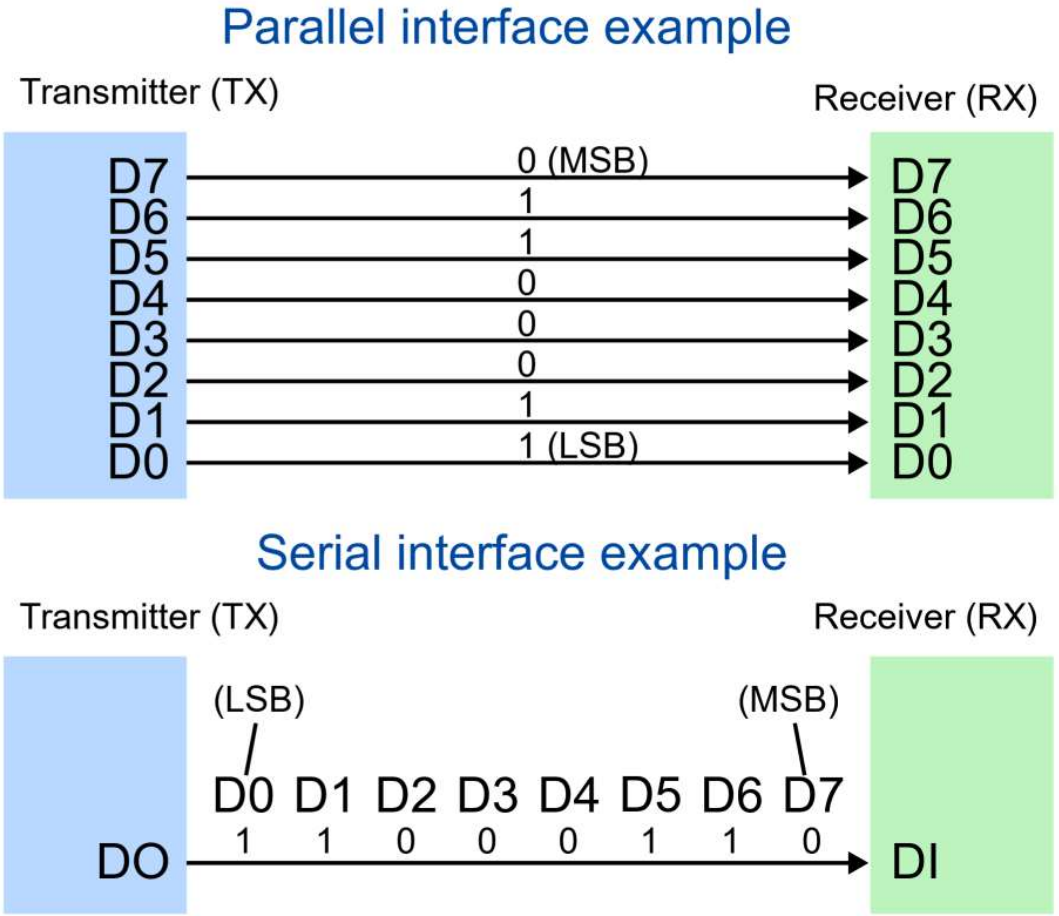
Komunikace

- paralelní vs. sériová
- RS-232 standard
- komunikace mikrokontroleru s počítačem
  - sériové rozhraní a jeho USB převodníky
  - příklady kódu pro Arduino
  - zpracování sériových dat na straně PC
- přenos dat z periferií – I2C, OneWire a SPI sběrnice
- příklad měření teploty a přenos do PC
  - USB teploměr

Sériový a paralelní přenos dat

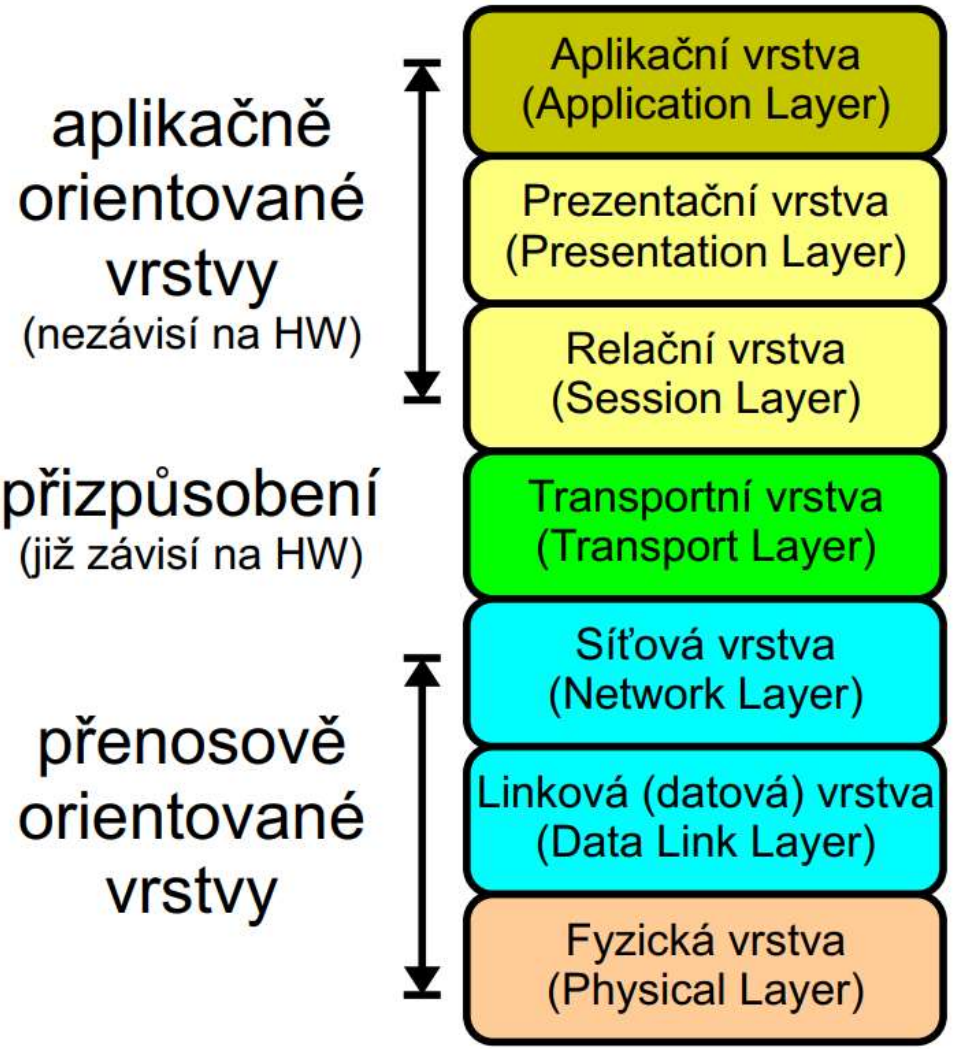


- **paralelní přenos posílá více (obvykle 8) bitů současně po jednotlivých vodičích (+ GND)**
  - **historicky takto PC komunikovalo s tiskárnami – LPT porty**
  - **ATA / ISA sběrnice pro HD**

- **při sériovém přenosu dat je to realizováno bit po bitu za sebou jeden vodič je pro vysílání, při obousměrném přenosu pak druhý pro příjem, opět plus GND**
  - **u PC se jedná o sériový (COM) port standardu RS232c**
  - **varianty jsou simplex - pouze jedním směrem, ten nelze obrátit,**
  - **poloduplex (half-duplex) - najednou jde přenášet pouze jedním směrem, ten ale jde obrátit**
  - **duplex (full-duplex) - data lze přenášet oběma směry současně**

Vrstvy komunikace

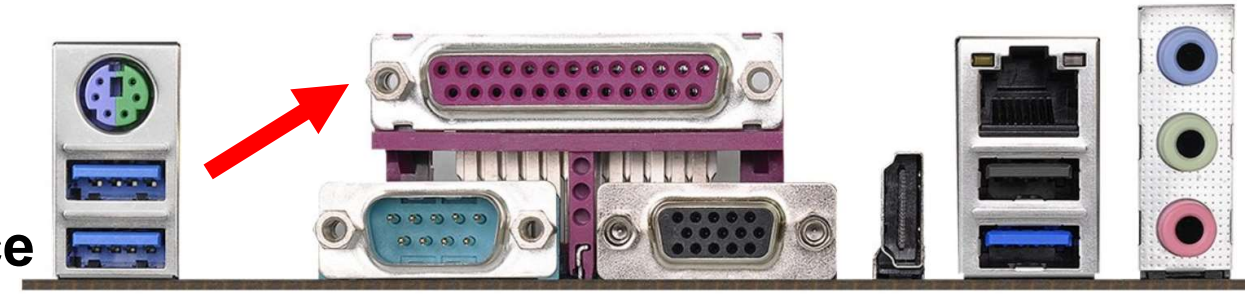
- **aplikační (FTP, DNS, DHCP, POP3, SMTP, SSH, Telnet, TFTP)**
  - **aplikace ze serveru, přenos souborů**
- **prezentační (Samba)**
  - **zajistit, aby data znamenala totéž na obou stranách přenosu**
  - **konverze čísel mezi formáty, převod znaků ASCII / UTF**
  - **náhrada unixového LF na CRLF pro Win**
- **relační (NetBIOS, RPC)**
  - **vedení dialogu mezi účastníky: navazování spojení, udržování, rušení, sdílení jednoho transportního spojení více relačními spojeními, nebo naopak více současných transportních spojení pro jedno relační**
  - **poslední vrstva nezávislá na konkrétním typu síte a jejím transportním protokolu**
- **transportní (TCP, UDP) – zajištění spolehlivosti, řídí provoz, data do bloků**
- **síťová (IP, Ethernet) – doručení dat, směr přenosu přes uzly do cíle**
- **linková / datová (PPP, Frame Relay) – mezi sousedními uzly, bezchybnost**
- **fyzická vrstva (RS232, ...) – skutečný přenos (dráty...)**



Jaký spoj?

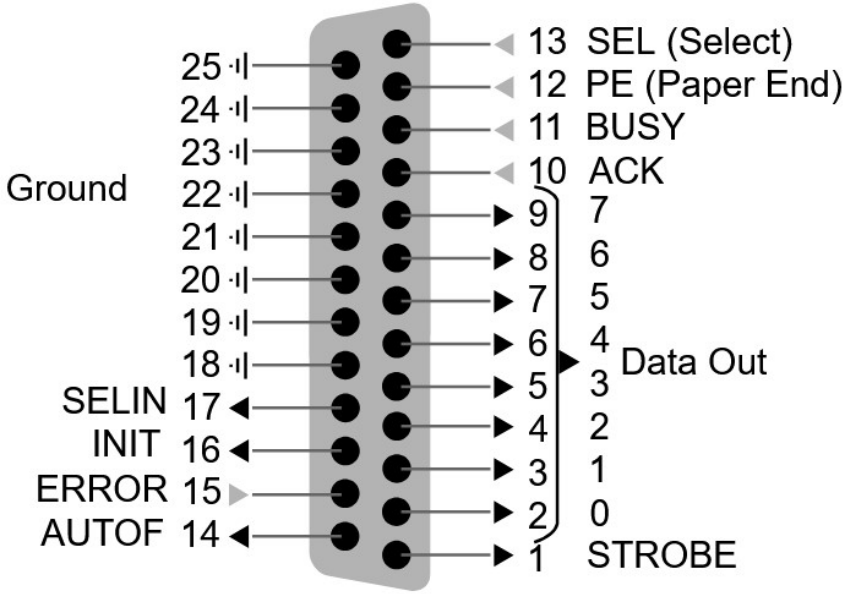
- drát - binární hodnota, malé vzdálenosti
- dvoulinka - vhodná pro analogový signál
  - jen pro velmi krátké digitální spojení
- kroucená dvoulinka (TP, twisted pair) - levné a dostupné, s vyšší odolností proti šumům
  - dva navzájem zkroucené izolované vodice, výsledné rušivé napětí je až 200x menší než u nezkroucené dvoulinky
- stíněná kroucená dvoulinka (STP, shielded ...)
  - vysoká šumová imunita, až na 1 km, přenosová rychlost Mbit/s
- koaxiální kabel – malý útlum na vysokých frekvencích
  - několik frekvenčních pásem současně - až desítky Mbit/s, na několik km
- světlovodný kabel - přenos modulovaného světelného paprsku
  - až několik Gbit/s, ale obtížné spojování, drahé, speciálních přijímač a vysílč
  - necitlivost k rušení (snad když to někdo přesekne ...), nemožnost odposlechu
  - úplné elektrické oddělení obou stran, nízké ztráty

Paralelní port



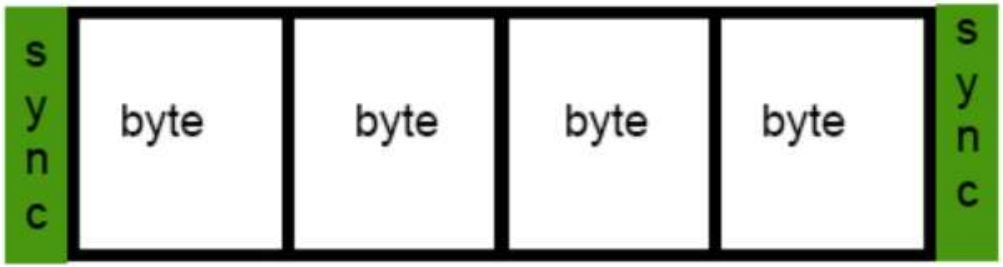
- dříve byl součástí počítače – 25-pinový konektor pro připojování tiskáren pomocí „Centronix“ kabelu
  - jednosměrná data, k tomu pár dalších kontrolních vodičů
  - varianta Bitronix byla obousměrná

- pro jiné účely rychlé komunikace se používal zřídka
  - morálně zastaral, dnes se dá nahradit přídavnou kartou nebo USB-LPT adaptérem (LPT ... line printer terminal)
  - fakticky jeho úlohu převzalo USB (taky svým
- úrovně signálu byly 5V (TTL)
- býval využíván pro přímé ovládání různých periferií vlastní konstrukce na nejnižší úrovni
  - nebezpečí pro PC v případě zkratu

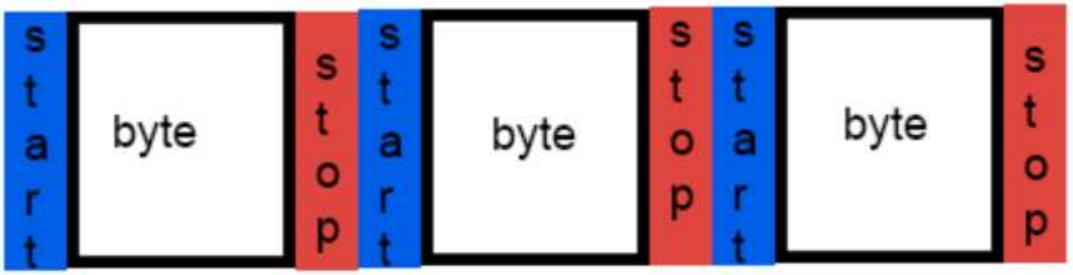


Synchronní vs. asynchronní přenos

- **synchronně** je vysílán nepřetržitý řetězec bitů, na začátku přenosu je vyslán synchronizační signál (SYN) pro spárování s druhým zařízením
  - určí se intervaly, kdy se vyhodnotí jednotlivé bity - jediný generátor synchronizačních impulsů na straně zdroje dat, po celou dobu přenosu
  - rychlejší přenos. ale nárokv na kvalitu linkv a větší riziko chyb

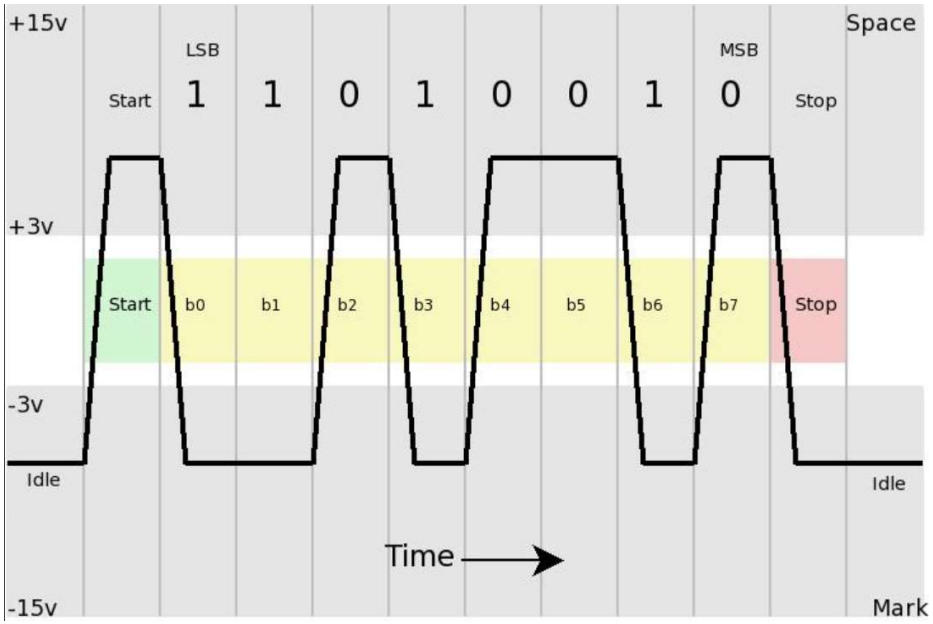


- **asynchronně** - sled vysílaných / přijímaných pulsů není v čase vázán
  - vyhodnocení log. 0 nebo 1 určuje délka odpovídající napěťové úrovně
  - při přenosu nekolika bitů stejné úrovně obtížné jejich rozpoznání
- **arytmicky** – start / stop systém, kompromis mezi předchozími
  - nepředpokládá se trvalý přenos
  - zdroj dat vyšle nejprve jeden bit (rozběhový, start bit), teprve potom následuje posloupnost vlastních informačních bitů a na závěr se vyšle 1- 2 ukončovací bity (stop bity)



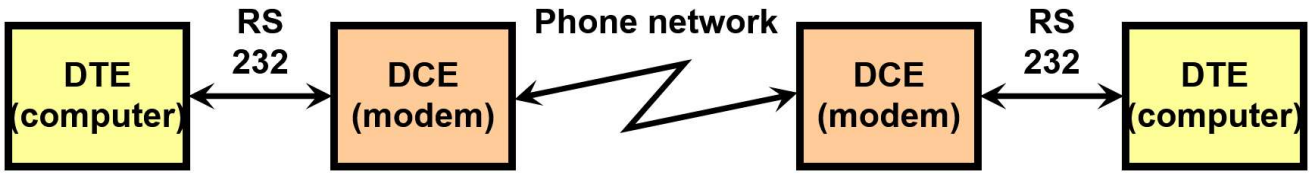
Sériový port RS-232

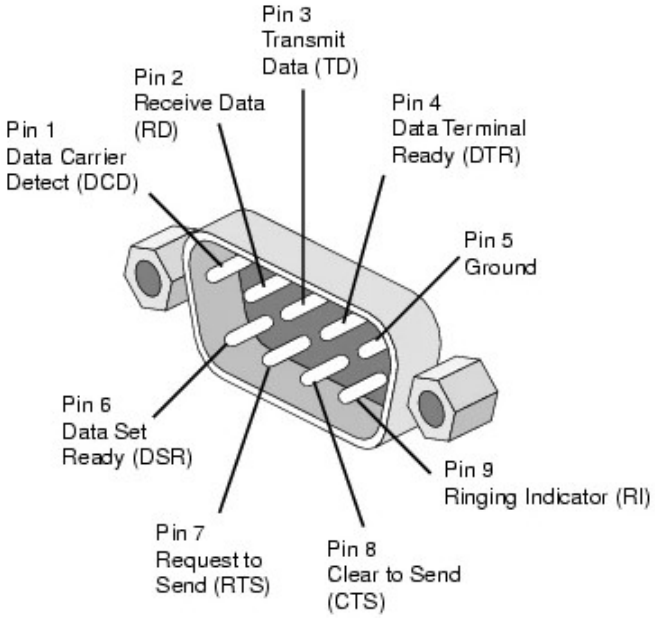
- u PC využíván pro komunikaci s periferními zařízeními, dnes ustoupil USB (mnohem složitější komunikace), ale zachovává se v průmyslové a vědecké oblasti, včetně odvozených variant RS-422 a RS-485
  - je vyveden na 9pinový D-Sub konektor DE-9 M, dříve byla častá i varianta DB-25 M (M ... male / samec, piny)
  - pokud již schází, tak se nabízí na přídavných kartách, nebo velmi jednoduše jako USB-RS232 převodník
  - poslední definice standardu je
- standard (z roku 1969!) definuje asynchronní sériovou komunikaci
  - pořadí přenosu datových bitů je od nejméně významného bitu (LSB) po nejvýznamnější (MSB)
  - (volitelný) počet datových bitů je obvykle 8 bitů, lze nastavit i 7 nebo 9 bitů, k tomu přistupují start a stop bity
  - nastavuje se určitá komunikační rychlost



Konektor

- historicky se počítač (DTE zařízení) propojí s periferií (modem, DCE zařízení)
  - použije se přímý kabel, pro přenos po telefonní lince

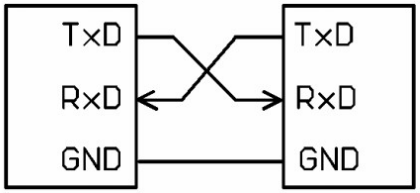


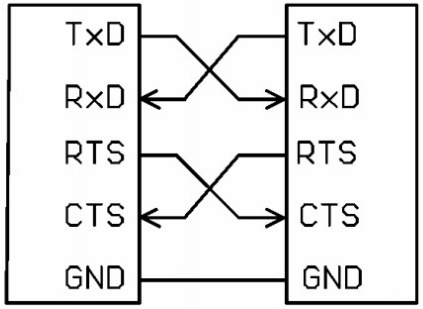


Zkratka	Popis	Pin	Směr u DTE (PC)	Směr u DCE (perif.)
TxD, TX	**Data posílaná z DTE do DCE**	**3**	Výstup	Vstup
RxD, RX	**Data přijímaná v DTE z DCE**	**2**	Vstup	Výstup
RTS	„Požadavek na vysílání“; Log 1 zde signalizuje, že DTE chce vysílat. Některé převodníky RS232/RS485 používají pro přepínání směru linky.	**7**	Výstup	Vstup
CTS	„Povolení k vysílání“; Log 1 na tomto vstupu protistrana signalizuje, že DTE může vysílat data	**8**	Vstup	Výstup
DSR	Log 1 na tomto vstupu protistrana signalizuje, že je připravena (což neznamená že DTE může okamžitě zaslat data, viz CTS)	**6**	Vstup	Výstup
GND	**Signálová zem. Hodnoty napětí na pinech jsou určeny proti této zemi.**	**5**	—	—
DCD, CD, RLSD	Log 1 na tomto vstupu protistrana signalizuje, že detekovala na vedení nosný signál a může komunikovat (DCE je např. modem na telefonní lince)	**1**	Vstup	Výstup
DTR	Log 1 na tomto výstupu DTE signalizuje protistraně svoji připravenost. Protistrana (např. modem) se tím aktivuje nebo zase deaktivuje. Modem obvykle odpovídá nastavením DSR na log. 1.	**4**	Výstup	Vstup
RI	Log 1 signalizuje do DTE příchozí hovor, tedy že někdo požaduje datové spojení („ring“ je anglicky „zvonit“; zvl. u telefonního modemu).	**9**	Vstup	Výstup

Propojení RS-232

- pro datové signály (RXD a TXD) je log. úrovní signálu mezi +3 V až +15 V, log. 1 je mezi -3 V až -15 V (bipolární)
- pro řídící signály (RTS, CTS, DTR, DSR, …) je to naopak, log. 0 je -3 V až -15 V, log. 1 je +3 V až +15 V
- takto to funguje u standardních RS-232 portů
- k propojení slouží mimo přímý kabel (DTE-DCE, M-F) nejčastěj tzv. „null modem cable“ – překříží se signály RxD a TxD (DTE-DTE, F-F)
  - propojí se RxD-TxD, TxD-RxD a GND-GND v nejjednodušší variantě
  - délka kabelu může být několik metrů, kratší pro vyšší přenosové rychlosti
  - je možné propojit i kontrolní signály

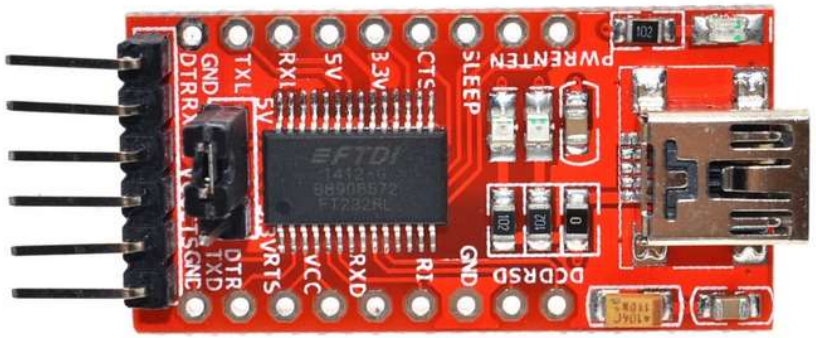


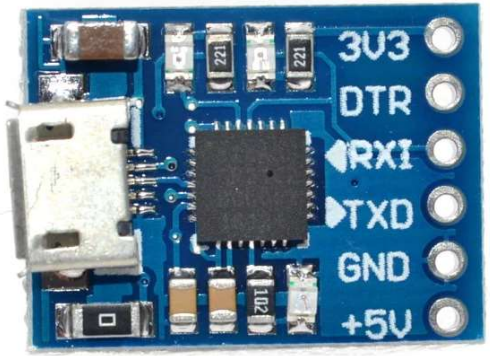


- na běžných sériových portech v PC lze dosáhnout rychlost max. 115200 Bd (baud, bitů za s)
  - ostatní baudové rychlosti jsou odvozeny dělením - 57600, 38400, 28800, 23040, 19200, 9600, 4800, 2400, … Bd
  - přenosová rychlost je nižší než baudová rychlost - ke každým 8 datovým bitům se navíc přenáší ještě 1 startbit, 1 – 2 stopbity a případně také 1 paritní bit

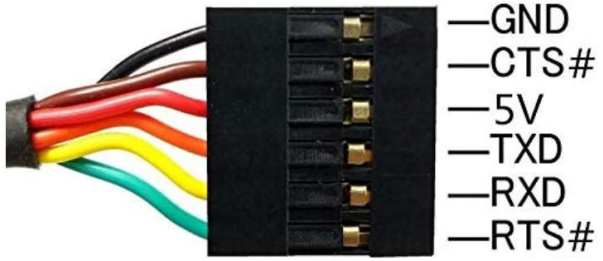
Sériové přenosy v TTL úrovních

- pro komunikaci s mikrokontrolery je zbytečné používat standardní RS-232 bipolární úrovně signálů, stačí obyčejné TTL úrovně – log. 0 je nulový signál (blízko 0V či GND), log. 1 je pak blízko 5V resp. 3.3 V
  - pozor – nezaměnit, mikrokontroler by se mohl zničit
  - tyto úrovně nabízí USB-Serial převodníky (USB-Serial TTL, USB-UART), které jsou často přímo na desce Arduino a kompatibilních, které pak poskytují přímo USB připojení
  - pokud nejsou, tak se použije externí USB-Serial modul či kabel, dle čipů existují FT232 (FTDI), CP2102 (Silicon Labs), CH340



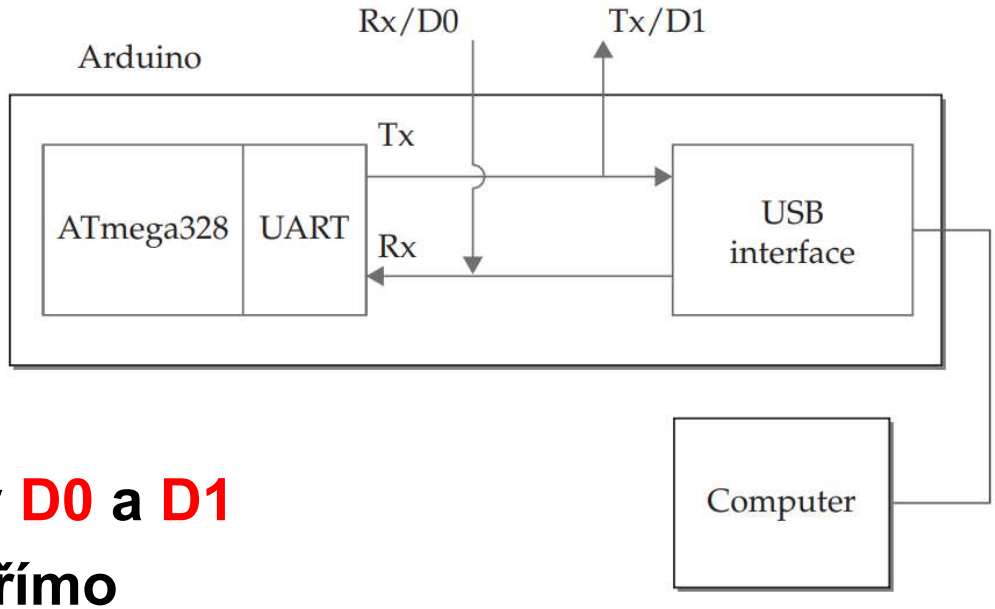


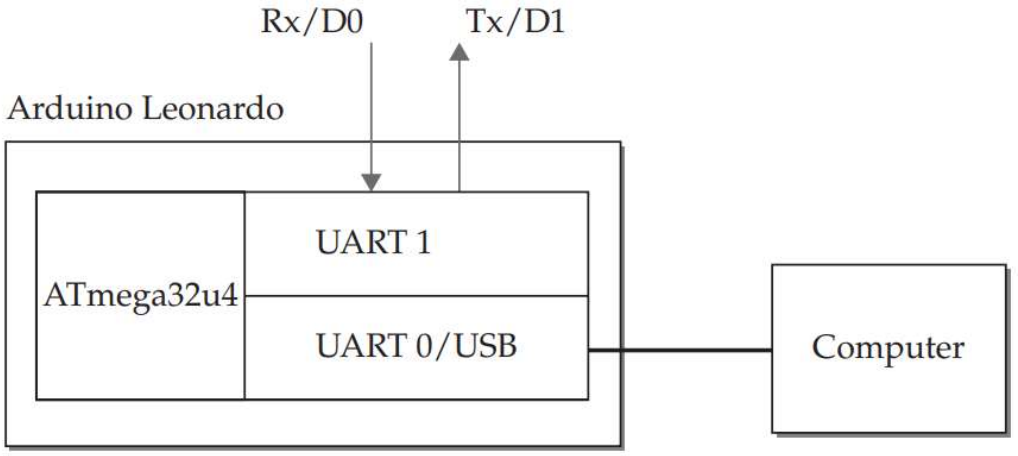
FTDI:



Sériový hw u Arduina

- hw sériový submodul je UART
  - (universal asynchronous receiver transmitter)
- pokud je na modulu i čip pro USB-sériový převodník, tak jsou Rx a Tx signály vyvedeny i na piny D0 a D1
- někdy je USB přenos generován přímo v mikrokontroleru
  - (např. Arduino Leonardo)
- objekt Serial obvykle odpovídá portu vyvedenému na D0, D1
  - iniciace probíhá v rámci setup() části – Serial.begin
  - obsluha pak v části loop()
- pokud je potřeba další sériový port, ale už není UART:






```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // piny pro RX, TX
void setup() {
  mySerial.begin(9600); mySerial.println("Hello, world?");
}
```

Seriové porty u Arduina a spol.

Deska	Počet portů	Detaily
Uno, Nano	1	Rx D0 a Tx D1, současně přes USB (standardní situace)
Leonardo	2	Serial jde přes USB. Serial1 přes Rx D0 a Tx D1
Mega2560	4	Serial jde na USB a D0 a D1. Serial1 na pinech 19 Rx a 18 Tx, Serial2 17 Rx a 16 Tx, Serial3 na 15 Rx a 14 Tx
Seeeduino XIAO	2	Serial je SerialUSB, Serial1 vyveden na Rx a Tx piny
MKR	2	SerialUSB, Serial1 na pinech 13, 14
ESP8266 (NodeMCU, Wemos D1 Mini)	1.5	Serial TX a Rx piny přes USB, Serial1 jen vstupní - Rx je na pinu GPIO2
ESP32 (Lolin32)	3	Serial jde přes USB interface, další dva hw porty lze namapovat na lib. piny - viz knihovna HardwareSerial

- **mimo Serial, který funguje vždy, je u ostatních sériových portů vhodné prostudovat dokumentaci ...**
- **v případě potřeby lze sw sériové porty (budou pomalejší) vytvořit na lib. pinech pomocí knihovny SoftwareSerial**

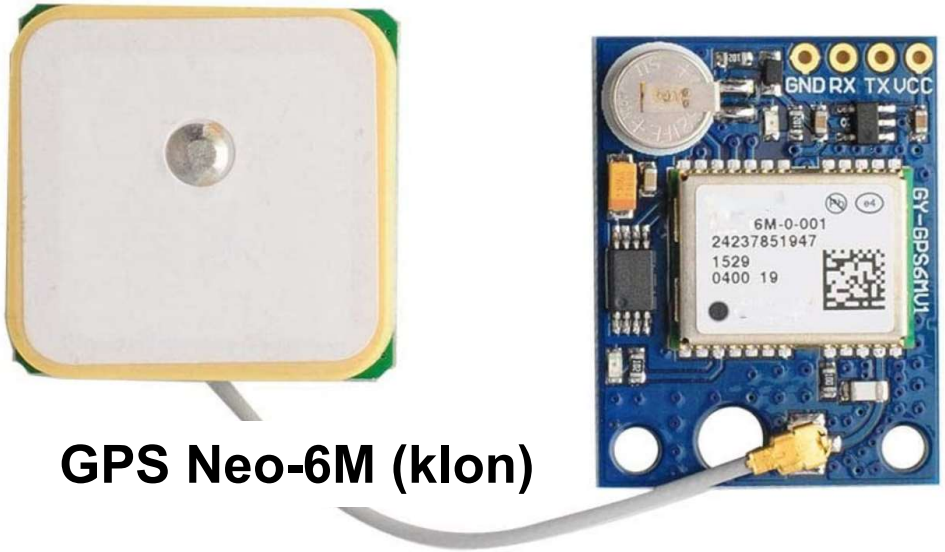
příklad

- zvenčí se přijme příkaz pro započetí či ukončení sériového přenosu analogové hodnoty:

```
const int readingPin = A0;
boolean sendReadings = false;

void setup() { Serial.begin(9600); }

void loop() {
  if (Serial.available()) {
    char ch = Serial.read();
    if (ch == 'g') { sendReadings = true; }    // go!
    else if (ch == 's') { sendReadings = false; }  // stop!
  }
  if (sendReadings) {
    int reading = analogRead(readingPin);
    Serial.println(reading);
    delay(1000);
  }
}
```

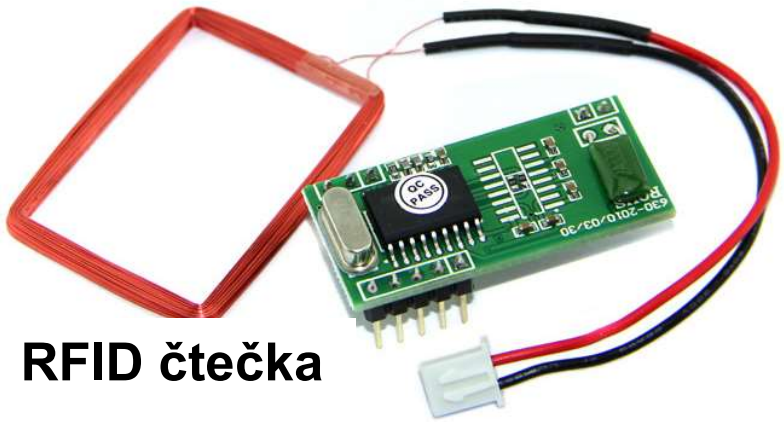
Seriové periferie

- obvykle jsou to větší moduly, často i s jistou vlastní inteligencí, které historicky komunikovaly sériově s PC, pouze se přešlo na TTL úrovně signálu
- příklady – GPS, telefonní GPRS, multimetry, RFID čtečky, grafické displeje s vnitřním vykreslovacím jazykem



GPS Neo-6M (klon)

GSM SIM800

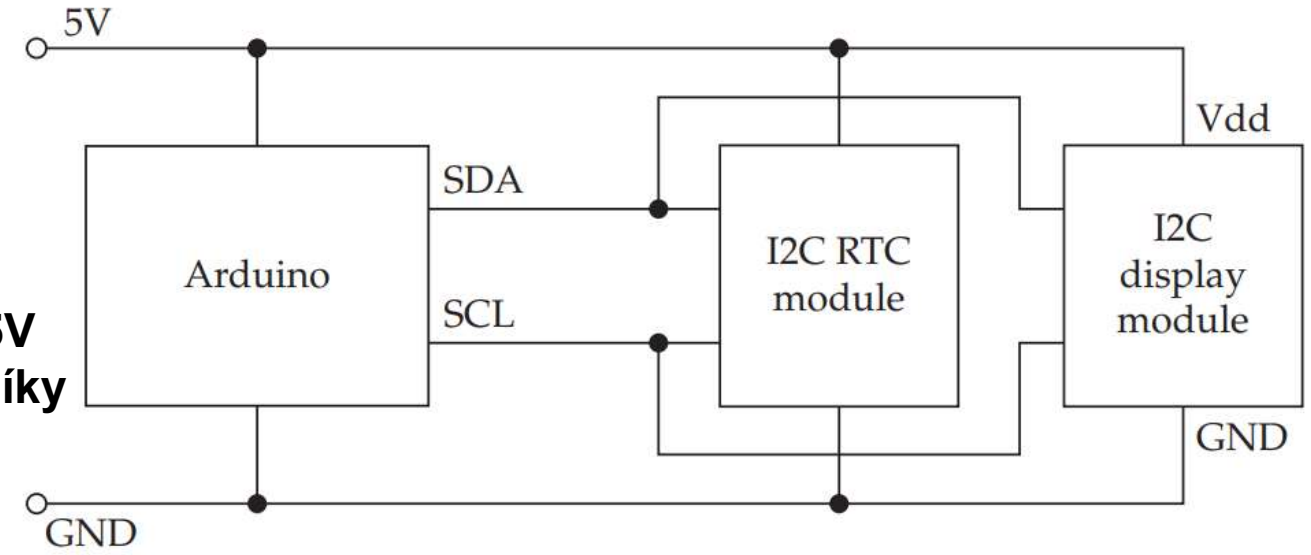


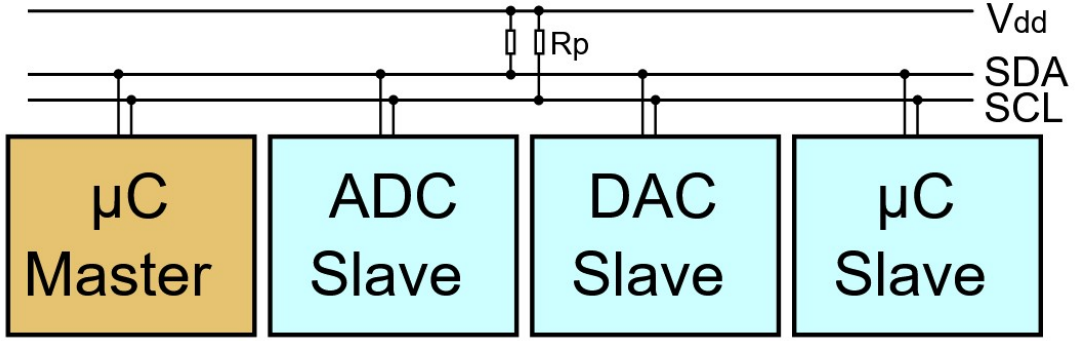
RFID čtečka

displej (inteligentní)

I2C ("I squared C", two-wire interface)

- další sériové rozhraní pro dvoudrátovou sběrnici připojující 1 a více (až 128) periferních modulů identifikovaných pevnou adresou (sensory, displeje, hodiny, ADC, DAC, ...)
- signály jsou data SDA a hodiny SCL, plus GND a také napájení V_{dd}
- na sběrnici je vždy jedno zařízení řídící přenosy (master, typicky mikrokontroler jako Arduino) a několik podřízených modulů (slaves)
  - SDA a SCL linky musí být připojeny přes odpor cca 4k7 na napájecí napětí (pull up, zdvihací)
  - mezi moduly na 3.3 a 5V se musí vložit převodníky úrovní signálu
- c

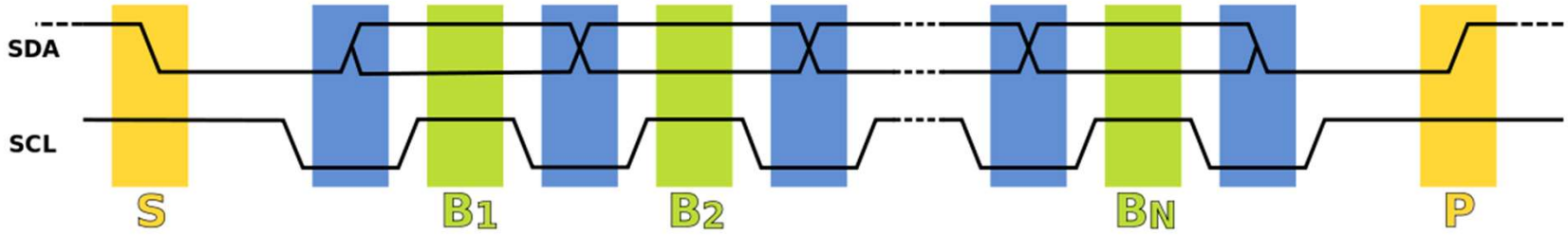




Způsob komunikace

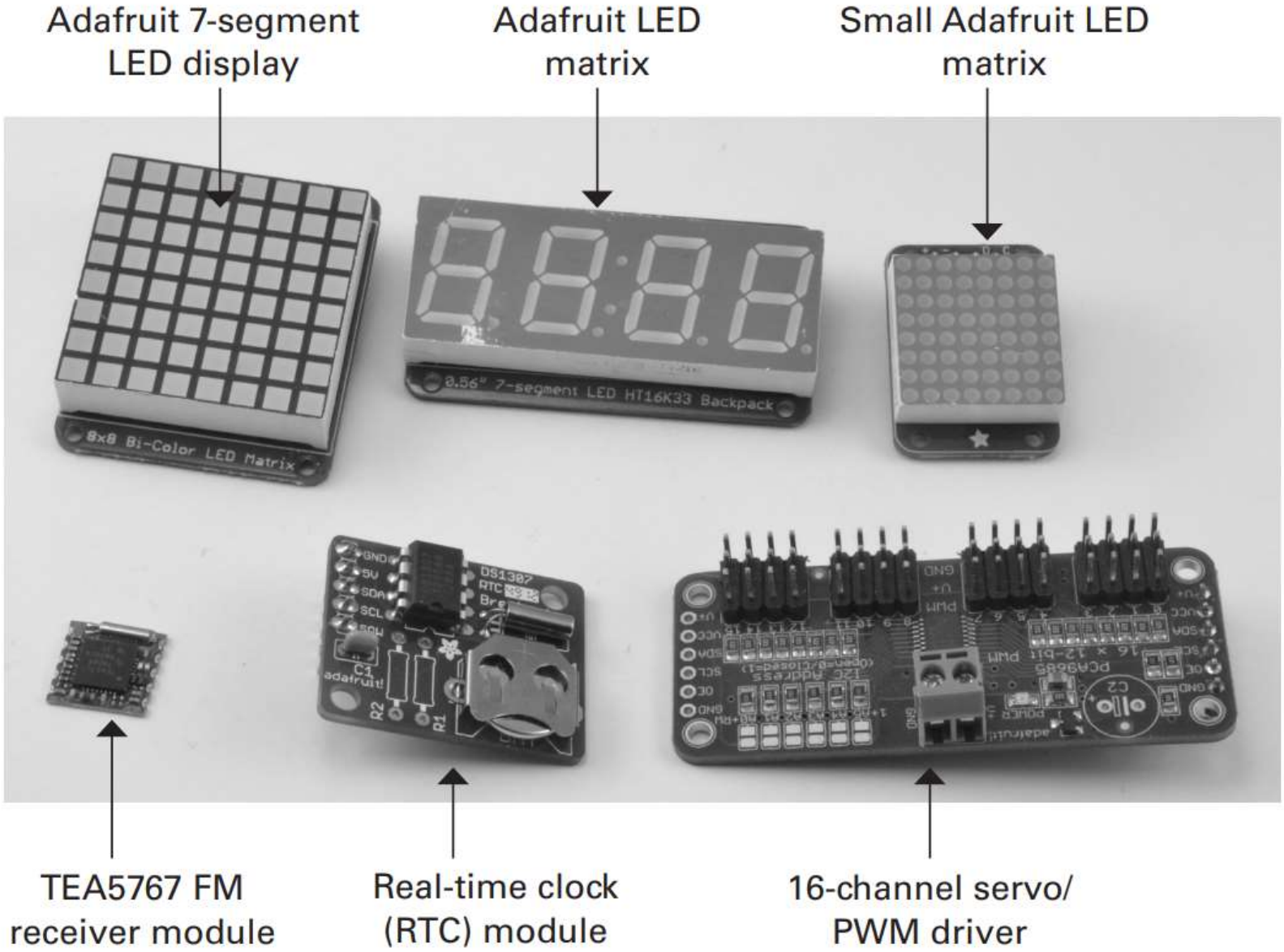
Board	Pins
Uno	A4 (SDA) and A5 (SCL)
Leonardo	D2 (SDA) and D3 (SCL)
Mega2560, Due	D20 (SDA) and D21 (SCL)
NodeMCU/Wemos D1 Mini (ESP8266)	D1 (SCL) and D2 (SDA)
Lolin32 (ESP32)	21 (SDA) and 22 (SCL)

- **SDA, SCL jsou u modulů na různých pinech, některé varianty mohou mít i dvě I2C rozhraní (I2C0, I2C1)**
  - **při přenosu jsou na SDA vysílány jednotlivé datové bity, logická úroveň na SDA se smí měnit pouze je-li SCL v úrovni L**
  - **toto pravidlo je porušeno při vysílání podmínek START a STOP, které se používají k zahájení komunikace a k ukončení přenosu**
  - **v jednom okamžiku vysílá jen jedno zařízení**



  - **přenos dat se zahajuje START bitem (S), když je SDA nízká, zatímco SCL zůstává vysoká**
  - **pak SDA nastaví přenášený bit zatímco SCL je nízká (modrá) a jsou odebrány vzorky dat (přijaté) při SCL stoupá (zelená)**
  - **když je přenos dokončen, je poslaný STOP bit (P) pro uvolnění datové linky, změnou SDA na vysokou, zatímco SCL je trvale vysoká**
  - **aby se zabránilo falešně detekci, je úroveň na SDA změněna na negativní hraně a je zachycena na kladné hraně SCL**

Ukázky periferií



- **výhodou I2C je malý počet vodičů nutných ke komunikaci**

Použití Wire knihovny

- uživatel nemusí řešit komplexnost protokolu, jednoduše vybere zařízení, posílá data a určuje, kolik dat chce přijmout

- zavede se a inicializuje se v rámci setup()

```
#include <Wire.h>
void setup() { Wire.begin(); } // master nemusí zadat adresu
```

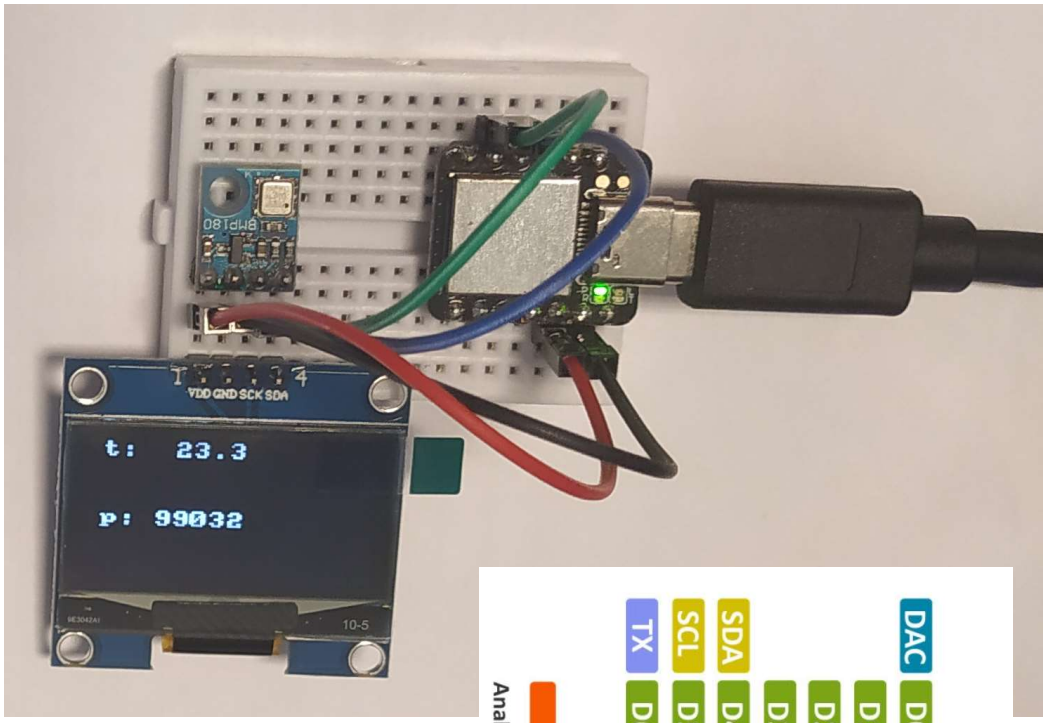
- poslání dat:

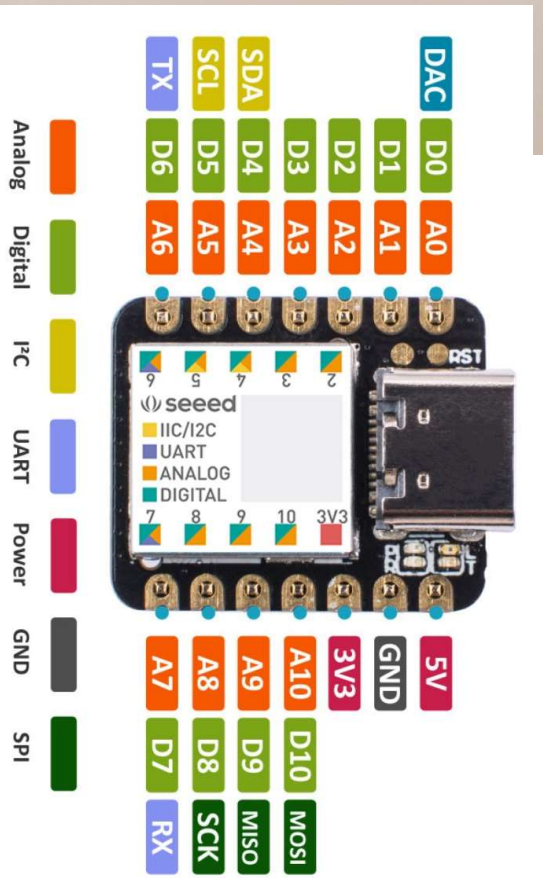
```
Wire.beginTransmission(4);  // zadává se adresa periferie (slave)
Wire.send(123) // send the byte 123
Wire.send("ABC"); // send the string of chars "ABC"
Wire.endTransmission();  // ukončení bloku přenosu
```

- příjem dat:

```
void loop() {
  Wire.requestFrom(4, 6); // request 6 bytes from slave address 4
  while(Wire.available()) {  // slave may send less than requested
    char c = Wire.receive(); // receive a byte as character
    Serial.print(c); // print the character
  }  delay(500);
}
```

BMP180 a OLED





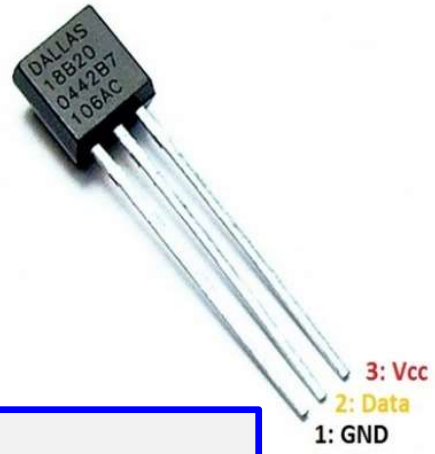
- Xiao s připojeným sensorem BMP180 – měří teplotu (oC) a atmosférický tlak (Pa)
- data se zobrazují na 1,3“ OLED display
- obě periferie připojeny na I2C sběrnici (mají náhodou piny se shodnými signály)
- současně se data posílají sériově na USB

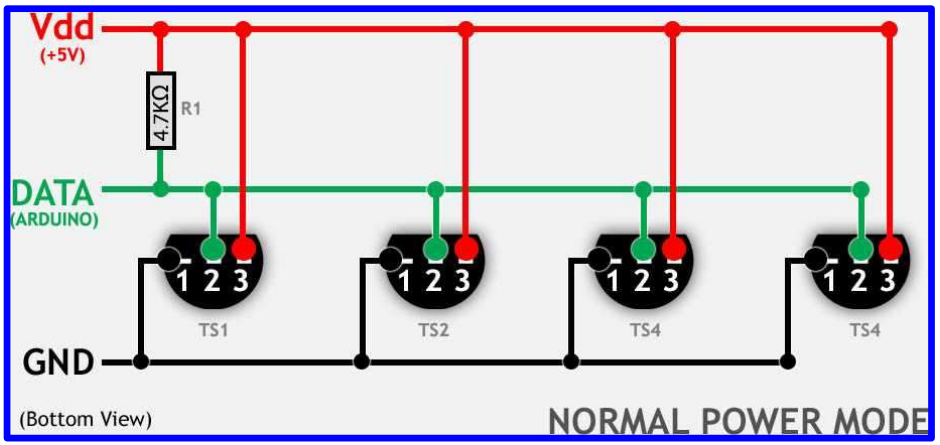
- následuje ukázka ovládacího programu

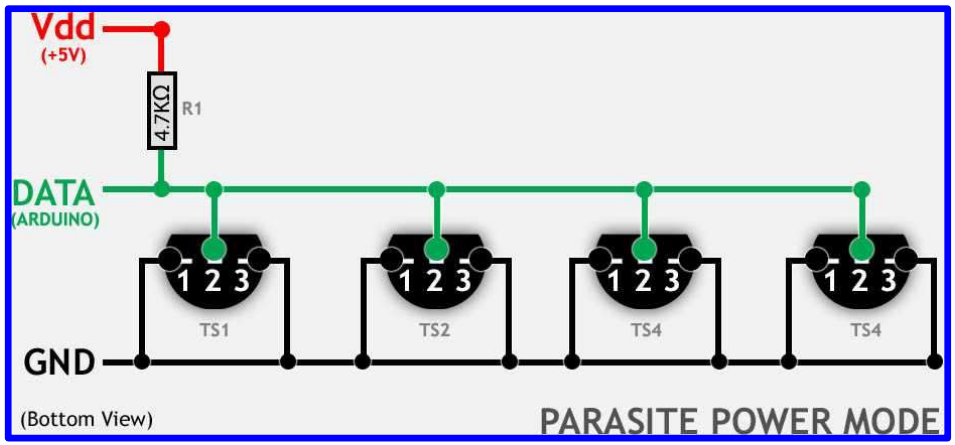
```
#include <U8x8lib.h>
#include <Adafruit_BMP085.h>

U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE);
Adafruit_BMP085 bmp;
char buf[10],flt[10];
float t;

void setup(void) {
  u8x8.begin(); u8x8.setPowerSave(0);
  Serial.begin(9600);  if (!bmp.begin()) {
    Serial.println("Valid BMP180 sensor not found!"); while (1) { }
  }
  u8x8.setFont(u8x8_font_amstrad_cpc_extended_r);
}

void loop(void) {
  u8x8.clear(); t = bmp.readTemperature(); dtostrf(t,5,1,flt);  // float > string
  sprintf(buf, "t: %s",flt); u8x8.drawString(1,0,buf);
  Serial.print(buf); Serial.print('\t');
  sprintf(buf, "p: %d", bmp.readPressure()); u8x8.drawString(1,4,buf);
  Serial.println(buf);  delay(2000);
}
```

1-Wire sběrnice navržená Dallas Semiconductor pro komunikaci zařízení nízkou rychlostí včetně napájení

- podobná ke I2C, s nižší datovou propustností ale s delším dosahem
- lze jednoduše vytvořit síť z několika 1-Wire zařízení a s jedním master zařízením - MicroLAN
  - je možné použití pouze dvou drátů: data a země; 1-wire zařízení obsahují 800 pF kondenzátor k uchování náboje pro napájení během aktivního datové přenosu
  - síť je implementována jako master s otevřeným kolektorem, který je připojený k jednomu nebo více slavům s unikátní adresou, také s otevřeným kolektorem
  - jeden pull-up rezistor je společný pro všechna zařízení, na sběrnici dovoluje napětí až na 3, nebo 5 voltů a může poskytnout energii pro slave zařízení
  - komunikace se zahájí, když někdo stáhne sběrnici k nule
- nejčastější je dig. teploměr DS18B20 (12 bit rozlišení)
  - jiná zařízení jsou známá jako iButton či Dallas Key







OneWire knihovna

příklad prohledání zařízení připojených na sběrnici:


```
#include <OneWire.h>

OneWire bus(10);  // pin where data output is connected

void setup()  {
  Serial.begin(9600);
  byte address[8]; // 64 bits
  while (bus.search(address))  {
    for(int i = 0; i < 7; i++)  {  Serial.print(address[i], HEX); Serial.print(" ");  }
    // checksum OK or Fail
   if (OneWire::crc8(address, 7) == address[7])  { Serial.println(" CRC OK");  }
   else  {  Serial.println(" CRC FAIL"); }
  }
}
void loop() {  }  // nothing to do ...
```

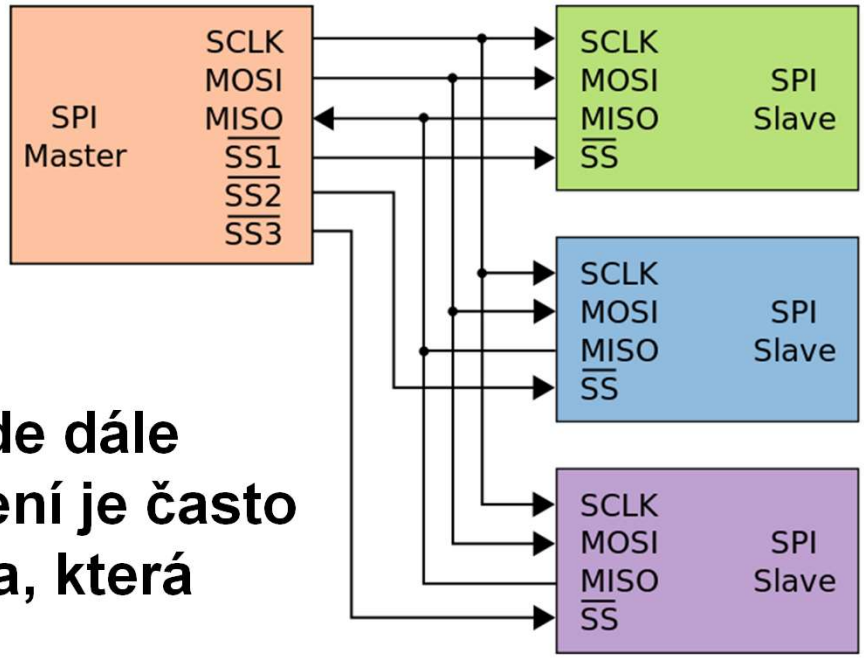
Čtení teploty z 18B20

```
#include <OneWire.h>
#include <DallasTemperature.h>

const int busPin = 10;

OneWire bus(busPin);

DallasTemperature sensors(&bus);

DeviceAddress sensor;

void setup() {
  Serial.begin(9600);
  sensors.begin();
  if (!sensors.getAddress(sensor, 0)) {  Serial.println("NO DS18B20 FOUND!");  }
}
void loop() {
  sensors.requestTemperatures();
  float tempC = sensors.getTempC(sensor);
  Serial.println(tempC);
  delay(1000);
}
```

podobně lze číst i další zařízení na stejné sběrnici, v getAddress se druhé číslo zvětšuje o 1

SPI serial peripheral interface

- sériové periferní rozhraní, komunikace je realizována pomocí společné sběrnice tvořené hodinovým signálem a dvěma datovými
  - SCLK, MOSI master out slave in, MISO master in slave out
  - bývají na Arduino a klonech vyvedeny na určitých datových pinech
- adresace se provádí pomocí zvláštních vodičů, které při log. 0 aktivují příjem a vysílání zvoleného zařízení
  - signály SS nebo CS, slave select nebo chip select, lib. datové piny
  - příklad propojení se třemi periferiemi
- je to rychlejší alternativa k I2C
  - mnohé periferie existují v I2C i SPI variantách, případně nabízí oboje současně
- vzhledem k relativní složitosti to nebude dále rozváděno, pro SPI-komunikující zařízení je často k dispozici stejně dedikovaná knihovna, která komplikace skryje



Další informace

- **Simon Monk: Hacking Electronics, 2017, 305 stran.**
  - **Stručný úvod do elektroniky, základních součástek a prvních triviálních experimentů, včetně různých mikrokontrolerů.**
- **Simon Monk: Programing Arduino Next Steps, 2019, 321 stran.**
  - **vysvětlení principů pro zkušenější začátečníky**