Teste dos dispositivos da Caixa Preta

|  |  |  |
| --- | --- | --- |
| Nr | Nome | O que faz |
| 1 | Leds | Faz um contador binário com os leds |
| 2 | LCD | Mostra todos os caracteres do LCD |
| 3 | Teclado | Mostra os códigos das teclas |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

// TESTE - Mensagens do modo de teste

char \*teste\_msg[]={ "ERRO", //0

"1-LEDs", //1

"2-LCD", //2

"3-Teclado", //3

"4-Acel e giro", //4

"5-Magnetometro"}; //5

A função **byte teste\_sel(void)** permite fazer a seleção entre as opções

**Teste 1 – Leds**

Faz um contador binário com os 4 leds. Usa a função **leds\_cont(char val)**

|  |  |  |  |
| --- | --- | --- | --- |
| Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Laranja | Verde | Amarelo | Vermelho |
| AMO | VD | AM | VM |

**Teste 2 – LCD**

Mostra todos os caracteres do LCD, de 0x00 até 0xFF. Por vez são 64 caracteres, 16 por cada linha. Na coluna da esquerda apresenta em Hexa o código do primeiro caracter. Faz pausa de 1 segundo antes e mudar a tela.

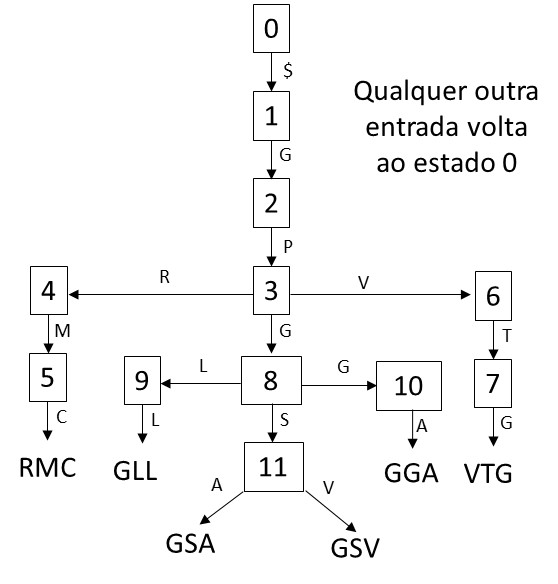
**Teste 3 – Teclado**

Na porta serial é possível ver os transientes que surgem quando uma tecla é acionada ou solta.

A variável **byte sw\_val** é atualizada pelo Timer 1 e contém a média das duas últimas leituras (8 bits) feitas com o ADC.

**Teste 9 – GPS Tudo**

Diagrama de estados para identificar as mensagens.



**Teste 10 – Filtrar informações de RMC e GSA**

------------------------------------------------------------------------------------------------------------------------------------------------

GPRMC

$GPRMC,hhmmss,status,latitude,N,longitude,E,spd,cog,ddmmyy,mv,mvE,mode\*cs<CR><LF>

$GPRMC,083559.00,A,4717.11437,N,00833.91522,E,0.004,77.52,091202,,,A\*57

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| $GPRMC, | hhmmss.sss, | Stat, | Lat:ddmm.mmmmm, | N/S, | Long:dddmm.mmmmm, | E/W, |
| $GPRMC, | 083559.00, | A, | 4717.11437, | N, | 00833.91522, | E, |
| 7 | 12 | 2 | 11 | 2 | 12 | 2 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Speed:ddd.ddd, | Curso:ddd.ddd, | Data:ddmmyy | Mv, | mvE, | Modo | \*Check | CR LF |
| 0.004, | 77.52, | 091202, | , | , | A | \*57 | 0xD 0xA |
| ?8 | ?8 | 7 | ?8 | ?2 | 1 | 3 | 2 |

Tamanho = 80 bytes, vou usar tamanho 100.

Lido com Arduino: $GPRMC,131732.00,A,1548.62581,S,04748.65809,W,0.299,,260120,,,A\*72

------------------------------------------------------------------------------------------------------------------------------------------------

GPGSA

$GPGSA,Smode,FS{,sv},PDOP,HDOP,VDOP\*cs<CR><LF>

$GPGSA,A,3,23,29,07,08,09,18,26,28,,,,,1.94,1.18,1.54\*0D

Mostra identificador de até 12 satélites

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $GPGSA, | Smode, | Fix, | Sat1, | Sat2, | Sat3, | Sat4, | Sat5, | Sat6, | Sat7, | Sat8 |
| $GPRMC, | A, | 3, | 23, | 29, | 07, | 08, | 09, | 18, | 26, | 28, |
| 7 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| Sat9, | Sat10 | Sat11, | Sat12, | PDOP, | HDOP, | VDOP | \*Check | CR LF |
| 18, | 18, | 18, | 18, | 1.94, | 1.18, | 1.54 | \*0D | 0xD 0xA |
| 3 | 3 | 3 | 3 | 5 | 5 | 5 | 3 | 2 |

Tamanho = 67 bytes, vou usar tamanho 100.

Lido com Arduino: $GPGSA,A,3,07,09,16,23,04,01,,,,,,,8.16,1.24,8.06\*09

Criar um vetor para armazenar as informações importantes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RMC | RMC | RMC | RMC | RMC | RMC | RMC |
| 0 | 2 | 13 | 20 | 31 | 33 | 45 |
| Status | Hora | Data | Latitude | N/S | Longitude | E/W |
| A0 | hhmmss.sss0 | ddmmyy0 | ddmm.mmmmm0 | N0 | dddmm.mmmmm0 | E0 |
| A | 083559.00 | 091202 | 4717.11437 | N | 00833.91522 | E |
| 1+1 | 10+1 | 6+1 |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RMC | RMC | GSA | GSA | GSA | VTG | VTG |
| 47 | 55 | 63 | 69 | 75 | 81 | 88 |
| Velocidade Nós | Curso | PDOP | HDOP | VDOP | Speed km/h | Unidade |
| ddd.ddd0 | ddd.ddd0 | dd.dd0 | dd.dd0 | dd.dd0 | xxx.xx0 | K0 |
| 0.004 | 77.52 | 99.99 | 99.99 | 99.99 | ?125.12 | K |
| 1+1 |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| GGA | GGA | GGA | GGA |  |  |  |
| 90 | 92 | 95 | 101 | 104 |  |  |
| Fix | Qtd Sat | Altitude | Unidade |  |  |  |
| d0 | dd0 | dddd.d0 | m0 |  |  |  |
| 1 | 4 | 627.4 | m |  |  |  |
| 1+1 | 2+1 |  | 1+1 |  |  |  |

**Teste 11 – GPS 🡪 u-Center**

**Teste 12 – MPU 🡪 Matlab**

Aqui existe um problema: quando o Matlab abre a porta serial (COMi), o Arduino Resseta. Isso acontece porque ao abrir a porta COMi, ele desativa e ativa novamente o DTR. Isto significa Reset para o Arduino. O Matlab, espera receber dados e, no entanto, chegam mensagens das opções de teste.

A solução foi colocar pausas e exigir uma sequência para sincronizar Arduino e Matlab.

|  |  |
| --- | --- |
| Matlab | Arduino |
| Abre a porta COMi | RESET |
| Mensagem: Prepare Arduino  Pausa (10) | Tempo para o Arduino ressetar e permitir ao usuário posicionar para a opção 12 |
| flushinput(sid) = apaga o passado serial | - |
| Mensagem: Pronto para receber dados. | (SEL) Seleciona a opção 12 |
| Pausa (4) | Inicia a transmissão |
| fprintf(1,'Recebendo ...\n');  flushinput(sid); |  |

; Exemplo Teraterm - força reset do Arduino

; enable dtr

setdtr 1

; clear rts

;setrts 0

; now clear dtr, so that the system will reset and see rts clear

setdtr 0

**Sugestão 1)**

No futuro usar duas posições da memória SRAM para indicar que o Arduino foi ressetado pelo Matlab. Por exemplo, quando entrar no modo que envia para o Matlab, gravar 0x55 e 0xAA nas posições 0 e 1 da SRAM. Ao acorda após o reset, ele verifica essas duas posições e se forma0x55 e 0xAA, entra diretamente no modo que envia para o Matlab. Dispensa ação do usuário para selecionar novamente e evita o envio de texto pela porta serial, o que confunde o Matlab.

**Sugestão 2)**

O Matlab abre a porta COMi e fica esperando uma sequência especial antes de iniciar a recepção das leituras do MPU. O Arduino após entrar novamente na opção correta, transmite a sequência especial, por exemplo 0x55 e 0xAA e em seguida inicia o envio dos dados.