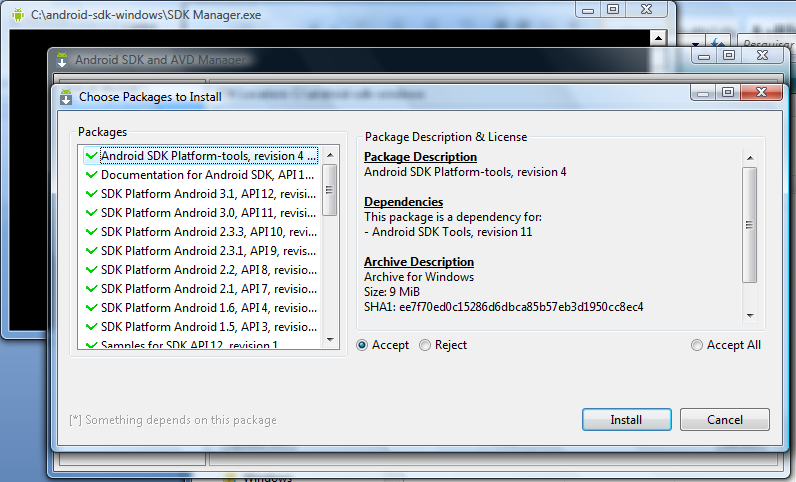
O projeto esta em fase de desenvolvimento e já possui algumas funções implementadas e testadas.

A seguir é apresentado um guia de configuração de ambiente para o desenvolvimento Android. Conforme já foi mencionado, a aplicação Android será responsável por interpretar os comandos passados por SMS e acionar o microcontrolador via bluetooh para refletir os comandos no carro.

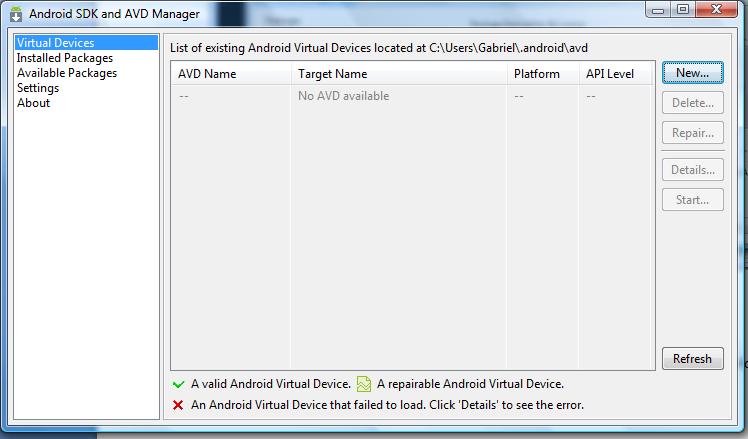
Para começar a configuração é necessário fazer o download do SDK em <http://dl.google.com/android/android-sdk_r11-windows.zip> (acesso em 13/05/2011).

Descompactar o arquivo em qualquer lugar (sugestão em c:\), em seguida abrir o executável “SDK Manager.exe” e clicar em Install, conforme figura XX.



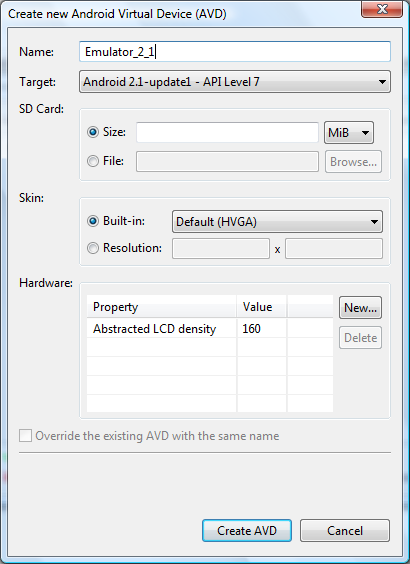
(figura xx)

Para criar um emulador clique em “Virtual Devices” e em seguida em “New”, esse passo pode ser visto na figura xx



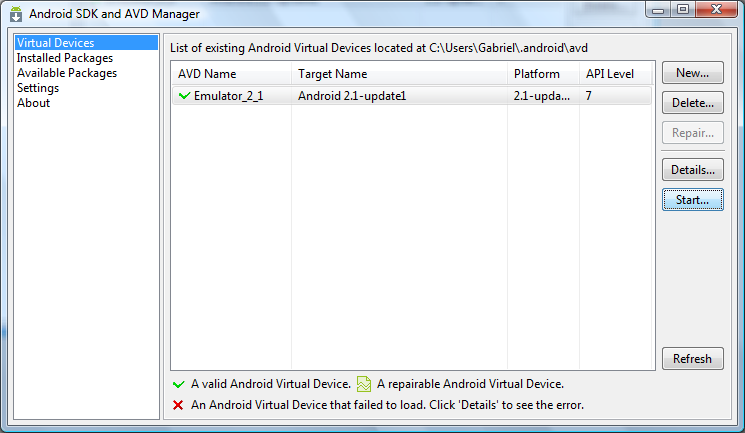
(figura xx)

Adicionar um nome qualquer para o emulador, selecionar a versão do android que irá rodar e clicar em Create AVD, conforme figura xx

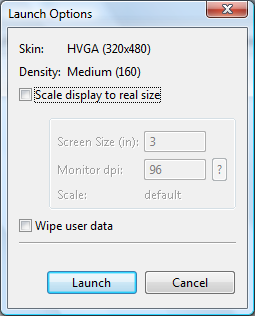


(figura xx)

Para testar o emulador criado, basta selecioná-lom, clicar em “Start...” e em seguida em Launch, passos exibidos nas figura xx e figura xx respectivamente.



(figura xx)



(figura xx)

Se tudo ocorrer sem erros, o emulador irá aparecer conforme figura xx.

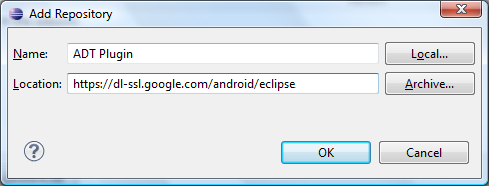


(figura xx)

A IDE utilizada para programar é o Eclipse e para fazer o download acesse <http://www.eclipse.org/downloads/packages/eclipse-classic-362/heliossr2> (acesso em 13/05/2011).

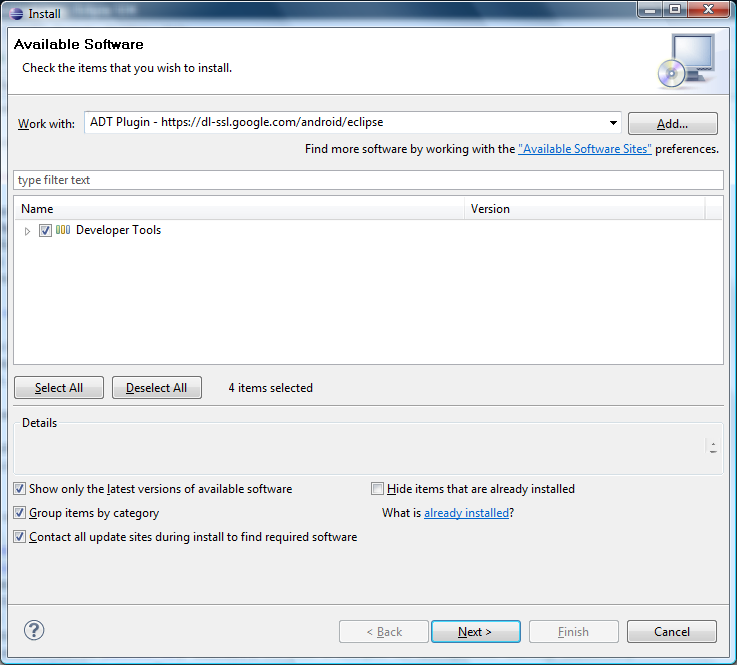
A IDE Eclipse não é necessário instalar, é preciso apenas descompactá-la em qualquer lugar do computador. Sugestão de local é em C:\eclipse.

Para desenvolver para Android dentro do eclipse, é necessário instalar o ADT plugin, para isso, basta dentro do eclipse acessar “Help > Install New Software “, em seguida clicar em “Add”. No campo Name digitar “ADT plugin” e no Location “ <https://dl-ssl.google.com/android/eclipse/> “ (acesso em 13/05/2011), conforme figura xx.



(figura xx)

Em seguida, selecionar em “work with” o ADT Plugin adicionado no passo anterior, clicar no check box da opção Developer Tools, clicar em next e terminar a instalação, conforme figura xx.



(figura xx)

Com o ambiente android pronto, foi necessário desenvolver o esquema elétrico e o protótipo do hardware que será responsável por receber os códigos da aplicação Android e refletir no veiculo, como por exemplo, o código que pede para desligar o carro e acionar o alarme, para travar ou destravar as portas. Na figura xxxx e xxxx é apresentado o protótipo e esquema elétrico do hardware.



Figura xxxx

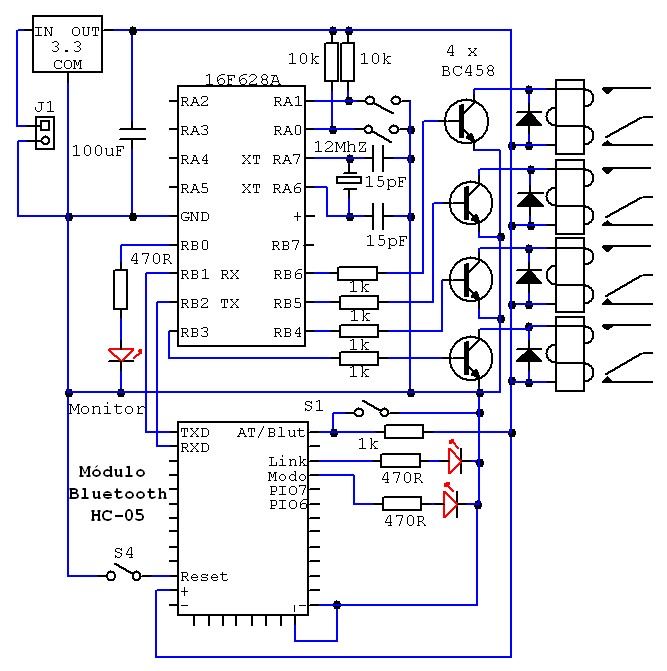


Figura xxxx

Como um dos pontos mais críticos e difíceis do projeto era a comunicação entre o celular e o microcontrolador, todo esforço inicial foi focado na resolução desse problema. Para solucioná-lo, foi utilizada uma API nativa chamada BluetoothAdapter. Com ela é possível rastrear os dispositivos Bluetooth ao alcance e parear com o módulo. O código utilizado para rastear os dispositivos e conectar e eles pode ser visto na figura xxxx e figura xxxx respectivamente.

**private** **final** BroadcastReceiver mReceiver = **new** BroadcastReceiver() {

**public** **void** onReceive(Context context, Intent intent) {

String action = intent.getAction();

**if** (BluetoothDevice.*ACTION\_FOUND*.equals(action)) {

BluetoothDevice deviceFound =

intent.getParcelableExtra(BluetoothDevice.*EXTRA\_DEVICE*);

**for** (BluetoothDevice device : lstDevices) {

**if**(deviceFound.getAddress()

.equals(device.getAddress())) {

**return**;

}

}

lstDevices.add(deviceFound);

}

}

};

Figura xxxx

**private** **void** connect() {

BluetoothSocket tmp = **null**;

**try** {

Method m = alarmeDevice.getClass().getMethod(

"createRfcommSocket", **new** Class[] {**int**.**class**});

tmp = (BluetoothSocket) m.invoke(alarmeDevice, 1);

} **catch** (SecurityException e) { e.printStackTrace();

} **catch** (NoSuchMethodException e) { e.printStackTrace();

} **catch** (IllegalArgumentException e) {

e.printStackTrace();

} **catch** (IllegalAccessException e) {

e.printStackTrace();

} **catch** (InvocationTargetException e) {

e.printStackTrace();

}

btSocket = tmp;

mBluetoothAdapter.cancelDiscovery();

**try** {

btSocket.connect();

outStream = btSocket.getOutputStream();

} **catch** (IOException e) {

**try** {

btSocket.close();

} **catch** (IOException e2) {

e2.printStackTrace();

}

**return**;

}

}

Figura xxxx

Para testar a comunicação de envio e recebimento de dados foi o utilizado o módulo Bluetot