Usando o APP "Serial Bluetooth Terminal" no controle de um Sistema baseado no ESP32 e Plataforma Arduino, através do PCOM WiFi/Bluetooth.

revisão-00, agosto de 2019

Elcids H. das Chagas

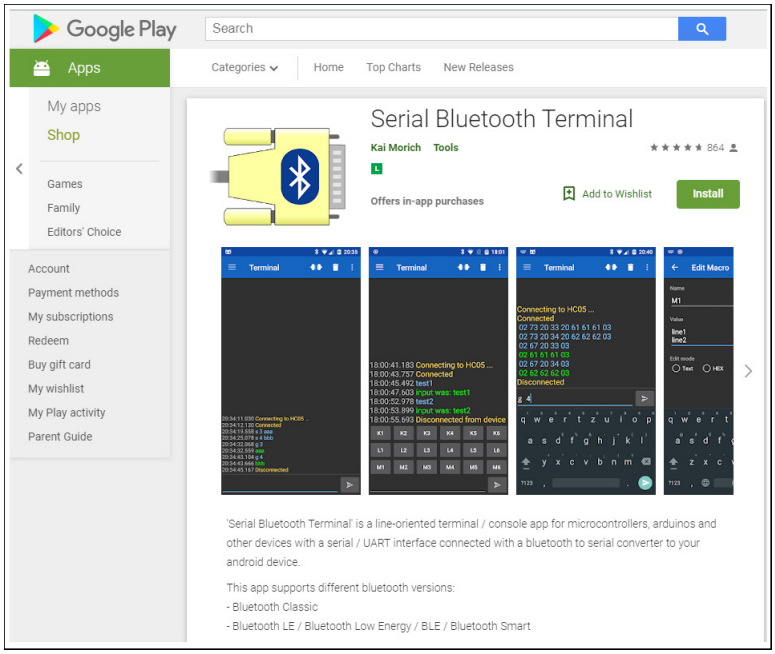
Obs.:

- ***este documento foi escrito considerando a versão 1.24 do APP "Serial Bluetooth Terminal".***

Configuração do APP "Serial Bluetooth Terminal":

Os códigos escritos no ***Arduino*** para ***placas ESP32*** usando o ***PCOM WiFi/Bluetooth***, podem usar para testes de controle via Bluetooth, um ***APP*** de ***Terminal Serial*** sendo executado em um ***Smartphone*** ou ***Tablet***. Se a aparência da Interface não for um fator primordial, um APP deste tipo pode até mesmo ser usado para o controle final, já que é possível fazer este controle através do envio de caracteres simples que correspondam aos comandos implementados no código do Arduino. A única exigência, é que o APP implemente um Terminal "passivo" através de uma ***Interface Bluetooth SPP*** (Serial Port Profile), que essencialmente é o envio e a recepção de ***caracteres ASCII*** através do ***Bluetooth "Classic"***.

Um dos APPs que podem ser usados, é o "***Serial Bluetooth Terminal***", que reúne todos os requerimentos para o controle via Interface SPP. Este APP é de autoria de ***Kai Morich***, e está disponível gratuitamente no ***Google Play***, conforme pode ser visto na figura a seguir:



No momento que este texto é escrito, o link da página do APP é este:

https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal&hl=en

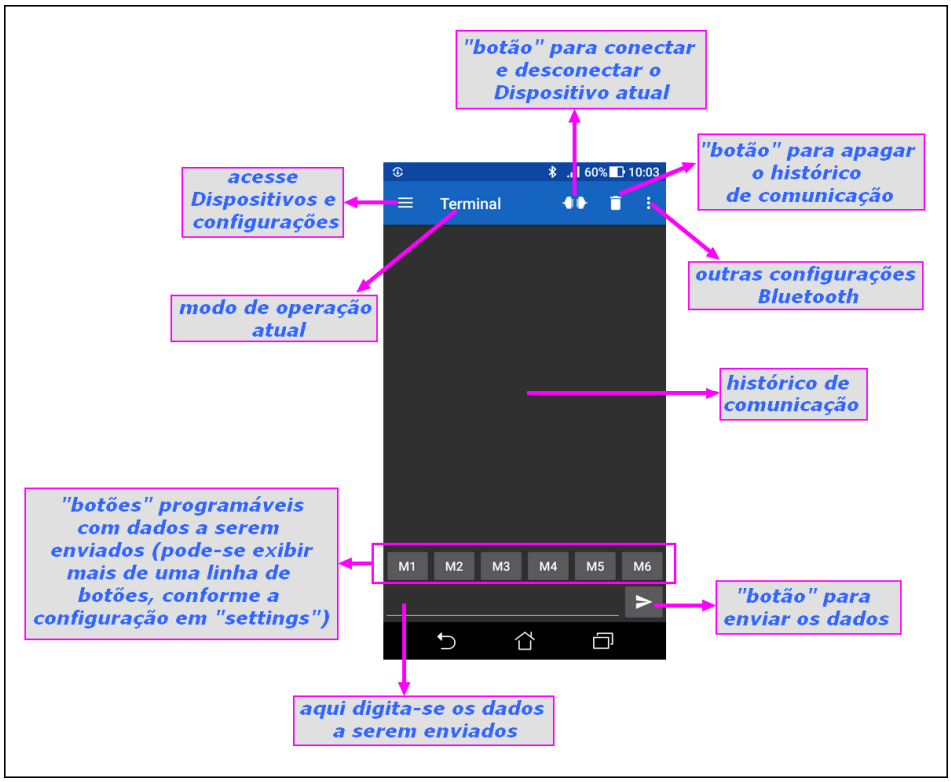
Para o controle via ***WiFi***, o ***PCOM WiFi/Bluetooth*** tem suporte para gerenciar facilmente ***Páginas HTML*** para a ***Interface de Controle***. Também é possível utilizar qualquer Protocolo baseado em ***TCP/IP***, seja um protocolo padrão ou um dedicado desenvolvido pelo usuário. Mas este controle via WiFi não é o objeto deste texto (e será posteriormente abordado em outro documento).

O texto desse documento descreve como configurar e utilizar o ***APP*** "***Serial Bluetooth Terminal***" para controlar um Sistema baseado no ***ESP32*** na ***Plataforma Arduino***, com código usando o ***PCOM WiFi/Bluetooth***.

Obs.: no momento que este texto é escrito, não há propagandas no ***APP*** "***Serial Bluetooth Terminal***", sendo louvável a atitude do autor ***Kai Morich***. Então, fazendo uma propaganda merecida: quem tiver condições, entre no site do autor do APP e faça uma doação, em reconhecimento ao grande trabalho que ele disponibilizou gratuitamente.

Apresentação Geral do APP "Serial Bluetooth Terminal":

Após a instalação do APP, ao abri-lo sua aparência é a mostrada na figura a seguir, onde também estão identificados os vários elementos que compõe a ***Interface do APP*** com o usuário:



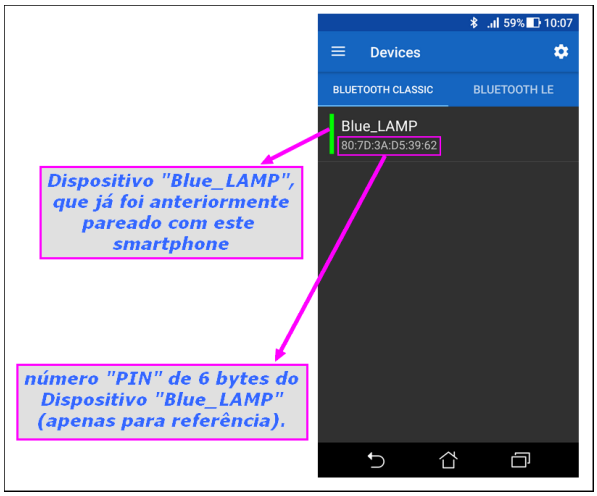
A figura anterior é "auto-explicativa", dispensando descrições adicionais.

Mesmo assim, observe o símbolo que aparece para o "botão" para conectar e desconectar o ***Dispositivo Bluetooth*** atualmente selecionado. Na figura ele indica que não há Dispositivo conectado. Atente em algumas figuras exibidas mais à frente, onde o símbolo do botão mudará quando houver um Dispositivo Bluetooth conectado, indicando assim esta condição.

Ao se clicar no ícone para acessar ***Dispositivos e Configurações***, é visualizada a lista de opções, conforme mostrado na figura a seguir:

A opção "***Terminal***", simplesmente retorna à tela anterior do APP, a qual corresponde à "janela" do Terminal (onde se envie e se recebe dados via ***SPP Bluettoh***). A opção "***Info***" exibe informações sobre o APP e links relacionados.

A opção "***Devices***", exibe a lista de Dispositivos Bluetooth que estejam no alcance atualmente, sejam pareados ou não. Para exibir os que não estão pareados, deve-se usar o "scan" (deslize a tela para a esquerda, para depois da tela "Bluetooth LE"). Também pode-se fazer o pareamento antes de se abrir o APP, acessando as opções do Bluetooth no Smartphone ou Tablet, e neste caso quando se abrir o APP, o Dispositivo pareado já aparecerá na lista do APP. A figura a seguir mostra o ***Dispositivo*** "***Blue_LAMP***", que está implementado no exemplo aqui usado para demonstrar o controle através do APP:



Notar que é exibido também o ***número*** "***PIN***" do Dispositivo Bluetooth. O "PIN" é a identidade do Dispositivo, consistindo de *6 bytes* (ou seja, *48 bits*).

Os Smartphones e Tablets, também tem seu próprio número "PIN" (assim como o Bluetooth do Computador), que identifica estes ***Clientes Bluetooth***. No código do Arduino, a ***Interface*** do ***PCOM WiFi/Bluetooth*** permite se obter o "PIN" do Cliente atualmente conectado ao Bluetooth do ESP32 (o "PIN" é representado na forma de um vetor de 6 bytes). Então isto abre uma gama de possibilidades de aplicações, já que é possível identificar cada Cliente Bluetooth conectado ao Arduino (e portanto permite que se dê um tratamento diferenciado a cada Cliente Bluetooth, se isto for desejado).

"Modos de Comando" para a Autenticação no PCOM Bluetooth:

O ***PCOM*** permite duas formas de fazer a ***Autenticação Bluetooth***, as quais são chamadas de "***Modo de Comando simples***" e "***Modo de Comando texto***". A seguir há uma descrição destes dois modos de comando:

1) No "***Modo Simples***", os comandos de Autenticação se iniciam com "***#***" e são constituídos por sequências simples de ***caracteres ASCII***. Os comandos sempre terminam com a sequência "*Carriage Return*" e "*New Line*", ou seja, em hexadecimal os caracteres ASCII "não-imprimíveis" ***0x0D*** e ***0x0A***, que na Linguagem C/C++ são também chamados de ***'\r'*** e ***'\n'***. O "modo simples" é indicado quando os comandos serão interpretados por outro programa ou uma máquina similar, pois este formato simplifica o processo de decodificação.

Exemplo: ***#TA*** , que significa "***Timeout de Autenticação***".

2) No "***Modo Texto***", os comandos de Autenticação se iniciam com "***$***" e são constituídos por palavras/textos "legíveis" de ***caracteres ACII***. Os comandos sempre terminam com a sequência "*Carriage Return*" e "*New Line*", ou seja, em hexadecimal os caracteres ASCII "não-imprimíveis" ***0x0D*** e ***0x0A***, que na linguagem C/C++ são também chamados de ***'\r'*** e ***'\n'***. O "modo Texto" é indicado quando os comandos serão interpretados por pessoas, ou seja, quando os caracteres do comando são exibidos em algo semelhante a uma caixa de diálogo ou Terminal de Texto.

Exemplo: ***$ terminou tempo!!!*** , indicando "***Timeout de Autenticação***".

Portanto, ao se escrever o ***código*** no ***Arduino***, é preciso que se saiba antes, qual "***Modo de Comando***" será usado para se fazer a ***Autenticação Bluetooth***. Esta característica seria normalmente fixa no código. Mas há diversas formas de se flexibilizar isso, como por exemplo ter uma chave no hardware (ou apenas um "jumper" físico), que estando em "High" ou "Low" permite ao código no Arduino selecionar um dos dois Modos de Comando, inclusive "on the fly".

O ***PCOM Bluetooth***, salva o "***PIN***" do ***Cliente*** que fez a Autenticação, em uma ***Lista de Clientes Autenticados***. Esta Lista é salva na ***EEPROM*** do ***ESP32***, de forma que mesmo após uma reinicialização, o Sistema sempre tem a informação de quais Clientes já fizeram a Autenticação. Assim, um Cliente que já fez a Autenticação (forneceu a senha), não precisará fazê-la novamente.

O PCOM permite ao código no Arduino "*resetar*" a qualquer momento, a Lista de Clientes Autenticados (ou seja, apague a Lista atual, "esvaziando" a mesma). A Lista também tem mecanismos de validação do seu conteúdo (via ***CRC***), para detectar qualquer informação que possa ter sido corrompida.

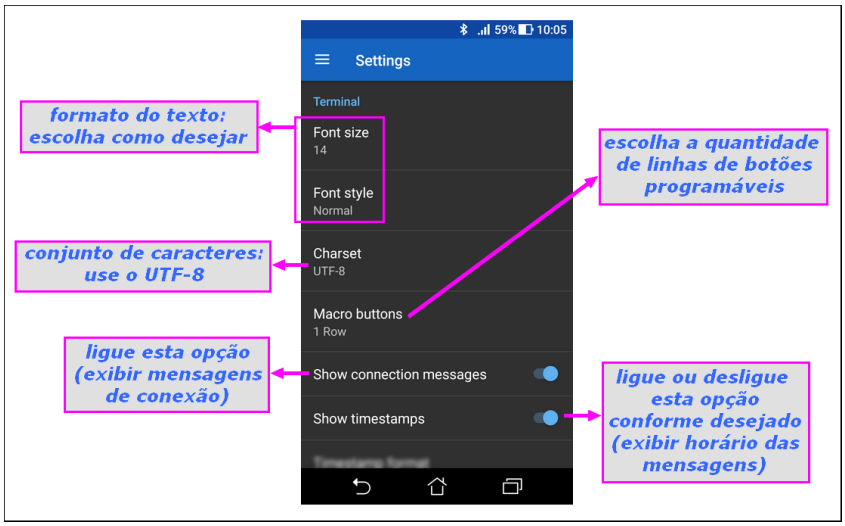
Parece bastante óbvio que quando se utiliza o ***APP*** "***Serial Bluetooth Terminal***", o processo de ***Autenticação Bluetooth*** usando o "***Modo de Comando texto***" é o que deve ser usado, já que na tela no APP iremos ver mensagens indicando todo o processo, que será sempre visualizado e interpretado por quem estiver iniciando a conexão.

Já o "***Modo de Comando simples***", é o indicado quando se utiliza um programa para fazer automaticamente a Autenticação. Um uso claro disso, é quando se desenvolve um APP dedicado, pois se pode instruir o APP a fazer a Autenticação "sozinho" após a conexão Bluetooth for iniciada. Um exemplo disso poderá ser desenvolvido em breve.

Configurações Gerais do APP:

Como descrito anteriormente, o "***Modo de Comando texto***" é o indicado quando se usa o ***APP*** "***Serial Bluetooth Terminal***". Assim, este APP deve ser configurado para que se possa facilmente fazer a Autenticação através da tela do Terminal do APP.

Para fazer a configuração do APP, selecione a opção "***Settings***" através do ícone ***Dispositivos e Configurações***. O início da tela com a lista de configurações do APP, é mostrada na figura a seguir:



A figura anterior é "auto-explicativa", dispensando descrições adicionais. Observar que alguns itens se configura conforme desejado. Mas há itens que devem ser configurados como mostrado. Esta regra vale para todas as demais figuras que mostram as opções de configuração.

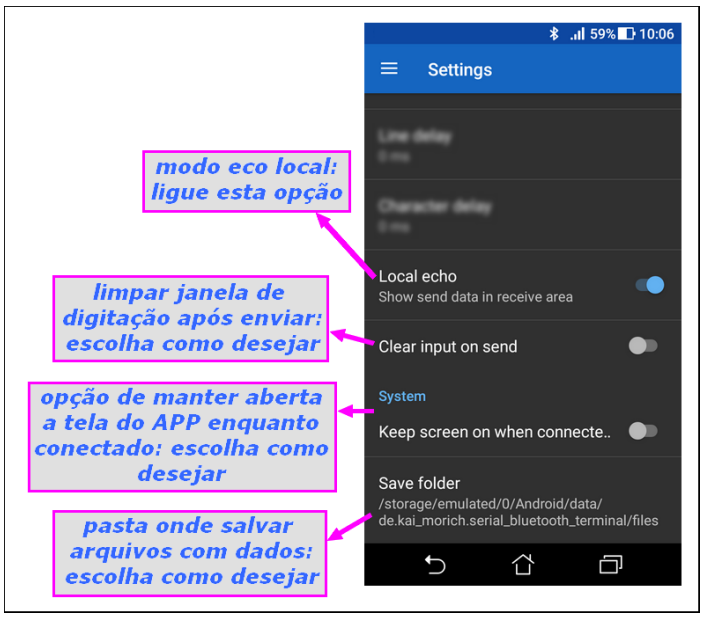
Deslize a tela para ver os demais itens a serem configurados.

A figura seguinte mostra duas configurações importantes: o ***modo de recepção*** (em "***Receive Newline***") e o ***modo de operação*** (em "***Display mode***"). Configure-os como mostrado:

Seguindo na próxima figura, há mais configurações importantes, que são o ***modo de envio*** (em "***Send Newline***"), e o ***modo de edição*** (em "***Edit mode***"). Configure-os como mostrado:

Embora seja possível, *evite* alterar o "*Line delay*" e o "*Character delay*".

Finalmente, a próxima figura mostra os últimos itens a serem configurados. O "***Local echo***" e o "***Clear input on send***", devem ser configurados como mostrado:



Já o "***Keep screen on...***", ligue ou desligue conforme achar conveniente.

É aconselhável não alterar o "***Save folder***", se você não tem intimidade em manusear arquivos no ***Sistema Android*** (esta configuração também só tem sentido se você for salvar algo em algum momento).

Uso do APP para Controle via Arduino e PCOM Bluetooth:

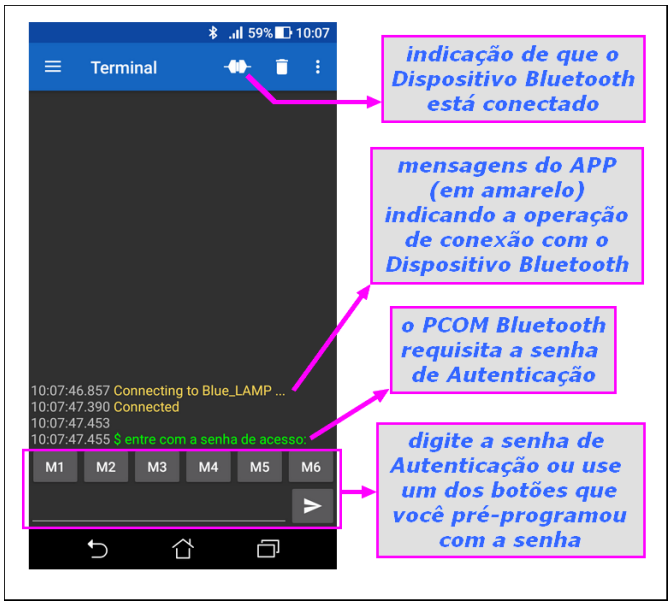
O uso do **APP** "***Serial Bluetooth Terminal***" para controle de um ***Arduino/ESP32*** executando o ***PCOM Bluetooth***, é bastante simples.

Mas é importante também lembrar que o código de controle executando no ESP32 e na Plataforma Arduino, deve estar preferencialmente usando o "***Modo de Comando texto***", que é o modo de comando mais adequado para a Interface com pessoas através de um diálogo tipo ***Terminal de Texto*** "***passivo***" (um terminal tipo "*passivo*" é aquele que apenas exibe os caracteres recebidos e envia os caracteres digitados, sem interpretar estes caracteres).

Após ter configurado o APP conforme descrito anteriormente, faça o pareamento com o ***Dispositivo Bluetooth*** que corresponde ao Sistema ***ESP32/Arduino*** (neste exemplo estaremos usando o "***Blue_LAMP***"). No pareamento, não será solicitada senha, pois até o momento a ***Espressif*** não implementou este recurso no ***ESP32*** para a ***Plataforma Arduino***.

Estar pareado, não significa estar conectado. Significa que os dois Dispositivos (***ESP32/Arduino*** e por exemplo um ***Smatphone***) concordaram se comunicar um com o outro.

Para conectar ao ***Dispositivo*** "***Blue_LAMP***" (que estaremos usando como exemplo), é preciso clicar no ***botão conectar/desconectar*** do APP. Uma vez conectado, a aparência do botão muda, conforme pode ser visto na figura a seguir:



Observe que as mensagens na cor amarela, são do próprio APP. Logo que a conexão é feita, o ***Arduino*** (***ESP32***) envia um "texto de comando" (começando com o caractere "***$***") solicitando a ***senha de Autenticação***.

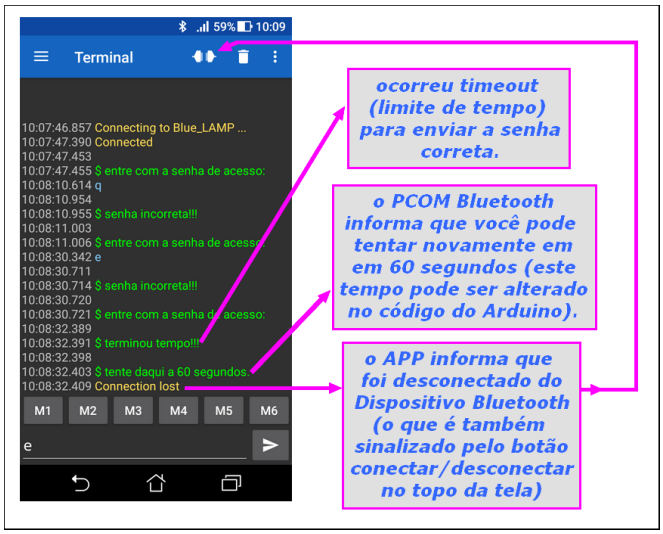
Como dito anteriormente, a senha será solicitada apenas na primeira conexão do ***Cliente Bluetooth*** (na figura este "Cliente" é um *Smartphone*), já que depois de Autenticado o Cliente é inserido em uma ***Lista de Clientes Autenticados***, dispensando Autenticações posteriores deste Cliente (a não ser que a Lista seja intencionalmente "resetada" pelo código executando no Arduino).

O ***número de tentativas*** para inserir a senha correta, é especificado no código do Arduino (e pode ser alterado "*on the fly*", ou seja, a qualquer momento). Ao ser excedido este número de tentativas, o ***Processo de Autenticação*** é interrompido e será enviada uma mensagem ao Cliente informando que ele deve esperar um intervalo de tempo antes de iniciar um novo Processo de Autenticação. Este intervalo de tempo é chamado de "***Intervalo de Autenticação***", sendo especificado no código do Arduino (pode ser alterado "*on the fly*").

Para cada uma das tentativas de inserção de senha, o ***PCOM Bluetooth*** aguarda um ***tempo mínimo*** de ***15 segundos***. Assim se foram especificadas até 3 tentativas de inserção, o Cliente tem até 45 segundos para fazer as 3 tentativas. Após este período de tempo que o Cliente tem para fazer as

tentativas, ocorre o que é chamado de "***Timeout de Autenticação***", o que significa que terminou o tempo que o Cliente tinha. Assim, independente do que ocorrer primeiro, seja exceder o limite de tentativas ou exceder o tempo do "Timeout de Autenticação", o Processo de Autenticação será interrompido e será exibida a mensagem para o Cliente esperar até que possa iniciar uma nova tentativa.

Nas duas próximas figuras, onde foram inseridas senhas incorretas, ocorreu o "Timeout de Autenticação":

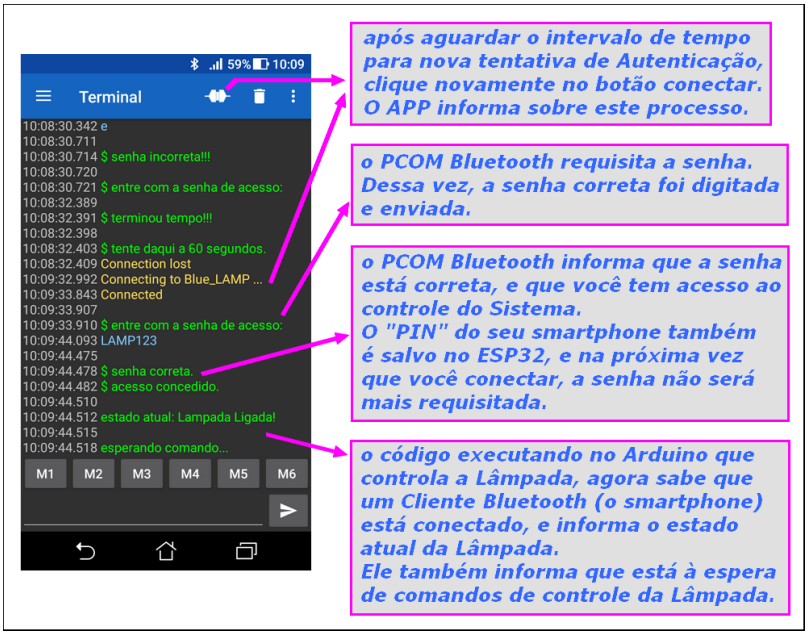


Como já dito, o ***PCOM Bluetooth*** considera um tempo mínimo de 15 segundos para cada tentativa de inserção de senha. Mas este período pode ser aumentado, pois o código no Arduino pode especificar o tempo do "***Timeout de Autenticação***" (a especificação pode ser "*on the fly*"). Esta especificação será ignorada pelo PCOM Bluetooth se o tempo especificado for menor que os 15 segundos por tentativa de inserção de senha.

Observe que para inserir a senha, pode-se digitá-la caractere por caractere, ou então usar um dos botões na base da tela, uma vez que estes botões podem ser pré-programados (neste caso, basta programar o botão desejado, com o texto completo da senha). Para programar um botão, segure o mesmo até que se abra a tela de programação.

Quando terminam as tentativas de inserção de senha ou ocorre o "Timeout de Autenticação", o ***PCOM Bluetooth*** desconecta o Cliente Bluetooth *automaticamente*. Então é preciso que o Cliente Bluetooth se reconecte novamente para iniciar novo Processo de Autenticação, mas apenas depois de esperar que tenha passado o tempo do "Intervalo de Autenticação". Somente após isso, é que será solicitado novamente que se insira a senha de Autenticação do Sistema.

Isto pode ser visto na figura a seguir, onde após ocorrer o "Timeout de Autenticação" e tendo sido aguardado o "Intervalo de Autenticação", a senha correta é inserida:



Então após isso, o Cliente está Autenticado no Sistema (tendo sido salvo na ***Lista de Clientes Autenticados***), e tem acesso para controlar este Sistema, conforme permitir o código executando no Arduino.

Na figura anterior, veja que a mensagem informando o ***estado da Lâmpada***, assim como a mensagem dizendo que está "***esperando comando...***", são do ***código executando no Arduino***, pois neste ponto o ***PCOM Bluetooth*** não mais interfere no processo (ele apenas é usado como ***Port de Comunicação*** entre o ***Cliente Bluetooth*** e o código executando no Arduino).

Então depois que o Cliente conectou e Autenticou, o código executando no Arduino tem a liberdade para usar qualquer forma de controle que ache conveniente, e pode implementar seu próprio protocolo (que ainda estará dentro da ***Interface SPP*** do ***Bluetooth***, mas para o código executando no Arduino e para o Cliente isto será transparente, e eles não precisarão se preocupar com isso).

No caso desse exemplo do ***Dispositivo*** "***Blue_LAMP***", que apenas controla o ***liga/desliga*** de uma ***Lâmpada***, foi utilizado um comando simples, usando o ***caractere*** "***L***" (o código no Arduino pode aceitar maiúsculo e/ou minúsculo mas à critério dele, já que o ***PCOM Bluetooth*** não interfere mais no processo). Cada vez que o "***L***" é enviado, o código no Arduino inverte o estado da Lâmpada (esta "inversão" de estado é tecnicamente chamada de "***toggle***"). E para que o Cliente saiba o que está ocorrendo, o código no Arduino envia uma mensagem informando como ficou o estado da Lâmpada, após executar o comando. Isto pode ser visto na figura a seguir:

Pode-se usar um dos ***botões pré-programáveis*** para enviar o comando de controle (na figura anterior a letra "***L***" foi programada no botão "***M1***"). Assim, caso se implemente no código do Arduino vários comandos, cada um pode ser pré-programado em um botão (e pode-se ter outras linhas de botões, além daquela mostrada na figura). Claro, se o número de comandos for muito grande, talvez seja melhor construir um APP dedicado, onde o nome dos comandos aparece escrito nos botões.

Na próxima figura, uma desconexão é feita (através do botão no topo da tela), e logo em seguida é feita uma reconexão. Observe que desta vez a Autenticação não é necessária:

No ***código do Arduino*** (***ESP32***), as possibilidades são muitas. Pode-se implementar praticamente o que vier à mente. E se for desenvolvido um APP dedicado, então estas possibilidades vão ainda mais além.

AFA

(Arduino for all)