



UNIVERSITATEA DIN
BUCUREȘTI
— VIRTUTE ET SAPIENTIA —



Facultatea de
Matematică și Informatică
— Universitatea din București —

Introducere în Robotică


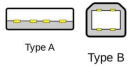

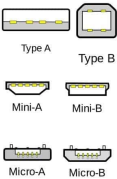

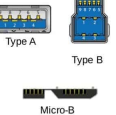

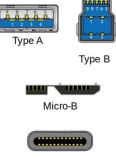

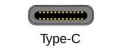
Cursul 8
USB, CAN, MODBUS

Facultatea de Matematică și Informatică
Universitatea București

Universal Serial Bus

- Standard definit în anii 90
- Unifică interfețe și alimentarea cu energie electrică
- USB 1.0 – 1996
- USB 2.0 – 2000
 - Viteze de transfer mai mari (480Mbps)
 - Conectori Mini-USB
 - USB On The Go (OTG) în 2006
- USB 3.0 – 2011
 - 5Gbps
 - Comunicație full-duplex

- USB 3.1 – 2014
 - 10Gbps
 - Conector unic, simetric
- USB 4 – 2019
 - 40Gbps
- USB 4, 2.0 – 2022
 - 120Gbps

| USB 1.0 12mbps | USB 2.0 480mbps | USB 3.2 Gen 1 (Previously 3.0, then 3.1 Gen 1) | USB 3.2 Gen 2 (Previously 3.1 Gen 2) | USB 3.2 Gen 2x2 (Previously 3.2) |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12mbps | 480mbps | 5gbps | 10gbps | 20gbps |
|   Type A Type B |   Type A Type B Mini-A Mini-B Micro-A Micro-B |   Type A Type B Micro-B |   Type A Type B Micro-B Type-C |   Type-C |

*Many of these connectors are designed to be backwards compatible, for example the Type-C connector will function even at USB 1.0 speeds.
What has been represented here is when one might commonly find a connector and the speed it was designed to support.

Avantaje pentru utilizator

Număr redus de conectori (A, B, micro AB, mini AB, C)

Fără cablu de alimentare

Număr extensibil de porturi

Rată de transfer mare

Hot plugging

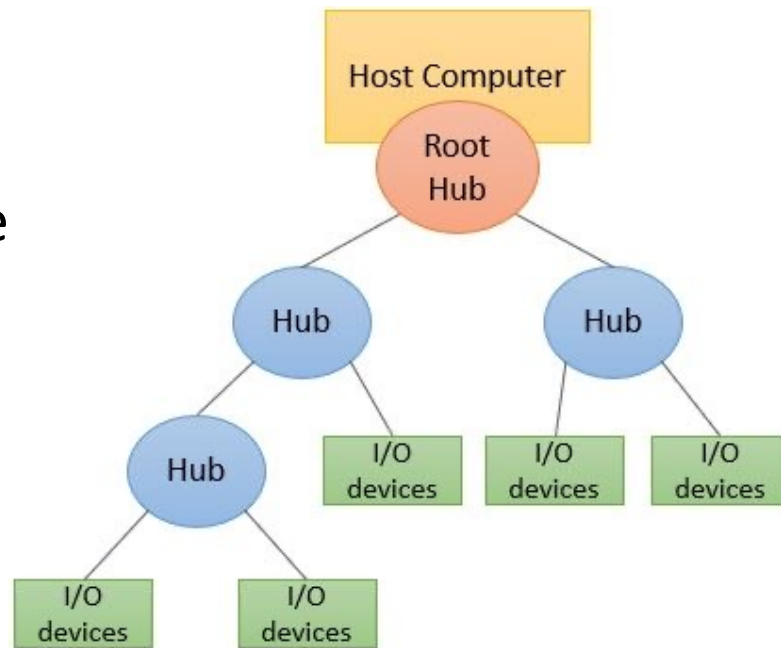
Fără configurare manuală

Pentru dezvoltator

- Avantaje
 - Fără alimentare separată
 - Interfațare unică pentru dispozitive diferite
 - Se pot implementa cipuri dedicate
 - Suport software în SO
 - Documentație din partea USB-IF
 - Dezavantaje
 - Complexitate mărită (de ex. față de RS232)
-

Arhitectură

- USB este un arbore
 - Perifericele sunt frunze
 - Host, hub-uri USB sunt nodurile
- Comunicație
 - Half-duplex
 - Condiționată de host
 - Punct la punct (host – device)



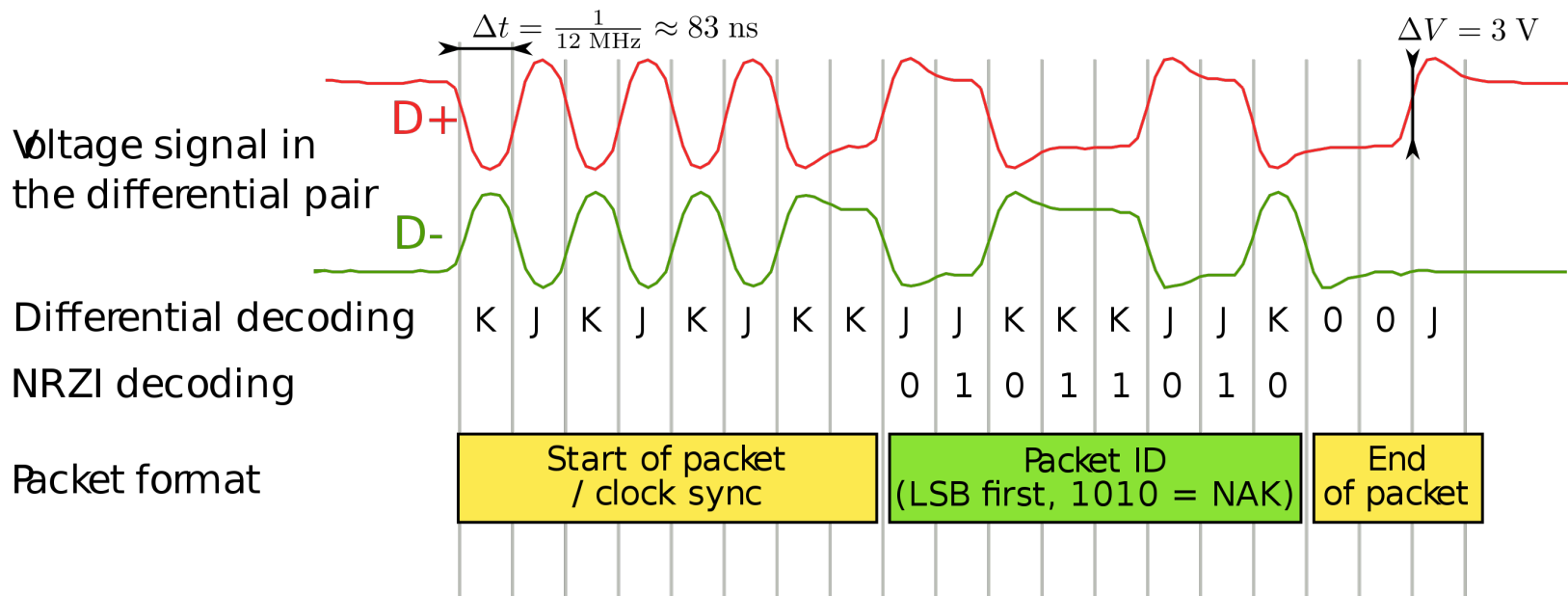
USB Host

- Cunoaște toate dispozitivele
 - Capabilitățile fiecăruia
 - Nevoile de transfer de date
 - Cerințele de consum
 - Administrează comunicația
 - Alocă lățime de bandă pentru dispozitive
 - Anunță comunicația dispozitiv-host
 - Interoghează dispozitive
-

USB Device

- Responsabil să se identifice host-ului
 - Canale de comunicație (endpoints – ports)
 - Recunoaște comunicația adresată lui
 - Răspunde la cererile host-ului
 - Menține consumul de energie în limitele declarate
 - Reduce consumul când magistrala e în Sleep
 - Comunică cu host-ul atunci când acesta cere
-

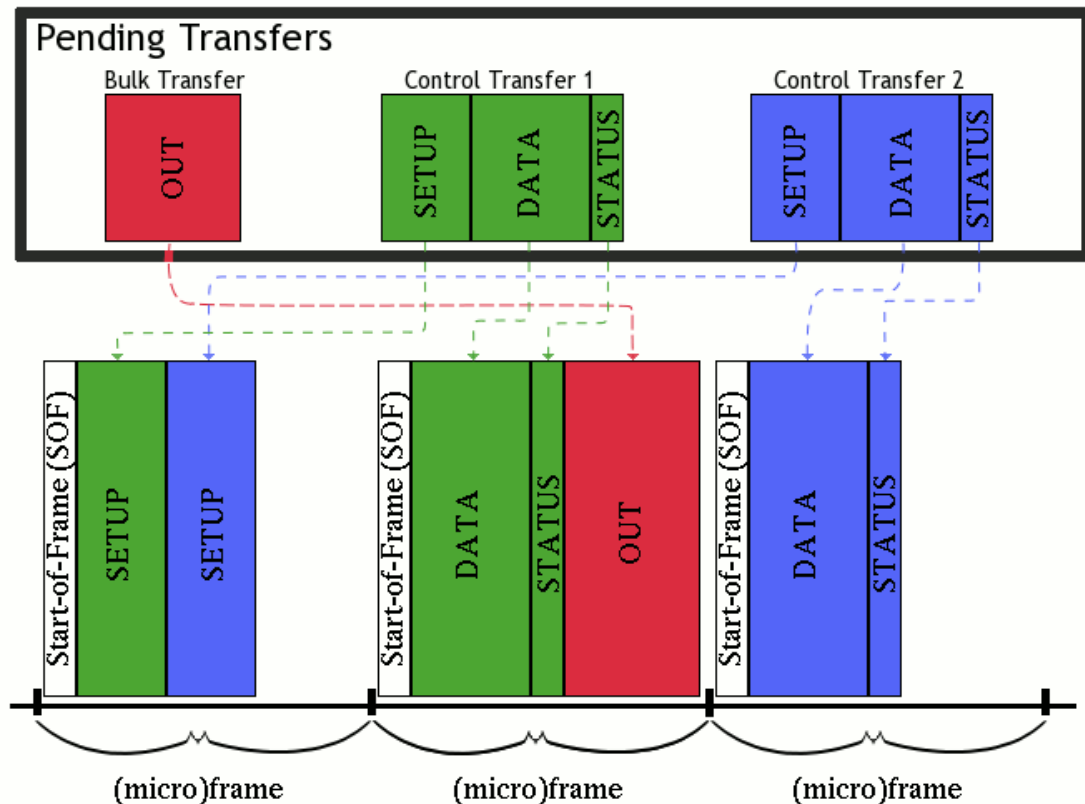
Comunicația pe USB



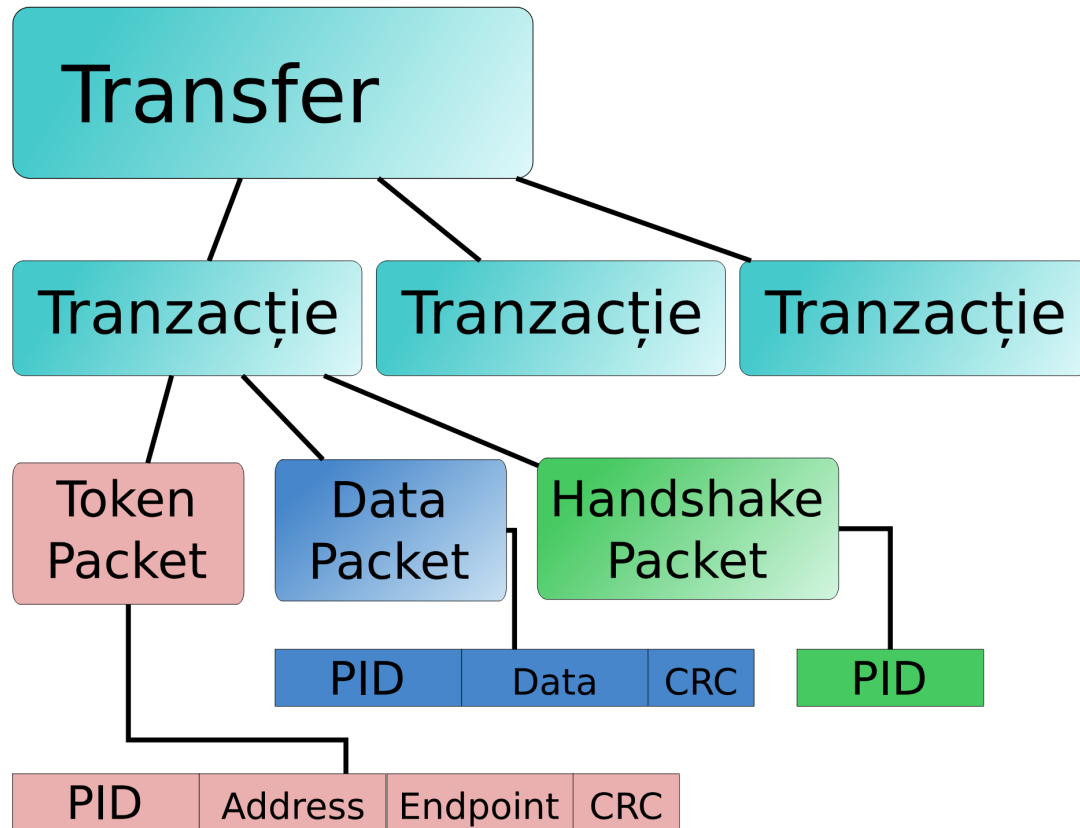
- Non Return to Zero Inverted (NRZI)
 - 0 bit is transmitted by toggling the data lines from J to K or vice versa.
 - 1 bit is transmitted by leaving the data lines as-is.

USB Frame

- Comunicația pe USB se face prin tranzații
- Orice tranzație trece prin un cadru de timp controlat de Host – numit Frame
- Lungimea și frecvența tranzațiilor depind de tipul transferului
 - Interrupt
 - Bulk
 - Isochronous
 - Control



Un transfer USB



Tipuri de transferuri

| Tip | Control | Interrupt | Isochronous | Bulk |
|------------------|----------------------------|-------------------|-------------|---------|
| Exemple folosire | Identificare / Configurare | Mouse Keyboard | Webcam | Storage |
| Obligatori | Da | Nu | Nu | Nu |
| Low-speed | Posibil | Posibil | Nu | Nu |
| Rată transfer | Medie | Mică | Medie | Mare |
| Trafic | Min. 20% | - | Max. 80% | - |

Endpoints

- Analog porturilor de la web sockets
 - USB endpoint este unidirecțional – poate fi IN sau OUT
 - Poate avea un singur tip de transfer – tip endpoint
 - Endpoint 0 – de control
 - Fiecare funcție a dispozitivului – endpoints
 - `/sys/bus.usb.devices` – conține toate informațiile
 - `$ cat /sys/bus/usb/devices/usb1/ep_00/type`
-

Enumerare

- Procedura de înregistrare a unui Device la Host
 - Dispozitivul este atașat fizic
 - Hub-ul
 - Detectează dispozitivul
 - Informează Host-ul
 - Detectează viteza cu care se poate comunica
 - Resetează dispozitivul
 - Host-ul află dacă dispozitivul este high-speed
 - Conectează dispozitivul la bus
 - Host-ul
 - Alocă o adresă
 - Cere descriptorii dispozitivului
 - Selectează un driver potrivit
 - Driver-ul selectează o configurație din cele disponibile
 - Încercați să urmăriți procesul prin Wireshark!
-

Descriptori

- Sunt structuri care prezintă capacitățile și configurările unui dispozitiv către Host
 - Device Descriptor
 - Vendor ID (VID), Product ID (PID), link către string descriptors cu numele
 - Nivelul de compatibilitate USB
 - Numărul de configurații posibile
 - Configuration Descriptor
 - Descrie un set de funcționalități care pot fi active simultan
 - Configurația curentă este selectată de Host bazându-se pe compatibilitate
 - Interface Descriptor
 - Endpoint Descriptor
 - String Descriptor
-

Exemplu descriptor

| Type | Field | Value |
|----------|-----------------|-----------------------------|
| uint8_t | bLength | 18 |
| uint8_t | bDescriptorType | 0x01 (Device descriptor) |
| uint16_t | bscUSB | 0x0200 |
| uint8_t | bDeviceClass | 0x00 (Composite) |
| uint8_t | bDeviceSubclass | 0x00 (No specific subclass) |
| uint8_t | bDeviceProtocol | 0x00 (No specific protocol) |
| uint8_t | bMaxPacketSize | 64 |
| uint8_t | idVendor | 0x03eb (VID of Atmel) |
| uint8_t | idProduct | 0x2018 |
| uint16_t | bcdDevice | 0x1000 |
| uint8_t | iManufacturer | 0 (no manufacturer string) |
| uint8_t | ... | ... |

Exemple de clase

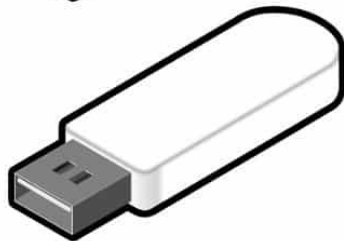
- Communication Device Class (CDC)
 - Abstract Control Model
 - /dev/tty/ACM0
 - Mass Storage
 - Stick-uri memorie
 - Video class – Webcams
 - Audio class – microfoane
 - Printer
-

How to plug in a USB key

Wrong



Wrong



Right



USB și AVR

- ATmega324p nu are interfață USB hardware (nici ATmega328)
 - Unele uC AVR au acest periferic, de ex ATmega32U4
 - Provocări
 - Funcționarea USB full-speed necesită un ceas de 48MHz
 - Transmisia la viteze superioare necesită mult buffering
 - Endpoint-uri limitate ca număr și memorie, deoarece orice endpoint are nevoie de un buffer
 - 1 control endpoint – 64 bytes
 - 1 endpoint de 256 bytes
 - 5 endpoints de 64 bytes
-

USB și AVR

- Dacă este prezent perifericul USB HW
 - Management alimentare
 - Management conectare bus + enumerare
 - Power management
 - Management endpoints
 - Management al memoriei alocate endpoints
 - 20 registre IO pentru control/status
 - Biblioteci:
 - LUFA: <https://www.fourwalledcubicle.com/LUFA.php>
 - Microchip: <https://www.microchip.com/en-us/development-tool/avr-usb-series6-software-packages>
-

Exemplu cod LUFA

```
/* Exemplu HID — emulator mouse/tastatura */

// (...) Structuri configurare

void main()
{
    SetupHardware();
    GlobalInterruptEnable();

    for (;;)
    {
        HID_Device_USBTask(&Generic_HID_Interface);
        USB_USBTask();
    }
}

// (...) SetupHardware (apeleaza si USB_Init)

// (...) Event callbacks (e.g. EVENT_USB_Device_Disconnect

CALLBACK_HID_Device_CreateReport(USB_ClassInfo_HID_Device_t* const HIDInterfaceInfo,
                                uint8_t* const ReportID,
                                const uint8_t ReportType,
                                void* ReportData,
                                uint16_t* const ReportSize)
{
    /* TODO date puse in ReportData si ReportSize */
}

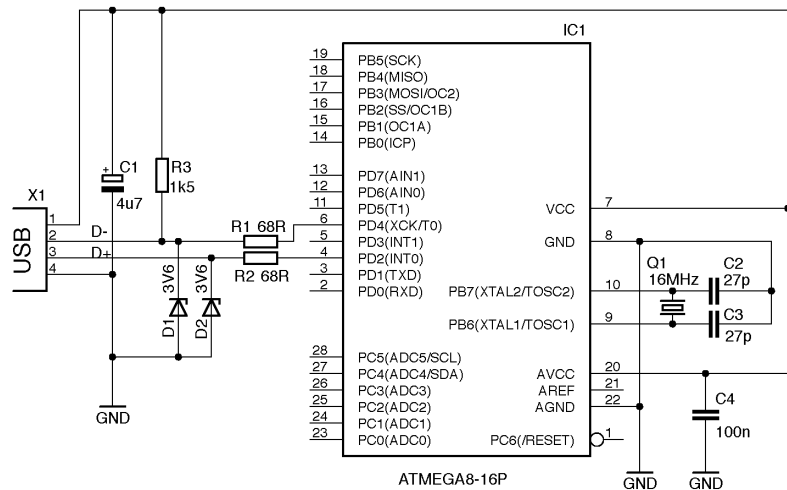
// (...) Callback host→device CALLBACK_HID_Device_ProcessHIDReport
```

Interfațare USB pe ATMega324p

- Putem folosi USB software
 - V-USB: <https://www.obdev.at/products/vusb/index.html>
 - Bit-banging
 - Implementează funcționalitate USB low-speed device
 - USB HID – mouse, tastatură, joystick
 - USB ACM – serială peste USB
 - Clase custom
 - Folosește pinii INT0 și INT1 (PD2 și PD3)
-

Exemplu conectare USB la ATMega324p

- Circuit minimalist
 - Alimentare direct din USB
 - Un pull-up pe D-
 - Două rezistențe 68-22 Ohmi pe liniile de date
 - Două diode Zener 3V6 pentru a clampa tensiunea pe liniile de date la maxim 3.6V



Exemplu cod V-USB

```
/* Exemplu HID – emulator mouse/tastatura */  
  
// (...) Structuri configurare ex: descriptorii  
  
void main()  
{  
    (...) // initialize  
  
    for (;;)   
    {  
        usbPoll();  
  
        if(usbInterruptIsReady())  
        {  
            /* TODO pus date in reportBuffer */  
  
            usbSetInterrupt((void *) &reportBuffer , sizeof(reportBuffer));  
        }  
    }  
}
```

Concluzii USB

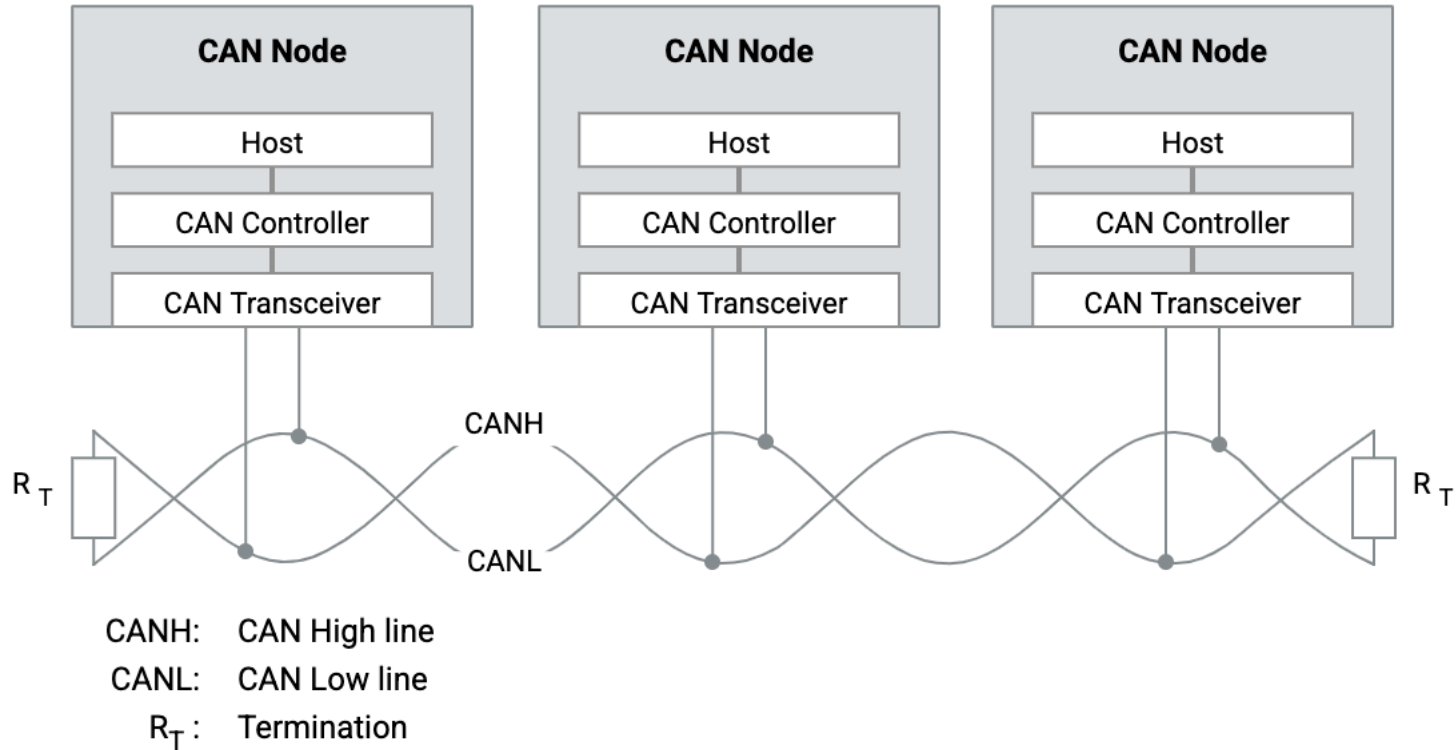
- Serială cu puține linii de legătură
 - Hot-plugging
 - Gestionare dificilă pe un controler cu putere de procesare limitată
 - Nu există periferic USB pe ATMega324p
 - Folosim biblioteci software sau un periferic dedicat (de ex. FT231XS)
-



CAN BUS

- Controller Area Network
 - Dezvoltat inițial de Bosch pentru piața automotive în anii 80
 - Acum este folosit pe scară largă (automotive + mediu industrial)
 - Gamă largă de producători de cipuri – NXP, Philips, Intel, Microchip etc.
 - Half-duplex, protocol bazat pe mesaje
 - Mai simplu și mai ieftin ca Ethernet, poate trimite pachete la distanțe similare
 - ISO-11898 – standardul care descrie CAN
 - High-Speed CAN – 1MBps (de ex. controlul suspensiei, motorului) și Low-Speed CAN – 125kbps (de ex. Control ștergătoare parbriz)
-

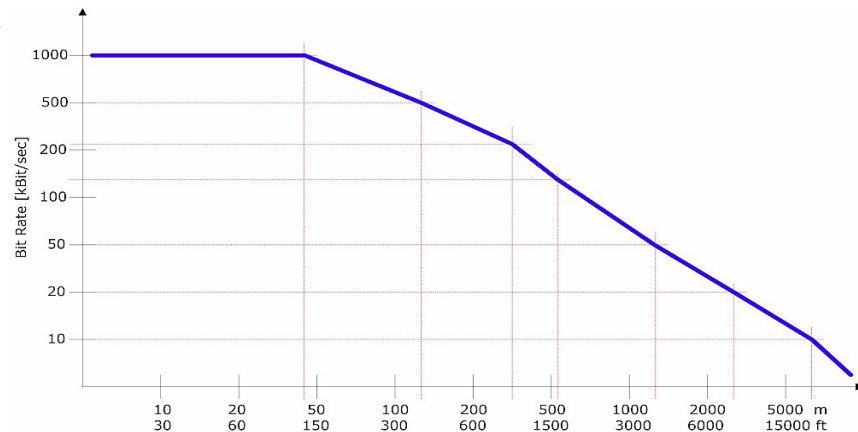
Rețea CAN – schema electrică



Cerințe de rețea

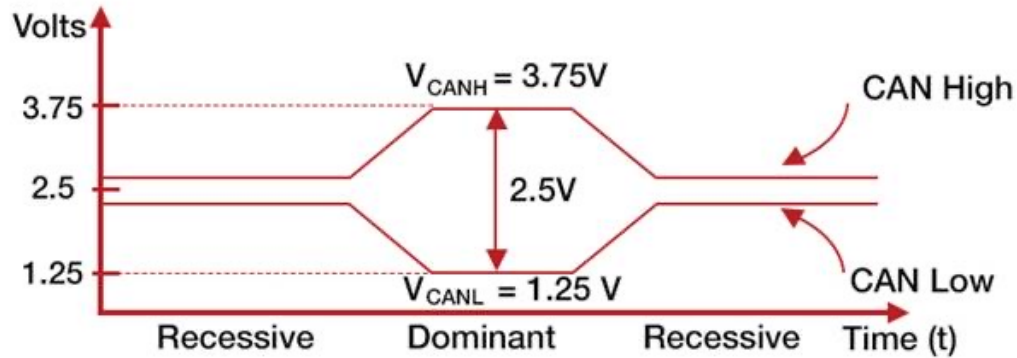
- Transmisia este diferențială
- Codificare NRZ
- Cablu torsadat cu o singură pereche – elimină interferențele electromagnetice
- Rezistențe de terminare de 120 Ohmi
- Recomandat să se lege maxim 30 de noduri într-o rețea
- Regula de bază: $signal_rate * cable_length \leq 50$

| Bus length (meters) | Signal rate (Mbps) |
|---------------------|--------------------|
| 40 | 1 |
| 100 | 0.5 |
| 500 | 0.10 |
| 1000 | 0.05 |

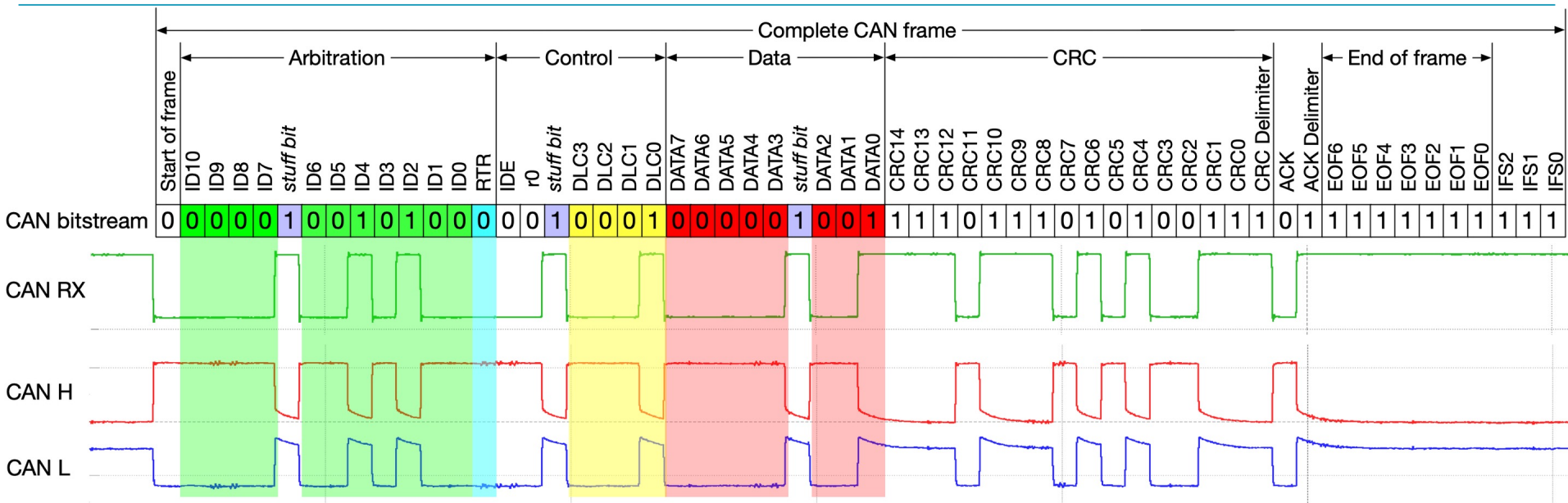


CAN Physical Layer

- Două tipuri de stări pe linia diferențială
 - Recesiv – corespunde lui 1 logic
 - Dominant – corespunde lui 0 logic



CAN Frame Format

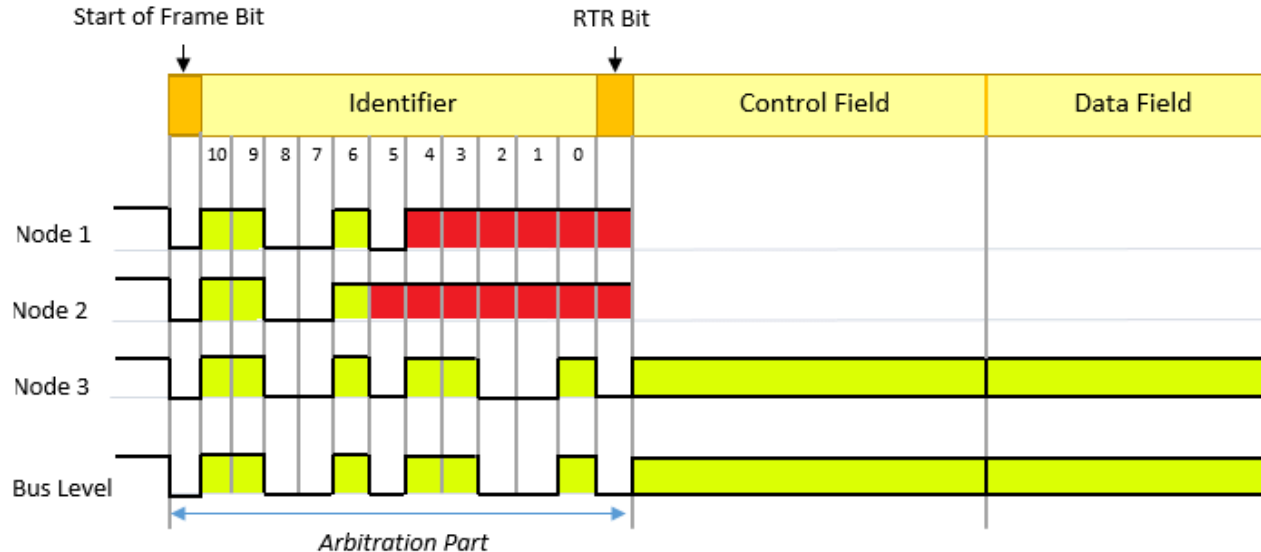


- Adresă (11 biți), Remote Transmission Request (RTR), Control (Data Length), Data bits, CRC, EOF
- Bit stuffing pentru a menține sincronizarea pe linia diferențială cu codificare NRZ
 - Transmițătorul adaugă un bit de nivel opus dacă detectează 5 biți consecutivi cu aceeași valoare

CAN Frame Format

- DLC – Data Length Code (4 biți) conține numărul de octeți de date transmiși în data frame
 - DATA – până la 64 de biți de date (8 octeți) pot fi transmiși
 - CRC – 16 biți
 - ACK - dacă datele recepționate sunt corecte, se răspunde cu bitul dominant (0 logic), dacă nu, cu bitul recesiv 1 logic (NACK).
 - EOF – End Of Frame, dezactivează și bit stuffing
 - IFS – Inter Frame Space este un timp necesar pentru procesarea datelor primite
-

CAN Arbitration



- Carrier Sense: un nod verifică starea magistralei înainte de a transmite. Dacă este în starea idle, atunci intră în comunicație
- Multiple Access / Collision Detect: evitarea coliziunilor când noduri multiple accesează magistrala simultan

Rezumat CAN

- Protocol facil și ieftin de comunicație multi-master multi-slave
 - Complexitate redusă a cablării rețelei – doar 2 linii torsadate
 - Capabilități de detecție și corecție a erorilor
 - Arbitrare CSMA/CD
 - High-speed – până la 1Mbps
 - Viteze și mai mari – Flex Ray (10Mbps)
-

CAN BUS?



CAN'T BUS

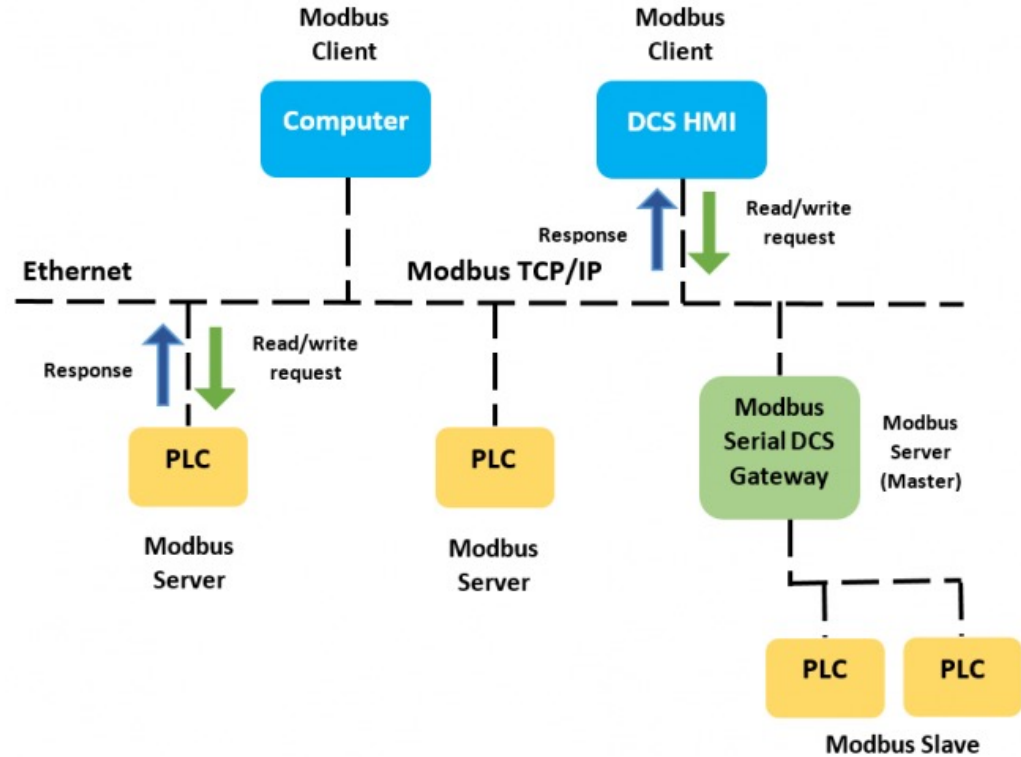
MODBUS

- Protocol serial de comunicație dezvoltat de Modicon (Schneider Electric) și publicat în 1979
- Ținta inițială a fost mediul industrial – comunicația cu PLC-uri
- Protocol deschis, tip request-reply
- Utilizat azi pe scară largă în mediul industrial
- Este protocolul standard de-facto pentru interfațarea dispozitivelor



Model Comunicație

- MODBUS este un protocol Client-Server (Master-Slave)
- Permite stabilirea unei rețele în care există un Server și mai mulți clienți
- La nivel fizic comunicația poate fi făcută peste linia serială (RS232 sau RS485) sau peste Ethernet



Tipuri MODBUS

- Există trei tipuri de comunicație MODBUS folosite pe scară largă
 - MODBUS RTU (Remote Terminal Unit)
 - Folosit pe scară largă în mediul industrial
 - Codificare binară a datelor
 - Peste RS232 sau RS485 (serială diferențială)
 - Baud rate de la 1200bps la 115200bps (depinde de distanță)
 - MODBUS TCP/IP
 - MODBUS ASCII
-

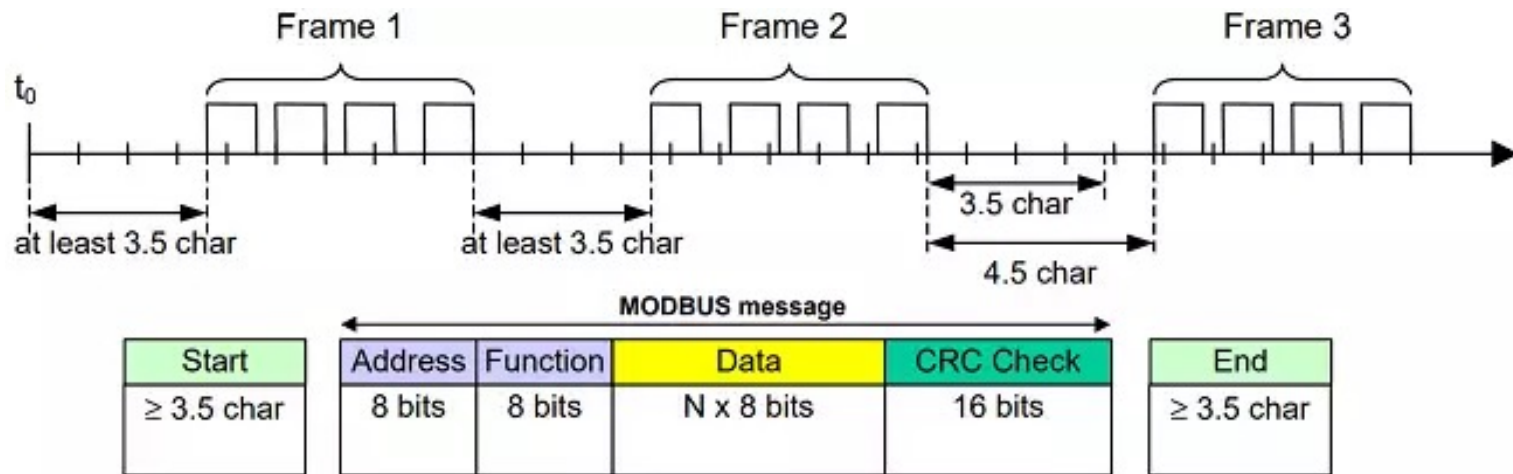
Structura de date MODBUS

- Fiecare dispozitiv MODBUS stochează (minim) următoarele informații în 4 tabele
 - Două tabele pentru valori discrete on/off (coils)
 - Două tabele cu valori numerice (registre)
 - Tabelele au unele valori read-only și valori read-write
 - Fiecare tabelă are 9999 valori
 - Fiecare *coil* este codificat ca un singur bit (on/off) și are o adresă de la 0x0000 la 0x270E
 - Fiecare registru este un cuvânt de 16 biți

| Coil/Register Numbers | Data Addresses | Type | Table Name |
|-----------------------|----------------|------------|---------------------------------|
| 1-9999 | 0000 to 270E | Read-Write | Discrete Output Coils |
| 10001-19999 | 0000 to 270E | Read-Only | Discrete Input Contacts |
| 30001-39999 | 0000 to 270E | Read-Only | Analog Input Registers |
| 40001-49999 | 0000 to 270E | Read-Write | Analog Output Holding Registers |

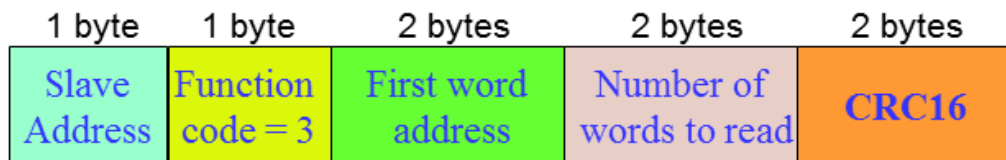
MODBUS RTU Data Frame

- Serverul trimite un cadru de date cu următoarea structură
 - Adresa dispozitiv: 0 = broadcast, 1-247 = adrese valide
 - Cod funcție (0x01-Read Coils, 0x02-Read Inputs, 0x03-Read Holding Registers etc.)
 - Date
 - CRC
 - Liniște (cel puțin 3.5 caractere)

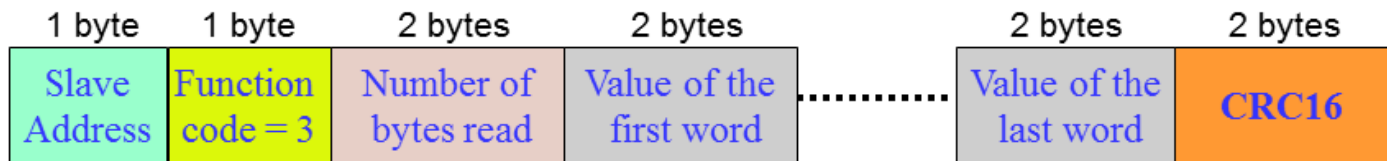


Exemplu comunicație

Request :



Response :



MODBUS Request

Să trimitem o comandă care cere starea contactelor (On/Off) de la 20 la 56 de la un client cu adresa 17

Codificarea binară a cadrului RTU: 11 01 0013 0025 0E84

11: The Slave Address (11 hex = address 17)

01: The Function Code 1 (read Coil Status)

0013: The Data Address of the first coil to read.
(0013 hex = 19 , + 1 offset = coil #20)

0025: The total number of coils requested. (25 hex = 37, inputs 20 to 56)

0E84: The CRC (cyclic redundancy check) for error checking.

| Command | Function Code |
|---------|---------------------------------------------------|
| 01 | Read Coils |
| 02 | Read Discrete Inputs |
| 03 | Read Holding Registers |
| 04 | Read Input Registers |
| 05 | Write Single Coil |
| 06 | Write Single Register |
| 07 | Read Exception Status |
| 08 | Diagnostics |
| * | |
| * | |
| Xx | Up to 255 function codes, depending on the device |

MODBUS Reply

11 01 05 CD6BB20E1B 45E6

11: The Slave Address (11 hex = address17)

01: The Function Code 1 (read Coil Status)

05: The number of data bytes to follow (37 Coils / 8 bits per byte = 5 bytes)

CD: Coils 27 - 20 (1100 1101)

6B: Coils 35 - 28 (0110 1011)

B2: Coils 43 - 36 (1011 0010)

0E: Coils 51 - 44 (0000 1110)

1B: 3 space holders & Coils 56 - 52 (0001 1011)

45E6: The CRC (cyclic redundancy check).



**MODBUS
RTU**



**MODBUS
TCP**

MODBUS TCP/IP

- Folosește TCP/IP și Ethernet ca mijloc de transmisie a pachetelor MODBUS
- Permite utilizarea infrastructurii de rețea pentru conectarea de dispozitive programabile și pentru automatizare
- Folosește protocolul TCP/IP (portul 502) pentru managementul transmisiei de pachete și al verificării/corectării erorilor

