Universidade Federal Fluminense - UFF
Escola de Engenharia - TCE
Curso de Engenharia de Telecomunicações - TGT

Programa de Educação Tutorial – PET
Grupo PET-Tele

Tutoriais PET-Tele

Sistema de aquisição de dados utilizando Arduino e RFID para sistemas operacionais Windows e Linux

(Versão: A2017M08D13)

Autores: Thiago Chequer Coelho
Cássio Francklin Matos Carvalho Soares Xavier

Tutor: Alexandre Santos de la Vega

Niterói – RJ
Agosto / 2017

Sumário

Lista de Tabelas

Lista de Figuras

1 Introdução

O grupo PET-Tele trabalha há algum tempo desenvolvendo com a plataforma de prototipagem Arduino, e assim vem resultando na criação de materiais didáticos, cursos práticos, projetos e até uma optativa ministrada na Universidade Federal Fluminense. Isso tudo para, de alguma forma, repassar conhecimento e auxiliar primariamente os alunos de Engenharia de Telecomunicações e outros cursos.

Mais recentemente, o grupo iniciou pesquisas e projetos utilizando uma outra tecnologia, que nessa oportunidade é a *Radio Frequency Identification* (*RFID*, e em português: Identificação por Rádio Frequência).

O Arduino em sua arquitetura permite a associação com diversos outros periféricos, deixando assim mais ampla a gama de possibilidades a serem construídas. No grupo, o projeto pioneiro já consolidado foi uma sistema de acesso à sala do PET utilizando a carteirinha da Universidade como chave e um leitor *RFID* para habilitar a abertura da porta, isso tudo controlador pelo Arduino.

O tutorial desse projeto mencionado, bem como diversos outros materiais didáticos, podem ser encontrados pela página online do grupo: *http://www.telecom.uff.br/pet*

Neste presente documento, foi registrada a construção passo-a-passo de um projeto que adveio do sucesso do sistema de identificação. Após pesquisarmos e aprendermos sobre as tecnologias em questão, iniciamos um projeto de um sistema de aquisição de dados por rádio frequência. E parte dos mesmo princípios do sistema de acesso à sala do grupo, mas busca outros fins.

Resumidamente, esse sistema foi pensado à fim de adquirir uma informação presente em um identificador *RFID* (novamente a carteirinha da UFF), registrá-la e administrá-la de uma determinada forma em um banco de dados. Sempre com o suporte da plataforma Arduino realizando o intermédio.

Nas próximas páginas estarão os motivos que nos levaram a desenvolver esse sistema, uma explicação em si do que são as tecnologias utilizadas, o hardware necessário e como ele foi montado, e por fim os códigos de programação comentados que foram utilizados para construção do software.

2 Motivação

Como todo avanço tecnológico, partimos de uma necessidade ou então da ausência de algo de nosso dia-a-dia. Em outras palavras, encontramos algo que, por exemplo, dificulta ou atrasa a realização de alguma tarefa diária e à partir disso nós estudamos, pesquisamos e por fim construímos o que for necessário para enfim solucionar esses detalhes. De certa forma, é a realização básica de engenharia.

Seguindo essa ideia, no projeto piloto que envolve *RFID* e Arduino realizado pelo Grupo PET-Tele, uma tarefa diária comum à todas as pessoas foi automatizada. A abertura da porta, nesse caso, a de acesso à sala do grupo, bem como a chave utilizada para tal, passaram por um processo de "avanço tecnológico".

No lugar de abrir convencionalmente utilizando uma chave mecânica, a utilização do Arduino em conjunto com *RFID*, permitiu a utilização da carteirinha da Universidade Federal Fluminense (UFF) como chave.

No projeto tratado neste documento, temos um princípio semelhante. Almejamos construir um sistema que permita, por exemplo, automatizar todo processo de cadastro de um grupo de alunos. Utilizamos o mesmo princípio do acesso à sala. Todos os alunos da UFF tem uma

carteirinha, ou seja, uma forma de identificação única, muito semelhante ao Registro Geral (RG).

O funcionamento resumido do projeto é o seguinte: o aluno aproxima a carteirinha (que é uma *tag RFID*) do leitor, assim as informações serão extraídas e armazenadas num banco de dados anexo junto às informações básicas do aluno.

Com as informações armazenadas no computador, pode-se administrá-las de uma forma mais segura e eficaz utilizando alguma interface programada. Além disso, fica mais simples a formatação dos dados para criação de certificados.

Outra situação que o sistema pode ser empregado, é como "lista de presença". No momento de uma aula, palestra ou curso, o aluno passaria a carteirinha num leitor fixado próximo à porta e possibilitaria o sistema fazer o registro de presença.

Nas páginas seguintes, estarão algumas informações que buscam familiarizar o leitor com as tecnologias utilizadas para confecção do sistema.

3 Um pouco sobre as ferramentas utilizadas

3.1 *RFID* - Identificação por rádio frequência

Utilizando radiofrequência concomitantemente com variações de campo eletromagnético busca-se um simples e único objetivo: A comunicação entre componentes distintos. Se esse foi atingido, o *RFID* permite uma maior eficiência no rastreamento, localização e registro de objetos (sejam esses de várias naturezas), por exemplo.

Enraizada na Segunda Guerra Mundial, essa tecnologia advém dos radares que eram usados para verificar a aproximação de aviões com uma certa antecedência.

Porém, a história do *RFID* começa veementemente em 1973 quando foi patenteado um sistema de memória regravável e outro sistema para destravar uma porta utilizando uma etiqueta. Esse era o momento em que a solução para muitos problemas nascia.

Radio Frequency Identification é uma tecnologia focada em rastreamento, identificação e gerência de produtos das mais variadas naturezas. O diferencial dela está exatamente de não ser necessário campo visual direto com o que está sendo monitorado. Isso possibilita essa tecnologia ser adotada como solução para diversos problemas de logística. A estratégia é simples.

Usar RF em uma frequência específica, perturbação de campo eletromagnético, além de um conjunto de antenas e transmissores corretamente estruturados para capturar dados. Aparentemente é algo não muito complexo, mas quando bem trabalhado possibilita soluções inovadoras.

Essa tecnologia baseia-se na transmissão de dados via onda de rádio. Entretanto, ela não é a única que tecnologia que utiliza desse tipo de onda eletromagnética. Por isso, é importante garantir que a frequência utilizada não interfera em outros tipos de serviços que demandam da radiofrequência. Alguns exemplos desses outros serviços são: telefones móveis, rádios de comunicação de natureza militar ou aéreas e inúmeras outras.

Como todo sistema eletrônico, o RFID também é passível de erros. Esses que muitas vezes podem comprometer o funcionamento da tecnologia.

Afinal operar utilizando eletromagnetismo infere na possibilidade de interferência. Elas podem ser causadas por diversos motivos, inclusive ataques causados por ações de má fé. Até porque, estamos falando em transmissão de dados. Esses que podem conter informações privilegiadas sobre um determinado objeto e por isso necessitam de uma proteção específica para garantir o sigilo.

O *RFID* é composto por três vertentes principais: um identificador, um leitor e um banco de dados anexo.

Cada componente tem suas próprias características e subdivisões que quando organizadas se uma certa maneira gera um método de utilização distinto do *RFID*. E esse visa atuar como solução para um determinado tipo de problema.

3.2 Um pouco sobre Arduino

O Arduino faz parte do conceito de hardware e software livre e está aberto para uso e contribuição de toda sociedade. O conceito Arduino surgiu na Itália, em 2005, com o objetivo de criar um dispositivo que fosse utilizado em projetos/protótipos construídos de uma forma menos dispendiosa do que outros sistemas disponíveis no mercado.

Ele pode ser usado para desenvolver artefatos interativos stand-alone ou conectados ao computador, utilizando diversos aplicativos, tais como: Adobe Flash, Processing, Max/MSP, Pure Data ou SuperCollider.

O Arduino foi projetado com a finalidade de ser de fácil entendimento, de fácil programação e de fácil aplicação, além de ser multiplataforma, podendo ser configurado em ambientes Linux, Mac OS e Windows. Além disso, um grande diferencial deste dispositivo é ser mantido por uma comunidade que trabalha na filosofia open-source, desenvolvendo e divulgando gratuitamente seus projetos.

O equipamento é uma plataforma de computação física: são sistemas digitais ligados a sensores e atuadores, que permitem construir sistemas que percebam a realidade e respondem com ações físicas. Ele é baseado em uma placa microcontrolada, com acessos de Entrada/-Saída (I/O), sobre a qual foram desenvolvidas bibliotecas com funções que simplificam a sua programação, por meio de uma sintaxe similar à das linguagens C e C++.

O Arduino utiliza o microcontrolador Atmega. Um microcontrolador (também denominado MCU) é um computador em um chip, que contém um microprocessador, memória e periféricos de entrada/saída. Ele pode ser embarcado no interior de algum outro dispositivo, que, neste caso, é o Arduino, para que possa controlar suas funções ou ações.

Em resumo, o Arduino é um kit de desenvolvimento, que pode ser visto como uma unidade de processamento capaz de mensurar variáveis do ambiente externo, transformadas em um sinal elétrico correspondente, através de sensores ligados aos seus terminais de entrada. De posse da informação, ele pode processá-la computacionalmente. Por fim, ele pode ainda atuar no controle ou no acionamento de algum outro elemento eletro-eletrônico conectado ao terminal de saída.

Uma vez que o Arduino é baseado em um microcontrolador e, portanto, é programável, torna-se possível criar diversas aplicações diferentes com uma certa facilidade. Além disso, o próprio equipamento pode ser reutilizado, através de uma nova programação. Por sua vez, a sua programação é simplificada pela existência de diversas funções que controlam o dispositivo, com uma sintaxe similar a de linguagens de programação comumente utilizadas (C e C++).

Assim sendo, em um ambiente profissional, as características do Arduino fazem dele uma boa ferramenta de prototipagem rápida e de projeto simplificado. Por outro lado, em um ambiente acadêmico, ele pode ser perfeitamente utilizado como ferramenta educacional, uma vez que não requer do usuário conhecimentos profundos de eletrônica digital nem da programação de dispositivos digitais específicos.

4 Componentes necessários para construção do projeto

Segue esquematizado mais abaixo uma tabela com os componentes que foram utilizados para construção do projeto, bem como a quantidade respectivas de cada um deles.

Pode-se imaginar que a construção baseia-se em dois materiais principais, que englobam já em sua arquitetura todos os componentes necessários para o funcionamento do sistema. Sâo eles: o Arduino e o Shield *RFID*.

O restante são materiais para montagem do circuito e associação das peças, como os fios de conexão, o led e o resistor. Na tabela estão caracterizados os detalhes de cada componente.

Quantidade	Material
1	*Arduino UNO R3*
1	Sensor *RFID MRC522*
1	*Protoboard* Perfurada
7	Fios de conexão simples

Tabela 1: Materiais necessários para confecção do projeto

5 Prototipagem

Neste momento, inicia-se a parte da construção do hardware, estará esquematizado quais as conexões devem ser feitas.

Os nomes das portas podem variar para a cor do shield RC522 utilizado, o exemplo principal utiliza a nomenclatura de portas para o shield da cor vermelha, a varição nominal está na tabela conseguinte.

Assim, o circuito se dá dessa forma:

Arduino UNO R3	Sensor *RFID MRC522*
10	NSS (SDA)
13	SCK
11	MOSI
12	MISO
GND	GND
9	RST
3.3V	VCC (3.3V)

Tabela 2: Conexões necessárias para prototipagem

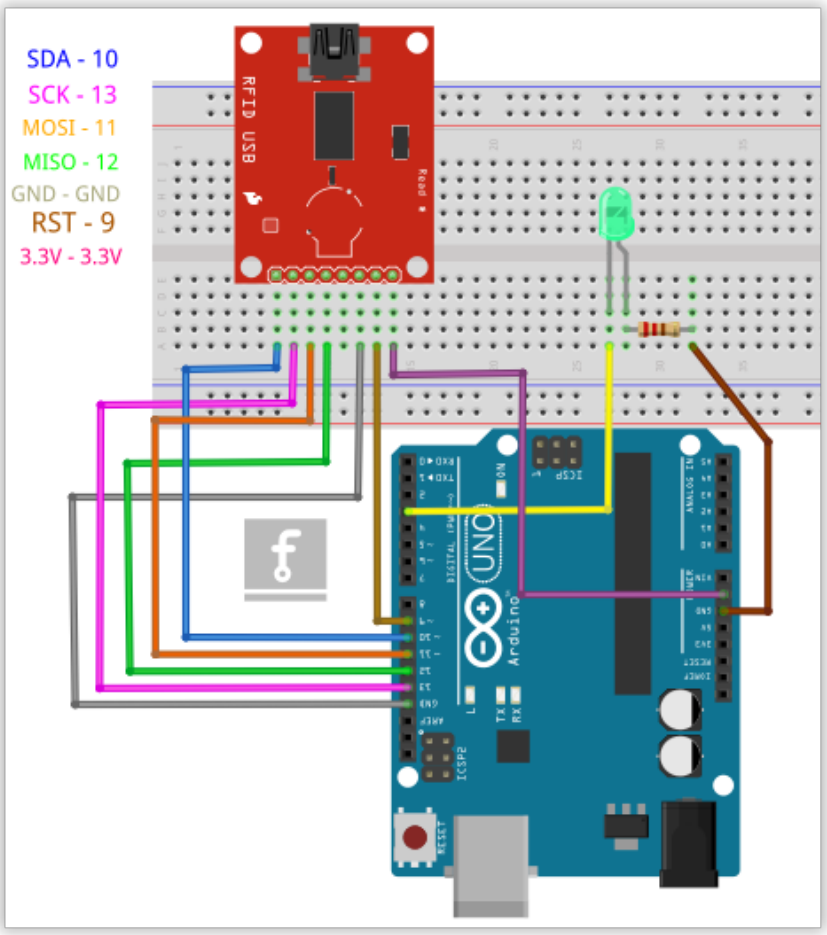

Figura 1: Esquema do hardware feito no software Fritzing.

6 Programação

Após toda a parte da construção do hardware do sistema, chegamos no nível da programação utilizada para execução das ações.

São dois códigos principais que foram desenvolvidos, um para programar o Arduino para adquirir os dados e outro para administrá-los em uma interface baseada em C.

6.1 Código para leitura do cartão

Esse é o código que deve ser inserido na interface de programação do Arduino, ele é responsável pela configuração de aquisição dos dados da tag utilizada, no caso, da carteirinha da UFF.

É de suma importância deixar claro, que para a execução do código .ino é necessária a instalação da biblioteca referente ao sensor *RFID*. Essa que está disponível em diversos sites na internet.

Após o *download* do arquivo referente à biblioteca necessária, ainda é preciso instalar nos diretórios do Arduino.

Na interface do compilador Arduino, siga o caminho *Sketch* → *Include Library* → *Add Library* e então vá no diretório onde está armazenada a biblitoca recém baixada.

```
#include <SPI.h>
#include <MFRC522.h>
```

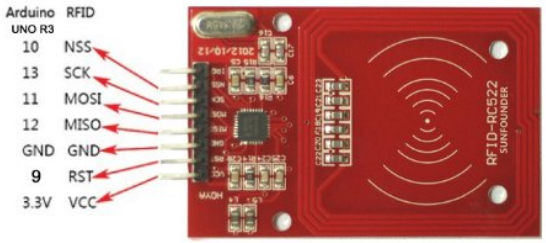

Figura 2: Foto real do sensor utilizado.

```
#define SS_PIN 10
#define RST_PIN 9

MFRC522 mfrc522(SS_PIN, RST_PIN); // Instancia a biblioteca com os pinos do arduino

void setup() {
  Serial.begin(9600); // Inicia a comunicação serial com o PC
SPI.begin();
mfrc522.PCD_Init();
}

void loop() {
if ( ! mfrc522.PICC_IsNewCardPresent()) {
return;
}
if ( ! mfrc522.PICC_ReadCardSerial()) {
return;
}
for (byte i = 0; i < mfrc522.uid.size; i++) {
            Serial.print(mfrc522.uid.uidByte[i]);
}
        delay(1000);
}
```

6.2 Aplicações para recepção dos dados no Windows e Linux

6.2.1 Código para controle da recepção dos dados no sistema operacional Windows

A seguir, o código em C para criação de uma aplicação em console que vai nos permitir administrar os dados adquiridos com os devidos fins.

Recomenda-se a utilização do software de programação *Visual Studio*, da *Microsft*. Isso se deve a utilização de bibliotecas e funções que já estão pré-configuradas no *Visual*.

Em poucas palavras, em sua primeira parte, o código apresenta instruções de verificação de possíveis erros de conexão do Arduino com a porta utilizada no computador (*host*), para depois definir algumas configurações básicas necessária para a interface serial realizada.

O menu presente no final do código controla a rotina *readArduino()* que inicia a aquisição dos dados que seriam, até então, "printados" na *Serial Monitor* do *Arduino*. Ao acioná-la

algumas funções encadeadas são iniciadas para preencher o vetor *buffer* com a *string* de dados da *tag* lida.

Feito isso, será retornado em console um valor que corresponde a *Tag ID* referente a cada carteira de identificação do aluno.

Para uma descrição minuciosa sobre as técnicas utilizadas recomendamos fortemente a leitura das referências citadas no final deste documento.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include "stdafx.h"
#include <Windows.h>
#include <locale.h>

void printErro() {
printf("\n :: Falha de conexão com o Arduino :: \n\n");
}

void readArduino(HANDLE hSerial) {
char buffer[20] = { 0 };
DWORD qtdBytesLida = 0;

printf("-------------------------------------------------\n\n");
printf(":: RECEBENDO DADOS DO ARDUINO ::\n\n");

while (strlen(buffer) == 0) {
if (!ReadFile(hSerial, buffer, 19, &qtdBytesLida, NULL)) {
//printErro();
}
}

printf("Identificador lido: %s \n\n", buffer);
printf("Conexão com arduino realizada com sucesso! \n\n");
printf("-------------------------------------------------\n\n");

}

int writeArduino(HANDLE hSerial, const char* buffer) {
DWORD qtdBytesEscrita = 0;

printf("-------------------------------------------------");
printf("\n\n");
printf(":: ENVIANDO DADOS AO ARDUINO ::\n\n");

if (!WriteFile(hSerial, buffer, 1, &qtdBytesEscrita, NULL)) {
//printErro();
return (-1);
```

```
}

return 0;
}

HANDLE conectArduino(LPCWSTR porta) {
HANDLE hSerial;

printf("-------------------------------------------------\n\n");
printf(":: REQUISITANDO CONEXAO COM O ARDUINO ::\n\n");

hSerial = CreateFile
(porta, GENERIC_READ | GENERIC_WRITE, 0,
0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);

if (hSerial == INVALID_HANDLE_VALUE) {
if (GetLastError() == ERROR_FILE_NOT_FOUND) {
//printErro();
}
//printErro();
}

DCB parametros = { 0 };

parametros.DCBlength = sizeof(parametros);
parametros.BaudRate = CBR_9600;
parametros.ByteSize = 8;
parametros.StopBits = ONESTOPBIT;
parametros.Parity = NOPARITY;

if (!SetCommState(hSerial, &parametros)) {
// error setting serial port state
//printErro();
}

COMMTIMEOUTS timeouts = { 0 };

timeouts.ReadIntervalTimeout = 10000;
timeouts.ReadTotalTimeoutConstant = 1000;
timeouts.ReadTotalTimeoutMultiplier = 0;
timeouts.WriteTotalTimeoutConstant = 100;
timeouts.WriteTotalTimeoutMultiplier = 0;

if (!SetCommTimeouts(hSerial, &timeouts)) {
//printErro();
}

return hSerial;
}
```

```
int _tmain(int argc, _TCHAR* argv[]){

int numero = 0;

while (numero != 3) {

printf(":: CONEXAO ARDUINO ~ C++ with RFID ::\n");

printf("\n\n");
printf("Digite:\n");
printf("1: para inciar a leitura no Arduino;\n");
printf("2: para sair");
printf("\n\n");

scanf_s(&numero);
printf("\n\n");

switch (numero){
case 1:
   readArduino();
break;

case 2:
   return 0;

default:
   break;
}

}

return 0;
}
```

6.2.2 Código para controle da recepção dos dados no sistema operacional Linux

Por fim, vamos dar uma olhada na versão do código específica para o sistema operacional Linux. Também é um código em linguagem C que cria uma aplicação em console para recebermos os valores lidos pelo *Arduin*.

Este código foi desenvolvido e testado com utilizando o próprio compilador C (GCC) que já vem embarcado no sistema operacional, por isso, diferente da versão Windows, não é necessária qualquer *software* específico para executar a aplicação.

Em poucas palavras, em sua primeira parte, o código iniciar definindo o caminho em que o Arduino foi conectado no computador, apóis isso apresenta instruções de verificação de possíveis erros de conexão do Arduino com a porta utilizada no computador(*host*), um bloco de linhas para configuração da interface de comunicação serial, semelhante a utilizada na versão do código para Windows e por fim a parte de menu de usuário para controlar a rotina.

Ao pressionar a tecla para iniciar a leitura, um vetor que espera a *string* com o valor da *tag*

lida é preenchido com os valores que iriam para *Serial Monitor*.

Feito isso, será retornado em console um valor que corresponde a *Tag ID* referente a cada carteira de identificação do aluno.

Novamente, vale enfatizar que, para uma descrição minuciosa sobre as técnicas utilizadas recomendamos fortemente a leitura das referências citadas no final deste documento.

```
#include <stdlib.h>
#include <stdio.h>

#include <sys/ioctl.h>
#include <fcntl.h>
#include <termios.h>

int main() {
/* Definindo o caminho da porta USB
conectada ao arduino */

char *portname = "/dev/ttyACM0";
char buf[256];

/* Função de término de rotina */
void getch(void) {
system("read stop");
}

int read_arduino(){

    /* Abre o descritor de arquivo em modo non-blocking */
    int fd = open(portname, O_RDWR | O_NOCTTY);

    /* Inicializa a struct de controle */
    struct termios toptions;


    /* Define as configurações padrão de comunicação */
    tcgetattr(fd, &toptions);
    cfsetispeed(&toptions, B9600);
    cfsetospeed(&toptions, B9600);
    toptions.c_cflag &= ~PARENB;
    toptions.c_cflag &= ~CSTOPB;
    toptions.c_cflag &= ~CSIZE;
    toptions.c_cflag |= CS8;
    toptions.c_cflag &= ~CRTSCTS;
    toptions.c_cflag |= CREAD | CLOCAL;
    toptions.c_iflag &= ~(IXON | IXOFF | IXANY);
    toptions.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);
    toptions.c_oflag &= ~OPOST;
```

```
    /* espera 12 chars */
    toptions.c_cc[VMIN] = 12;

    /* define o tempo de espera para o retorno da leitura */
    toptions.c_cc[VTIME] = 0;

    tcsetattr(fd, TCSANOW, &toptions);


    /* Faz o dump da serial */

    tcflush(fd, TCIFLUSH);


    int n = read(fd, buf, 128);

    /* printar a quantidade de bytes lidos */

    printf("%i bytes got read...\n", n);

    /* printar o buffer*/

    printf("O buffer contem essa tag:\n %s \n", buf);

    getch();
}

 int escolha = 0;
while (escolha != 2) {
here: printf("-------------------------------------------------\n\n");
printf(":: COMUNICACAO COM ARDUINO ::\n");
printf("\n\n");
printf("Digite:\n");
printf("   1: para receber uma tag;\n");
printf("   2: para sair");
printf("\n\n");

scanf("%d", &escolha);
switch (escolha){

case 1:{

            read_arduino();

            system("clear");

            goto here;
        }
```

```
default:
break;
}
}

return 0;



}
```

Referências

[1] Introdução ao kit de desenvolvimento Arduino (Grupo PET-Tele) *http://www.telecom.uff.br/pet/petws/downloads/tutoriais/arduino/Tut_Arduino.pdf*

[2] Tutorial sobre sistema de controle de acesso *RFID* (Grupo PET-Tele) *http://www.telecom.uff.br/pet/petws/downloads/tutoriais/rfid/Tut_Aquisicao_Dados_RFID.pdf*

[3] Windows Serial Port Programming (Robertson Bayer) *http://programmingworld.50webs.com/serial-win.pdf*

[4] How to read serial data from an Arduino in Linux with C (Chris Heydrick) *https://chrisheydrick.com/2012/06/12/how-to-read-serial-data-from-an-arduino-in-linux-with-c-part-1/*