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:

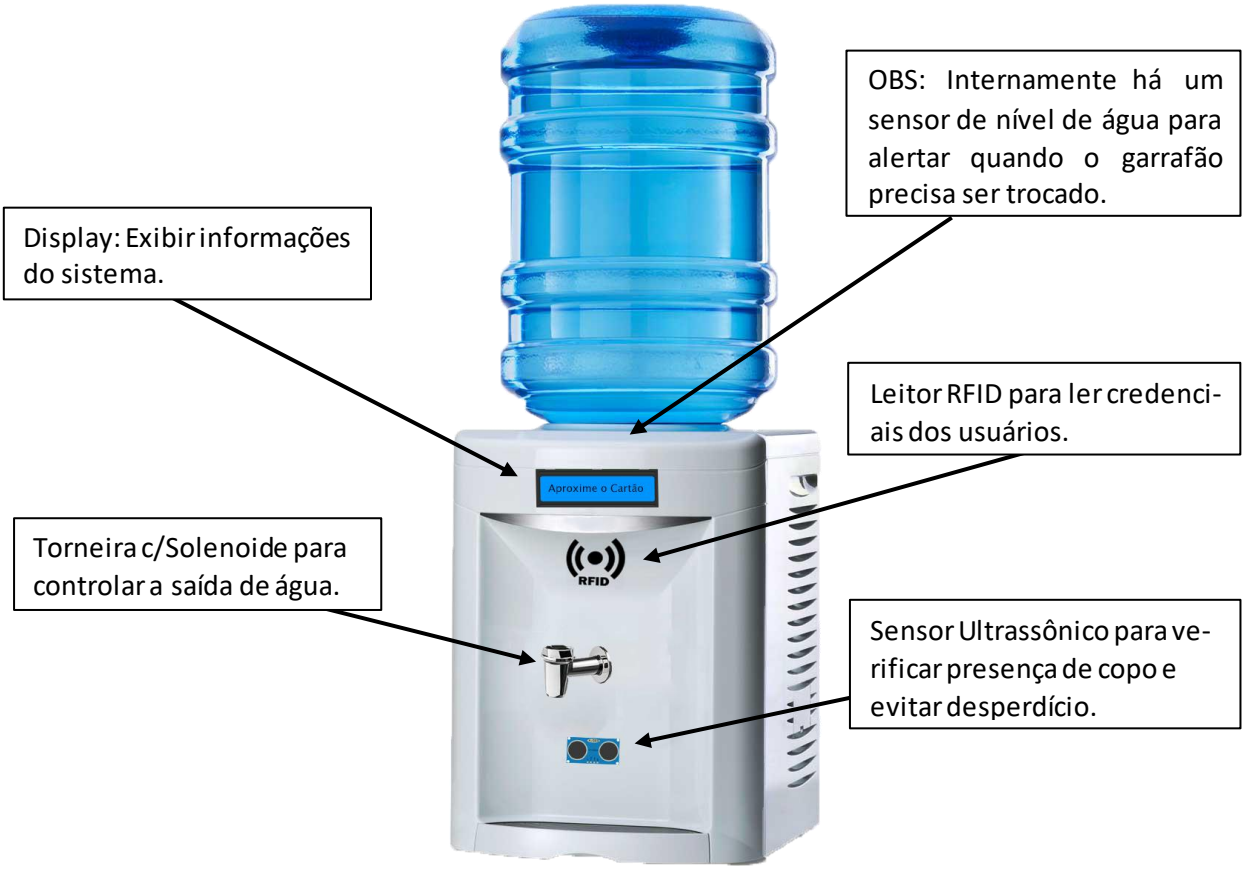


Figura 1 - Esquema do Bebedouro Inteligente

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.

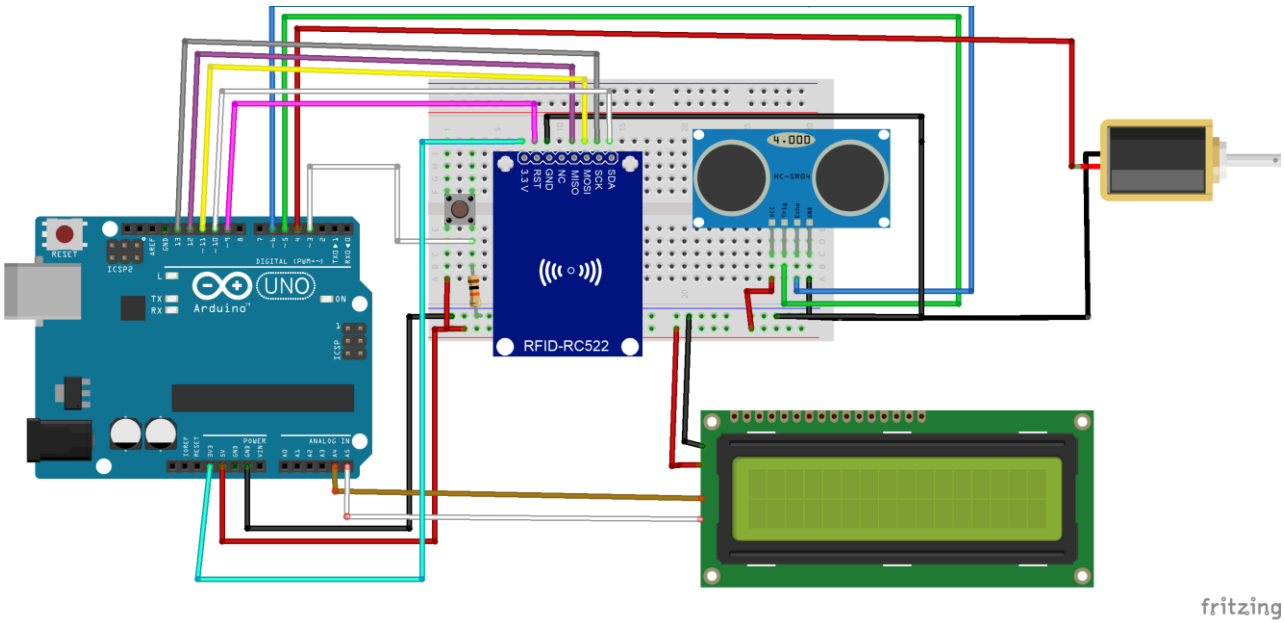
Avaliação dos Custos do Projeto

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:

```
1   #include <EEPROM.h>     // Biblioteca para facilita o uso da EEPROM
2   #include <SPI.h>
3   #include <MFRC522.h>
4   #include <Ultrasonic.h>
5
6   boolean match = false;        // Inicia cartão como falso
7   boolean programMode = false;   // Inicia o modo programador como falso
8   boolean replaceMaster = false;
9   boolean trocarGarrafao = false; // Variável para uso do sensor de nível de água
10  boolean condicaoPRI = false;
11
12  int successRead;
13  int ultra;
14
15  byte storedCard[4];  // Armazena uma ID lida da EEPROM
16  byte readCard[4];  // Armazena a identificação lida a partir do módulo RFID
17  byte masterCard[4];  // Armazena ID do cartão master lido da EEPROM
18
19  #define SS_PIN 10 //Pino do RFID
20  #define RST_PIN 9 //Pino do RFID
21  #define TRIG_PIN 6  //Pino do Sensor Ultrassônico
22  #define ECHO_PIN 7  //Pino do Sensor Ultrassônico
23
24  #define MNA_PIN 2 //Medidor de nível de água
25  #define AR_PIN 4 //Relé
26  #define wipeB 3 //Apagando memória de credenciais
27
28  MFRC522 mfrc522(SS_PIN, RST_PIN);
29  Ultrasonic ultrasonic(TRIG_PIN,ECHO_PIN);
30
31  void setup() {
32    Serial.begin(9600);
33    SPI.begin();
34    mfrc522.PCD_Init();    // Inicializar MFRC522 Hardware
```

```
35
36    ShowReaderDetails();  // Mostrar detalhes de PCD - MFRC522 leitor de cartão
37
38    //Apagar Código se Botão Pressionado enquanto a configuração é executada (ligado) limpa EEPROM
39    if (digitalRead(wipeB) == LOW) {  // Quando botão pressionado pino deve ficar baixo, botão conec-
40  tado à terra
41
42      Serial.println(F("Botão de limpeza precionado"));
43      Serial.println(F("Você tem 15 segundos para cancelar"));
44      Serial.println(F("Será removido todos os registro, e não será poderá ser desfeito"));
45
46      delay(15000);                    // Tempo suficiente para cancelar a operação
47      if (digitalRead(wipeB) == LOW) {    // Se o botão estiver precionado limpa a EEPROM
48        Serial.println(F("Iniciando limpeza da EEPROM"));
49        for (int x = 0; x < EEPROM.length(); x = x + 1) {    //Loop pelos endereços da EEPROM
50          if (EEPROM.read(x) == 0) {
51            //Se endereço tiver 0 pule para o próximo
52          }
53          else {
54            EEPROM.write(x, 0);     // Senão escreve 0 no endereço para "limpar"
55          }
56        }
57        Serial.println(F("EEPROM limpa com sucesso"));
58      }
59      else {
60        Serial.println(F("Reset Cancelado"));
61      }
62    }
63    if (EEPROM.read(1) != 143) {
64      Serial.println(F("Cartão Master Não Definido"));
65      Serial.println(F("Digitalizar um PICC para definir como Master Card"));
66      do {
67        successRead = getID();          // Define successRead para 1 quando obtemos leitura do leitor caso
68  contrário 0
69
70      }
71      while (!successRead);                // O programa fica aguardando uma leitura de cartão para prosse-
72  guir
73      for ( int j = 0; j < 4; j++ ) {
74        EEPROM.write( 2 + j, readCard[j] ); // Escreve o ID do cartão na EEPROM
75      }
76      EEPROM.write(1, 143);
77      Serial.println(F("Definido Master Card"));
78    }
79    Serial.println(F("--------------------"));
80    Serial.println(F("ID do Master Card"));
81    Serial.println(F("--------------------"));
82    for ( int i = 0; i < 4; i++ ) {        // Ler cartão master da EEPROM
83      masterCard[i] = EEPROM.read(2 + i);   // Salva na variável mastercard
84      Serial.print(masterCard[i], HEX);
85    }
86    Serial.println("");
87    Serial.println(F("--------------------"));
88    Serial.println(F("Tudo pronto"));
89    Serial.println(F("Esperando cartão"));
90    Serial.println(F("--------------------"));
91   }
```

```
92
93  void loop () {
94   checkLevelAgua();
95   do {
96    successRead = getID(); // Define successRead para 1 quando obter leitura do leitor caso contrário 0
97    if (digitalRead(wipeB) == LOW) {
98     Serial.println(F("Botão de limpeza pressionado"));
99     Serial.println(F("Master Card será apagado! Em 3 segundos"));
100    delay(3000);
101    if (digitalRead(wipeB) == LOW) {
102     EEPROM.write(1, 0);
103     Serial.println(F("Reiniciar o dispositivo para reprogramar Mastercard"));
104     while (1);
105    }
106   }
107  }
108
109  while (!successRead);   //O programa fica aguardando uma leitura de cartão para prosseguir
110
111  if (programMode) {
112   if ( isMaster(readCard) ) { //Se ler o cartão master, saia do modo programador
113    Serial.println(F("-----------------------------"));
114   Serial.println(F("Cartão Mestre Digitalizado"));
115    Serial.println(F("Sair do modo de programação"));
116    Serial.println(F("-----------------------------"));
117    programMode = false;
118    return;
119   }
120   else {
121    if ( findID(readCard) ) { // Se o cartão digitalizado for conhecido, exclua-o
122     Serial.println(F("ID achado, removendo..."));
123     deleteID(readCard);
124     Serial.println(F("-----------------------------"));
125     Serial.println(F("Aproxime o cartão para ADD ou REMOVER da EEPROM"));
126   Serial.println(F("-----------------------------"));
127    }
128    else {              //Se o cartão digitalizado não for conhecido adicione-o
129     Serial.println(F("Novo cartão lido, adicionando na EEPROM ..."));
130     writeID(readCard);
131     Serial.println(F("-----------------------------"));
132     Serial.println(F("Aproxime o cartão para ADD ou REMOVER da EEPROM"));
133   Serial.println(F("-----------------------------"));
134    }
135   }
136  }
137  else {
138   if ( isMaster(readCard)) {   // Se ler cartão master, entre no modo de programação
139    programMode = true;
140    Serial.println(F("Modo de programação ativo"));
141    int count = EEPROM.read(0);   // Leia o primeiro Byte da EEPROM que armazena o número de ID's
142 na EEPROM
143    Serial.print(F("Existem "));
144    Serial.print(count);
145    Serial.print(F(" ID(s) na EEPROM"));
146    Serial.println("");
147    Serial.println(F("Aproxime um cartão para ADD ou REMOVER da EEPROM"));
148    Serial.println(F("Aproxime o cartão Master para sair do modo programação"));
```

```
149         Serial.println(F("-----------------------------"));
150       }
151       else {
152         if ( findID(readCard) ) { // Veja se o cartão está na EEPROM
153           Serial.println(F("Acesso permitido!"));
154           ultra = ultrasonic.convert(ultrasonic.timing(), Ultrasonic::CM);;
155           if (ultra < 3){ //Liberar a água
156           liberaAgua (true);
157           while (successRead);
158           liberaAgua (false);
159           checkLevelAgua();
160       } else {
161         Serial.println("Copo longe do bebedouro");
162       }
163         }
164         else {     // Usuário sem permissão
165           Serial.println(F("Acesso negado!"));
166         }
167       }
168     }
169   }
170
171   int getID() {
172     // Preparando-se para a leitura de PICCs
173     if ( ! mfrc522.PICC_IsNewCardPresent()) { //Se um novo PICC colocado no leitor RFID continuar
174       return 0;
175     }
176     if ( ! mfrc522.PICC_ReadCardSerial())  {  //Uma vez que um PICC colocado obter Serial e continuar
177       return 0;
178     }
179     // Só é compatível a leitura de cartões de 4bytes!
180     Serial.println(F("UID do cartão:"));
181     for (int i = 0; i < 4; i++) {  //
182       readCard[i] = mfrc522.uid.uidByte[i];
183       Serial.print(readCard[i], HEX);
184     }
185     Serial.println("");
186     mfrc522.PICC_HaltA();  // para leitura
187     return 1;
188   }
189
190   void ShowReaderDetails() {
191     // Get the MFRC522 software version
192     byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
193     Serial.print(F("MFRC522 Software Version: 0x"));
194     Serial.print(v, HEX);
195     if (v == 0x91)
196       Serial.print(F(" = v1.0"));
197     else if (v == 0x92)
198       Serial.print(F(" = v2.0"));
199     else
200       Serial.print(F(" (unknown),probably a chinese clone?"));
201     Serial.println("");
202     // When 0x00 or 0xFF is returned, communication probably failed
203     if ((v == 0x00) || (v == 0xFF)) {
204       Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?"));
205       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 nú-
mero 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 nú-
mero 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 posi-
ções 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"));
```

```
263     }
264   }
265
266   boolean checkTwo ( byte a[], byte b[] ) {
267     if ( a[0] != NULL )      // Certifique-se de que há algo na matriz primeiro
268       match = true;      // Suponha que eles correspondam no início
269     for ( int k = 0; k < 4; k++ ) {
270       if ( a[k] != b[k] )    // Se a! = B então defina match = false, um falha, todos falham
271         match = false;
272     }
273     if ( match ) {
274       return true;
275     }
276     else  {
277       return false;
278     }
279   }
280
281   int findIDSLOT(  byte find[] ) {
282     int count = EEPROM.read(0);       // Leia o primeiro Byte da EEPROM
283     for ( int i = 1; i <= count; i++ ) {   // Repetir uma vez para cada entrada EEPROM
284       readID(i);              // Ler uma ID da EEPROM, ela é armazenada no storedCard[4]
285       if ( checkTwo(  find, storedCard ) ) {   // Verifique se o cartão armazenado leu da EEPROM
286         // É o mesmo que o cartão de identificação find [] passou
287         return i;        // O número do slot do cartão
288         break;
289       }
290     }
291   }
292
293   boolean findID(  byte find[] ) {
294     int count = EEPROM.read(0);      // Leia o primeiro Byte da EEPROM
295     for ( int i = 1; i <= count; i++ ) {   // Repetir uma vez para cada entrada EEPROM
296       readID(i);          // Ler uma ID da EEPROM, ela é armazenada em storedCard[4]
297       if ( checkTwo(  find, storedCard ) ) {   // Verifique se o cartão armazenado leu da EEPROM
298         return true;
299         break;
300       }
301       else {
302       }
303     }
304     return false;
305   }
306
307   boolean isMaster(  byte test[] ) {
308     if ( checkTwo(  test, masterCard ) )
309       return true;
310     else
311       return false;
312   }
313
314   void checkLevelAgua(){
315
316     if (digitalRead(MNA_PIN)) {      // Nível da água está baixo;
317       trocarGarrafao  = true;     //Trocar garrafão;
318     Serial.println ("-> Agua acabando  <-");
319     }
```

```
320     else {              // Nível da água está alto;
321       trocarGarrafao = false;
322     }
323   }
324
325   void liberaAgua(boolean condicao){
326     if (condicao /*true*/){
327       digitalWrite(AR_PIN, HIGH);
328     } else {
329       digitalWrite(AR_PIN, LOW);
330     }
331   }
```

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