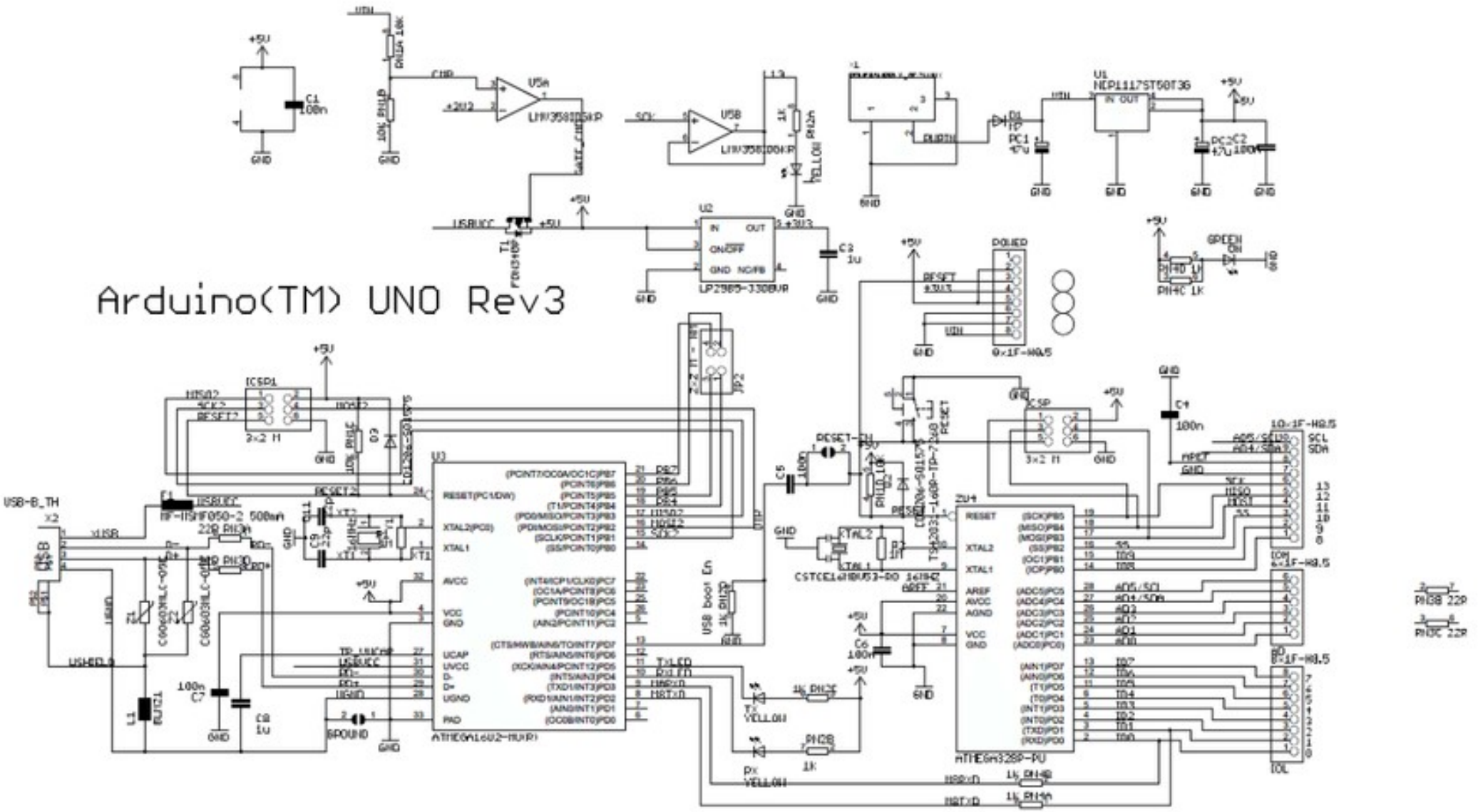
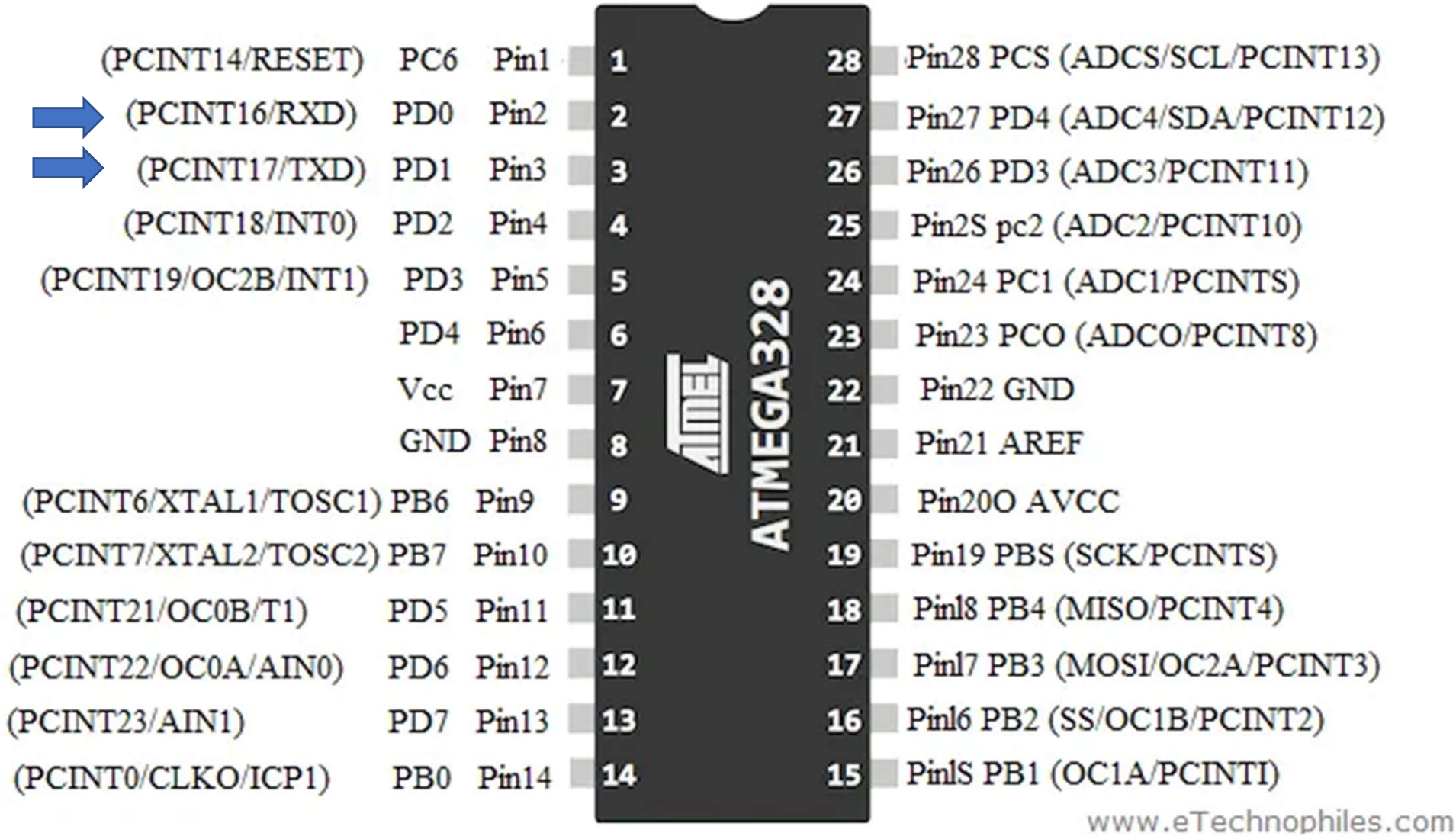
Comunicação Serial USART

Sistemas Microcontrolados

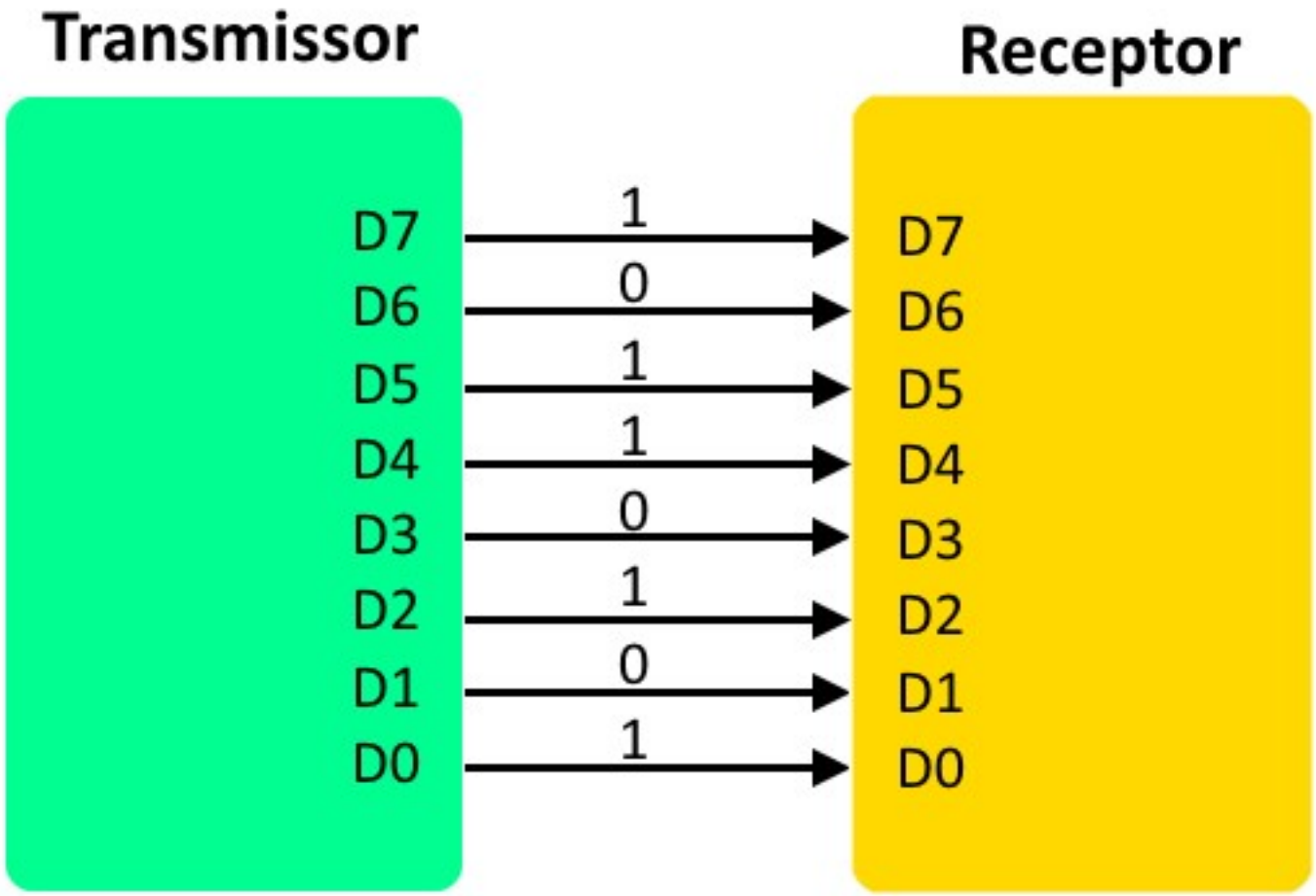
Prof. Marcos Chaves

Fonte: Avr Técnicas de Projetos
Programação de Sistemas Embarcados

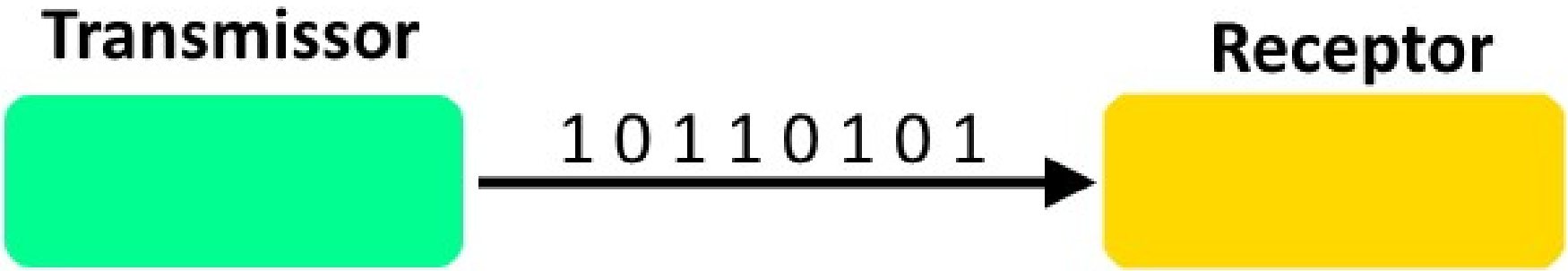

Arduino(TM) UNO Rev3



Comunicação Paralela

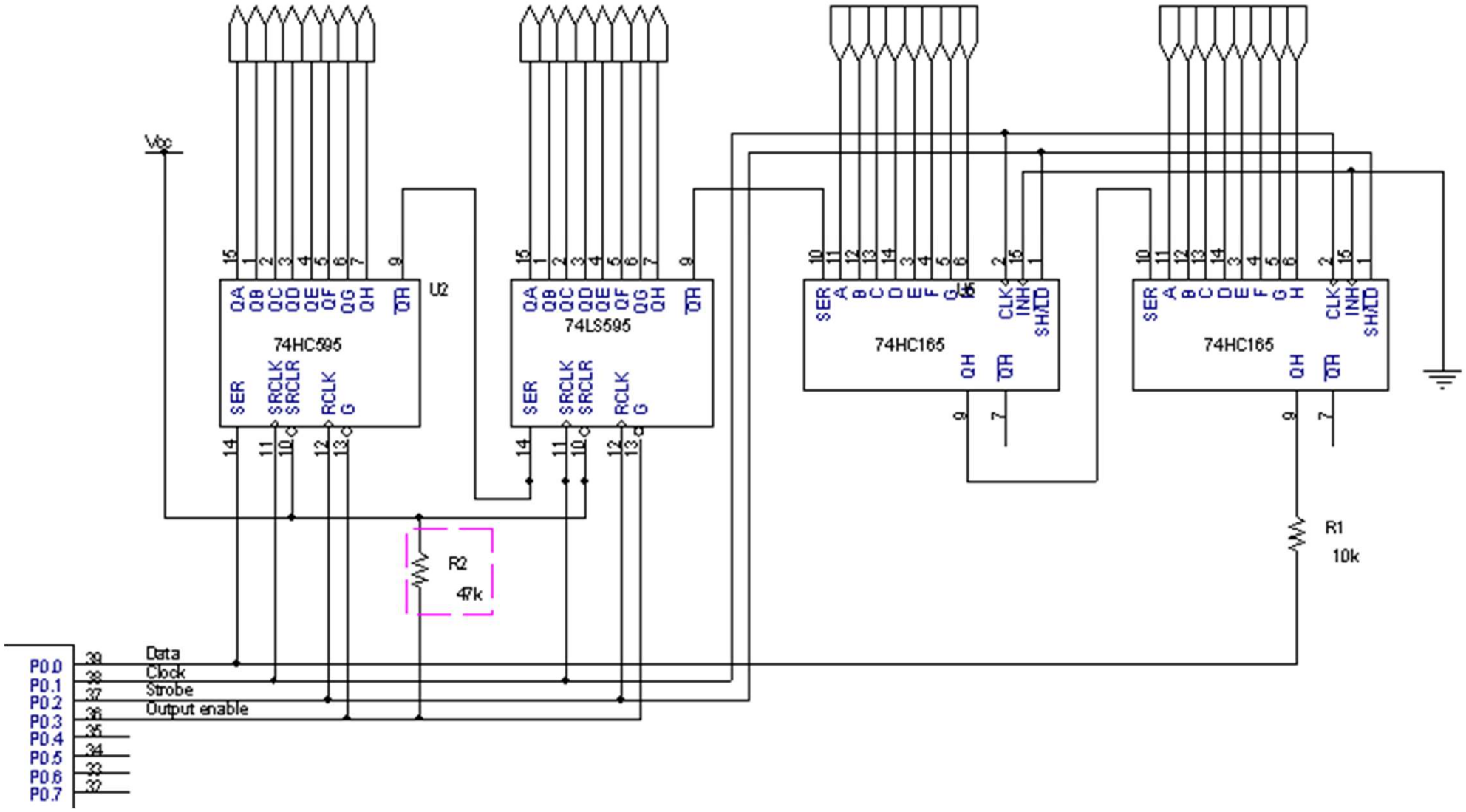


Comunicação Serial



Comunicação Serial

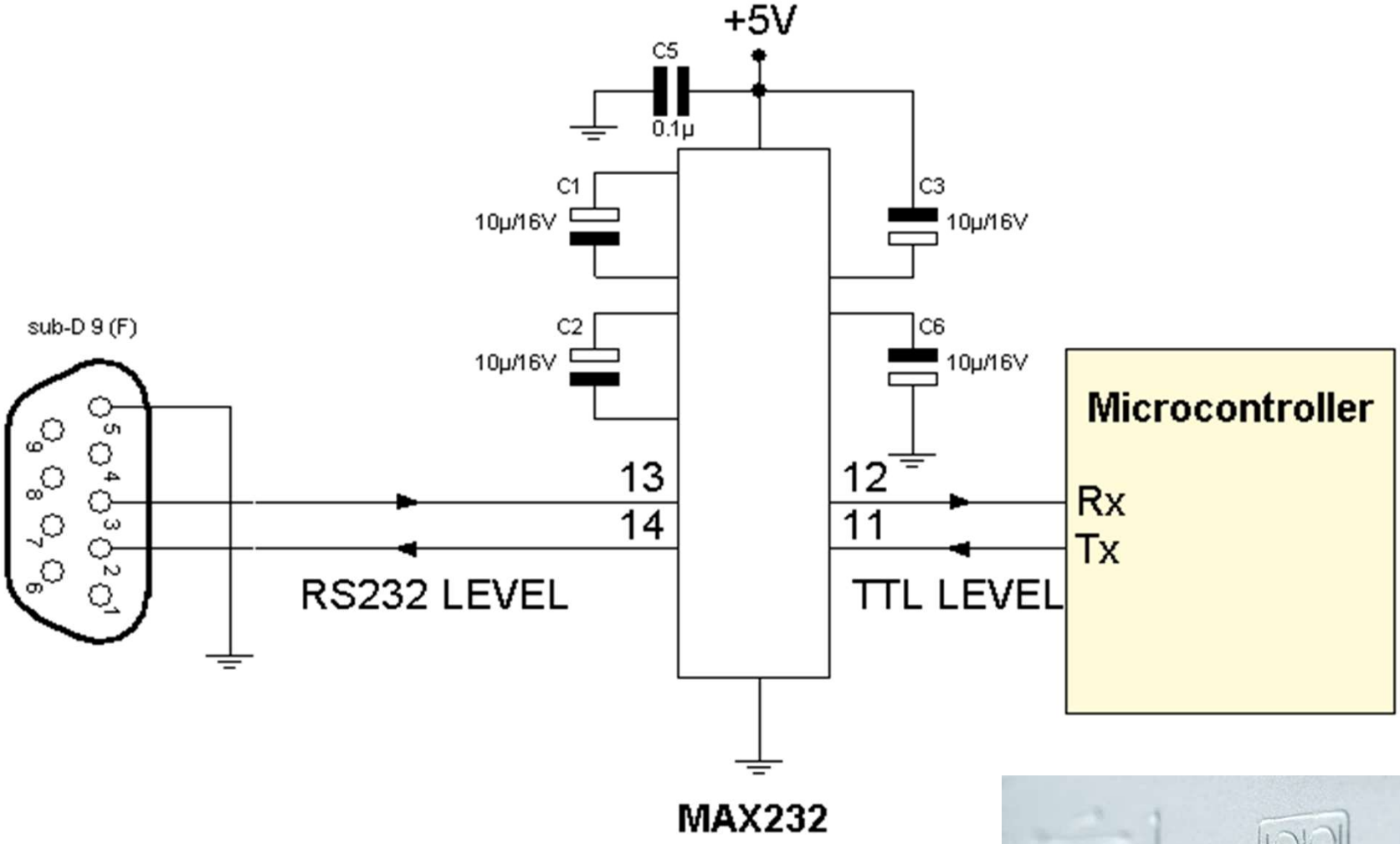
- Vantagens
  - Simplificação do hardware
  - Aumento da taxa de transmissão devido a redução de problemas de sincronia
  - Economia de cabos
- Desvantagens
  - Necessidade de circuitos extras
  - Maior complexidade no código
- O conversor 74hc595 utilizado para expandir as saídas digitais possui uma comunicação serial



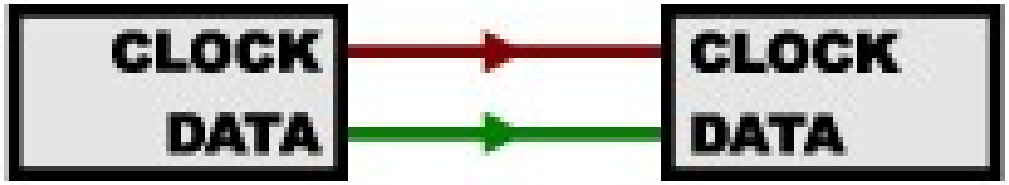
Utilizando 3 pinos, Latch, Clock e Data é possível expandir o número de pinos para saídas CI 74HC595 e entradas 74HC165

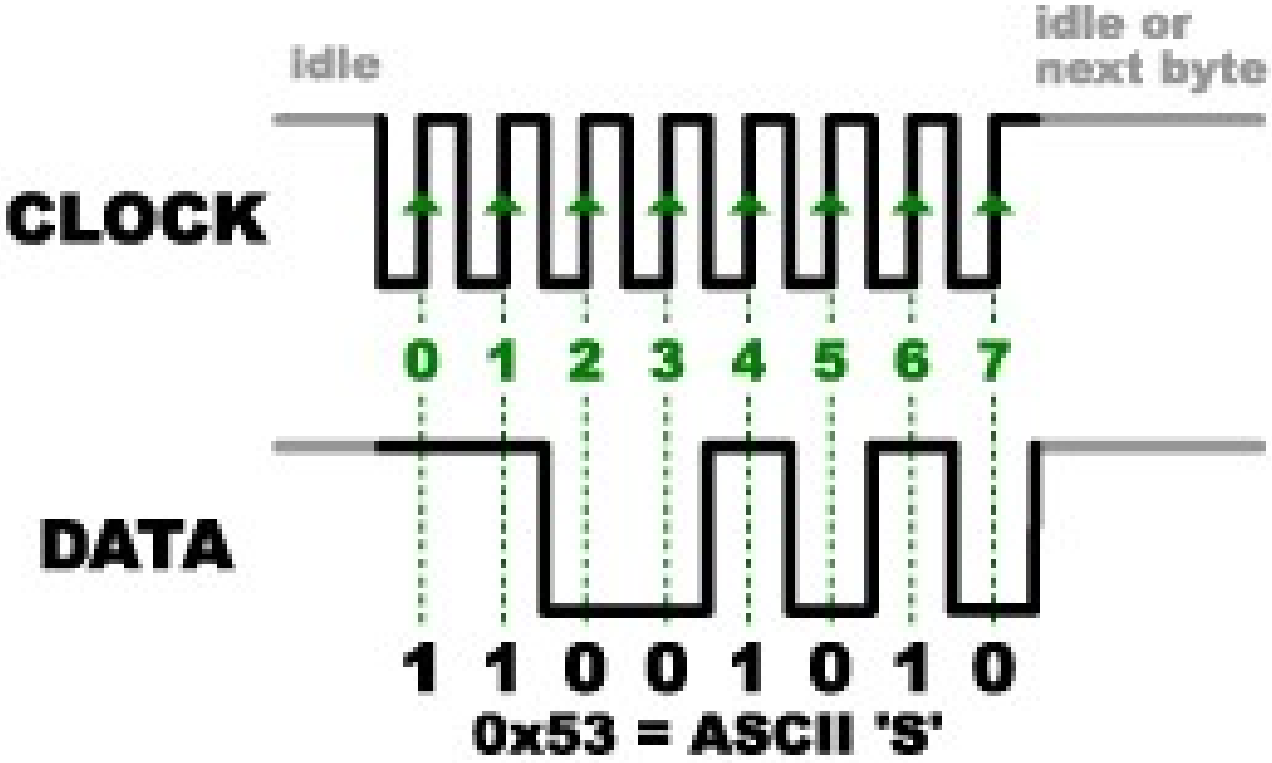
Simulação UNO 8 leds: https://wokwi.com/projects/301188813482361352

Comunicação Serial (RS232)



Comunicação Serial Síncrona e Assíncrona





Principais interfaces/protocolos Seriais

- UART - Universal asynchronous receiver-transmitter
- RS-232 / RS-422 / RS-485
- 1-Wire
- I2C - Inter integrated circuit
- I3C (2017)
- SPI - Serial Peripheral Interface
- CAN - Controller Area Network
- *Bit banging**

- ***USB - Universal Serial Bus***
- ***Thunderbolt***
- ***Ethernet****

Padrões de comunicação existentes

Protocolo	Taxa (bits/s)	Taxa (bytes/s)
Serial MIDI	31.25 kbit/s	3.9 kB/s
Serial EIA-232 max.	230.4 kbit/s	28.8 kB/s
Serial UART max	2.7648 Mbit/s	345.6 kB/s
I2C	3.4 Mbit/s	425 kB/s
Serial EIA-422 max.	10 Mbit/s	1.25 MB/s
SPI Bus (Up to 100MHz)	100 Mbit/s	12.5 MB/s
USB super speed (USB 3.0)	5 Gbit/s	625 MB/s
HDMI v. 1.3	10.2 Gbit/s	1.275 GB/s
Ultra DMA ATA 133 (paralelo 16 bits)	1,064 Mbit/s	133 MB/s
Serial ATA 3 (SATA-600)	6,000 Mbit/s	600 MB/s
Ultra-640 SCSI (paralelo 16 bits)	5,120 Mbit/s	640 MB/s
Serial Attached SCSI (SAS) 3	9,600 Mbit/s	1,200 MB/s

Registradores para comunicação serial ATmega328P

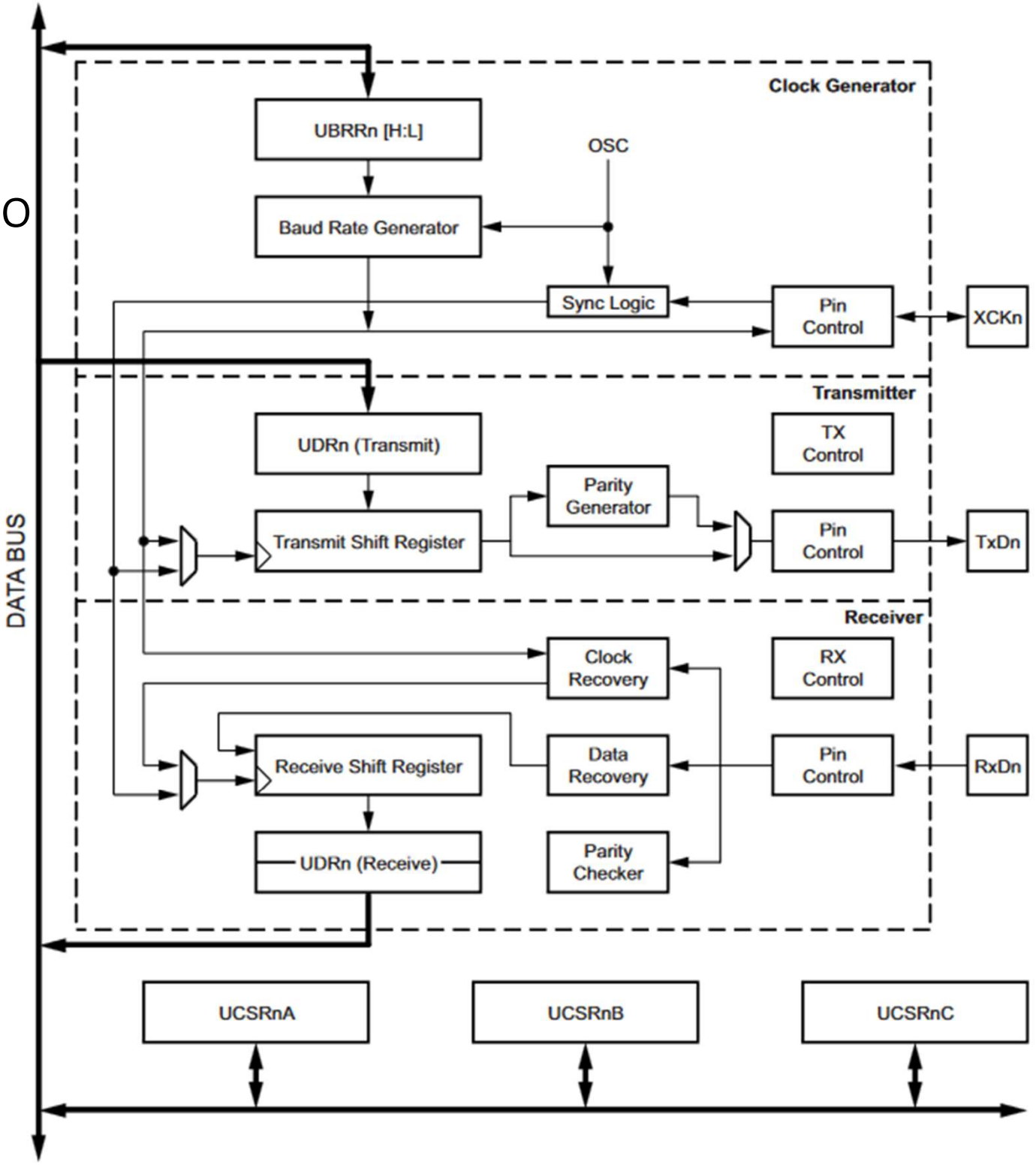
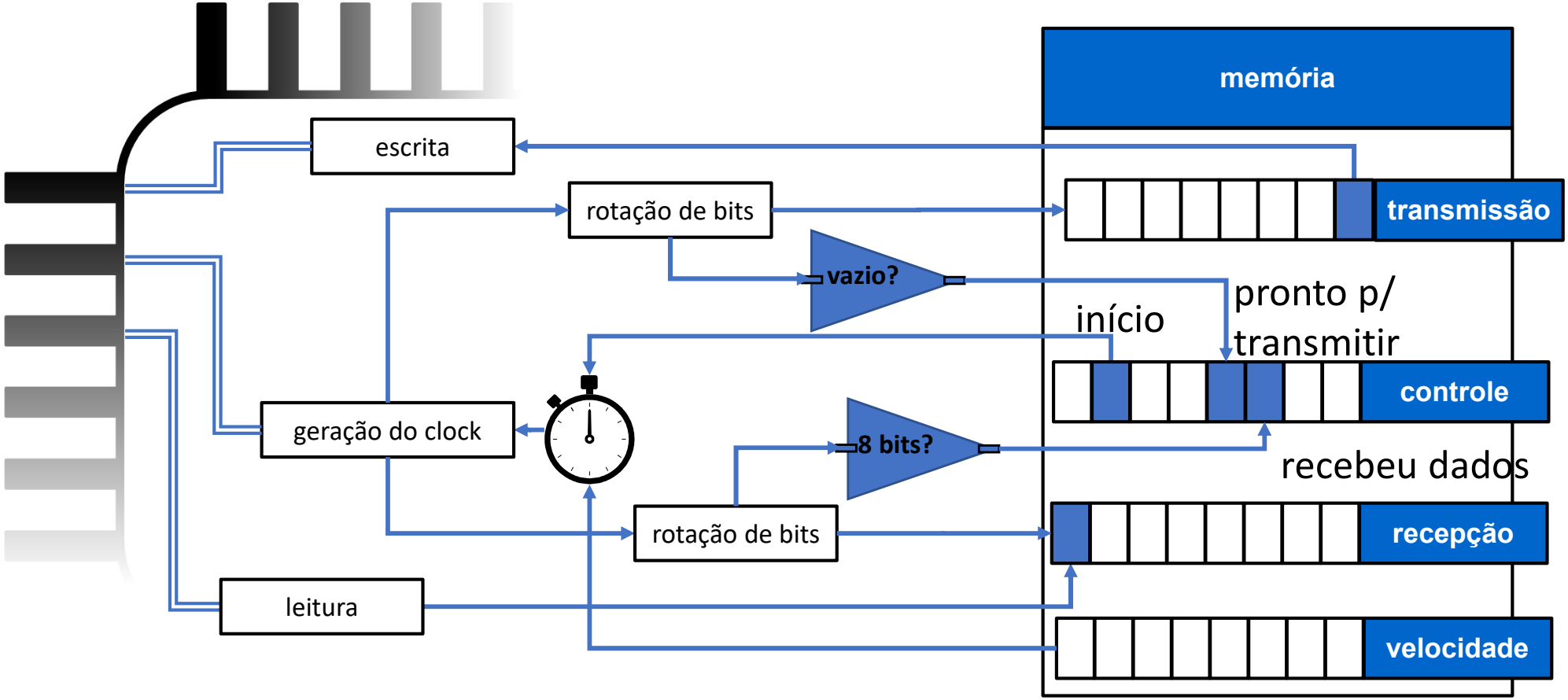


Table 19-11. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	f_{osc} = 8.0000MHz				f_{osc} = 11.0592MHz				f_{osc} = 14.7456MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	207	0.2%	416	–0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	–0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	–0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	–3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	–7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	–3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	–7.8%	5	–7.8%	3	–7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	–7.8%	1	–7.8%	3	–7.8%
1M	–	–	0	0.0%	–	–	–	–	0	–7.8%	1	–7.8%
Max.(1)	0.5Mbps		1Mbps		691.2kbps		1.3824Mbps		921.6kbps		1.8432Mbps	

Note: 1. UBRRn = 0, error = 0.0%

Modelo de funcionamento

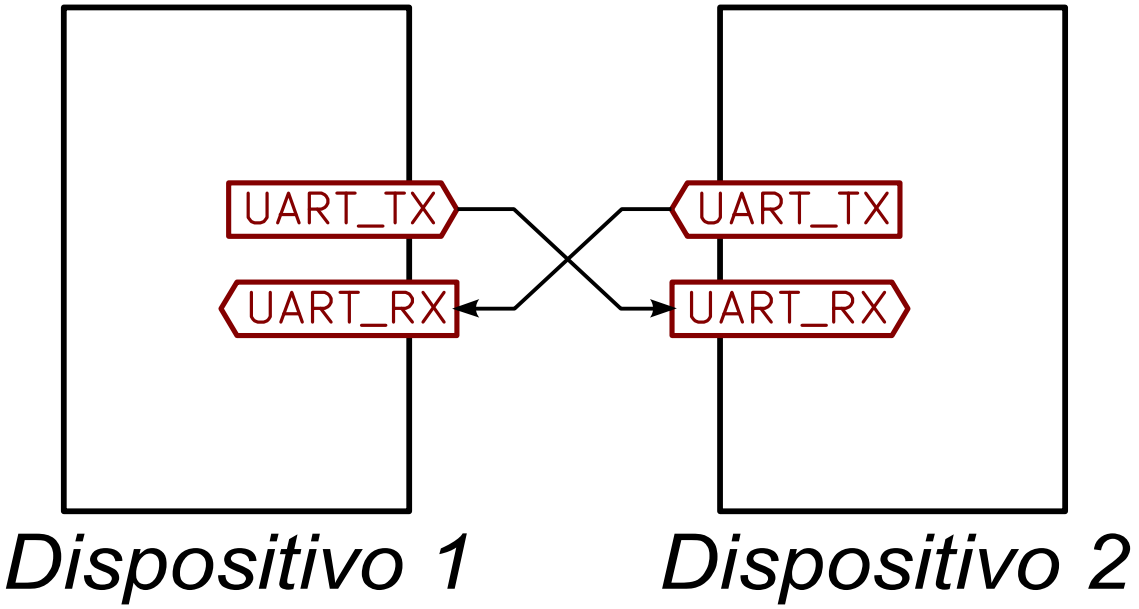


RS232

Protocolo TIA RS232

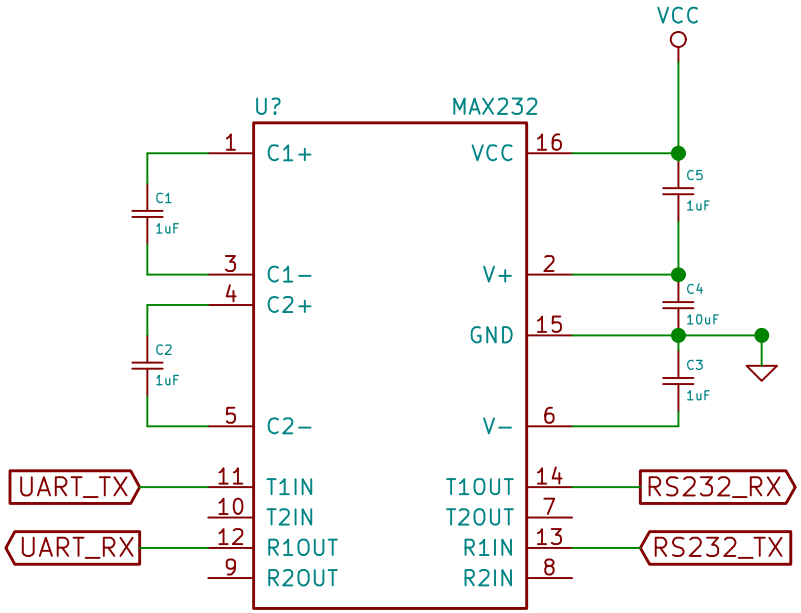
RS232 versus UART

- Similaridades, diferenças e observações:
  - UART é uma versão generalizada dos protocolos 232, 485, 422, EIA.
  - Visa fornecer uma base padrão para todos estes protocolos
  - As tensões são em geral no mesmo nível do processador
- As conversões para os protocolos ficam a cargo de chips externos
  - MAX232 (ICL232, ST232, ADM232, HIN232)
  - FT232
  - SP3485

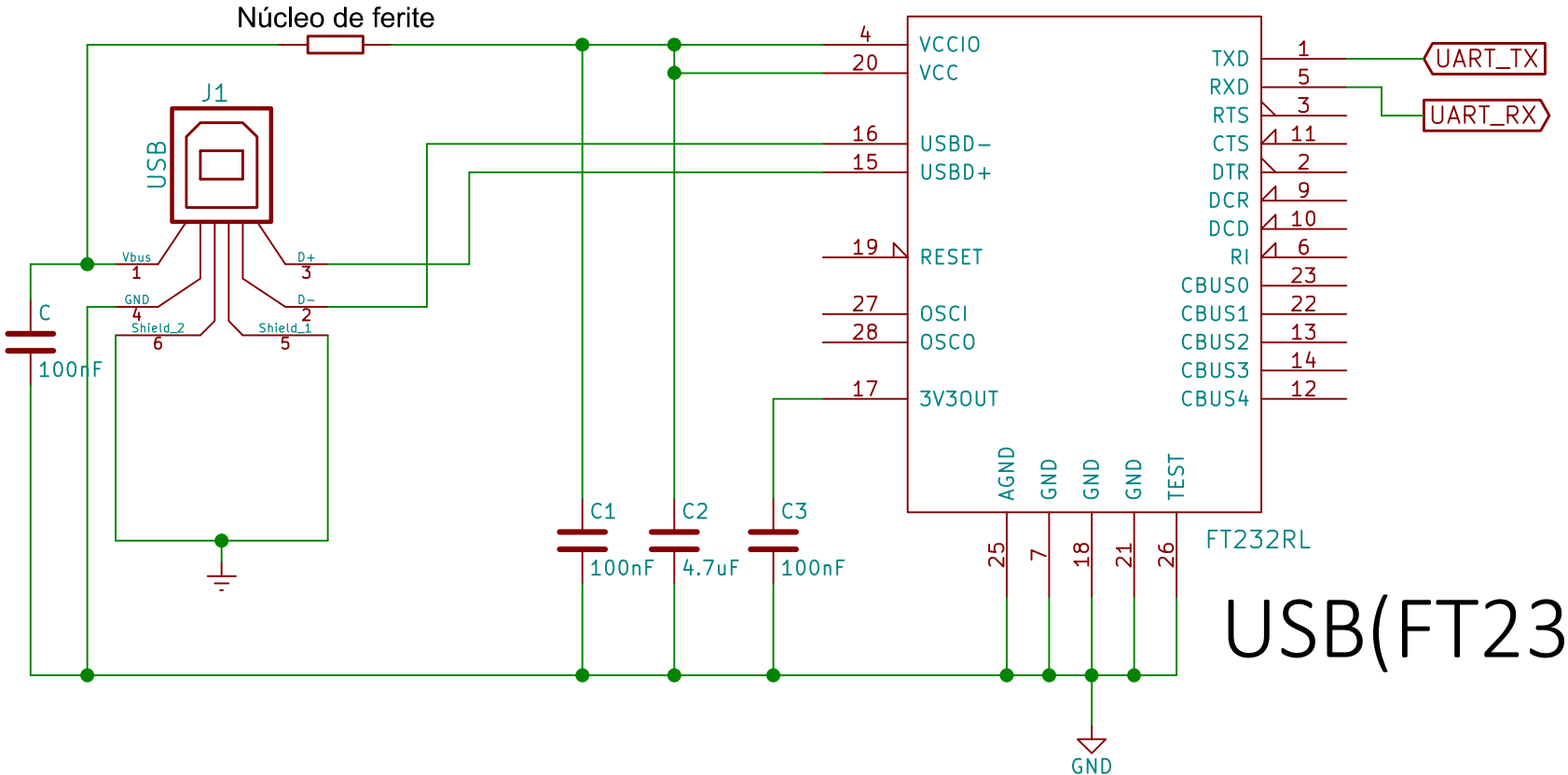


Transceptores Seriais

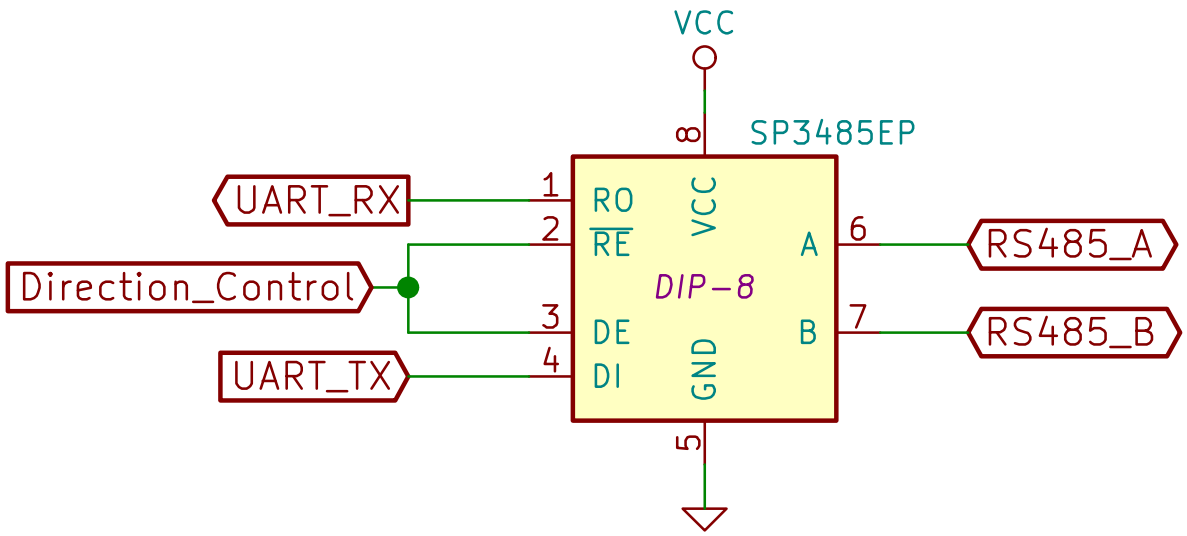
Uart(MAX232)



USB(FT232RL)

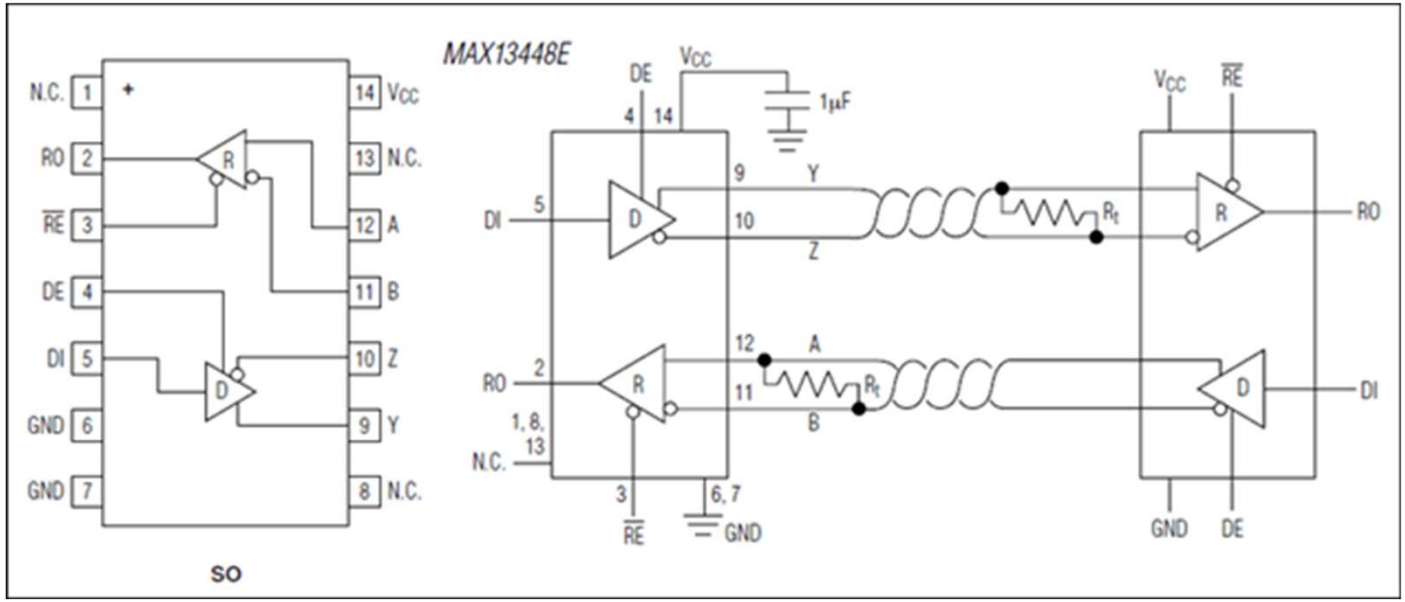


Transceptores Seriais

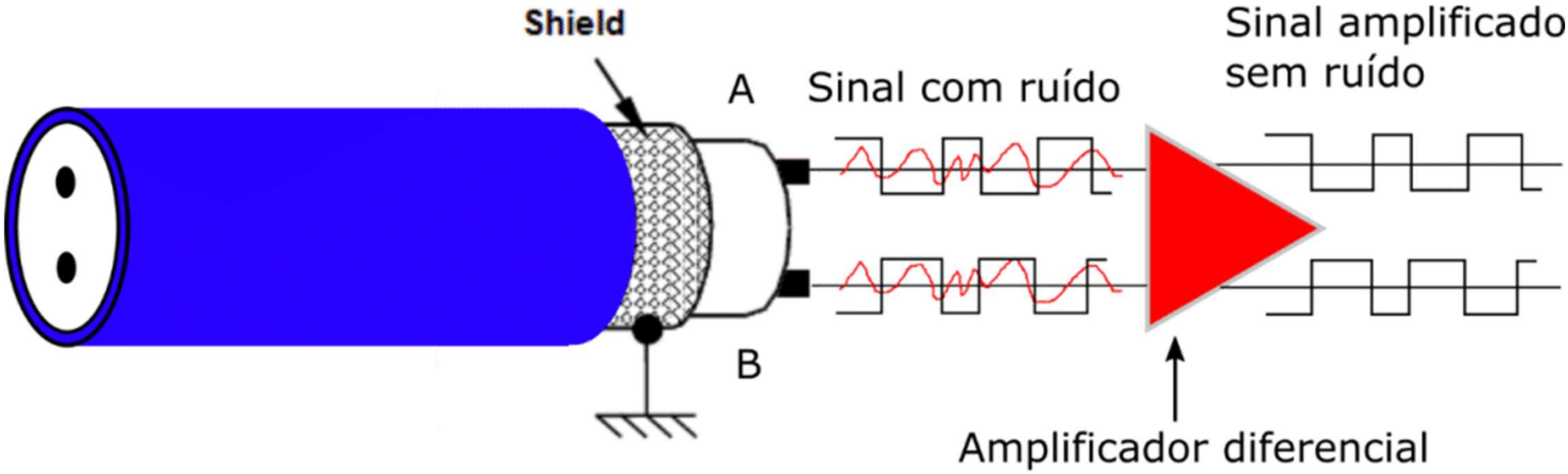


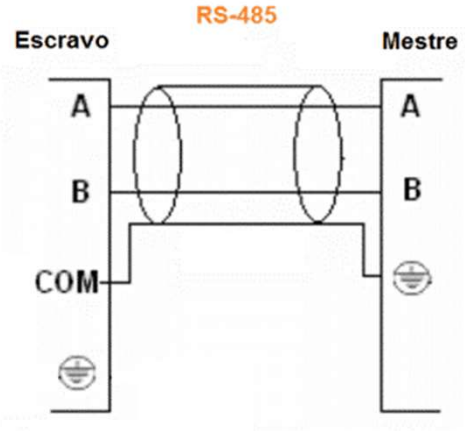
RS485(MAX485)
Half-Duplex)

RS422(MAX1344)
Full-Duplex)



Meio Físico par trançado (rs485, rede..)

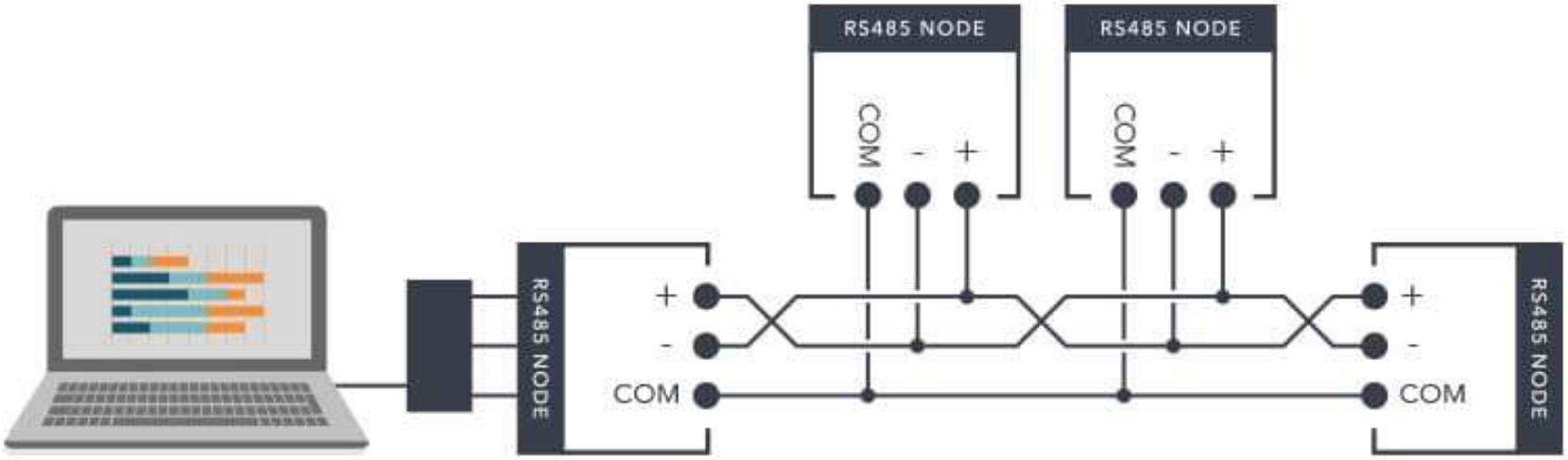


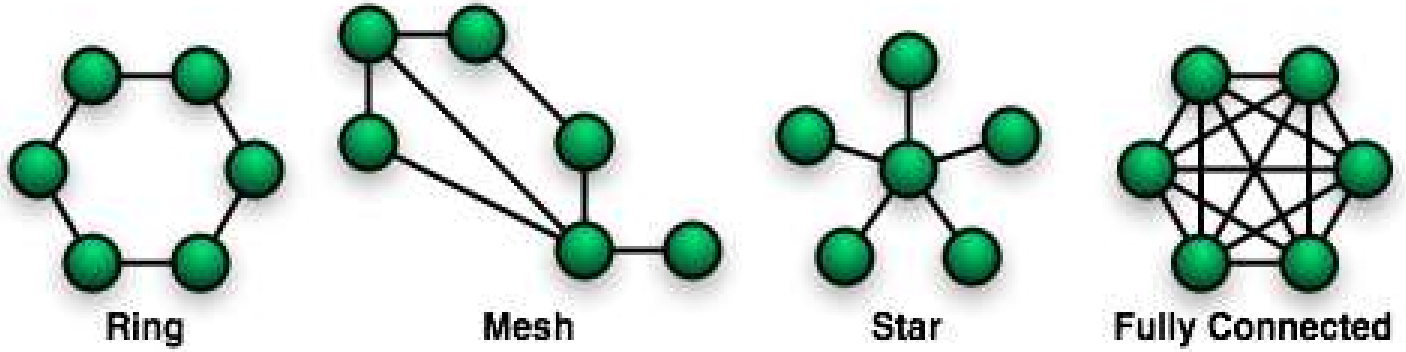


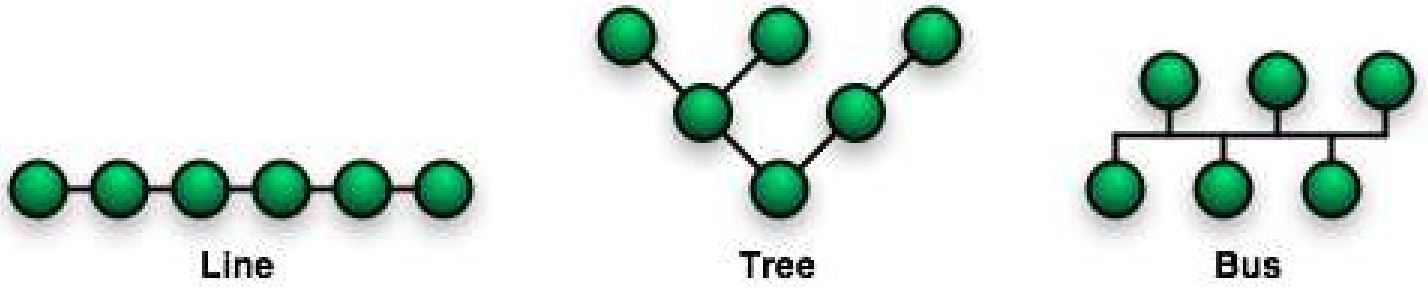
sinal é diferencial,
não é obrigatória
a utilização do fio
terra

https://embarcados.com.br/redes-de-comunicacao-em-rs-485/

Meio Físico e topologias







O padrão RS232 (base do UART)

- Comunicação serial 232
  - 1962 - Recommended Standard 232
  - 1969 - Recommended Standard 232 rev. C
  - 1986 - EIA 232 rev C
  - 1997 - TIA 232 rev F
- Procedimento de envio
  - Um bit de marcação de início de mensagem
  - Os 8 bits de dados
    - Primeiro o menos significativo. Alguns sistemas mais antigos operam com 5 ou 7 bits por vez
  - Um bit de paridade (opcional)
  - Um bit de marcação de fim de mensagem

O padrão RS232

- Para dois dispositivos se comunicarem é necessário compatibilizar o sistema físico e o sistema lógico.
- Físico (Hardware)
  - Tipo de conector
  - Pinagem do conector
  - Níveis de tensão
- Lógico (Software)
  - Codificação utilizada
  - Tamanho da palavra
  - Tamanho do stop/start bits
  - Bit de paridade
  - Taxa de transmissão

Aspectos da comunicação serial

- **Taxa de transmissão (Baud-Rate)**

- Taxa a ser usada na transmissão dos dados em bits por segundo (bps) 1200, 2400, ..9600, 57600, 115200)
- Depende do meio físico e suscetível a ruído

- **Tamanho da palavra (Framing)** (5, 6, 7, 8..)

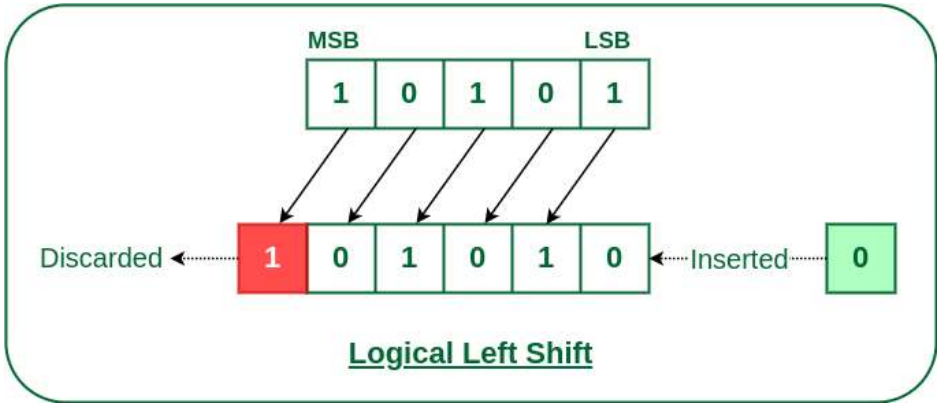
- **Sincronia** (utilizados no inicio e final de cada palavra

- **Controle de Erro**

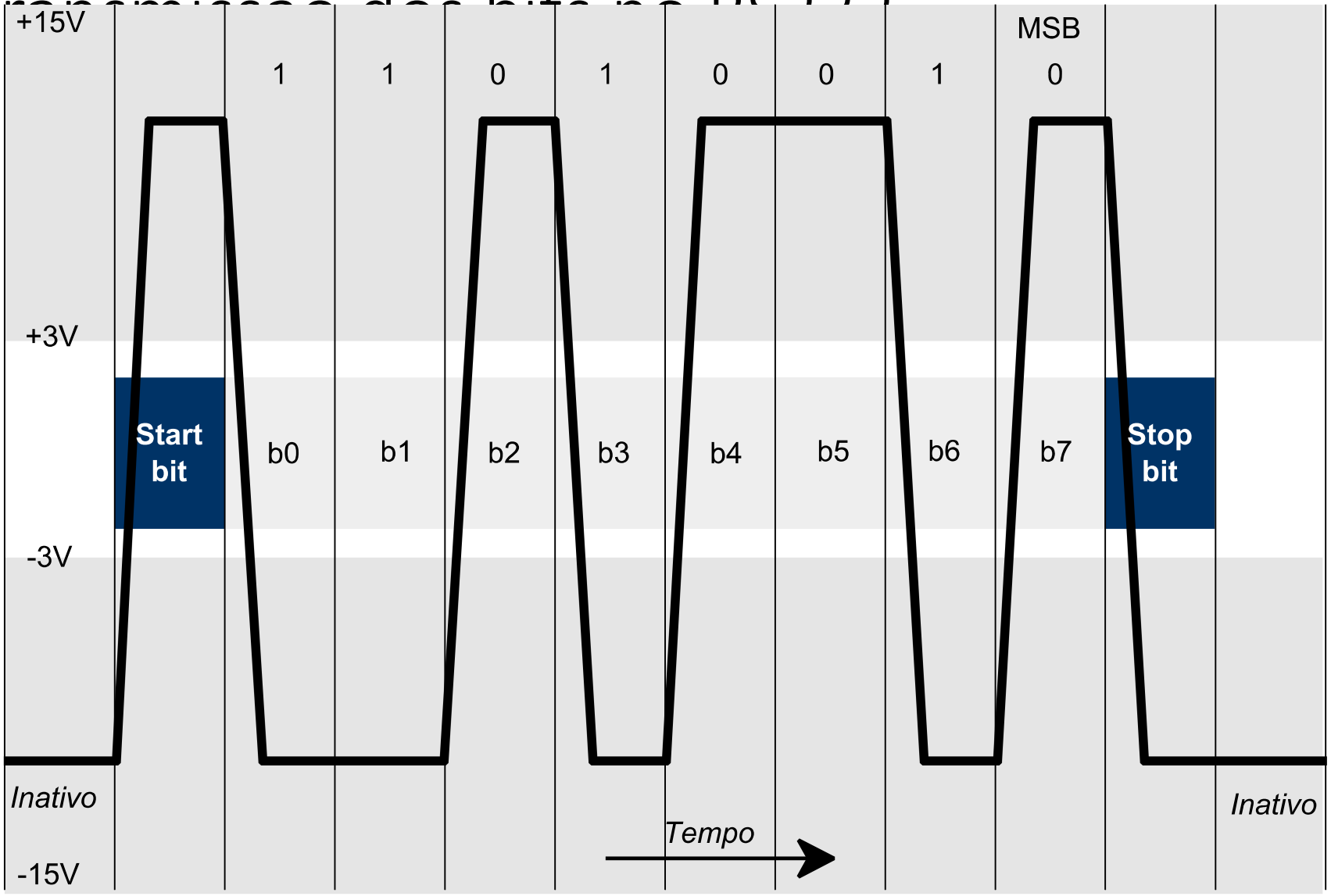
- SNR (relação sinal-ruído)
- Mecanismos de verificação (bit de paridade)

A transmissão de dados

- O envio de um dado funciona como uma operação de bit-shift.
- A primeira transmissão é conhecida como start bit
  - O start bit é sempre uma transição do sinal negativo para o positivo.
- A transmissão começa pelo bit menos significativo.
  - Este bit permanece estável durante um determinado tempo, baseado na velocidade de transmissão.
- Realiza-se um shift para direita e o ``novo'' bit menos significativo é enviado.
- Repete-se a operação oito vezes, uma para cada bit.
- Por fim é enviado um último bit indicando o fim da transmissão.



Transmissão dos bits no RS232



Transmissão apoiada por hardware

- É possível criar as rotinas de transmissão e recepção por software
- A utilização de um periférico dedicado soluciona esses problemas
- A rotina de inicialização é geralmente feita em 5 passos
  - 1 - Desligar o hardware antes de configurá-lo
  - 2 - Configuração dos terminais
  - 3 - Configuração do oscilador da serial (sistema de temporização)
  - 4 - Configuração o valor do baudrate
  - 5 - Ligar o transmissor e o receptor

Criação da biblioteca de UART em C

Biblioteca

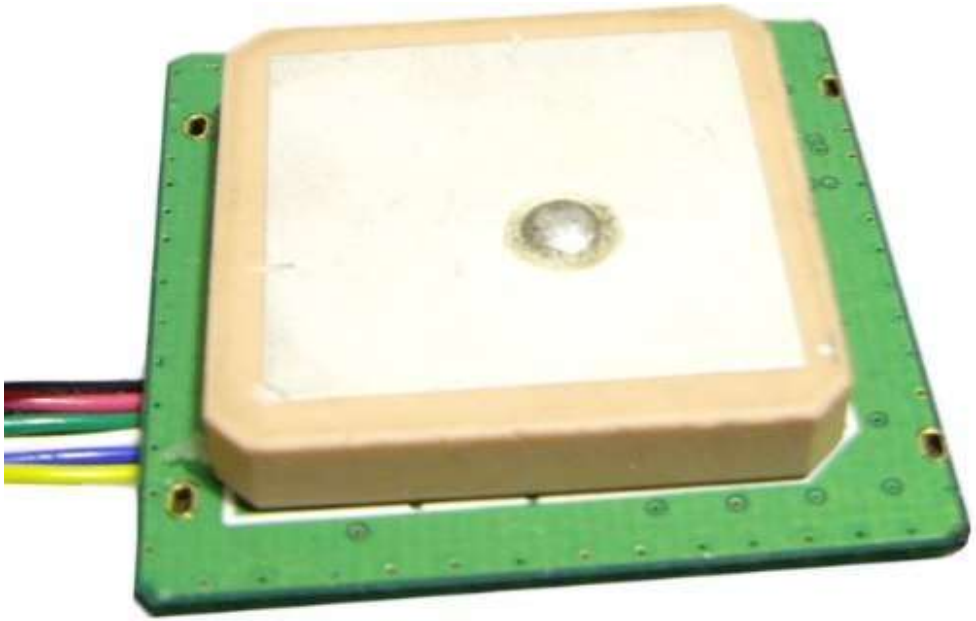
Biblioteca

- Utilizando um hardware dedicado o processo de envio/recepção é simplificado

```
void serialSend(char c) {tido num registo dedicado
    while (!(UART0_S1_REG & UART0_S1_TDRE_MASK));
    UART0_D_REG = c;
}
char serialRead(void) {
    //Verificar se há algo disponível
    if ((UART0_S1_REG & UART0_S1_RDRF_MASK)) {
        //Lê o registro da serial
        return UART0_D_REG ;
    } else {
        //Código para "não há caracter disponível"
        return 0xff;
    }
}
```

Comunicando através da UART/RS232

Exemplo de interface com dispositivo UART



Receptor GPS

Exemplo de interface com dispositivo

- GPS integrado com antena ME-1000RW
  - Comunica-se usando padrão UART 9600 bps
  - Os dados são passados através de uma string
  - Implementa vários padrões NEMA
    - GPGSA - Precisão e quantidade de satélites ativos
    - GPGCA - Dados fixados (lat/long/alt/diferencial)
    - GPRMC - Dados mínimos (lat/long/alt)
    - GPVTG - Velocidade e deslocamento
    - GPGSV - Dados dos satélites visíveis
  - Cada um destes padrões possui uma espécie de protocolo que deve ser decodificado

Mensagem do GPRMC

- Formada por 14 campos:
  - 0 - ID da mensagem
  - 1 - Tempo (UTC)
  - 2 - Rastreando (R) ou Aceitável (A)
  - 3 - Latitude
  - 4 - Norte/Sul
  - 5 - Longitude
  - 6 - Leste/Oeste (E/W)
  - 7 - Velocidade (magnitude)
  - 8 - Velocidade (ângulo)
  - 9 - Data (UTC)
  - 10 - Variação Magnética (ângulo em graus)
  - 11 - Variação Magnetica (direção, E/W)
  - 12 - Modo de operação (N não válido, A autônomo, D diferencial, E estimado, M manual, S simulação)
  - 13 - Checksum

Mensagem do GPRMC

$ID,000000,X,0000.00,X,00000.00,X,000.0,000.0,000000,000.0,????

- Formato:

$GPRMC,220516,A,5133.82,N,00042.24,W,173.8,231.8,130694,004.2,W*70

- Exemplo

Comunicação

- Rotina para recepção dos dados
  - Armazenar cada byte recebido num buffer temporário.
  - Aguardar dois bytes de fim de linha: <cr> e <lf>, também descritos em ASCII como '\n' e '\r', nas posições 13 e 10.
  - Caso chegar "fim de linha", resetar o buffer e processar a linha
  - No processamento do buffer
  - Verificar o ID da mensagem (\$GPRMC)
  - Cada campo é separado por vírgulas
  - Percorrer o campo atual e realizar conversão dos dados ou enviar p/ o LCD
- Não esquecer de verificar o checksum para segurança

Armazenamento do buffer

```
#include "serial.h"
#include "lcd.h"
void main(void) {
     unsigned char pos=0; //posição atual do buffer
     char buffer[100]; //buffer temporário
     char data;
     serialInit();
     SPBRG = 207; //configura para 56k
     lcdInit();
     for(;;) {
          data = serialRead(); //recebimento dos dados
          if (data != 0) {
               buffer[pos] = data;
               pos++;
               if (pos >= 100) { pos = 0; }
          }
          // Processamento do buffer
     }
}
```

Verificação de início de mensagem

```
//teste de fim de mensagem
if((pos > 2) &&
       (buffer[pos-2] == 13) && //CR
       (buffer[pos-1] == 10)) { //LF
       pos = 0;
       //se o ID estiver correto processa o buffer
       if( (buffer[0] == '$') &&
                     (buffer[1] == 'G') &&
                     (buffer[2] == 'P') &&
                     (buffer[3] == 'R') &&
                     (buffer[4] == 'M') &&
                     (buffer[5] == 'C'))
       {
              // parser da mensagem $GPRMC
              // ...
       }
}
```

```
//começa depois do campo ID
pos = 7;

//1 - Hora UTC
while(buffer[pos] != ','){
	//EnviaDados(buffer[pos]);
	pos++;
}
pos++;//pula a vírgula

//2 A(OK)
while(buffer[pos] != ','){
	//EnviaDados(buffer[pos]);
	pos++;
}
pos++;//pula a vírgula
```

Parsear dados

```
//3-Latitude

lcdCommand(0x80);
lcdChar('L');
lcdChar('A');
lcdChar('T');

while(buffer[pos] != ','){
	lcdChar(buffer[pos]);
	pos++;
}
pos++;//pula a vírgula
//4 N/S
while(buffer[pos] != ','){
	lcdChar(buffer[pos]);
	pos++;
}
pos++;//pula a vírgula
```

```
//5-Longitude dados

lcdCommand(0xC0);
lcdChar('L');
lcdChar('O');
lcdChar('N');

while(buffer[pos] != ','){
      lcdChar(buffer[pos]);
      pos++;
}
pos++;//pula a vírgula
//6-Leste(E)/Oeste(W)
while(buffer[pos] != ','){
      lcdChar(buffer[pos]);
      pos++;
}
pos++;//pula a vírgula
//..continua para os demais campos
```