Vitor Amadeu Souza

Desenvolvimento de Projetos com

ESP32

Programado em Arduino – Volume único

© 2020 by Cerne Tecnologia e Treinamento Ltda.
© 2029 by Vitor Amadeu Souza
Nenhuma parte desta publicação poderá ser reproduzida sem autorização prévia e escrita de **Cerne Tecnologia e Treinamento Ltda.** Este livro publica nomes comerciais e marcas registradas de produtos pertencentes a diversas companhias. O editor utiliza as marcas somente para fins editoriais e em benefício dos proprietários das marcas, sem nenhuma intenção de atingir seus direitos.

Abril de 2020
Direitos reservados por:

Cerne Tecnologia e Treinamento Ltda
Produção: Cerne Tecnologia e Treinamento
E-mail da Empresa: cerne@cerne-tec.com.br
Home Page: www.cerne-tec.com.br.com.br
Atendimento ao Consumidor: sac@cerne-tec.com.br
Contato com o Autor: vitor@cerne-tec.com.br



FEITO NO BRASIL

Dedicatória

Como nos meus outros livros, dedico este livro a minha querida esposa Renata Leal.

"As equações são mais importantes para mim que a política,

pois esta é feita para o presente,

enquanto que as primeiras são algo para a Eternidade."

Albert Einstein

Cerne Tecnologia

A Cerne Tecnologia tem uma equipe preparada para desenvolvimento de projetos eletrônicos em diversas áreas: Médica, Entretenimento, Industrial, Robótica, Científica, Automobilística, Aeronáutica, etc. Trabalhamos com tecnologia microcontrolada usando o PIC, ARM, AVR, 8051, dsPIC, PIC24, PIC32 além do Arduino, Raspberry, Beaglebone etc. Desenvolvemos o projeto desde sua concepção até a entrega do produto final, passando pelas etapas de esquema elétrico, protótipo e desenvolvimento de circuito impresso.

Desenvolvemos aplicativos para smartphones/tablets Android, iOS, Blackberry, Windows Phone e no desenvolvimento de softwares a nível PC para plataforma Windows, usando ferramentas como o Visual Basic, C# e C++.

Atuamos na parte de montagem de placas, onde podemos fornecer ambos os serviços de desenvolvimento de projetos e produção ou apenas um destes.

Desenvolvemos esquemas elétricos e layout de PCI, tanto em tecnologia convencional como SMD.

Temos a flexibilidade de customizar um de nossos produtos, de modo a atender a uma necessidade específica do cliente, tornando o custo de desenvolvimento menor se comparado a construção de um projeto desde a sua fase inicial.

Desenvolvemos e fornecemos kits didáticos para diversos microcontroladores além de apostilas, livros e e-books.

Na hora de desenvolver um projeto ou equipar seu laboratório não hesite em nos contatar. Entre em contato conosco através do endereço cerne-tec.com.br para obter mais informações.

Cerne
Conhecimento para o Desenvolvimento

cerne-tec.com.br

Sumário

01. Metodologia de desenvolvimento	**8**
02. Comunicação Bluetooth	**9**
03. Comunicação GSM e GPRS	**20**
04. Comunicação MODBUS	**39**
05. Comunicação LIN	**67**
06. Amperímetro	**84**
07. Comunicação PS2	**92**
08. RFID	**104**
09. Comunicação ZigBee	**115**
10. Barômetro	**129**
11. Higrômetro	**134**
12. Comunicação com DDS AD9835	**140**
13. Termômetro com NTC	**159**
14. Comunicação RS485	**165**
15. Geração de barras de vídeo	**175**
16. Voltímetro	**180**
17. Potenciômetro digital	**187**
18. Recepção IR no protocolo RC5	**193**
19. Comunicação SPI	**206**
20. Comunicação MiWi	**218**
21. Partida de motor AC	**249**
22. Comunicação com leitor de Código de Barras	**308**
23. Comunicação Microwire	**329**
24. Projetos com Energia Solar	**347**
25. Acelerômetro LIS302DL	**363**
26. Gaussímetro	**376**
27. Comunicação com Leitor Biométrico	**387**
28. Detecção de DTMF	**412**
29. Decodificação do HT6P20B	**426**

30. Medindo Etanol em ppm **437**
31. Medição de Monóxido de Carbono **446**
32. Medição de Dióxido de Carbono **460**
33. Medindo a concentração de GLP **474**
34. Medindo a concentração de H_2 **485**
35. Medindo a concentração de Amônia **496**
36. Comunicação com Leitor MIFARE **508**
37. Analisador Lógico **532**
38. Medindo a concentração de Benzeno **553**
39. Célula de Carga **564**
40. Decibelímetro **574**
41. Medindo o fluxo de massa **587**
42. Osciloscópio **596**
43. Medindo a concentração de Ozônio **612**
44. Medição de pH **623**
45. Reconhecendo comandos de voz **634**
46. Amperímetro **646**
47. Medição de volume de fluido por Ultrassom **654**
48. Medindo a radiação ultravioleta **664**
49. VU Meter **677**
50. UART por software **686**

Capítulo 1
Metodologia de desenvolvimento

1. Introdução

Esta literatura é uma continuação da obra *Programando o ESP32 no Arduino (2018)* do mesmo autor e editora, onde outros exemplos serão explorados de modo que o leitor possa ampliar seu embasamento teórico e prático para desenvolver mais aplicações nesta ferramenta. É importante que o leitor tenha ciência dos assuntos abordados na obra citada, para que haja um melhor aproveitamento do conteúdo a ser apresentado.

Acompanhe o sumário para verificar os temas abordados neste volume.

A placa didática utilizada foi a NodeMCU-32S, onde os recursos para testar os circuitos propostos serão conectados através das conexões disponíveis na lateral do kit. Tal kit está à venda no site www.cerne-tec.com.br.

Capítulo 2

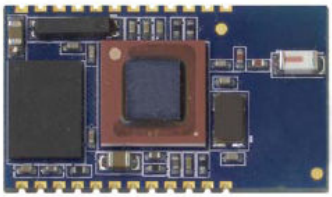
Comunicação Bluetooth

1. O que é Bluetooth?

Bluetooth é uma tecnologia de baixo custo para a comunicação sem fio entre dispositivos eletrônicos a pequenas distâncias. Com o Bluetooth o usuário pode detectar e conectar o seu aparelho de forma rápida a outros dispositivos que tenham a mesma tecnologia.

2. O módulo KC-21

Para comunicar com a rede Bluetooth, será utilizado o módulo KC-21, que pode ser visualizado na figura abaixo:

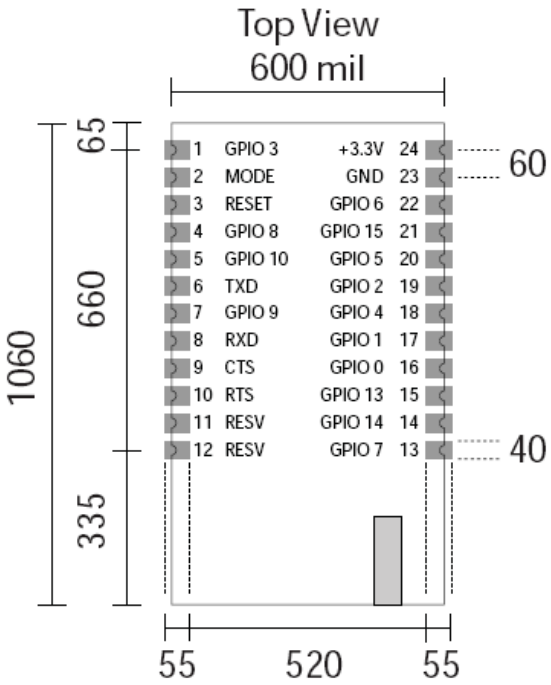


15mm x 27mm

Obs: *Outros módulos também podem ser utilizados.*

Através deste módulo a rede Bluetooth será tratada como um canal serial de comunicação do tipo RS232, onde poderemos enviar e receber dados oriundos desta rede. O KC21 já implementa a pilha de conexão Bluetooth no sistema. O KC21 foi desenvolvido pela WireFree, e mais informações sobre o mesmo podem ser conseguidas no site www.kcwirefree.com.

A pinagem deste módulo está disponível na figura abaixo:



A seguir a descrição destes pinos.

Nome	Tipo	Pino	Descrição
Interface Serial			
RXD	Entrada	8	Recepção de dados
TXD	Saída	6	Transmissão de dados
CTS	Entrada	9	Pronto para enviar (ativo em nível baixo)
RTS	Saída	10	Pedido para enviar (ativo em nível baixo)
Reservado			
MODE	Entrada	2	Reservado
DM	Entrada/ Saída	11	Reservado
DP	Entrada/ Saída	12	Reservado
Alimentação			
VDD		24	Alimentação positiva (2,7V a 3,6V)
GND		23	Terra da alimentação
Reset			
RESET	Entrada	3	Entrada de reset(ativo em nível baixo)
GPIO – General Purpose Input/Output (Pino de Entrada e Saída de propósito geral)			
GPIO [0]	I/O	16	Pino de I/O
GPIO [1]	I/O	17	Pino de I/O
GPIO [2]	I/O	19	Pino de I/O
GPIO [3]	I/O	1	Pino de I/O
GPIO [4]	I/O	18	Pino de I/O
GPIO [5]	I/O	20	Pino de I/O
GPIO [6]	I/O	22	Pino de I/O
GPIO [7]	I/O	13	Pino de I/O
GPIO [8]	I/O	4	Pino de I/O
GPIO [9]	I/O	7	Pino de I/O
GPIO [10]	I/O	5	Pino de I/O
GPIO [11]	I/O		Não disponível
GPIO [12]	I/O		Não disponível

3. Comandos AT

Os comandos AT são usados de forma a permitir a comunicação entre o microcontrolador e o módulo de comunicação Bluetooth. Para conhecer todos os comandos AT disponíveis neste módulo, recomenda-se a leitura do arquivo *kcSerial_RefGuide.pdf* disponível para download no site do fabricante. Neste ponto serão estudados dois comandos que permitem inicializar o módulo ao ponto de comunicar com o mesmo através de um computador ou tablet/smartphone.

4. Comandos AT+ZV DefaultLocalName

Através do comando `AT+ZV DefaultLocalName` é possível alterar o nome no qual será apresentado no momento em que conectarmos com a rede bluetooth. A sua sintaxe é a seguinte:

`AT+ZV DefaultLocalName` *`nome_a_ser_apresentado`* ***`fim_de_texto`***

No parâmetro *`nome_a_ser_apresentado`* informamos o nome no qual será reconhecido pelo computador, podendo ser um texto com até 40 caracteres. O parâmetro ***`fim_de_texto`*** serve para informar ao módulo que o comando está sendo finalizado e vale os

caracteres CR (13) e LF (10) na tabela ASCII. Abaixo um exemplo prático deste comando:

```
AT+ZV DefaultLocalName Cerne Bluetooth
```

O comando `Serial.println()` já envia o terminador CR+LF.

5. Comandos AT+ZV EnableBond

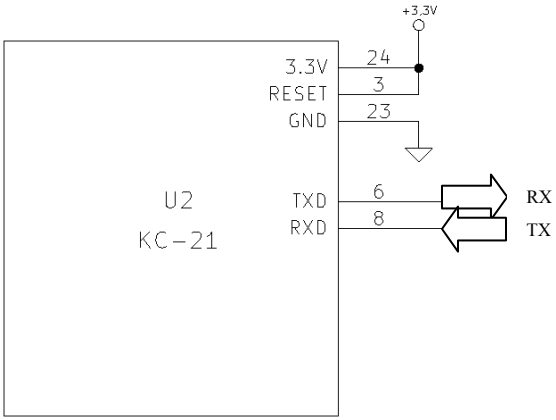
O comando `AT+ZV EnableBond` serve para ativar o módulo Bluetooth. Feito isso, ele estará habilitado a comunicar nesta rede. Abaixo um exemplo prático deste comando:

```
AT+ZV EnableBond
```

Obs: *A taxa de comunicação default do módulo é 115kbps, 8 bits de dados, sem paridade e 1 bit de stop.*

6. Esquema Elétrico

Alimente o módulo e conecte os pinos de comunicação nos pinos TX e RX do NodeMCU-32S .



7. Código Fonte

O código fonte apresentado a seguir permite ligar ou desligar o led de acordo com o caracter recebido via Bluetooth. Caso o caracter seja “A” o led liga e caso seja “a” o led desliga. Observe que o código em si parece muito com a comunicação serial do tipo RS-232, haja vista que o módulo KC-21 funciona como um conversor Bluetooth para serial.

```
/*  *  *  *  *  *  *  *  *  *  *  *  *
 *          Cerne Tecnologia          *
 *        www.cerne-tec.com.br        *
 *  Projetos em Arduino - Bluetooth   *
 *  *  *  *  *  *  *  *  *  *  *  *  */
```

```
/*  *  *  *  *  *  *  *  *  *  *  *  *
 *          Declaração de I/Os       *
 *  *  *  *  *  *  *  *  *  *  *  *  */
#define  LED    0

/*  *  *  *  *  *  *  *  *  *  *  *  *
 *        Função de inicialização    *
 *  *  *  *  *  *  *  *  *  *  *  *  */
void setup(void)
{
  pinMode(LED,OUTPUT);

  Serial.begin(115200);
  Serial.println("AT+ZV DefaultLocalName ARDUINO");
  Serial.println("AT+ZV EnableBond");
}

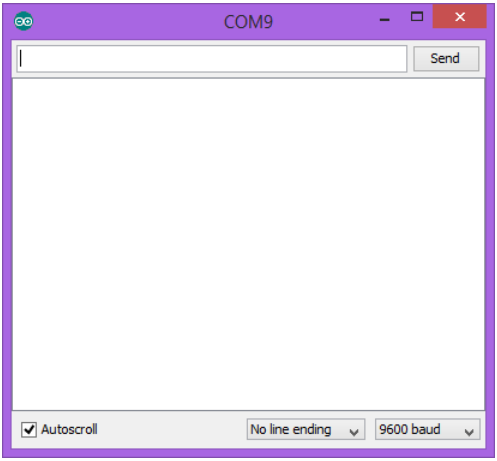
/*  *  *  *  *  *  *  *  *  *  *  *  *
 *          Função de loop           *
 *  *  *  *  *  *  *  *  *  *  *  *  */
void loop(void)
{
  if(Serial.available()>0)
  {
    char rx;
    rx=Serial.read();

    if(rx=='A')
      digitalWrite(LED,1);

    if(rx=='a')
      digitalWrite(LED,0);
  }
}
```

8. Comunicando com a Placa

Para comunicar com o ESP32 é possível utilizar a ferramenta de comunicação serial disponível na própria ferramenta. Para isso, vá ao menu Tools -> Serial Monitor. Neste instante, a seguinte tela será carregada.



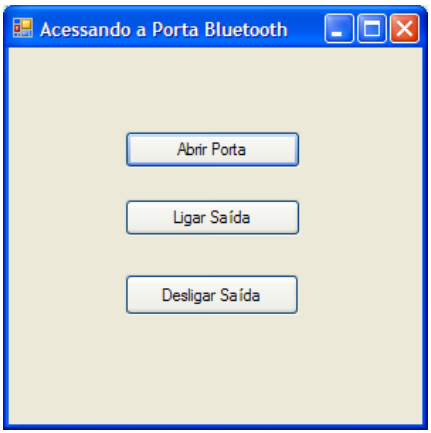
Altere o baud-rate para 115200 e digite o caracter na caixa de texto ao lado esquerdo do botão Send. Envia o caracter A para ligar e a para desligar o led e observe o seu funcionamento.

Obs: *Verifique em qual porta o adaptador bluetooth foi reconhecido no seu PC e configure corretamente tal porta através da opção Tools -> Serial Port.*

9. Aplicação em Visual Basic

A comunicação Bluetooth é idêntica a comunicação serial do tipo RS232, pois o serviço disponível é o serial, permitindo ao computador criar uma porta de comunicação para que o Visual Basic possa comunicar. Baseado no exemplo anterior o programa permitirá ligar e desligar o led através de um programa elaborado no computador.

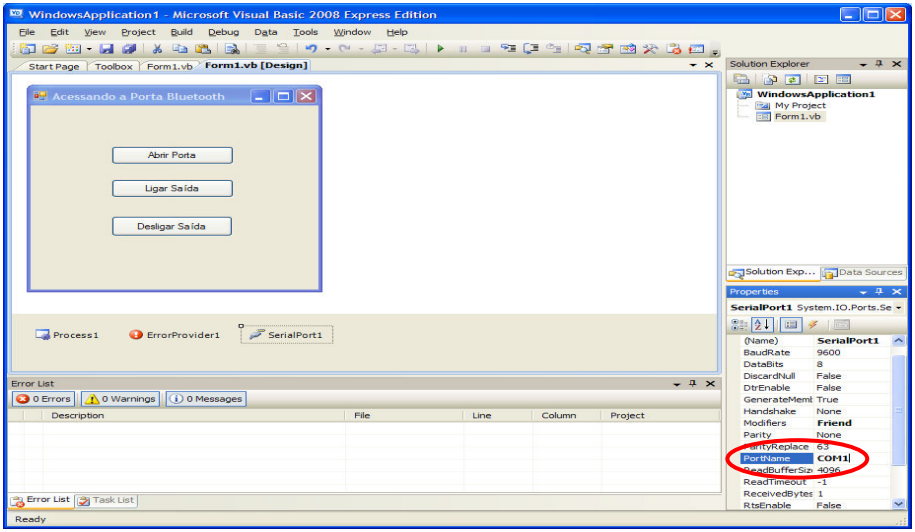
Crie uma nova interface no Visual Basic Express e deixe-a com a seguinte interface.



Adicione o componente `SerialPort` no formulário e em seguida, clique duas vezes no botão Abrir Porta. Coloque o seguinte trecho de código associado a este botão:

```
SerialPort1.Open()
```

Através deste comando é possível abrir a porta de comunicação e deixá-la pronta para enviar dados. No entanto, é importante observar qual foi o número de porta que o gerenciador Bluetooth do Windows disponibilizou para a placa e alterar neste componente tal parâmetro. Para isso, dê um clique no componente `SerialPort` e observe na janela de propriedades a propriedade `PortName`, conforme marcado a seguir.



A opção default selecionada é a porta COM1. É justamente neste ponto que deve-se alterar para a porta disponibilizada pelo gerenciador do Windows. Suponha que a porta 5 foi selecionada, neste caso, deve-se colocar neste campo o valor COM5.

O programa gravado no ESP32 irá ligar o led sempre que receber o caracter “A” e desligar o led sempre que receber o caracter “a”. Associe os seguintes trechos de código ao botão Ligar Saída e Desligar Saída respectivamente:

```
SerialPort1.Write("A") 'Liga a saída

SerialPort1.Write("a") 'Desliga a saída
```

Teste o exemplo e observe que o led liga e desliga de acordo com o caracter recebido. Através do Bluetooth é possível estabelecer comunicações via RF (rádio frequência) em distâncias de até 10 m, pois o módulo utilizado é do tipo classe 2, onde a distância máxima é desta ordem.

Capítulo 3

Comunicação GSM e GPRS

1. Introdução

Nesta etapa da literatura é abordado como realizar comunicação SMS com o ESP32 e o módulo de comunicação SIM340 da SIMCOM. O leitor verá como ler um pino do microcontrolador e enviar os comandos para o módulo de forma a enviar mensagem SMS para o número celular informado. Também é apresentado como realizar uma chamada para um telefone.

2. Módulo SIM340

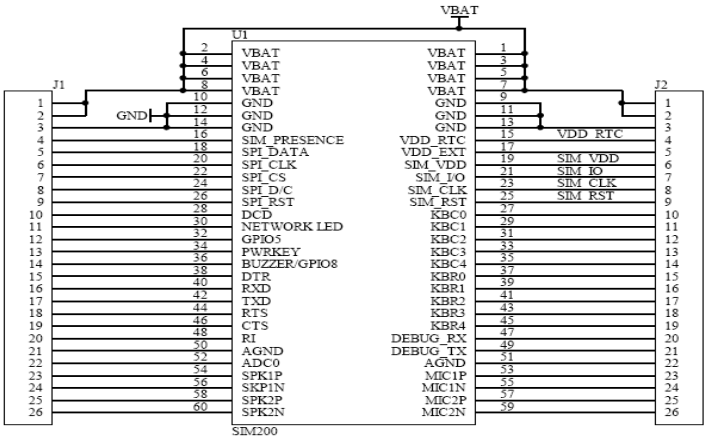
O módulo SIM340 foi desenvolvido pela SIMCOM (www.simcom.com). Este módulo permite realizar conexão GSM além da comunicação GPRS. Observe abaixo uma imagem deste módulo.



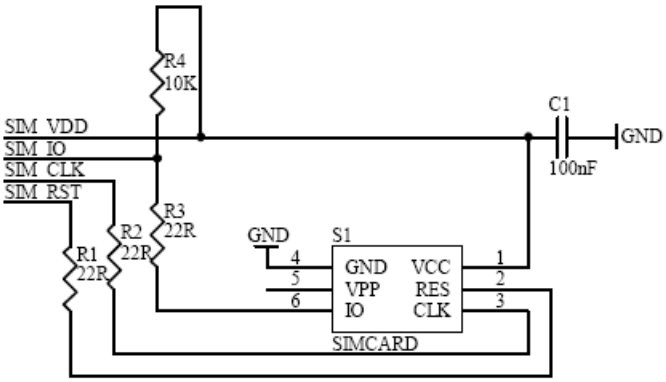
A seguir as principais características deste módulo.

Características
Quad-band, ou seja, funciona em 4 bandas de frequência que são 850, 900, 1800 e 1900 MHz;
Ideal para comunicação de Voz, SMS, GPRS e Fax;
Alimentação de 3,4 V a 4,5V;
Baixo consumo;
Temperatura de operação de -20ºC a 60ºC;
Peso de 8g;
Comunicação com o módulo via comandos AT;
Duas entradas analógicas de 10 bits;
Entrada serial para debugação;
Interface com o teclado matricial;
Entrada para conector de antena.

Observa-se pelas informações apresentadas que o módulo praticamente é um celular completo. A seguir a pinagem completa do módulo SIM340.



A seguir a conexão com o receptáculo do chip GSM.

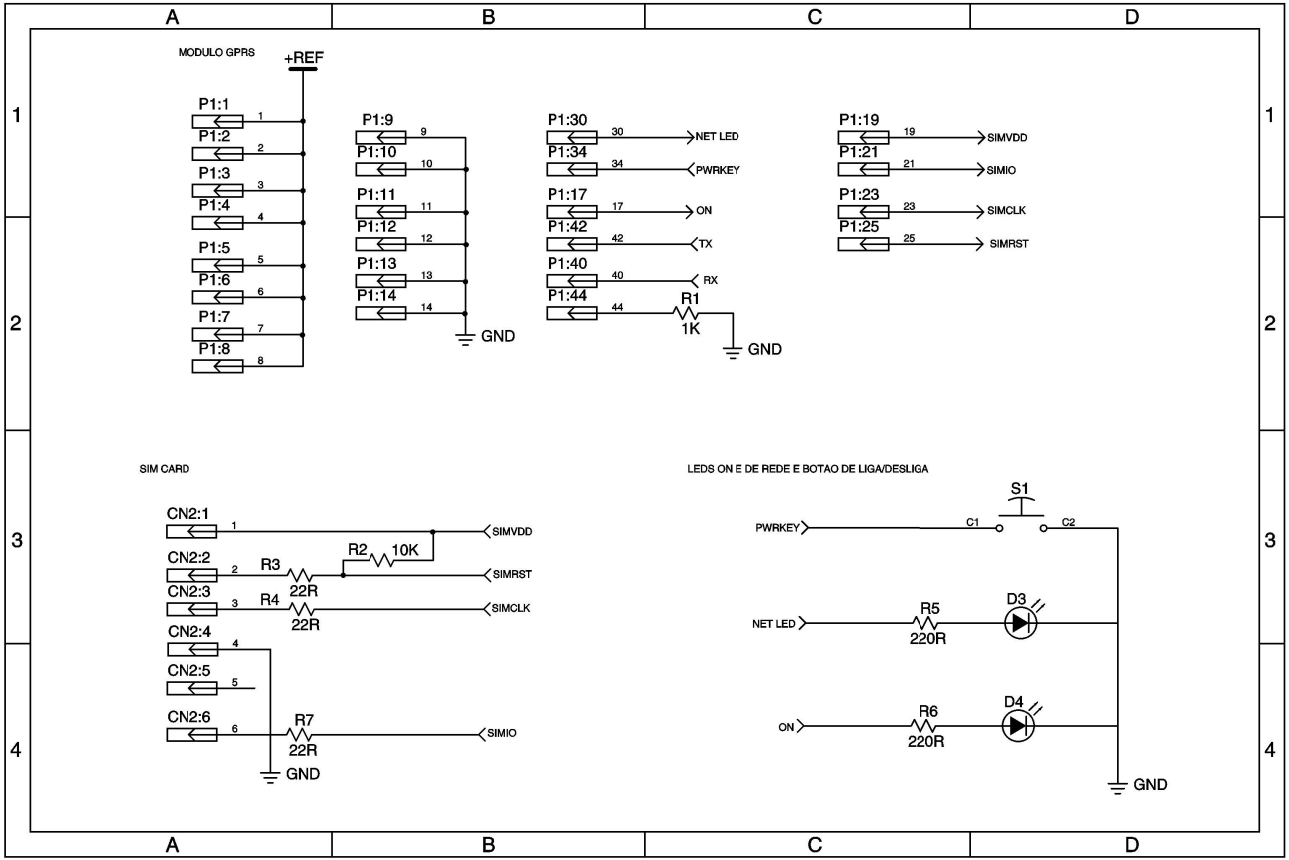


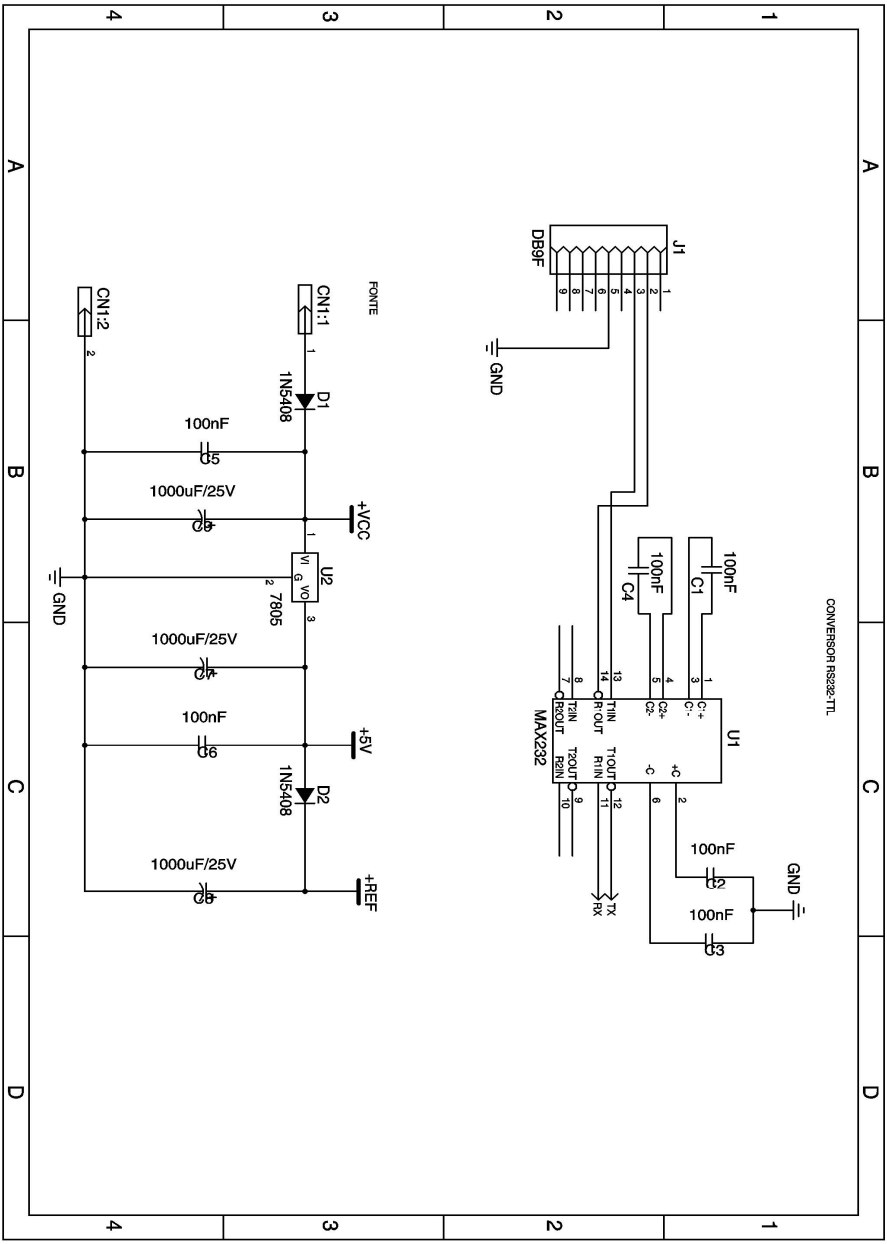
Na tabela abaixo a descrição da pinagem deste chip.

Interface On-Off	
PWRKEY	Através deste pino podemos ligar o módulo ao aterrar tal entrada por 1,5 segundo o módulo pode ser ligado ou desligado.
Interface de Áudio	
MIC1P e MIC1N	Primeira entrada para microfone de eletreto
MIC2P e MIC2N	Segunda entrada para microfone de eletreto
SPK1P e SPK1N	Primeira saída para alto-falante
SPK2P e SPK2N	Segunda saída para alto-falante
Buzzer	Saída para buzina
AGND	Terra analógico
Pinos de entrada e saída	
KBC0 a KBC4	Conexão com o teclado matricial
KBR0 a KBR4	Conexão com o teclado matricial
SPI_DATA	Conexão de Dados via SPI
SPI_CLK	Conexão de Clock via SPI
SPI_CS	Seleção via SPI
SPI_D/C	Conexão para comunicação SPI externa
SPI_RST	Reset externo para comunicação SPI
Network_led	Conexão com led externo de conexão de rede

GPIO8	Conexão de IO8
Interface Serial 1	
DTR	Pino de Data Terminal Ready
RXD	Pino de Recepção de Dados
TXD	Pino de Transmissão de Dados
RTS	Pino de Request to Send
CTS	Pino de Clear do Send
RI	Indicador de Chamada
DCD	Detecção de Data Carrier
Interface Serial 2	
DBGTX	Pino de transmissão da interface serial 2
DBGRX	Pino de recepção da interface serial 2
Interface com SIM CARD	
SIM_VDD	Alimentação para o chip SIM
SIM_I/O	Linha de comunicação com SIM Card
SIM_CLK	Linha de clock para comunicação com SIM CARD
SIM_PRESENCE	Linha de presença de SIM CARD
SIM_RST	Linha de reset com SIM CARD
AUXADC	
ADC0	Pino de entrada analógica
Pinos de Alimentação	
VBAT	Entrada de alimentação para o módulo
GND	Entrada de GND
VRTC	Entrada de alimentação para o RTC interno do módulo
VDD_EXT	Saída de alimentação de 3V

Recomenda-se ao leitor que monte o hardware abaixo, de modo a poder testar o módulo SIM340 através do Serial Monitor do Arduino, simulando assim suas funções antes de programá-lo através do ESP32.


A
B
C
D
1
2
3
4
MODULO GPRS
+REF
P1:1
P1:2
P1:3
P1:4
P1:5
P1:6
P1:7
P1:8
P1:9
P1:10
P1:11
P1:12
P1:13
P1:14
GND
P1:30
NET LED
P1:34
PWRKEY
P1:17
ON
P1:42
TX
P1:40
RX
P1:44
R1
1K
GND
P1:19
SIMVDD
P1:21
SIMIO
P1:23
SIMCLK
P1:25
SIMRST
SIM CARD
CN2:1
CN2:2
CN2:3
CN2:4
CN2:5
CN2:6
R2
10K
R3
22R
R4
22R
R7
22R
SIMVDD
SIMRST
SIMCLK
SIMIO
GND
LEDS ON E DE REDE E BOTAO DE LIGA/DESLIGA
S1
C1
C2
PWRKEY
NET LED
R5
220R
D3
ON
R6
220R
D4
GND


CONVERSOR RS232-TTL
J1
DB9F
GND
100nF
C1
100nF
C4
U1
MAX232
C1+
C1-
C2+
C2-
T1IN
T2IN
R1OUT
R2OUT
T1OUT
T2OUT
R1IN
R2IN
+C
-C
100nF
C2
100nF
C3
GND
TX
RX
FONTE
CN1:1
CN1:2
D1
1N5408
100nF
C5
1000uF/25V
C8
+VCC
U2
7805
VI
VO
G
GND
1000uF/25V
C7
100nF
C6
+5V
D2
1N5408
1000uF/25V
C9
+REF

No esquema há uma entrada de 12V seguido por um regulador de tensão de 5V. Na saída do regulador verifica-se um diodo do tipo 1N4007 que deixa a saída após o mesmo com uma tensão da ordem de 4,4V, ideal para o funcionamento do módulo. Além disso, também observa-se a presença de um conversor TTL-RS232 que converte o nível TTL do módulo para RS232 de forma a comunicar com o computador através deste meio físico. O chip utilizado pode ser de qualquer operadora.

3. Comandos utilizados

Neste ponto são tratados os comandos utilizados para comunicação com o módulo SIM340. É importante que o leitor inicialize o Serial Monitor ou algum software similar e tenha o esquema apresentado anteriormente ou a placa de Comunicação Cerne GSM de forma a poder testar na prática estes comandos. A velocidade de comunicação utilizada são de 115kbps com 1 stop bit e sem paridade. Não deixe de conectar algum SIM CARD na placa com crédito caso seja pré-pago ou pós-pago (conta) de forma a poder realizar chamadas e enviar mensagens SMS e de ligar o módulo através do botão conectado ao pino PWRKEY.

Como placa de apoio que implementa o esquema apresentado foi utilizado o kit de comunicação Cerne GSM, apresentado na próxima figura.



Tal kit encontra-se a venda no site www.cerne-tec.com.br.

ATD

Através deste comando é possível efetuar uma chamada para um telefone específico, como mostra o próximo exemplo.

ATD XXXXXXXXX; + CR + LF

Desta forma, o módulo GSM faz uma chamada para o telefone XXXXXXXXX. Note que esta chamada é feita para o código de área local de onde você estiver. Este comando deve ser seguido de CR + LF de

forma que o módulo entenda o fim do comando. O CR+LF é o pressionar da tecla Enter do teclado, que se refere aos valores decimais 13 e 10 que são chamados de *carriage return (retorno de carro)* e *line feed (avanço de linha)*.

ATH

Encerra-se a chamada feita pelo comando ATD. Observe o exemplo:

ATD XXXXXXXXX; + CR + LF

ATH

ATA

Recebe uma ligação de qualquer telefone.

ATA + CR + LF

AT+CMGF

Configura o modo de interação com o módulo para transmitir e receber os dados por SMS. No exemplo o modo texto é utilizado.

AT+CMGF=1 + CR + LF

AT+CMGS

Permite enviar mensagens SMS para celulares. Observe o exemplo abaixo.

AT+CMGS=”0ZZYYXXXXXXXXX” + CR + LF + texto + 26 (em decimal)

Após enviar o comando AT+CMGS=”0ZZYYXXXXXXXXX”+ CR + LF é aberto um prompt para digitar a mensagem a ser enviada. Feito isso, deve se enviar o valor decimal 26 de forma a encerrar o comando e a mensagem ser transmitida para o celular especificado. O valor ZZ refere-se a operadora e YY ao código de área.

AT+CMGR

Através deste comando podemos ler mensagens SMS. Veja o exemplo abaixo:

AT+CMGR=1 + CR + LF

Neste exemplo é solicitada a leitura da posição 1 da memória SMS.

AT+CMGD

Permite apagar mensagens SMS, como mostra o próximo exemplo.

AT+CMGD=1 + CR + LF

Neste exemplo a posição 1 da memória SMS é apagada.

4. Chamada para telefone

Para realizar uma chamada telefônica o primeiro passo consiste em conectar o kit de comunicação GSM ao NodeMCU-32S . Para isso, conecte os pinos TX e RX entre os mesmos e programe-o para utilizar os comandos apresentados, como o ATD CELULAR; + CR + LF, onde CELULAR é o número no qual deseja que seja chamado seguido de ; e finalizado com o CR+LF.

A seguir um exemplo que faz tal solicitação, bastando substituir o campo XXXXXXXXX pelo número telefônico em si.

```
/*  *  *  *  *  *  *  *  *  *  *  *  *
 *          Cerne Tecnologia          *
 *        www.cerne-tec.com.br        *
 *   Projetos em Arduino - Telefone   *
 *  *  *  *  *  *  *  *  *  *  *  *  */

/*  *  *  *  *  *  *  *  *  *  *  *  *
 *        Função de inicialização     *
 *  *  *  *  *  *  *  *  *  *  *  *  */
```

```
void setup(void)
{
  delay(20000);    //Tempo para conectar a rede
  Serial.begin(115200);
  Serial.println("ATD XXXXXXXXX;");
}

/*  *  *  *  *  *  *  *  *  *  *  *
 *          Função de loop          *
 *  *  *  *  *  *  *  *  *  *  *  *  */
void loop(void)
{

}
```

O próximo exemplo faz a chamada apenas quando o botão conectado ao pino 5 for pressionado.

```
/*  *  *  *  *  *  *  *  *  *  *  *
 *          Cerne Tecnologia        *
 *        www.cerne-tec.com.br      *
 *   Projetos em Arduino - Telefone *
 *  *  *  *  *  *  *  *  *  *  *  *  */

/*  *  *  *  *  *  *  *  *  *  *  *
 *        Declaração de I/Os        *
 *  *  *  *  *  *  *  *  *  *  *  *  */

#define  BOT  5
#define  LED  0

/*  *  *  *  *  *  *  *  *  *  *  *
 *        Função de inicialização   *
 *  *  *  *  *  *  *  *  *  *  *  *  */
void setup(void)
{
  delay(20000);
  Serial.begin(115200);
  pinMode(BOT,INPUT);
```

```
  pinMode(LED,OUTPUT);
  digitalWrite(BOT,1);
}

/*  *  *  *  *  *  *  *  *  *  *  *  *
 *           Função de loop            *
 *  *  *  *  *  *  *  *  *  *  *  *  */
void loop(void)
{
  if(digitalRead(BOT)==0)
  {
      Serial.println("ATD XXXXXXXXX;");
      digitalWrite(LED,1);
      delay(20000);
      Serial.println("ATH");
      digitalWrite(LED,0);
  }
}
```

5. Envio de mensagem SMS

Neste exemplo com o módulo devidamente ligado, basta digitar e enviar via Serial Monitor o comando AT+CMGF=1 + CR + LF de forma a configurar o módulo para funcionar no modo texto. Em seguida, digite AT+CMGS=”0 OPERADORA + CÓDIGO DE ÁREA + CELULAR” + CR + LF, onde operadora é uma das que possibilitam conexão em uma cidade ou entre estados, código de área vai de acordo com a localidade e CELULAR é o número no qual deseja-se enviar a mensagem. Após informar isso, é aberto um prompt com o sinal >, onde é possível informar um texto de até 160 caracteres a ser enviado para o celular.

Após este passo deve-se enviar o valor decimal 26 de forma a encerrar a comunicação. Neste momento a mensagem será enviada para o celular informado.

O exemplo a seguir envia uma mensagem SMS para o celular especificado de acordo com o comando apresentado.

```
/*  *  *  *  *  *  *  *  *  *  *  *  *
 *          Cerne Tecnologia          *
 *        www.cerne-tec.com.br        *
 *      Projetos em Arduino - SMS     *
 *  *  *  *  *  *  *  *  *  *  *  *  */

/*  *  *  *  *  *  *  *  *  *  *  *  *
 *        Função de inicialização     *
 *  *  *  *  *  *  *  *  *  *  *  *  */
void setup(void)
{
  delay(20000);
  Serial.begin(115200);
  Serial.println("AT+CMGF=1");
  Serial.println("AT+CMGS=\"0ZZYYXXXXXXXXX\"");
  Serial.println("Teste GSM com Arduino");
  Serial.write(26);
}

/*  *  *  *  *  *  *  *  *  *  *  *  *
 *          Função de loop            *
 *  *  *  *  *  *  *  *  *  *  *  *  */
void loop(void)
{
}
```

O próximo exemplo envia o SMS sempre que o botão conectado ao pino 5 for pressionado.

```
/*  *  *  *  *  *  *  *  *  *  *  *  *
 *          Cerne Tecnologia          *
 *        www.cerne-tec.com.br        *
 *      Projetos em Arduino - SMS     *
 *  *  *  *  *  *  *  *  *  *  *  *  */

/*  *  *  *  *  *  *  *  *  *  *  *  *
 *        Declaração de I/Os          *
 *  *  *  *  *  *  *  *  *  *  *  *  */
#define  BOT  5
#define  LED  0

/*  *  *  *  *  *  *  *  *  *  *  *  *
 *        Função de inicialização     *
 *  *  *  *  *  *  *  *  *  *  *  *  */
void setup(void)
{
  delay(20000);
  Serial.begin(115200);
  pinMode(BOT,INPUT);
  pinMode(LED,OUTPUT);
  digitalWrite(BOT,1);
}

/*  *  *  *  *  *  *  *  *  *  *  *  *
 *          Função de loop            *
 *  *  *  *  *  *  *  *  *  *  *  *  */
void loop(void)
{
  if(digitalRead(BOT)==0)
  {
    Serial.println("AT+CMGF=1");
    Serial.println("AT+CMGS=\"0ZZYYXXXXXXXXX\"");
    Serial.println("Teste GSM com Arduino");
    Serial.write(26);
```

```
    digitalWrite(LED,1);
    delay(2000);
    digitalWrite(LED,0);
  }
}
```

6. Comunicação GPRS

Para realizar esta comunicação outros comandos ATs são necessários, conforme listado a seguir.

AT+GSN

Através deste comando é possível obter o código chamado IMEI (*International Mobile Equipment Identifier*) que é um código único por celular disponível no mundo.

Envio: AT+GSN + CR + LF

Resposta: Código IMEI do modulo + CR + LF

AT+CSQ

Permite saber a potência do sinal que o módulo está recebendo.

Envio: AT+CSQ + CR + LF

AT+CSTT

Permite conectar com um servidor remoto, usado para comunicação GPRS.

Envio: AT+CSTT=”APN”,”USUÁRIO”,”SENHA”+CR+LF

Exemplo para a operadora TIM:

AT+CSTT=”TIM.BR”,”TIM”,”TIM”

AT+CIFSR

Pega o IP local do módulo após conexão com rede.

Envio: AT+CIFSR+ CR+LF

AT+CDNSCFG

Configura o endereço do servidor DNS

Envio: AT+CDNSCFG="208.67.222.222”+ CR+LF

No exemplo acima o endereço "208.67.222.222” é configurado como servidor DNS.

AT+CDNSGIP

Solicita o IP de uma página na Internet.

Envio: AT+CDNSGIP="www.cerne-tec.com.br”+ CR+LF

Resposta: IP da página informada

AT+CIPSEND

Envia uma mensagem para o IP informado.

Envio: AT+CIPSEND+CR+LF

Resposta: Abre prompt (>) para enviar a mensagem. Após enviar a mensagem, deve se encerrar com o caracter 26 em decimal.

AT+CIPSTART

Faz conexão com servidor remoto.

Envio: AT+CIPSTART=”TIPO DE CONEXÃO”,”IP DO SERVIDOR”,”PORTA”+CR+LF

Onde TIPO DE CONEXÃO pode ser TCP ou UDP dependendo da sua aplicação. O IP DO SERVIDOR é o endereço no qual deseja-se comunicar e PORTA é o número aberto para realizar a comunicação.

AT+CIPCLOSE

Fecha a porta de comunicação aberta.

Envio: AT+CIPCLOSE+CR+LF

7. Exemplo de comunicação GPRS

O próximo exemplo faz uso dos comandos apresentados para conectar e enviar uma string a um IP de um servidor de internet.

```
/*  *  *  *  *  *  *  *  *  *  *  *  *
 *          Cerne Tecnologia          *
 *        www.cerne-tec.com.br        *
 *      Projetos em Arduino - GPRS    *
 *  *  *  *  *  *  *  *  *  *  *  *  */

/*  *  *  *  *  *  *  *  *  *  *  *  *
 *        Função de inicialização     *
 *  *  *  *  *  *  *  *  *  *  *  *  */
void setup(void)
{
  delay(20000);
  Serial.begin(115200);

  Serial.println("AT+CSTT=\"tim.br\",\"tim\",\"tim\"");
  Serial.println("AT+CDNSCFG=\"208.67.222.222\"");
  Serial.println("AT+CDNSGIP=\"cerne-tec.com.br\"");
  Serial.println("AT+CIFSR");
  Serial.println("AT+CIPSTART=\"TCP\",\"201.29.122.164\",\"6080\"");

  Serial.println("AT+CIPSEND");
  Serial.println("Testando GPRS...");
  Serial.write(26);
}

/*  *  *  *  *  *  *  *  *  *  *  *  *
 *          Função de loop            *
 *  *  *  *  *  *  *  *  *  *  *  *  */
void loop(void)
{
}
```

Capítulo 4

Comunicação MODBUS

1. Protocolo MODBUS

O protocolo MODBUS é um dos mais usados a nível industrial atualmente. O mesmo é usado por CLPs e computadores para monitorarem, por exemplo, uma rede de periféricos como sensores, válvulas, entradas analógicas e saídas analógicas através desta rede. Observe a seguir algumas características deste protocolo.

Características
Utilizado para comunicação entre CLPs
Define a estrutura de dados e não de meio físico
Usualmente utilizado sobre RS232, RS485 ou Ethernet
Arquitetura do tipo mestre/escravo
Suporta broadcast
Dois tipos de quadros

O protocolo MODBUS foi desenvolvido pela *Modicon Industrial Automation Systems*, hoje Schneider, para comunicar um dispositivo mestre com outros dispositivos escravos. Embora seja utilizado normalmente sobre conexões seriais padrão RS-232, ele também pode ser usado como um protocolo da camada de aplicação de redes industriais tais como TCP/IP sobre Ethernet e MAP. Possivelmente

este é o protocolo de mais larga utilização em automação industrial, pela sua simplicidade e facilidade de implementação.

2. Tipos de Quadros

Os tipos de quadros ou frames disponíveis para o MODBUS são os seguintes.

Tipos de quadros
Quadro de consulta
Endereço
Código da função
Dados
Verificação de erro
Dois tipos de quadros

Ou seja, o quadro que pode fazer a consulta, por exemplo, em um sensor ou CLP contém nele um campo que identifica o endereço no qual o mestre deseja comunicar com o escravo, podendo ser do endereço de 1 a 247. Caso o endereço seja 0, é identificado como um tipo de mensagem broadcast, onde todos os elementos da rede podem receber tal comando. Em seguida é informado o código da função que identifica o que o frame em si faz. Seguido do identificador da função vem à parte de dados, onde neste há o pacote de dados referente às informações que foram enviadas. Finalmente há a parte de verificação de erros, onde ocorre a

checagem de dados enviados, a fim de verificar se foi enviado corretamente.

O próximo frame é o de resposta, composto dos seguintes itens:

Tipo de resposta
Quadro de resposta
Endereço
Confirmação (eco do código de função enviado)
Dados
Verificação de erro

3. Modos de Transmissão

Existem dois tipos de modos de transmissão usados no MODBUS. O primeiro deles chama-se modo do tipo ASCII e o segundo RTU. No mercado industrial e em geral, o modo do tipo RTU é o mais usado frente ao tipo ASCII. No modo de transmissão ASCII há o desperdício de dados do canal, pois cada caracter enviado pela rede é enviado como um caracter ASCII enquanto que no modo RTU o dado é transmitido em forma de bits. Observe abaixo como é formado o frame do tipo RTU:

Start	Endereço	Função	Dados	CRC	END
Silêncio 3..5 chars	← 8 bits→	← 8 bits→	←N x 8 bits→	←16 bits→	Silêncio 3..5 chars

Inicialmente há um período de silêncio, onde sempre que as linhas de comunicação ficarem sem tráfego de dados pelo tempo de até 3,5 caracteres, o escravo entende que a rede entrou em estado idle, ou seja, repouso. Como a comunicação utilizada terá uma taxa de comunicação de 9600 bps, pode-se observar que o tempo de transmissão de 1 bit será de.

$$T= 1/9600= 104\ \mu s$$

Como a comunicação utilizada terá 1 bit de start, 8 de dados e 1 de stop, ao todo há 10 bits. Para achar o tempo para transmitir 1 byte na rede MODBUS, deve-se multiplicar 104 µs x 10, encontrando o tempo de 1,04 ms. Ou seja, o tempo para transmitir 1 byte no MODBUS são de 1,04 ms e caso a rede fique 3,64 ms em inatividade (que é referente a 3,5 caracteres), tem-se o estado de silêncio ou *idle*. Em seguida, observa-se os 8 bits referentes a endereço. Caso este campo esteja em 0 é uma indicação da comunicação do tipo broadcast, ou seja, o mestre comunica com todos os elementos da rede e caso o endereço seja de 1 a 247 indica que comunica com um elemento específico da rede. No campo de função há a indicação do que faz o comando enviado, segundo a tabela abaixo.

Código	Descrição
01	GND
02	RESET
03	Antena
04	Antena
05	Não conectado
06	Não conectado
07	Formato de seleção
08	Dado 1
09	Dado 0
10	Buzzer
11	5V

No campo de dados estão os dados referentes ao tipo de comando enviado e o campo de CRC a checagem de dados.

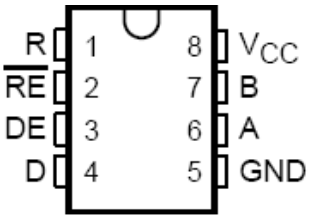
Nos experimentos a serem implementados apenas as funções 0x01 e 0x05 são utilizadas.

4. Comunicação RS-485

O RS485 é uma evolução do padrão RS422 tendo como principal enfoque a comunicação em rede, ou seja, com apenas um par de fios é possível comunicar com diversos equipamentos em rede usando o mesmo barramento. Assim como o RS422, o RS485 utiliza linha de dados balanceada, bastante similar as linhas de dados da

interface RS422, logo permite a comunicação em distâncias de até 1200 metros de maneira extremamente confiável.

A interface de hardware para comunicação RS485 utilizada é baseada no chip 75176, um conversor TTL para RS485 como mostra a pinagem da figura abaixo.



Observe as funções destes pinos na tabela a seguir.

Código	Descrição
1	Saída de dados TTL
2	Seleção de modo TX/RX
3	Seleção de modo TX/RX
4	Entrada de dados para transmissão
5	GND
6	Saída RS485 D+ ou A
7	Saída RS485 D- ou B
8	VCC

Os níveis elétricos 1 e 0 na interface RS485 estão apresentados na próxima tabela.

Function Tables

DRIVER

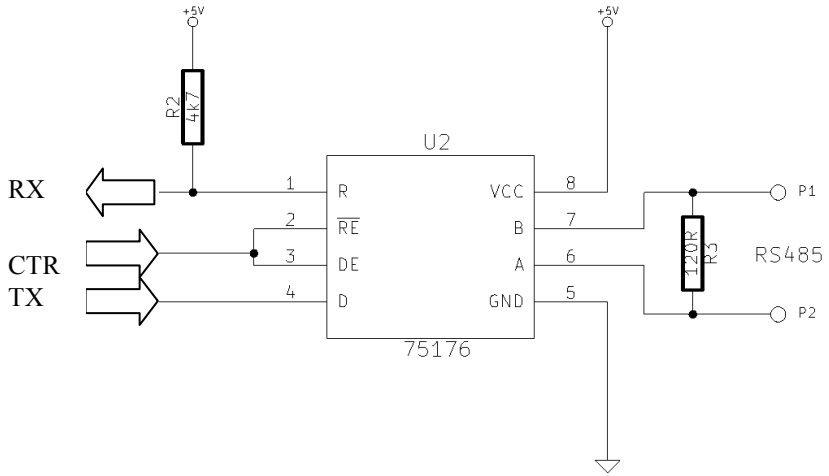
INPUT D	ENABLE DE	OUTPUTS A	OUTPUTS B
H	H	H	L
L	H	L	H
X	L	Z	Z

RECEIVER

DIFFERENTIAL INPUTS A – B	ENABLE $\overline{RE}$	OUTPUT R
$V_{ID} \geq 0.2$ V	L	H
-0.2 V $< V_{ID} < 0.2$ V	L	?
$V_{ID} \leq -0.2$ V	L	L
X	H	Z
Open	L	?

H = high level, L = low level, ? = indeterminate, X = irrelevant, Z = high impedance (off)

O circuito apresentado a seguir deve ser montado de modo a permitir a comunicação RS485 com o módulo ESP32, haja vista que a mesma não possui tal recurso. A seguir o esquema elétrico.

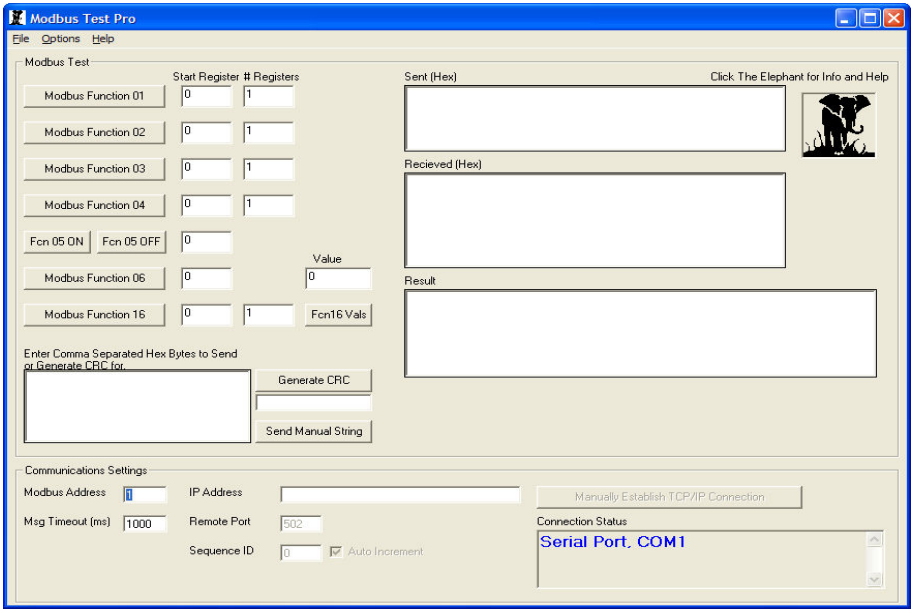


Atente que os pinos de comunicação TTL estão conectados ao 75176, além de um pino adicional ser utilizado para colocá-lo em modo de transmissão ou recepção.

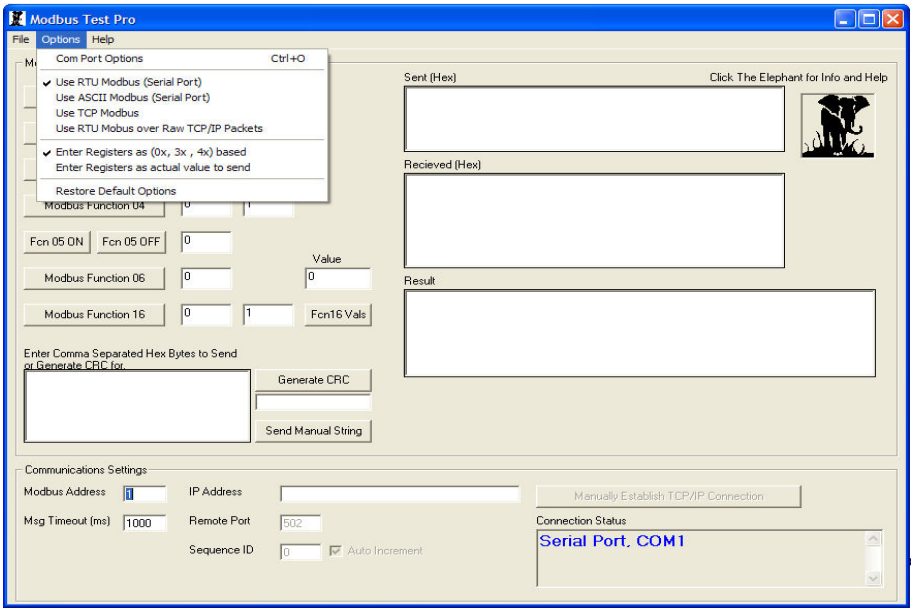
Obs: *O leitor deve possuir um conversor RS232 para RS485 conectado a serial do PC.*

5. Software de Comunicação MODBUS

Existem vários softwares que permitem a comunicação com dispositivos MODBUS. Nesta literatura, será usado o software MODBUS Test Pro, que pode ser visualizado abaixo:



Este software pode funcionar em modo serial ou TCP/IP, bastando para isso selecionar o modo indo no menu *Options*, onde a seguinte tela surgirá.

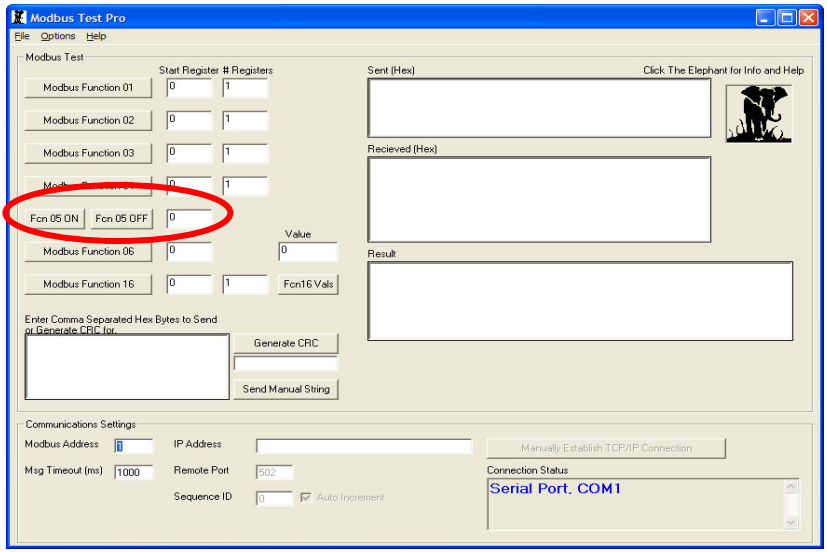


Observe que além de poder escolher o funcionamento entre Serial e TCP/IP é possível selecionar a opção RTU ou ASCII. Deixe neste caso a opção *Use RTU MODBUS (Serial Port)* marcada. A versão DEMO deste software pode ser baixado gratuitamente no seguinte link: www.rogue-engr.com.

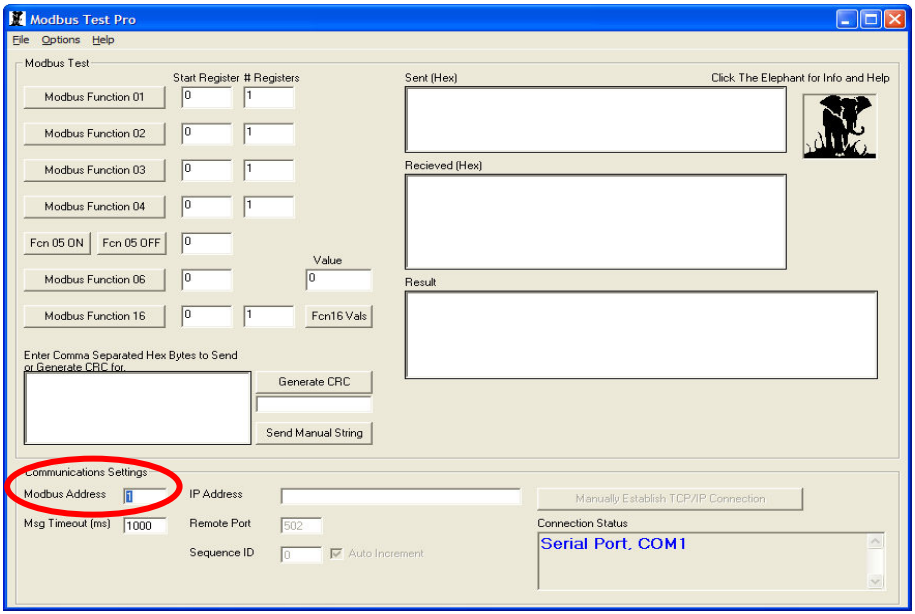
Verifique as funcionalidades deste software, observando melhor o seu funcionamento, pois em breve ele será usado em conjunto com o NodeMCU-32S .

6. Protocolo de Comunicação

No exemplo proposto, o objetivo é ligar ou desligar o led presente no pino 16 mo NodeMCU-32S . Para isso, foi utilizada a interface RS485 e o endereço da placa será o endereço 1. Foi utilizada a função 5 da tabela de comandos do MODBUS, pois este permite ligar ou desligar uma saída (bobina) na rede MODBUS. Esta opção está marcada na figura a seguir.

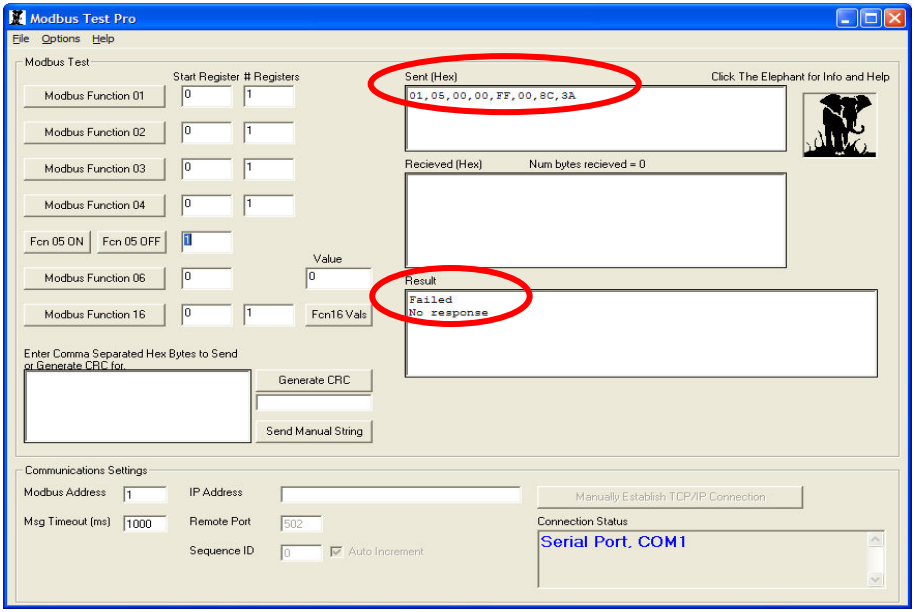


Especifique o endereço do dispositivo que deseja acionar na caixa apresentada a seguir.



Informe o endereço 1, pois este é o endereço programado no ESP32.

Pressione o botão Fcn 05 ON, o resultado será o seguinte.



No campo Send (Hex) é apresentado o conjunto de dados enviado na rede MODBUS. Observe que foram enviados os seguintes dados em hexadecimal: 01, 05, 00, 00, FF, 00, 8C e 3A. Compare estes dados enviados com o protocolo apresentado abaixo do quadro de funcionamento RTU.

Start	Endereço	Função	Dados	CRC	END
Silêncio 3..5 chars	← 8 bits→	← 8 bits→	←N x 8 bits→	←16 bits→	Silêncio 3..5 chars

O endereço enviado é 0x01, conforme configurado no campo apresentado. Em seguida, o valor 0x05 refere-se à função usada que