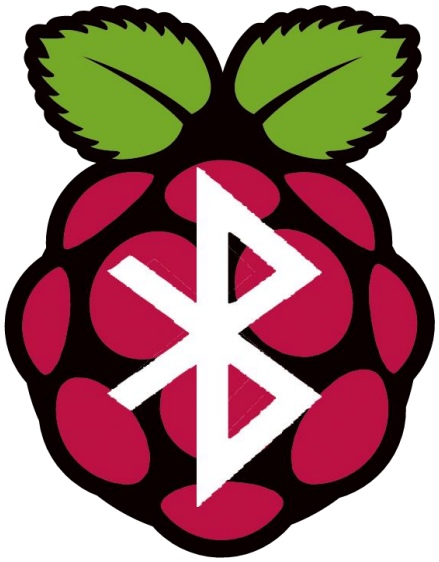
HC-05: Comunicação Bluetooth aplicada ao Raspberry Pi

Wilson Borba da Rocha Neto



Teresina – Piauí

2015

1. Introdução

O Raspberry Pi (a partir de agora chamado de RPi) é relativamente novo no Brasil. Isso faz com que os tutoriais em português sejam escassos, e, também, os estudantes ou curiosos no assunto sentirem-se desmotivados a começar o processo de aprendizagem.

Pensando nisso, estaremos trabalhando aqui na configuração do Módulo Bluetooth HC-05 e a sua comunicação com o raspberry via terminal, utilizando tanto a comunicação serial pré-programada quanto uma com propriedades escolhidas pelo usuário.

Para isso, é interessante que você, aquele que está interessado no assunto, já saiba o básico da configuração, funções e utilização do LXTerminal do RPi, pois não é nosso intuito dar uma introdução ao RPi e sim trabalha-lo com o bluetooth e, em foco, utilizando o módulo HC05.

Além disso, será trabalhado, também, o protocolo de comunicação Bluetooth em si, para que se possa ter uma maior compreensão da sua utilização no RPi.

2. O que é Bluetooth?

Bluetooth é um padrão global de comunicação sem fio e de baixo consumo de energia que permite a transmissão de dados entre dispositivos, desde que um esteja próximo ao outro.

A velocidade de transmissão de dados no Bluetooth é relativamente baixa:

- ✓ Versão 1.2 – até 1 Mb/s (megabit por segundo)
- ✓ Versão 2.0 - até 3 Mb/s.
- ✓ Versão 3.0 - capaz de atingir taxas de até 24 Mb/s.

Figura 1 – Símbolo representativo do Bluetooth

Fonte 1: http://www.masada.com.br/

3. Por que utilizar o Bluetooth?

Um dos pontos mais evidentes deste tipo de comunicação é a ausência de fios entre os dispositivos, o que é um ponto positivo tanto na questão estética quanto na ergonômica. Mas não só isso, o Bluetooth é um padrão global, ou seja, está presente em uma imensa gama de aparelhos de diferentes marcas, modelos e características, desde periféricos para computador até celulares, o que faz com o que não se precise ter problemas com a falta de dispositivos compatíveis entre si.

Entretanto, o ponto mais importante do Bluetooth, e o que o faz ser usado nos mais diversos aparelhos, é o consumo baixo de energia. Assim, ele foi preparado para gastar o mínimo possível, com implementações como o modo “sleep”.

4. Como funciona o Bluetooth?

4.1. O canal da comunicação

O Bluetooth adota a frequência de rádio aberta e aceita em praticamente qualquer lugar do planeta: A faixa *ISM* (*Industrial, Scientific, Medical*), que opera à frequência de 2,45 GHz, sendo utilizada em vários países, com variações que vão de 2,4 GHz a 2,5 GHz.

Como a faixa ISM é aberta, isto é, pode ser utilizada por qualquer sistema de comunicação, é necessário garantir que o sinal do Bluetooth não sofra interferência, assim como não a gere.

O esquema de comunicação *FH-CDMA* (*Frequency Hopping - Code-Division Multiple Access*), utilizado pelo Bluetooth, permite tal proteção, já que faz com que a frequência seja dividida em vários canais. O dispositivo que estabelece a conexão muda de um canal para outro de maneira bastante rápida. Este procedimento é chamado "salto de frequência" (*frequency hopping*) e permite que a largura de banda da frequência seja muito pequena, diminuindo sensivelmente as chances de interferência.

No enlace, isto é, à ligação entre o emissor e receptor, o Bluetooth faz uso, basicamente, de dois padrões: *SCO* (*Synchronous Connection-Oriented*) e *ACL* (*Asynchronous Connection-Less*).

4.2. Os padrões de comunicação

- ✓ SCO: estabelece um link sincronizado entre o dispositivo emissor e o dispositivo receptor, separando slots para cada um. Assim, acaba sendo utilizado no envio contínuo de dados, como transmissão de voz, não permitindo a retransmissão de pacotes de dados perdidos. Quando ocorre perda em uma transmissão de áudio, por exemplo, o dispositivo receptor acaba reproduzindo som com ruído.

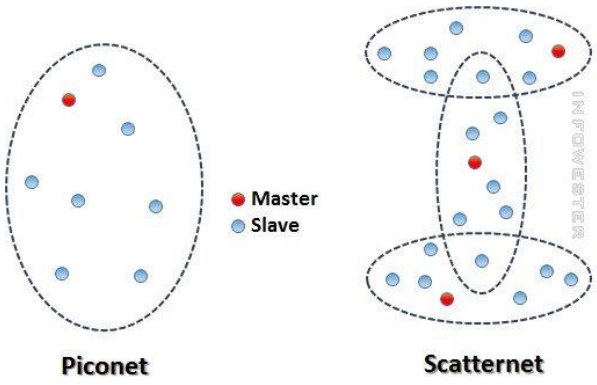
- ✓ ACL: estabelece um link entre o dispositivo que inicia e gerencia a comunicação e os demais que estão em sua rede. Este link é assíncrono, já que utiliza slots previamente livres. O ACL permite o reenvio de pacotes de dados perdidos, garantindo a integridade das informações trocadas entre os dispositivos. Assim, este padrão acaba sendo útil para aplicações que envolvam transferência de arquivos, por exemplo.

4.3. Redes e Hierarquia

Quando dois ou mais dispositivos se comunicam por meio de uma conexão Bluetooth, eles formam uma rede denominada piconet. Nesta comunicação, o dispositivo que iniciou a conexão assume o papel de master (mestre), enquanto que os demais dispositivos se tornam slave (escravos). Cabe ao master a tarefa de regular a transmissão de dados na rede e o sincronismo entre os dispositivos.

Cada piconet pode suportar até 8 dispositivos (um master e 7 slaves), no entanto, é possível elevar este número a partir da sobreposição de piconets. Em poucas palavras, este procedimento consiste em fazer com que uma piconet se comunique com outra que esteja dentro do limite de alcance, esquema este denominado *scatternet*. Note que um dispositivo slave pode fazer parte de mais de uma piconet ao mesmo tempo, no entanto, um master pode ocupar esta posição somente em uma única piconet.

Figura 2



Fonte 2

4.4. Uma visão mais técnicas: As camadas e o protocolo em si*

Pode-se dividir a comunicação bluetooth e os enlaces falados anteriormente em camadas:

- ✓ Física: Define as especificações que um aparelho deve possuir para operar com a tecnologia bluetooth

- ✓ Enlace: São os links descritos anteriormente (SCO e ACL)

A informação é transmitida pelo enlace na forma de pacotes os quais possuem seus modelos explicitados a seguir:

Tabela 1 – O pacote de dados Bluetooth

Código de Acesso	Cabeçalho	Dados
72 bits	54 bits	Até 2745 bits

Há três tipos de Código de Acesso:

- ✓ CAC (Channel Access Code – Código de Acesso ao Canal) – identifica a piconet
- ✓ DAC (Device Access Code – Código de Acesso do Aparelho) – usado durante o pedido e resposta de conexão (chamado de Paging Procedure).
- ✓ IAC (Inquiry Access Code – Código de Acesso de Inquérito) – usado durante o chamado Processo de Inquérito, quando um aparelho determina quais os aparelhos estão operantes, seus endereços e clocks.

O Cabeçalho pode ser detalhado da seguinte forma:

Tabela 2 – O cabeçalho Bluetooth

Endereço	Tipo	Controle de Fluxo	ACK/NACK	Número de sequencia	Controle de erro do cabeçalho
3 bits	4 bits	1 bit	1 bit	8 bits	

Isto dá um total de 18 bits, que, usando-se codificação FEC 1/3, resulta no valor de 54 bits para o cabeçalho. O número de sequência serve para estabelecer uma ordem para os pacotes enviados. Há 15 diferentes tipos de pacotes, listados a seguir:

- ✓ ID: Tipo de pacote de 68 bits, usado para pedido de conexão e resposta ao pedido.
- ✓ NULL: tipo de pacote de 126 bits que tem somente o código de acesso ao canal e cabeçalho. É usado para retornar a informação de conexão à fonte e não requer resposta do tipo ACK ou NACK.
- ✓ POLL: similar ao pacote NULL, porém requer uma confirmação (ACK ou NACK) do destino.
- ✓ FHS: *Frequency Hopping Synchronization* (Sincronização dos saltos de frequência). Entre outras funções, este pacote carrega as informações de clock (sincronismo) do aparelho mestre.
- ✓ DM1: carregam somente dados. Em seu corpo têm até 18 bytes de informação e ainda 16 bits de CRC (Cyclic Redundancy Check – efetua controle de erro no corpo). São codificados com FEC 2/3.
- ✓ DH1: similar ao DM1, exceto que sua informação não recebe codificação FEC.
- ✓ AUX1: similar ao DH1, exceto que não possui o CRC.
- ✓ DM3, DH3, DM5 e DH5: muito similares aos já mencionados DM1 e DH1. O que muda em cada pacote é o tamanho da informação que transportam.
- ✓ **HV1**: High quality Voice. Estes pacotes carregam 10 bytes de informação, codificadas com FEC 1/3.
- ✓ HV2: similar ao HV1, porém carrega 20 bytes de informação, codificada com FEC 2/3.
- ✓ HV3: similar aos anteriores, porém carrega 30 bytes de informação e não possui codificação FEC.
- ✓ DV: Data Voice. Pacote para ambos dados e voz. É divido em 80 bits para voz e 150 bits para dados.

A parte de voz não é protegida por FEC, mas a parte de dados é codificada com FEC 2/3. Há controle de erros somente na parte de dados que, caso necessário, será retransmitida independentemente da parte de voz.

Os pacotes do tipo ID, NULL, POLL, FHS e DM1 são usados em conexões com enlace SCO e ACL; os DH1, AUX1, DM3, DH3, DM5 e DH5 são definidos somente para ACL; os HV1, HV2, HV3 e DV somente para SCO.

A parte de Dados pode ser detalhada da seguinte forma:

Tabela 3 - Dados Bluetooth resumidadmente

Cabeçalho	Corpo	Código CRC
De 8 a 16 bits	~Varia de acordo com a informação	16 bits

O cabeçalho dos Dados contém, principalmente, a informação do tamanho do Corpo, onde as informações são enviadas. O Código CRC, já mencionado acima, é responsável pelo controle de erro.

5. O Módulo HC05

Para que todas as praticidades do Bluetooth também sejam levadas até o RPi, precisamos encontrar um meio de inserir algum componente que torne possível a comunicação.

Muitas pessoas utilizam um simples *dongle bluetooth usb*, que pode ser plugado no RPi e instantaneamente, mas por que não utilizar um dongle?

5.1. Motivos para utilizar o HC05

- ✓ Facilmente encontrado nas mais diversas lojas de material eletrônico para Arduino e RPi: Este módulo Bluetooth é barato e pode ser facilmente encontrado nas mais diversas lojas virtuais especializadas no comércio de material eletrônico. Com uma boa pesquisa no Mercado Livre e verificando, sempre, a qualificação do vendedor, você pode encontra-lo em faixas de preço que não ultrapassam trinta reais.
- ✓ Compatibilidade completa com o RPi: Um dos grandes problemas de comprar um dongle é a compatibilidade e disponibilidade de drivers para o raspbian. Muitas vezes é difícil encontrar um dongle compatível no mercado brasileiro.
- ✓ Literatura disponível: Já existem diversas aplicações e guias de instalação disponíveis em sites especializados e em artigos. Fica fácil adaptar tais textos para a utilização e aplicação.
- ✓ Possibilidade de programação: Os comandos AT (que serão explicados posteriormente) podem ser utilizados para configurar o módulo do jeito necessário, de uma forma completamente intuitiva e otimizada.

5.2. Explorando a parte física

Uma das características desse módulo e a possibilidade de duplo “estado”, o que permite dois modos diferentes.

Geralmente, os módulos vendidos já vêm soldados em uma placa de apoio da cor azul. Essa placa, facilita a conexão e identificação dos pinos, além de possuir um circuito interno que protege o módulo de voltagem mais elevadas e facilita a pinagem, porém não o torna indestrutível.

O Módulo HC-05, nesse caso, possui seis pinos disponíveis para utilizarmos:

- ✓ STATE: Um pino de saída (OUTPUT). Indica a atividade do módulo bluetooth de acordo com a configuração da sua mudança entre nível alto e baixo. Um led pode ser ligado a este pino e ao pino GND do RPi para verificação do estado de conexão, por exemplo. Porém, a maioria dos módulos com placa já posuem um led embutido com esta função.
- ✓ GND: Este pino deve ser conectado ao Ground do raspberry, ou qualquer outro aterramento.
- ✓ VCC: Um pino de entrada(INPUT). Alimenta o módulo com 5V ou 3.3V, porém utilizaremos APENAS 3.3V, para evitar qualquer dano ao módulo e ao raspberry.
- ✓ WAKEUP: Um pino de entrada. Aceita apenas 3.3V, caso você coloque 5V irá, literalmente, fritar o módulo. Caso conecte-se esse pino a um nível HIGH, o módulo entrará em modo de Configuração, caso nível baixo, ele entrará no modo Passivo.

Figura 3 – O módulo Bluetooth HC-05

Fonte 3 – wikipedia.com

Antes de falar dos dois pinos restantes, vamos fazer uma breve explanação sobre comunicação serial.

5.3. Os pinos RXD, TXD e a comunicação serial

- ✓ TXD é traduzido do inglês como “Dados a serem transmitidos de uma ligação comum”.
- ✓ RXD é traduzido como “Dados a serem recebidos de uma ligação comum”.

Por ligação comum, entende-se um link entre dispositivos através de um canal comum a ambos, como o jumper que ligará nosso módulo ao raspberry.

Assim, para que o módulo possa se comunicar com o RPi, precisamos conectar o pino RXD do RPi no TXD do HC05 e o TXD no RXD do HC05, para que possam trocar informações.

Esses pinos são para um conexão serial assíncrona, ou seja, uma comunicação na qual os bits são transferidos um por um, e podem ir e vir pelo canal sem se preocupar com um tempo predeterminado.

De um forma mais técnica, pode-se dizer que esses pinos fazem parte da UART (Universal Asynchronous Receiver Transmitter).

6. Modos e conexão com o RPi

Trabalharemos dois modos de conexão deste módulo com o RPi. Com a utilização do pino WAKE UP, como pode ser visto em Explorando a parte física, teremos o modo de CONFIGURAÇÃO e o modo SLAVE.

Em ambas, primeiramente será feita uma breve explicação para relembrar as características do Bluetooth vistas anteriormente em Redes e Hierarquia, e também será mostrado a configuração dos pinos para, logo após, ser introduzida a configuração do RPi.

6.1. Configuração em modo Passivo

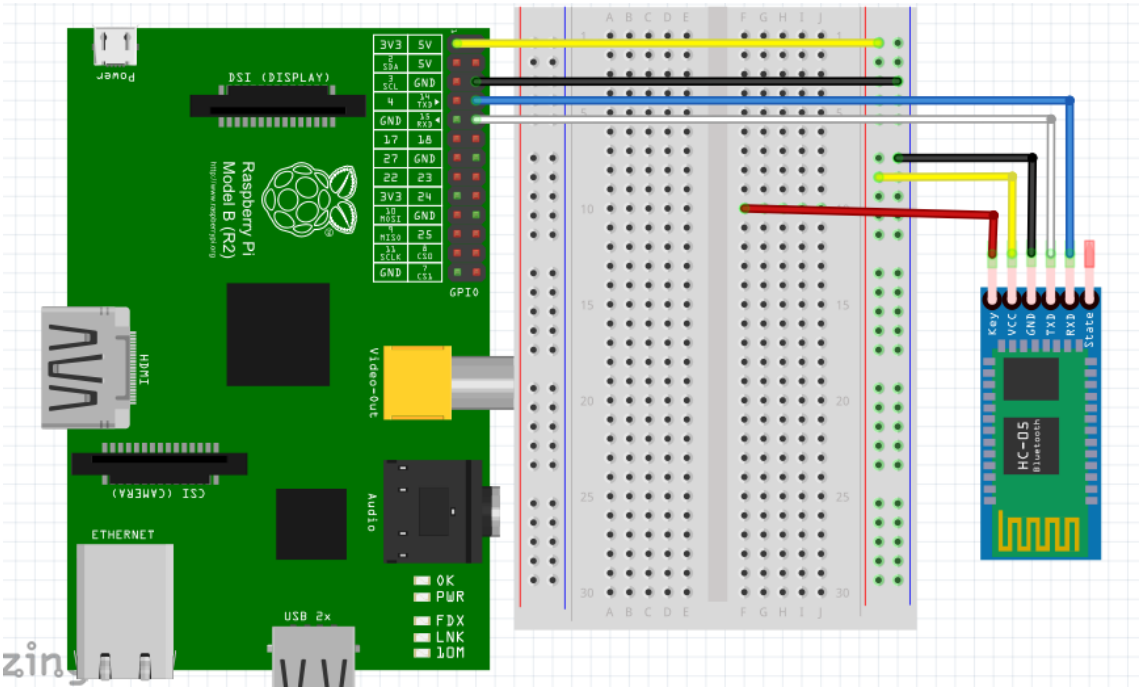
6.1.1. Conexão dos pinos

No modo Passivo, o módulo HC05 apresentará a configuração determinada pelo modo de configuração. Como o módulo trabalhado nunca foi utilizado, as configurações salvas serão as padrão, o que permite utilizá-lo sem maiores problemas. As configurações padrão são:

- ✓ Device type: 0. O tipo dispositivo é 0, ou seja, padrão.
- ✓ Inquire code: 0x009e8b33. Esse código limita ou não o número de respostas de dispositivos Bluetooth quando o módulo está buscando. Esse valor padrão não limita esse número.
- ✓ Module work mode: Slave Mode. O tipo de modo do dispositivo.
- ✓ Connection mode: Connect to the Bluetooth device specified. O modo de conexão Bluetooth é a um dispositivo especificado.
- ✓ Serial parameter: Baud rate: 38400 bits/s; Stop bit: 1 bit; Parity bit: None. Os bits de paridade, de parada e a taxa de transmissão (Baud rate).
- ✓ Passkey: “1234”. Senha, código ou PIN, o qual devemos digitar ao solicitar uma conexão.
- ✓ Device name: “H-C-2010-06-01”. O nome do dispositivo visível para outros.

Agora, será mostrado a conexão na Figura 4. Para representar essa ligação, foi utilizado o aplicativo Fritzing, disponível em <http://fritzing.org/home/>, que além de ser bastante intuitivo, possui uma quantidade imensa de possibilidades para emulação e representação, tanto para raspberry pi como arduino e outros dispositivos.

Figura 4 – A conexão para o modo passivo



Fonte 4 – Fritzing

Para melhorar a visualização, observe a

Tabela 4. Observe, também, que o pino WAKEUP, não foi conectado.

Tabela 4 – Conexão dos fios

COR	RASPBERRY		HC05
	PINO	NUM	
Amarelo	3.3V	1	VCC
Preto	GND	6	GND
Branco	RXD	10	TXD
Azul	TXD	8	RXD
Vermelho	-	-	KEY/WAKEUP

Quando ligado o Led do módulo piscará de forma repetitiva e constante. Isso indica que ele está procurando conexão.

6.1.2. Configuração do RPi

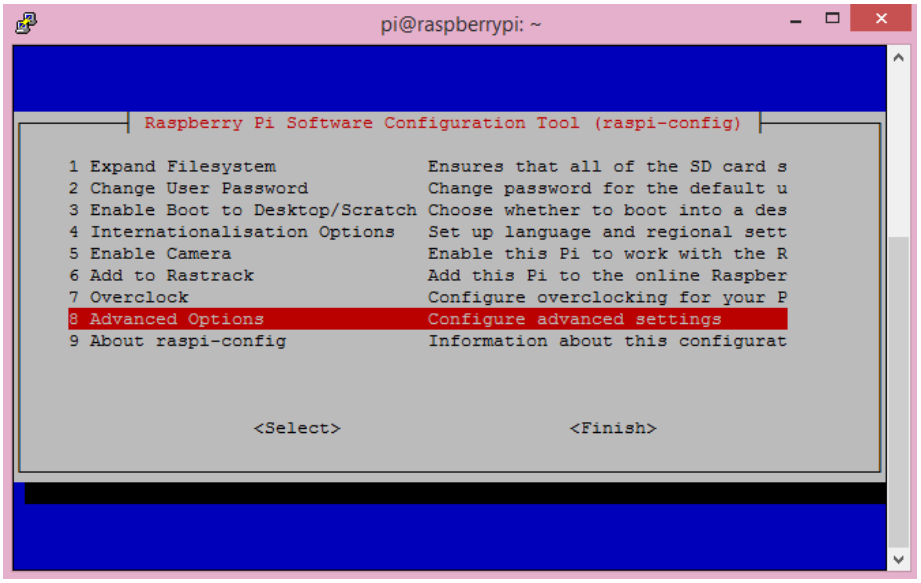
Para utilizar a configuração serial pré-programada do RPi será utilizado o terminal LXTerminal. Isso requer um conhecimento prévio de comandos, mas todos aquele utilizados serão listados.

Primeiramente abra o terminal e digite

```
$ sudo raspi-config
```

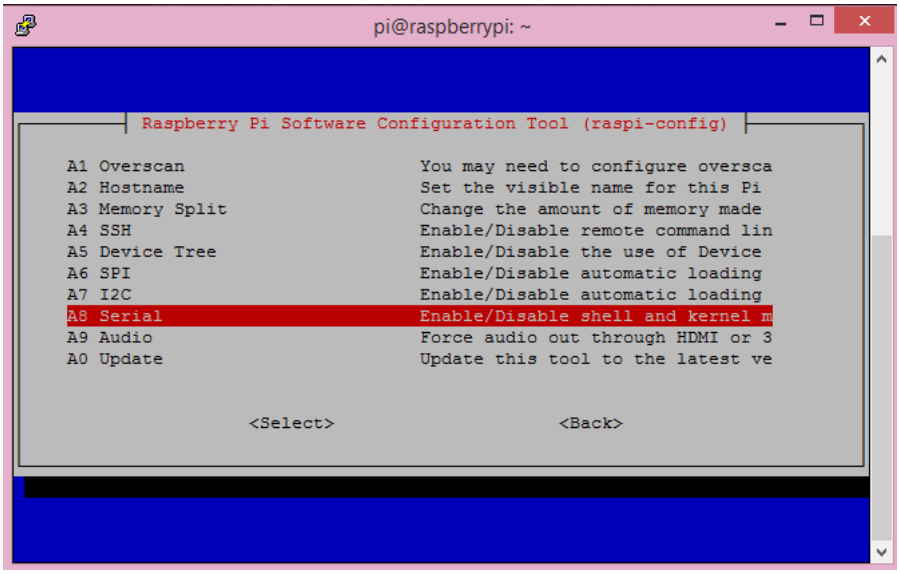
Após a tela de configuração ser aberta, selecione “Advanced Options”, “Serial”, “Yes” e se tudo ocorrer bem a tela da Figura 7 será apresentada.

Figura 5 – Configurado o serial



Fonte 5 – Arquivo pessoal do próprio autor

Figura 6 - Configurado o serial



Fonte 6 - Arquivo pessoal do próprio autor

Figura 7 - Configurado o serial

Fonte 7 - Arquivo pessoal do próprio autor

Logo após habilitar a comunicação serial comandados pela interação entre shell e kernel, voltados a uma experiência mais automática ao usuário, precisamos modificar a velocidade comunicação, ou baud rate, para que seja possível a transmissão ideal entre os dispositivos.

O celular, por exemplo, realiza essa comunicação com uma baud rate de 9600 bits por segundo. Caso usemos uma velocidade de sinalização diferente da apropriada, 384000 para celulares por exemplo, os dados não serão enviados corretamente e ocorrerá percas ou, em casos extremos, nenhuma conexão entre os dispositivos.

Assim, para demonstrar essa aplicação, usaremos um celular. Para isso precisamos instalar algum programa que possibilite comunicação Bluetooth via terminal. Um bastante

Figura 8 – Ícone do aplicativo Bluetooth Terminal

Fonte 8 - Configurado o serial

conhecido é o BLUETERM para o OS Android, mas nesse caso será utilizado o aplicativo BLUETOOH TERMINAL do Windows Store.

Para mudar a Baud rate digite no terminal o comando nano, que abre um editor de texto do raspbian. 4004 0001

```
$ sudo nano /boot/cmdline.txt
```

Na tela que segue, modifique o seguinte texto. Pode ser que seja necessária a primeira modificação em sistemas mais recentes e atualizados.

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

Modifique-o para o texto abaixo e salve o arquivo para concluir a alteração.

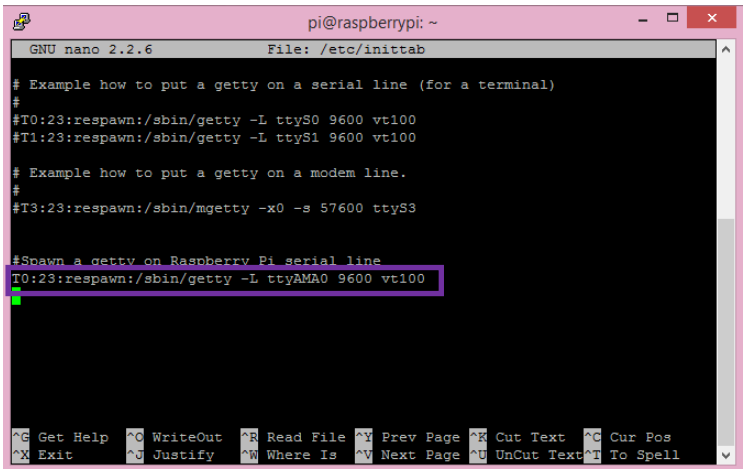
```
dwc_otg.lpm_enable=0 console=ttyAMA0,9600 kgdboc=ttyAMA0,9600 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

Após isso, pode ser necessária mais uma mudança. Para certificar-se disso digite no terminal o comando

```
$ sudo nano /etc/inittab
```

E verifique se a linha demarcada na está correta. Caso esteja diferente, modifique-a para que fiquem iguais.

Figura 9 – Alterando as configurações



Fonte 9 - Arquivo pessoal do próprio autor

6.1.3. A conexão

Agora com o módulo conectado corretamente e com o RPi configurado, conecte o celular ao módulo pelo programa instalado e observe o Led começar a piscar mais espaçadamente, piscando duas vezes sucessivas. Isso indica que ele está conectado.

Nesse caso, o nome do módulo é HC05, mas o seu nome padrão foi informado anteriormente.

Assim, basta reiniciar o raspberry e ver a conexão na prática. O vídeo abaixo mostra esse processo.

VÍDEO PARA SER GRAVADO

Pelo terminal você pode controlar o RPI da forma que desejar, modificando arquivos e fazendo alterações. Entretanto este programa não suporta o servidor gráfico necessário, por isso caso você abra o ambiente ou algum programa gráfico, o terminal ficará inacessível.

6.2. Modo AT ou modo de configuração

Depois de testar a configuração acima é necessário desfazer as alterações feitas para que possamos ter um comunicação serial direta com o módulo.

6.2.1. Conexão dos pinos

A única alteração que deve ser feita a conexão é conectar o pino WAKEUP, que antes estava desconectado, a uma tensão de 3.3v, ou nível alto. Assim, a tabela de conexão ficará:

Tabela 5 – Conexão dos pinos para o modo de configuração

COR	RASPBERRY		HC05
	PINO	NUM	
Amarelo	3.3V	1	VCC
Preto	GND	6	GND
Branco	RXD	10	TXD
Azul	TXD	8	RXD
Vermelho	3.3v	1	KEY/WAKEUP

Após essa mudança, o módulo precisa ser reiniciado. Apenas desplugue e plugue o fio que o alimenta e as mudanças surtirão efeito. Observe que agora o Led presente no módulo está piscando em um padrão diferente, de forma lenta e demorada. Isso indica que ele está em modo AT.

6.2.2. Configuração do RPi

Nesse modo, o módulo poderá ser configurado através dos Comandos AT, que são comando com este propósito de configuração.

Mas, para isso, precisa-se de um terminal no RPi, para que esta comunicação seja feita.

Além disso, é necessário desabilitar a conexão serial. Assim, utilize novamente o comando

```
$ sudo raspi-config
```

E, novamente, vá em Advanced Options > Serial, mas dessa vez selecione NO.

Existem diversos programas que possibilitam a comunicação via terminal. O mais intuitivo e agradável, como o próprio nome diz, é o CUTECOM. Você pode obtê-lo digitando no terminal

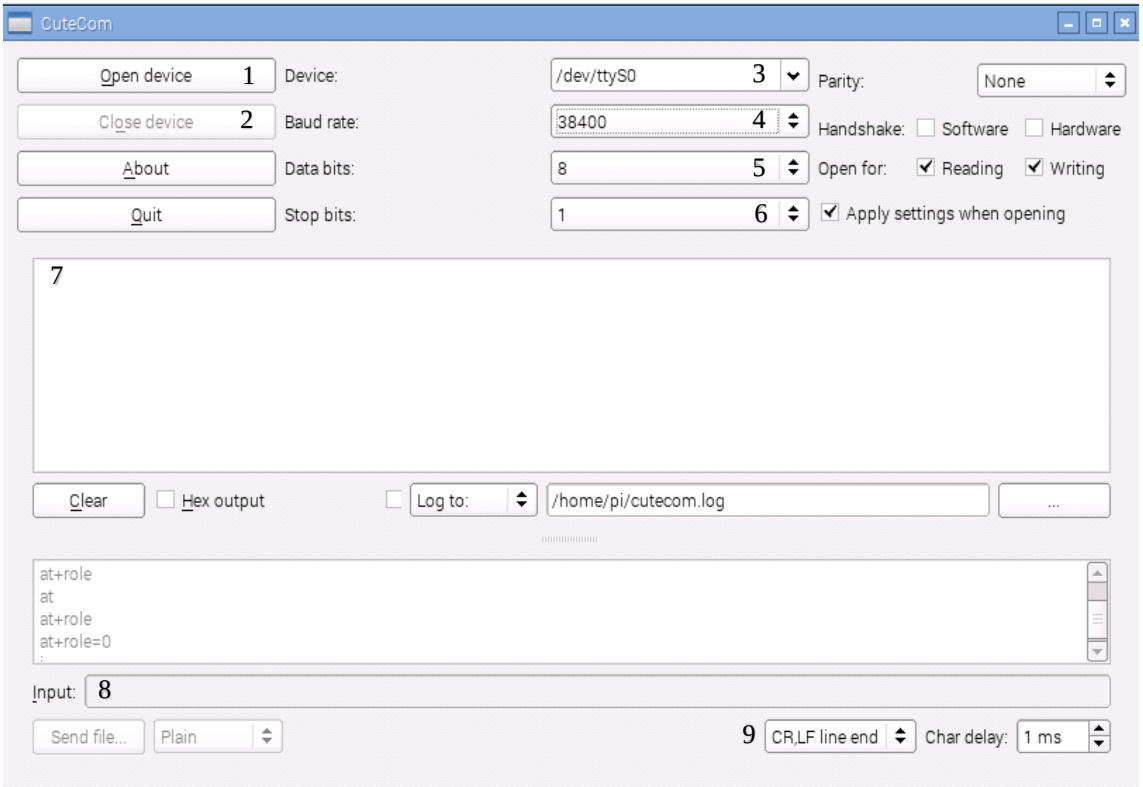
```
$ sudo apt-get install cutecom
```

Espere o término da instalação, ou seja, quando pi@raspberrypi ~ $ aparecer novamente, e abra o cutecom digitando

```
$ sudo cutecom
```

A seguinte tela deverá ser mostrada no seu RPi

Figura 10 – O aplicativo Cutecom



Fonte 10 - Arquivo pessoal do próprio autor

1. Open device: Inicia a conexão com o dispositivo bluetooth. Um celular, por exemplo.
2. Close device: Termina a conexão
3. Device: O tipo de dispositivo que será conectado e a sua porta específica. No caso ttyS0 se refere a uma das portas USB.
4. Baud Rate: Modifica a velocidade de sinalização.
5. Data bits: Número de bits de dados, assim como informados no protocolo de comunicação.
6. Stop bits: Número de bits de parada.
7. Tela Output: Os dados, situação, valores pedidos e recebidos serão apresentados nesta tela
8. Tela de Input: Comandos e requisições serão digitados aqui.
9. Indentação e quebra de linha: Nessa configuração o ENTER significa o fim do comando.

A única alteração que deve ser feita antes de iniciar este modo é o tipo de dispositivo, que no caso é serial. Por isso /dev/ttyS0 deve ser alterado para /dev/ttyAMA0, que representa a comunicação serial.

6.2.3. Códigos AT

Existem um total de 36 códigos AT, porém não é o intuito aqui descrever cada um deles e mostrar todas as suas funcionalidades. Então, será dado o exemplo de uma configuração a partir do zero e a conexão do módulo com algum dispositivo, no caso o mesmo celular usado anteriormente.

7. Códigos AT em uma aplicação: HC05 em modo de configuração

Após certificar-se da conexão dos pinos para configuração do modo AT, deve-se iniciar o programa cutecom com os parâmetros indicados anteriormente.

Para deixar tal exemplo mais didático, primeiramente será informada uma tabela com todos os códigos utilizados. Caso sinta dúvida, volte até a tabela e verifique o código. Também será disponibilizada uma tabela de códigos de erro, esta após o exemplo. Caso o módulo indique algum erro verifique nesta tabela do que se trata.

7.1. Tabela de Comandos

Tabela 6

Código	Parâmetros	Função
AT	Nenhum	Verificar o funcionamento do módulo
AT+ORGL	Nenhum	Limpar as configurações feitas no módulo, recuperando as de fábrica
AT+NAME	<NAME>	Verifica ou seta o nome do módulo para ser reconhecido por outros dispositivos
AT+ADDR	Nenhum	Informa o endereço bluetooth do módulo, em hexadecimal
AT+ROLE	0 = Slave 1 = Master 2 = Slave loop	Define a função desempenhada pelo módulo, ou seja, mestre ou escravo
AT+PWSD	<PWSD>	Verifica ou seta uma senha ou pin padrão para a conexão
AT+UART	<BAUDRATE>, <STOPBITS>, <PARITYBITS>	Define os padrões de comunicação serial entre os dispositivos que farão a conexão.
AT+IAC	9e (PADRÃO)	Define um código de acesso
AT+CMODE	0 = Conectar-se a qualquer endereço 1= Conectar-se a um endereço específico	Limita ou libera a conexão com um ou mais dispositivos
AT+BIND	<ADDR>	Define o endereço ao qual a conexão será obrigatória, caso configurado em AT+CMODE
AT+CLASS	0 (PADRÃO)	Seta o tipo de dispositivo
AT+INIT	Nenhum	Inicia a biblioteca SPP
AT+INQM	<RSSI><NUMDEVICES><TIME>	Busca um número máximo de dispositivos bluetooth, em um tempo limite dado por: <TIME> * 1.28
AT+INQ	Nenhum	Lista os endereços encontrados no comando anterior
AT+FSAD	<ADDR>	Verifica se o endereço está presente na lista de endereços auteticados
AT+PAIR	<ADDR>	Inicia um pareamento com o dispositivo indicado

AT+LINK	<ADDR>	Conecta-se com o dispositivo indicado
AT+DISC	<ADDR>	Desconecta-se de um dispositivo
AT+STATE	Nenhum	Informa o estado do dispositivo no momento

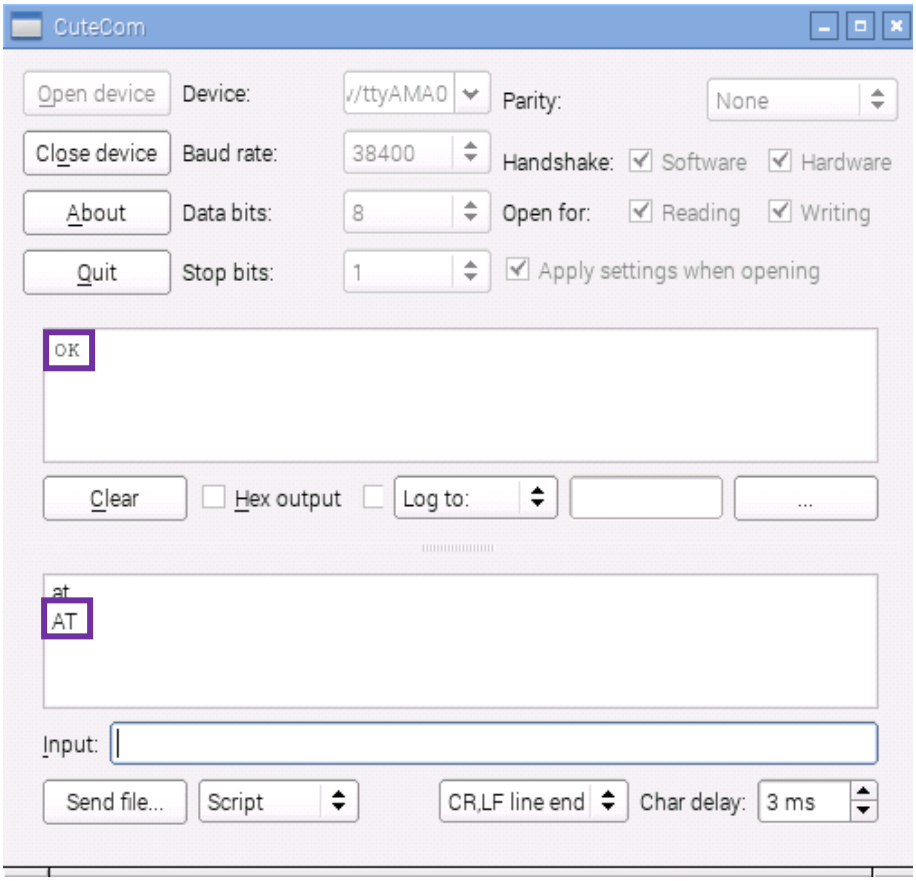
7.2. Utilizando o terminal

1) Verifique se o módulo está funcionando corretamente:

```
AT
```

Caso o módulo responda OK, tudo está funcionando. Caso responda FAIL verifique se há algum problema na conexão.

Figura 11 – Utilizando o aplicativo cutecom



Fonte 11 - Arquivo pessoal do próprio autor

2) Aplique as configurações padrões ao módulo

```
AT+ORGL
```

Aplica-se as configurações padrão para que possa ser feito uma configuração do zero. Novamente, caso o módulo responda OK as alterações foram feitas. Isso é comum a maioria dos comandos, por isso, não será mais comentado para evitar redundância.

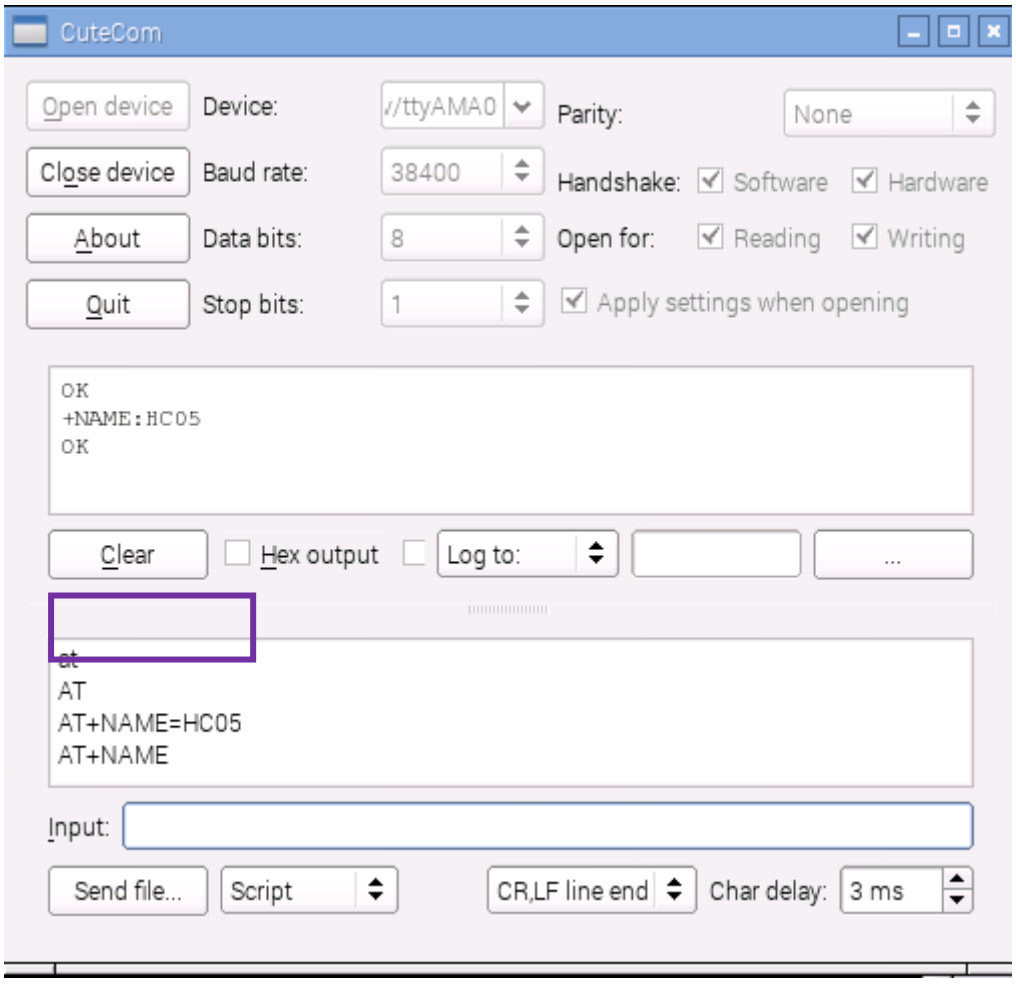
3) Verifique o nome do módulo e logo após o altere

```
AT+NAME

AT+NAME=HC05
```

O nome padrão do módulo será mostrado. A segunda linha de comando irá dar ao módulo o nome HC05. Caso queira, digite novamente o comando AT+NAME para verificar o novo nome.

Figura 12 – Alterando as configurações do módulo



Fonte 12 - Arquivo pessoal do próprio autor

4) Verifique e modifique a senha de acesso

```
AT+PWSD

AT+PWSD=1122
```

Verifique e sete uma nova senha. Nesse caso a nova senha será 1122.

5) Modifique a função do módulo

```
AT+ROLE=1
```

Configure o módulo para desempenhar a função de Mestre. Na tabela as outras funções disponíveis já foram listadas.

6) Configure a comunicação serial

```
AT+UART=9600,0,0
```

Configure a comunicação serial. Nesse caso o Baud Rate será de 9600bps, terá 1 bit de parada e nenhum bit de paridade.

7) Configure a prioridade de conexão

```
AT+CMODE=1
```

Nesse caso, o módulo só poderá se conectar a um endereço específico. Esse endereço será informado no comando AT+BIND

8) Defina o código de acesso Padrão

```
AT+IAC=9e8b33
```

Esse código de acesso é padrão. O único intuito desse comando é certificar-se de que o número de respostas de dispositivos bluetooth quando o módulo procurá-los será, teoricamente, ilimitado.

9) Defina o tipo de dispositivo

```
AT+CLASS=0
```

Esse código apenas indica o tipo de dispositivo que será utilizado. É um código padrão, apenas para que se verifique as configurações.

10) Inicie a biblioteca SPP

```
AT+INIT
```

Esse comando inicia a biblioteca SPP (Serial Port Profile). Que informa, basicamente, como estabelecer uma conexão entre dois dispositivos e criar portas seriais virtuais.

11) Encontre o endereço do dispositivo Bluetooth desejado

Para isso, primeiramente certifique-se de que apenas um dispositivo bluetooth está próximo do módulo. Caso isso não seja possível, há outros meios de encontrar esse endereço, como verificando-o nas configurações bluetooth do Windows ao conectar-se a ele pelo computador, ou utilizar um aplicativo específico para celular, como o SPPBLUE para Windows Phone.

```
AT+INQM=1,5,10
```

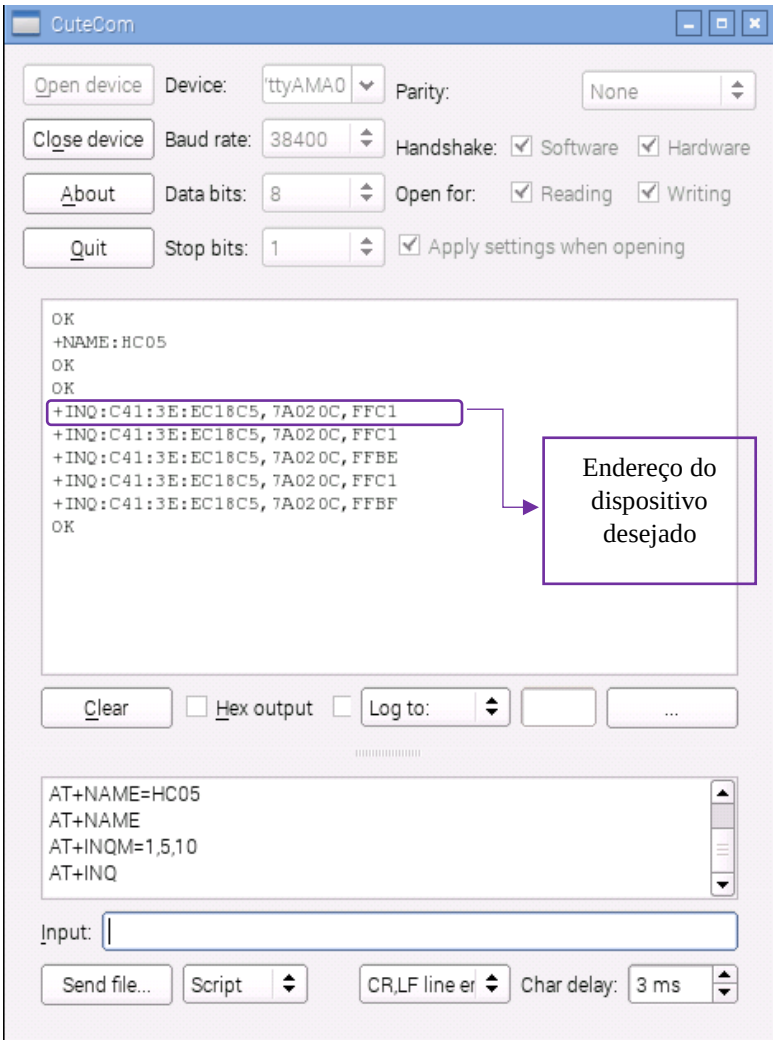
Esse comando informam o RSSI (Received Signal Strenght Indication) que é configurado como padrão pelo primeiro parâmetro (1), o número máximo de dispositivos que podem ser encontrados e o tempo máximo de procura, dado pelo terceiro parâmetro multiplicado por 1,28. Nesse caso será de 10*1,28.

12) Verifique os dispositivos encontrados

```
AT+INQ
```

Informa os endereços encontrados na procura. O endereço é dado em Hexadecimal, e deve-se, ao escrever um endereço, substituir dois pontos (:), por vírgula (,).

Figura 13 – Procurando o endereço do dispositivo



Fonte 13 - Arquivo pessoal do próprio autor

13) Restrinja a conexão ao endereço de preferência

```
AT+BIND=C41,3E,EC18C5
```

Restringe a conexão ao endereço indicado. Esse comando atua junto ao comando ATT+CMODE.

14) Inicie o pareamento ao dispositivo

```
AT+PAIR= C41,3E,EC18C5
```

Inicia o pareamento ao dispositivo. Será preciso digitar o código de acesso no dispositivo que deseja-se conectar, que, nesse caso, será 1122.

15) Conecte-se ao dispositivo

```
AT+LINK= C41,3E,EC18C5
```

Agora estabeleça uma conexão ao dispositivo.

É interessante o fato de que, caso você configure o módulo deste modo, desligue-o e o religue no modo passivo, ou seja, com WAKEUP em nível baixo, ele tentará se conectar ao dispositivo do endereço informado em AT+BIND automaticamente, isso é muito útil para conexões que precisam ser feitas toda vez que o RPi for ligado.

8. Um exemplo com Python

Agora, o módulo HC05 já pode ser configurado da forma desejada. Por isso, será feito um exemplo utilizando-o em modo slave e com um código em python.

A ideia é digitar um número entre 0 e 7 no terminal bluetooth do celular e, este, ser convertido em binário para então ser mostrado na protoboard através de led. Por exemplo, digitando o número 3 no celular dois leds serão acesos na protoboard, que representam o último e penúltimo algarismos menos significativos dos números binários.

8.1. A Configuração dos fios

Primeiramente, será necessário resetar o módulo bluetooth através do comando AT+ORGL. Caso não queira resetá-lo, apenas mude as configurações necessárias para que o mesmo seja configurado no modo SLAVE: AT+ROLE=0.

Pretende-se ser capaz de representar a faixa entre 0 e 7, assim serão necessários 3 leds. Além disso, será utilizado um led para representar um número inválido, ou seja, um que não esteja dentro da faixa previamente determinada.

A representação das ligações pode ser vista na figura a seguir produzida pelo aplicativo Fritizing.

REPRESENTAÇÂO DA CONFIGURAÇÃO

8.2. A biblioteca serial

No código será necessário importar duas bibliotecas, pois para que seja possível capturar e enviar informações do celular para o RPi e vice-versa via terminal bluetooth, será necessário utilizar uma biblioteca que possa fazer essa função pela comunicação serial. Assim, a biblioteca serial será utilizada.

Além disso, o número binário precisa ser externado através dos leds. Para isso, é necessário utilizar a biblioteca GPIO, para os pinos de entrada e saída de dados.

A biblioteca serial possui algumas funções que serão de grande ajuda:

- ✓ Serial.write(“”) – Escreve algo na porta serial. Nesse caso será enviado uma string e será preciso enviar um \r no final da mensagem, pois representará uma quebra de linha.
- ✓ Serial.read(bits) – Lê informação da porta serial do tamanho informado em bits no argumento.
- ✓ Serial.serial() - Abre uma porta serial com o formato desejado, baud rate e tempo de espera (timeout).
- ✓ Serial.flushInput() – Limpa o buffer do teclado de entrada, o que impede leitura de informação não necessária

Agora podemos passar para o código em si.

8.3. O código em Python

Observe o código a seguir:

```
import serial                                   1
import RPi.GPIO as GPIO
from time import sleep

GPIO.setwarnings(False)                         2

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11,GPIO.OUT)                         3
GPIO.setup(15,GPIO.OUT)
GPIO.setup(29,GPIO.OUT)
GPIO.setup(31,GPIO.OUT)
ser=serial.Serial('/dev/ttyAMA0',38400,timeout=5)  4
```

1. Primeiramente, serão importadas as bibliotecas utilizadas.
2. Depois, os avisos serão desativados, para evitar a advertência da utilização da mesma saída em diferentes execuções dos programas
3. Assim, serão configuradas as saídas para os leds.
4. E nessa linha, uma porta serial será aberta com baud rate de 38400 e timeout (tempo de espera) de 5 segundos.

```
while(1):


    ser.write("Compilando...\r")
    ser.flushInput()
    entrada=ser.read(2)
    print entrada + 'foi digitado'
    if entrada == 'cm':
        ser.flushInput()
        ser.write("Digite o numero decimal\r")
        dec=ser.read(1)
        num=int(".join(str(ord(c)) for c in dec))- 48
        if(num> 7):
            ser.write("Numero nao suportado\r")
            GPIO.output(11,1)
            GPIO.output(15,0)
            GPIO.output(29,0)
            GPIO.output(31,0)
        else:
            x=0
            binario=[0,0,0]
            GPIO.output(11,False)
            while( num >0):
                if(num<=1):
                    binario[x]=1
                    x=x+1
                    break
                else:
                    binario[x]=num%2
                    num=num/2
                    x=x+1
            GPIO.output(15,binario[2])
            GPIO.output(29,binario[1])
            GPIO.output(31,binario[0])
```

Nessa parte do código, teremos um laço para rodar o programa constantemente (while(1)). Após isso, será escrito na tela do terminal “Compilando...” para que saibamos que o programa está rodando corretamente, e ter noção do início de cada ciclo.

Depois será lido uma informação de dois bits do terminal (ser.read(2)), essa informação será comparada no “if” e caso seja cm, dará início a parte interessante do código.

Assim, o resto do código é apenas capturar o número decimal e transformá-lo em binário através de divisões sucessivas pelo número 2. Para isso é necessário transformar o char digitado pelo terminal em int, utilizando o trecho de código int(''.join(str(ord(c)) for c in dec)), que transforma o caractere digitado em um número da tabela ASCII, de forma que para que saibamos o número realmente digitado devemos subtraí-lo de 48.

Pronto, agora basta expressar esse número na protoboard setando as saídas em alto e baixo, o que ligará os leds da forma desejada.