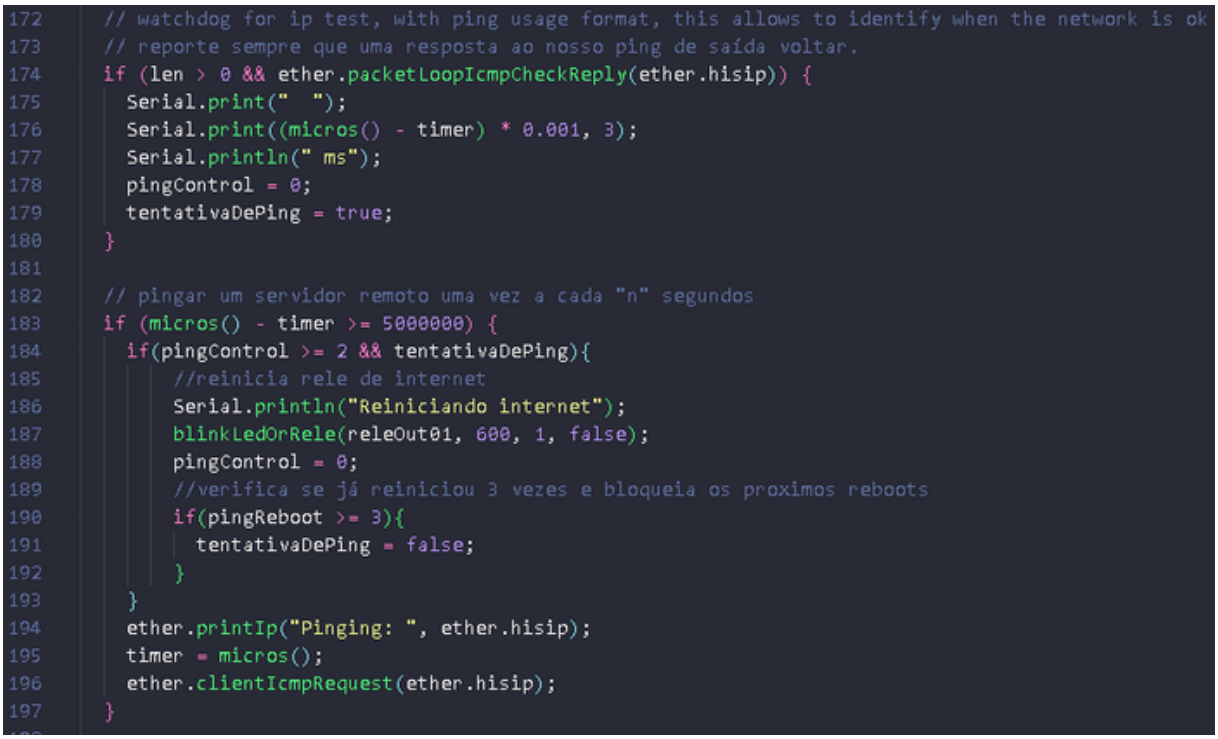
```
233  void server(
234    float volt01, float volt02, float volt03,
235    float amp01, float amp02, float amp03,
236    int dig01, int dig02, int dig03,
237    int rele01, int rele02, int rele03, int rele04,
238    int ac, char * msdg
239    ){
240
241     String page = "HTTP/1.1 200 OK\r\n"
242      "Content-Type: application/json\r\n"
243      "Access-Control-Allow-Origin: *\r\n"
244      "Access-Control-Allow-Headers: Content-Type\r\n"
245      "Access-Control-Allow-Methods: GET\r\n"
246      "Retry-After: 1000\r\n"
247      "\r\n"
248      "{";
249
250      page = String(page +
251        "\"voltimetro\":{" +
252            "\"volt01\":" + volt01 +
253            ",\"volt02\":" + volt02 +
254            ",\"volt03\":" + volt03 +
255        "},\"amperimetro\":{"+
256            "\"amp01\":" + amp01 +
257            ",\"amp02\":" + amp02 +
258            ",\"amp03\":" + amp03 +
259        "},\"digitalIn\":{" +
260            "\"dig01\":" + dig01 +
261            ",\"dig02\":" + dig02 +
262            ",\"dig03\":" + dig03 +
263        "},\"acIn\":{" +
264            "\"acStatus\":" + ac +
265        "},\"releStatus\":{" +
266            "\"rele01\":" + rele01 +
267            ",\"rele01\":" + rele02 +
268            ",\"rele01\":" + rele03 +
269            ",\"rele01\":" + rele04 +
270        "}}  "
271        );
272
273      msdg = &page[0];
274
275      int s = page.length();
276
277      memcpy(ether.tcpOffset(), msdg, s);
278      ether.httpServerReply(s - 1);
279  }
280
281  String converter(uint8_t *str){
282      return String((char *)str);
283  }
```

Fonte: O autor (2022) Campos.

4.3.4 **Watch-dog**

O primeiro validador verifica se existe resposta para nossa tentativa de ping e caso exista, mantém o controle de tentativas de ping igual a zero, já o segundo envia um ping de tempos em tempos (linha 183) e aumenta o valor do contador em +1, caso o contador já esteja maior ou igual a 3 o roteador é reiniciado, é bom lembrar que caso a rede caia e o roteador for iniciado e não possibilitar a conexão o hardware ficará travado, precisa de melhorias nesse sistema.

Figura 7 — Verificação de resposta e envio de pings com watch -dog



```
172   // watchdog for ip test, with ping usage format, this allows to identify when the network is ok
173   // reporte sempre que uma resposta ao nosso ping de saída voltar.
174   if (len > 0 && ether.packetLoopIcmpCheckReply(ether.hisip)) {
175     Serial.print("  ");
176     Serial.print((micros() - timer) * 0.001, 3);
177     Serial.println(" ms");
178     pingControl = 0;
179     tentativaDePing = true;
180   }
181
182   // pingar um servidor remoto uma vez a cada "n" segundos
183   if (micros() - timer >= 5000000) {
184     if(pingControl >= 2 && tentativaDePing){
185         //reinicia rele de internet
186         Serial.println("Reiniciando internet");
187         blinkLedOrRele(releOut01, 600, 1, false);
188         pingControl = 0;
189         //verifica se já reiniciou 3 vezes e bloqueia os proximos reboots
190         if(pingReboot >= 3){
191           tentativaDePing = false;
192         }
193     }
194     ether.printIp("Pinging: ", ether.hisip);
195     timer = micros();
196     ether.clientIcmpRequest(ether.hisip);
197   }
```

Fonte: O autor (2022) Campos.

4.3.5 **Detalhes do projeto**

Algumas outras variáveis estáticas para detalhes do projeto foram implementadas, como a variável page[ ], trata se de uma constante na forma de array de caracteres, herança de exemplos implementados contidos no repositório do código fonte da biblioteca Ethercard, eu usei com modificações necessárias para enviar como response de comandos recebidos:

Quadro 1 — Response de comandos

Response contendo header e body
char const page[ ] PROGMEM = "HTTP/1.1 200 OK\r\n" "Content-Type: application/json\r\n" "Access-Control-Allow-Origin: *\r\n" "Access-Control-Allow-Headers: Content-Type\r\n" "Access-Control-Allow-Methods: GET\r\n" "Retry-After: 1000\r\n" "\r\n" "{\"Comand\": \"ok\"}" ;

Fonte: O autor (2022).

Algumas outras funções também foram implementadas e não discutidas no item 4.3, são as seguintes:

Quadro 2 — Lista de funções

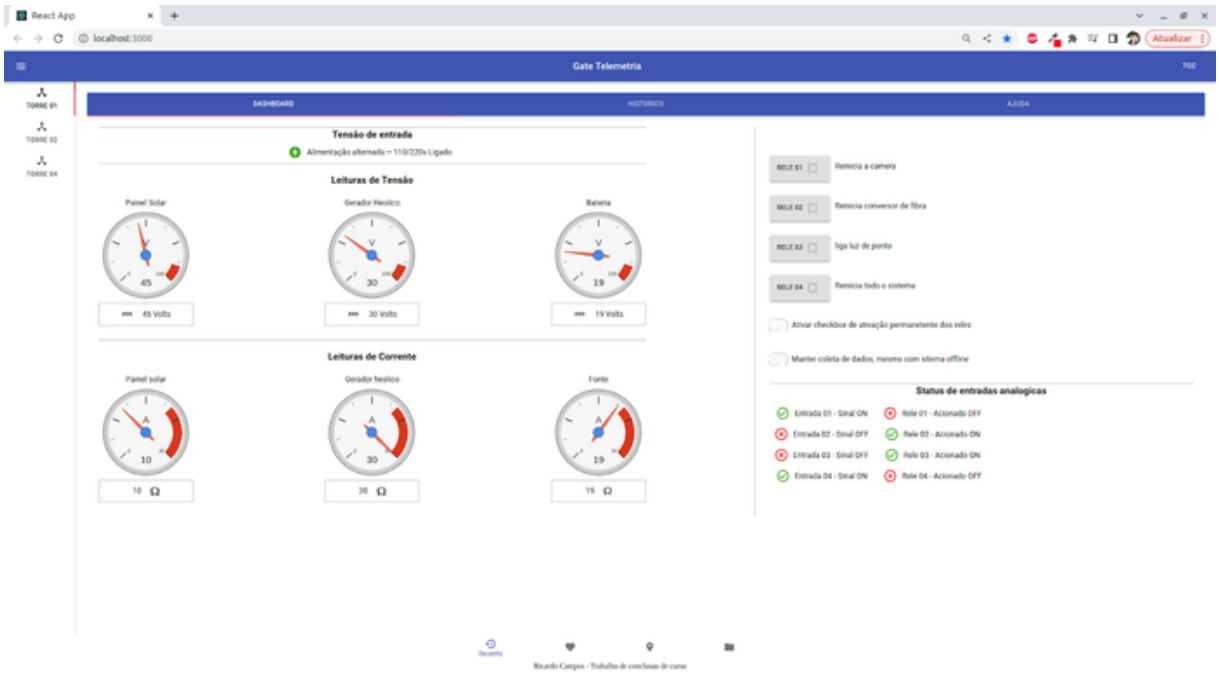
Item	Função	Descrição
01	reAction()	Faz a verificação das requests do tipo GET, a verificação analisa o parâmetro passado na url e executa o comando desejado
02	converter()	Converte um dado uint8_t em String
03	server()	Envia os dados de telemetria no formato JSON
04	comandResult()	Envia uma response com status code 200 quando executa um comando remoto
05	calculaCorrent()	Calcula a corrente de cada entrada analógica correspondente.
06	calculaTensao()	Calcula a Tensão de cada entrada analógica correspondente.
07	blinkLedOrRele()	ativa e desativa de forma intermitente ou momentaneamente um led ou relé

Fonte: O autor (2022).

5 SOFTWARE PARA DASHBOARD

O hardware que projetamos tem como seu principal objetivo, apresentar dados de telemetria e não menos importante, de forma secundária, a função de controle através comandos remotos, ambas essas funcionalidade precisam de uma interface, de preferência que possa visualizar e controlar diversos pontos onde o hardware é implantado, para isso projetamos uma interface simples, porém funcional, com botões para comandos e dashboard para visualização dos dados de telemetria, para melhor visualização vamos apresentar a tela por setores, nos próximos subcapítulos.

Figura 8 — Layout do dashboard.



Fonte: O autor (2022) Campos.

5.1 MÉTODOS PARA CONSTRUÇÃO DO CLIENT

Para a construção do software, utilizamos a tecnologia React um framework de javascript, porém o código foi escrito em Typescript, para os mostradores “analogicos” estamos usando google charts, uma biblioteca disponibilizada pelo google e totalmente compatível com React e open source assim como todas os frameworks, bibliotecas e ides utilizadas.

5.2 LAYOUT

Para tela, não trabalharemos com algo complexo, como o foco do projeto é o hardware e o firmware, não vamos entrar nos detalhes do software mas será disponibilizado no meu repositório do github apresentado no final deste projeto. para um projeto mais elaborado as regras de negócio e design do layout exigem um estudo à parte. Esse projeto foi pensado para receber dados e controlar múltiplos dispositivos , e possui um menu lateral esquerdo onde é possível mudar de dispositivo em torre 01, torre 02 e torre 03.

5.2.1 **Mostradores de grandezas elétricas**

Na parte superior adicionamos um label apresentando a tensão de entrada na forma de ligado e desligado, quando ligado (representado na figura 19) o símbolo em formato de raio, fica verde a apresenta a seguinte mensagem: “Alimentação alternada ~ 110/220v Ligado”, quando o sensor não identifica tensão o símbolo fica vermelho com a mensagem : “Alimentação alternada ~ 110/220v Desligado”. Para os mostradores utilizei uma representação “analógica”, sendo os três superiores para tensão, na ordem “painel solar”, “gerador eólico" e “bateria”, para os inferiores, mostradores de corrente, na ordem “Painel Solar”, “Gerador eólico" e “Fonte”, os mostradores são atualizados em tempo real, com um get de dados a cada minuto, as descrições que aparecem e a ordem estão fixas, mas pode ser adicionado uma tela de gerenciamento para nomear corretamente cada label, para utilizar em outras aplicações, a ordem dos mostradores, a zona de risco (em vermelho no mostrador) e níveis máximos também podem ser configuráveis caso seja implementado, esse projeto é escalável devido essas possibilidades.

Figura 9 — Mostradores de tensão e corrente e indicador de energia fornecida pela concessionária

Fonte: O autor (2022) Campos.

5.2.2 **Botões e informações das entradas e saídas digitais**

Começando com os botões, cada clique, no botão liga e desliga pois dispara o modo pulso, porém caso a chave “ativar checkbox de ativação permanente dos relés" (parte central da figura 20) é possível clicar no checkbox (quadradinho selecionável localizado na parte interior de cada botão, quando selecionado ele muda de cor e se mantém selecionado até um novo click), quando o checkbox está selecionado, o botão muda de função e casa clique mantém o relé correspondente ativado, esta função é útil para controle remoto de ações que precisam permanecer ligadas por determinado período, neste caso a maioria das ações está focada em reiniciar o dispositivo, por isso a função pulsar está ativada em primeiro plano. Na parte central está localizado outro interruptor “Manter coleta de dados, mesmo com o sistema off-line, isso permite que o servidor client continue a obter os dados de telemetria mesmo quando o sistema não esta sendo acessado, esses dados podem ser usados para aprimoramento dos produtos, gráficos de desempenho, estudo de dimensionamento de equipamentos de micro geração de energia conforme cada região utilizada. Por último na parte inferior apenas mostradores do estado dos relés,

quais estão ligados e desligados no momento da visualização, além disso é possível ver o estado de entradas digitais, para estado Falso o símbolo fica vermelho com um “x” e mensagem “sinal OFF” e “Acionado OFF”, para "true", símbolo verde e mensagem “Sinal ON” e “Acionado ON”, cada dado utilizado é obtido diretamente do firmware apresentado no capítulo 4 item 4.3.3 Loop.

Figura 10 — Botões, estado dos relés e portas de entrada digital .

Fonte: O autor (2022) Campos.

5.2.3 **Outras considerações**

Por último é possível ver uma barra de topo com as abas “dashboard” que é essa que comentamos no momento, aba "histórico" e “ajuda” que não foram implementadas, a ideia é tem uma aba com representação do histórico de dados trazidos através de um filtro de intervalo de data, para além de utilizar para estudos conforme já foi comentado, também visualizar quais as últimas leituras da telemetria

em caso de pane no local.

6 **SIMULAÇÃO**

A simulação é a parte mais importante do nosso projeto, pois permite ver a efetividade dos circuitos de sensores, de potência e digital além da funcionalidade do firmware, para simulação utilizamos o proteus, sendo esse o único software ou recurso utilizado neste projeto que não é open source , porém possui licença para testes, também utilizamos um software gratuito que já defasado, porém funciona ao nosso propósito, o Wincamp , totalmente compatível com a biblioteca no enc28j60 que utilizamos no proteus, e com windows 10 (esse software e passivo). Após o circuito montado e ip configurado no simulador do enc28j60 é só configurar o firmware no simulador do Atmega2560, o arquivo do firmware compilado deve ter a extensão “.efl” o arquivo gerado pelo compilador fica como: \firmware.elf. Após iniciar a simulação utilizando os recursos do proteus é possível utilizar os recursos do servidor web disponibilizado pelo hardware e software que projetamos diretamente pelo navegador já que todas as requisições são do tipo GET e nosso hardware software não possui recursos adicionais de segurança.

6.1 MÉTODOS DE TESTE

O principal método de teste é a adição de geradores de tensão e corrente variáveis, utilizamos esse recurso para visualizar a visualização de dados de telemetria em tempo real, no software dashboard, variando as tensões que simulam painel solar, bateria e gerador eólico, para simular os leitores de tensão adicionamos lâmpadas em paralelo, conforme se liga as lâmpadas a corrente aumenta e resultado pode ser visto na tela que apresentamos no capítulo 5, ou fazendo um get diretamente no navegador, podemos obter os dados via “json” na seguinte estrutura :

Quadro 3 — Response para get de dados (continua)

Response Json

{
"voltimetro":{
"volt01":24.98,
"volt02":0.00,
"volt03":0.00
},
"amperimetro": {
"amp01":23.95,

Quadro 3 — Response para get de dados (conclusão)

Response Json

"amp02":23.95,
"amp03":23.95
},
"digitalIn":{
"dig01":0,
"dig02":0,
"dig03":0
},
"acIn":{
"acStatus":1
},
"releStatus":{
"rele01":0,
"rele01": 0,
"rele01":0,
"rele01":0
}
}

Fonte: O autor (2022).

CONCLUSÃO

Os sistemas de monitoramento de focos de incêndio apresentavam inviabilidade do operação na maior parte dos caso pois estão instaladas em regiões de difícil acesso, matas fechadas ou locais remotos, por esse motivo até problemas muito simples tornavam cada ponto inoperante por tempo superior a uma semana, por esse motivo tornava inviável a implantação na maioria dos casos, porem com esse problema resolvido, o custo de implantação de pontos de monitoramento se justifica pois geram um custo extremamente menor no combate ao foco de incêndio se comparado a um incêndio de grandes proporções , pois além do valor financeiro ser elevado, um incêndio de pequena escala não atinge grandes áreas de fauna.

Inicialmente concluímos um levantamento de recursos para então prosseguir para o desenvolvimento do hardware e em seguida o firmware, após o firmware ser implantado e simulado algumas modificações no hardware foram necessárias, embora o objetivo da pesquisa fosse voltado para o hardware e firmware, desenvolvi um sistema client composto por um dashboard e botões para controle feito em React.js permitido visualizar os dados recebidos de forma mais intuitiva.

A metodologia usada para essa pesquisa foi a o desenvolvimento de um hardware e software próprio autoral, feito exclusivamente para esse problema, mas que pode ser modificado e escalado como previsto.

Para resolver o problema, o conjunto deste projeto formado por hardware, firmware e software client disponibiliza visualização de dados de telemetria em tempo real, dados armazenados de períodos passados e controle remoto do dispositivo, essas ações resolvem os problemas propostos no caso de uso apresentado no capítulo 1, o dispositivo faz ping a cada minuto no servidor, caso a resposta for vazia em três tentativas, reinicia automaticamente o roteador ou conversor de fibra utilizado, resolve o problema que é o mais recorrente no conjunto de monitoramento de focos de incêndios em área de mata preservada, caso a câmera trave, pode ser reiniciada remotamente pelo operador, além de outros três relés que também podem ser reiniciados, mantido ligado ou desligados. os dados armazenados auxiliam na melhoria do produto, pois é possível verificar a quantidade de energia gerada pelo painel solar e gerador eólico em diferentes épocas do ano e em diferentes regiões, isso permite dimensionar os equipamentos entendendo na prática o consumo e disponibilidade de energia gerada em cada um dos equipamentos, um exemplo seria analisar uma região, que de forma fictícia chove muito e venta mais (em média por ano e por época do ano), dependendo da demanda pode ser colocar um eólico mais efetivo e um painel solar menor de forma precisa. Monitorar o consumo também auxilia na manutenção preventiva, um

exemplo seria um gerador que gerava uma quantidade “n” de energia e depois de certo período a efetividade caiu pela metade (n/2) indicando problemas a serem resolvidos, antes que o sistema entre em colapso, monitorando esses dados é possível corrigir vários problemas preventivamente de forma que o sistema não fique parado, um sistema que para repentinamente pode demorar semanas para ser colocado novamente em atividade, já a manutenção preventiva tem uma janela de tempo maior, mesmo as mais urgentes onde algum equipamento começa apresentar defeitos antes de parar totalmente. O hardware ainda pode ser adaptado para uso em outros tipos de monitoramentos industriais e comerciais, um exemplo seria o monitoramento de níveis de combustível nos tanques de geradores de uma prefeitura, onde se possui diversos geradores espalhados em áreas urbanas e rurais, alguns locais o gerador é usado ou pode ser usado com mais frequência, outros não, assim a equipe consegue manter todos os geradores abastecidos, acompanhar o tempo que ficam ligados, ligar de vez em quando remotamente equipamentos que ficam muito parados e monitorar indicadores do motor como temperatura, pressão do óleo e outros, tudo isso mudando apenas a divisão de valor utilizada nas entradas analógicas diretamente no firmware, isso torna o hardware muito versátil inclusive para outras aplicações.

7 **CONTINUAÇÃO DO PROJETO**

Como o projeto é didático e vou adicionar uma licença open source para hardware e software, disponível no github (https://github.com/ricardocvel, outras referencias abaixo) , vou fazer algumas breves sugestões de melhorias, para ambas as plataformas e servirem como projetos de estudo futuro para quem interessar

7.1 HARDWARE

Para o hardware a principal mudança é separar as tensões de 5v para interna e externa, pois da forma que está, alguns optoacopladores não estão isolando o circuito digital, um dos casos é o optoacoplador que aciona o rele, ambos os lados do circuito possuem a mesma alimentação. Outra mudança é a implementação de um novo chip para rede ethernet, esse que usamos é o mais simples que encontrei, os melhores chips mais próximos a este dão um salto considerável na complexidade. por último, o atmega2560 possui diversas portas sobrando, propositalmente, isso porque a ideia é implementar cinco botões e um display para configuração além de adição de novos sensores.

7.2 FIRMWARE

O firmware pode ser melhorado adicionando comentários, se necessário se livrar do framework do Arduino e adicionado novas rotinas de controle, adição de um sistema de autenticação para os clients que consomem a api.

7.3 SOFTWARE CLIENT

A principal mudança é adicionar um sistema de salvamento de histórico, o menu já foi deixado no lugar bastando apenas implementar uma tela para visualização e geração de relatórios, melhorias no layout, responsividade, a versão atual deste software está disponível no meu github.

REFERÊNCIAS

CAMPOS, Ricardo . **Repositório de imagens do firmware autoral** : imagens do código . github. Disponível em: https://github.com/ricardocvel/FirmwareTccTelemetria/tree/master/arq/firmware. Acesso em: 8 out. 2022.

CAMPOS, Ricardo. **Firmware**: Repositório de código. github. Disponível em: https://github.com/ricardocvel/FirmwareTccTelemetria/. Acesso em: 17 out. 2022.

CAMPOS, Ricardo. **Repositório de imagens do projeto de hardware**: Desenvolvido no Protheus. github. img 01 p. Disponível em: https://github.com/ricardocvel/FirmwareTccTelemetria/tree/master/arq/hardware. Acesso em: 8 out. 2022.

CAMPOS, Ricardo. **Software dashboard**. github. Disponível em: https://github.com/ricardocvel/dashboardClientTelemetriaTcc/blob/master/README.md/. Acesso em: 9 out. 2022.

LABCENTER . **Proteus**. labcenter.com. Disponível em: https://www.labcenter.com/. Acesso em: 9 out. 2022.

MICROCHIP. **Ethernet PICtail™ Plus Daughter Board**: Datasheet ENC28J60 QFN. microchip.com. Disponível em: https://www.microchip.com/en-us/development-tool/AC164123. Acesso em: 8 out. 2022.

PLATFORMIO . **Complemento para vscod, permite desenvolver e compilar diretamente na IDE**. Platformio . Disponível em: https://platformio.org/. Acesso em: 4 out. 2022.

WINMPCAP . **Winpcap** . winpcap.org. Disponível em: https://www.winpcap.org/. Acesso em: 9 out. 2022.

YOSEENIR. **Prevenção de incêndios florestais**: sistema de monitoramento por imagem térmica infravermelha. yoseenir.com. 2022. 1 p. Disponível em: https://www.yoseenir.com/2717.html. Acesso em: 8 out. 2022.

GLOSSÁRIO

Expressão	Descrição
github	um dos mais populares Repositórios de código
client	Expressão usada para referenciar software que consome uma api, geralmente a parte que contem interface com usuário ou front-end
Arduino	plataforma de aprendizado open-sourse, interface de hardware para chips Atmega
open source	Significa código aberto de utilização livre.
ping	é um utilitário que usa o protocolo ICMP para testar a conectividade entre equipamentos.
dashboard	Painel de informações e indicadores, pode ter gráficos, relatórios e etc.
enc28j60	chip controlador ethernet.
checkbox	caixa de seleção html, usada para selecionar uma opção.
framework	Caixa de ferramentas, composta de conjunto de bibliotecas e funções , para uma linguagem de programação.
label	Etiqueta html usada para dar informações a um elemento na tela.
array	Em programação é uma lista de elementos com um mesmo tipo de dados, por exemplo, um array de inteiros : [0, 1, 2, 3] .
Watch-dog	sistema de automação realizado de forma autônoma e desencadeado por uma determinada ação.
print	imprimir, em programação utilizado para referenciar a impressão de dados na tela independente do tipo.
microcontrolador	é um pequeno computador num único circuito integrado o qual contém um núcleo de processador, memória e periféricos programáveis de entrada e saída.
datasheet	folha de dados ou folha de especificações.