UNIVERSIDADE FEDERAL DE LAVRAS - UFLA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

CASTRO, LIMA E NEVES

TRABALHO DE ESTRUTURAS DE DADOS:

CADASTRO DE TEMPERATURA E LUMINOSIDADE EM UM SISTEMA EMBARCADO

Lavras

2018

DAVI HORNER HOE DE CASTRO
LUCAS ANTÔNIO LOPES NEVES
THIAGO LUIGI GONÇALVES LIMA

TRABALHO DE ESTRUTURAS DE DADOS

Trabalho apresentado como requisito parcial para aprovação na disciplina Estrutura de Dados do Departamento de Ciência da Computação, Universidade Federal de Lavras.

Prof. Dr. Joaquim Quinteiro Uchôa
Prof. Dr. Renato Ramos da Silva

Lavras
2018

RESUMO

Neste trabalho será criado um sistema de cadastro em um sistema embarcado. Este sistema será responsável por capturar dados como temperatura e luminosidade de um ambiente. Será utilizado um Arduino.

Palavras-chave: Cadastro; Arduino; Medições; Temperatura; Luminosidade.

ABSTRACT

In this project it will be created a register system in an embedded system. This system will be responsible of capturing temperature and luminosity levels of an ambient. It will use an Arduino.

Keywords: Register; Arduino; Measurements; Temperature; Luminosity.

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

A ideia principal do projeto é fazer um sistema de cadastro com luminosidade e temperatura usando para tal um microcontrolador. Este sistema pode ser usado para acompanhar uma determinada área e verificar as condições de temperatura e luminosidade. Um bom exemplo de uso real para este projeto seria em uma estufa onde pode-se checar se a temperatura e a iluminação estão ideais para as plantas que ali são cultivadas.

Neste projeto utilizaremos o modelo *Uno* da empresa italiana Arduino. Utilizaremos também um sensor *LM35* para medir a temperatura do ambiente e outro sensor *LDR* para medir a luminosidade.

As medições serão inseridas em primeiro momento em uma fila onde estas serão uma-a-uma inseridas numa *hash* de módulo 24. Representando assim a temperatura e a luminosidade em um período de 24 horas.

Ao fim do uso antes de desligar o Arduino salvará todas as informações de sua memória principal em um cartão de memória *micro SD.* Será possível acessar estas informações caso seja necessário em um outro momento utilizando para tal um computador. Assim torna possível uma análise mais complexa e aprofundada do comportamento térmico e luminoso de uma determinada área.

2 DESENVOLVIMENTO

O projeto possui em sua arquitetura dois botões. O botão que está próximo da borda da *protoboard* é responsável por enfileirar os dados, ou seja, quando pressionado ele pegará os dados do sensor de temperatura e de luminosidade, criará um *noh* que receberá em seus atributos *mTemp* e *mLuz,* respectivamente, além do *mTime* e irá inserir em uma fila de *noh's*[1]. O botão que se encontra no centro da *protoboard* é o responsável por finalizar o programa. Ao ser pressionado o programa insere todos os *noh's* da fila numa *tabelaHash* e escreve esta tabela em um arquivo de texto no cartão de memória.

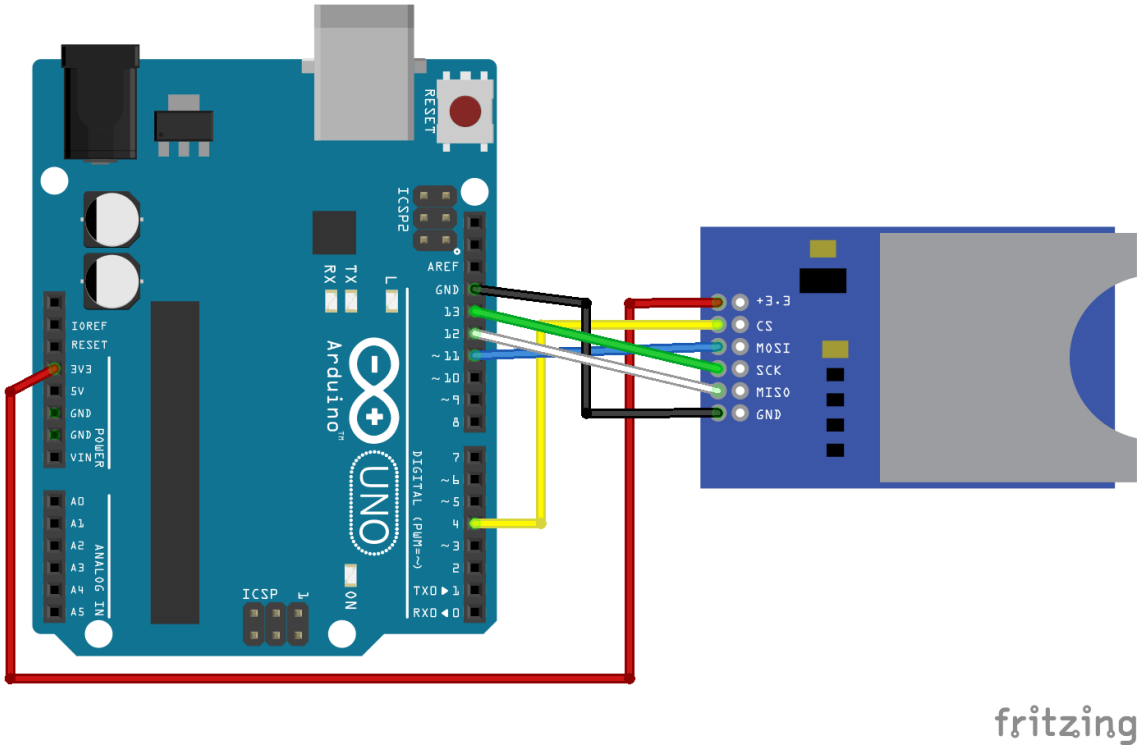

Figura 1: Esquema do Projeto/Créditos na Imagem

A fila terá uma quantidade de 20 nós. Ao chegar na décima quinta inserção será emitido um aviso no serial[2] avisando o usuário sobre o limite – caso o usuário esteja

[1] As classes serão explicadas e comentadas mais à frente na seção 2.1 Estrutura do Trabalho.
[2] Terminal onde mostra no computador algumas informações do código em execução no Arduino.

usando o Arduino ligado à um computador. Após a vigésima inserção o primeiro *noh* da fila (o mais antigo) será removido para liberar espaço para a nova inserção.

A fila possui esta limitação devido aos recursos escassos do microcontrolador. E a mesma será responsável por guardar os dados enquanto o programa está em execução.

O *noh* que será inserido na *tabelaHash* passa por uma função *hash*[3] que direciona o *noh* para o seu devido local na tabela. Esta tabela será dividida em 24 linhas. Cada linha poderá ter infinitas colunas. Uma vez que os dados estão posicionados na tabela esta é escrita no cartão e o programa é finalizado.

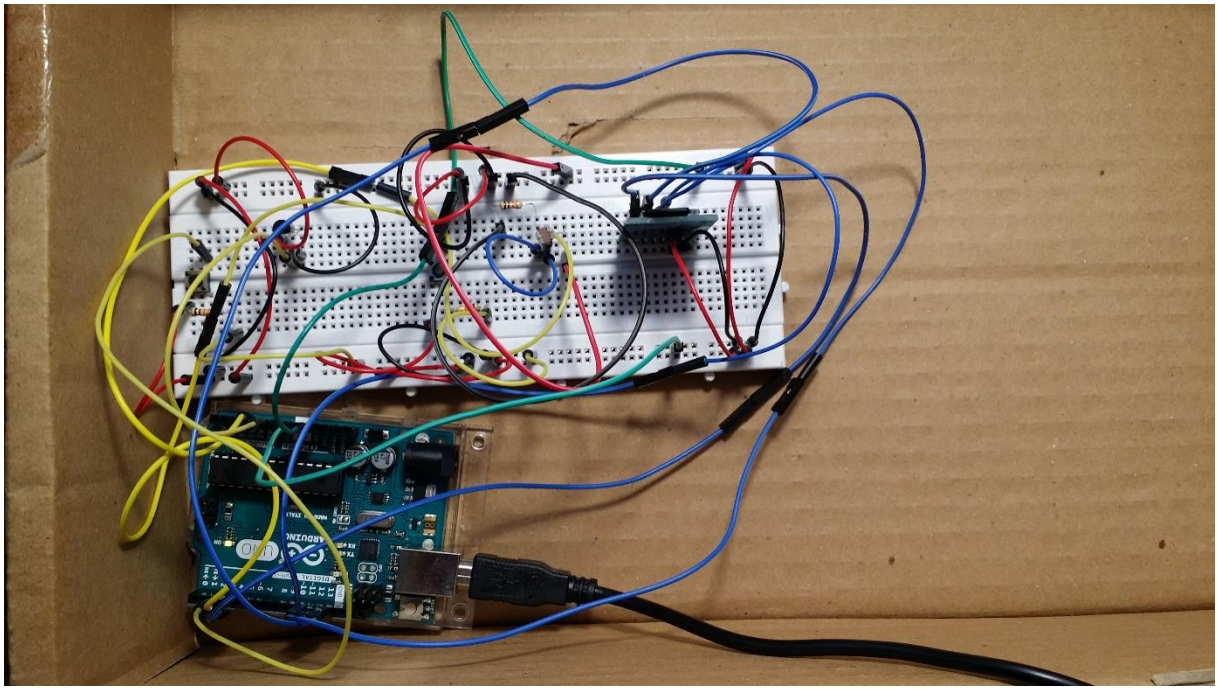

Figura 2: Projeto Final

[3] Os métodos também serão explicados mais à frente na seção 2.1 Estrutura do Trabalho.

2.1 ESTRUTURA DO TRABALHO

Foram criadas três classes: *Noh,* Fila e *TabelaHash.* As funções desenvolvidas para este projeto são: *setup* e *funçãoHash*. Não utilizaremos a função *loop* nativa do Arduino[4].

2.1.1 Classe *Noh*

Também chamada de nó, ou nós, esta classe será onde se armazenará a temperatura e a luminosidade, um ponteiro para o próximo *Noh*[5], além de um atributo *mTime*[6].

2.1.2 Classe Fila

Responsável por receber os nós até que o comando de enviar para a *TabelaHash* seja executado. Esta é uma fila de nós com limite para 20 nós.

2.1.3 Classe TabelaHash

Uma vez que os nós começam a sair da fila eles passarão por uma *funçãoHash* para serem devidamente alocados na tabela. A tabela informará quando no dia foi capturada as informações, por exemplo, se foi medido às 9 horas da manhã será salvo na tabela na posição relacionada às 9 da manhã. Agora caso seja medido às 22 este nó será armazenado na posição relacionada às 22 horas.

2.1.3.1 funçãoHash

Esta função redireciona o nó para um ponteiro dentre todos os ponteiros disponíveis no vetor de ponteiros da *tabelaHash*. A função *hash* pegará o milissegundo salvo no nó (atributo *mTime*) e dividirá por 3.600.000 convertendo, assim, para um valor em horas que será então aplicada uma divisão modular neste valor retornando assim o resultado desses milissegundos num período de 24 horas. Este resultado remete a um ponteiro.

---

[4] O motivo pelo qual a função não foi utilizada está explicado na seção 2.2 Dificuldades Encontradas.

[5] Este ponteiro é necessário para a organização interna da estrutura.

[6] Este atributo será utilizado mais pra frente na função *Hash*.

2.1.4 Função Setup

A função setup é onde o projeto é preparado e executado. Primeiro ela prepara os sensores e as variáveis. Depois começa um loop que irá rodar até que o botão de finalizar seja pressionado.

2.2 DIFICULDADES ENCONTRADAS

No decorrer do projeto encontramos dificuldades em adaptar nossa ideia inicial as mudanças propostas pelos professores – que alteravam a ideia principal do projeto. Entretanto este problema foi de fácil resolução pois bastava recomeçar o projeto que não estava em um estado tão avançado de desenvolvimento.

A utilização de um cartão de memória *micro SD* nos trouxe dificuldade pois o primeiro que nós obtemos estava corrompido. Isso causou um transtorno, mas foi resolvido com a substituição do cartão por outro.

Além de ser necessário trocar o cartão de memória também foi necessário rever o código no qual trabalhávamos pois o que foi usado inicialmente continha algum erro que o grupo não conseguiu encontrar e consertar. A solução foi implementar um *template* básico da biblioteca *Arduino SD Library ReadWrite example sketch*, modificando-o para atingir o objetivo do trabalho.

Com as modificações acima citadas foi possível abrir o arquivo, entretanto, uma nova complicação apareceu. Não estava sendo possível escrever no arquivo dentro do cartão de memória com o código implementado dentro da função padrão *loop* do Arduino. A solução para este problema foi implementar um *loop* a parte que não envolva o *loop* padrão que já é nativo ao Arduino.

Quando o microcontrolador percorria a *hash* ele entrava em um *loop* infinito. Este problema era causado pois a *funçãoHash* gerava um valor negativo – coisa que não pode ocorrer neste sistema. Uma vez corrigido isso o problema foi sanado.

No momento de se construir a *tabelaHash* a tabela era montada, mas, acontecia colisão. Esta colisão era na verdade valores que ainda estavam na fila pois os *noh's*

saiam da fila e, a partir da função *hash*, todos os valores da fila eram inseridos neste mesmo ponteiro. Após esta inserção a fila ficava com um nó a menos e o processo se repetia. A *funçãoHash* era chamada mais uma vez para o novo nó inicial da fila. Assim cada vez que um nó saía da fila todos os valores da fila eram inseridos na *tabelaHash* na mesma posição que este nó. Seguindo este processo a tabela era montada com todos os nós da fila sendo que cada andar[7] possuía todos os nós da fila. Para solucionar este erro bastou adicionar uma linha de código que ajustava o atributo *mPtProx* do nó a ser inserido na *hash* para que o mesmo aponte para NULL. Feito isso não era mais inserido todos os valores da fila, somente o que deveria ser inserido.

O método Remover da classe Fila quando precisava deletar o nó demorava bastante e fazia com que houvesse lixo e acontecia o mesmo que no problema citado no parágrafo anterior. A solução para isso foi adicionar *delays*.

[7] Um andar da *tabelaHash* corresponde ao ponteiro ao qual a função *hash* remete.

3 SUGESTÕES

Este é um projeto que pode ser generalizado para uma infinidade de aplicações bastando pouquíssima alteração. Utilizar outros sensores (com a devida modificação em código para manter a coesão e o sentido) permite que a área de atuação deste projeto mude.

Como exemplo, usando um sensor de umidade junto com um sensor de temperatura, é possível calcular uma estimativa de sensação térmica num quarto para saber se um bebê está em uma temperatura confortável ou não.

A utilização de uma placa de expansão para o microcontrolador possibilita o uso de mais sensores trazendo consigo mais informações que poderão enriquecer os resultados.

4 CONCLUSÃO

Após várias semanas de projeto finalmente foi obtido um programa que atendeu os requisitos planejados inicialmente. Foram muitos problemas encontrados no decorrer deste projeto. Não foi fácil como se foi pensado inicialmente. Entretanto muitas das dúvidas que surgiram no decorrer deste trabalho foram sanadas graças aos orientadores.

O experimento proposto aqui pode ser facilmente modificado para atender uma ampla gama de setores. Sendo possível usar não só como foi descrito aqui – numa estufa – mas para medições de temperatura/tempo, claridade/tempo/temperatura, etc.

No fim o projeto foi um sucesso. A equipe trabalhou muito bem e soube administrar bem o tempo e como proceder diante dos problemas enfrentados. Todos os integrantes foram assíduos nas tarefas as quais foram designados. Não houve qualquer conflito interno e o trabalho terminou com total concordância por parte dos participantes.

5 BIBLIOGRAFIA CITADA

Fórum Arduino. Disponível em: <https://forum.arduino.cc/>. Acesso em: 03 de dez. 2018.

Fórum Robocore. Disponível em: <https://www.robocore.net/modules.php?name=Forums>. Acesso em: 04 de dez. 2018.