Persistência de Dados no MySQL com Arduino: Uma Proposta Utilizando MySQL Connector / Arduino

Data persistence in MySQL with Arduino: a Proposal Using MySQL Connector / Arduino

Alexandre Aprato Ferreira da Costa[a]*; Lucas Dornelles Nunes[b]

[a]Universidade Norte do Paraná, Pós-Graduação Lato Sensu em Tecnologias para Aplicações Web.
[b]Instituto Federal Farroupilha, Curso de Análise e Desenvolvimento de Sistemas.
*E-mail: alexandreiffca@gmail.com

Resumo

Frequentemente sistemas de automação desenvolvidos utilizando a plataforma Arduino fazem o armazenamento de dados em bases MySQL através da utilização de um middleware. Tal metodologia, embora eficaz, necessita de aquisição e desenvolvimento tornando-se custosa e onerosa. Este artigo apresenta uma proposta para a persistência de dados dispensando a camada intermediária, com objetivo de tonar essa tarefa mais fácil e eficiente. Para o desenvolvimento do trabalho foram utilizadas tecnologias como Arduino Uno, Ethernet Shield e MySQL. Concluiu-se que a utilização do sistema proposto é vantajosa ao tornar a tarefa de persistência de dados com Arduino em uma base MySQL fácil e eficiente, poupando tempo e recursos.

Palavras-chave: Arduino. MySQL. Persistência de Dados.

Abstract

Frequently automation systems developed using the Arduino platform make data storage MySQL databases by using a middleware. Such method, though effective, requires the acquisition and development making it costly and burdensome. This paper presents a proposal for data persistence eliminating the intermediate layer, in order to behold this easy and efficient task. For the development of this work were used technologies like Arduino Uno, Shield Ethernet and MySQL. It was concluded that the use of the proposed system is advantageous by making the task of data persistence with Arduino on an easy and efficient MySQL database, saving time and resources.

Keywords: Arduino. MySQL. Data Persistence.

1 Introdução

Os estudos desenvolvidos em pesquisas que utilizam a plataforma Arduino (ARDUINO, 2016) são divididos em duas principais vertentes: Automação, para o controle automático onde os mecanismos verificam seu próprio funcionamento efetuando medições e introduzindo correções sem a interferência humana (LACOMBE, 2004); e robótica, para realização de tarefas com eficiência e precisão, até mesmo onde a presença do homem se torna difícil (ROSÁRIO, 2010). Quando o assunto é automação, frequentemente estas pesquisas fazem a utilização de banco de dados.

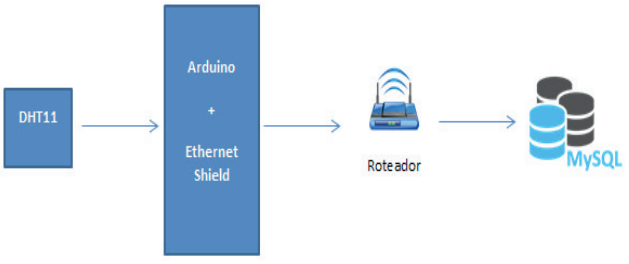
O foco desta pesquisa foi estudo dos recursos e tecnologias disponíveis, e a identificação de qual é adequado para persistir os dados com Arduino e MySQL, pois salvar os dados em um banco de dados não só os preserva para posterior análise, mas também significa que seu projeto pode alimentar dados para aplicações mais complexas (BELL, 2015).

Assim a presente pesquisa teve como objetivo demonstrar as metodologias amplamente empregadas no mercado e no meio acadêmico, ou seja, por empresas e pesquisadores, de forma a verificar que possibilidade ainda pode ser explorada no sentido de desenvolver sistemas com maior facilidade. Além de propor a comunicação entre uma aplicação utilizando a plataforma Arduino e um banco de dados MySQL, estando disponíveis para a implementação as ferramentas: Arduino Uno R3, Ethernet Shield e o sistema gerenciador de banco de dados MySQL.

2 Material e Métodos

Para o desenvolvimento da proposta foi utilizado computador com o sistema operacional Linux Ubuntu versão 16.04.1 LTS, Arduino IDE versão 1.6.10, placa Arduino UNO Rev3, Ethernet Shield W5100, sensor de umidade e temperatura DHT11, MySQL versão 5.7.6 e a biblioteca MySQL Connector/Arduino 1.1. A Figura 1 representa a arquitetura desta proposta.

Figura 1: Arquitetura da Proposta



Fonte: O autor

Como é possível visualizar, foi dispensado o uso de uma camada intermediária, *middleware*, para realizar a persistência dos valores adquiridos através do sensor DHT11 na base de dados MySQL.

2.1 Base de dados

Primeiramente foi criada a base de dados e a tabela que armazena a temperatura e umidade. O *script* para a criação do banco de dados e da tabela pode ser visualizado na Figura 2.

Figura 2: Script para criação do banco e tabela

```
CREATE DATABASE Arduino;
CREATE TABLE Arduino.Valores (
id integer primary key auto_increment,
humidade numeric(9,2),
temperatura numeric(9,2),
PRIMARY KEY (id)
);
```

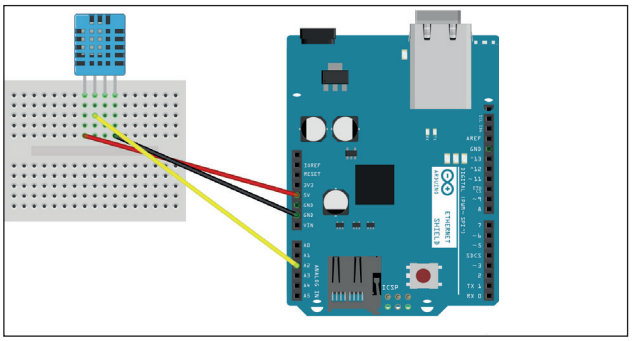
Fonte: O autor

Através do *script* pode-se verificar que primeiramente é criada a base de dados denominada Arduino. Na sequência é criada a tabela com o nome Valores, que possui os campos: id, humidade, temperatura. Respectivamente eles possuem a função de armazenar: o identificador do registro, o valor de umidade do ambiente e o valor de temperatura do ambiente.

2.2 Aquisição de dados

A plataforma de aquisição de dados é composta basicamente do Arduino Uno, Ethernet Shield e do sensor DHT11. Abaixo, na Figura 3, é possível visualizar a ligação elétrica dos componentes da referida plataforma.

Figura 3: Ligação Elétrica



Fonte: O autor

Foi utilizada uma placa de ensaio, *protoboard*, para facilitar as conexões, que por sua vez, foram feitas através de alguns fios de ligação. Esta ligação foi feita da seguinte forma: a linha vermelha representa a ligação do sensor na alimentação de 5V, a linha preta representa o GND (neutro), enquanto o fio amarelo contém o sinal do sensor enviado para o Arduino através do pino analógico dois.

3.3 Firmware

O firmware é o conjunto de instruções que fazem com que sejam realizadas as leituras dos valores de temperatura e umidade e, após, que estes sejam enviados a base de dados. A Figura 4 exibe a primeira parte deste conjunto de instruções.

Figura 4: Bibliotecas e Declarações

```
#include <Ethernet.h>
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>
#include "DHT.h"
#define DHTPIN A2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
byte mac_addr[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress server_addr(192,168,1,6);
char user[] = "root";
char password[] = "";

char INSERT_DATA[] = "INSERT INTO arduino.valores (temperatura, humidade)
                                                VALUES ('%s','%s')";
char query[128];
char temperatura[10];
char humidade[10];
EthernetClient client;
MySQL_Connection conn((Client *)&client);
```

Fonte: O autor.

O trecho acima corresponde à inclusão das bibliotecas, declarações e inicialização de algumas variáveis. O destaque é a inclusão das bibliotecas MySQL_Connection.h e MySQL_Cursor.h. Também destaca-se as variáveis que contém o usuário do banco de dados, a senha do usuário no banco de dados e a instrução para inserções dos dados. Abaixo, na Figura 5, está a codificação da função *setup( )*.

Figura 5: Código do *Setup ( )*

```
void setup() {
  Serial.begin(115200);
  while (!Serial);
  Ethernet.begin(mac_addr);
  dht.begin();
  Serial.println("Connecting...");
  if (conn.connect(server_addr, 3306, user, password)) {
    delay(1000);
  }
  else
    Serial.println("Connection failed.");
}
```

Fonte: O autor.

Este segundo trecho inclui a inicialização da porta serial, da rede, do sensor de temperatura e é feita a conexão com o banco de dados. O próximo trecho, Figura 6, corresponde a codificação da função *loop( )*.

Figura 6: Código do Loop ( )

```
void loop() {

  float h = dht.readHumidity();
  float t = dht.readTemperature();

  MySQL_Cursor *cur_mem = new MySQL_Cursor(&conn);

  dtostrf(t, 1, 1, temperatura);
  dtostrf(h, 1, 1, humidade);
  sprintf(query, INSERT_DATA, temperatura, humidade);

  cur_mem->execute(query);

  delete cur_mem;

  delay(5000);
}
```

Fonte: O autor.

No *loop( ),* primeiramente vai ser feita a leitura dos valores de temperatura e umidade, e criada uma instância da classe MySQL_Cursor. Após, os valores de temperatura e humidade são convertidos em um vetor de caracteres e é retornado o ponteiro desta *string*. Na sequência, todos estes valores são montados em uma *string* única e a instrução resultante é executada. Por fim, é realizada a exclusão do cursor e feita e liberação da memória utilizada.

3 Resultados e Discussão

3.1 Plataforma Arduino

Arduino é o nome dado ao projeto que teve início na Itália no ano de 2005 e consiste basicamente em placas de controle de entrada e saída baseadas no microcontrolador Atmel AVR. É uma plataforma de prototipação eletrônica de código-aberto, flexível, e de fácil utilização tanto em nível de hardware quanto em software (ARDUINO, 2016). Esta plataforma possui vários modelos de placas, e o que diferencia esses modelos, basicamente, são: o número de pinos, a quantidade de memória e, por ser de código aberto, algumas funcionalidades adicionadas pelo seu construtor. Outro fator importante, é que a placa Arduino suporta a utilização de Shields, que são placas que podem ser conectadas ao Arduino, a fim de estender as suas capacidades, tornando-o capaz de executar outras tarefas como, por exemplo: acessar a Internet, controlar motores e comunicação Wi-Fi. A linguagem de programação utilizada é semelhante ao C++ e um programa básico para rodar na plataforma necessita de duas funções básicas que são o "void setup()" que é executado uma única vez no início do programa com informações gerais sobre este, e a função "void loop()" que é a função executada repetidamente pelo microcontrolador.

3.2 Linguagens de Programação

Uma linguagem de programação é um método padronizado para comunicar instruções para um computador. É um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador. C++ é uma linguagem que deriva do C retendo a maioria das características desta, porém oferecendo maior suporte à maior quantidade de estilos de programação, tornando-a mais versátil e flexível (SILVA FILHO, 2010). O PHP é uma das linguagens orientada objetos que mais cresce no mundo, e disparadamente a linguagem mais utilizada para o desenvolvimento Web (DALL'OGLIO, 2015). Java é uma linguagem de programação de uso geral, extremamente rica e poderosa sendo uma das principais soluções se tratando de desenvolvimento (COSTA, 2008). Por ser completa, pode ser utilizada tanto para produção de jogos, programas corporativos, processamento científico, programas em rede, entre outros.

3.3 Sistema de Gerenciamento de Banco de Dados

Um sistema de gerenciamento de banco de dados, ou SGBD, consiste de um software que foi projetado para auxiliar na manutenção e utilização de vastos conjuntos de dados. Um SGBD é um "software que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados" (HEUSER, 2009, p.15). Seu principal objetivo é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a manipulação e a organização dos dados. O SGBD disponibiliza uma interface para que seus clientes possam incluir, alterar ou consultar dados previamente armazenados. Em bancos de dados relacionais a interface é constituída pelas APIs (Application Programming Interface) ou drivers do SGBD, que executam comandos na linguagem SQL (Structured Query Language).

O MySQL é o banco de dados de código aberto mais popular do mundo, que possibilita a entrega econômica de aplicativos de banco de dados confiáveis, de alto desempenho e escaláveis, com base na Web e incorporados (ORACLE, 2016). É um SGBD que utiliza a linguagem SQL como interface. É atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações pelo mundo.

A biblioteca MySQL Connector/Arduino, é a tecnologia criada para Arduino, para que este possa se conectar a um

banco de dados MySQL. Através desta tecnologia, é possível conectar o Arduino a um banco de dados MySQL dispensando o uso de computador intermediário ou Web-Service, e a comunicação pode ser tanto através da Ethernet Shield quanto Wi-fi Shield (BELL, 2015).

Comumente, sistemas de automação que necessitam de persistência de dados utilizam estrutura de três camadas, onde: existe uma camada de aplicação, uma camada de banco de dados e uma camada de troca de dados (SILVA, 2015). A camada de troca de dados é o *middleware* que intermedia a comunicação entre as camadas de aplicação e de banco de dados. Loureiro *et al.* (2003), ressalta que o *middleware* permite que informações adquiridas no canal de sensoriamento sejam repassadas para a pilha de protocolos de rede, a fim de serem transmitidos a outro nó. O nó neste caso é o software que depende da coleta e tratamento de dados através destes dispositivos externos, e que necessita de uma linguagem diferente a qual o software utiliza usualmente, sendo assim é feita a integração das linguagens.

Conforme IBM (2016), o Registro de dados com hardware e software livre no setor de energia, é realizado com Arduino, PHP e MySQL. Da plataforma Arduino utiliza-se o Uno e o Ethernet Shield, o PHP é utilizado como linguagem de programação do *backend* para consultar o Arduino e realizar o armazenamento, e o MySQL foi o SGBD utilizado para armazenar os dados.

Já no "Desenvolvimento de um Sistema para Monitoramento Remoto em Centrais de Microgeração Fotovoltaica" proposto por Halmeman (2014), foram utilizados o Arduino modelo Duemilanove, comunicação sem fio e MySQL. A interface de usuário foi desenvolvida em PHP, enquanto o *middleware* para coleta e armazenamento de dados foi escrito em Java.

Para Lu (2014) que trata de do desenvolvimento de um sistema de aquisição de dados e controle supervisório de níveis de CO_2 em recuperação aprimorada de óleo, os scripts em PHP possuem duas finalidades que são: prover a interface para o usuário e persistir na base de dados. Neste projeto foi utilizado Arduino Yun, WiFi Shield e MySQL.

Observa-se uma singularidade no que diz respeito à estrutura destes sistemas que utilizam a plataforma Arduino e armazenamento em base de dados, e, embora amplamente utilizada, essa arquitetura que utiliza *middleware*, pode ser facilmente substituída pela utilização da biblioteca MySQL Connector/Arduino a fim de facilitar o desenvolvimento dos sistemas, promovendo redução de tempo e custo. Desta forma, devolve-se a plataforma sua principal vantagem que é a facilidade com que pessoas "não técnicas" tem de criar os seus projetos a partir do básico e em um período relativamente curto de tempo (MCROBERTS, 2011).

3.4 Discussão dos dados

Como resultado tem-se uma aplicação totalmente funcional, onde os valores coletados pelo sensor são armazenados diretamente na base de dados sem necessidade de um intermediário. Na Figura 7 é possível visualizar parte destes registros.

Figura 7: Registros na Base de Dados

	id	humidade	temperatura
Editar Copiar Remover	52	56.00	15.00
Editar Copiar Remover	53	55.00	15.00
Editar Copiar Remover	54	55.00	15.00
Editar Copiar Remover	55	55.00	15.00
Editar Copiar Remover	56	55.00	15.00
Editar Copiar Remover	57	55.00	15.00
Editar Copiar Remover	58	55.00	15.00
Editar Copiar Remover	59	55.00	15.00
Editar Copiar Remover	60	57.00	16.00

Fonte: O autor.

É importante ressaltar que não foram discutidas soluções semelhantes para outros gerenciadores do mercado como: Oracle (www.oracle.com), PostgreSQL (www.postgresql.org) ou SQL Server (www.microsoft.com/SQL) pela inexistência de bibliotecas que realizassem a comunicação direta com esses SGBD's.

4 Conclusão

A utilização da biblioteca, instrumento desta proposta, é vantajosa ao tornar a tarefa de persistência de dados com Arduino em uma base MySQL fácil e eficiente, poupando tempo e recursos, uma vez que não se faz necessário codificar ou adquirir uma camada intermediária.

Outro fator importante é que as tecnologias utilizadas são livres e adequadas tanto para o estudo acadêmico como para projetos comerciais, fazendo com que interessados possam testar, reprogramar e modelar diversas situações. A documentação referente à biblioteca é completa e objetiva, e, embora ainda existam poucos exemplos práticos de sua utilização baseia-se na característica da plataforma Arduino na qual usuários com pouco domínio de programações conseguem realizar tarefas de níveis mais elevados de complexidade.

Para pesquisas futuras sugere-se o desenvolvimento e aplicação que realize as quatro operações utilizadas em bases de dados relacionais – inserção, recuperação, atualização e remoção, além da utilização de comunicação sem fio em substituição a cabeada. Além da realização de medidas de desempenho buscando comprar a utilização da biblioteca com outras formas de conexão entre o Arduino e o MySQL.

Referências

ARDUINO. Disponível em: < https://www.arduino.cc/>. Acesso em: ago. 2016.

BELL, C. *Beginning Sensor Networks with Arduino and Raspberry* Pi. Nova Iorque: Springer, 2015.

COSTA, D.G. *Java em Rede: programação distribuída na internet*. Rio de Janeiro: Brasport, 2008.

DALL'OGLIO, P. *PHP*:Programando com orientação a objetos. São Paulo: Novatec, 2015.

HALMEMAN, R.J. *Desenvolvimento de um sistema para o monitoramento remoto de centrais de microgeração fotovoltaica*. 2014. 202 f. Tese (Doutorado) – Universidade Federal Paulista Júlio Mesquita Filho, Botucatu, 2014.

HEUSER, C. A. *Projeto de banco de dados*. Rio de Janeiro: Elsevier, 2010.

IBM. Disponível em: <http://www.ibm.com/developerworks/br/library/os-arduinophp/>. Acesso em: ago. 2016.

LOUREIRO, A.A.F. *et al*. *Rede de sensores sem fio*. 2003. Disponível em: <http://homepages.dcc.ufmg.br/~loureiro/cm/docs/sbrc03.pdf>. Acesso em: ago. 2016.

LACOMBE, F.J.M. *Dicionário de administração*. São Paulo: Saraiva, 2004.

LU, X. *Supervisory Control and Data Acquisition System - Design for CO2 Enhanced Oil Recovery*. 2014. 26 f. Dissertação (Mestrado em: Engenharia Elétrica e Ciência da Computação) – University of California, Berkley, 2014.

MCROBERTS, M. *Arduino básico*. São Paulo: Novatec, 2011.

ROSÁRIO, J.M. *Robótica industrial I*: modelagem, utilização e programação. São Paulo: Baraúna, 2010.

ORACLE. Disponível em: <https://www.oracle.com/br/products/mysql/overview/index.html>. Acesso em: ago. 2016.

SILVA, L.G. *Park My Ride*: um sistema inteligente para monitoramento e gerenciamento de vagas em estacionamentos públicos e privados. 2015. 82 f. Monografia (Graduação em Engenharia de Computação) - Universidade Federal de Ouro Preto, João Monlevade, 2015.

SILVA FILHO, A.M.S. *Introdução à programação orientada a objetos com C++*. São Paulo: Elsevier, 2010