Comunicação Serial

Prof. Dr. Roberto Kenji Hiramatsu

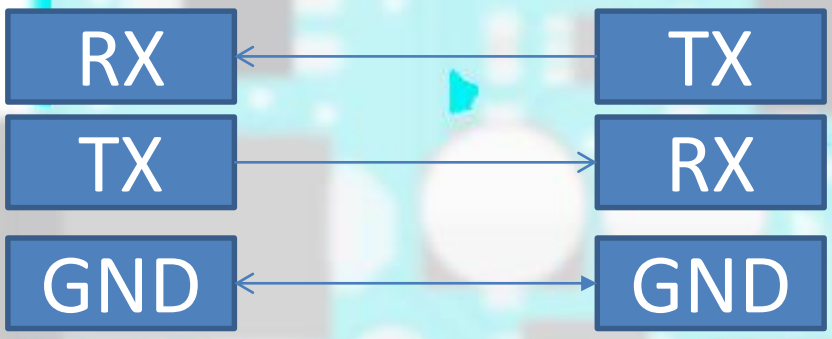
Prof. Dr. João Henrique Correia Pimentel

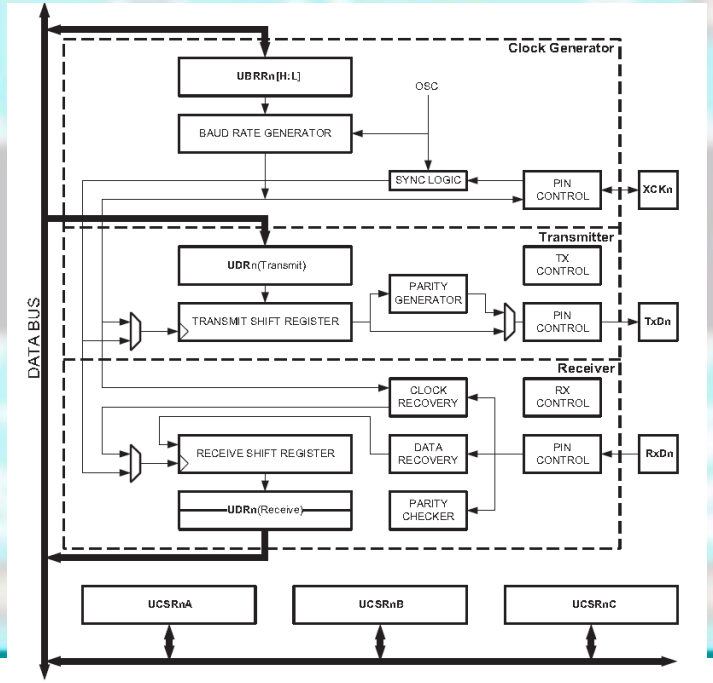
Comunicação assíncrona

- BAUD – Taxa de transmissão de sinais que também define a taxa que deve ser trabalhada a o processamento da recuperação do sinal
- Na assíncrona o sinal precisa que a taxa para transmissão seja configurada e que a diferença de relógio do que transmite em relação ao do que recebe seja menor que 2%
- Algumas padrões de comunicação serial são os RS-232, RS-485 , RS-422, CAN BUS, USB

Conexão física

- Simplex – somente vai numa direção
- Half-Duplex – Tem somente 1 linha em que transmiti os dados
- Full-Duplex – Conversa simultaneamente nos dois sentidos.





Comunicação assíncrona

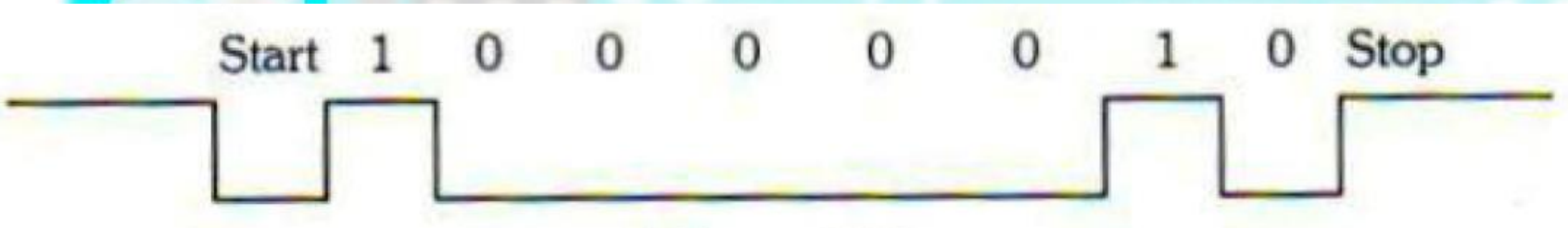


Table 20-7. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2Xn = 0		U2Xn = 1								2Xn = 1	
	UBRRn	Error	UBRRn	Error							n	Error
2400	416	-0.1%	832	0.0%								0.0%
4800	207	0.2%										0.0%
9600		0.2%	207	0.2%								0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%

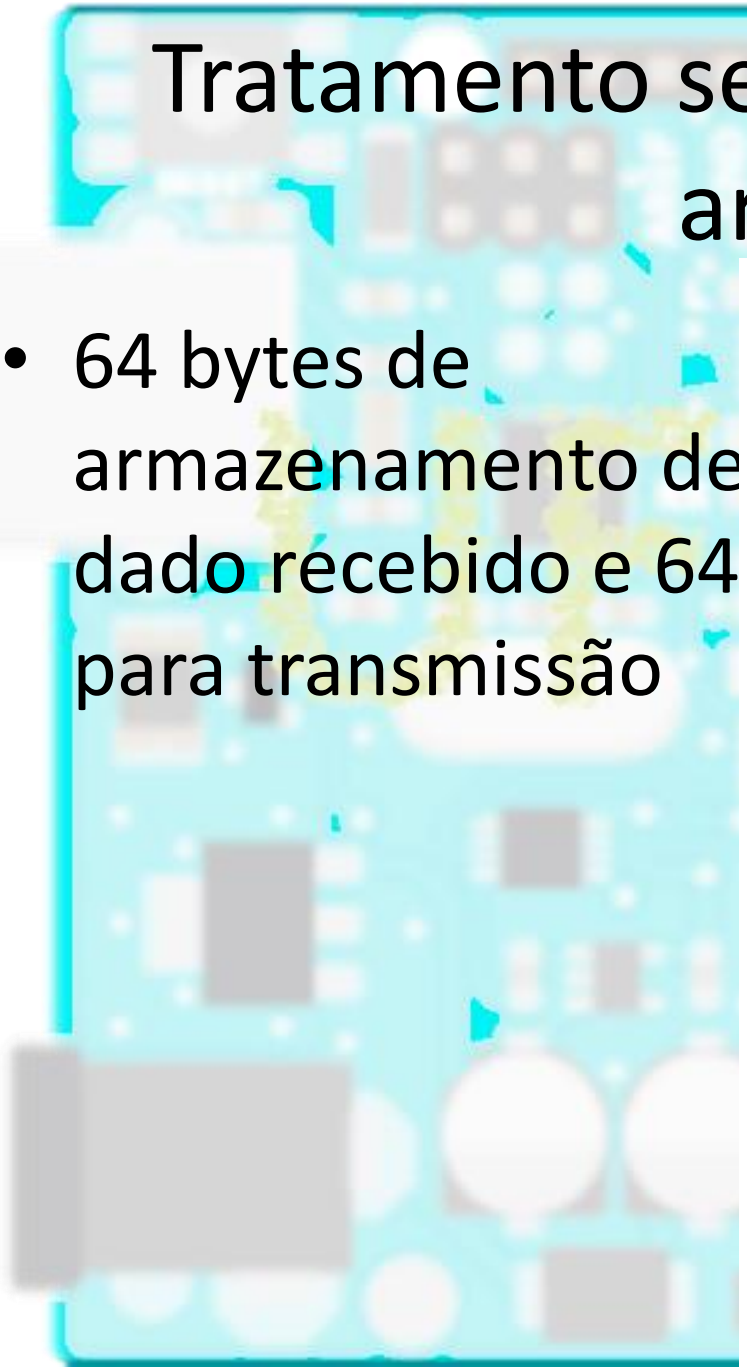
104us por bit e 1,04ms por caractere em 8bits sem paridade e 1 stop bit

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Tratamento serial na plataforma arduino

- 64 bytes de armazenamento de dado recebido e 64 para transmissão



LANGUAGE

FUNCTIONS

VARIABLES

STRUCTURE

LIBRARIES

GLOSSARY

The Arduino Reference text is licensed under a Creative Commons Attribution-Share Alike 3.0 License.

Find anything that can be improved? Suggest corrections and new documentation via

Functions

If (Serial)
available()
availableForW
begin()
end()
find()
findUntil()
flush()
parseFloat()
parseInt()
peek()
print()
println()

Recebendo letras pela serial

```
void setup(){
  pinMode(13,OUTPUT);
  Serial.begin(9600);
}
void loop(){
  if(Serial.available()){              // verifica se tem dados disponiveis, não bloqueia
     char c=Serial.read();             // faz a leitura de dados
    if(c=='l'){                        //Se for letra 'l' liga led da placa
      digitalWrite(13,HIGH);
    }
    else{                              // Senao desliga
      digitalWrite(13,LOW);
    }
  }
}
```

A serial tem limite para receber e enviar

- Teste uma mensagem muito longa no código abaixo do tipo:
  - 012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
- Capture o máximo que puder quando recebe
- Se precisar enviar muito dado em intervalos muito pequenos e necessário aumentar o baudrate.

Para escria pode se verificar a quantidade disponível com Serial.availableForWrite()

```
void setup(){
  pinMode(13,OUTPUT);
  Serial.begin(9600);
  while(Serial.available()); // somente iniciar o loop quando receber
}
void loop(){
  delay(100);
  if(Serial.available()){   // poderia ser substituído por while
     char c=Serial.read();
     Serial.print(c);
  }
}
```

Tratamento com mensagem terminadas por caractere

```
void setup(){
  Serial.begin(9600);
}
void loop(){
  if(Serial.available()){  // Verifica se tem dado no buffer serial
     char mens[64];
     int n=Serial.readBytesUntil('e',mens,64);  // somente termina a leitura quando encontra
                                                // letra e ou depois de um determinado tempo
     mens[n]='\0';                        // termina o grupo da mensagem necessaria para retorno
     Serial.println(mens);                   // imprime de volta a mensagem
  }
}
```

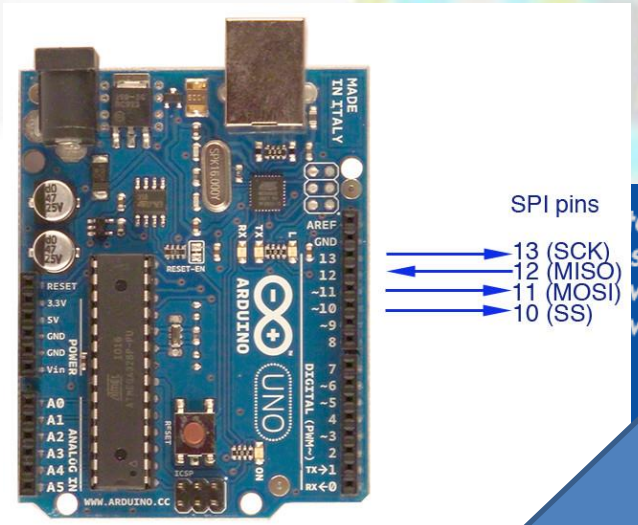
```
1  String inputString = "";          // a string to hold incoming
2  boolean stringComplete = false;   // whether the string is com
3  void setup() {
4    Serial.begin(9600);
5    inputString.reserve(200);
6  }
7  void loop() {
8    if (stringComplete) {
9      Serial.println(inputString);
10     inputString = "";
11     stringComplete = false;
12   }
13 }
14 void serialEvent() {
15   while (Serial.available()) {
16     char inChar = (char)Serial.read();
17     inputString += inChar;
18     if (inChar == '\n') {
19       stringComplete = true;
20   } } }
```

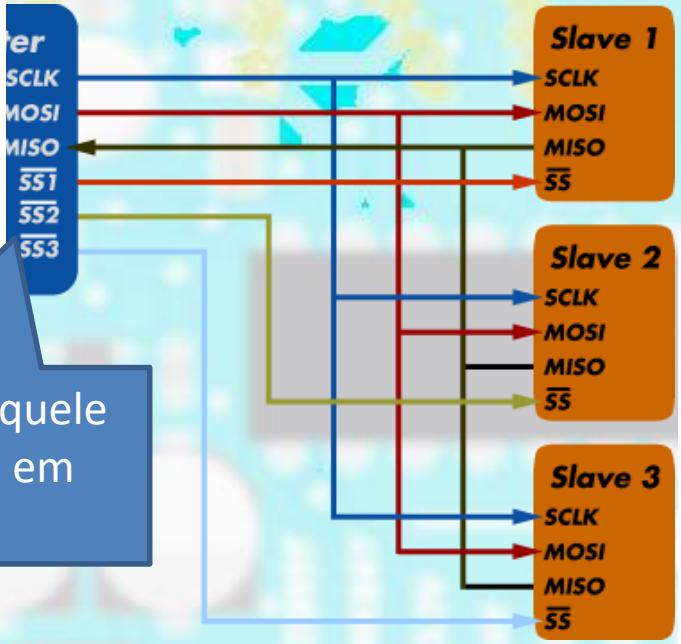
Exemplo de SerialEvent encontrado em:
https://www.arduino.cc/en/Tutorial/SerialEvent

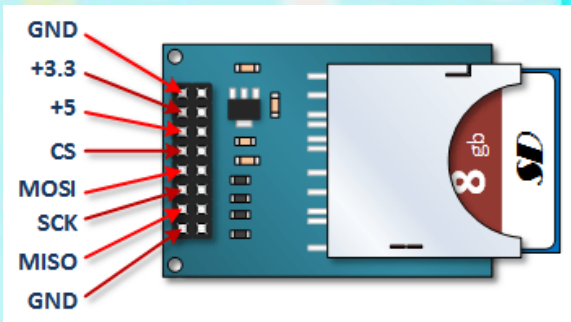
Similar a interrupção externa de pelos pinos do UNO R3. Não precisa registrar uma fun mas o nome não pode ser mudada.

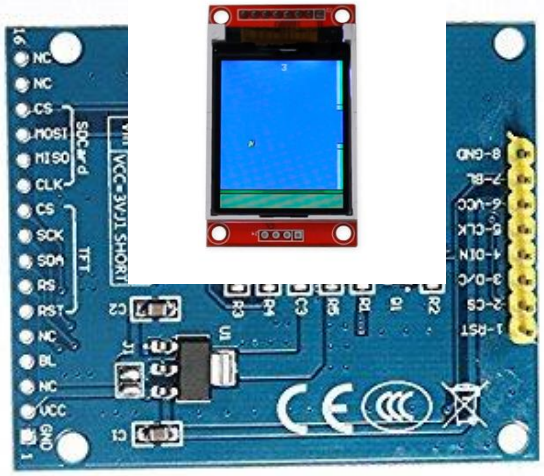
SPI (Serial Peripherical Interface)

- Na síncrona o sinal de relógio é definido pelo mestre.
  - Permite operação Full-Duplex





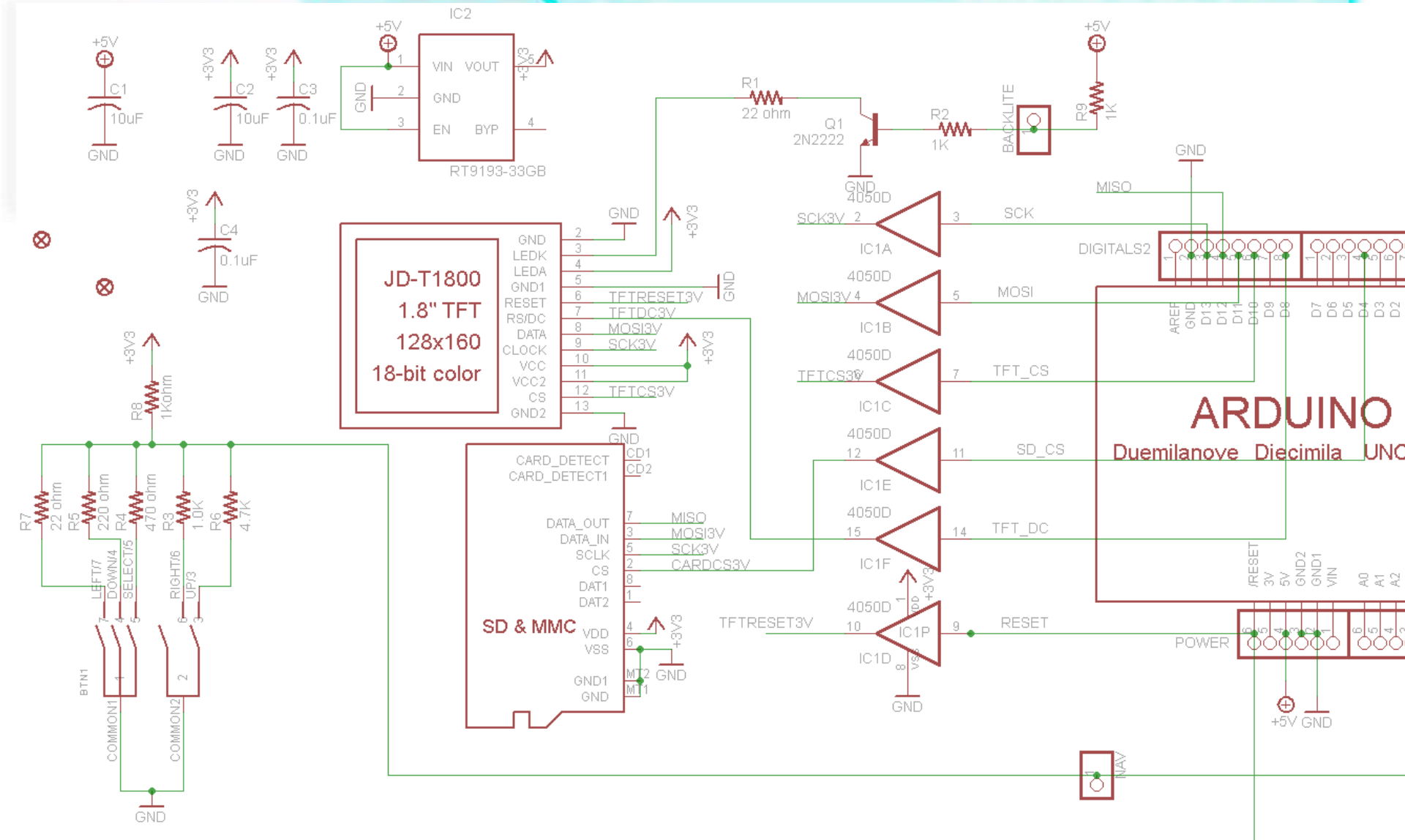




Quem é o escolhido e aquele que esta com ss ativo em baixo.

https://www.totalphase.com/support/articles/200349236

Como é ligado a shield 1.8" TFT Display Breakout and Shield



Modos SPI

- No SPI existem 4 modos de configuração de comunicação. E necessário ver o manual do componente se for trabalhar com dispositivo novo.
- As bibliotecas de módulos para Arduino já escondem esta configuração

Modo	CPOL	CPHA	Descrição
0	0	0	O dado é armazenado na borda de subida do clock.
1	0	1	O dado é armazenado na borda de descida do clock.
2	1	0	O dado é armazenado na borda de descida do clock.
3	1	1	O dado é armazenado na borda de subida do clock.

Tabela 12.1

PEREIRA,2003

SPI leitura

- A leitura de dispositivos segue um padrão iniciado pelo comando, seguindo de

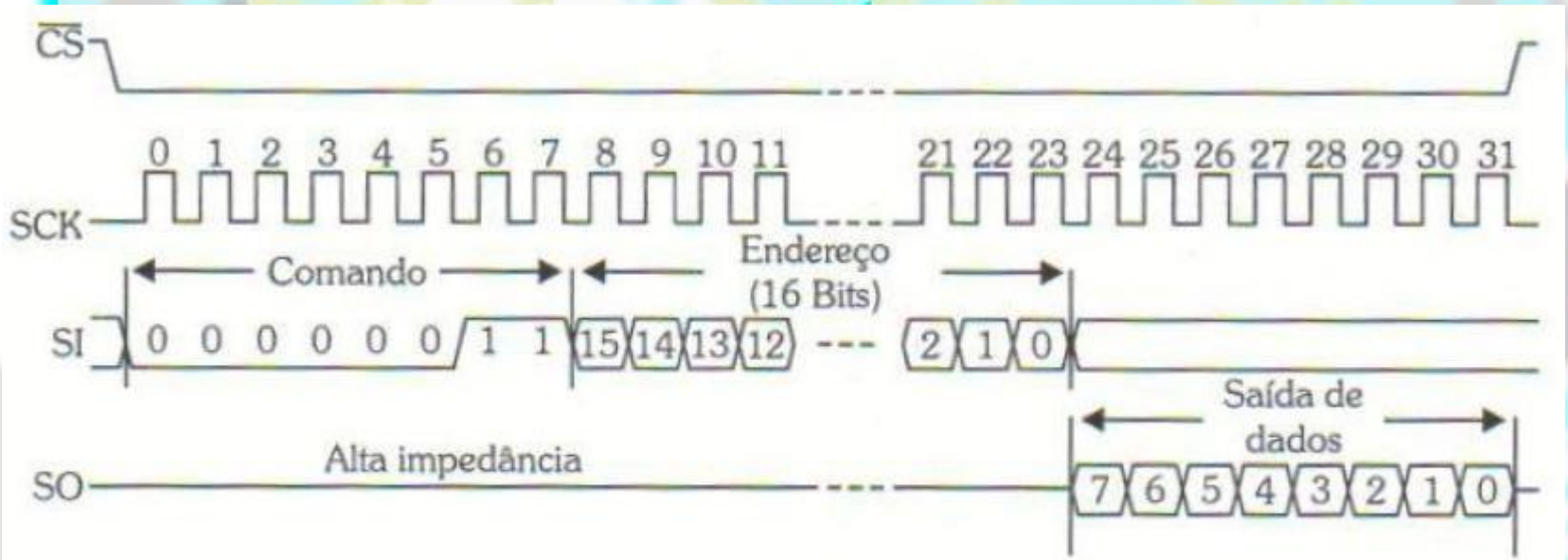


Figura 12.3

SPI escrita

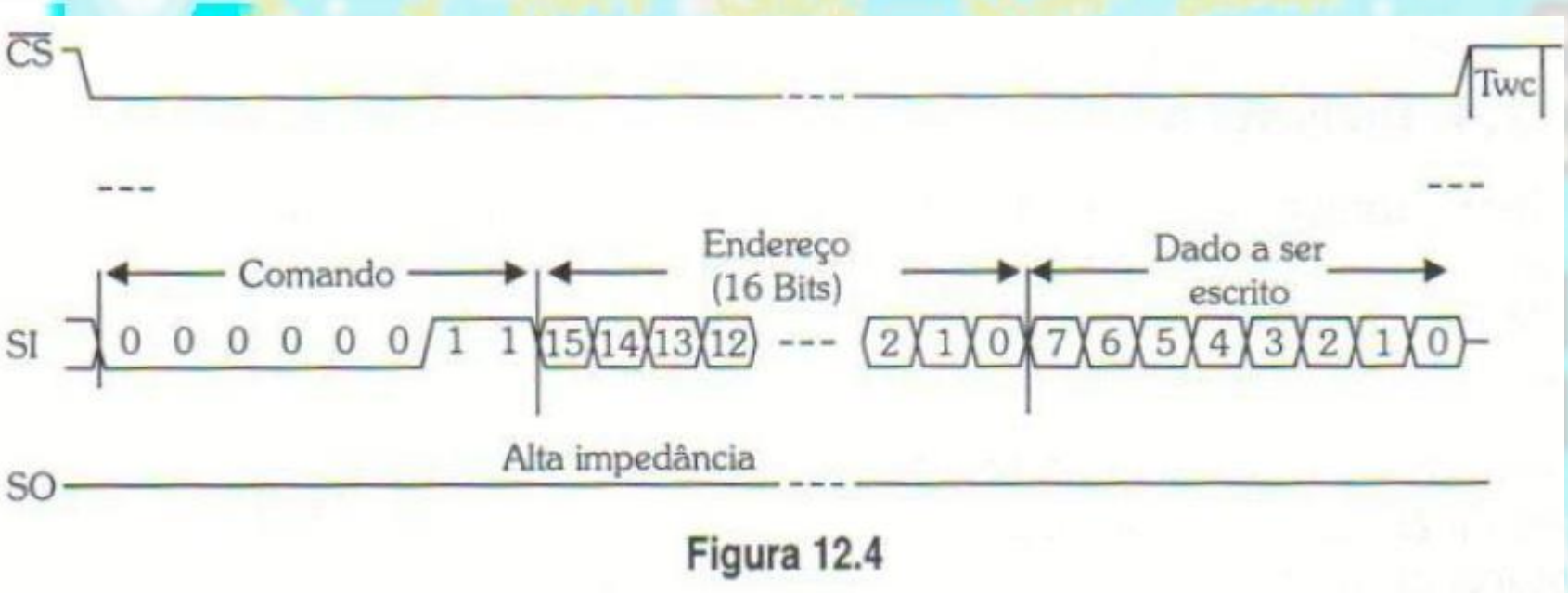
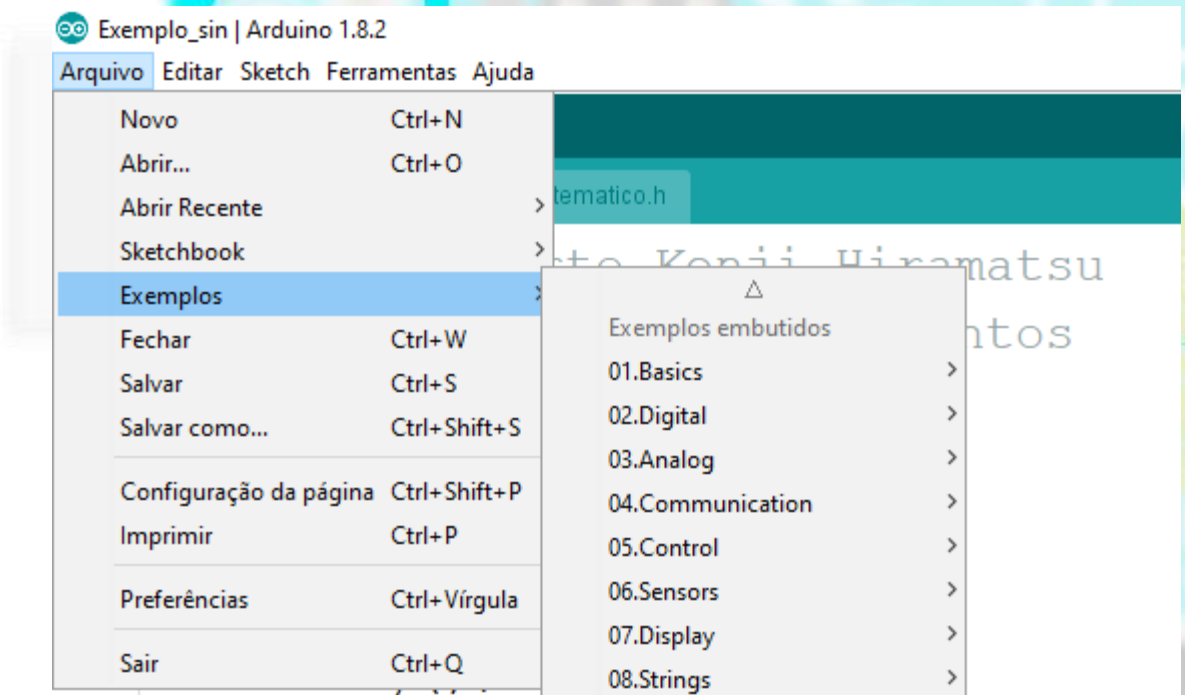


Figura 12.4

Exemplo de acesso ao SD CARD



Entendendo o exemplo - Inicialização

```
The circuit:
* SD card attached to SPI bus as follows:
** MOSI - pin 11
** MISO - pin 12
** CLK - pin 13
** CS - pin 4 (for MKRZero SD: SDCARD_SS_
*/

#include <SPI.h>
#include <SD.h>
File myFile;
void setup() {
 Serial.begin(9600); // Open serial communications and wait for port to open:
 while (!Serial);
 Serial.print("Initializing SD card...");
 if (!SD.begin(4)) {
  Serial.println("initialization failed!");
  return;
 }
```

Os pinos do sdcard devem ser conectado na sequencia definida
O pino pode ser mudado para outro pois este somente habilita acesso

Neste exemplo usa-se o pino 4.
Se precisar trabalhar com outro dispositivo SPI execute “SD.end()” que desabilita o SDCARD.
Se for usar novamente e necessário inicializar.
O processo e demorado para inicializar.

Para escrever no SD card

```
Serial.println("initialization done.");
 myFile = SD.open("test.txt", FILE_WRITE);   // open the file. note that only one file can be
                                             //open at a time.
 if (myFile) {    // if the file opened okay, write to it:
   Serial.print("Writing to test.txt...");
   myFile.println("testing 1, 2, 3.");
   myFile.close();      // close the file:
   Serial.println("done.");
 } else {
   Serial.println("error opening test.txt");
 }
```

Um arquivo de cada vez

Nome da variável que manipula arquivo

Suporta as mesmas operações da Serial

Feche o arquivo Se tiver que ler outro arquivo ou o mesmo

Ler o SDCARD

```
myFile = SD.open("test.txt");  // re-open the file for reading:
 if (myFile) {
   Serial.println("test.txt:");
   while (myFile.available()) {     // read from the file until there's nothing else in it:
     Serial.write(myFile.read());
   }
   myFile.close();    // close the file:
 } else {
   Serial.println("error opening test.txt");  // if the file didn't open, print an error:
 }
```

Mesmo procedimento de ler da serial em que a leitura começa do inicio até o fim do arquivo.

Para mais informações veja:
https://www.arduino.cc/en/Reference/SD
Inspecione também o SD.h quando a biblioteca do SD CARD estiver instalada
%HOMEPATH%\Documents\Arduino\libraries\SD\src

Características do I²C (Inter-Integrated Circuit)

- Múltiplos mestres
  - Comunicação Half-Duplex
  - Endereçamento de 7bits
- A linha é do tipo open-collector /open-drain
  - Isto é a tensão na linha serial e mantida HIGH e somente quando querem conversar e que vai para LOW

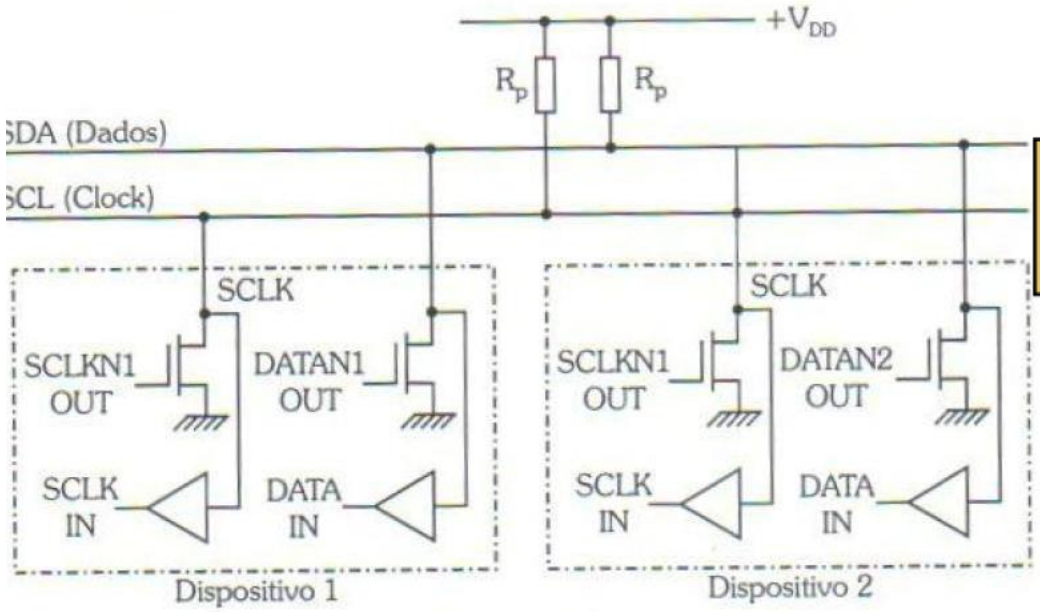
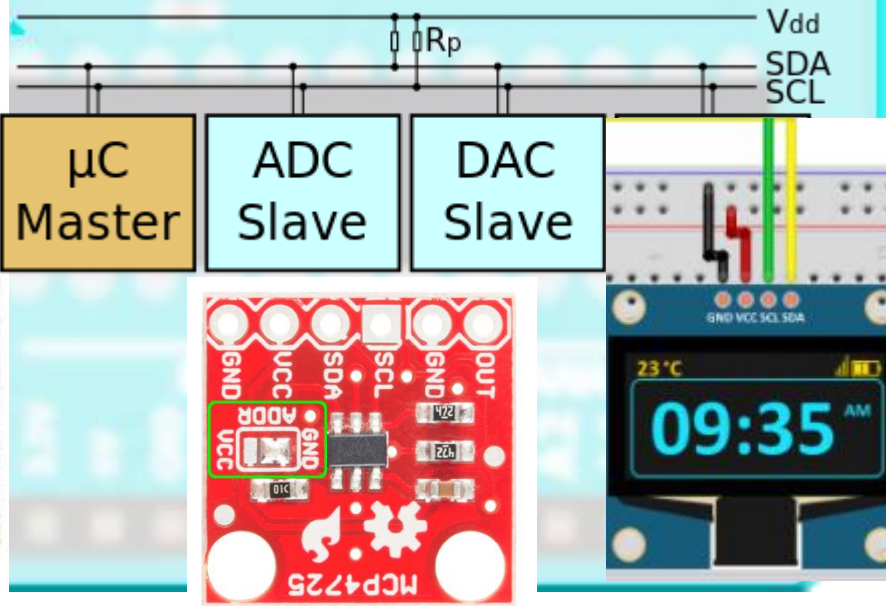
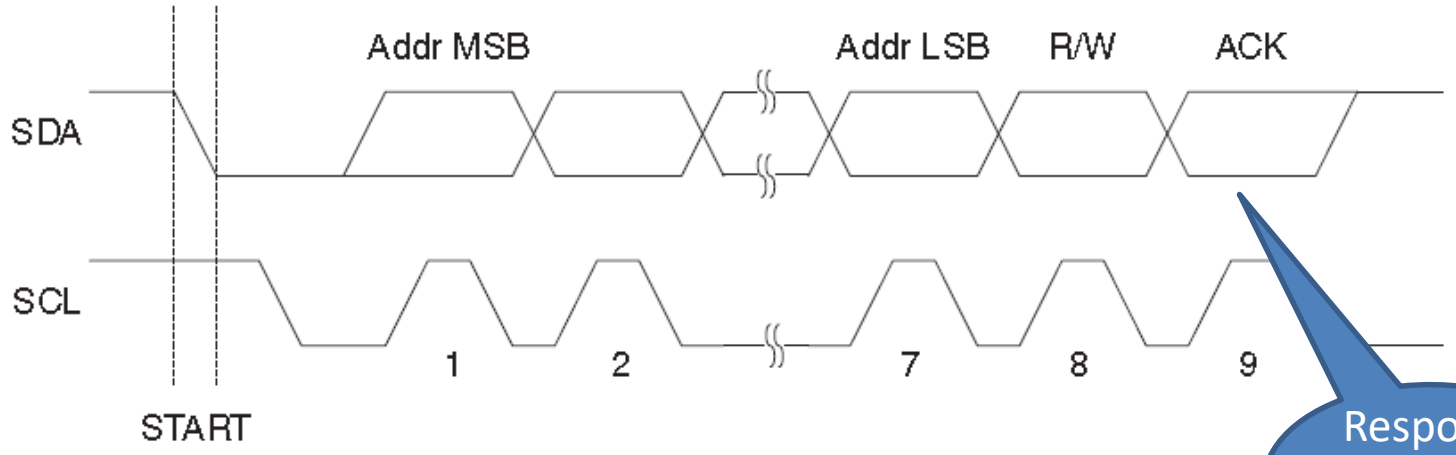


Figura 12.5



Formato do sinal do I2C

Figure 22-4. Address Packet Format



Resposta do escravo

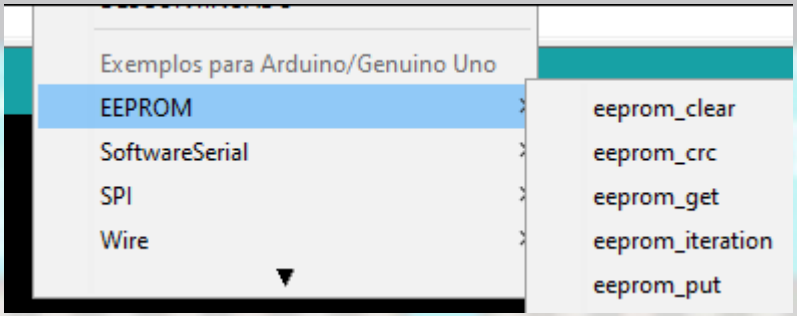
Até aqui o mestre envia o endereço com quem quer conversar e diz se e para uma operação de gravação ou leitura

EEPROM

- O UNO R3 tem 1024 bytes de EEPROM que podem ser gravadas até 100000 vezes
  - Gravar dados de configuração ou dados necessários para posterior recuperação, mas não muito frequentemente gravados
- EEPROM.write(address, value);
  - Escreve na EEPROM um byte no endereço
  - Esta operação demora 3,3ms

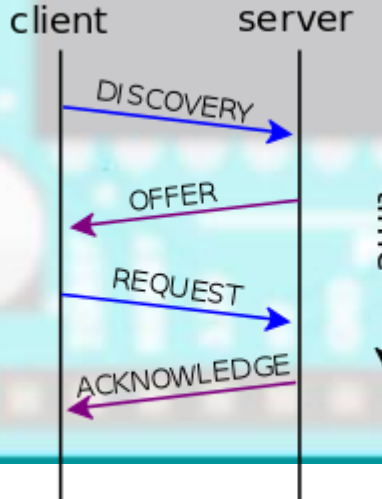
EEPROM

- value = EEPROM.read(address);
  - Leitura da EEPROM – neste caso não tem limitação da quantidade de vezes que pode ser lida.
  - Veja mais exemplos aqui:
  - https://www.arduino.cc/en/Reference/EEPROM



Protocolo de mensagens

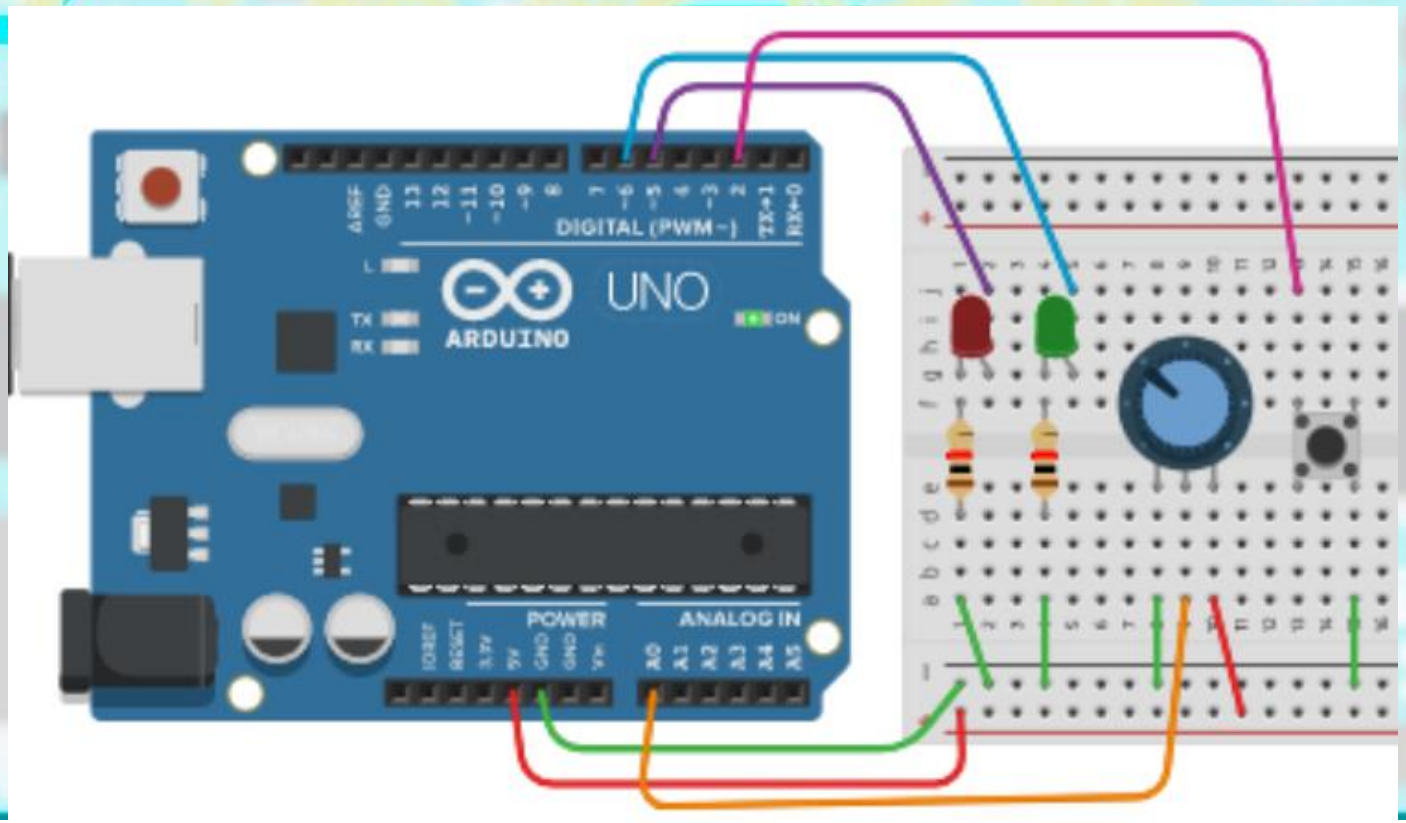
- Considerando que deseje controlar vários dispositivos acionados remotamente. O que precisamos para conversar com o microcontrolador?
- Melhor que cada mensagem tenha uma resposta para saber que ele entendeu o que você pediu.
- Escreva no papel o padrões das mensagens que você vai trabalhar

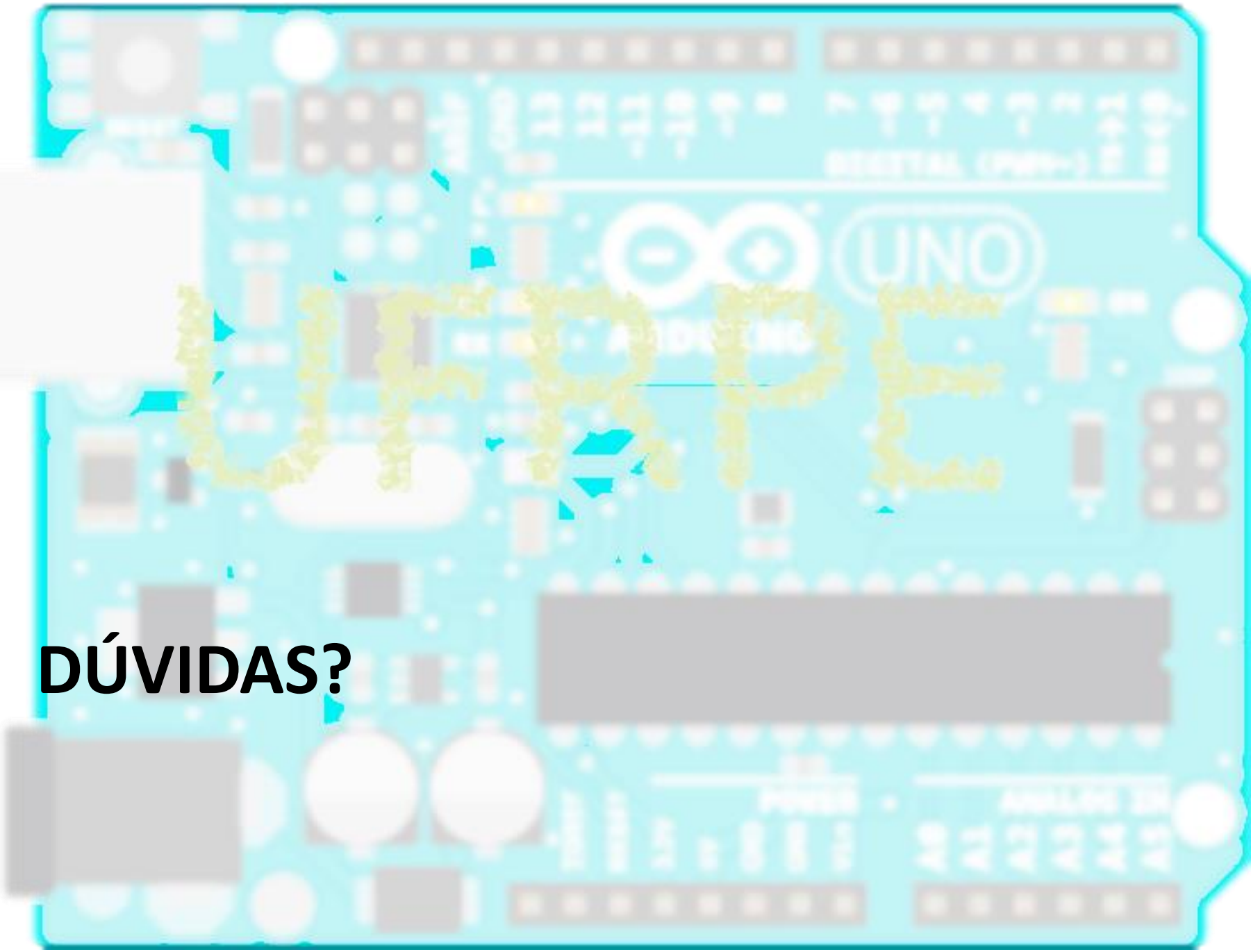


Mensagens

Protocolo de mensagens

- Acesse o projeto:
- https://www.tinkercad.com/things/d6ctTW0g7ds





DÚVIDAS?

Leitura adicional

- Leitura
- Protocolos de comunicação serial
  - https://www.robocore.net/tutoriais/comparacao-entre-protocolos-de-comunicacao-serial.html
  - https://forum.arduino.cc/index.php?topic=483133.0
  - SPI: https://www.totalphase.com/support/articles/200349236-SPI-Background
- Suporte do arduino
  - https://www.arduino.cc/reference/en/language/functions/communication/serial/
- Veja também o capitulo de serial do livro:
  - PEREIRA, Fábio. Microcontroladores PIC: Programação em C. 2. ed. São Paulo: Érica, 2003.