https://eventos.utfpr.edu.br//sei/sei2020

Redução de custos na assistência a projetos no processo de coleta de dados

Cost reduction in assisting projects in the data collection process

João Paulo Bachega
joaopaulobachega@gmail.com
Universidade Tecnológica Federal do Paraná, Londrina, PR, Brasil

Roger Nabeyama Michels
rogernmichels@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Londrina, PR, Brasil.

Guilherme Quereza Vieira
guiquereza@hotmail.com
Universidade Tecnológica Federal do Paraná, Londrina, PR, Brasil

Leonardo Galice Chies
leonardo.g.chies@hotmail.com
Universidade Tecnológica Federal do Paraná, Londrina, PR, Brasil

Janksyn Bertozzi
janksynbertozzi@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Londrina, PR, Brasil

Recebido: 19 ago. 2020.
Aprovado: 01 out. 2020.

RESUMO

Este trabalho busca a adaptação de um sistema para assistência na coleta de dados de projetos inicialmente desenvolvido com base em Arduino Mega para seu funcionamento em Arduino Uno. O objetivo é reduzir custos e ampliar suas possibilidades de aplicação. Para tal, o código foi submetido a uma análise na qual os pontos de maior consumo de memória RAM foram identificados e alterados, resultando em uma versão capaz de ser executada pelo hardware com segurança. A possibilidade de uso do sistema com base em um Arduino Uno e a consequente redução de custos obtida permitem a ampliação na utilização deste, cumprindo com o objetivo maior de viabilizar um número maior de projetos.

PALAVRAS-CHAVE: Arduino, *Open Source,* gerenciamento de memória.

ABSTRACT

This work seeks to adapt a system to assist in the collection of data from projects designed based on Arduino Mega for its operation on Arduino Uno. The objective is to reduce costs and expand application possibilities. To this end, the code was subjected to an analysis in which the points of greatest consumption of RAM were identified and altered, detected in a version capable of being safely executed by the hardware. The possibility of using the system based on an Arduino Uno and the consequent reduction in costs obtained, obtained by expanding its use, fulfilling the greater objective of enabling a greater number of projects.

KEYWORDS: Arduino, *Open Source*, memory management.

INTRODUÇÃO

Segundo Azúa-Barrón,

> Com os imensos avanços da eletrônica, o mercado está saturado de instrumentos e sensores que permitem que os dados sejam coletados e armazenados, mas estes normalmente apresentam custos muito elevados, criando a necessidade de desenvolver dispositivos que coletem e armazenem informações a baixo custo que podem ser acessíveis a produtores, pesquisadores e estudantes que dispõem de baixo orçamento (Azúa-Barrón et al., 2017,p.1, tradução nossa).

Neste contexto, o Arduino se mostra como uma alternativa interessante.

"O Arduino é uma plataforma de prototipagem baseada em microcontrolador que consiste em uma placa com microcontrolador (MCU), estrutura de *software* para este MCU e ambiente de desenvolvimento integrado" (Dolinay et al., 2016, tradução nossa). Trata-se de uma plataforma de prototipagem e programação *open source*, permitindo a criação dos mais variados sistemas.

Além da possibilidade de uso de diversos componentes eletrônicos simples, como sensores e atuadores, existem diversos componentes com maior grau de complexidade chamados *Shields*. Os *Shields* permitem ao Arduino diversas funcionalidades, por exemplo, sua integração com *smatphones* via *bluetooth*, permitindo a visualização dos dados coletados em tempo real de forma simples (Ouariach et al., 2020).

Assim sendo, este trabalho trata da adaptação para Arduino Uno de um sistema de coleta de dados de temperatura com a funcionalidade bluetooth desenvolvido para Arduino Mega, sendo o objetivo desta alteração uma redução significativa no valor final do sistema e a ampliação de suas aplicações, além de seu uso no atendimento a um projeto de trabalho de conclusão de curso (TCC) existente no campus.

MATERIAIS E MÉTODOS

O *hardware* utilizado é basicamente composto pelo Arduino, um shield RTC (real time clock) com capacidade de registro de dados em cartão SD (Secure Digital), um *shield bluetooth* HC-06 e sensores de temperatura DS18B20.

Este sistema já foi utilizado para a coleta de dados em diversos projetos desenvolvidos no campus, comunidade e industrias da região, dentre eles a coleta de dados para um projeto de TCC que visa a redução de temperatura de ambientes internos a partir do uso de uma parede verde que consiste no cultivo de plantas anexas à parede.

Para comparar de forma consistente as temperaturas ambiente e do experimento é necessária a realização de diversas medições ao longo do dia em

horários pré-determinados, o que implicaria em um custo e uma margem de erro consideráveis no caso de se adotar o método de medição manual.

Para resolver este problema, o sistema automatizado foi instalado com um total de 10 sensores de temperatura, sendo 9 distribuídos no interior do ambiente de testes e um externo, para obtenção da temperatura ambiente. A temperatura aferida pelos sensores é captada e registrada no cartão SD juntamente ao horário da aferição fornecido pelo módulo RTC.

Após este experimento, mesmo com o perfeito funcionamento do sistema, foram constatados alguns pontos que poderiam ser trabalhados, sendo o principal deles relacionado ao próprio Arduino, sendo o sistema baseado em microcontrolador do modelo Mega, o de maior capacidade disponível no mercado e, consequentemente, o mais custoso.

Apesar do sistema apresentar um valor acessível em relação aos equipamentos comerciais que apresentam a mesma função, ainda é possível reduzir de forma significativa os custos substituindo o Arduino por uma versão menos onerosa, viabilizando o atendimento a um número maior de projetos.

Para isso, foi proposta uma adaptação do programa original desenvolvido no Arduino Mega para o Arduino Uno. Para justificar a necessidade de adaptação é necessário esclarecer a existência de algumas diferenças entre eles. Cada modelo de Arduino apresenta diferentes capacidades de processamento, quantidades de memória *RAM* (*random access memory*) e quantidades de memória *flash,* o que influencia no tamanho total do código e no número de variáveis que podem ser utilizadas

O compilador do Arduino é capaz de apresentar a quantidade de memória *RAM* e *flash* consumidas pelo código quando instalado em um modelo de hardware, permitindo assim uma previsão de seu comportamento.

A tentativa de uso do programa original diretamente no *hardware* Uno resulta em uma ocupação de 96% do total disponível de *RAM* e 71% do total de memória *flash*. O valor elevado de *RAM* pode levar a falhas ou bugs do sistema, pois parte da *RAM* pode não ser reconhecida ou apresentar um maior uso devido a condições não previstas. Desta forma, se fez necessária uma adaptação no código para permitir seu uso de forma segura.

Figura 1 – Quantidade de memórias RAM e flash consumidas pelo programa original

```
#include <SPI.h>      // SD Card
#include <DallasTemperature.h>
#include <Wire.h>     // RTC
#include <RTClib.h>   // RTC
#include "RTClib.h"   // RTC
RTC_Millis rtc;
RTC_DS1307 RTC;
byte Rele1   = 2;
//byte Rele2 = 3;

// the logging file
File logfile;
const int ONE_WIRE_BUS = 10;               // signal pin of DS18B20 connected to pin 3
const int rxpin = 19;
const int txpin = 18;
SoftwareSerial bluetooth (rxpin, txpin);
OneWire ourWire(ONE_WIRE_BUS);
DallasTemperature sensors(&ourWire);
//#define ONE_WIRE_BUS 10
#define LOG_INTERVAL 2000 //define o intervalo entre o registro
 //de duas medidas, neste caso 20000 ms, ou 20s.
#define SYNC_INTERVAL 10000 // milisegundos entre a chamada para
 //limpar e para gravar dados no cartão. 2
 uint32_t syncTime = 0; // time of last sync()
 #define ECHO_TO_SERIAL 1 // echo data to serial port
#define WAIT TO START 0 // Wait for serial input in s
```

```
Compilação terminada.
O sketch usa 23092 bytes (71%) de espaço de armazenamento para programas. O máximo são 32256 bytes.
Variáveis globais usam 1985 bytes (96%) de memória dinâmica, deixando 63 bytes para variáveis locais. O máximo são 2048 bytes.
Pouca memória disponível, problemas de estabilidade podem ocorrer.
```

Fonte: Autoria própria (2020).

Para isso, o primeiro passo foi a análise do código em busca dos pontos de maior consumo de *RAM* a fim de definir como alterar o programa para enquadrá-lo ao *hardware*. Neste sentido, os pontos que chamaram mais atenção foram a criação de um vetor com tamanho fixo de 50 posições para armazenamento de valores de temperatura e o grande número de variáveis de texto (*strings*) existentes.

A primeira alteração foi a redução do vetor para apenas 10 posições (o número de sensores pretendido), pois, apesar de não utilizadas, as outras 40 posições do vetor continuavam a utilizar espaço na *RAM*.

No que diz respeito à quantidade de *strings* apresentadas durante a execução do programa, a solução encontrada foi a transferência destas para a memória *flash* através da macro F().

De acordo com a documentação oficial do Arduino, a macro F() é uma forma de utilizar o modificador de variáveis PROGMEM, pertencente à biblioteca pgmspace.h que atualmente é automaticamente adicionada à IDE (*Integrated Development Environment*) do Arduino. Este modificador faz com que os dados da variável sejam armazenados em memória *flash* ao invés de memória RAM. Apesar de limitada, por reduzir o acesso do programa a essas variáveis e impedir sua alteração durante a execução do código, esta macro pode seguramente ser utilizada neste tipo de situação por se tratar de frases fixas.

Figura 2 – Uso de memória após as alterações citadas

```
#include <DallasTemperature.h>
#include <Wire.h>     // RTC
#include <RTClib.h>   // RTC
#include "RTClib.h"   // RTC
RTC_Millis rtc;
RTC_DS1307 RTC;
byte Rele1   = 7;
//byte Rele2 = 3;

// the logging file
File logfile;
const int ONE_WIRE_BUS = 9;               // signal pin of DS18B20 connected to pin 3
const int rxpin = 2;
const int txpin = 3;
SoftwareSerial bluetooth (rxpin, txpin);
OneWire ourWire(ONE_WIRE_BUS);
DallasTemperature sensors(&ourWire);
//#define ONE_WIRE_BUS 10
#define LOG_INTERVAL 2000 //define o intervalo entre o registro
 //de duas medidas, neste caso 20000 ms, ou 20s.
#define SYNC_INTERVAL 10000 // milisegundos entre a chamada para
 //limpar e para gravar dados no cartÃ£o. 2
 uint32_t syncTime = 0; // time of last sync()
 #define ECHO_TO_SERIAL 1 // echo data to serial port
#define WAIT_TO_START 0 // Wait for serial input in s
OneWire oneWire(ONE_WIRE_BUS);

Compilação terminada.

O sketch usa 23332 bytes (72%) de espaço de armazenamento para programas. O máximo são 32256 bytes.
Variáveis globais usam 1423 bytes (69%) de memória dinâmica, deixando 625 bytes para variáveis locais. O máximo são 2048 bytes.

12
```

Fonte: Autoria própria (2020).

RESULTADOS

O sistema original baseado no Arduino Mega apresentou um bom funcionamento, atendendo ao objetivo de ler e armazenar os dados de temperatura dos diversos projetos em que foi aplicado.

O uso da macro f() e a redução de tamanho do vetor para armazenamento de dados apresentou um resultado inicial extremamente satisfatório, permitindo uma redução significativa no custo do sistema sem nenhuma modificação em seu funcionamento, pois as alterações realizadas limitam-se a realocação de conteúdos e gerenciamento da memória disponível.

Dessa forma, será possível aplicar o mesmo sistema com o funcionamento já conhecido e com alto grau de confiabilidade a um número maior de projetos.

CONCLUSÕES

Tendo em visto o exposto, o Arduino é uma solução viável para a criação de sistemas de coleta de dados de baixo custo, apresentando a possibilidade de aplicações diversas com alta confiabilidade.

É possível observar através do que anteriormente apresentado que, no que diz respeito ao objetivo principal do trabalho, a redução de custos do sistema existente para permitir o atendimento a um número maior de projetos através de

alterações no *software* e, consequentemente, no *hardware* é uma possibilidade real, cujo funcionamento de forma prática será testado assim que possível.

AGRADECIMENTOS

Os autores agradecem o apoio da Fundação Araucária no desenvolvimento deste trabalho, através da concessão de bolsa de extensão.

REFERÊNCIAS

Azúa-Barrón, M.; Vázquez-Peña, M. A.; Arteaga-Ramírez, R.; Hernández-Saucedo, R. **Sistema de adquisición de datos de bajo costo con la plataforma arduino**. Revista mexicana de ciencias agrícolas, v.8, n.1, p.1-12. 2017.

Dolinay, Jan; Dostálek, Petr; Vašek, Vladimír. **Arduino Debugger**, IEEE EMBEDDED SYSTEMS LETTERS, Vol.8(4), pp.85-88, DECEMBER 2016.

Ouariach, A.; El Hadi, M.; El Moussaouy, A.; Hachmi, A.; Magrez, H.; Bria, D. **Development of an education low-cost teslameter by using Arduino and Smartphone application**. Physics Education, v.55, n.3, 033008. 2020.