CEEL - ISSN 2596-2221
Universidade Federal de Uberlândia
25 a 29 de novembro de 2019

MONITORAMENTO REMOTO DE UM AMBIENTE CLIMATIZADO

Gabriel Felipe Vieira de Sousa*[1], Aniel Silva de Morais[1]

[1]FEELT – Universidade Federal de Uberlândia

***Resumo* – A internet vem modificando a percepção que temos do mundo. A Internet das Coisas (IoT – Internet of Things) tem permitido e consolidado tais modificações. Seja em ambientes corporativos, indústrias e até mesmo residências, o controle de sensores tem se tornado realidade em grande parte do mundo. Assim, em virtude do rápido desenvolvimento de equipamentos que se conectam via internet, torna-se promissor a implementação de sistemas de controle remoto autônomo.**

Por conseguinte, o presente artigo tem por finalidade desenvolver um sistema para controle de temperatura via Ethernet utilizando um microcontrolador ATMEGA328.

***Palavras-Chave* – Controle Remoto, Controle de Temperatura, Ethernet, Internet das Coisas, Microcontrolador.**

REMOTE MONITORING OF A CLIMATIZED ENVIRONMENT

***Abstract* - The internet has been changing our perception of the world. And the Internet of Things (IoT) has allowed and consolidated such modifications. Whether in corporate environments, industries and even homes, sensor control has become a reality in much of the world. Thus, due to the rapid development of equipment that connects via the Internet, it becomes promising the implementation of autonomous remote control systems.**

Therefore, this article aims to develop a system for temperature control by Ethernet using an ATMEGA328 microcontroller.

***Keywords* - Remote Control, Temperature Control, Ethernet, Internet of Things, Microcontroller.**

I. INTRODUÇÃO

Com o enorme avanço no campo tecnológico, uma ampla variedade de sistemas de medição de temperatura estão sendo usados em diferentes condições atmosféricas com alta precisão e exatidão, visto que a temperatura é um parâmetro crítico em indústrias ou em laboratórios [1]. Além disso, a evolução da tecnologia da informação abriu as portas para muitos casos que anteriormente eram considerados impossíveis ou apenas fictícios. Ao longo dos anos, os celulares, tablets, automóveis, alcançaram a ascensão da tecnologia "inteligente" que consumiu o mercado e tornaram-se o novo padrão nas indústrias [2]. Desta forma, pode-se observar que tais aparelhos pessoais tornaram-se um ponto de partida para tarefas automatizadas, seja para facilitar a vida de usuários ou para economia de tempo, por exemplo. Assim, ações que apenas podiam ser vistas em filmes, ficções, histórias, hoje são comumente aplicadas ao dia a dia. Visto que por trás desta alta tecnologia possui o árduo trabalho de engenheiros, programadores, instaladores, técnicos, entre outros profissionais, que buscam sempre aplicar tal avanço.

Essa nova tecnologia viabilizou o controle e monitoramento à distância de dispositivos de segurança, sistemas de calor, ou quaisquer outros sistemas que necessitam de monitoramento constante, rápido e preciso. Em virtude da viabilidade do desenvolvimento de tecnologias inteligentes e acessíveis, torna-se prático o desenvolvimento de sistemas controlados via internet [7]. Este controle à distância visa a aproximação do usuário mesmo que esteja longe do local a ser controlado, e que por vários dispositivos distintos possa monitorar, controlar e aplicar funções ao mesmo. Tais parâmetros são positivos para a segurança, comodidade e praticidade tanto do local que sofrerá a aplicação quanto do usuário, além de cada dia mais ser aplicado em diferentes áreas tornando-se “comum” e mais próximo da realidade.

Dessa forma, este trabalho almeja desenvolver um sistema autônomo, em que seja possível monitorar a temperatura e também fazer intervenções no ambiente a ser supervisionado. O usuário terá acesso a uma página da internet, que será desenvolvida em HTML, com o auxílio de um módulo de internet, essa página fornecerá informações do local monitorado, e ainda também permitirá realizar alterações de parâmetros. O sistema será composto de um microcontrolador ATMEL ATMEGA328 que estará acoplado a um módulo Ethernet, que juntos à outros componentes vistos a seguir, serão programados de modo a oferecer todas as ações necessárias ao projeto

*gabrielfelipe@ufu.br

II. SISTEMA DE AQUISIÇÃO DE DADOS

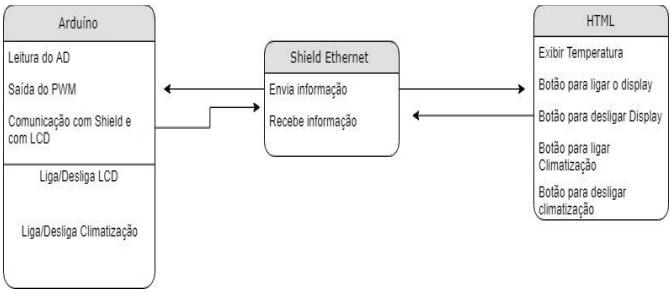
O projeto foi dividido em 4 partes: solução do problema, esquemático do hardware, implementação do código e resultados da implementação.

A. Solução do Problema

Para o desenvolvimento do projeto, foi utilizado o microcontrolador ATMEL 328, como dito anteriormente.

O sistema fará a aquisição de dados de temperatura, e enviará a informação para uma página criada em HTML através do *shield* ethernet, onde o usuário poderá fazer manipulações como: ligar ou desligar a refrigeração do ambiente, ligar ou desligar o display e verificar qual o estado atual do ambiente, sendo eles: ambiente climatizado, ambiente com climatização desativada, ou alerta de incêndio. Quando o sistema encontra-se com a climatização ativada, a saída será um sinal de PWM que acionará o cooler, no qual este sinal é proporcional à temperatura do local, ou seja, quanto maior a temperatura, mais refrigerado deve ser o ambiente. A Figura 1 ilustra o funcionamento do sistema de maneira simplificada:

Figura 1: Diagrama de Classes do Software.



Visto que a solução do programa requer um baixo processamento e as tarefas são bem definidas toda a programação, foi feita em *Bare-Metal* utilizando funções ANSI C. Além do mais, foi feito o uso de boas práticas em programação, afim de garantir a legibilidade e reprodutibilidade do software.

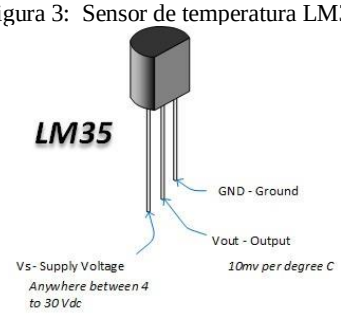
A elaboração do software consiste na manipulação de registradores para a implementação de recursos de I/O, conversor analógico digital (A/D), display alfanumérico e PWM. Tal manipulação permite obter um maior domínio de hardware e software, garantindo um projeto otimizado, dessa forma, faz-se necessário conhecer cada componente a ser utilizado e seu respectivo funcionamento:

1) *Arduino UNO*: cujo principal componente de placa é o microcontrolador ATMEL ATMEGA 328, de 8 bits, com arquitetura RISC. Ele opera com um cristal de 16Mhz. Possui 28 pinos, sendo 23 pinos que podem ser usados como entrada/saída digital, 6 pinos que podem ser usados como entrada analógica, e 6 pinos que podem ser usados como saída PWM.

Figura 2: Arduíno UNO.

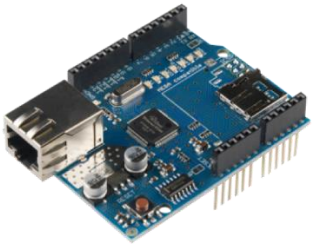
2) Sensor de Temperatura LM35: Para a coleta de dados de temperatura, foi utilizado o sensor LM35 (vide Figura 3). Esses sensores variam linearmente a tensão de saída e a temperatura Celsius, onde a cada 10mV de tensão de saída correspondem à um grau na escala Celsius.

Figura 3: Sensor de temperatura LM35.



3) Ethernet Shield Arduino – W5100: É um módulo desenvolvido para aplicações que integram a internet ao Arduino. Sua comunicação é serial síncrona, do tipo SPI.

Figura 4: Ethernet Shield Arduino.



*4) Motor 5V (Cooler) e Mosfet IRF840: S*erá utilizado um cooler de 5V que será responsável por refrigerar o ambiente. Ele funcionará através de um sinal de PWM (*Pulse Width Modulation*).

Foi necessário utilizar um circuito composto por uma fonte externa de 5V, e um mosfet IRF840, pois o arduino não pode fornecer corrente suficiente ao cooler.

5) Buzzer: Será colocado no ambiente onde fará as medições e o monitoramento, ele será responsável a alertar o usuário caso o ambiente em questão esteja em um início de incêndio.

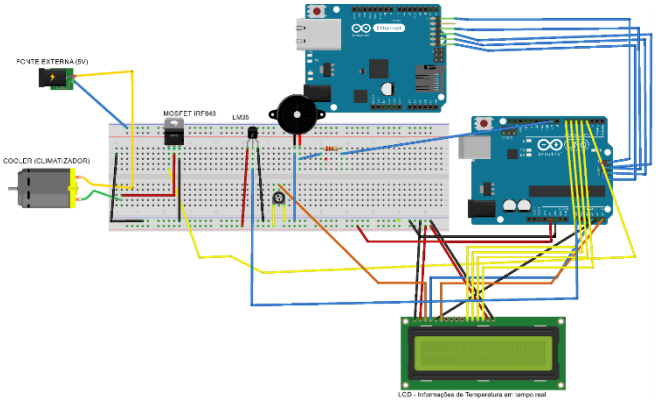
6) Display alfanumérico: Será utilizado ainda um visor LCD 16x2, tal componente será responsável por informar ao

usuário que estiver presente no local monitorado, sobre as condições atuais de temperatura, quando o mesmo estiver ligado, sem se fazer necessário acessar a internet.

B. Esquemático do Hardware

O esquema de ligação dos componentes que aqui foram apresentados podem ser vistos na figura 5.

Figura 5: Esquema de ligação dos componentes do sistema.



C. Implementação do projeto

A implementação do software foi feita por partes, sendo elas: Integração do A/D com o sinal PWM, Implementação da biblioteca para o LCD e envio de informações para o mesmo, Comunicação do módulo de Internet com o microcontrolador, e por fim a integração de todo o projeto.

1) Implementação do A/D e PWM: Neste primeiro estágio foi implementado o seguinte código como mostra a figura 6:

Figura 6: Definição da função PWM.

```
void Pwm_Init(void)
{
  DDRD |= (1 << DDD3);
  TCCR2A |= (1 << COM2B1);
  TCCR2A |= (1 << WGM21) | (1 << WGM20);
  TCCR2A |= (1 << WGM22);
  TCCR2B |= (1 << CS21);
}
```

Inicialmente, o bit 3 do registrador D é definido como saída, entretanto, como este registrador não é um registrador específico de PWM, é necessário ativar a função OC2B. Para tal fato, foi consultado o manual do fabricante do ATMEL para se informar quais bits devem ser *setados* no registrador TCCR2A.

Em seguida, foi configurado o conversor A/D (vide Fig. 7). Optou-se por utilizar a porta A0 do Arduino (PC0 do microcontrolador), pois nesse caso não se faz necessário *setar* nenhum bit do MUX[3:0] de acordo com o manual do fabricante[9].

Todas as definições foram feitas em uma biblioteca nomeada de "*ad.c*", tais como: o *prescaler*, a frequência e inicio do A/D. Foi utilizado para este conversor A/D, uma frequência de 128KHz, visto que o Arduino UNO opera com frequências entre 50KHz e 200KHz

Figura 7: Biblioteca "ad.c".

```
#include <avr/io.h>

const unsigned char PS_128 = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);


void Ad_Init(void)
{

   ADCSRA |= PS_128;
   ADMUX |= ( 1 << REFS0 );

  ADCSRA |= ( 1 << ADATE );
  ADCSRA |= ( 1 << ADEN );
  ADCSRA |= ( 1 << ADSC );
}
```

2) Implementação do LCD: Como mencionado anteriormente, o LCD será utilizado para exibir a temperatura instantânea do local. Nele o pino 1 é ligado ao GND, o pino 2 ao VCC e o pino 3 é ligado a um potenciômetro para fazer ajustes de contraste do display.

Os pinos 4, 5 e 6 são usados como variáveis de controle, sendo elas: RS (*register select*), RW (*read/write*) e EN (*enable*). Esses pinos foram ligados no registrador C, conforme a figura 8. Os pinos 11, 12, 13 e 14 do LCD foram conectados nas portas 4, 5, 6 e 7 do Arduíno (PD4, PD5, PD6, PD7 do microcontrolador). O LCD envia e recebe dados de 8 bits, pois são caracteres. Optou-se enviar e receber dados de 4 em 4 bits. Ou seja, um dado de 8 bits é mandado e recebido de forma segmentada para poupar as portas digitais do Arduino. A partir dessas informações, foi criada uma biblioteca denominada "*lcd.c*", responsável por fazer a definição de cada pino e enviar as informações ao LCD:

Figura 8: Definições das portas do LCD

```
#include <avr/io.h>
#include <util/delay.h>

#define DataBus PORTD
#define DataDir_DataBus DDRD
#define ControlBus PORTC
#define DataDir_ControlBus DDRC
#define EN 5
#define RW 4
#define RS 3
```

Como mostrado na figura 8, foi feita as definições dos registradores como variáveis globais, pois caso haja necessidade de trocar a pinagem, basta fazer modificações nesta sessão.

O registrador DDRD configura os pinos como input ou output. Ele determina se o microcontrolador está recebendo

ou mandando um caractere. Ele foi denominado *DataDir_DataBus*, pois determina a direção dos dados (Arduino->LCD ou LCD->Arduino). A mesma coisa ocorre com o registrador DDRC para os pinos de controle. Foi denominado *DataDir_ControlBus*.

O display alfanumérico exige configurações a serem feitas na inicialização, essas configurações foram feitas em código hexadecimal de 8 bits, para isso foram definidas *flags* de inicialização conforme o datasheet do *LCD Hitachi HD44780*. Após as definições e as flags, foram criadas as funções para mandar e receber comandos.

Inicialmente, para enviar o comando é necessário enviar um pulso ao *enable*. Para isso torna-se necessário criar uma função responsável estritamente por pulsar o *enable*. Da mesma maneira, torna-se necessário criar funções para enviar um comando, e enviar uma *string*. Tais funções configuram inicialmente os pinos de comando (RS, RW e E), tendo em vista que, para mandar um comando ao LCD, RS e RW devem estar *setados* em *LOW*, e logo em seguida o *eneble* deve receber um pulso. A função para enviar uma *string* recebe uma string e manda um caractere por vez para o LCD.
Além dessas funções é preciso criar uma função para iniciar o LCD, essa função primeiramente *seta* todos os pinos do LCD como saída, em seguida, deve-se *setar em LOW* os pinos de controle para enviar comandos ao LCD. Esses comandos consistem em definir o modo como o LCD receberá informações, neste caso em 4 bits, a quantidade de linhas do LCD, e em seguida o *buffer* do display é limpo através de uma flag definida no início do código. A partir dessas configurações, é necessário apenas chamar a função de início do LCD no escopo da função principal, e em seguida, quando necessário utilizar a função para enviar uma *string* para realizar o envio de informações ao LCD.

3) Implementação da Página HTML :

Como descrito anteriormente, o microcontrolador fará comunicação com um Shield Ethernet W5100, o qual será responsável pelo envio e recebimento de informações, sendo elas: os comandos para ligar e desligar o LCD e o sistema de refrigeração, e também exibir a atual temperatura. Para isso foi criado um Servidor Web.

A comunicação entre o Arduíno e o Shield Ethernet é feita através do barramento SPI (*Serial Peripheral Interface*), e segue os conceitos de Mestre-Escravo, onde os pinos 10, 11, 12 e 13 do arduíno são usados da seguinte maneira: O pino 10 do arduíno é o *Chip Select*, que é ativo em nível baixo, ou seja, quando este pino recebe 0V, isto significa que o arduíno fará comunicação com o shield e vice e versa. O pino 11 do arduíno (MISO) é o *Slave Input*, que é responsável por enviar dados ao módulo de internet. O pino 12(MOSI), *Slave Output* é responsável por enviar a resposta do Shield ao Arduíno, e o pino 13, *Serial Clock*, atua simplesmente como o *clock*.

Para o desenvolvimento do *WebServer*, foram utilizadas as bibliotecas *Ethernet.h* e *Spi.h*, na qual a primeira atua possibilitando a conexão com uma rede local e a segunda na conexão do *Shield* com o Arduíno. A construção da página HTML foi feita através da função *client.println*, seguindo as normas da linguagem HTML. O recebimento dos comandos realizados pelo usuário é feito pelos comandos indicados na figura 9:

Figura 9: Funções em HTML responsáveis por criar os botões de comando no interior da página

```
client.println("<INPUT type=button value=Refrigeração_Automática
onClick=window.location='/?Lig_Air\'>");

client.println("<INPUT type=button value=Refrigeração_Desligada
onClick=window.location='/?Off_Air\'>");
```

Através disso, é definido um botão no interior da página criada. Caso o usuário aperte algum dos botões, a variável *Lig_Air ou Off_Air* receberá 1, e em seguida o programa fará a comparação como mostra a figura 10:

Figura 9: Comparação onde será ligado ou desligado o cooler.

```
if(readString.indexOf("?Lig_Air") >0)
        {
            OCR2B = ad; //  recebe o valor proporcional à
temperatura atual
        }

       else{
        if(readString.indexOf("?Off_Air") >0)
        {
         OCR2B = 0;
        }
```

Analogamente, foram feitos botões para ligar e desligar o display LCD.

D. Resultados Experimentais

Durante a implementação do projeto, alguns erros surgiram, tais como a falha na comunicação entre o LCD e o Arduino, e a configuração do PWM, fundamentalmente por se tratar de uma programação desenvolvida utilizando registradores. Com a agregação do Shield, algumas portas do Arduino ficaram inviáveis de utilização. Visto que ele utiliza o barramento SPI (através da porta ICSP) para se comunicar tanto com o cartão SD (que pode ser acoplado no mesmo) quando com o W5100. Diante disso, foi devido se atentar a algumas limitações, tendo em vista que:

- O pino PB2 (10 do uC) é usado para selecionar o W5100 e dessa maneira ele não pode ser utilizado para entrada/saída de informação;
- O pino PD4 (4 do uC) é usado para o cartão, e só pode ser usado se o slot SD não estiver ocupado;
- Embora não seja usado na maioria das bibliotecas de cartões SD, o A0 é conectado ao pino de proteção contra gravação (WP) do slot SD, e o A1 é usado para a verificação e detecção de cartão SD.

Devido às limitações, os pinos de comando do LCD foram ligados ao registrador DDRC. Ademais, os problemas

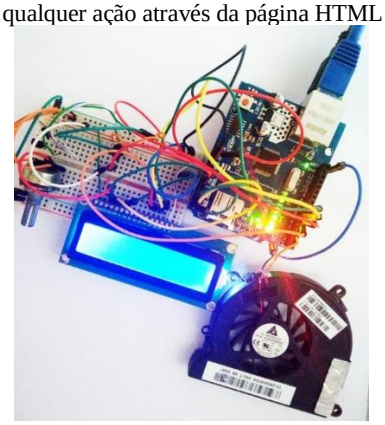
durante a implementação do software foram facilmente resolvidos.

Ao compilar o código, e acessar o endereço: 192.168.1.16 (que varia de acordo com a rede conectada), a seguinte página é aberta conforme mostra a Figura 11.

Figura 10: Página em HTML na qual fará interações com o usuário

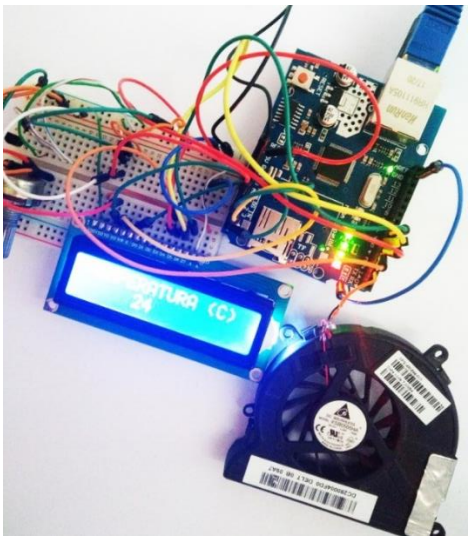
Neste momento inicial o LCD encontra-se desligado (vide Figura 12):

Figura. 11. Estado inicial do sistema desenvolvido antes de receber qualquer ação através da página HTML



Caso o usuário opte por visualizar a temperatura através no LCD, ele pode a qualquer momento fazer o acionamento do mesmo clicando no botão LIGAR_LCD (vide figura 13).

Figura 12: Estado do sistema após o usuário apertar em LIGAR_LCD



O usuário terá também as opções de manter a refrigeração automática, ou desligada, e os comandos são feitos de forma análoga aos comandos para ligar e desligar o LCD.

Figura 13. Sistema com a refrigeração ligada

III. CONCLUSÕES

Por meio desse projeto, foi possível realizar a implementação de um sistema de monitoramento de sensores via ethernet. Esta que, além do baixo custo, permitiu empregar a programação bare-metal em sua grande parte e ao mesmo tempo, robustecer os conhecimentos sobre o protocolo de comunicação SPI.

Os resultados da implementação foram satisfatórios e o protótipo não apresentou problemas de software durante sua execução.

Como futuros trabalhos, pretende-se fazer um controle PID de temperatura, aplicado à um ambiente real.

AGRADECIMENTOS

Os autores agradecem ao Laboratório de Automação, Sistemas Eletrônicos e Controle - LASEC, pela colaboração neste trabalho.

REFERÊNCIAS

[1] Chatterjee, S. Chatterjee and R. Gupta, "Arduino based real-time wireless temperature measurement system with GSM based annunciation," 2017 *International Conference on Communication and Signal Processing (ICCSP)*, Chennai, 2017, pp. 0840-0844.

[2] S. Sbîrnă, P. V. Søberg, L. S. Sbîrnă and M. Coşulschi, "Sensor programming and concept implementation of a temperature monitoring system, using Arduino as prototyping platform," 2016 *20th International Conference on System Theory, Control and Computing (ICSTCC)*, Sinaia, 2016, pp. 848-853.

[3] TIMMIS, H. Practical Arduino Engineering, 1a Ed., Technology in Action, 2011 ;

[4] Vincenzi, F. (2015). *Programação Bare Metal*. Em F. Vicenzi, "Boas Práticas de Programação de Sistemas Embarcados" (pp. 2-21). Uberlândia.

[5] Grusin, M. (06 de MARÇO de 2018). Serial Peripheral Interface (SPI). Fonte: SPARKFUN:

[6] Sacco, F. *Comunicação SPI* Fonte: Embarcados: https://www.embarcados.com.br/spi-parte-1/ Acesso em: 23 de junho de 2018.

[7] P. Agrawal and G. Chitranshi, "Internet of Things for monitoring the environmental parameters," *2016 International Conference on Information Technology (InCITe) - The Next Generation IT Summit on the Theme - Internet of Things: Connect your Worlds*, Noida, 2016.

[8] D. Wu, L. Likun and L. Yeli, "The design of data mutual conversion system between serial port and ethernet based on W5100," *2012 International Conference on Image Analysis and Signal Processing*, Hangzhou, 2012, pp. 1-5.

[9] Atmel, *Datasheet complete – Atmega328/P* . Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf . Acesso em: 12 de junho de 2018.