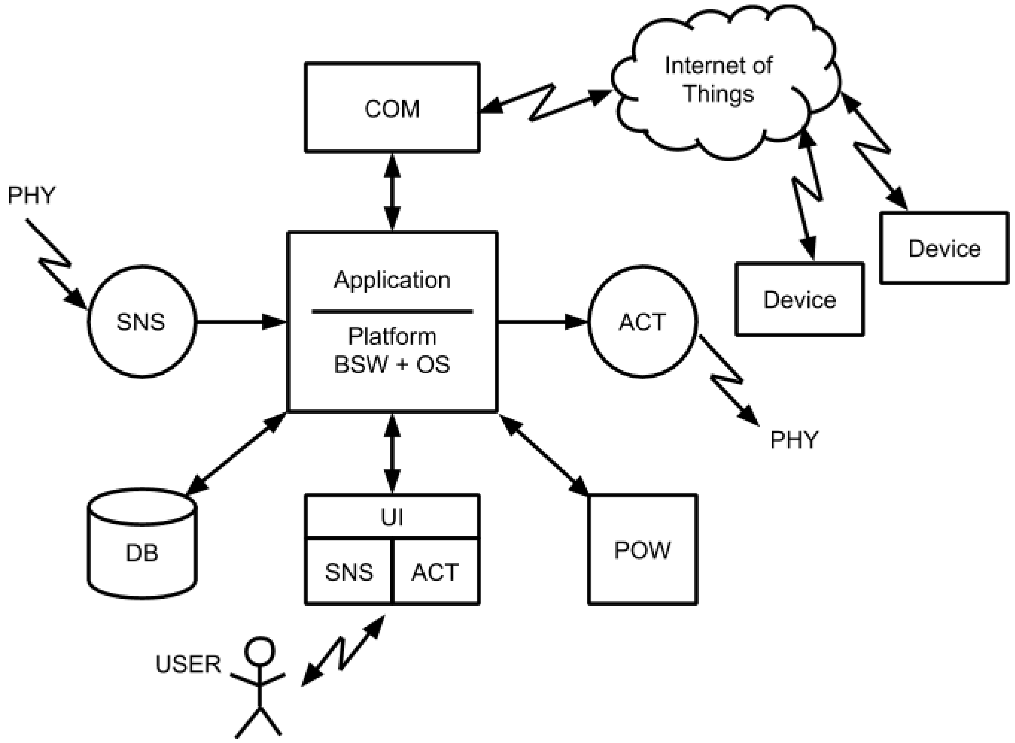
Internetul Lucrurilor

Comunicare
Interconectare echipamente

Comunicare

Schimb de informație între interlocutori

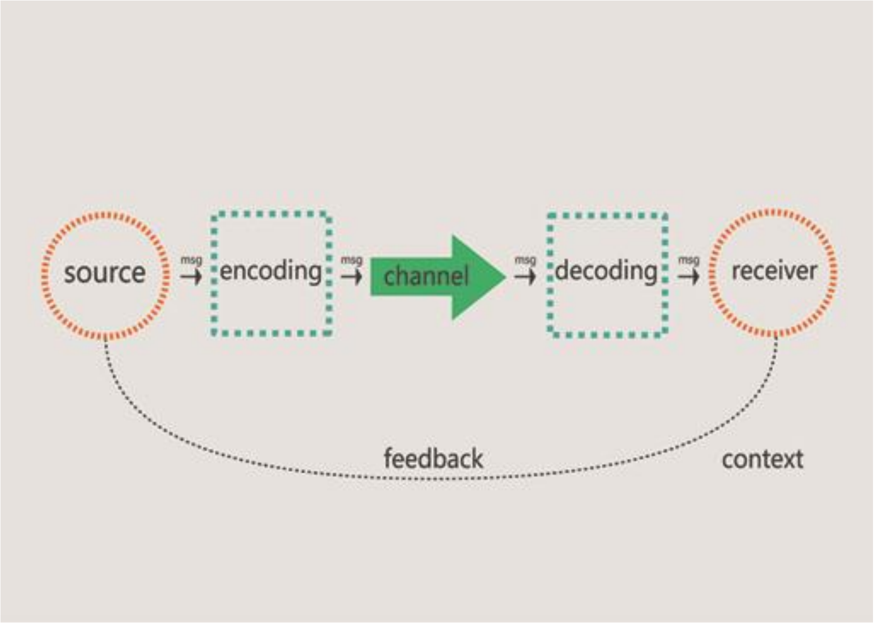
- Notiune de comunicare
- Mediu de transmise
- Topologie retea
- Protocol fizic
- Protocol logic
- Internet/Clouding



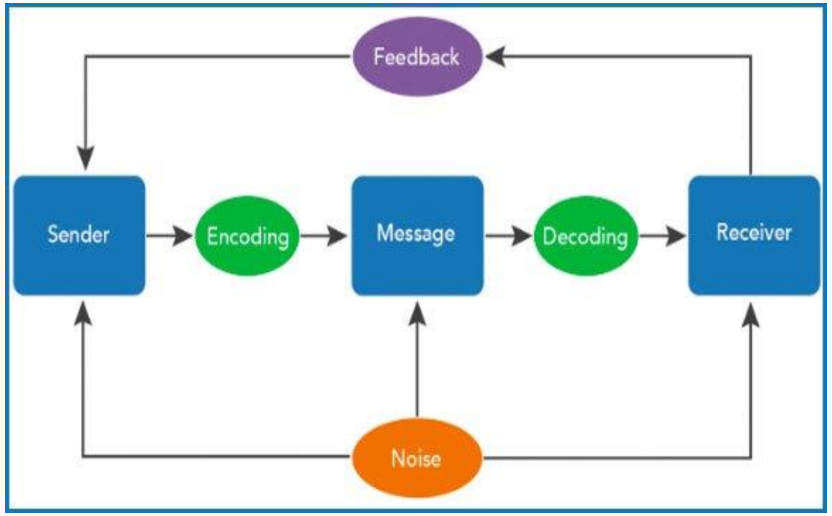
Notiune de comunicare

Schimb de informație între interlocutori

- Mesaj
- Emițător
- Codare
- Canal
- Decodare
- Receptor
- Raspuns
- Zgomot



https://www.open.edu/openlearn/ocw/mod/oucontent/view.php?id=87012&section=4

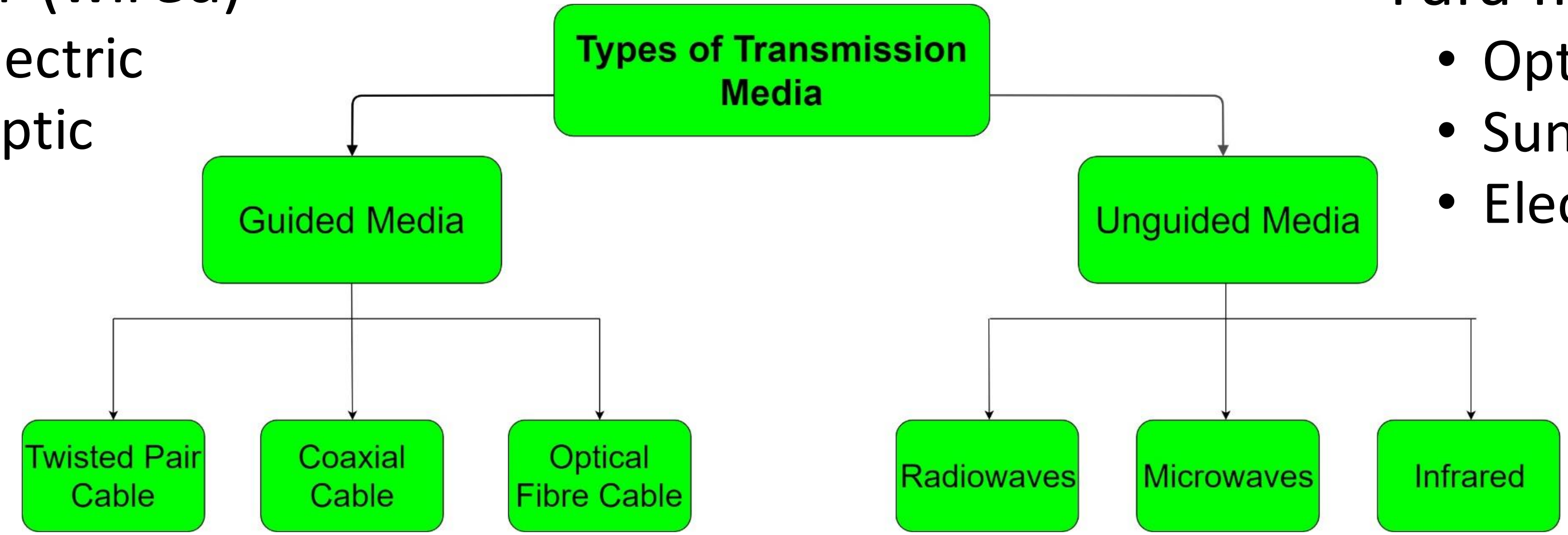


https://learntechit.com/the-process-of-communication/

Mediu de comunicare

- Cu fir (wired)
  - Electric
  - Optic

- Fara fir (wireless)
  - Optic
  - Sunet
  - Electromagnetic



Types of Transmission Media

Guided Media

Twisted Pair Cable

Coaxial Cable

Optical Fibre Cable

Unguided Media

Radiowaves

Microwaves

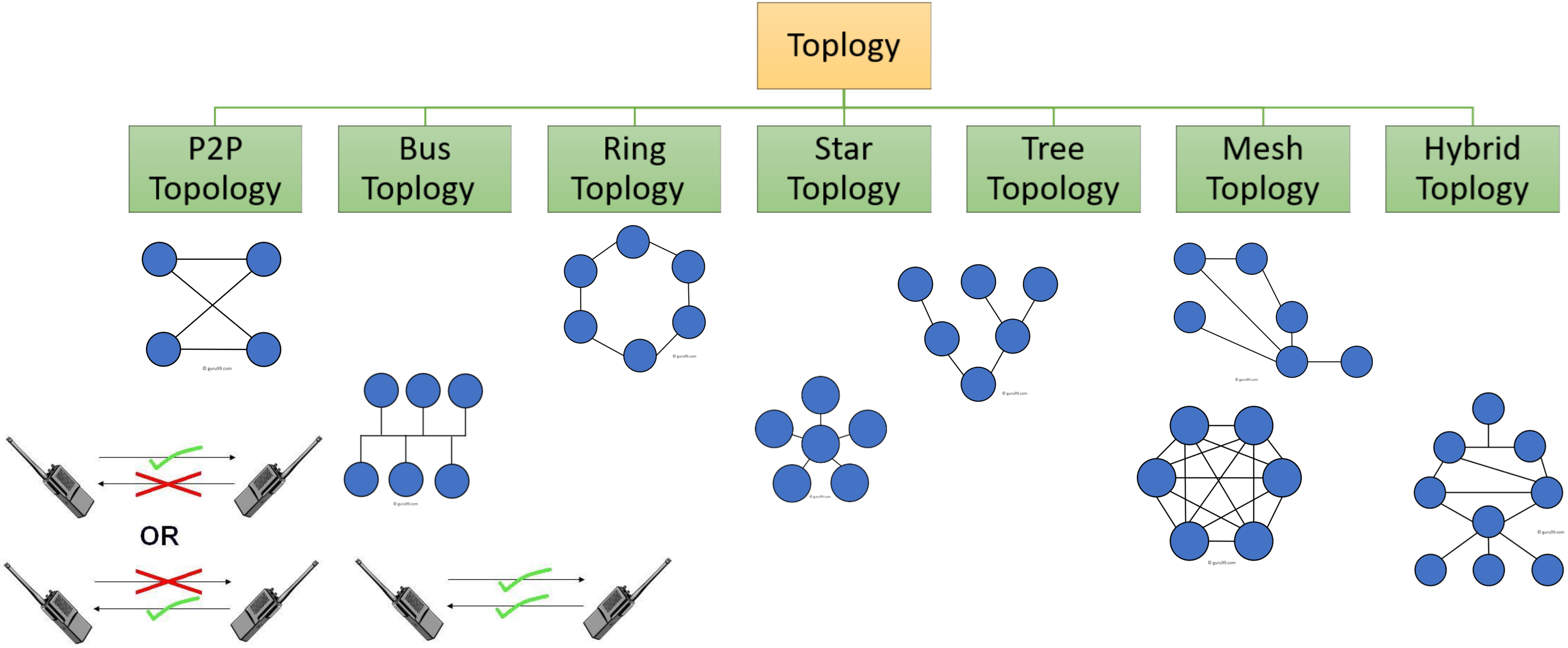
Infrared

https://www.geeksforgeeks.org/types-transmission-media/

https://en.wikipedia.org/wiki/List_of_interface_bit_rates

https://www.electronicdesign.com/technologies/communications/article/21800967/serial-io-interfaces-dominate-data-communications

Topologie Rețea



Toplogy

P2P Topology

Bus Toplogy

Ring Toplogy

Star Toplogy

Tree Topology

Mesh Toplogy

Hybrid Toplogy

Half Duplex

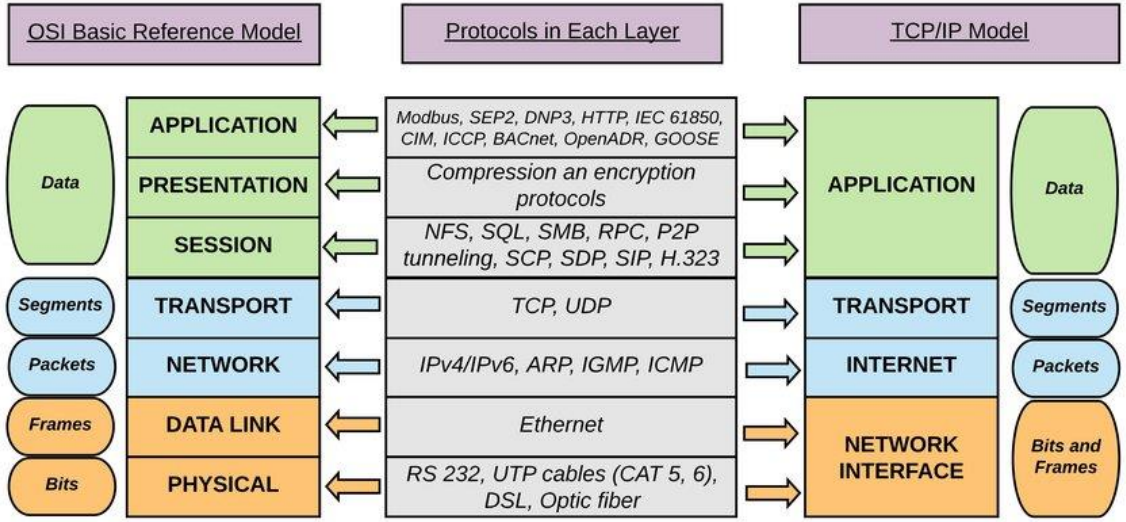
Full Duplex

https://en.wikipedia.org/wiki/Duplex_(telecommunications)

https://www.guru99.com/type-of-network-topology.html

Protocol de comunicare

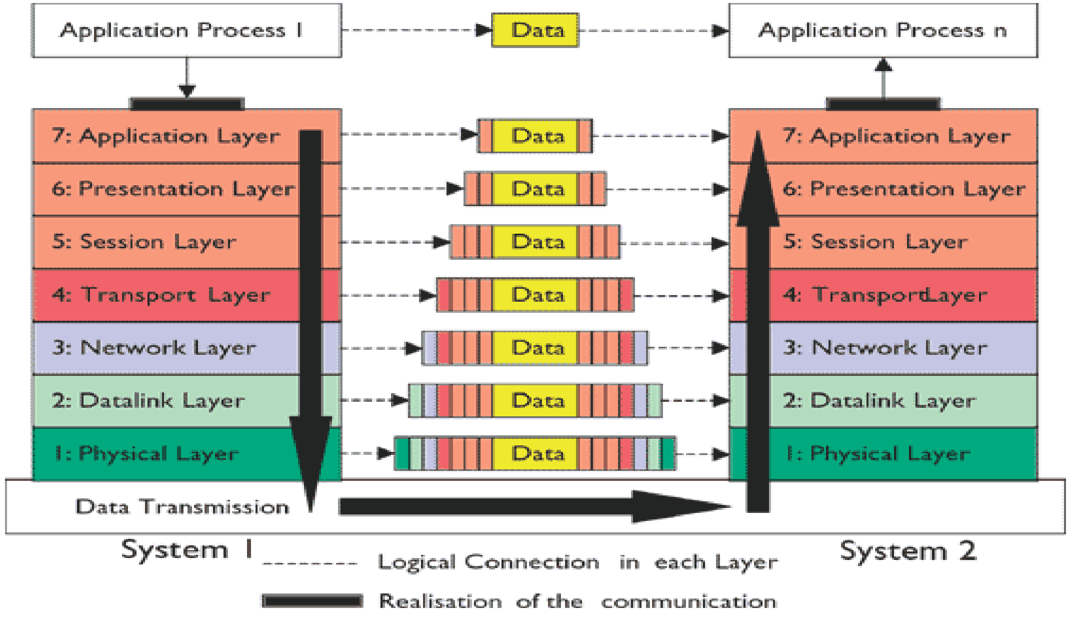
Un set de reguli agrea intre interlocutori pentru a asigura transmiterea sigura a informatiei



OSI Basic Reference Model		Protocols in Each Layer	TCP/IP Model	
Data	APPLICATION	Modbus, SEP2, DNP3, HTTP, IEC 61850, CIM, ICCP, BACnet, OpenADR, GOOSE	APPLICATION	Data
	PRESENTATION	Compression an encryption protocols		
	SESSION	NFS, SQL, SMB, RPC, P2P tunneling, SCP, SDP, SIP, H.323		
Segments	TRANSPORT	TCP, UDP	TRANSPORT	Segments
Packets	NETWORK	IPv4/IPv6, ARP, IGMP, ICMP	INTERNET	Packets
Frames	DATA LINK	Ethernet	NETWORK INTERFACE	Bits and Frames
Bits	PHYSICAL	RS 232, UTP cables (CAT 5, 6), DSL, Optic fiber		

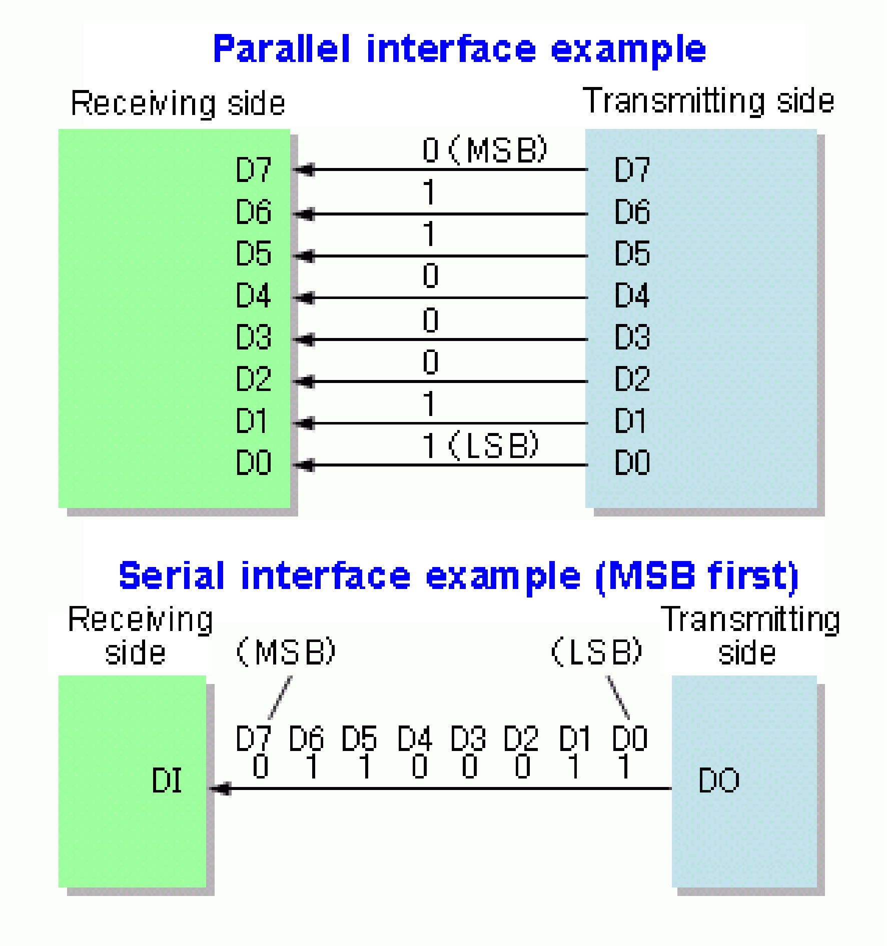
- Protocoale fizice
- Protocoale logice

Mesaj - structura de date impachetată conform protocolului specific de comunicare



https://iebmedia.com/index.php?id=4582&parentid=63&themeid=255&showdetail=true

Protocoale Fizice – Serial vs Paralel



ARINC 818 Avionics Digital Video Bus

Atari SIO (Joe Decuir credits his work on Atari SIO as the basis of USB)

Binary Synchronous Communications BSC - Binary Synchronous Communications

CAN Control Area Network Vehicle Bus

ccTalk Used in the money transaction and point-of-sale industry

CoaXPress industrial camera protocol over Coax

DMX512 control of theatrical lighting

Ethernet

Fibre Channel (high-speed, for connecting computers to mass storage devices)

FireWire

HyperTransport

InfiniBand (very high speed, broadly comparable in scope to PCI)

I²C multidrop serial bus

MIDI control of electronic musical instruments

MIL-STD-1553A/B

Morse code telegraphy

PCI Express

Profibus

RS-232 (low-speed, implemented by serial ports)

RS-422 multidrop serial bus

RS-423

RS-485 multidrop multimaster serial bus

SDI-12 industrial sensor protocol

Serial ATA

Serial Attached SCSI

SONET and SDH (high speed telecommunication over optical fibers)

SpaceWire Spacecraft communication network

SPI

T-1, E-1 and variants (high speed telecommunication over copper pairs)

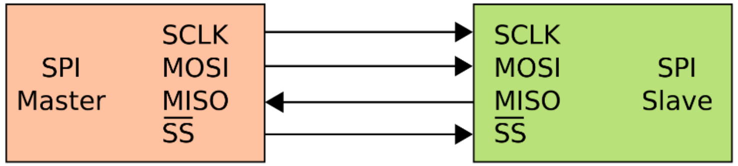
Universal Serial Bus (for connecting peripherals to computers)

UNI/O multidrop serial bus

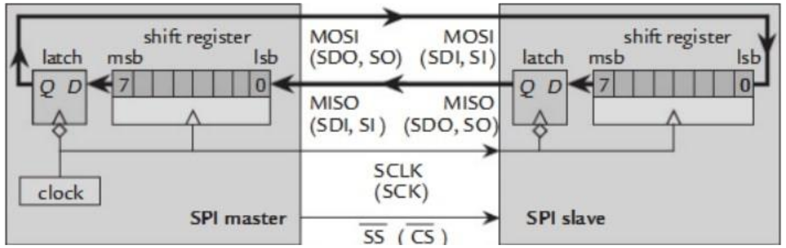
1-Wire multidrop serial bus

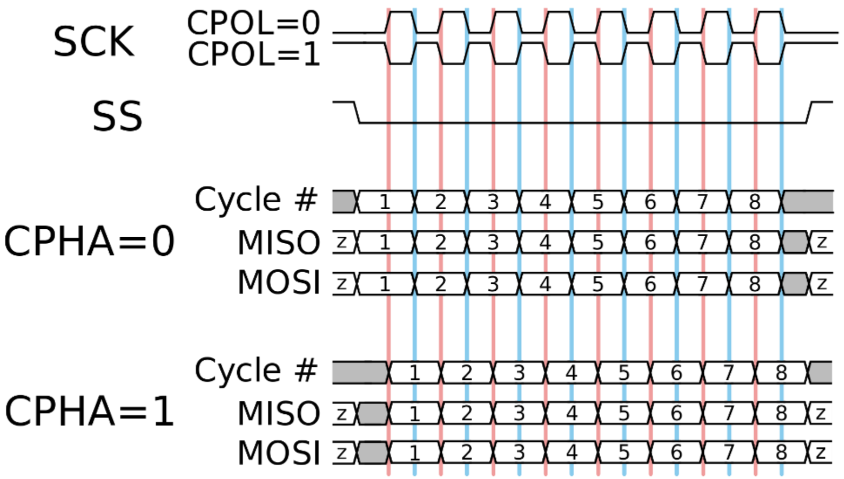
https://en.wikipedia.org/wiki/Serial_communication https://en.wikipedia.org/wiki/Parallel_communication https://en.wikipedia.org/wiki/Wireless_network

Protocoale Fizice - SPI

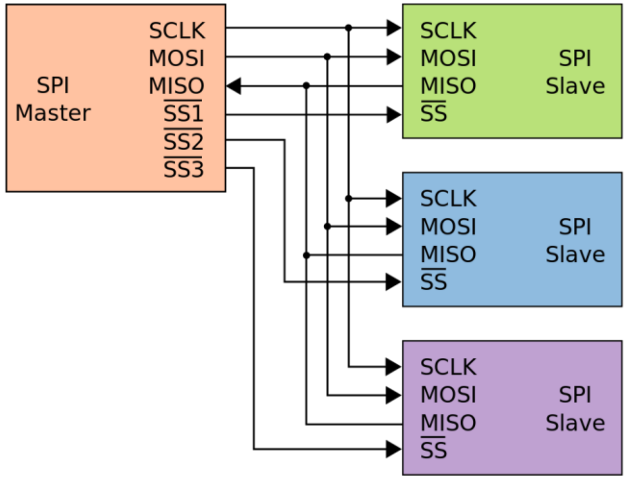

SPI Master
SCLK
MOSI
MISO
SS
SCLK
MOSI
MISO
SS
SPI Slave

Peer to peer

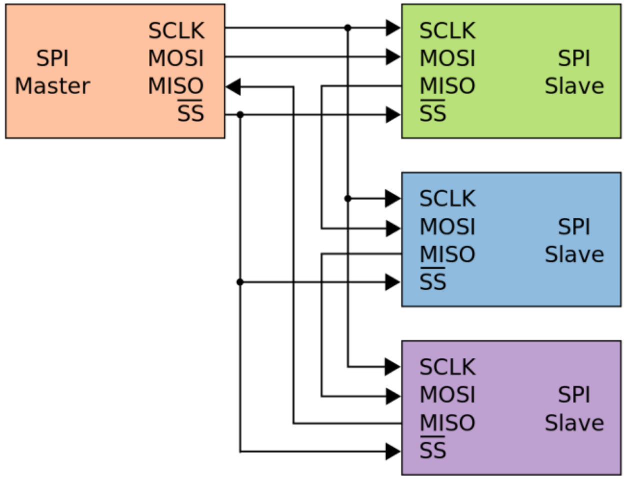



SCK
CPOL=0
CPOL=1
SS
CPHA=0
Cycle #
MISO
MOSI
CPHA=1
Cycle #
MISO
MOSI

https://microcontrollerslab.com/introduction-to-spi-communication-protocol/


SPI Master
SCLK
MOSI
MISO
SS1
SS2
SS3
SCLK
MOSI
MISO
SS
SPI Slave
SCLK
MOSI
MISO
SS
SPI Slave
SCLK
MOSI
MISO
SS
SPI Slave

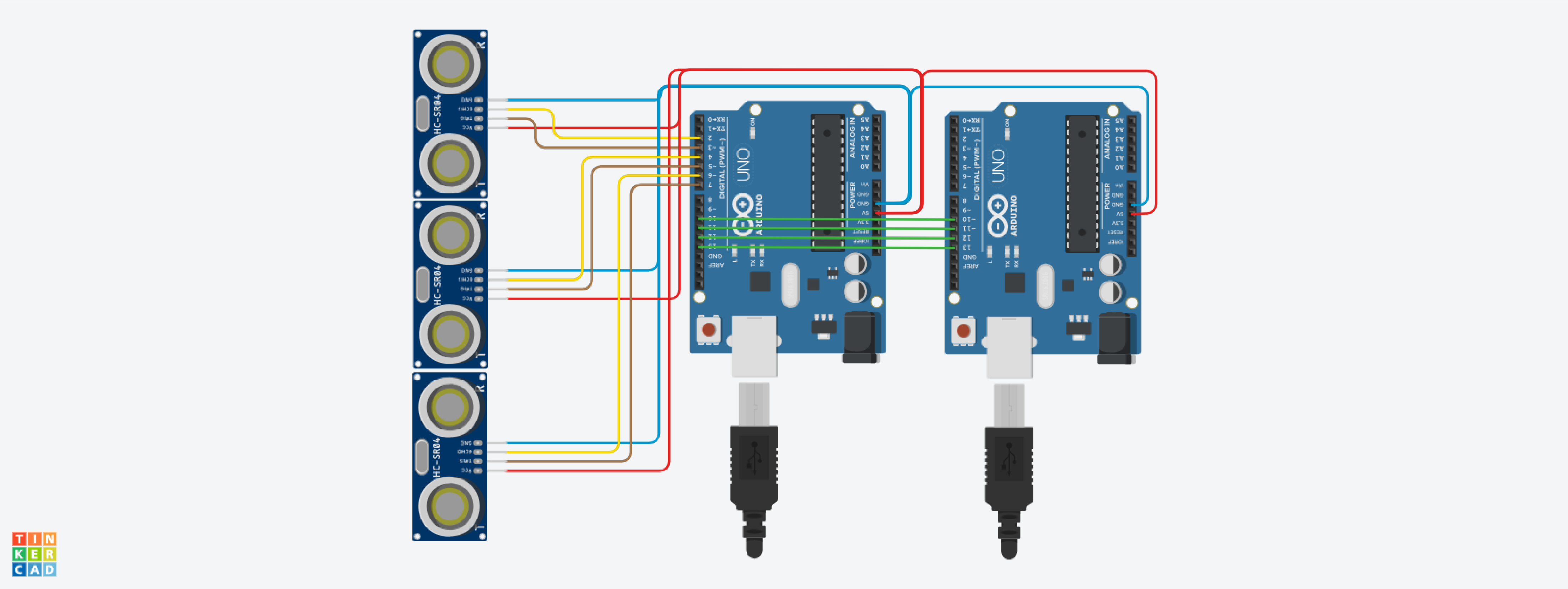
paralel


SPI Master
SCLK
MOSI
MISO
SS
SCLK
MOSI
MISO
SS
SPI Slave
SCLK
MOSI
MISO
SS
SPI Slave
SCLK
MOSI
MISO
SS
SPI Slave

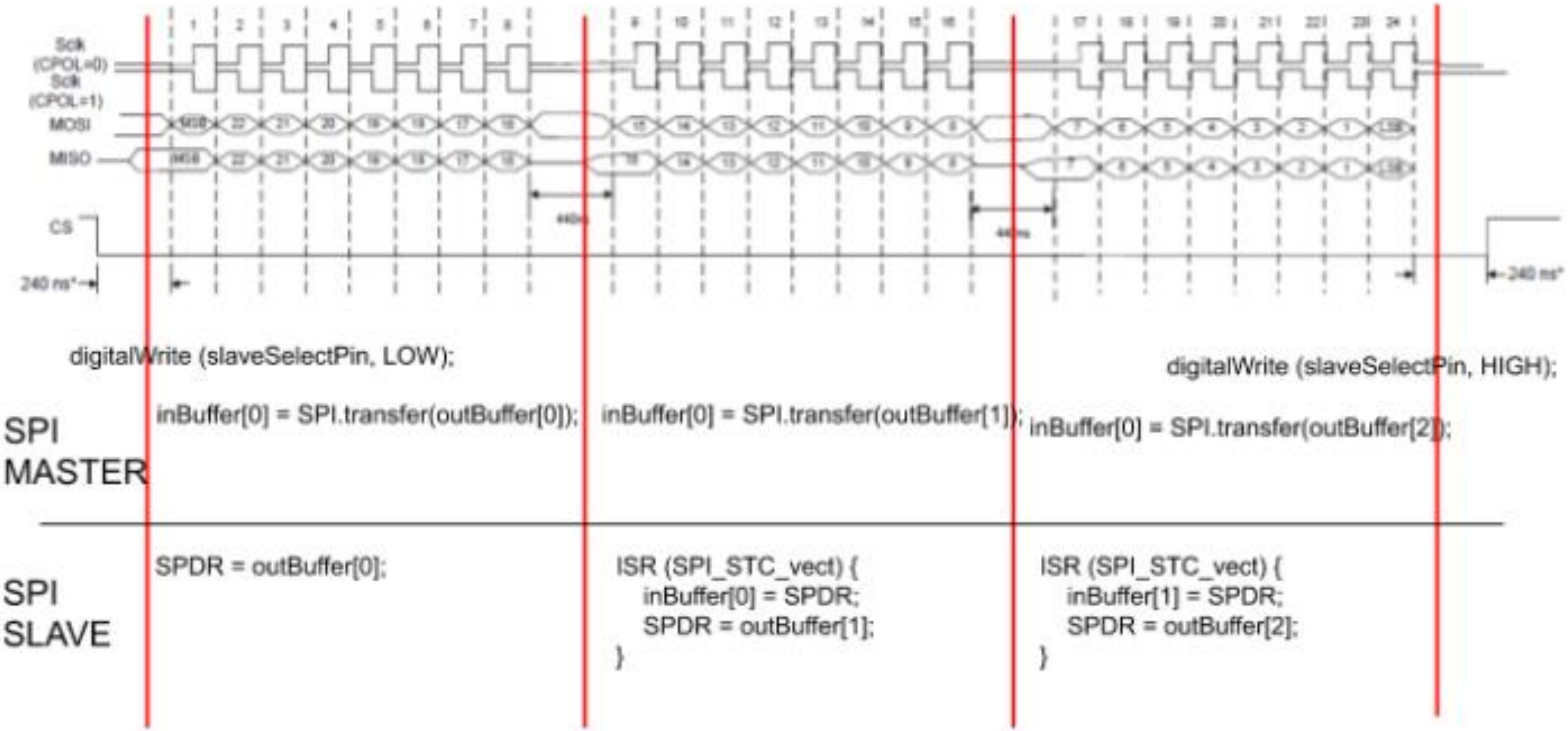
serial
în lant

https://en.wikipedia.org/wiki/Serial_Peripheral_Interface

SPI – Digital Ultrasonic Sensor HCS-04



SPI -Protocol



digitalWrite (slaveSelectPin, LOW);

inBuffer[0] = SPI.transfer(outBuffer[0]);

inBuffer[0] = SPI.transfer(outBuffer[1]);

inBuffer[0] = SPI.transfer(outBuffer[2]);

digitalWrite (slaveSelectPin, HIGH);

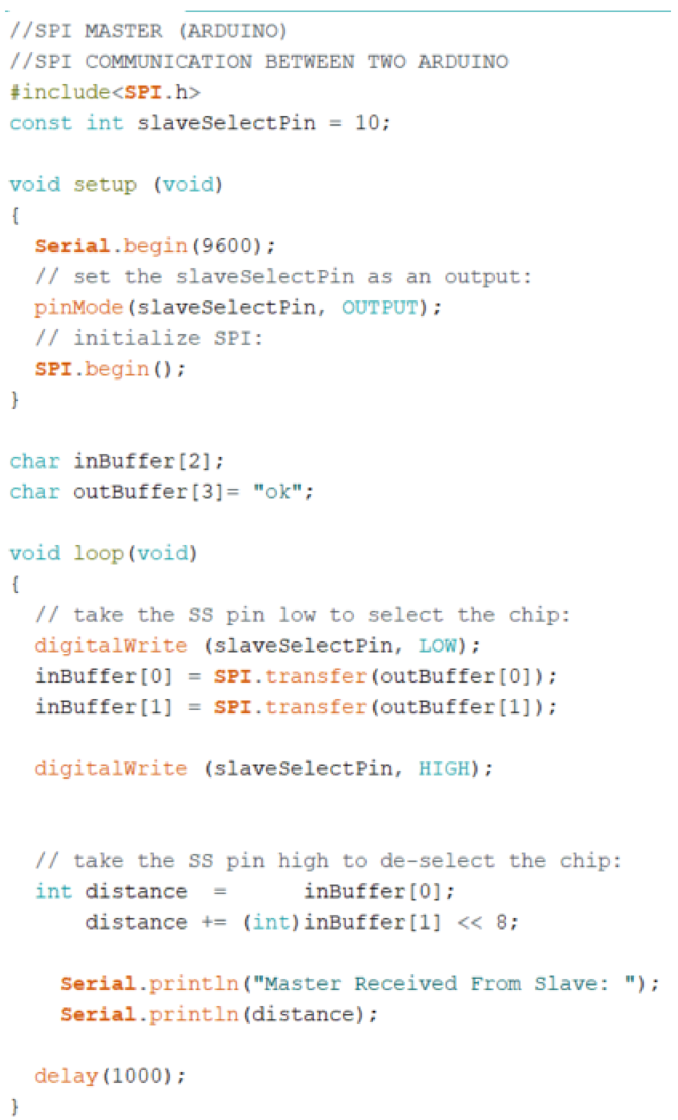
```
SPDR = outBuffer[0];
```

```
ISR (SPI_STC_vect) {
   inBuffer[0] = SPDR;
   SPDR = outBuffer[1];
}
```

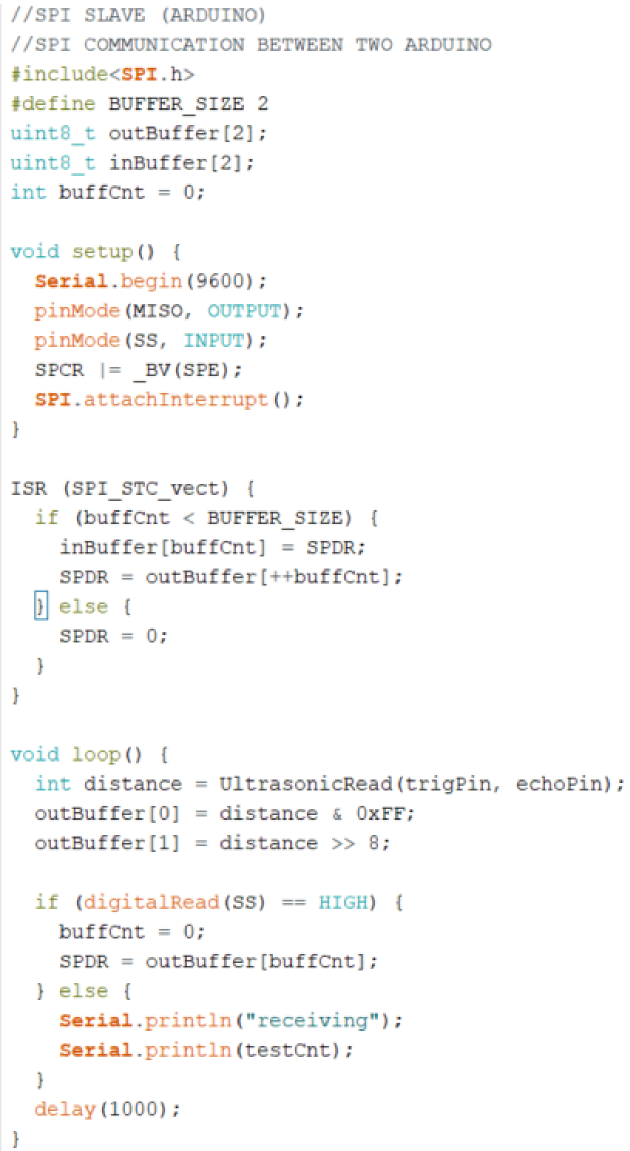
```
ISR (SPI_STC_vect) {
   inBuffer[1] = SPDR;
   SPDR = outBuffer[2];
}
```

https://wiki.analog.com/resources/eval/sdp/sdp-b/peripherals/spi

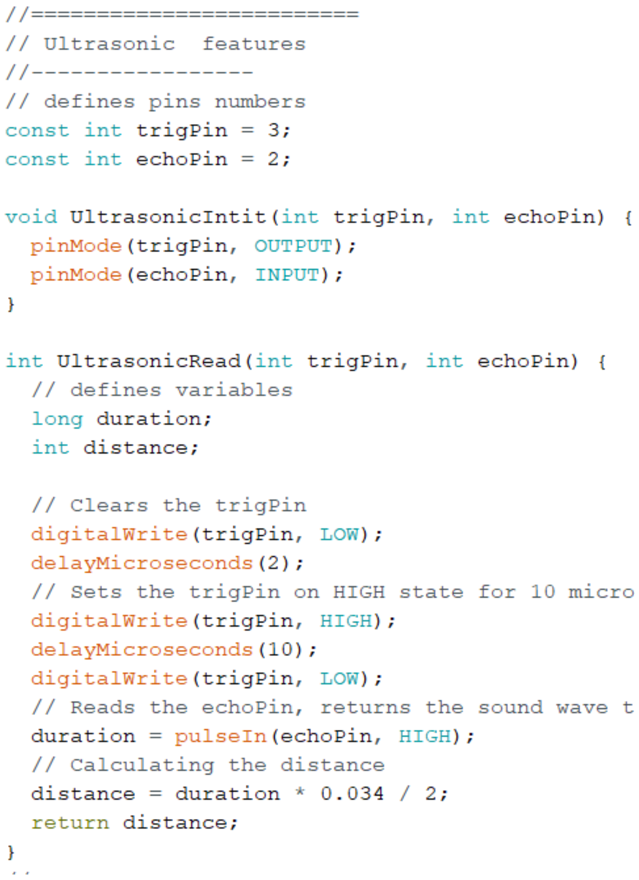
SPI – Digital Sensor Implementare



```
//SPI MASTER (ARDUINO)
//SPI COMMUNICATION BETWEEN TWO ARDUINO
#include<SPI.h>
const int slaveSelectPin = 10;

void setup (void)
{
  Serial.begin(9600);
  // set the slaveSelectPin as an output:
  pinMode(slaveSelectPin, OUTPUT);
  // initialize SPI:
  SPI.begin();
}

char inBuffer[2];
char outBuffer[3]= "ok";

void loop(void)
{
  // take the SS pin low to select the chip:
  digitalWrite (slaveSelectPin, LOW);
  inBuffer[0] = SPI.transfer(outBuffer[0]);
  inBuffer[1] = SPI.transfer(outBuffer[1]);

  digitalWrite (slaveSelectPin, HIGH);


  // take the SS pin high to de-select the chip:
  int distance  =      inBuffer[0];
      distance += (int)inBuffer[1] << 8;

    Serial.println("Master Received From Slave: ");
    Serial.println(distance);

  delay(1000);
}
```

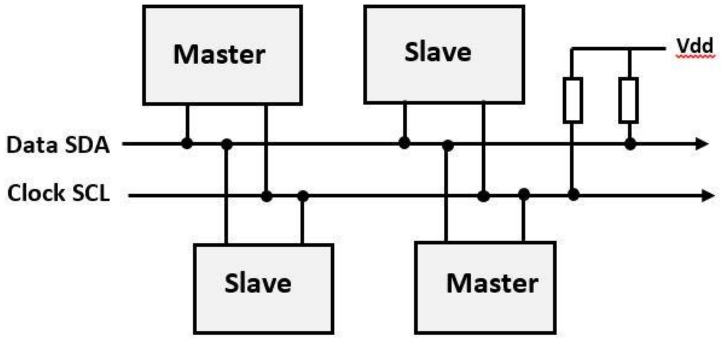


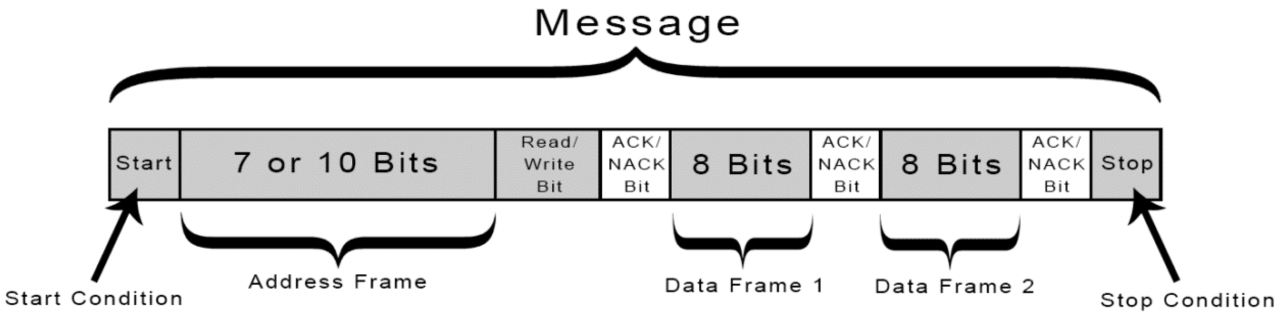
```
//SPI SLAVE (ARDUINO)
//SPI COMMUNICATION BETWEEN TWO ARDUINO
#include<SPI.h>
#define BUFFER_SIZE 2
uint8_t outBuffer[2];
uint8_t inBuffer[2];
int buffCnt = 0;

void setup() {
  Serial.begin(9600);
  pinMode(MISO, OUTPUT);
  pinMode(SS, INPUT);
  SPCR |= _BV(SPE);
  SPI.attachInterrupt();
}

ISR (SPI_STC_vect) {
  if (buffCnt < BUFFER_SIZE) {
    inBuffer[buffCnt] = SPDR;
    SPDR = outBuffer[++buffCnt];
  } else {
    SPDR = 0;
  }
}

void loop() {
  int distance = UltrasonicRead(trigPin, echoPin);
  outBuffer[0] = distance & 0xFF;
  outBuffer[1] = distance >> 8;

  if (digitalRead(SS) == HIGH) {
    buffCnt = 0;
    SPDR = outBuffer[buffCnt];
  } else {
    Serial.println("receiving");
    Serial.println(testCnt);
  }
  delay(1000);
}
```

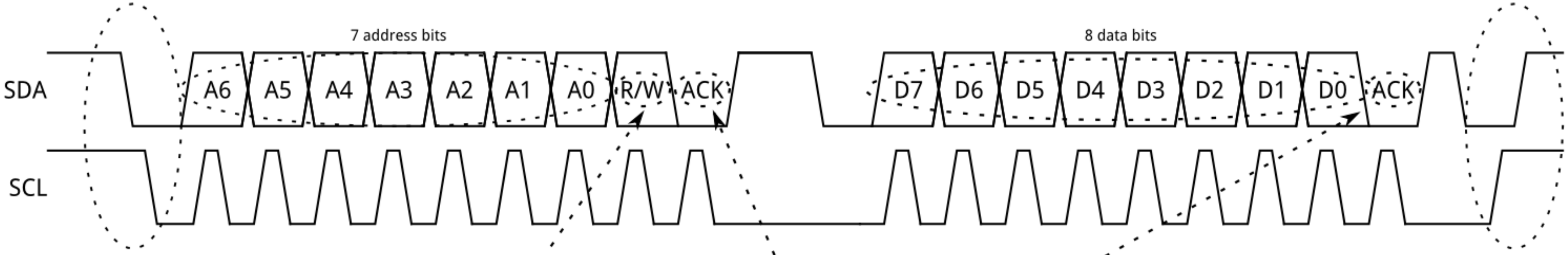


```
//=========================
// Ultrasonic  features
//-----------------
// defines pins numbers
const int trigPin = 3;
const int echoPin = 2;

void UltrasonicIntit(int trigPin, int echoPin) {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

int UltrasonicRead(int trigPin, int echoPin) {
  // defines variables
  long duration;
  int distance;

  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave t
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2;
  return distance;
}
```

Protocoale fizice - I2C







Start condition:
SDA goes low before SCL

'1' - Master is requesting data,
'0' - Master is sending data

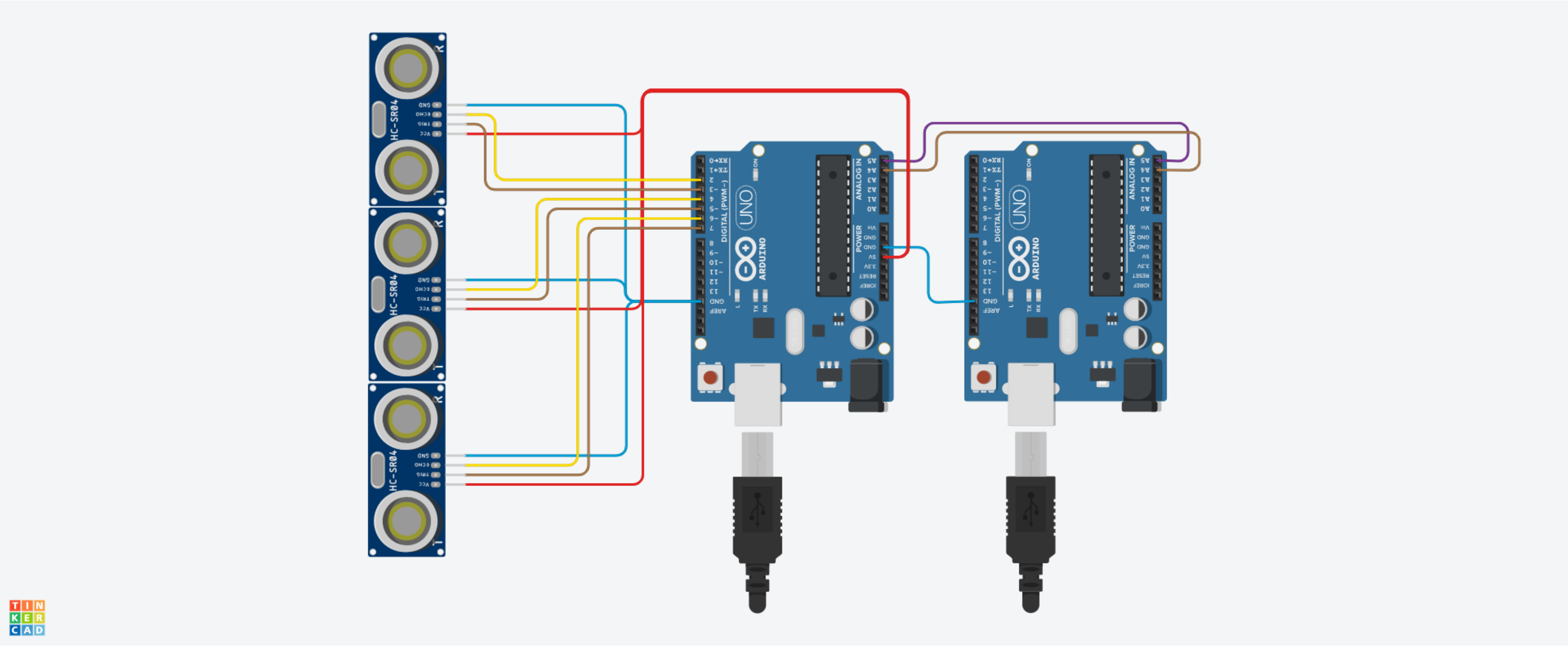
ACK/NACK: A '1' in this position indicates that the addressed slave did not respond or was unable to process the request.

Stop condition:
SDA goes high after SCL

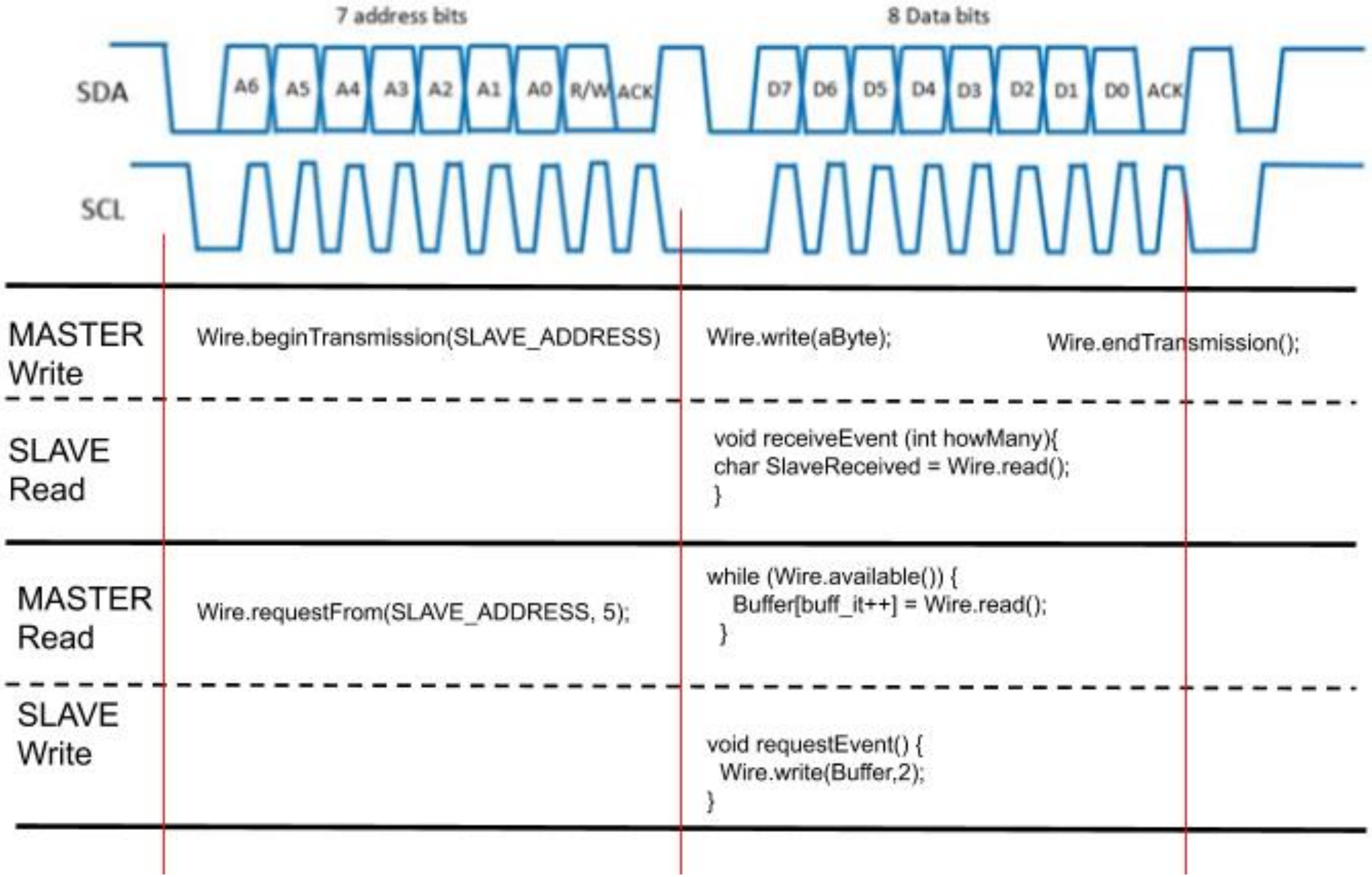
- https://www.slideshare.net/shudhanshu29/i2c-protocol-94259889
- https://www.slideshare.net/komalmehna/38-i2-c-protocol-spi-protocol

https://learn.sparkfun.com/tutorials/i2c/all

I2C – Digital Ultrasonic Sensor HCS-04



I2C -Protocol



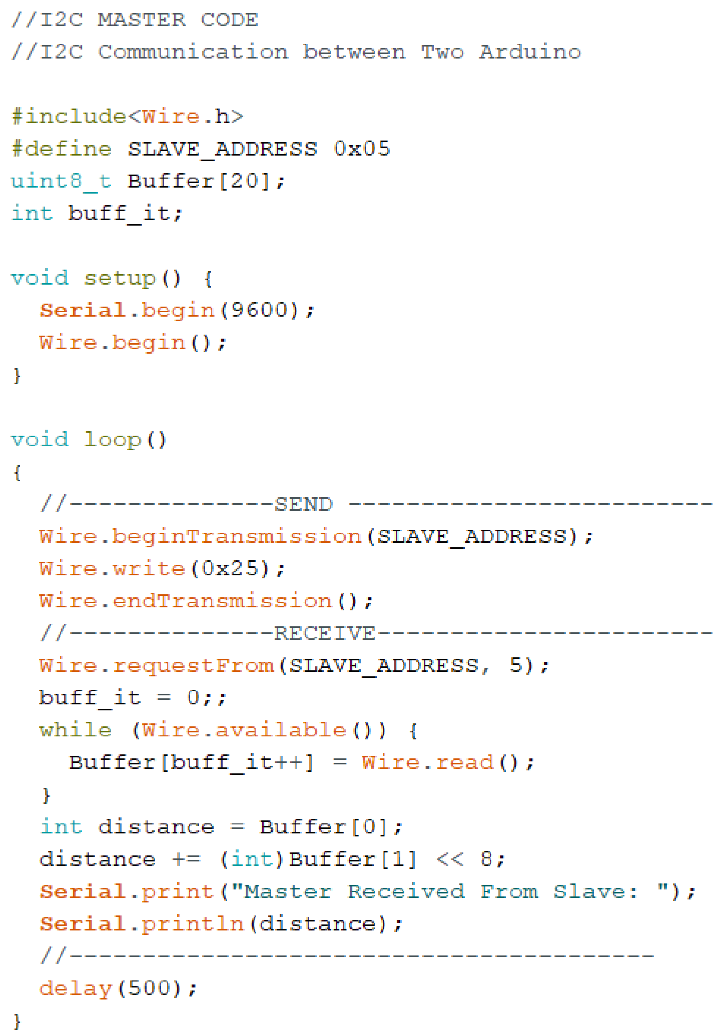
MASTER Write	Wire.beginTransmission(SLAVE_ADDRESS)	Wire.write(aByte);	Wire.endTransmission();

SLAVE Read		void receiveEvent (int howMany){
		char SlaveReceived = Wire.read();
		}	

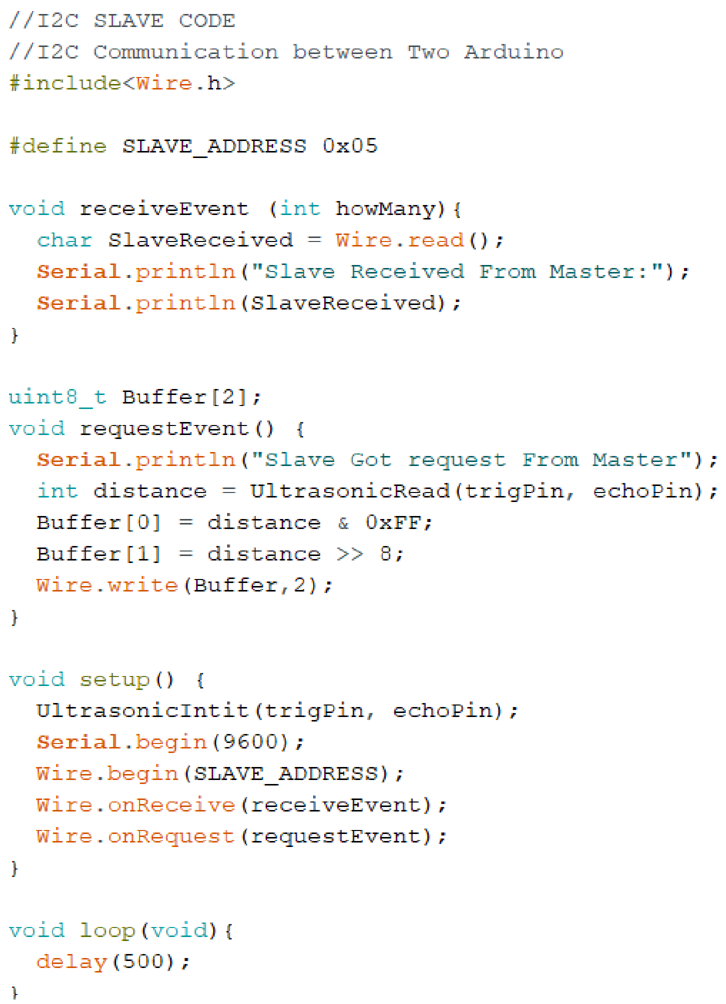
MASTER Read	Wire.requestFrom(SLAVE_ADDRESS, 5);	while (Wire.available()) {
		Buffer[buff_it++] = Wire.read();
		}	

SLAVE Write		void requestEvent() {
		Wire.write(Buffer,2);
		}	

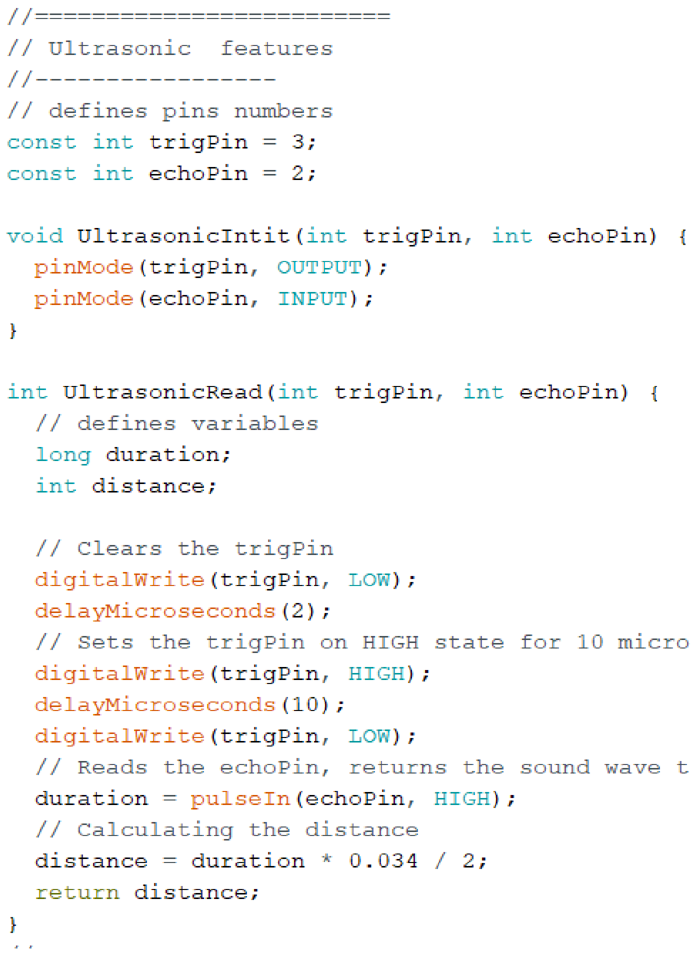
I2C – Digital Sensor Implementare



```
//I2C MASTER CODE
//I2C Communication between Two Arduino

#include<Wire.h>
#define SLAVE_ADDRESS 0x05
uint8_t Buffer[20];
int buff_it;

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

void loop()
{
  //--------------SEND -------------------------
  Wire.beginTransmission(SLAVE_ADDRESS);
  Wire.write(0x25);
  Wire.endTransmission();
  //--------------RECEIVE-----------------------
  Wire.requestFrom(SLAVE_ADDRESS, 5);
  buff_it = 0;;
  while (Wire.available()) {
    Buffer[buff_it++] = Wire.read();
  }
  int distance = Buffer[0];
  distance += (int)Buffer[1] << 8;
  Serial.print("Master Received From Slave: ");
  Serial.println(distance);
  //----------------------------------------
  delay(500);
}
```

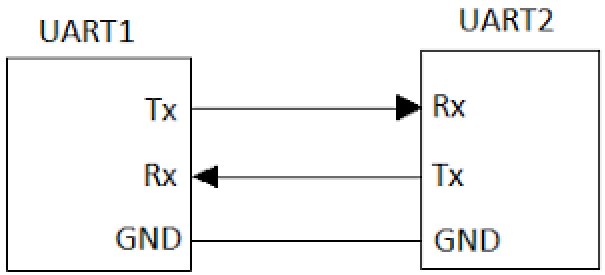


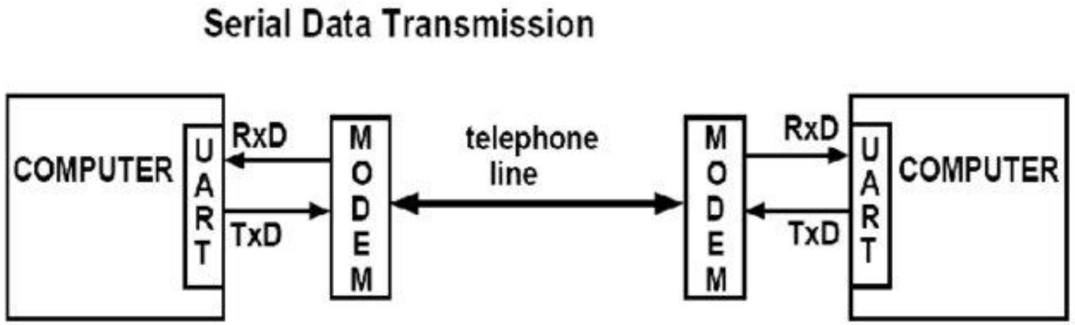
```
//I2C SLAVE CODE
//I2C Communication between Two Arduino
#include<Wire.h>

#define SLAVE_ADDRESS 0x05

void receiveEvent (int howMany){
  char SlaveReceived = Wire.read();
  Serial.println("Slave Received From Master:");
  Serial.println(SlaveReceived);
}

uint8_t Buffer[2];
void requestEvent() {
  Serial.println("Slave Got request From Master");
  int distance = UltrasonicRead(trigPin, echoPin);
  Buffer[0] = distance & 0xFF;
  Buffer[1] = distance >> 8;
  Wire.write(Buffer,2);
}

void setup() {
  UltrasonicIntit(trigPin, echoPin);
  Serial.begin(9600);
  Wire.begin(SLAVE_ADDRESS);
  Wire.onReceive(receiveEvent);
  Wire.onRequest(requestEvent);
}

void loop(void){
  delay(500);
}
```

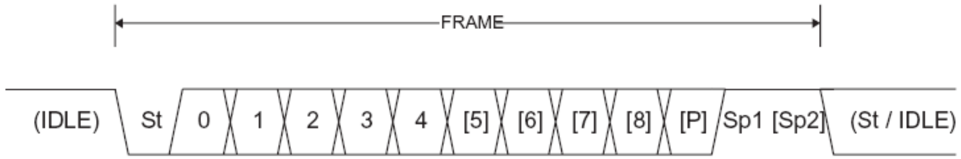


```
//========================
// Ultrasonic  features
//----------------
// defines pins numbers
const int trigPin = 3;
const int echoPin = 2;

void UltrasonicIntit(int trigPin, int echoPin) {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

int UltrasonicRead(int trigPin, int echoPin) {
  // defines variables
  long duration;
  int distance;

  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave t
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2;
  return distance;
}
```

Protocoale Fizice - USART

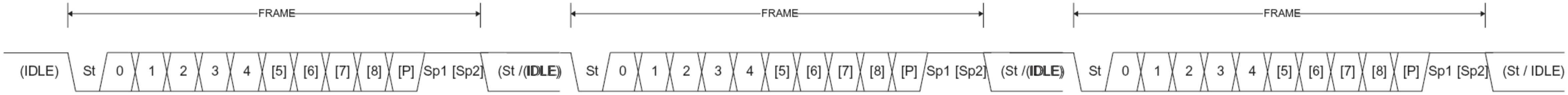




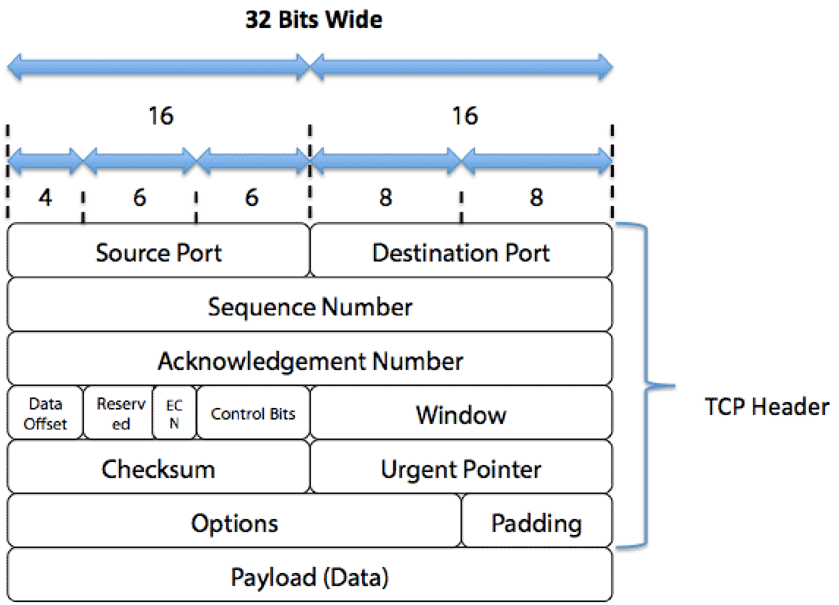


UART protocol

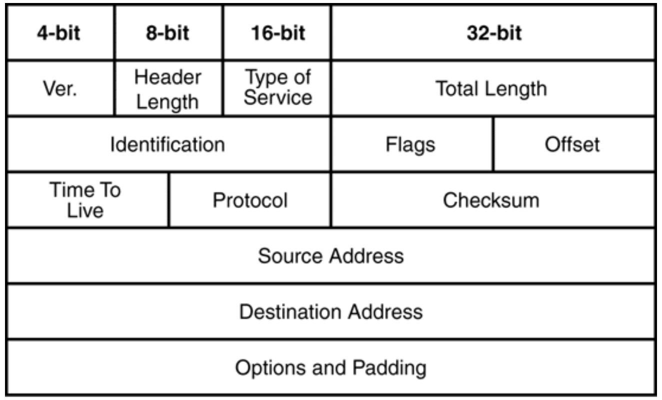
1. Idle – ”1”
2. Start bit – ”0”
3. Data – 5-9 bits
4. Parity
5. Stop bit – ”1” ; 1, 1.5, 2 bits



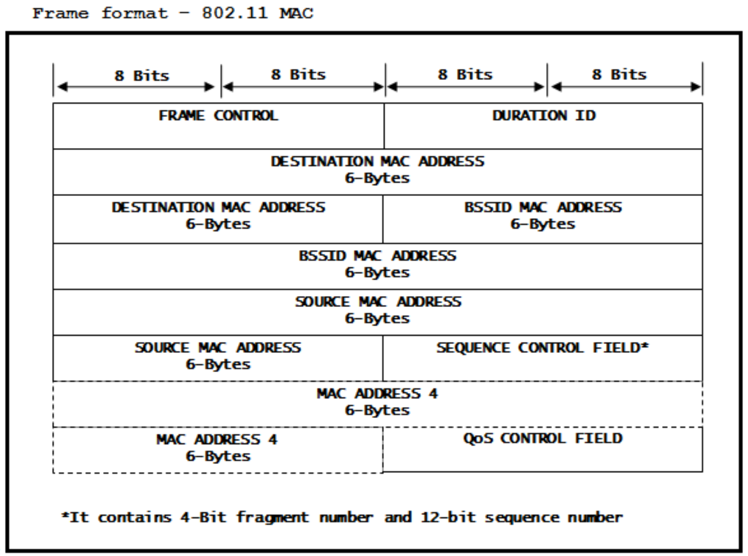
Protocoale Logice



Source Port			Destination Port	
Sequence Number				
Acknowledgement Number				
Data Offset	Reserved	ECN	Control Bits	Window
Checksum			Urgent Pointer	
Options				Padding
Payload (Data)				



4-bit	8-bit	16-bit	32-bit
Ver.	Header Length	Type of Service	Total Length
Identification		Flags	Offset
Time To Live	Protocol	Checksum	
Source Address			
Destination Address			
Options and Padding			



Frame format – 802.11 MAC

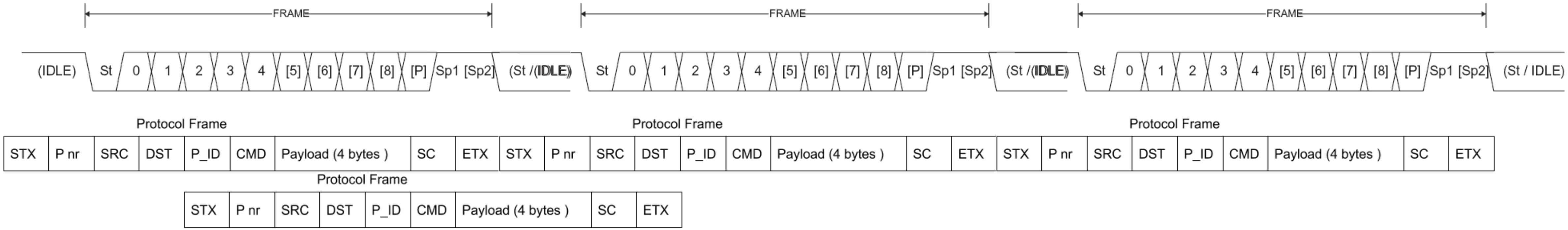
8 Bits	8 Bits	8 Bits	8 Bits
FRAME CONTROL		DURATION ID	
DESTINATION MAC ADDRESS 6-Bytes			
DESTINATION MAC ADDRESS 6-Bytes		BSSID MAC ADDRESS 6-Bytes	
BSSID MAC ADDRESS 6-Bytes			
SOURCE MAC ADDRESS 6-Bytes			
SOURCE MAC ADDRESS 6-Bytes		SEQUENCE CONTROL FIELD*	
MAC ADDRESS 4 6-Bytes			
MAC ADDRESS 4 6-Bytes		QoS CONTROL FIELD	

*It contains 4-Bit fragment number and 12-bit sequence number

https://jialinwu.com/post/ip-network-stack-writing-network-apps/

http://tefnutsecure.blogspot.com/2014/03/ip-address-ipv4-header.html

USART – Protocol Implementare



Protocol Frame

STX	P nr	SRC	DST	P_ID	CMD	Payload (4 bytes)	SC	ETX

Emitere

1. Selectie Date
2. Impachetare
3. Creare SC
4. Trimitere

Receptie

1. Colectare byte
2. Buferizare
3. Verificare
4. Interpretare Date

Stx – 0x02

Etx - 0x03

Pnr – contorizare pachete

SRC – emitatorul

DST – receptorul

P_id – tipul pachetului

CMD – comada

Payload – date pachet

SC – suma de control