

**PROJETO DE LINGUAGEM DE PROGRAMAÇÃO II**

**Professor:** Dr. Fernando Ferreira de Carvalho.

**Projeto**: Bebedouro Inteligente.

**Alunos:**

* Filipe Tabosa da Silva;
* Gleyson Rhuan Nascimento Campos.

Caruaru – 2017

**Introdução**

Em nosso campus temos um grande problema com relação ao controle de acesso de pessoas ao interior da universidade, onde qualquer pessoa pode ingressar no ambiente acadêmico já que não há maneiras de controlar o acesso. Desta forma é comum ver pessoas que não pertencem ao contexto da universidade utilizando seus recursos que são escassos, um dos inconvenientes causados por isso é a constante falta de água nos bebedouros que ficam nos corredores do campus, muitas vezes deixando os discentes, docentes e funcionários da faculdade em geral com sede. Abaixo será descrito a solução encontrada para sanar este contratempo.

**Objetivos**

O projeto tem como base, adaptar o(s) bebedouro(s) do campus de maneira que para ter acesso a água, o usuário aproxime um cartão RFID para liberar o sistema e a água seja dispensada. Onde não só haverá o controle da saída de água, mas também será possível através de sensores posicionados no bebedouro ter informações úteis para aperfeiçoar o uso do dispositivo, como por exemplo: emitir um alerta se a água estiver perto de acabar, alertando o funcionário responsável pela troca do garrafão antes que o dispositivo fique sem água...

Abaixo podemos ver uma imagem que ilustra o projeto:

****

Figura 1 - Esquema do Bebedouro Inteligente

OBS: Internamente há um sensor de nível de água para alertar quando o garrafão precisa ser trocado.

Display: Exibir informações do sistema.

Leitor RFID para ler credenciais dos usuários. 

Torneira c/Solenoide para controlar a saída de água.

Sensor Ultrassônico para verificar presença de copo e evitar desperdício.

**O Sistema**

O funcionamento do bebedouro é descrito resumidamente a seguir:

Ao aproximar um cartão do leitor, é feita a validação para saber se o usuário possui permissão para utilizar o equipamento, caso o usuário não possua autorização será exibida uma mensagem no display informando “Acesso negado”. Por outro lado, caso o usuário possua direito de utilização, é verificado o sensor ultrassônico para saber se o copo está próximo. O copo estando em uma distância aceitável, o sistema então ativa a saída da água enquanto o cartão estiver próximo e encerra a dispersão da água quando o cartão for afastado.

Ao fim da operação o sistema verifica o nível da água através do sensor de nível e caso o bebedouro esteja com água somente da sua reserva interna é acionada uma mensagem de atenção no display para que seja efetuada a troca do garrafão.

O cadastro dos usuários é feito através do cartão “mestre”, que possibilita a adição e exclusão de credenciais que ficam salvas na memória EEPROM do Arduino, evitando a perca de informações caso o equipamento seja desligado.

É interessante ressaltar que dependendo do leitor RFID utilizado é possível a utilização do cartão LEVA de transporte público da cidade de Caruaru, que opera em frequência aceitável para o leitor (Utilizado por grande parte dos estudantes) como credencial única para acesso ao sistema, já que cada cartão possui numeração única.

**Pontos Positivos**

* Controle no acesso;
* Monitoramento para evitar que fique sem água;
* Fácil implementação;
* Custo baixo;

**Pontos Negativos**

* Quantidade limitada de armazenamento de credenciais na EEPROM;
* Quantidade limitada de escrita na EEPROM;

**Situação da Versão Atual do Sistema**

O sistema foi implantado em versão de teste em um protótipo de bebedouro a fim de mostrar na prática o funcionamento idealizado no projeto. Sendo assim ainda não é uma versão final tendo em vista que podem ocorrer variações que podem influenciar nas funções a depender de eventuais necessidades de adaptação no seu funcionamento.

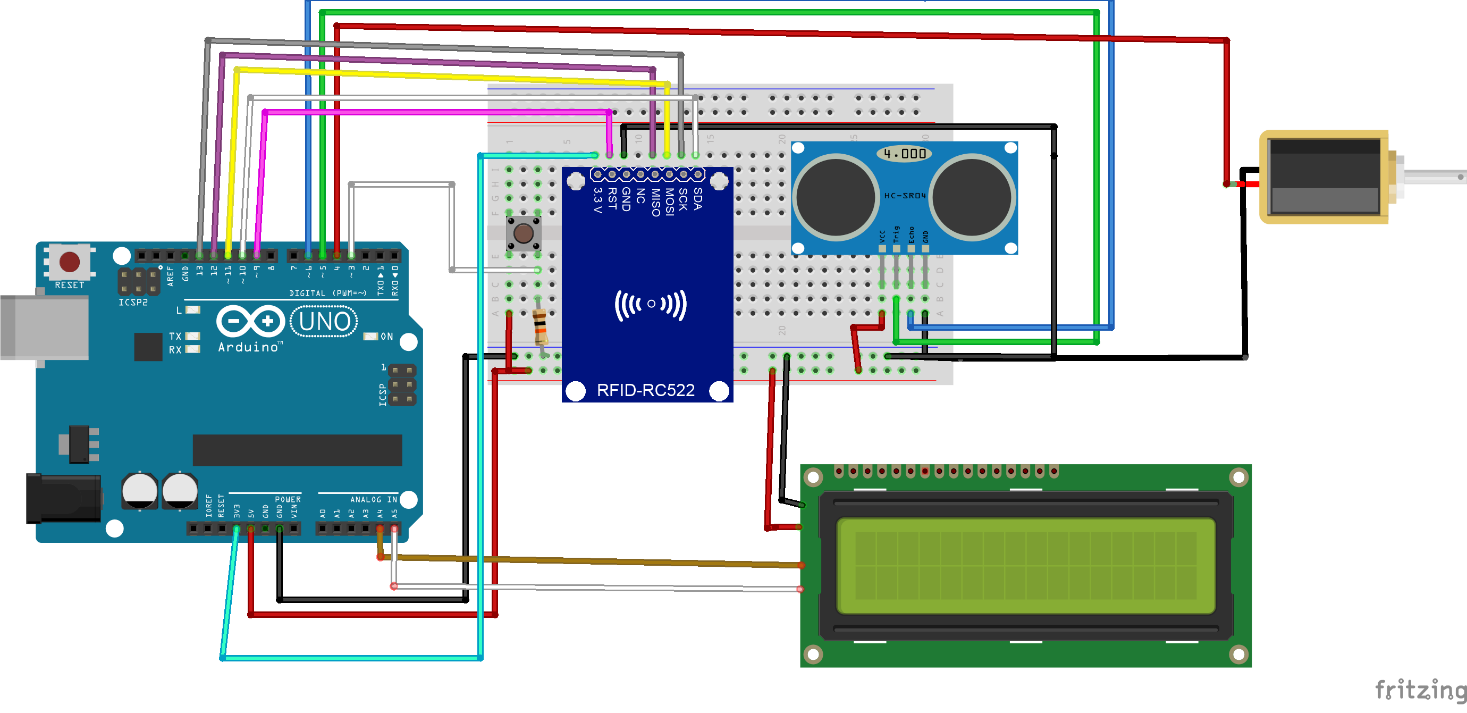
**Avaliação dos Custos do Projeto**

Quase todos os componentes utilizados foram emprestados pelo professor Fernando Carvalho e são listados a seguir:

|  |
| --- |
| Componente |
| Arduíno Uno R3 |
| protoBoard |
| display 16x2 |
| sensor de nível de água |
| Sensor ultrassônico |
| Sensor rfid |
| Relé |
| solenóide |
| jumpers |

**Infraestrutura**

Esquema de montagem:



**Código:**

**#include <EEPROM.h> // Biblioteca para facilita o uso da EEPROM**

**#include <SPI.h>**

**#include <MFRC522.h>**

**#include <Ultrasonic.h>**

**boolean match = false; // Inicia cartão como falso**

**boolean programMode = false; // Inicia o modo programador como falso**

**boolean replaceMaster = false;**

**boolean trocarGarrafao = false; // Variável para uso do sensor de nível de água**

**boolean condicaoPRI = false;**

**int successRead;**

**int ultra;**

**byte storedCard[4]; // Armazena uma ID lida da EEPROM**

**byte readCard[4]; // Armazena a identificação lida a partir do módulo RFID**

**byte masterCard[4]; // Armazena ID do cartão master lido da EEPROM**

**#define SS\_PIN 10 //Pino do RFID**

**#define RST\_PIN 9 //Pino do RFID**

**#define TRIG\_PIN 6 //Pino do Sensor Ultrassônico**

**#define ECHO\_PIN 7 //Pino do Sensor Ultrassônico**

**#define MNA\_PIN 2 //Medidor de nível de água**

**#define AR\_PIN 4 //Relé**

**#define wipeB 3 //Apagando memória de credenciais**

**MFRC522 mfrc522(SS\_PIN, RST\_PIN);**

**Ultrasonic ultrasonic(TRIG\_PIN,ECHO\_PIN);**

**void setup() {**

**Serial.begin(9600);**

**SPI.begin();**

**mfrc522.PCD\_Init(); // Inicializar MFRC522 Hardware**

**ShowReaderDetails(); // Mostrar detalhes de PCD - MFRC522 leitor de cartão**

**//Apagar Código se Botão Pressionado enquanto a configuração é executada (ligado) limpa EEPROM**

**if (digitalRead(wipeB) == LOW) { // Quando botão pressionado pino deve ficar baixo, botão conectado à terra**

**Serial.println(F("Botão de limpeza precionado"));**

**Serial.println(F("Você tem 15 segundos para cancelar"));**

**Serial.println(F("Será removido todos os registro, e não será poderá ser desfeito"));**

**delay(15000); // Tempo suficiente para cancelar a operação**

**if (digitalRead(wipeB) == LOW) { // Se o botão estiver precionado limpa a EEPROM**

**Serial.println(F("Iniciando limpeza da EEPROM"));**

**for (int x = 0; x < EEPROM.length(); x = x + 1) { //Loop pelos endereços da EEPROM**

**if (EEPROM.read(x) == 0) {**

**//Se endereço tiver 0 pule para o próximo**

**}**

**else {**

**EEPROM.write(x, 0); // Senão escreve 0 no endereço para "limpar"**

**}**

**}**

**Serial.println(F("EEPROM limpa com sucesso"));**

**}**

**else {**

**Serial.println(F("Reset Cancelado"));**

**}**

**}**

**if (EEPROM.read(1) != 143) {**

**Serial.println(F("Cartão Master Não Definido"));**

**Serial.println(F("Digitalizar um PICC para definir como Master Card"));**

**do {**

**successRead = getID(); // Define successRead para 1 quando obtemos leitura do leitor caso contrário 0**

**}**

**while (!successRead); // O programa fica aguardando uma leitura de cartão para prosseguir**

**for ( int j = 0; j < 4; j++ ) {**

**EEPROM.write( 2 + j, readCard[j] ); // Escreve o ID do cartão na EEPROM**

**}**

**EEPROM.write(1, 143);**

**Serial.println(F("Definido Master Card"));**

**}**

**Serial.println(F("-------------------"));**

**Serial.println(F("ID do Master Card"));**

**Serial.println(F("-------------------"));**

**for ( int i = 0; i < 4; i++ ) { // Ler cartão master da EEPROM**

**masterCard[i] = EEPROM.read(2 + i); // Salva na variável mastercard**

**Serial.print(masterCard[i], HEX);**

**}**

**Serial.println("");**

**Serial.println(F("-------------------"));**

**Serial.println(F("Tudo pronto"));**

**Serial.println(F("Esperando cartão"));**

**Serial.println(F("-------------------"));**

**}**

**void loop () {**

**checkLevelAgua();**

**do {**

**successRead = getID(); // Define successRead para 1 quando obter leitura do leitor caso contrário 0**

**if (digitalRead(wipeB) == LOW) {**

**Serial.println(F("Botão de limpeza pressionado"));**

**Serial.println(F("Master Card será apagado! Em 3 segundos"));**

**delay(3000);**

**if (digitalRead(wipeB) == LOW) {**

**EEPROM.write(1, 0);**

**Serial.println(F("Reiniciar o dispositivo para reprogramar Mastercard"));**

**while (1);**

**}**

**}**

**}**

**while (!successRead); //O programa fica aguardando uma leitura de cartão para prosseguir**

**if (programMode) {**

**if ( isMaster(readCard) ) { //Se ler o cartão master, saia do modo programador**

**Serial.println(F("-----------------------------"));**

**Serial.println(F("Cartão Mestre Digitalizado"));**

**Serial.println(F("Sair do modo de programação"));**

**Serial.println(F("-----------------------------"));**

**programMode = false;**

**return;**

**}**

**else {**

**if ( findID(readCard) ) { // Se o cartão digitalizado for conhecido, exclua-o**

**Serial.println(F("ID achado, removendo..."));**

**deleteID(readCard);**

**Serial.println(F("-----------------------------"));**

**Serial.println(F("Aproxime o cartão para ADD ou REMOVER da EEPROM"));**

**Serial.println(F("-----------------------------"));**

**}**

**else { //Se o cartão digitalizado não for conhecido adicione-o**

**Serial.println(F("Novo cartão lido, adicionando na EEPROM ..."));**

**writeID(readCard);**

**Serial.println(F("-----------------------------"));**

**Serial.println(F("Aproxime o cartão para ADD ou REMOVER da EEPROM"));**

**Serial.println(F("-----------------------------"));**

**}**

**}**

**}**

**else {**

**if ( isMaster(readCard)) { // Se ler cartão master, entre no modo de programação**

**programMode = true;**

**Serial.println(F("Modo de programação ativo"));**

**int count = EEPROM.read(0); // Leia o primeiro Byte da EEPROM que armazena o número de ID's na EEPROM**

**Serial.print(F("Existem "));**

**Serial.print(count);**

**Serial.print(F(" ID(s) na EEPROM"));**

**Serial.println("");**

**Serial.println(F("Aproxime um cartão para ADD ou REMOVER da EEPROM"));**

**Serial.println(F("Aproxime o cartão Master para sair do modo programação"));**

**Serial.println(F("-----------------------------"));**

**}**

**else {**

**if ( findID(readCard) ) { // Veja se o cartão está na EEPROM**

**Serial.println(F("Acesso permitido!"));**

**ultra = ultrasonic.convert(ultrasonic.timing(), Ultrasonic::CM);;**

**if (ultra < 3){ //Liberar a água**

**liberaAgua (true);**

**while (successRead);**

**liberaAgua (false);**

**checkLevelAgua();**

**} else {**

**Serial.println("Copo longe do bebedouro");**

**}**

**}**

**else { // Usuário sem permissão**

**Serial.println(F("Acesso negado!"));**

**}**

**}**

**}**

**}**

**int getID() {**

**// Preparando-se para a leitura de PICCs**

**if ( ! mfrc522.PICC\_IsNewCardPresent()) { //Se um novo PICC colocado no leitor RFID continuar**

**return 0;**

**}**

**if ( ! mfrc522.PICC\_ReadCardSerial()) { //Uma vez que um PICC colocado obter Serial e continuar**

**return 0;**

**}**

**// Só é compatível a leitura de cartões de 4bytes!**

**Serial.println(F("UID do cartão:"));**

**for (int i = 0; i < 4; i++) { //**

**readCard[i] = mfrc522.uid.uidByte[i];**

**Serial.print(readCard[i], HEX);**

**}**

**Serial.println("");**

**mfrc522.PICC\_HaltA(); // para leitura**

**return 1;**

**}**

**void ShowReaderDetails() {**

**// Get the MFRC522 software version**

**byte v = mfrc522.PCD\_ReadRegister(mfrc522.VersionReg);**

**Serial.print(F("MFRC522 Software Version: 0x"));**

**Serial.print(v, HEX);**

**if (v == 0x91)**

**Serial.print(F(" = v1.0"));**

**else if (v == 0x92)**

**Serial.print(F(" = v2.0"));**

**else**

**Serial.print(F(" (unknown),probably a chinese clone?"));**

**Serial.println("");**

**// When 0x00 or 0xFF is returned, communication probably failed**

**if ((v == 0x00) || (v == 0xFF)) {**

**Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?"));**

**Serial.println(F("SYSTEM HALTED: Check connections."));**

**while (true); // do not go further**

**}**

**}**

**void readID( int number ) {**

**int start = (number \* 4 ) + 2; // Descobrir a posição inicial**

**for ( int i = 0; i < 4; i++ ) { // Loop 4 vezes para obter os 4 bytes**

**storedCard[i] = EEPROM.read(start + i); // Atribuir valores lidos da EEPROM para o array**

**}**

**}**

**void writeID( byte a[] ) {**

**if ( !findID( a ) ) { // Antes de escrever para a EEPROM, verificar se cartão já é cadastrado**

**int num = EEPROM.read(0); // Obter o número de espaços utilizados, a posição 0 armazena o número de cartões de identificação**

**int start = ( num \* 4 ) + 6; // Descobrir onde começa o próximo slot**

**num++;**

**EEPROM.write( 0, num ); // Escreva a nova contagem para o contador**

**for ( int j = 0; j < 4; j++ ) {**

**EEPROM.write( start + j, a[j] ); // Escreva os valores do array para EEPROM na posição correta**

**}**

**Serial.println(F("ID adicionado com sucesso à EEPROM"));**

**}**

**else {**

**Serial.println(F("ERRO! Algum problema com o ID do cartão"));**

**}**

**}**

**void deleteID( byte a[] ) {**

**if ( !findID( a ) ) { // Antes de excluir da EEPROM, verifique se tem este cartão!**

**Serial.println(F("ERRO! Há algo de errado com ID ou EEPROM ruim"));**

**}**

**else {**

**int num = EEPROM.read(0); // Obter o número de espaços utilizados, a posição 0 armazena o número de cartões de identificação**

**int slot; // Descobrir o número do slot do cartão**

**int start; // = ( num \* 4 ) + 6; // Descobrir onde começa o próximo slot**

**int looping; // O número de vezes que o loop repete**

**int j;**

**int count = EEPROM.read(0); // Leia o primeiro Byte da EEPROM que armazena o número de cartões**

**slot = findIDSLOT( a ); // Descobrir o número do slot do cartão para apagar**

**start = (slot \* 4) + 2;**

**looping = ((num - slot) \* 4);**

**num--;**

**EEPROM.write( 0, num ); // Define um novo valor para o contador**

**for ( j = 0; j < looping; j++ ) {**

**EEPROM.write( start + j, EEPROM.read(start + 4 + j)); //Desloque os valores da matriz para 4 posições anteriores na EEPROM**

**}**

**for ( int k = 0; k < 4; k++ ) { // Deslocando Loop**

**EEPROM.write( start + j + k, 0);**

**}**

**Serial.println(F("ID removida com sucesso da EEPROM"));**

**}**

**}**

**boolean checkTwo ( byte a[], byte b[] ) {**

**if ( a[0] != NULL ) // Certifique-se de que há algo na matriz primeiro**

**match = true; // Suponha que eles correspondam no início**

**for ( int k = 0; k < 4; k++ ) {**

**if ( a[k] != b[k] ) // Se a! = B então defina match = false, um falha, todos falham**

**match = false;**

**}**

**if ( match ) {**

**return true;**

**}**

**else {**

**return false;**

**}**

**}**

**int findIDSLOT( byte find[] ) {**

**int count = EEPROM.read(0); // Leia o primeiro Byte da EEPROM**

**for ( int i = 1; i <= count; i++ ) { // Repetir uma vez para cada entrada EEPROM**

**readID(i); // Ler uma ID da EEPROM, ela é armazenada no storedCard[4]**

**if ( checkTwo( find, storedCard ) ) { // Verifique se o cartão armazenado leu da EEPROM**

**// É o mesmo que o cartão de identificação find [] passou**

**return i; // O número do slot do cartão**

**break;**

**}**

**}**

**}**

**boolean findID( byte find[] ) {**

**int count = EEPROM.read(0); // Leia o primeiro Byte da EEPROM**

**for ( int i = 1; i <= count; i++ ) { // Repetir uma vez para cada entrada EEPROM**

**readID(i); // Ler uma ID da EEPROM, ela é armazenada em storedCard[4]**

**if ( checkTwo( find, storedCard ) ) { // Verifique se o cartão armazenado leu da EEPROM**

**return true;**

**break;**

**}**

**else {**

**}**

**}**

**return false;**

**}**

**boolean isMaster( byte test[] ) {**

**if ( checkTwo( test, masterCard ) )**

**return true;**

**else**

**return false;**

**}**

**void checkLevelAgua(){**

**if (digitalRead(MNA\_PIN)) { // Nível da água está baixo;**

**trocarGarrafao = true; //Trocar garrafão;**

**Serial.println ("-> Agua acabando <-");**

**}**

**else { // Nível da água está alto;**

**trocarGarrafao = false;**

**}**

**}**

**void liberaAgua(boolean condicao){**

**if (condicao /\*true\*/){**

**digitalWrite(AR\_PIN, HIGH);**

**} else {**

**digitalWrite(AR\_PIN, LOW);**

**}**

**}**

**Referências**

< Acessados em maio de 2017>

<https://www.arduino.cc/>

<http://blog.filipeflop.com/sensores/sensor-ultrassonico-hc-sr04-ao-arduino.html#_ga=2.185137578.2011555212.1494987892-1782034102.1494793337>

<http://blog.filipeflop.com/wireless/controle-acesso-leitor-rfid-arduino.html#_ga=2.183851302.1874105706.1494987921-1782034102.1494793337>

<http://www.arduinoecia.com.br/2014/07/arduino-sensor-de-nivel-de-liquidos.html>

<http://blog.filipeflop.com/display/controlando-um-lcd-16x2-com-arduino.html#_ga=2.148518647.1319380480.1494987966-1782034102.1494793337>