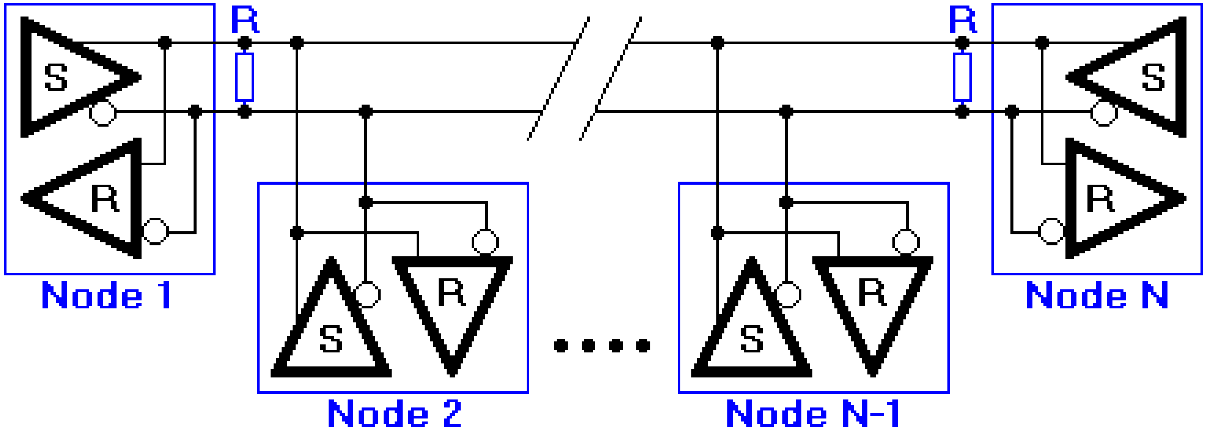
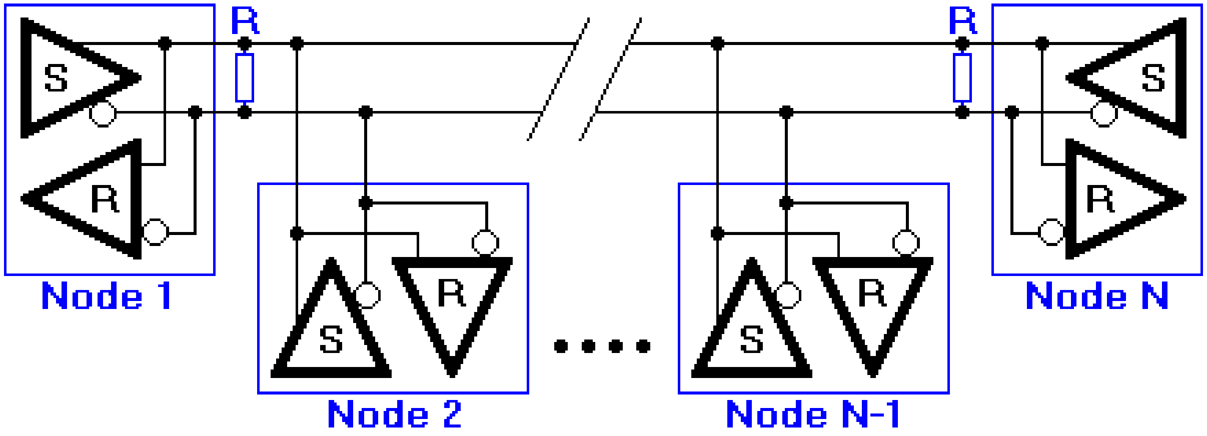
RS485

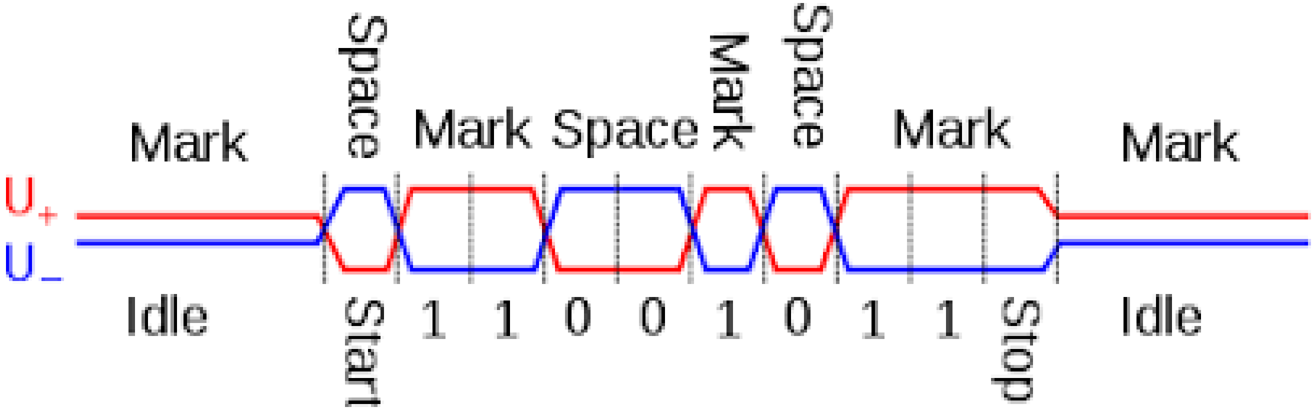
- Serial Communication
- Nr of Nodes >= 2
- No automatic access mechanism’s
  - Easy to get collision between messages
- A node read what it is transmitting
  - Meaning TX and RX channels 100% separated logic wise



RS485-II

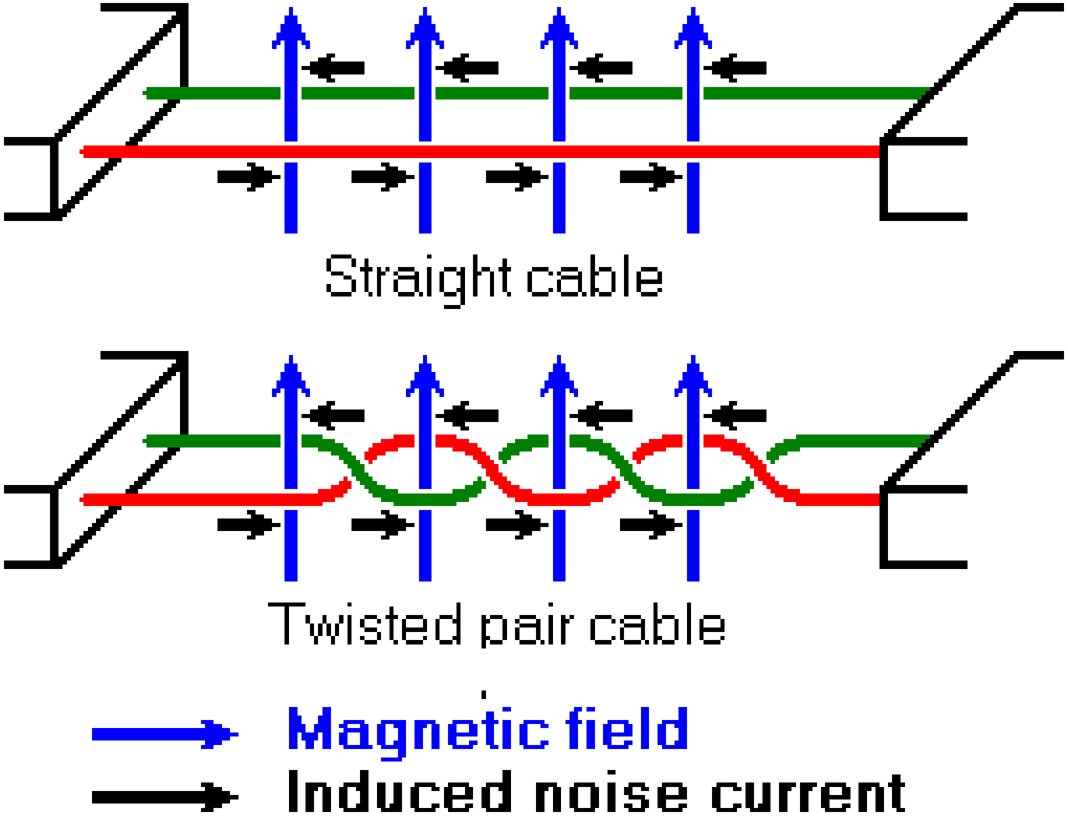
- TX is a “double signal”
- Receiver is differential so receive a “0” or “1” is not an absolute voltage
- But rather “positive” or negative”





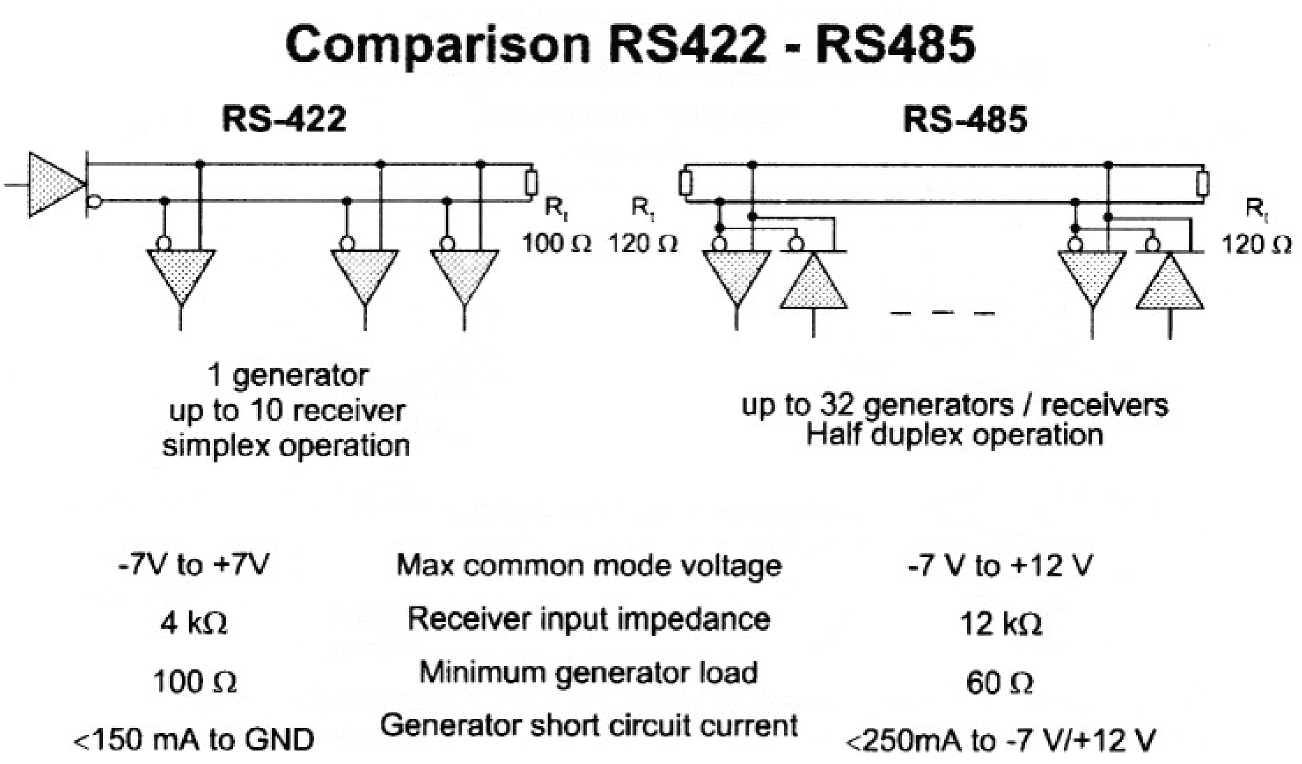
Noise immunity

- Twisted cables and ...
- Differential signals
- “makes” cables longer :-)



How long ? How many ?

- RS422 is a differential variant of rs232
- 120 ohm terminating resistors is tp avoid standing waves on the cables
- No of nodes is restricted by “fanout” - how much a TX nodes can “produce” og amperes on the cable
- Length normally < 1200m (rs232 < 15m)



Comparison RS422 - RS485

1 generator
up to 10 receiver
simplex operation

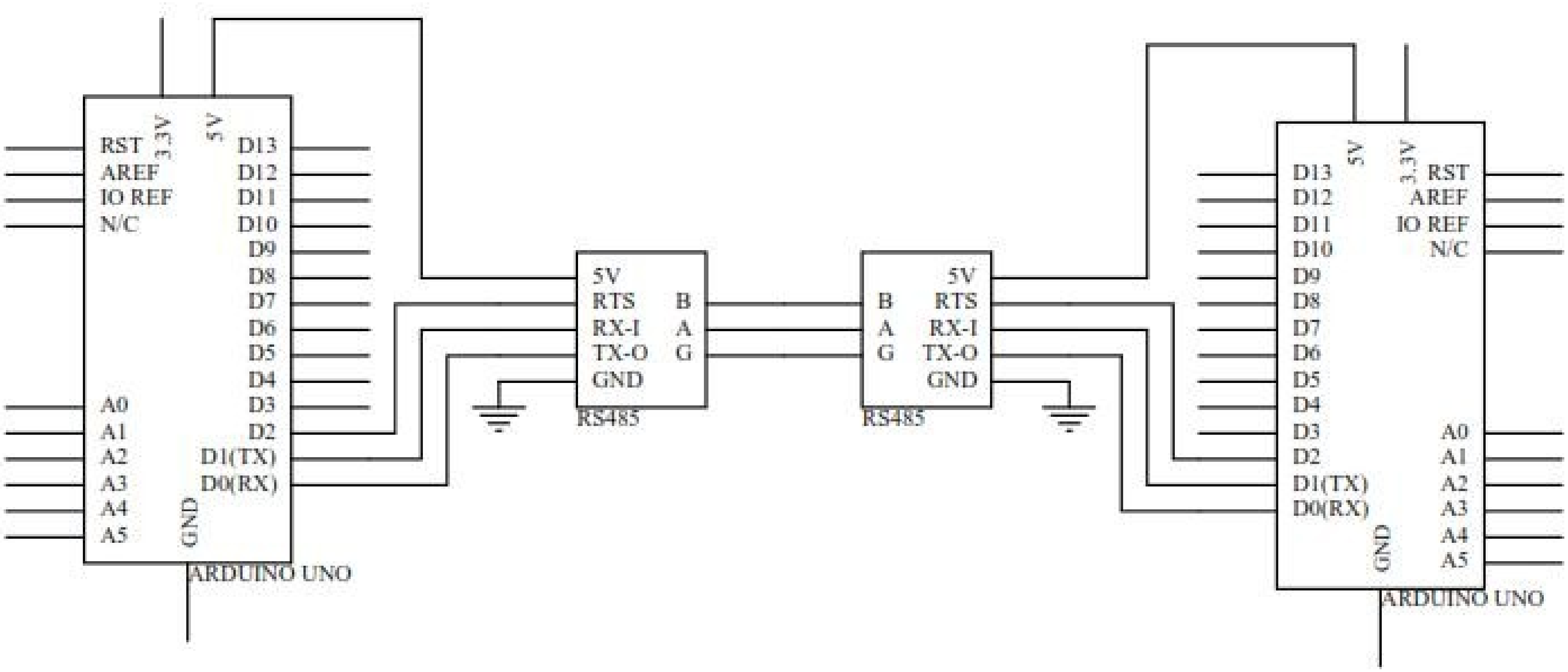
up to 32 generators / receivers
Half duplex operation

-7V to +7V	Max common mode voltage	-7 V to +12 V
4 kΩ	Receiver input impedance	12 kΩ
100 Ω	Minimum generator load	60 Ω
<150 mA to GND	Generator short circuit current	<250mA to -7 V/+12 V

How ?

- Default in RX-only mode
- Activate TX circuit by “RTS”   Request To Send

- NB: Arduinos TX with 0-5V signals
- The rs485 interface convert it to symmetric



Access to rs485 network – Ostrich protocol

- Ostrich method
  - Go for it just …
  - Activate your rs485 tx
  - Send your bits and bytes
  - Deactivate your rs485 tx

- You might be lucky
- Or have message collision == you loose your data

- Open question(s)
  - How do you detect message loss ?
  - What will you do upon detection ?

Media Access Policy

- RS485 has no builtin (hardware) mechanism
- You will do design your own access protocol
- ...

Media Access Policy

- RS485 has no builtin (hardware) mechanism

- You will do design your own access protocol

- The next simples (Ostrich is the simplest)

- Master slave(s) policy

- Only master can initiate communication
- Its convenient that all slave nodes has a known identity
- Slaves will reply upon request from master

- Master can initiate
  - Tx data to slave(s)
  - Request to slave to TX data back to master

- In basic version it’s a little uphill (performancewise) to move data
  - From one slave to another slave
  - -

Protocols – here the problems starts

1) Master asks slave A to send data to master
2) Master sends slave As data to slave B which need it

Potential problems

3) A slave has some important info that it “need” to deliver now
   1) What is now ?
   2) The slave has to wait on the master requesting for info
   3) When will thois happen ?
   4) Its up to the master

So you have to design a protocol which

- Takes care of timing in your system
- On regular(?!) intervals ask relevant slaves for “new news”

Regular cyclic executive systems are well suited because

- You can design a master driven cyclic service scheme for the slaves

Dynamixel

- Two protocols:
  - Protocol 1
  - Protocol 2 (love the names)

▲
TOP

1. Introduction

- Protocol 2.0 supported devices: MX-28, MX-64, MX-106(MX Series with Firmware V39 or above), X Series, DYNAMIXEL Pro
- Protocol 2.0 supported controllers: CM-150, CM-200, OpenCM9.04, OpenCR
- Other: 2.0 protocol from R+ Smart app

2. Instruction Packet

Instruction Packet is the command data sent to the Device.

Header1	Header2	Header3	Reserved	Packet ID	Length1	Length2	Instruction	Param	Param	Param	CRC1	CRC2
0xFF	0xFF	0xFD	0x00	ID	Len_L	Len_H	Instruction	Param 1	...	Param N	CRC_L	CRC_H

2. 1. Header

The field indicates the start of the Packet

2. 2. Reserved

0x00 (0xFD cannot be used)

2. 3. Packet ID

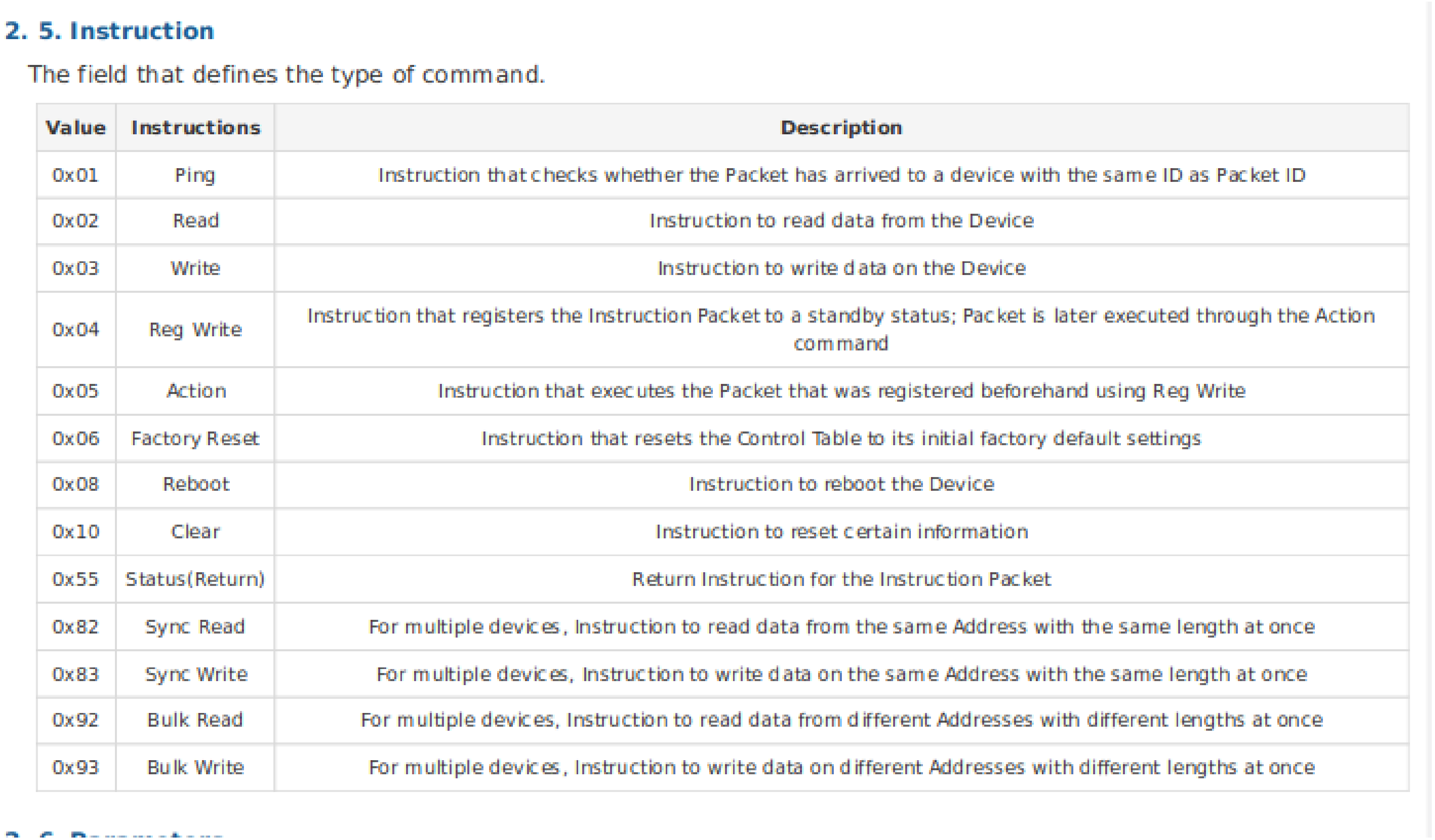
The field that indicates the ID of the Device that should receive the Instruction Packet and process it

1. Range : 0 ~ 252 (0x00 ~ 0xFC), which is a total of 253 numbers that can be used
2. Broadcast ID : 254 (0xFE), which makes all connected devices execute the Instruction Packet
3. 253(0xFD), 255(0xFF) : These are not used in order to avoid duplicate use with Header

2. 4. Packet Length

The length after the Packet Length field (Instruction, Parameter, CRC fields). Packet Length = number of Parameters + 3

Instructions field



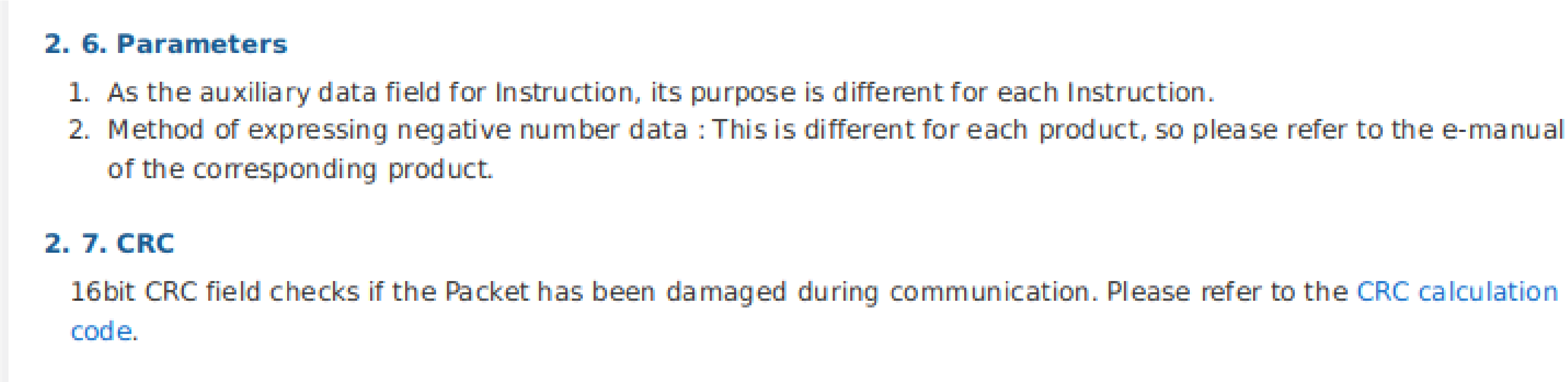
2. 5. Instruction

The field that defines the type of command.

Value	Instructions	Description
0x01	Ping	Instruction that checks whether the Packet has arrived to a device with the same ID as Packet ID
0x02	Read	Instruction to read data from the Device
0x03	Write	Instruction to write data on the Device
0x04	Reg Write	Instruction that registers the Instruction Packet to a standby status; Packet is later executed through the Action command
0x05	Action	Instruction that executes the Packet that was registered beforehand using Reg Write
0x06	Factory Reset	Instruction that resets the Control Table to its initial factory default settings
0x08	Reboot	Instruction to reboot the Device
0x10	Clear	Instruction to reset certain information
0x55	Status(Return)	Return Instruction for the Instruction Packet
0x82	Sync Read	For multiple devices, Instruction to read data from the same Address with the same length at once
0x83	Sync Write	For multiple devices, Instruction to write data on the same Address with the same length at once
0x92	Bulk Read	For multiple devices, Instruction to read data from different Addresses with different lengths at once
0x93	Bulk Write	For multiple devices, Instruction to write data on different Addresses with different lengths at once

Add parameters and CRC-16 check

- Th CRC-16 field is to ensure that the intended receiver receives an ok pkg
- There is a broadcast address (0xfe) so you can contact all
  - For reset etc



2. 6. Parameters

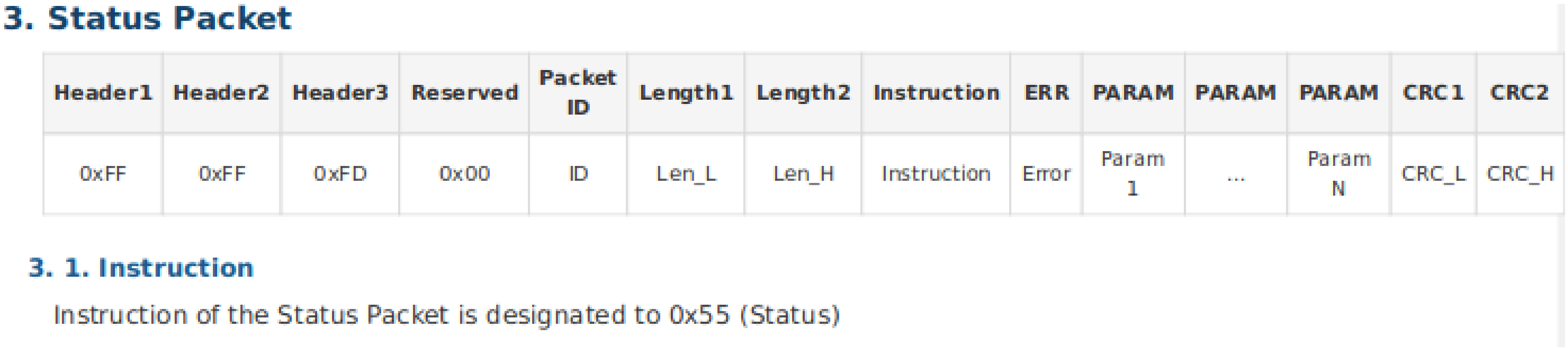
1. As the auxiliary data field for Instruction, its purpose is different for each Instruction.
2. Method of expressing negative number data : This is different for each product, so please refer to the e-manual of the corresponding product.

2. 7. CRC

16bit CRC field checks if the Packet has been damaged during communication. Please refer to the CRC calculation code.

Example: status Pkg

- All Dynamixels motors has some command id for status: 0x55



3. Status Packet

Header1	Header2	Header3	Reserved	Packet ID	Length1	Length2	Instruction	ERR	PARAM	PARAM	PARAM	CRC1	CRC2
0xFF	0xFF	0xFD	0x00	ID	Len_L	Len_H	Instruction	Error	Param 1	...	Param N	CRC_L	CRC_H

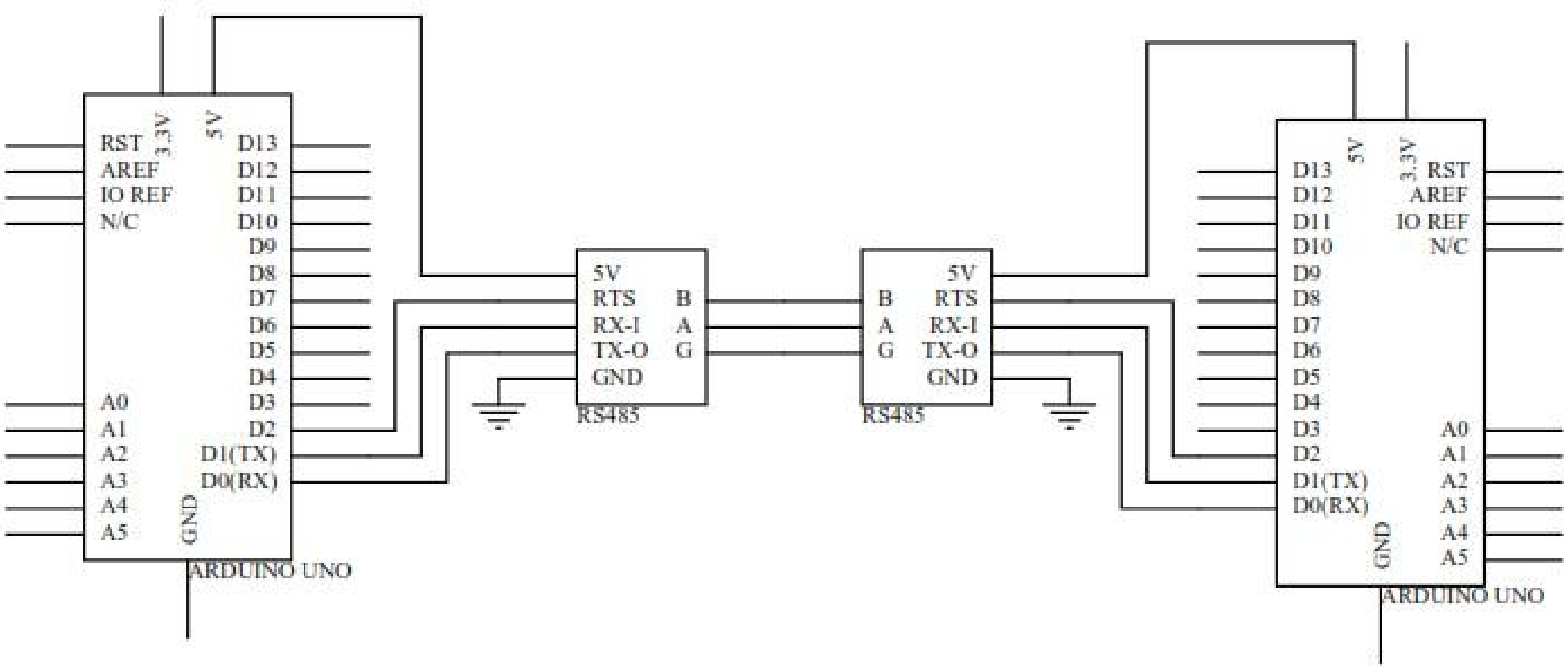
3. 1. Instruction

Instruction of the Status Packet is designated to 0x55 (Status)

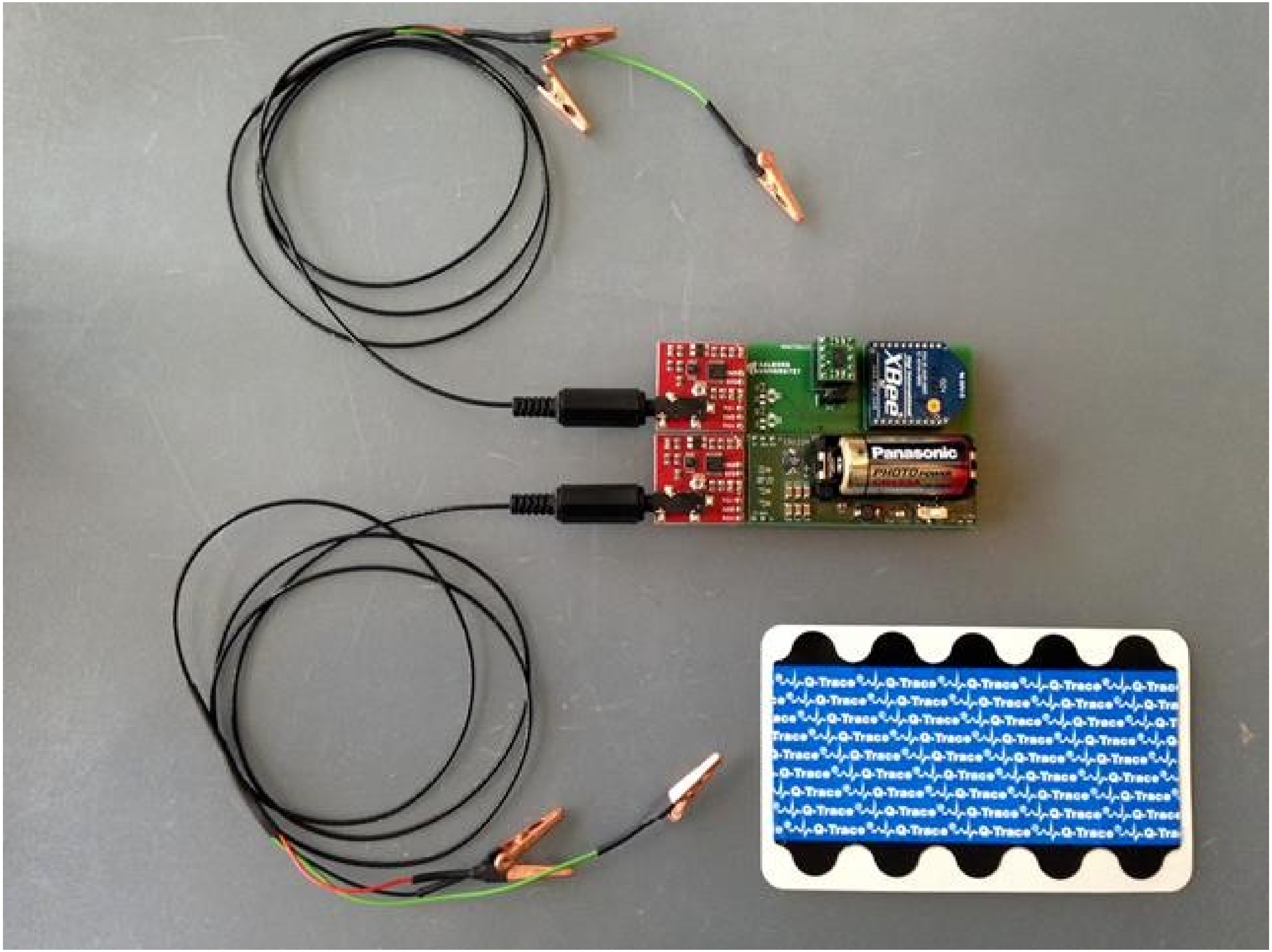
Your setup

- An arduino or eq
- A rs485 net
- Mx motors on the rs485 net

- So tx/rx code as normal but with surrounding “activate/deactivate interface”

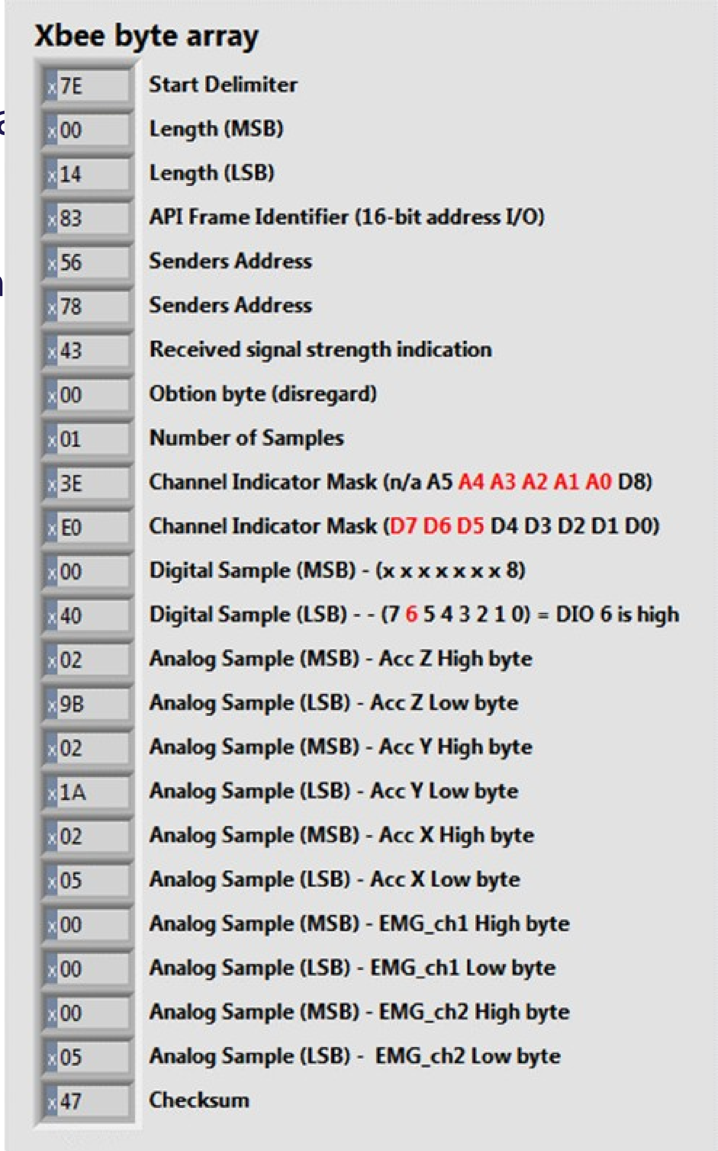


XBEE



One way communication

- Measuring devices sends messages on regula
- Only one type of message
  - 2 analog EMG 16 bit int measurements
  - 3 analog accelerometer x,y,z 16 bit int measurem
  - A simple checksum



Xbee byte array

x7E	Start Delimiter
x00	Length (MSB)
x14	Length (LSB)
x83	API Frame Identifier (16-bit address I/O)
x56	Senders Address
x78	Senders Address
x43	Received signal strength indication
x00	Obtion byte (disregard)
x01	Number of Samples
x3E	Channel Indicator Mask (n/a A5 A4 A3 A2 A1 A0 D8)
xE0	Channel Indicator Mask (D7 D6 D5 D4 D3 D2 D1 D0)
x00	Digital Sample (MSB) - (x x x x x x x 8)
x40	Digital Sample (LSB) - - (7 6 5 4 3 2 1 0) = DIO 6 is high
x02	Analog Sample (MSB) - Acc Z High byte
x9B	Analog Sample (LSB) - Acc Z Low byte
x02	Analog Sample (MSB) - Acc Y High byte
x1A	Analog Sample (LSB) - Acc Y Low byte
x02	Analog Sample (MSB) - Acc X High byte
x05	Analog Sample (LSB) - Acc X Low byte
x00	Analog Sample (MSB) - EMG_ch1 High byte
x00	Analog Sample (LSB) - EMG_ch1 Low byte
x00	Analog Sample (MSB) - EMG_ch2 High byte
x05	Analog Sample (LSB) - EMG_ch2 Low byte
x47	Checksum

- http://kom.aau.dk/~jdn/edu/courses/19-2/sensact/xbee.html

Short about time – cyclic executive

- Given ...
  - No kernel or operating system
  - Nothing to do except for the regulary executive

```
unsigned long tt=1000;  // 100 msec executive
unsigned long t1;

void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
  t1=millis(); // init
}

boolean isItTime()
{
  unsigned long t;
  t = millis();

  if (tt <= t- t1) {
    t1 = t;
    return true;
  }
  else { return false; }
}

void doYourCode()
{
Serial.println(t1);
delay (random(200,600));
}
void loop() {
  while (! isItTime()) ;
  doYourCode();
}
```

```
unsigned long tt=1000;
unsigned long t1;

void setup() {
Serial.begin(9600);
  t1=millis(); // init
}

boolean isItTime()
{
  unsigned long t;
  t = millis();

  if (tt <= t- t1) {
    t1 = t;
    return true;
  }
  else {
    return false;
 }
}
```

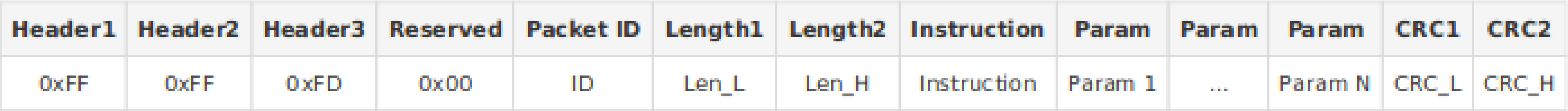
```
void doYourCode()
{
 Serial.println(t1);
 delay (random(200,600));
}

void loop() {
  while (! isItTime()) ;
  doYourCode();
}
```

Who is who ?

- You are the master
- You shall talk with your nodes/slaves

Time and speed

- Your RS485 is signalwise driven by your Adruino/Teensy/...
- So you have given baudrate (1 byte ~= 10 bits)
  - 9600 baud ~= 900 bytes/second
- A protocol pkg is a least 12 bytes and more often 20 bytes
- So @9600 you can TX at most 450 pkg/sec
- An if you need a reply 200 call-reply pkgs – at most !!!
- Welcome to a new world



Header1	Header2	Header3	Reserved	Packet ID	Length1	Length2	Instruction	Param	Param	Param	CRC1	CRC2
0xFF	0xFF	0xFD	0x00	ID	Len_L	Len_H	Instruction	Param 1	...	Param N	CRC_L	CRC_H

More complicated code

- You might want to have more than one service running
  - A service can be your controller ...
    1) Getting data from your motors (angles, momentuum,...) I assume you are the master
    2) Calculating controller
    3) Tx setpoints back to the motors
    4) Wait until next sample time

- Solutions might be...

- “complicated” intereupt system
- A realtime operating system or kernel

- Not the scope 2day

- 2day – get into communication with your robot with your own code

EOL