UMA ARQUITETURA PARA CONTROLE DE ACESSO À AMBIENTES FÍSICOS: Automatização de um sistema de fechadura utilizando Arduino e Android

Jocimar Roberto Silva[*]
Everton Rafael da Silva[**]
Breno Lisi Romano[***]

RESUMO

Este relatório técnico tem por finalidade propiciar a automatização de portas que dão acesso a ambientes restritos de forma a criar uma opção computacional para o controle destes acessos. Como proposta de projeto, foi desenvolvido uma solução automatizada com a criação de um protótipo Arduino para controlar a liberação do acesso de um ambiente restrito através da autenticação pelo utilizador no momento do acesso. O desenvolvimento da parte física foi realizado por meio de um protótipo que realizará a interação dos componentes Arduino integrantes do projeto, afim de, demonstrar a viabilidade de sua utilização. Para a parte lógica da solução Arduino foi desenvolvido um sistema que gerencia a interação entre os componentes ligados a placa Arduino Mega 2560 R3 utilizando o Arduino IDE. Ainda para a parte lógica foi desenvolvido um aplicativo Android para o dispositivo móvel que realizará a coleta dos registros de acesso armazenados no Arduino quando solicitado via aplicação e as exibirão em uma lista na tela do dispositivo. Foi realizada uma pesquisa de mercado para mensurar o custo e viabilidade para produção do protótipo proposto neste projeto. Tanto os desenvolvimentos da parte física, bem como da parte lógica foram devidamente documentados. Ao final do desenvolvimento foi apresentado o protótipo que foi finalizado com sucesso e de produção viável, quando comparado à soluções comerciais similares do mercado.

Palavras-chave: Arduino, Android, relatório técnico, controle de acesso, dispositivo móvel, registro de acessos, Arduino Mega 2560 R3, protótipo, projeto.

[*] Aluno do curso Lato Sensu de Especialização em Desenvolvimento de Aplicações para Dispositivos Móveis do IFSP – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – campus São João da Boa Vista.
[**] Professor do IFSP – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – campus São João da Boa Vista.
[***] Professor do IFSP – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – campus São João da Boa Vista.

1 INTRODUÇÃO

Atualmente a segurança patrimonial é tratada como causa de apreensão nos diversos segmentos de edificações comerciais de acordo com Moreira (2007), onde o mercado imobiliário utiliza-se desta preocupação para agregá-la como uma das vantagens em sua comercialização. Para Moreira (2007), para se proteger da onda de violência e invasão que não escolhe mais o tipo de local que irá agir, está sendo criada uma nova arquitetura onde as instalações estão sendo protegidas por barreiras físicas e controles de acesso. Esse tipo de proteção vem acompanhada de soluções tecnológicas com a inclusão de equipamentos de segurança.

Dentro deste contexto é possível afirmar que os estabelecimentos dentro de sua estrutura devem garantir a segurança no acesso à ambientes restritos para garantir a integridade de seus bens, sejam eles materiais, financeiros ou intelectuais.

O valor desses bens deve ser calculado não apenas em custo de aquisição, mas principalmente em prejuízos para toda a instituição caso sejam danificados ou até mesmo caso ocorra um sinistro (roubo) destes bens. Por isso, é imprescindível que se tenha um controle de acesso aos ambientes restritos que em muitos casos não mantém qualquer tipo de registro do utilizador, seja ele manual (papel) ou informatizado (digital), garantindo assim uma maior segurança.

A proposta deste relatório técnico é a criação de uma solução automatizada para o controle de acesso à ambientes restritos, através da implementação de uma arquitetura distribuída para controle de ambientes físicos utilizando Arduino e Android.

Este projeto segue a linha de pesquisa da engenharia eletrônica e computação, tendo como objetivo propiciar a automatização de tarefas e garantir uma maior segurança e controle de utilização.

Será executada a confecção de um protótipo de trava automatizada composta por uma placa Arduino Mega 2560 R3 que realizará, por intermédio de programação, o controle dos demais módulos que serão:

1. Um módulo de identificação por radiofrequência (RFID – *Radio Frequency IDentification*), modelo MFRC522 para a autenticação do usuário via programação Arduino;
2. Um módulo Bluetooth modelo HC-06 para a conexão entre o dispositivo móvel e o Arduino;
3. Um módulo de Relógio em Tempo Real (RTC – *Real Time Clock*) modelo DS3231 para possibilitar o registro da data / hora da tentativa de acesso;

4. Um *Display* de Cristal Líquido (LCD - *Liquid Cristal Display*) modelo 16x2 HD44780 para retornar informações do Arduino para o usuário;
5. Um módulo de armazenamento via Cartão Leitura/Escrita (MicroSD – *Secure Digital*) para registro das marcações e uma mini trava eletromagnética Solenoide 12V para destravamento da porta que será acionada via programação Arduino após autenticação via módulo RFID modelo MFRC522;

Para verificar se o utilizador possui permissão de acesso ao ambiente, o Arduino realizará uma consulta local via identificação de uma etiqueta (*Tag*) vinculada a um (Cartão ou Chaveiro) RFID MFRC522 previamente cadastrada no sistema Arduino que através de uma função realizará uma verificação quanto as *tags* cadastradas para saber se o usuário possui ou não permissão de acesso ao ambiente solicitado. Caso o retorno seja positivo, é liberado o acesso com o destravamento da porta, onde o utilizador é informado visualmente pelo display Arduino através da mensagem de permissão "Olá usuário acesso permitido", além de ser emitido 2 *Bips* via *Buzzer* e um LED verde posicionado no painel de acesso RFID piscar por duas vezes. Caso contrário, o utilizador é informado visualmente no display do Arduino através das mensagens "Cartão desconhecido" e "Acesso negado contate o Adm" que ele não possui acesso a este ambiente e é orientado a contatar o responsável pela administração do ambiente restrito, além de ser emitido 1 *Bip* via *Buzzer* e um LED vermelho posicionado no painel de acesso RFID piscar uma vez. Todas as tentativas de acesso são registradas através de um *Log* (Registro de eventos informatizado) em um arquivo de texto locado no cartão de memória.

Este projeto visa atender a todos os utilizadores de ambientes com acesso restrito, criando uma mecânica de controle da liberação de acesso a estes ambientes através de registro informatizado, que posteriormente poderá ser consultado pelo aplicativo Android via Bluetooth.

2 TRABALHOS RELACIONADOS

No trabalho realizado por Kamogawa e Miranda (2012), foi apresentado pelos autores uma solução Arduino para acionamento de válvulas solenoides que têm a função de automatizar um sistema de análise de fluxo de água.

Através da programação do Arduino foi criada uma rotina de acionamento das válvulas solenoides para liberação e contenção de fluxo e adição de reagentes com intervalos de tempo predeterminados e repetições sequenciais de 5 (cinco) vezes para

posterior análise das amostras de água. Desta forma foi realizada uma automatização de processos através da manipulação de fluxo de água com a ajuda de válvulas solenoides controlados por um módulo Arduino (KAMOGAWA e MIRANDA, 2012).

O trabalho proposto por Lima (2013) teve por finalidade a criação de uma solução automatizada para controle de acesso físico seguro utilizando Android e NFC (*Near Field Communication* – ou em tradução livre, "comunicação por proximidade"), sendo que esta comunicação é realizada pelo pareamento físico (alguns centímetros) entre um dispositivo móvel com o recurso NFC e um terminal disposto em um módulo NFC acoplado a uma placa Arduino que realiza o controle das liberações de acesso via internet junto a um Servidor WEB.

A maior dificuldade no projeto idealizado por Lima (2013), está no fato de que o recurso NFC encontra-se atualmente disponível apenas em alguns dispositivos de última geração restringindo assim a gama de usuários que teriam acesso a este Sistema através deste recurso.

De acordo com Sharma (2013), a implementação indiscriminada de RFID pode trazer riscos de segurança. Por exemplo, um leitor pode emitir periodicamente requisições e qualquer dispositivo dentro do alcance pode responder a este leitor, que pode redirecionar as informações a um banco de dados.

Até o momento, RFID enfrenta principalmente três tipos de ameaças de segurança:

- Interceptação das informações das tags;
- Tags podem ser crackeadas;
- Tags podem ser clonadas. (SHARMA, 2013).

Os sistemas RFID segundo Mota (2012), logo se tornaram bastante presentes, o que levou algumas pessoas a pensarem em um novo comportamento quanto a consumidores e a cidadãos comuns. Este fato deve-se à ubiquidade da tecnologia, ou seja, a maioria das pessoas comuns utilizarão a tecnologia, sem mesmo saber que estão utilizando, gerando alguns problemas já citados relacionados à privacidade. No entanto prevemos que os benefícios serão bem maiores que os problemas. Mota (2012) concluiu em seu estudo da literatura sobre o RFID que mesmo que possam haver possíveis problemas de segurança e privacidade e que potencialmente possam ser apresentados pelo sistema RFID, um esquema de autenticação mútua será provavelmente a melhor alternativa para resolver o problema, o que justifica uma grande quantidade de publicações a respeito de diversos protocolos de autenticação.

No trabalho de Silva (2017) foi apresentado um projeto Arduino para a automatização de um veículo provido de sensores a serem controlados por um aplicativo Android para realizar rotinas pertinentes a um automóvel, tais como, acelerar, frear, virar à direita, virar à esquerda, sempre visando atender as necessidades cognitivas tanto de adultos, jovens e crianças. Neste trabalho foi descrito o funcionamento do módulo Bluetooth, o qual utiliza basicamente 4 (quatro) ligações para se conectar fisicamente a placa Arduino.

São elas:

- As portas de comunicação TX e RX para a troca de dados;
- A porta GND (*Ground* ou terra);
- Além da porta de conexão de energia conforme figura 1 abaixo:

Neste cenário, de acordo com Silva (2016), para que a conexão entre o Arduino Mega 2560 R3 e o módulo Bluetooth HC-05 seja efetiva se faz necessário alguns ajustes relacionados à comunicação para garantir que os módulos possam se comunicar corretamente, o procedimento é realizado com o acesso a configuração padrão de fábrica, habilitando-se o modo administrador do componente para que via software possam ser alterados o nome e senha do módulo.

3 ABORDAGEM PROPOSTA

Nesta seção será apresentada a metodologia utilizada para a construção do protótipo Arduino que ficará responsável pela parte física (*hardware*) do projeto, demonstrando cada componente utilizado individualmente e sua funcionalidade dentro do sistema de fechaduras, além da sua comunicação com o aplicativo Android.

Inicialmente serão definidos os componentes e ferramentas utilizados para a montagem do protótipo.

Após serão descritas as etapas para o desenvolvimento do protótipo (*hardware*) demonstrando os passos de construção física e também da programação do *software* embarcado no Arduino.

Posteriormente será implementado o aplicativo Android que será adicionado ao protótipo final. O aplicativo Android realizará a comunicação com o Arduino através do módulo Bluetooth e realizará a coleta das informações contidas no arquivo log.txt gravado no módulo leitor de cartão microSD, que é o componente Arduino responsável por armazenar estas informações.

A delimitação do tema foi realizado através do estudo de projetos anteriores, que possam contribuir de alguma forma para o desenvolvimento do projeto.

Será realizado a construção e testes do protótipo, que será devidamente documentado passo a passo.

Após o término do desenvolvimento serão analisados os resultados obtidos com o protótipo quanto a mecânica de funcionamento e custo de produção e verificados eventuais melhorias futuras.

3.1 DEFINIÇÃO DOS COMPONENTES E FERRAMENTAS UTILIZADOS NO DESENVOLVIMENTO

No desenvolvimento deste projeto foram utilizados além de uma placa Arduino Mega 2560 R3 para inclusão de toda a programação do *software* embarcado responsável pela automatização das funcionalidades do hardware, vários módulos e componentes Arduino para a construção física do protótipo do projeto e dentre os quais é possível citar: 01 *kit* módulo RFID modelo MFRC522 com 02 duas tags (1 cartão + 1 chaveiro), 01 módulo Display LCD 16x2 HD44780, 01 módulo Cartão Escrita/Leitura microSD, 01 módulo Relógio em Tempo Real RTC modelo DS3231, 01 módulo Bluetooth HC-06, 01 modulo Relé 5V 2 canais, 01 Mini trava eletromagnética Solenoide 12v, 01 Bateria 12v, 01 Potenciômetro 10k, 01 *Protoboard* 830 pinos, 01 *Buzzer* (buzina), 01 Cabo USB, 01 Fonte 12V, 01 Botão (*pushbutton*), 06 Resistores 330Ω (*ohms*), 01 Diodo modelo 1N4007, cabos *Jumpers*, fios de energia, 01 caixa plástica para fixação e organização dos componentes Arduino e 01 *display* plástico para fixação dos módulos RFID MFRC522, *display* LCD 16x2 HD44780 e 02 Leds (01 vermelho e 01 verde).

Os materiais utilizados para a construção da porta foram: placa de madeira MDF, 02 dobradiças de alumínio, parafusos, pistola de cola quente.

Como plataforma de desenvolvimento do software embarcado Arduino foi utilizado a IDE Arduino, para o aplicativo Android foi utilizado a IDE Android Studio e para a criação dos Esquemáticos foi utilizado o aplicativo Adobe Fireworks CS6.

3.2 CONFIGURAÇÃO DO MÓDULO RFID MFRC522

Segundo Ahson e Ilyas (2008) a Identificação por Radiofrequência (*RFID – Radio Frequency Identification*) é uma tecnologia que utiliza ondas de radiofrequência para transmissão de dados.

As potenciais aplicações para sistemas de RFID são inúmeras, variando desde a simples identificação remota de objetos até avançadas utilizações em cadeia de suprimentos, segurança e rastreamento de objetos (AHSON e ILYAS, 2008).

Um sistema RFID é formado por 3 componentes básicos (Want, 2006) (Verdult, 2008):

- Transponder ou tag RFID: localizada no objeto a ser identificado, armazenando o código de identificação.
- Transceiver ou leitor RFID: responsável pela leitura/escrita na *tag*.
- Middleware ou banco de dados: responsável pelo processamento da informação obtida pelo leitor.

Para a criação do protótipo optou-se pela utilização de um módulo RFID MFRC522 para validar os acessos aos ambientes restritos, onde a checagem será feita utilizando-se de *tags* ou *transponders*, para realizar a confirmação de seu código de acesso junto ao banco de dados ou *middleware* via terminal de acesso ou *transceiver* pela aproximação da *tag* ao campo eletromagnético do terminal.

O trecho de código da programação Arduino com a função responsável pela checagem e validadação da *tag* está descrita no Quadro 1 e o vetor para inclusão de novas *tags* descrita no Quadro 2:

Quadro 1 - Funcão que realiza a leitura e validação das tags em um vetor de tags cadastradas

```
// Funcao que realiza a Leitura das Tags
void Leitura()
{
   IDtag = ""; // Inicialmente IDtag deve estar vazia.

   // Verifica se existe uma Tag presente
   if ( !LeitorRFID.PICC_IsNewCardPresent() || !LeitorRFID.PICC_ReadCardSerial() )
   {
      delay(50);
      return;
   }
   // Pega o ID da Tag através da função LeitorRFID.uid e Armazena o ID na variável IDtag
   for (byte i = 0; i < LeitorRFID.uid.size; i++)
   {
      IDtag.concat(String(LeitorRFID.uid.uidByte[i], HEX));
   }
   // Compara o valor do ID lido com os IDs armazenados no vetor TagsCadastradas[ ]
   for (int i = 0; i < (sizeof(TagsCadastradas)/sizeof(String)); i++)
   {
     if(  IDtag.equalsIgnoreCase(TagsCadastradas[i])  )
```

```
    {
        Permitido = true; // Variável Permitido assume valor verdadeiro caso o ID Lido esteja cadastrado
    }
  }
  if(Permitido == true)
  {
    acessoLiberado(); //Se a variável Permitido for verdadeira será chamada a função acessoLiberado()
  }
  else
  {
    acessoNegado(); //Se não será chamada a função acessoNegado()
  }
  delay(2000); // aguarda 2 segundos para efetuar uma nova leitura
}
```

Quadro 2 - Vetor responsávelo por armazenar os códigos das tags permitidas

```
// Vetor responsável por armazenar os ID's das Tag's cadastradas
String TagsCadastradas[ ] = {"45a3b14f", "Nova_Tag"}; // Chaveiro Cadastrado - ID 45a3b14f
```

O acoplamento do módulo RFID MFRC522 a placa Arduino Mega 2560 R3 de acordo com a documentação oficial é realizado através da conexão dos pinos (*SDA or SS – Slave Select*) utilizado pelo mestre (*master*) para ativar ou desativar dispositivos específicos onde:

- (SCK - *Serial Clock*), é utilizado como temporizador para sincronizar a transmissão de dados gerada pelo mestre (*master*);
- (MOSI - *Master Out Slave In*), é utilizado pelo mestre para envio de dados para os periféricos;
- (MISO - *Master In Slave Out*), é utilizado pelo escravo (slave) para enviar dados para o mestre;
- (GND – *Ground*), é utilizado pelo fio terra, (RST – *Reset*) utilizado para restabelece ou reiniciar o periférico e 3,3V utilizado pelo pino de tensão do periférico conforme Figura 1:

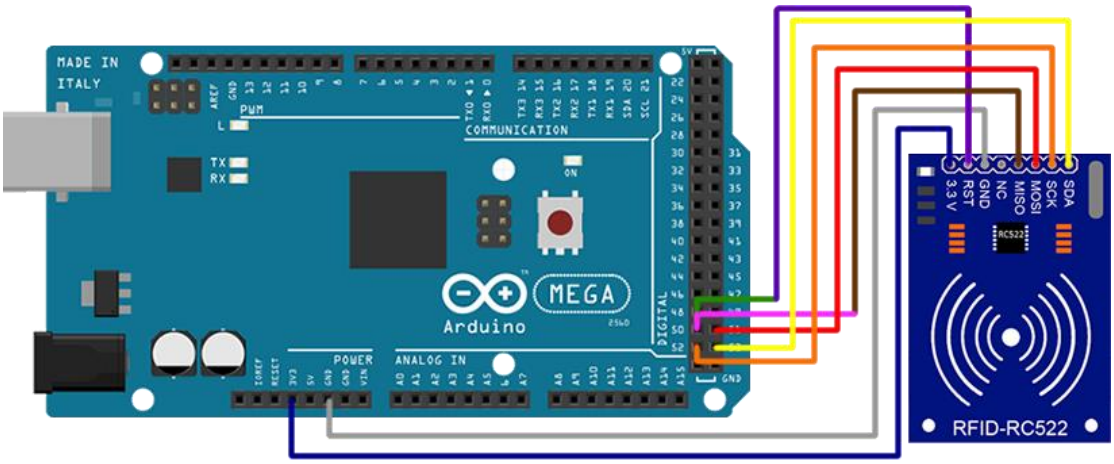


Figura 1- Esquemático da conexão MFRC522 e o Arduino Mega 2560 R3
Fonte: Elaboração do autor

A disposição de seus pinos em relação as portas do Arduino Mega 2560 R3, mostrado na Figura 1, é demonstrado na Tabela 1:

Tabela 1 - Conexões entre o Módulo RFID MFRC522 e o Arduino Mega 2560 R3
Fonte: Elaboração do autor

RFID MFRC522 – Pino	**Arduino Mega 2560 R3 – Porta**
SDA	Porta digital 53
SCK	Porta digital 52
MOSI	Porta digital 51
MISO	Porta digital 50
NC	Não conectado
GND (Ground) – terra	GND (*Protoboard*)
RST	Porta digital 48
3,3V (VCC) – tensão	3,3V (*Protoboard*)

Em conjunto ao módulo RFID MFRC522 será utilizado um módulo *Display* LCD 16x2 HD44780 que exibirá mensagens ao usuário pertinentes a liberação ou não liberação. A cada checagem de uma tentativa de acesso via RFID é exibida uma mensagem de retorno no *display* LCD informando se o acesso foi liberado ou não, além de uma mensagem inicial que solicita a passagem do cartão RFID pelo leitor. Ao *display* LCD também será acoplado um potenciômetro de 10k que será responsável por regular o brilho da tela LCD.

3.3 CONFIGURAÇÃO DO MÓDULO DISPLAY LCD 16x2 HD44780

Para realizar a função de exibição das mensagens pertinentes ao retorno da checagem do módulo RFID MFRC522 junto ao utilizador do protótipo optou-se pela

utilização de um módulo *Display* LCD 16x2 HD44780, além de um potenciômetro de 10k que realizará o controle de brilho dos textos que serão exibidos no *display* do LCD conforme Figura 2, onde:

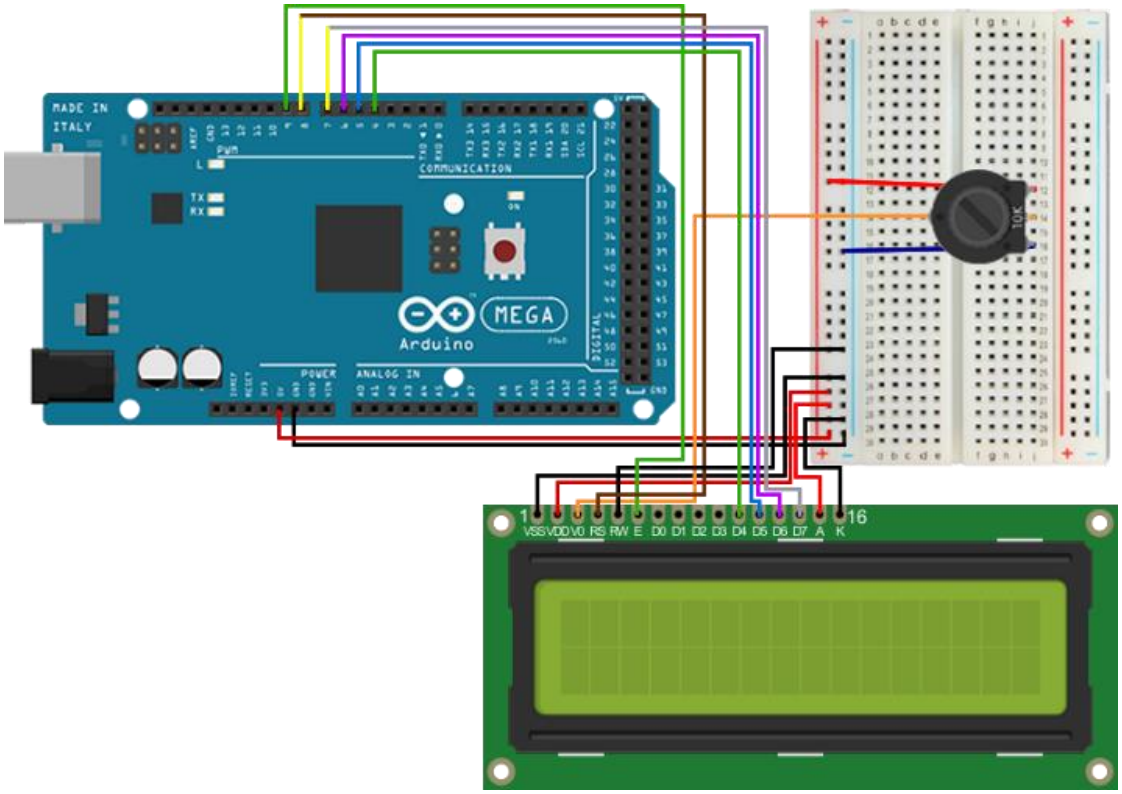


Figura 2 - Esquemático da conexão Display LCD HD44780 + Potenciômetro 10k e o Arduino Mega 2560 R3
Fonte: Elaboração do autor

O trecho de código de programação Arduino do Quadro 3 mostra a função que gera a mensagem inicial que é exibida no *display* LCD:

Quadro 3 – Função que exibe a Mensagem Inicial no Display

```
// Funcao que imprime a mensagem inicial no Display LCD
void mensagemInicial()
{
  lcd.clear();
  lcd.print( "   APROXIME O   " );
  lcd.setCursor(0, 1);   // Posiciona o cursor na posição 0 da segunda linha do Display LCD
  lcd.print( "CARTAO NO LEITOR" );
}
```

O Quadro 4 mostra a função LCD responsável por exibir o texto de permissão de acesso e logo abaixo no Quadro 5 a função LCD responsável por exibir o texto negando o acesso e solicitando ao utilizador contatar o administrador:

Quadro 4 – Função que exibe os textos no display LCD quando o acesso é permitido

```
// Funcao que imprime os textos de permissao no Display LCD
void lcdMsgPermitido()
{
  lcd.clear();  // Limpa a tela do Display LCD
  lcd.setCursor(0, 0);  // Posiciona o cursor na posição 0 da primeira linha do Display LCD
  lcd.print("  OLA USUARIO   ");
  lcd.setCursor(0, 1);  // Posiciona o cursor na posição 0 da segunda linha do Display LCD
  lcd.print( "ACESSO PERMITIDO" );
  delay(3000);  // Espera de 3 seg
  lcd.clear();  // Limpa a tela do Display LCD
  mensagemInicial(); // Exibe a mensagem inicial no Display LCD
}
```

Quadro 5 - Função que exibe os textos no display LCD quando o acesso é negado

```
// Funcao que imprime os textos de negado no Display LCD
void lcdMsgNegado ()
{
  lcd.clear();  // Limpa a tela do Display LCD
  lcd.setCursor(0, 0);  // Posiciona o cursor na posição 0 da primeira linha do Display LCD
  lcd.print( "  CARTAO  " );
  lcd.setCursor(0, 1);  // Posiciona o cursor na posição 0 da segunda linha do Display LCD
  lcd.print( "  DESCONHECIDO  " );
  delay(2000);  // Espera de 3 seg
  lcd.clear();  // Limpa a tela do Display LCD
  lcd.setCursor(0, 0);  // Posiciona o cursor na posição 0 da primeira linha do Display LCD
  lcd.print( " ACESSO NEGADO  " );
  lcd.setCursor(0, 1);  // Posiciona o cursor na posição 0 da segunda linha do Display LCD
  lcd.print( "CONTATE O ADMIN." );
  delay(3000);  // Espera de 3 seg
  lcd.clear();
  mensagemInicial(); // Exibe a mensagem inicial no Display LCD
}
```

O acoplamento do módulo Display LCD HD44780 a placa Arduino Mega 2560 R3 de acordo com a documentação oficial é realizado através da conexão dos 16 pinos (VSS, VDD, VO, RS, R/W, E, DB0, DB1, DB2, DB3, DB4, DB5, DB6, DB7, LED+ e LED-) onde:

- Pino 01 – (VSS), pino de alimentação de tensão neutro/terra (*Ground*);
- Pino 02 – (VDD), pino de alimentação de tensão positivo (5V);
- Pino 03 – (VO), pino de ajuste de contraste (via potenciômetro);
- Pino 04 – (RS - *Reset*), pino de reinicialização do LCD;
- Pino 05 – (R/W – *Read / Write*), pino de leitura e escrita;

- Pino 06 – (E - *Enable*), pino habilitação do LCD (1=*desable* / 2=*enable*);
- Pinos 07 à 14 – (DB0 – DB7), pinos de dados, lembrando que os pinos (DB0 à DB3), por padrão, não são utilizados;
- Pino 15 – (LED+), pino positivo (anodo) da luz de fundo (*backlight*) do *display*;
- (LED-), pino neutro/terra (catodo) da luz de fundo (*backlight*) do *display*.

A disposição de seus pinos em relação as portas do Arduino Mega 2560 R3, mostrado na Figura 2, é detalhado na Tabela 2:

Tabela 2 - Conexões entre o Módulo LCD HD44780 e o Arduino Mega 2560 R3
Fonte: Elaboração do autor

Display LCD HD44780 – Pino	**Arduino Mega 2560 R3 – Porta**
VSS	GND (Protoboard)
VDD	5V (Protoboard)
VO	Pino Central (Potenciômetro 10k)
RS	Porta digital 08
R/W	GND (Protoboard)
E	Porta digital 09
DB0	Não conectado
DB1	Não conectado
DB2	Não conectado
DB3	Não conectado
DB4	Porta digital 04
DB5	Porta digital 05
DB6	Porta digital 06
DB7	Porta digital 07
LED+	5V (Protoboard)
LED-	GND (Protoboard)
Potenciômetro 10k – Pino	**Arduino Mega 2560 R3 – Porta**
Central (Analógico)	Pino 03 (Display LCD)
Positivo (+)	5V (Protoboard)
Neutro (-)	GND (Protoboard)

3.4 CONFIGURAÇÃO DOS MÓDULOS LEITOR DE CARTÃO MICRO SD E RTC – REAL TIME CLOCK DS3231

Para realizar a função de armazenamento das informações pertinentes ao registro de (*Log*) de acesso optou-se pela utilização de um módulo Leitor de cartão microSD, enquanto que, para prover o horário no formato (dd/mm/aaaa hh:mm:ss), onde “dd” será

a representação numérica do dia com 2 dígitos, “mm” será a representação numérica do mês com 2 dígitos, “aaaa” será a representação numérica do ano com 4 dígitos, “hh” será a representação numérica da hora com 2 dígitos, “mm” será a representação numérica dos minutos com 2 dígitos e “ss” será a representação numérica dos segundos com 2 dígitos, e que, alimentará a programação Arduino no momento do registro da tentativa de acesso via RFID MFRC522, optou-se pela utilização de um módulo de relógio em Tempo real RTC DS3231 conforme Figura 3:

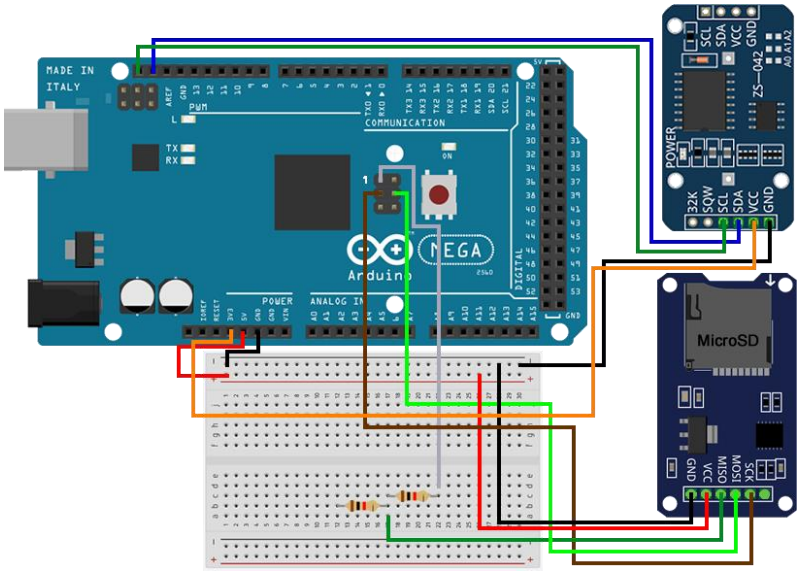


Figura 3 - Esquemático da conexão Leitor de cartão microSD + RTC DS3231 e o Arduino Mega 2560 R3
Fonte: Elaboração do autor

Para a programação Arduino das funcionalidades do Leitor de cartão microSD foram incluídas as funções para checagem e inicialização (abertura do arquivo log.txt) do cartão microSD, formatação da data do acesso e gravação dos dados de registro do acesso ao arquivo de *Log* conforme Quadro 6:

Quadro 6 - Funções responsáveis por gravar os registros de acesso no arquivo de Log do cartão microSD

```
// Funcao que inicializa o Cartao MicroSD
void inicializarCartaoSD()
{
  if (!SD.begin(SDCARD_SS_PIN))
 {
   Serial.println("Falha de inicializacao do CartaoSD!");
   return;
 }
 // Serial.println("Inicializacao do CartaoSD realizada com Sucesso!");

 /* Abra o arquivo. Note que apenas um arquivo pode ser aberto por vez,
  * então você precisa fechar esse arquivo antes de abrir outro.
```

```
  */
}

/*
* Funcao que escreve uma linha no formato:
* dd/mm/aaaa hh:mm:ss, <cod da Tag> P<numero>, Permitido ou Negado #
*/
void escreverCartaoSD()
{
  myFile = SD.open("log.txt", FILE_WRITE);
  if (myFile)  // Se o arquivo for aberto, escreva para ele:
  {
    DateTime now = rtc.now();  // Obtem a data e hora correntes do RTC e armazena em now

    // Imprime a data no formato ( dd/mm/aaaa )
    if (now.day() < 10) // Adiciona um 0 caso o valor seja menor que 10
    {
      myFile.print("0");
    }
    myFile.print(now.day(), DEC);
    myFile.print('/');
    if (now.month() < 10) // Adiciona um 0 caso o valor seja menor que 10
    {
      myFile.print("0");
    }
    myFile.print(now.month(), DEC);
    myFile.print('/');
    myFile.print(now.year(), DEC);
    myFile.print(' ');

    // Imprime a hora no formato ( hh:mm:ss )
    if (now.hour() < 10) // Adiciona um 0 caso o valor seja menor que 10
    {
      myFile.print("0");
    }
    myFile.print(now.hour(), DEC);
    myFile.print(':');
    if (now.minute() < 10) // Adiciona um 0 caso o valor seja menor que 10
    {
      myFile.print("0");
    }
    myFile.print(now.minute(), DEC);
    myFile.print(':');
    if (now.second() < 10) // Adiciona um 0 caso o valor seja menor que 10
    {
      myFile.print("0");
```

```
    }
    myFile.print(now.second(), DEC);
    myFile.print(", ");

    // Funcao que imprime o cod da Tag RFID + Porta Acessada. Ex:- ( 1A2B3C P01 )
    portaAcesso();
    myFile.print(IDtag + tagPorta);
    myFile.print(", ");

    escreverPermissao();  // Escreve o texto "Permitido" ou "Negado" conforme o resultado obtido

    myFile.println(" #");  // Imprime o caracter de checagem para quebra de linha no Android
    myFile.close();  // Fechando o arquivo:

    // Serial.println("arquivo atualizado com sucesso!");
  }
  else
  {
    Serial.println("ERRO ao abrir log.txt");  // Se o arquivo não foi aberto, imprima um erro:
  }
}
```

O acoplamento do módulo Leitor de cartão microSD a placa Arduino Mega 2560 R3 de acordo com a documentação oficial é realizado através da conexão dos pinos (GND, VCC, MISO, MOSI, SCK) onde:

- (GND - *Ground*), pino de alimentação de tensão neutro/terra;
- (VCC), pino de alimentação de tensão positiva (5V);
- (MISO – *Master In Slave Out*), é utilizado pelo escravo (*slave*) para enviar dados para o mestre (*master*);
- (MOSI – *Master Out Slave In*), é utilizado pelo mestre (*master*) para envio de dados para os periféricos;
- (SCK – *Serial Clock*), é utilizado como temporizador para sincronizar a transmissão de dados gerada pelo mestre (*master*).

Já o acoplamento do módulo de relógio em tempo real RTC DS3231 a placa Arduino Mega 2560 R3 de acordo com a documentação oficial é realizado através da conexão dos pinos (GND, VCC, SDA, SCL) onde:

- (GND - *Ground*), pino de alimentação de tensão neutro/terra;
- (VCC), pino de alimentação de tensão positiva (5V);
- (SDA – *Serial Data*), pino de dados;

- (SCL – *Serial Clock*), pino temporizador utilizado para sincronizar a transmissão de dados.

Já para a programação Arduino das funcionalidades do módulo RTC DS3231 foi implementada uma função que ajusta a data/hora do relógio em tempo real com a data/hora da compilação do programa em caso de perda das informações por falta de energia, conforme Quadro 7. Note que também é possivel através da função incluir a data/hora manualmente:

Quadro 7 - Função que ajusta data/hora do RTC DS3231

```
// Funcao que ajusta a Data/Hora caso o RTC fique sem energia (sem bateria)
void ajustarRTC()
{
  if (! rtc.begin())
  {
    Serial.println("Nao foi possivel encontrar o RTC");
    while (1);
  }
  if (rtc.lostPower())
  {
    Serial.println("Caso o RTC for desligado, redefine o horario!");

    // A linha a seguir define o RTC na data e hora em que esse Sketch foi compilado.
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    /*
     * A funcao acima ajusta o RTC com uma data e hora explícitas, por exemplo:
     * para definir a data --> Novembro 10, 2017 as 3 horas da manha seria feita a chamada:
     * rtc.adjust(DateTime(2017, 11, 10, 3, 0, 0));
    */
  }
}
```

A disposição de seus pinos em relação as portas do Arduino Mega 2560 R3, mostrado acima na Figura 3, é descrita de acordo com a Tabela 3:

Tabela 3 - Conexões entre os Módulos Leitor de cartão microSD e RTC DS3231 e o Arduino Mega 2560 R3
Fonte: Elaboração do autor

Leitor de cartão microSD – Pino	Arduino Mega 2560 R3 – Porta
GND	GND (Protoboard)
VCC	5V (Protoboard)
MISO	ICSP – Pino 01
MOSI	ICSP – Pino 04
SCK	ICSP – Pino 03

RTC DS3231 – Pino	Arduino Mega 2560 R3 – Porta
GND	GND (Protoboard)
VCC	3,3V (Protoboard)
SDA	SDA2
SCL	SCL2

O módulo RTC DS3231 é responsável por disponibilizar a data/hora quando solicitado, e para tanto deve ser previamente ajustado. O RTC DS3231 possui um encaixe (*slot)* para acoplamento de uma bateria tipo botão de Lithium modelo CR2032 3V conforme Figura 4 abaixo que manterá suas configurações de dia, mês, ano, hora, minuto e segundos mesmo que o Arduino seja desligado, persistindo estas configurações enquanto durar a bateria ou até que a mesma seja retirada do módulo.

Figura 4 - Módulo RTC DS3231 (Vista do socket + Bateria CR2032 3v comum)
Fonte: Elaboração do autor

Como visto na Tabela 3, o pino VCC (tensão) do RTC foi conectado a *Protoboard* (Matriz de contato) na porta de 3,3V para evitar uma sobrecarga e não de 5V. Isso foi necessário, pois, apesar do módulo trabalhar com tensões entre (3,3V-5V), no caso específico do módulo RTC do protótipo em questão irá operar com uma bateria CR2032 3V (comum) e para operar na voltagem 5V é recomendado a utilização de uma bateria CR2032 3,6V (recarregável) para que o circuito faça o recarregamento da bateria.

Para conexão do módulo Leitor de cartão microSD as portas utilizadas são: a (*MOSI - Master Output Slave Input*), (*MISO - Master Input Slave Output*), (*SCK - Serial Clock*) e (*SDA ou SS - Slave Select*) que são integrantes da (*SPI - Serial Peripheral Interface*), que é uma interface de comunicação serial (ARDUINO, 2017).

A biblioteca SPI é um protocolo de dados serial síncrono utilizado para realizar a comunicação entre a placa Arduino como mestre e outros periféricos escravos que utilizem esta comunicação (ARDUINO, 2017).

De acordo com Arduino (2017) o protocolo SPI utiliza algumas portas comuns para os periféricos, que normalmente são 3 (três) das quais:

- MOSI – é a porta *Slave* (escravo) para enviar dados para o mestre;
- MISO – é a porta *Master* (mestre) para enviar dados para os periféricos;
- SCK – é a porta responsável pelos pulsos de relógio que sincronizam a transmissão de dados gerada pelo mestre.

O protocolo SPI também utiliza uma porta específica para estes periféricos:

- SS – é a porta específica de cada periférico que o mestre pode utilizar para ativá-los e desativa-los individualmente.

A comunicação SPI é utilizada por alguns periféricos Arduino e no caso do projeto em questão, pelo módulo leitor de cartão microSD e o módulo RFID MFRC522 conforme a Figura 3 acima. Neste momento surgiu um problema, pois quando testados individualmente funcionaram normalmente, mas, quando conectados simultaneamente apresentaram mal funcionamento, onde, quando o módulo cartão microSD estava ativo, fazia com que o módulo RFID MFRC522 parasse de funcionar (coletar dados pelo sensor). Também houveram problemas na hora de conectar os pinos MOSI, MISO e SCK na placa Arduino Mega 2560 R3.

Após estudo da documentação oficial da placa Arduino Mega 2560 R3, foi obtido a informação de que as portas responsáveis por estas conexões são as portas 50, 51, 52 e 53, mas que no caso de conexões múltiplas poderiam ser utilizadas as portas contidas nos pinos centrais ICSP da placa Arduino Mega 2560 R3 conforme Figura 5:

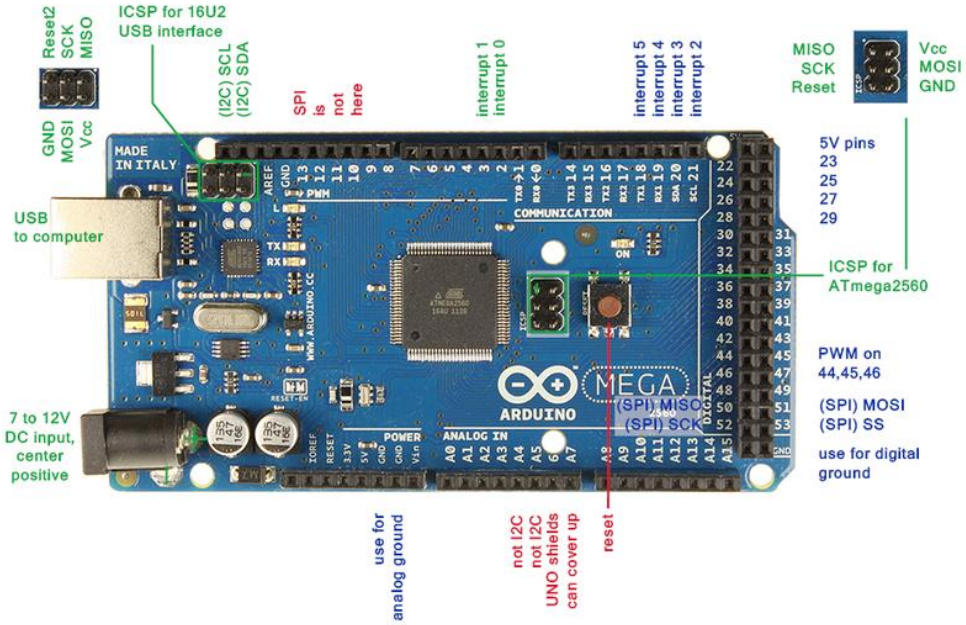


Figura 5 - Identificação das Portas/Pinos Arduino Mega 2560 R3
Fonte: https://www.claparts.com.br/product-page/arduino-atmega-mega-2560-r3-cabo-usb

Contudo, mesmo separando as conexões (portas) dos 2 (dois) componentes, ainda persistia o problema de não funcionarem simultaneamente.

Após vários testes foi descoberto que o problema se dava por conta do pino MISO que é responsável pelo envio de dados do Arduino Mega para os periféricos. O módulo RFID MFRC522 trabalha com uma alimentação de 3,3V enquanto que o módulo leitor de cartão microSD utiliza uma alimentação de 5V o que afeta na comunicação causando uma interferência de tensão quando os dois módulos estão ligados simultaneamente por conta de compartilharem a conexão do pino MISO mesmo que estando conectado fisicamente em pinos separados.

A solução encontrada para resolver o problema da interferência foi adicionar 2 resistores criando assim um divisor de tensão para reduzir a voltagem de operação que passa especificamente pelo pino de dados MISO do módulo leitor de cartão microSD, de 5V que é a tensão da porta digital trabalha, para 3,3V. Este procedimento anula a interferência causada pela diferença de tensão dos pinos MISO do módulo RFID MFRC522 e do módulo leitor de cartão microSD.

3.5 CONFIGURAÇÃO DA MINI TRAVA ELETROMAGNÉTICA SOLENOIDE 12V DO MÓDULO RELÉ 5V 2 CANAIS

Para automatizar o travamento e destravamento da porta que dará acesso ao ambiente restrito foi fixado ao batente desta porta uma mini trava eletromagnética Solenoide 12V ligada a um Relé 5V 2 canais que por sua vez é conectado a placa Arduino Mega 2560 R3. A mini trava eletromecânica Solenoide 12V quando acionada via programação Arduino realizará a liberação física da porta, que se dará ativando o Relé 5V 2 canais que acionará a Solenoide, que, ao receber energia irá retrair o êmbolo (pino da trava) para destrancar a porta e após 3 segundos será desligada via Relé 5V 2 canais liberando assim o êmbolo para que possa trancar a porta novamente.

No Quadro 8 é descrito a função da programação Arduino responsável por realizar o acionamento do relé e consequentemente da trava Solenoide:

Quadro 8 - Função que aciona a trava Solenoide através do acionamento do relé

```
// Funcao que realiza o acionamento da Trava Solenoide
void acionarSolenoide()
{
  digitalWrite(p_rele1, LOW);  //  Ativa o Rele 1 / Destrava a Solenoide
  delay(3000);
  digitalWrite(p_rele1, HIGH);  // Desativa o Rele 1 / Trava a Solenoide
}
```

Neste cenário foi encontrado uma dificuldade ao implementar o protótipo proposto. Quando a fechadura era trancada, surgia um erro no LCD, impedindo sua leitura pelo utilizador. Foi descoberto que isso ocorria quando a solenoide é desativada e o campo magnético é descarregado na forma de uma corrente reversa ao sentido de condução do circuito, causando uma interferência de tensão ao circuito principal.

Segundo Boylestad e Nashelsky (2004), o Diodo é o mais simples dos semicondutores, mas que exerce um papel muito importante em sistemas eletrônicos atuando como uma simples chave.

Um Diodo ideal tem como principal função conduzir corrente em um único sentido agindo como um circuito aberto e tem a finalidade de impedir qualquer tentativa de enviar corrente no sentido contrário (BOYLESTAD e Nashelsky, 2004).

Após vários testes e pesquisa, a solução implementada foi acrescentar um Diodo retificador modelo 1N4007 ao circuito da trava solenoide, impedindo o retorno de tensão para o circuito principal conforme demonstrado na Figura 6:

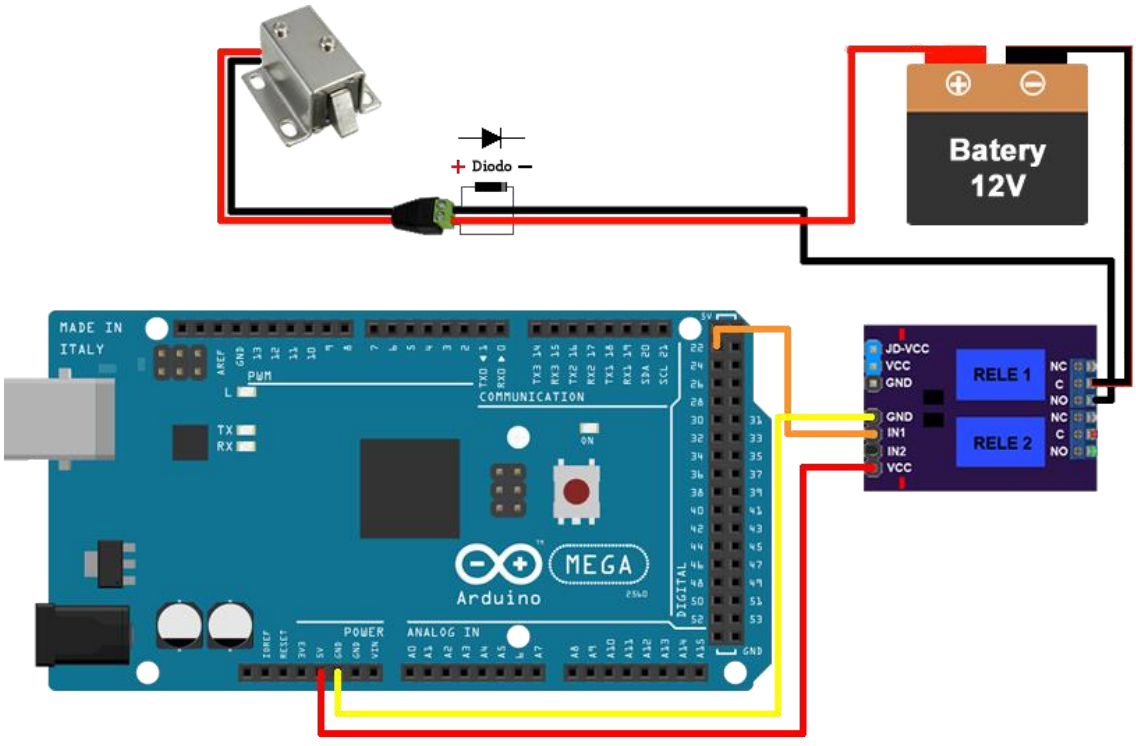


Figura 6 - Esquemático da conexão mini trava eletromagnética + Relé 5V 2 canis e o Arduino Mega 2560 R3
Fonte: Elaboração do autor

A disposição de seus pinos em relação as portas do Arduino Mega 2560 R3 conforme mostrado na Figura 6 está detalhado na Tabela 4:

Tabela 4 – Conexões entre Bateria + ministrava Solenoide 12V + Relé 5V + Arduino Mega 2560 R3
Fonte: Elaboração do autor

Relé 5V 2 canais – Conexão	**Arduino Mega 2560 R3 – Porta**
GND	GND (Protoboard)
IN1 (entrada de controle relé 1)	Porta digital 22
IN2 (entrada de controle relé 2)	Não conectado
VCC	5V
Relé 5V 2 canais – Conexão	**Bateria / Solenoide 12V – Conexão**
NC (Normal Fechado)	Não conectado
C (Comum)	Conector (-) = Bateria
NO (Normal Aberto)	Conector (-) = Solenoide
Mini trava eletromagnética Solenoide 12v	**Relé 5V 2 canais – Conexão**
Conector (+)	Conector (+) = Bateria
Conector (-)	Conector (+) = Relé 5V 2 canais

Foi incluído ao protótipo uma opção de destravamento pelo lado interno da porta com a inclusão de um *pushbutton* (Chave tatil) que é um botão que deve ser acionado manualmente caso a porta seja fechada com o utilizador dentro do ambiente restrito, visto que o painel de acesso RFID fica do lado de fora. Na programação Arduino foi incluída uma função para está ação que chama a função da trava Solenoide conforme descrito no Quadro 9:

Quadro 9 - Função que aciona a trava Solenoide manualmente via botão do lado interno da porta

```
// Funcao que libera a porta ativando a trava Solenoide manualmente pressionando o Push Button
void botaoAbreTranca()
{
  // Le o valor na porta digital e armazena na variavel
  botao = digitalRead(pushButton);
  if(botao == 0) // Se o valor lido for igua a 0:
  {
    acionarSolenoide();
  }
}
```

3.6 CONFIGURAÇÃO DO MÓDULO BLUETOOTH HC-06

Como componente Arduino de comunicação externa optou-se pela utilização do módulo Bluetooth HC-06 que assim como o módulo Bluetooth HC-05 utilizado no trabalho de Silva (2016) deve ser habilitado seu modo Administrador para que se possa alterar as configurações de fábrica padrão renomeando o dispositivo, alterando a sua senha e também obtendo o seu MAC *Address* (*Media Access Control*) que é endereço

físico único ligado a interface de comunicação que serão exigidos no pareamento de uma conexão com um dispositivo móvel. A diferença entre o módulo Bluetooth HC-05 utilizado no trabalho de Silva (2016) e o HC-06 que foi utilizado neste projeto está no fato do modo Bluetooth HC-05 permitir o modo Mestre/Escravo (*Master/Slave)* e o modelo HC-06 permitir apenas o modo escravo (*slave*), mas, que não afetará em nada no funcionamento do protótipo, uma vez que apenas receberá conexões.

No bloco de "*loop*" a programação Arduino foi incluída uma função para chamar a captura de dados pelo aplicativo Android quando solicitado. Esta função fica aguardando o envio do aplicativo Android do caracter "0", e após recebê-lo, retorna para o aplicativo em modo serial, os dados do arquivo log.txt conforme Quadro 10:

Quadro 10 - Função que envia os dados do arquivo log.txt para o APP Android

```
// Funcao que chama a captura de dados via "Serial" do App Android quando conectado
void capturaAndroid()
{
  while(Serial.available() > 0)
  {
    char opcao = Serial.read();
    if(opcao!='\n')
    {
      switch(opcao)
      {
        case '0':
          myFile = SD.open("log.txt");  // Reabrir o arquivo para leitura
          if (myFile)
          {
            while (myFile.available())  // Leia todo o conteudo do arquivo
            {
              Serial.write(myFile.read());
            }
            myFile.close();  // Fecha o arquivo
          }
          else
          {
            Serial.println("ERRO ao reabrir log.txt");  // Se o arquivo não foi aberto, imprima um erro
          }
          break;
        default:
          Serial.print("Sem leitura");
          break;
      }
    }
  }
}
```

O Bluetooth HC-06 conta com os pinos VCC (5V), GND (neutro) e (TX e RX) para troca de dados, assim como mostrado na Figura 7:

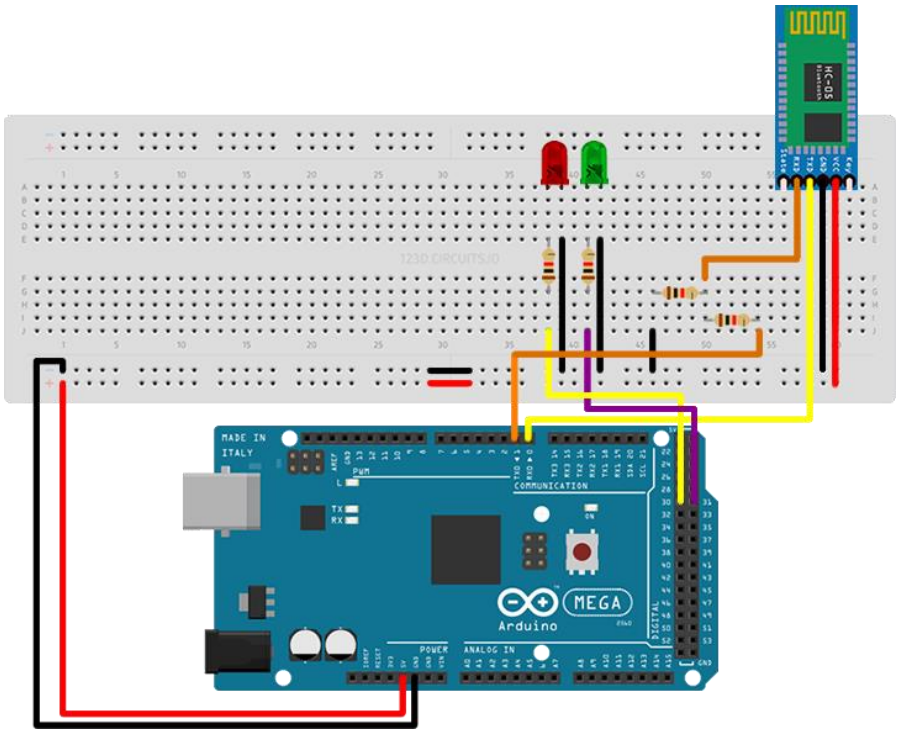


Figura 7 - Esquemático da conexão do módulo Bluetooth HC-06 e o Arduino Mega 2560 R3
Fonte: Elaboração do autor

A disposição de seus pinos em relação as portas do Arduino Mega 2560 R3 conforme mostrado na Figura 7 está representada na Tabela 5:

Tabela 5 - Conexões entre o módulo Bluetooth HC-06 e a placa Arduino Mega 2560 R3
Fonte: Elaboração do autor

Bluetooth HC-06 - Pino	**Arduino Mega 2560 R3 – Porta**
VCC	5V (Protoboard)
GND	GND (Protoboard)
RX	TX0 – Porta analógica 1
TX	RX0 – Porta analógica 0

Note que conforme visto na Figura 7 o pino RX do módulo Bluetooth HC-06 não é ligado diretamente a porta analógica TX do Arduino Mega 2560 R3 que opera com uma tensão de 5V. O pino RX assim como o pino TX são pinos de dados e operam com uma tensão de 3,3v, portanto, é necessário que o pino RX seja direcionado para a *Protoboard* passa por um divisor de tensão, neste caso, 2 resistores de 330Ω (*ohms*) antes de ser conectado à porta TX0 da placa Arduino Mega 2560 R3. O procedimento do divisor de tensão utilizado para solucionar o problema de comunicação do módulo bluetooth

também foi tratado na “seção 4”, “subseção 4.4” deste projeto e aplicado a conexão do pino MISO do módulo leitor de cartão microSD.

Também foram adicionados ao protótipo 2 LEDs (1 verde e 1 vermelho) que terão a finalidade de acender de acordo com o retorno da checagem da tag pelo RFID que ocorrerá uma vez para o LED vermelho acompanhado de um (1) *Bip* do *Buzzer* (emissor de sons Arduino) que por sua vez também será acoplado ao protótipo, no caso da negação do acesso e duas vezes para o LED verde acompanhado de dois (2) *Bips* do *Buzzer* no caso do acesso ser permitido. O trecho de código do Quadro 11 demonstra na programação Arduino da função que ficará responsável pela ação quando negado e o trecho de código do Quadro 12 pela ação quando permitido:

Quadro 11: Funcão que acende o LED vermelho 1 vez e emite um Bip via Buzzer

```
// Funcao que emite 1 Bip negando Permissao via Buzzer e
// acende o Led Vermelho 1 vez.
void efeitoNegado()
{
  int qtd_bips = 1;  // Define a quantidade de Bips.
  for(int j=0; j<qtd_bips; j++)
  {
    tone(BUZZER,500);  // Liga o Buzzer com uma frequencia de (500hz) e liga o LED vermelho.
    digitalWrite(LED_VERMELHO, HIGH);
    delay(500);

    noTone(BUZZER);  // Desliga o Buzzer e o LED vermelho.
    digitalWrite(LED_VERMELHO, LOW);
    delay(500);
  }
}
```

Quadro 12: Funcão que acende o LED vere 2 vez e emite dois Bips via Buzzer

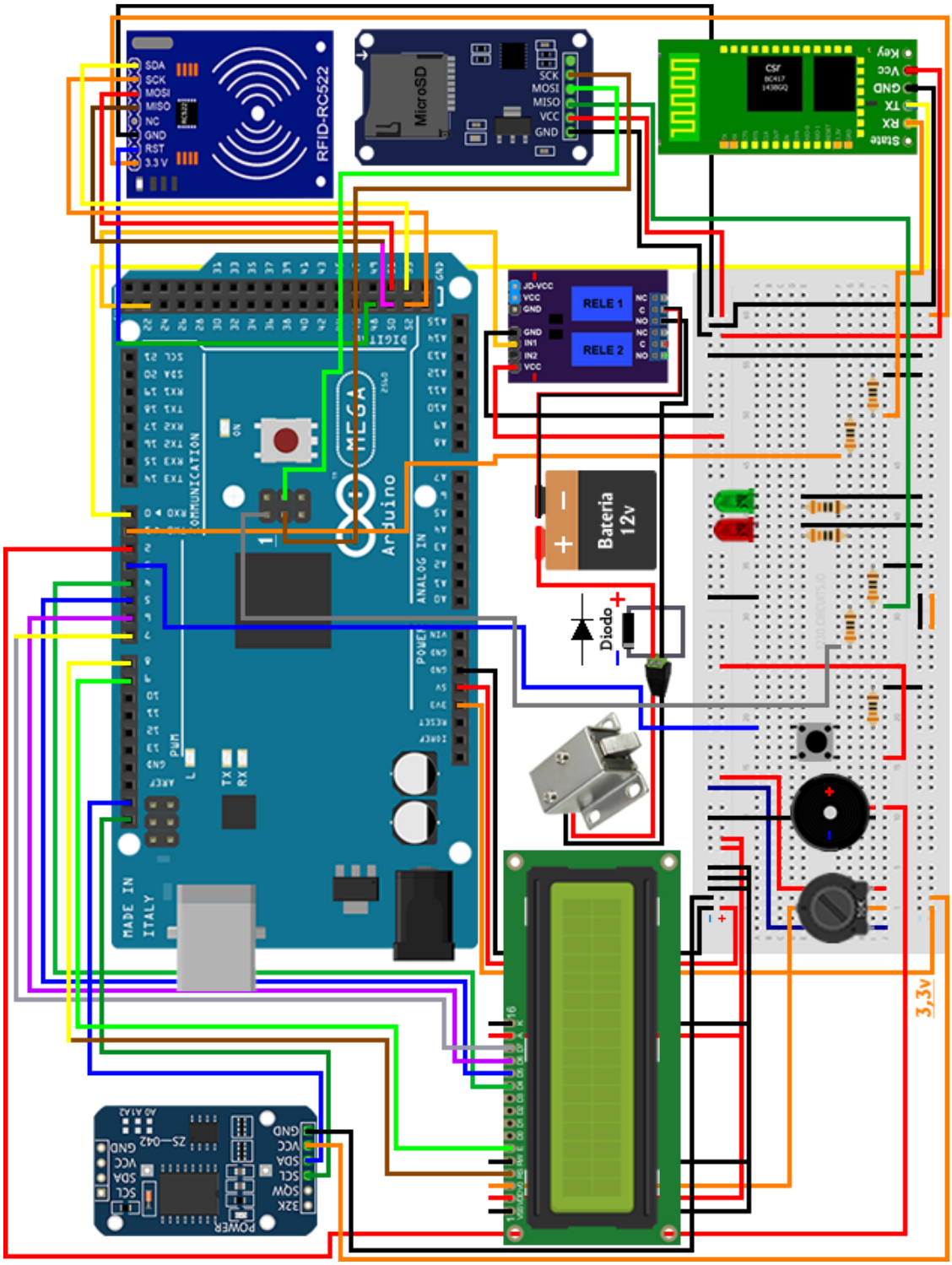
```
// Funcao que emite 2 Bips dando Permissao via Buzzer e acende o LED Verde (2 vezes).
void efeitoPermitido()
{
  int qtd_bips = 2;  // Define a quantidade de Bips.
  for(int j=0; j<qtd_bips; j++)
  {
    tone(BUZZER,1500);  // Liga o Buzzer com uma frequencia de (1500hz) e liga o LED verde.
    digitalWrite(LED_VERDE, HIGH);
    delay(100);

    // Desliga o Buzzer e LED verde.
    noTone(BUZZER);
    digitalWrite(LED_VERDE, LOW);
    delay(100);
  }
  acionarSolenoide(); // Chama a funcao que aciona a Solenoide.
}
```

Note que no Quadro 12 a função “efeitoPermitido()” além de realizar as ações de seus parâmetros no caso de “permitido” chama a função “acionarSolenoide()” que libera fisicamente a porta.

Ao final dos testes individuais dos componentes Arduino foi realizado a montagem e os testes do protótipo final de acordo com o esquema da Figura 8 :

Figura 8 - Esquemático Final das conexões dos módulos Arduino e o Arduino Mega 2560 R3
Fonte: Elaboração do autor



O protótipo final pode ser visto a seguir de acordo com a Figura 9 (imagem frontal) , Figura 10 (imagem traseira), Figura 11 (imagem do Painel de acesso) e Figura 12 (imagem das conexões Arduino) :

Figura 9: Imagem frontal do Protótipo Arduino
Fonte: Elaboração do autor

Figura 10: Imagen traseira do Protótipo Arduino
Fonte: Elaboração do autor

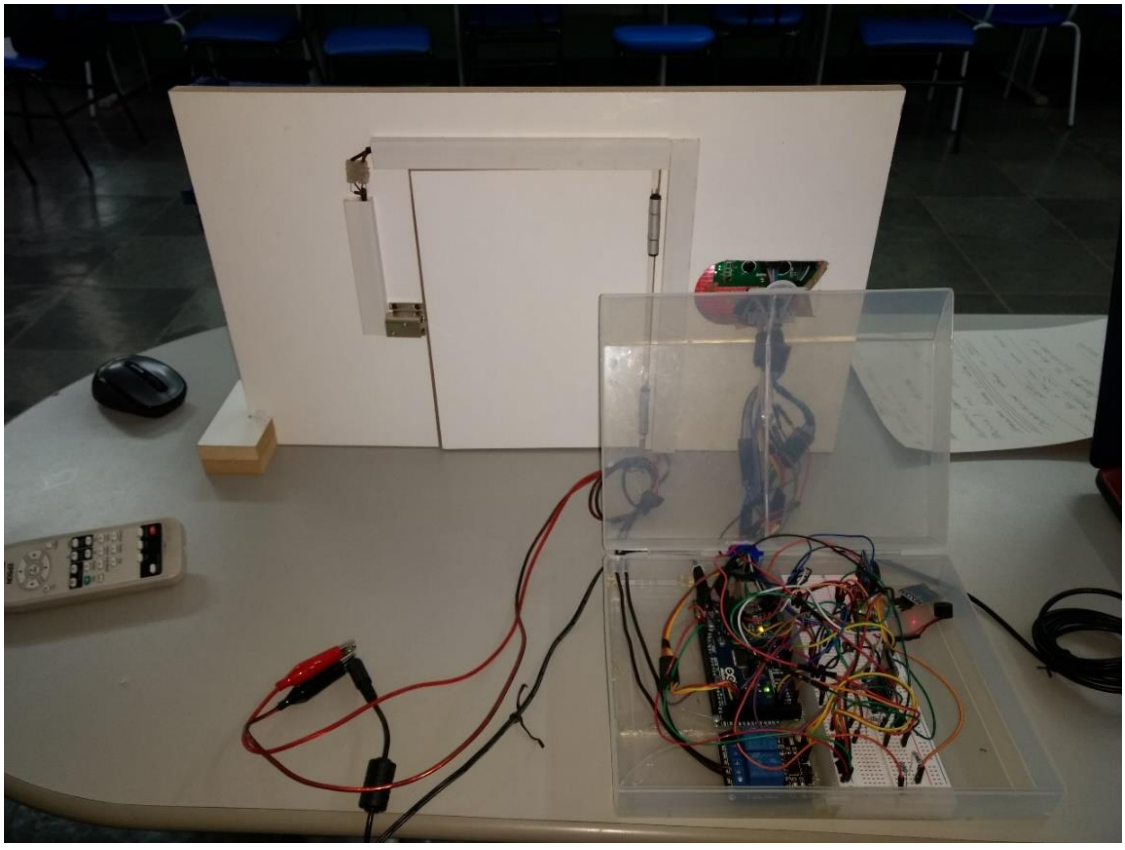
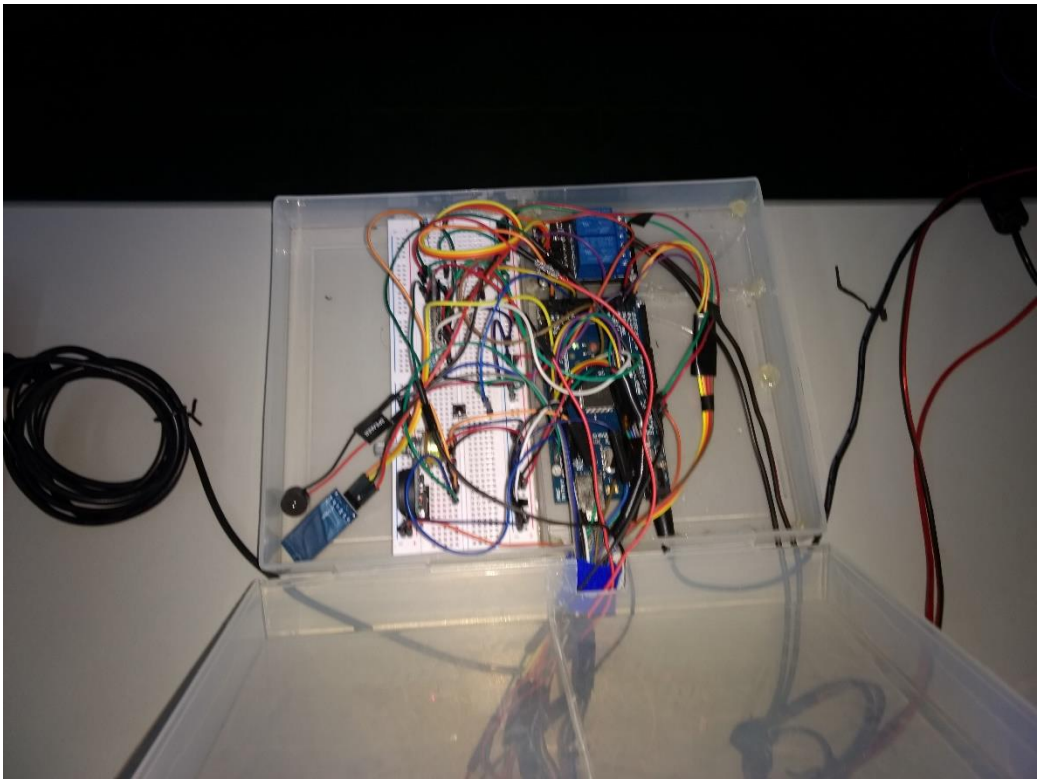


Figura 11: Imagem do Painel de acesso do Protótipo Arduino
Fonte: Elaboração do autor

Figura 12: Imagem das conexões do Protótipo Arduino
Fonte: Elaboração do autor



Durantes os testes algumas dificuldades foram encontradas. A solução aplicada neste projeto poderá vir a ser de grande contribuição para trabalhos futuros que se utilizem destes recursos. São elas:

- Um problema referente aos pinos comuns (MISO, MOSI, SCK e SDA) do módulo RFID MFRC522 descrita na subseção 3.2 e do módulo Leitor de cartão microSD na subseção 3.4 que utilizam as mesmas portas. A solução encontrada foi manobrar os pinos do módulo Leitor de cartão microSD para os pinos ICSP que é um conjunto de 6 pinos utilizados para as portas do protocolo SPI e que servem de opção quando mais de um periférico é conectado a placa Arduino Mega 2560 R3 e as portas 50, 51, 52 e 53 já estão sendo utilizadas. Também foi solucionado um problema na comunicação dos periféricos quando conectados simultaneamente ao Arduino Mega 2560 R3, onde, foi colocado como divisor de tensão 2 resistores de 330Ω (ohms) intermediando a conexão do pino MISO do módulo Leitor de cartão microSD e o Arduino Mega 2560 R3 para reduzir sua tensão de operação para até 3,3v;
- Um problema referente ao pino de tensão do módulo RTC DS3231 tratado na subseção 3.4 que foi solucionado conectando-o na Protoboard a faixa de conexões de tensão de 3,3v por se tratar da utilização de uma bateria de Lithium CR2032 de 3V (comum) e não uma bateria CR2032 de 3,6V (recarregável) que poderia teoricamente causar uma sobrecarga de tensão;
- Um problema tratado na subseção 3.5 relacionado a uma interferência na tela do módulo Display LCD 16x2 HD44780 causada pelo acionamento da mini trava eletromagnética Solenoide 12V que, por ser indutiva, após receber corrente elétrica libera o embolo da trava e após seu desligamento e travamento da trava retorna uma corrente reversa para o circuito principal, causando assim a referida interferência que causa uma mistura nos caracteres do Display LCD impossibilitando a sua leitura pelo utilizador. A solução encontra foi ligar um Diodo modelo 1N4007 ao conector da trava Solenoide para conter o retorno de corrente para o circuito principal neutralizando a interferência que afeta o *Display* LCD.

3.7 DESENVOLVIMENTO DA APLICAÇÃO ANDROID PARA COLETA DE DADOS

Para o software que consumirá os dados armazenados no cartão de memória do protótipo do controle de acesso Arduino optou-se pela criação de uma aplicação simples Android para dispositivo móvel que por intermédio do utilizador do dispositivo quando necessário, se conectará remotamente via conexão Bluetooth com o Arduino e realizará a coleta dos dados contidos no cartão microSD da lista de tentativas de acesso gravadas no arquivo (log.txt).

De posse destas informações a aplicação irá tratá-las e quando solicitado pelo utilizador do aplicativo exibi-las em uma lista.

Nesta seção serão apresentados os trechos de código de maior relevância para descrever as funcionalidades do aplicativo.

Inicialmente na criação do Aplicativo a primeira ação realizada foi a alteração do arquivo de manifesto da aplicação "*AndroidManifest.xml*" que é o arquivo que descreve as informações essenciais sobre o App ao Android declarando as permissões Bluetooth para que seja possível a comunicação e troca de dados com o dispositivo ao qual está instalado a aplicação e iniciar a descoberta de dispositivos conforme mostrado no Quadro 1313 abaixo:

Quadro 13 - Declaração das permissões Bluetooth no arquivo AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ifsp.tcc.jocimar.portaautomatica">
    <uses-permission android:name="android.permission.BLUETOOTH"></uses-permission>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"></uses-permission>
```

Para que o Bluetooth possa ser pareado com o Bluetooth HC-06 do Arduino é necessário que seja incluído no código do arquivo "*MainActivity.java*" da aplicação o MAC Address do módulo Bluetooth HC-06 para que seja possível a comunicação entre as interfaces conforme mostrado no trecho de código do Quadro 14:

Quadro 14 - Identificação do MAC Address do Bluetooth do Arduino

```
private static final String endereco_MAC_do_bluetooth_remoto = "98:D3:31:80:9A:55";
private static final UUID MEU_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
```

Seguindo com o desdobramento dos códigos da aplicação no arquivo "*Log.java*", o trecho principal da classe "*Main"* o aplicativo envia o caractere " 0 " para o Arduino via comunicação do Bluetooth e recebe todos os dados salvos no cartão de memória do módulo leitor de cartão microSD do Arduino utilizando-se para isso da classe

"OrganizaLog" para dividir a sequência de caracteres recebida em linhas, e depois, nos campos que são os atributos dos objetos do tipo "*LOG*" conforme descrito no Quadro 15:

Quadro 15 – Criação do Objeto com os Logs do Cartão de Memória do Arduino

```
@Override
   public void onClick(View view) {
      if (bluetoothSocket != null) {
         txtLog.setText("");
         //string para concatenar caracter sem usar +
         StringBuilder mensagem = new StringBuilder();

         try {
            outputStream = bluetoothSocket.getOutputStream();
            //o caractere 0 é o codigo para o arduino responder com o log completo
            sendData("0");
            SystemClock.sleep(2000);

            inputStream = bluetoothSocket.getInputStream();

            byte[] msgBuffer = new byte[1024];
            inputStream.read(msgBuffer);

            //A classe organiza vai fazer a divisão das strings em cada linha pelo caracter '#'
            //e a divisão de cada campo do log pelo caracter ',' e salvar dentro do vetor na classe SalvaLog
            OrganizaLog.trasformaStringEmLogs(new String(msgBuffer));

            //coloca na tela a string do log
            mensagem.append(SalvaLog.listarLogString());
            //so exibe no textView paia,pode tirar isso na versão final
            txtLog.setText(mensagem.toString());
            obterLog.setEnabled(false);

         } catch (IOException e) {
            Log.e("ERROR", "Erro:" + e.getMessage());
            Toast.makeText(getApplicationContext(), "Mensagem não recebida", Toast.LENGTH_LONG).show();
         }
```

No arquivo "OrganizaLog.java" o trecho de maior relevância é onde a codificação quebra a sequência de caracteres recebida em linhas onde houver o caractere "#" e em campos onde houver o caractere ",". É utilizado a classe "SalvarLog" para adicionar os objetos criados ao "List<>". Este trecho de código é demonstrado no Quadro 16:

Quadro 16 - Desdobramento da String do Objeto em linhas de marcações (Log)

```
public static void trasformaStringEmLogs(String logs) {
   linhas = vetorTorList(logs.split("#"));
   for (String linha : linhas) {
      String logEmVetorDeString[] = linha.split(",");
      //if para ignorar possivel sujeira que venha na ultima linha com caracteres ???
      if (linha.length() < 42) {
         Log logParaSalvar = new Log();
         logParaSalvar.setDataHora(logEmVetorDeString[0]);
         logParaSalvar.setCodigoPessoa(logEmVetorDeString[1]);
         logParaSalvar.setAprovado(logEmVetorDeString[2]);
         //Salvado o log dentro da lista
         SalvaLog.addLog(logParaSalvar);
      }
```

No arquivo "SarvarLog.java" é apresentado o trecho que cria o "*List*<>" que armazena os *Logs* recebidos e logo abaixo o método estático que adiciona os *Logs* ao "*List*<>" mencionado acima conforme mostrado no Quadro 17:

Quadro 17 - Criação do "List<>" referente as tentativas de acesso via RFID

```
//armazena os logs
private static List<Log> logsCapturados = new ArrayList<>();

public static void addLog(Log log){
   if(logsCapturados!=null){
      logsCapturados.add(log);
   }
```

Por fim é invocado uma segunda tela (*Activity*) que exibe este "*List*". Na Figura 13 é mostrado a segunda tela "Activity" com a lista de *Logs*:

Figura 13 - Tela do Aplicativo Android com a lista de Acessos (Permitido) e (Negado)
Fonte: Elaboração do autor

4 ANÁLISE DE CUSTO DO PROJETO

Espera-se com o desenvolvimento do projeto criar um Sistema automatizado que garanta o controle de acesso a ambientes restritos, visando uma maior segurança.

Para tanto, foi escolhido uma solução computacional com o desenvolvimento na plataforma Android e Arduino, visando um baixo custo de produção e também por tratar-se de tecnologias estudadas durante o decorrer do curso de Pós-Graduação.

Para embasar este ponto no quesito baixo custo e também levando em consideração a relação (Custo x Benefício) foi realizado uma pesquisa de preços dos componentes que foram utilizados em 3 (três) empresas do ramo como mostrado na Tabela 6:

Tabela 6 - Levantamento de Custo (Comparativo de Preços)
Fonte: Elaboração do Autor

Comparativo de Preços				
Componente	**Quant**	**FilipeFlop**	**Usina Info**	**Eletrogate**
Arduino Mega 2560 R3	1	R$ 74,90	R$ 79,90	R$ 78,90
Bluetooth RS232 - HC-05	1	R$ 36,90	R$ 34,90	R$ 29,90
Display LCD - 16 x 2	1	R$ 16,90	R$ 17,90	R$ 18,90
Potenciômetro 10k	1	R$ 2,40	R$ 2,50	R$ 1,80
LED (kit c/ 5 unid)	Kit	R$ 0,95	R$ 1,65	R$ 0,50
Protoboard 830 pontos	1	R$ 16,90	R$ 16,90	R$ 14,90
Relé 2 Canais	1	R$ 12,90	R$ 14,90	R$ 10,90
Mini Trava eletromagnética Solenoide 12V	1	R$ 29,90	R$ 69,90	R$ 39,90 *
Módulo RFID	1	R$ 39,90	R$ 35,90	R$ 26,90
Módulo RTC – DS3231	1	R$ 19,90	R$ 19,90	R$ 12,90
Módulo Cartão MicroSD	1	R$ 9,90	R$ 11,90	R$ 9,90
Jumper (kit c/ 65 unid)	Kit	R$ 16,90	R$ 19,90	R$ 9,90
Resistor (kit c/ 20 unid)	Kit	R$ 1,90	R$ 1,70	R$ 2,00
Subtotal		**R$ 280,25**	**R$ 327,85**	**R$ 257,30**
* **Preço médio do Mercado Livre.**				
Componente	**Quant**	**Walmart**	**Submarino**	**ShopSeg**
Bateria 12V 7A	1	R$ 58,90	R$ 65,71	R$ 61,11
Total		**R$ 339,15**	**R$ 393,56**	**R$ 318,41**

O valor médio da fechadura que será produzida neste trabalho é de R$ 350,00. Algumas fechaduras vendidas no mercado que possuem características similares ao do protótipo tem um custo aproximado de R$ 707,50 conforme a Tabela 7:

Tabela 7 - Comparativo de Preços de Fechadura eletrônica RFID industrializadas vendidas comercialmente

Comparativo de Preços					
Descrição	**Quant.**	**Magazine Luiza**	**Americanas**	**Mercado Livre**	**Preço Médio**
Fechadura Samsung Shs-1321	01	R$ 1.009,30	R$ 699,00	R$ 840,00	R$ 850,00
Fechadura Eletrônica Digital Intelbras FR 200	01	R$ 655,30	R$499,00	R$ 539,90	R$ 565,00
Média Total					**R$ 707,50**

Nota-se uma redução de 50,5 % de custo em relação a média de preço das fechaduras pesquisadas. e levando em consideração que a produzida neste trabalho ainda é um protótipo, possibilitando assim a personalização com a inclusão e de novos recursos por parte do utilizador, observa-se que a fechadura produzida neste projeto se torna viável.

5 CONCLUSÃO

É possível concluir com a construção deste experimento que é possível a confecção de uma solução Arduino / Android que seja capaz de gerenciar acessos a ambientes restritos com um baixo custo. Foi possível demonstrar o funcionamento de vários componentes Arduino atuando em conjunto para gerar um resultado satisfatório quanto a proposta inicial, e que nos revelou um alto grau de complexidade quando da utilização de vários componentes ligados ao mesmo tempo a placa Arduino, onde se fez necessário a utilização de uma placa Arduino modelo Mega, a qual possui uma maior capacidade de conexões simultâneas e que mesmo assim demandou bastante tempo e estudo dos componentes e da própria placa Arduino Mega 2560 R3 para que fosse possível a realização deste projeto. O documento final e os códigos fonte do sistema Arduino e Aplicação Android encontra-se disponível no endereço:

- https://github.com/JocimarSVN/Trava_Solenoide_Arduino

Como projeto futuro pode-se incluir a este projeto um gerenciamento *Web* via Painel Administrativo do site, substituindo o atual controle via hardware Arduino. Para tanto será necessário a substituição do módulo leitor de cartão microSD por um *Ethernet Shield*, módulo que já possui um acoplamento *onboard* para o cartão microSD, além de sua conexão de rede que será conectada à internet, por onde serão manipuladas todas as informações registradas no cartão microSD, bem como o cadastramento e exclusão de novas *tags* e usuários.

REFERÊNCIAS

AHSON, Syed A.; ILYAS, Mohammad. ***RFID handbook:*** *applications, technology, security, and privacy. CRC Press – Taylor & Francis Group,* Prefácio, p. 9, 2008.

ARDUINO CC: Site Oficial do Arduino. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 11 abril 2017.

ARDUINO CC: Site Oficial do Arduino. Disponível em: <https://www.arduino.cc/en/Reference/SPI>. Acesso em: 19 julho 2017.

BOYLESTAD, Robert L.; NASHELSKY, Louis. **Dispositivos Eletrônicos**: e teoria de circuitos. 8. ed. São Paulo: Pearson Education do Brasil, 2004.

GARTNER (Stamford***). Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016***. Disponível em: < http://www.gartner.com/newsroom/id/3609817>. Acesso em: 20 abril 2017.

KAMOGAWA, Marcos Y.; MIRANDA, Jeová Correia. Uso de hardware de código fonte aberto "Arduino" para acionamento de dispositivo solenoide em sistemas de análises em fluxo. **Quím. Nova**, São Paulo, v. 36, n. 8, p. 1232-1235, 2013. Disponível em <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-40422013000800023&lng=pt&nrm=iso>. Acesso em 11 mai. 2017. http://dx.doi.org/10.1590/S0100-40422013000800023.

LEAL, Nelson Glauber V. **Dominando o Android**: do básico ao avançado. 1. ed. São Paulo: Novatec Editora Ltda, 2015.

LIMA, D. S. **Proposta para controle de acesso físico seguro baseada em Near Field Communication (NFC) e Android**, 2013. Disponível em: http://www.tcc.sc.usp.br/tce/disponiveis/18/180450/tce-16012014-095759/?&lang=br>. Acesso em: 04 maio 2017.

MOREIRA, Katia Beatris Rovaron. **Diretrizes para projeto de segurança patrimonial em edificações.** Disponível em: <http://www.teses.usp.br/teses/disponiveis/16/16132/tde-17052010-090837/pt-br.php>. Acesso em: 08 novembro 2017.

MOTA, Rafael Perazzo Barbosa. ***RFID - Radio Frequency Identification***. 2012. Monografia (Pós-graduação em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2012.

SHARMA, Monika. A Research Survey: RFID Security & Privacy Issue. Data. Computer Science & Information Technology (CS & IT), Chittorgarh-Índia, v. 1, n. 1, p. 255-261. sep. 2013.

SILVA, Everton Rafael da. **CARduino:** A semi-automated vehicle prototype to stimulate cognitive development in a Learning-Teaching Methodology. In: 14th International Conference on Information Technology: New Generations (ITNG 2017), 2017, Las Vegas. 14th International Conference on Information Technology: New Generations (ITNG 2017), 2017.

VERDULT, Roel**. *Security analysis of RFID tags.*** Tese de Doutorado. *Master's thesis, Radboud University Nijmegen*, p. 5-7, 2008.

WANT, Roy. ***An introduction to RFID technology.*** *IEEE pervasive computing*, v. 5, n. 1, p. 25-33, 2006.