

**FACULDADE ANHANGUERA DE CAMPINAS**

**UNIDADE 1**

**TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**Manual do Projeto**

**Sistema de controle de acesso**

**CAMPINAS – SP**

**2015**

**FACULDADE ANHANGUERA DE CAMPINAS**

**UNIDADE 1**

**Manual do Projeto**

**Sistema de controle de acesso**

**Histórico da Revisão:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Criação:** | **Versão** | **Descrição** | **Data** |
| Sidmar Porfiírio | 1.0 | Elaboração manual técnico da aplicação | 30/04/2015 |
|  | | | |
| **Alteração:** | **Versão** | **Descrição** | **Data** |
| Sidmar Porfiírio | 1.0.1 | Alteração casos de uso, diagrama de classes, diagrama de sequência e diagrama de atividades. | 17/05/2015 |
| Sidmar Porfírio | 1.0.2 | Alteração diagrama de classes | 20/05/2015 |
|  |  |  |  |

**CAMPINAS**

**2015**

**SUMÁRIO**

Sumário

[1. INTRODUÇÃO 4](#_Toc418609609)

[2. PROJETO LÓGICO 5](#_Toc418609610)

[2.1 Modelagem de Negócio 5](#_Toc418609611)

[4. MODELAGEM DE DOMÍNIO 9](#_Toc418609612)

[4.1 Diagrama de Classes 9](#_Toc418609613)

[4.2 Diagrama de Atividades 10](#_Toc418609614)

[4.3 Diagrama de Sequencia 12](#_Toc418609615)

[4. HARDWARE 13](#_Toc418609616)

[4.1 ARDUINO MEGA 2560 13](#_Toc418609617)

[4.2 MÓDULO BLUETOOTH HC-06 14](#_Toc418609618)

[4.3 LCD 16X2 15](#_Toc418609619)

[4.4 ESQUEMA DE LIGAÇÃO 15](#_Toc418609620)

[5. PROJETO FÍSICO 16](#_Toc418609621)

[5.1 PASTAS 17](#_Toc418609622)

[5.1.1 PASTA /src 17](#_Toc418609623)

[5.1.2 PACOTE com.prointer4 17](#_Toc418609624)

[5.1.3 PASTA /gen 17](#_Toc418609625)

[5.1.4 PASTA /Android (número de versão) 17](#_Toc418609626)

[5.1.5 PASTA /assets 17](#_Toc418609627)

[5.1.6 PASTA /res 18](#_Toc418609628)

[5.1.7 /res/layout 18](#_Toc418609629)

[5.1.8 /res/values 19](#_Toc418609630)

[5.1.9 AndroidManifest.xml 19](#_Toc418609631)

[6. REFERÊNCIA BIBLIOGRÁFICAS 19](#_Toc418609632)

[7. ANEXOS 20](#_Toc418609633)

[7.1 DATASHEET ARDUINO 21](#_Toc418609634)

[7.2 DATASHEET MÓDULO BLUETOOTH HC-06 23](#_Toc418609635)

[7.3 DATASHEET LCD 1602ª 25](#_Toc418609636)

[Pin Description: 26](#_Toc418609637)

# INTRODUÇÃO

No mundo de hoje, com a evolução da tecnologia e a concorrência de mercado, faz-se necessário ter um maior controle sobre quem tem acesso as informações e ao ambiente das empresas. Garantir o não vazamento de informações tornou-se indispensável para que essas empresas não percam sua presença no mercado, e possam garantir

que, esta companhia seja a primeira a lançar determinado produto no mercado, que possa manter em segredo seus produtos não patenteados e novas tecnologias em desenvolvimento.

Hoje em dia o mercado conta com diversas soluções para identificação e controle de acesso a determinadas áreas de uma empresa, contamos desde as catracas simples que utilizam cartões do tipo RFID que possuem como nível de segurança, apenas um código que identifica o cartão e pode facilmente ser clonado.

Existem também os sistemas de controle de acesso por biometria, estes sistemas podem trabalhar identificando o usuário através da leitura da impressão digital, da leitura da íris, ou ambas. O nível de segurança desses sistemas é elevadíssimo, uma vez que, essas características biométricas são únicas e não podem ser clonas. Mas tão elevado quanto sua segurança é o seu custo, tornando-os inviável para uma empresa de pequeno porte.

Com ênfase nesta necessidade do mercado, justifica-se a concepção de um novo sistema de controle de acesso de baixo custo, onde o usuário efetua o acesso via smartphone, por meio de comunicação sem fio Bluetooth.

# 2. PROJETO LÓGICO

## 2.1 Modelagem de Negócio

**CASOS DE USO:**

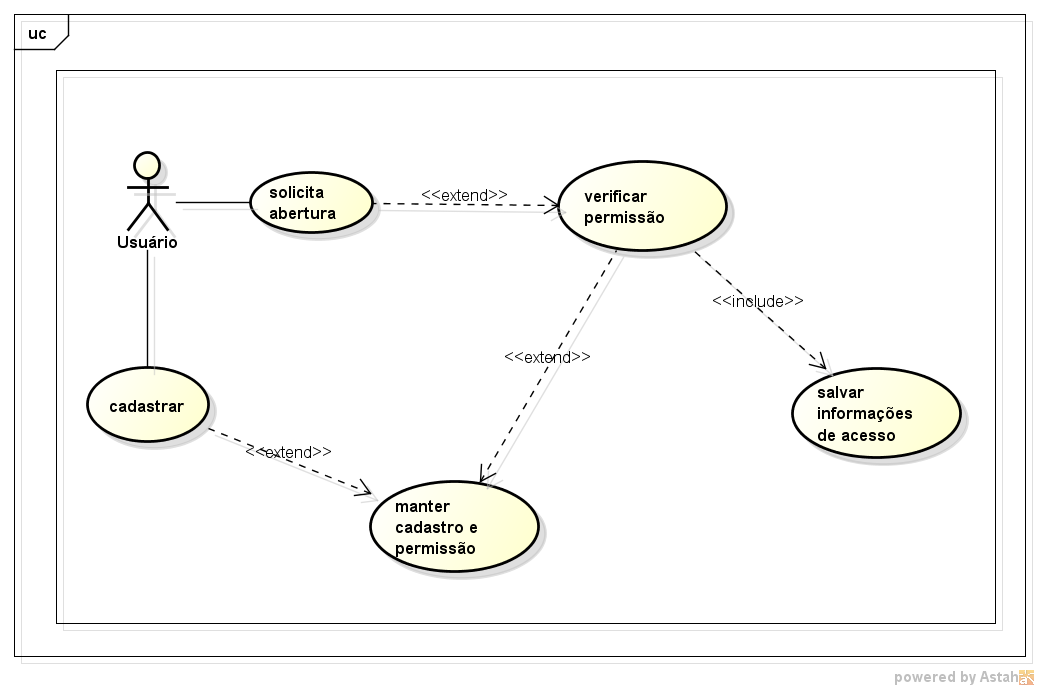


Figura Caso de uso

**DESCRIÇÃO DE CASOS DE USO:**

|  |  |  |  |
| --- | --- | --- | --- |
| UC001 – Solicitar Abertura | | | |
| **ATOR:** | Usuário | **REQUISITO FUNCIONAL** | REQ001 |
| **PRÉCONDIÇÃO:** | Possuir smartphone Android com o aplicativo de acesso instalado e emparelhado com o sistema de acesso. | **PÓS CONDIÇÃO:** | Possuir permissão de acesso. |
| **DESCRIÇÃO:** O usuário aproxima-se do dispositivo de acesso tendo em mãos o smartphone com o aplicativo de acesso ligado, com a conexão Bluetooth ligada e devidamente emparelhada com o sistema. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| UC002 – Conectar | | | |
| **ATOR:** | Smartphone | **REQUISITO FUNCIONAL** | REQ003 |
| **PRÉCONDIÇÃO:** | Sistema operacional Android com o aplicativo de acesso instalado e configurado com o sistema de acesso. | **PÓS CONDIÇÃO:** |  |
| **DESCRIÇÃO:** Após a solicitação do usuário, o smartphone envia uma mensagem ao sistema solicitando o acesso. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| UC003 – Enviar IMEI e senha | | | |
| **ATOR:** | Hardware | **REQUISITO FUNCIONAL** |  |
| **PRÉCONDIÇÃO:** |  | **PÓS CONDIÇÃO:** | Identificar a solicitação |
| **DESCRIÇÃO:** Após a conexão, o smartphone envia uma mensagem para o sistema contendo IMEI + senha, que foi previamente digitada pelo usuário. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| UC004 – Verificar Permissão IMEI | | | |
| **ATOR:** | hardware | **REQUISITO FUNCIONAL** | REQ003 |
| **PRÉCONDIÇÃO:** | Identificar solicitação | **PÓS CONDIÇÃO:** |  |
| **DESCRIÇÃO:** O sistema verifica em seu banco de dados se o smartphone que solicitou acesso possui autorização de acesso. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| UC005 – Autorizar Acesso | | | |
| **ATOR:** | Hardware | **REQUISITO FUNCIONAL** | REQ003 |
| **PRÉCONDIÇÃO:** | Possuir permissão de acesso | **PÓS CONDIÇÃO:** |  |
| **DESCRIÇÃO:** Após verificar que o IMEI+senha do smartphone possui cadastro e está autorizado a acessar a área desejada, o sistema libera a catraca e envia uma mensagem de autorização ao smartphone. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| UC006 – Negar Acesso | | | |
| **ATOR:** | Hardware | **REQUISITO FUNCIONAL** | REQ003 |
| **PRÉCONDIÇÃO:** | Não possuir permissão de acesso | **PÓS CONDIÇÃO:** |  |
| **DESCRIÇÃO:** Após verificar que o IMEI+senha do smartphone não possui cadastro, ou não possui permissão para acessar a área desejada, o hardware não destrava a catraca e envia uma mensagem de negação de acesso, o hardware também bloqueia uma nova tentativa de acesso por 5 segundos. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| UC007 – Exibir Mensagem | | | |
| **ATOR:** | Smartphone/hardware | **REQUISITO FUNCIONAL** | REQ003 |
| **PRÉCONDIÇÃO:** |  | **PÓS CONDIÇÃO:** |  |
| **DESCRIÇÃO:** Após executadas todas as operações anteriores, se o acesso for permitido; exibe uma mensagem de boas vidas na catraca com o nome do usuário e uma mensagem de autorização no dispositivo. Se o acesso for negado; a catraca permanece travada e é exibida uma mensagem de acesso negado na catraca e no smartphone além de fechar o aplicativo e travar a catraca para novo acesso durante 5 segundos. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| UC008 – Salvar Informações de Acesso | | | |
| **ATOR:** | Smartphone/hardware | **REQUISITO FUNCIONAL** | REQ002 |
| **PRÉCONDIÇÃO:** |  | **PÓS CONDIÇÃO:** |  |
| **DESCRIÇÃO:** Após executadas todas as operações anteriores, as informações de acesso são salvas tanto no smartphone, quanto no sistema de acesso. São salvos os dados do usuário cadastrados no sistema: nome, IMEI, data e hora; caso o acesso seja negado, estas informações também são armazenadas no banco de dados do sistema de acesso, a fim de detectar usuários tentando adentrar áreas da qual eles não possuam acesso. | | | |

# MODELAGEM DE DOMÍNIO

## 3.1 Diagrama de Classes

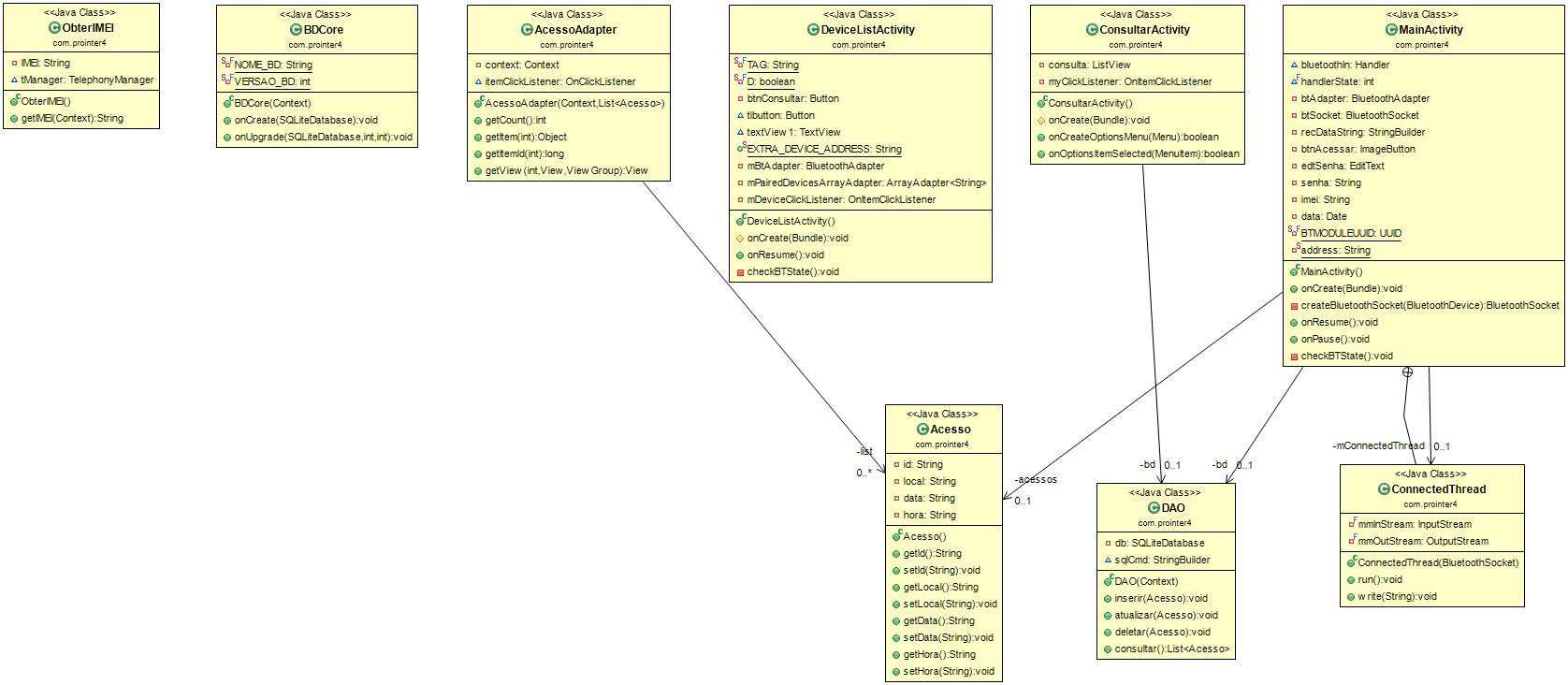


Figura Diagrama de Classe

A aplicação tem basicamente 9 classes, destas 9 as 2 mais importantes são DeviceListActivity.class que é responsável pela lógica da tela que exibe os dispositivos pareados, as sua principais variáveis são:

* EXTRA\_DEVICE\_ADDRESS do tipo String:

Que armazena o endereço do dispositivo ao qual queremos nos conectar.

* mBtAdapter do tipo BluetoothAdapter:

Armazena o adaptador Bluetooth padrão do dispositivo.

* mPairedDevicesArrayAdapter que é uma coleção do tipo String:

Contém todos os dispositivos pareados.

A classe MainActivity.class é a classe responsável pela atividade principal da aplicação, esta classe possui 7 variáveis principais:

* bluetoothIn do tipo Handler.

Existem dois principais usos para um Handler: (1) para agendar mensagens e Runnables para ser executado como algum momento no futuro; e (2) para enfileirar uma ação a ser executada em um segmento diferente do seu.

* handlerState do tipo int:

Usado para identificar a mensagem handler.

* btAdapter do tipo BluetoothAdapter:

Armazena o adaptador Bluetooth padrão do dispositivo.

* btSocket do tipo BluetoothSocket:

Interface para socketes Bluetooth, similar aos socketes TCP.

* recDataString do tipo StringBuilder:

Armazena as mensagens recebidas via Bluetooth.

* BTMODULEUUID do tipo UUID:

Cria um endereço de conexão única de 32 bits para o socket.

* address do tipo String:

String que armazena o endereço MAC.

Todas essas operações são executadas através de uma classe interna do tipo Thread (Threads são funções ou pequenos programas que podem ser executados em paralelo com a aplicação) ConnectedThread que possui apenas um método run(), que inicia a Thread e um método write(), que envia mensagens para o outro dispositivos Bluetooth.

A classe DAO.class (Data Acess Object) é a classe responsável por manipular as informações de banco de dados, ela possui 2 funções: salvar() responsável por persistir as informações referentes ao acesso no banco de dados, e também a função consultar(), responsável pelo acesso as consultas no banco de dados.

Esta classe possui também 7 variáveis:

* db do tipo SQLiteDatabase:

SQLiteDatabase possui métodos para criar, excluir, executar comandos SQL, e executar outras tarefas de gerenciamento de banco de dados comum.

* sqlCmd do tipo StringBuilder:

StringBuilder contendo os comandos SQL de inserção, consulta, exclusão etc.

* id do tipo int:

variável que armazena o código de identificação única de uma linha em uma tabela SQL.

* calendario do tipo Calendar:

Subclasses de Calendário, interpreta uma data de acordo com as regras de um sistema de calendário específico.

* posto do tipo String:

String que armazena o nome do ponto de acesso ao qual o usuário acesso, ou esta tentando acessar.

* data do tipo String:

Data em que o usuário acessou o posto.

* hora do tipo String:

Hora em que o usuário acessou o posto.

E finalmente a classe ObterIMEI.class, responsável por obter o IMEI do dispositivo e passar para a aplicação, esta classe possui apenas 2 variáveis e um método:

* IMEI do tipo String:

String do tipo private que armazena o IMEI do dispositivo, é encapsulada e só pode ser obtida através de método getIMEI().

* tManager do tipo TelephonyManager:

Fornece acesso a informações sobre os serviços de telefonia no dispositivo. Os aplicativos podem usar os métodos nesta classe para determinar os serviços de telefonia e estados, bem como acessar alguns tipos de informações.

* Função get IMEI(context):

Função usada para obter o IMEI do dispositivo, recebe como parâmetro o contexto da aplicação.

## Diagrama de Atividades

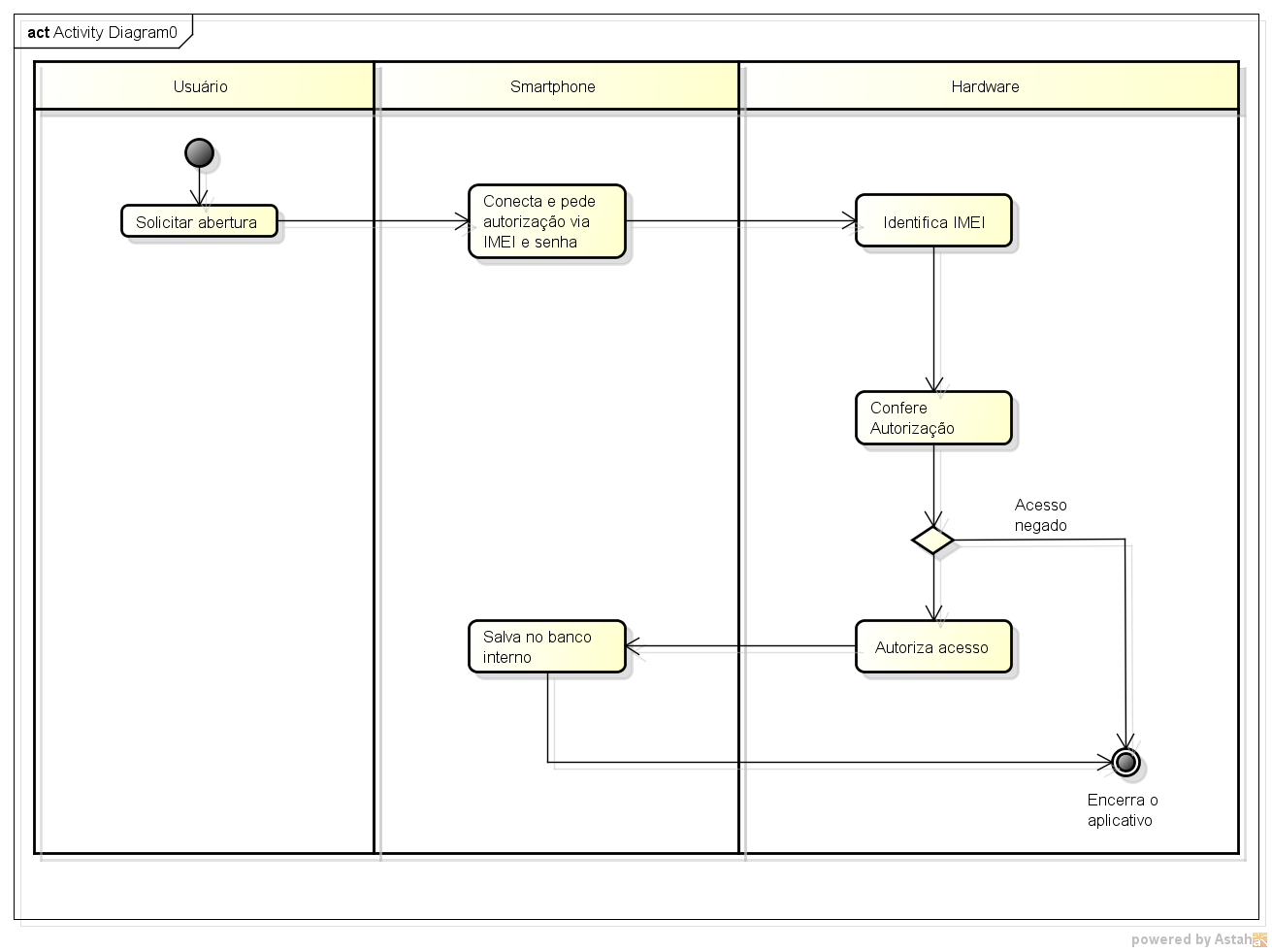


Figura Diagrama de Atividades

## Diagrama de Sequencia

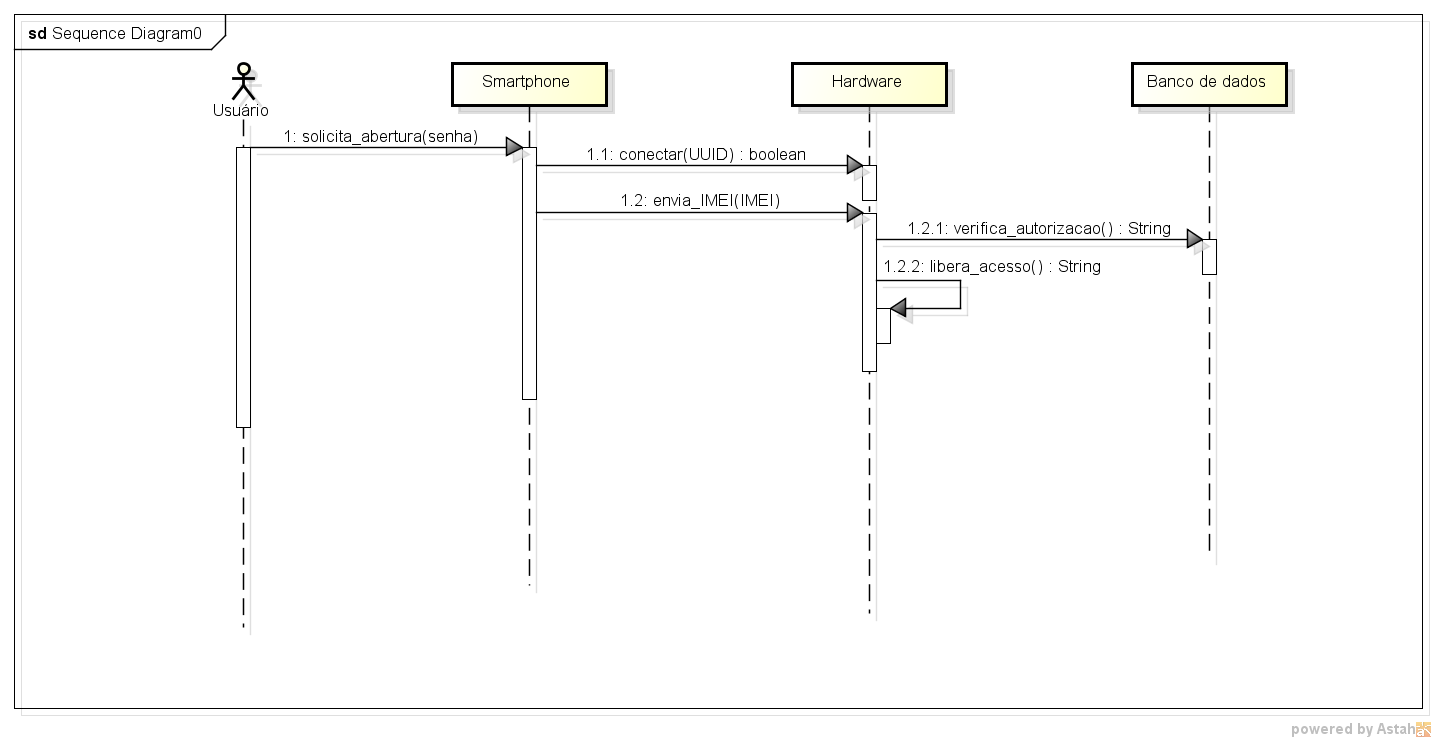


Figura Diagrama de sequencia

# HARDWARE

## 4.1 ARDUINO MEGA 2560

O Arduino Mega 2560 é uma placa de microcontrolador baseada no ATmega2560. Ele possui 54 pinos de entradas/saídas digitais, 16 entradas analógicas,4 UARTs (portas seriais de hardware), um oscilador de cristal de 16 MHz, uma conexão USB, uma entrada de alimentação, uma conexão ICSP e um botão de reset. Ele contém tudo o que é necessário para dar suporte ao microconrolador; basta conectar a um computador com um cabo USB ou a uma fonte de alimentação e já está pronto para começar. O mega é compatível com a maioria dos shields desenhados para os Arduino Uno, Duemilanove e para o Diecimila. Possui ainda o dobro de memória do antigo Arduino Mega. (Datasheet vide anexo)



Figura Arduino Mega 2560

## MÓDULO BLUETOOTH HC-06

O módulo Bluetooth HC-06 é usado para comunicação wireless entre o Arduino e algum outro dispositivo com bluetooth, como por exemplo um telefone celular, um computador ou tablet. As informações recebidas pelo módulo são repassadas ao Arduino (ou outro microcontrolador) via serial.

O alcance do módulo segue o padrão da comunicação bluetooth, que é de aproximadamente 10 metros. Esse módulo funciona apenas em modo slave (escravo), ou seja, ele permite que outros dispositivos se conectem à ele, mas não permite que ele próprio se conecte à outros dispositivos bluetooth. (Datasheet vide anexo)

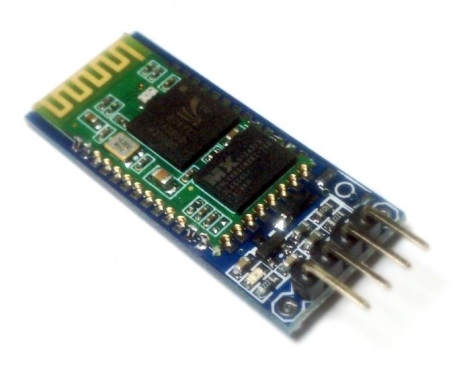


Figura Módulo Bluetooth HC-06

## 4.3 LCD 16X2

Esse tipo de display pode funcionar em modo 4 bits (usando 4 pinos), ou no modo 8 bits, usando todos os pinos de dados. Certamente quem usa o Arduino Uno e outras placas Arduino menores precisa economizar portas, especialmente se for ligar, ao mesmo tempo, o display e vários sensores. (Datasheet vide anexo)



Figura Display LCD 1602A

## ESQUEMA DE LIGAÇÃO

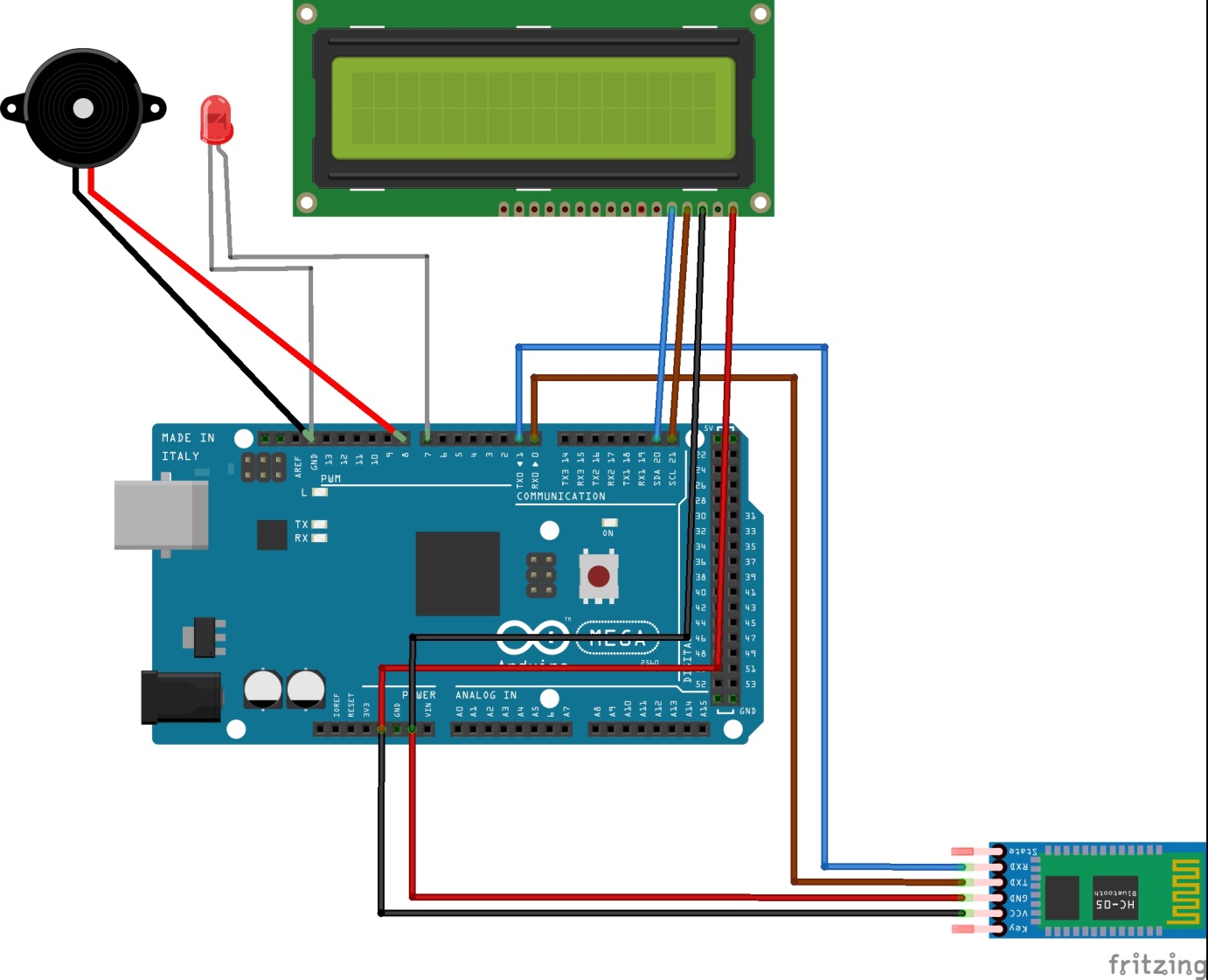


Figura Esquema de ligação dos componentes

# PROJETO FÍSICO

## 5.1 PASTAS

Abaixo seguem todos os diretórios e arquivos do projeto, e um descritivo breve de suas funções na estrutura do projeto.

### 5.1.1 PASTA /src

Esta pasta deverá conter os arquivos fontes Java. Os arquivos deverão estar organizados de acordo com a estrutura de pacotes do projeto, que é semelhante à qualquer projeto Java.

### 5.1.2 PACOTE com.prointer4

O empacotamento é uma abordagem comum usada para organizar classes e interfaces relacionadas. O código geralmente mais reutilizável é empacotado. As classes não empacotadas geralmente são encontradas em livros e tutoriais on-line, bem como em aplicativos de software com foco mais estrito.

O pacote **com.prointer4** armazena as classes responsáveis pela lógica da aplicação.

### 5.1.3 PASTA /gen

Também é uma pasta de arquivos fontes, só que neste caso são os arquivos fontes automaticamente gerados pela plataforma Android. A classe R gerada conterá a estrutura do layout para que seja referenciada no projeto.

### 5.1.4 PASTA ****/Android (número de versão)****

Esta pasta conterá as bibliotecas (jars) que serão necessárias no projeto.

### 5.1.5 PASTA /assets

Esta pasta também conterá recursos externos utilizáveis pela aplicação, porém diferentemente da pasta ‘/res’, estes recursos serão acessíveis somente programaticamente.

### 5.1.6 PASTA /res

Neste diretório serão colocados todos os recursos externos (como imagens, arquivos de dados) para serem usados na aplicação. Este diretório contem as subpastas a seguir.

#### 5.1.6.1 /res/drawable

Nesta pasta serão incluídas todas imagens usadas no projeto.

#### 5.1.6.1 /res/drawable-hdpi

Imagens para tela de alta densidade (~240dpi);

#### 5.1.6.1 /res/drawable-ldpi

Imagens para tela de alta densidade (~120dpi);

#### 5.1.6.1 /res/drawable-mdpi

Imagens para tela de alta densidade (~160dpi);

### ****5.1.7 /res/layout****

Conterá os layouts das interfaces criadas no projeto. Serão armazenados como arquivos xml que serão interpretados pelo Android.

### ****5.1.8 /res/values****

Nesta pasta serão salvos arquivos xml que deverão conter pares de chave-valor referenciados na aplicação. A ideia é que sejam arquivos contendo valores como arrays, cores, dimensões, strings e etc. É útil para organizar as mensagem exibidas na aplicação, por exemplo.

### ****5.1.9 AndroidManifest.xml****

Por fim, mas não menos importante, este arquivo é muito importante para o projeto porque ele conterá as meta informações sobre a aplicação Android. Conterá informações sobre as atividades, views, serviços e etc, bem como a lista de permissões que deverão ser necessárias para rodar a aplicação.

# REFERÊNCIA BIBLIOGRÁFICAS

# ANEXOS

## 7.1 DATASHEET ARDUINO

|  |  |
| --- | --- |
| Microcontroller | ATmega2560 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limits) | 6-20V |
| Digital I/O Pins | 54 (of which 15 provide PWM output) |
| Analog Input Pins | 16 |
| DC Current per I/O Pin | 40 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 256 KB of which 8 KB used by bootloader |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Clock Speed | 16 MHz |

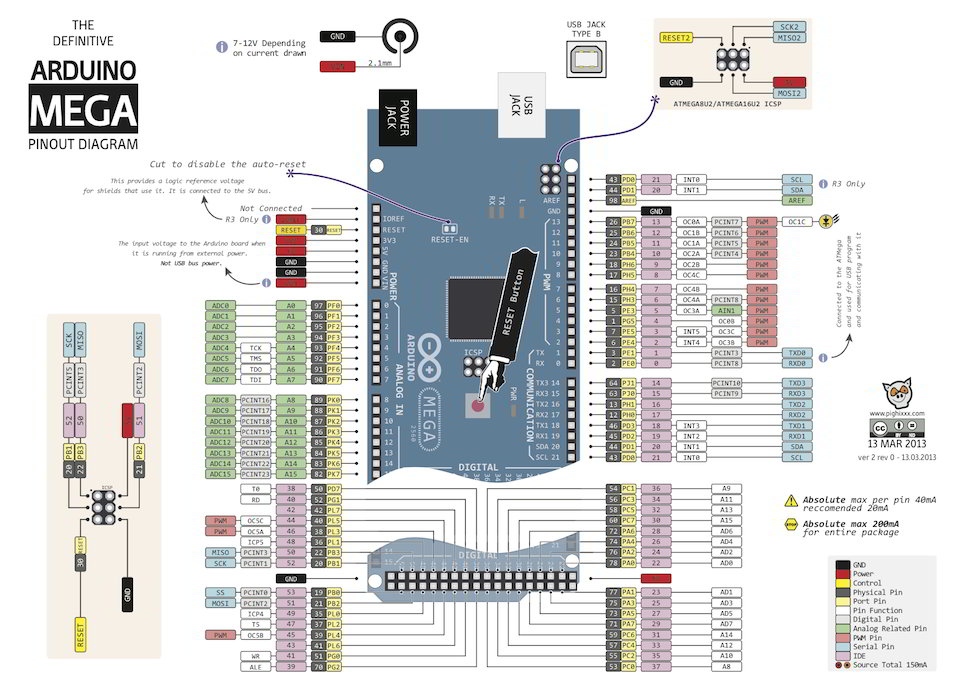


Figura Conexões Arduino Mega 2560

## DATASHEET MÓDULO BLUETOOTH HC-06

**Feature :**  
  
1、Bluetooth protocol:  Bluetooth Specification v2.0+EDR  
  
2、Frequency:  2.4GHz ISM band  
  
3、Modulation:  GFSK(Gaussian Frequency Shift Keying)  
  
4、Emission power:  ≤4dBm, Class 2  
  
5、Sensitivity:  ≤-84dBm at 0.1% BER  
  
6、Speed: Asynchronous:  2.1Mbps(Max) / 160 kbps, Synchronous: 1Mbps/1Mbps  
  
7、Security:  Authentication and encryption  
  
8、Profiles:  Bluetooth serial port  
  
9、Power supply: +3.3VDC 50mA  
  
10、Working temperature: -20 ~ +75 Centigrade  
  
11、Dimension: 26.9mm x 13mm x 2.2 mm  
  
**Default :**  
  
Slave, 9600 baud rate, N, 8, 1. Pincode 1234  
  
**AT command :**  
  
**1. Communications Test :**  
  
Sent : AT  
  
Receive : OK  
  
**2. Change baud rate :**  
  
Sent : AT+BAUD1  
  
receive : OK1200  
  
Sent : AT+BAUD2  
  
Receive : OK2400   
  
1---------1200  
2---------2400  
3---------4800  
4---------9600  
5---------19200  
6---------38400  
7---------57600  
8---------115200  
  
Baud rate setting can be save even power down.  
   
**3. Change Bluetooth device name :**  
  
Sent : AT+NAMEdevicename  
  
receive : OKname  
  
(devicename is the name you want the device to be , and it will be searched with this name)  
  
Name setting can be save even power down.  
  
**4. Change Pincode :**  
  
Sent : AT+PINxxxx  
  
receive : OKsetpin  
  
(xxxx is the pin code you set)  
  
Pin code can be save even power down.

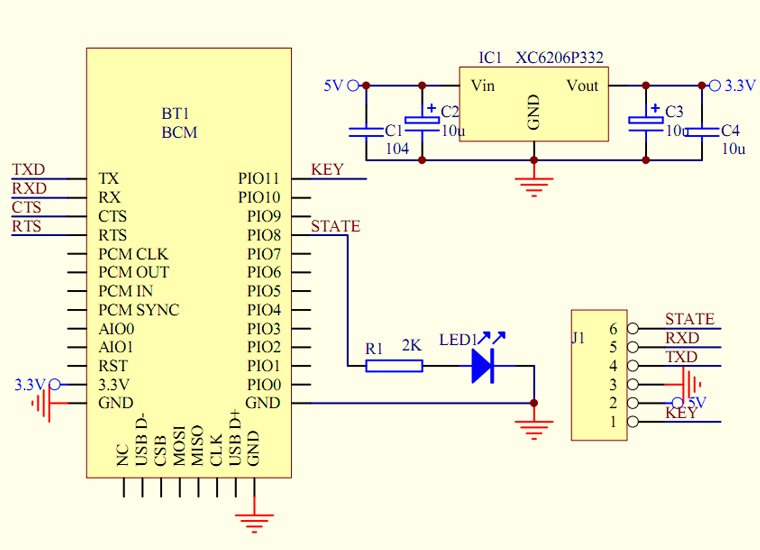


Figura Esquema elétrico sensor HC-06

## DATASHEET LCD 1602A

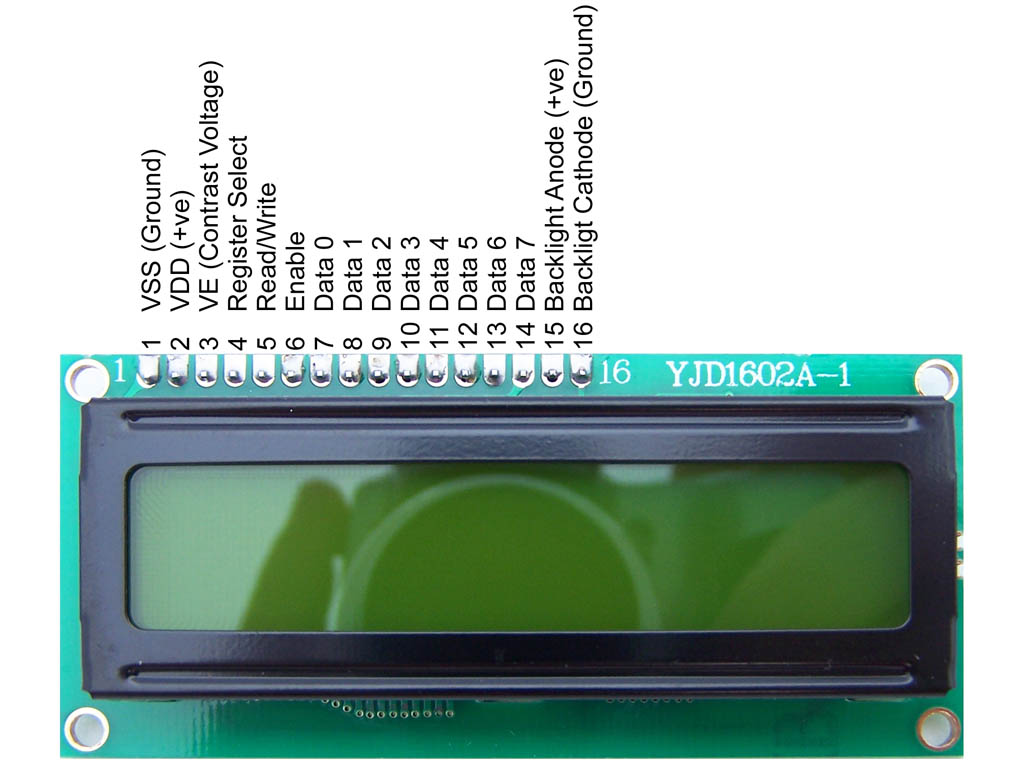


Figura Pinagem LCD 1602A

### Pin Description:

|  |  |  |
| --- | --- | --- |
| **Pin No** | **Function** | **Name** |
| 1 | Ground (0V) | Ground |
| 2 | Supply voltage; 5V (4.7V – 5.3V) | Vcc |
| 3 | Contrast adjustment; through a variable resistor | VEE |
| 4 | Selects command register when low; and data register when high | Register Select |
| 5 | Low to write to the register; High to read from the register | Read/write |
| 6 | Sends data to data pins when a high to low pulse is given | Enable |
| 7 | 8-bit data pins | DB0 |
| 8 | DB1 |
| 9 | DB2 |
| 10 | DB3 |
| 11 | DB4 |
| 12 | DB5 |
| 13 | DB6 |
| 14 | DB7 |
| 15 | Backlight VCC (5V) | Led+ |
| 16 | Backlight Ground (0V) | Led- |