Serial Interfaces: UART, I2C, SPI

Project-based Learning Center | ETH Zurich

Tommaso Polonelli tommaso.polonelli@pbl.ee.ethz.ch

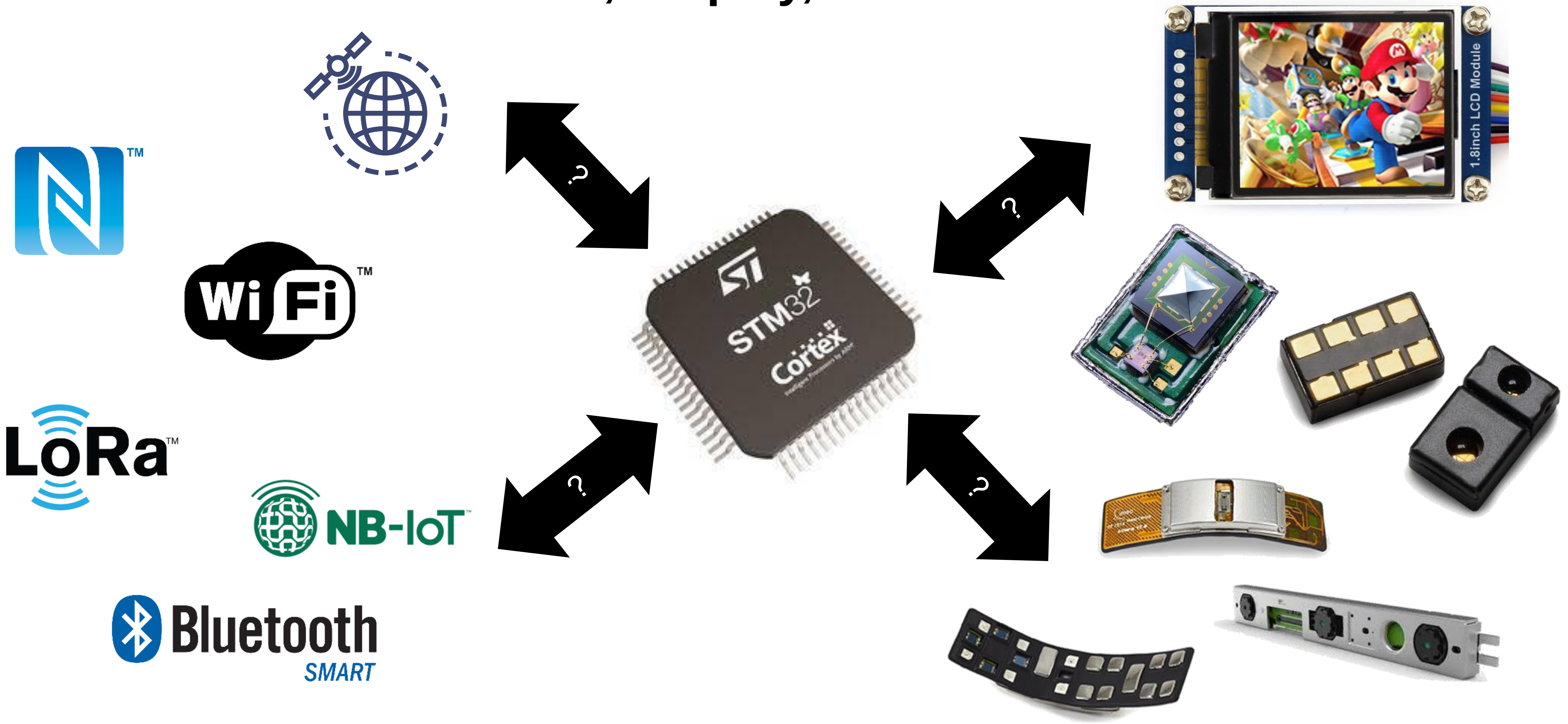
Michele Magno michele.magno@pbl.ee.ethz.ch

Vlad Niculescu vladn@iis.ee.ethz.ch

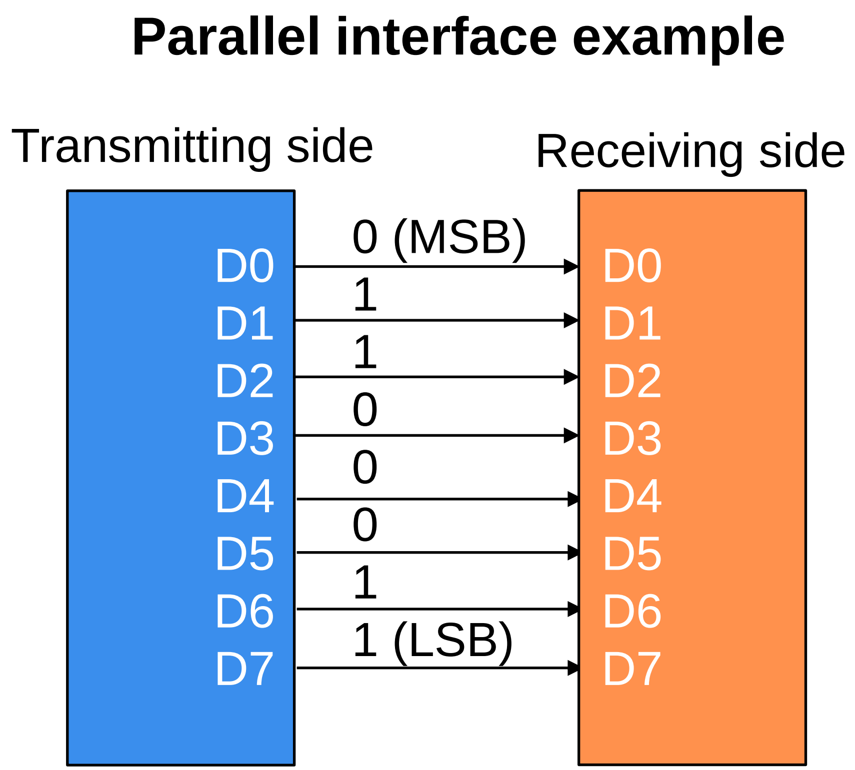
Program and structure of the course

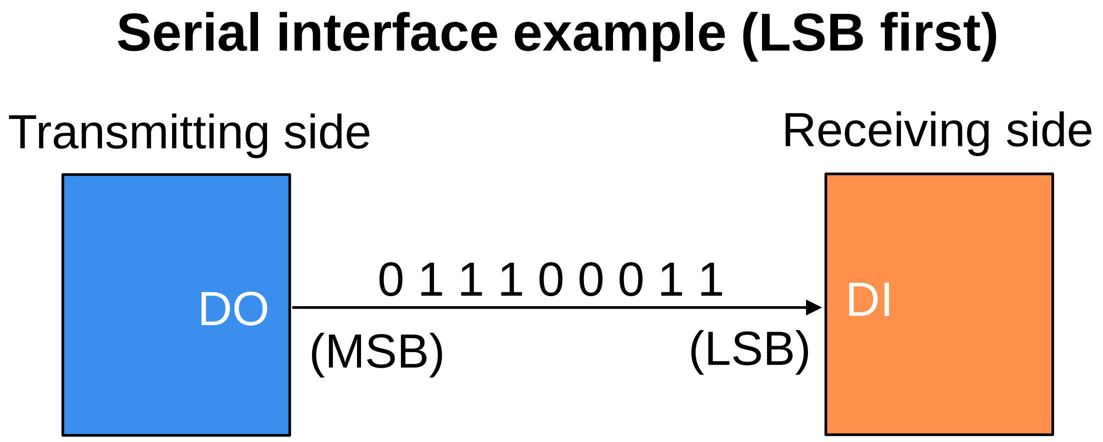
DATE	Topic	
04.03.2021	Introduction to the course • Microcontrollers in general and STM32 • Integrated Development Environment – STM32CubeIDE • LAB1: How to program an STM32	TP
11.03.2021	PWM and motors	TP
18.03.2021	Serial interfaces UART + SPI + I2C + IMU	TP
25.03.2021	EKF and AHRS	VN
01.04.2021	Bluetooth Low Energy	TP
15.04.2021	Control PID, attitude control	VN
22.04.2021	**Project presentation (Guest)** **Project selection and kick start**	TP/Guest
29.04.2021	Flight test (drone test room) **Exercise summary**	TP/VN
	Project	TP
	Project	TP
	Project	TP
	Project	TP
03/06/2021	**Final presentation**	MM/TP

Serial Interfaces: Sensors, Display, Radio etc.

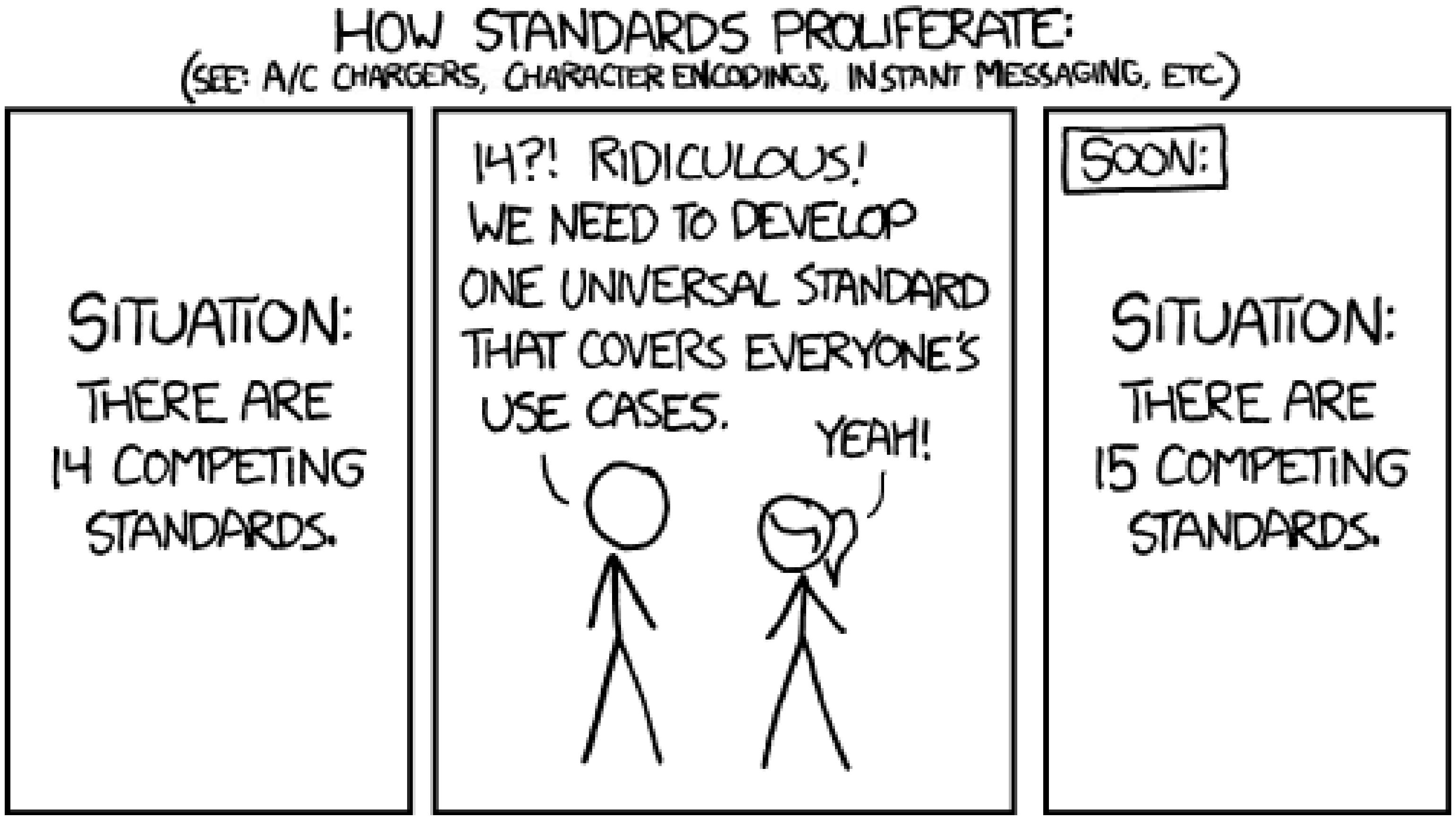


Comparison between Parallel and Serial Communication





Serial Interface Standards



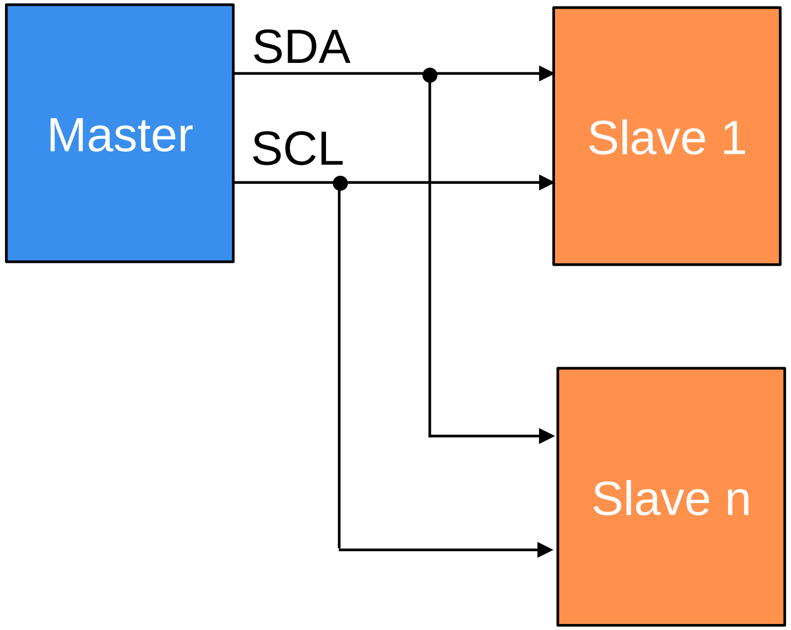
ETH zürich

I^{2}C – Inter-Integrated Circuit Bus

I^2C: Inter-Integrated Circuit Bus - 1

- Usually pronounced “I-Squared-C”
- Introduced by Philips (now NXP Semiconductors) in 1982
- Used for communication with external peripherals, for example:
  - EEPROMs
  - thermal sensors
  - real-time clocks
- Also used as a control interface for signal processing devices with separate data interfaces, for example:
  - radio frequency tuners
  - video decoders and encoders
  - audio processors

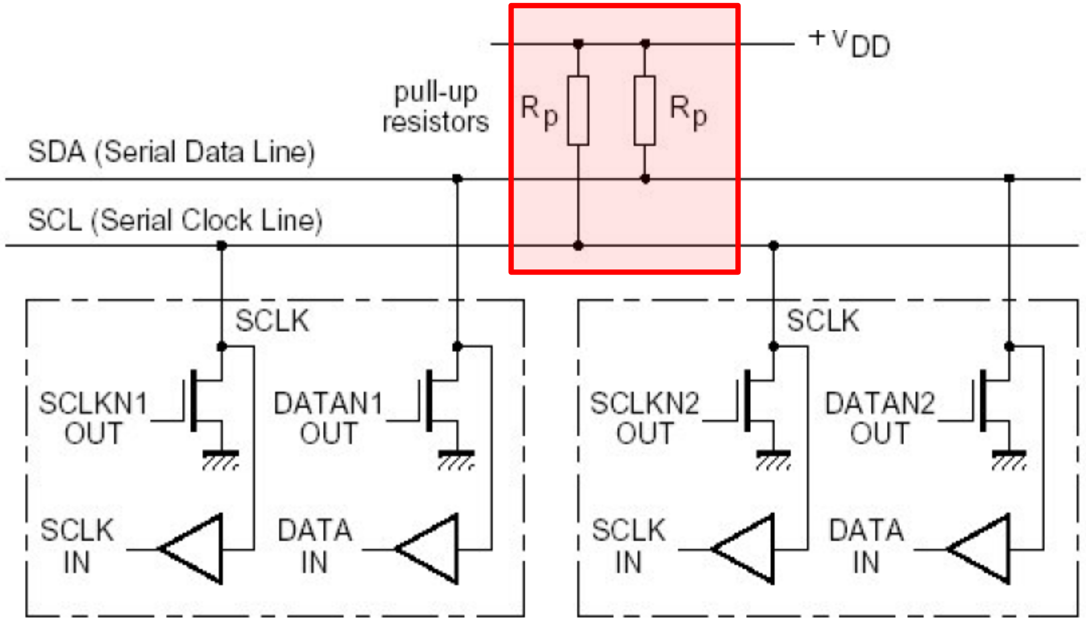
I²C: Inter-Integrated Circuit Bus - 2



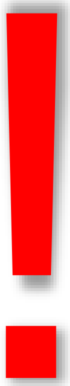
- Three supported speed modes:
  - slow (under 100 Kbps)
  - fast (400 Kbps)
  - high-speed (3.4 Mbps) – in I2C v.2.0
- Maximum inter-IC distance of about 3 meters
  - (for moderate speeds, less for high-speed)
- Can support multi-master mode
  - For complex applications
  - Communication is always started by a master, both in single-master and multi-master mode
- Half-duplex synchronous communication scheme
  - the master of the communication generates the clock (SCL) on which data (SDA) is synchronized

I²C: Inter-Integrated Circuit Bus - 3

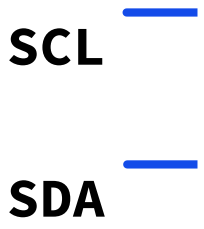
- Based on two lines:
  - SCL (serial clock)
  - SDA (serial data)



- **Pull-Up** resistors, **Pull-Down** by open-drain drivers
  - Wired-AND: if *any* driver pulls down, the line is low (avoids short circuits)
  - Any module on the bus can act as **master**, **slave** or both
    - typical case: MCU is the master, peripherals/sensors are slaves

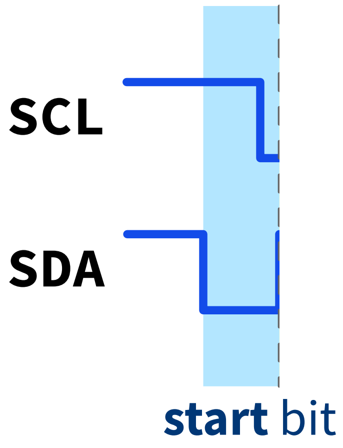


I²C: Interface Protocol



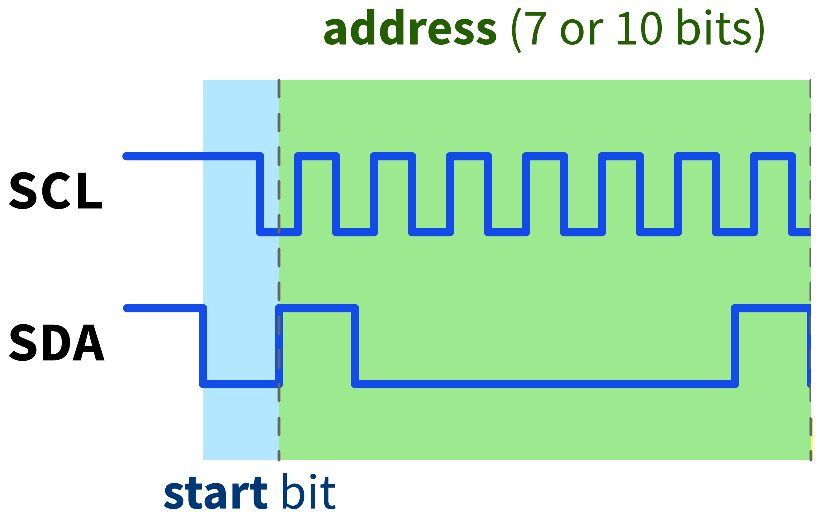
- In idle, both **SCL** and **SDA** are pulled-up to **1**

I²C: Interface Protocol



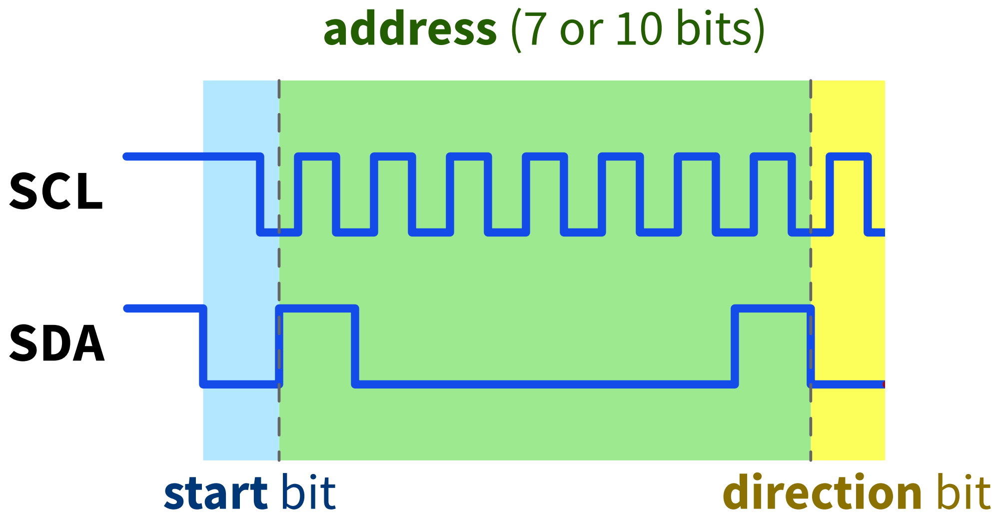
1. To start the communication, the **master**:
   - asserts the **start** bit (**SDA** 1→0 transition while **SCL** is **still 1**)
   - then, it starts generating the **SCL** clock
   - except for the start and stop bits, **SDA** transitions *only* when **SCL** is **0**

I²C: Interface Protocol



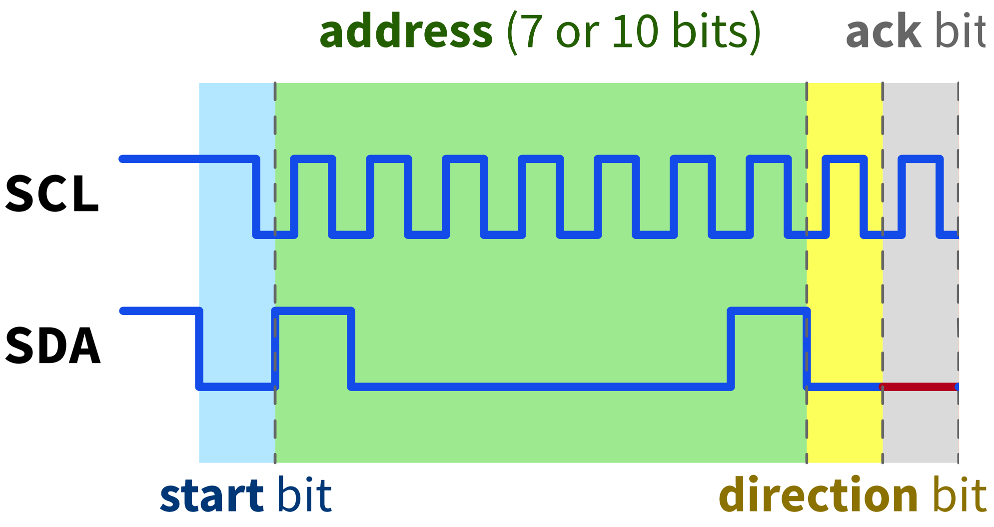
2. The **master** transmits the **slave** **address**:
   - **broadcasted** to all devices on the I²C bus
   - used to select the target **slave**
   - either **7 bits** or **10 bits** (newer devices – 7 bits address space is small!)
   - *in the example, the address is 7'b1000001*

I²C: Interface Protocol



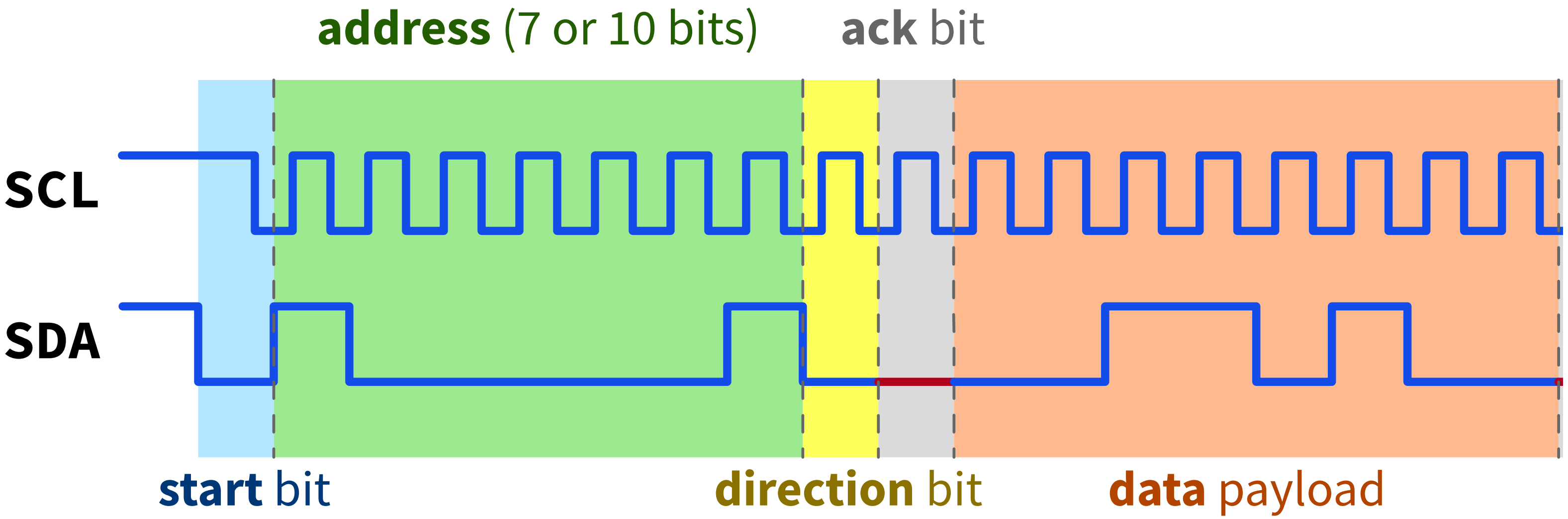
3. The **master** transmits a **direction** bit:
   - a **0** for **master** → **slave** (write) transfer
   - a **1** for **slave** → **master** (read) transfer
   - *in the example, suppose a write transfer*

I²C: Interface Protocol



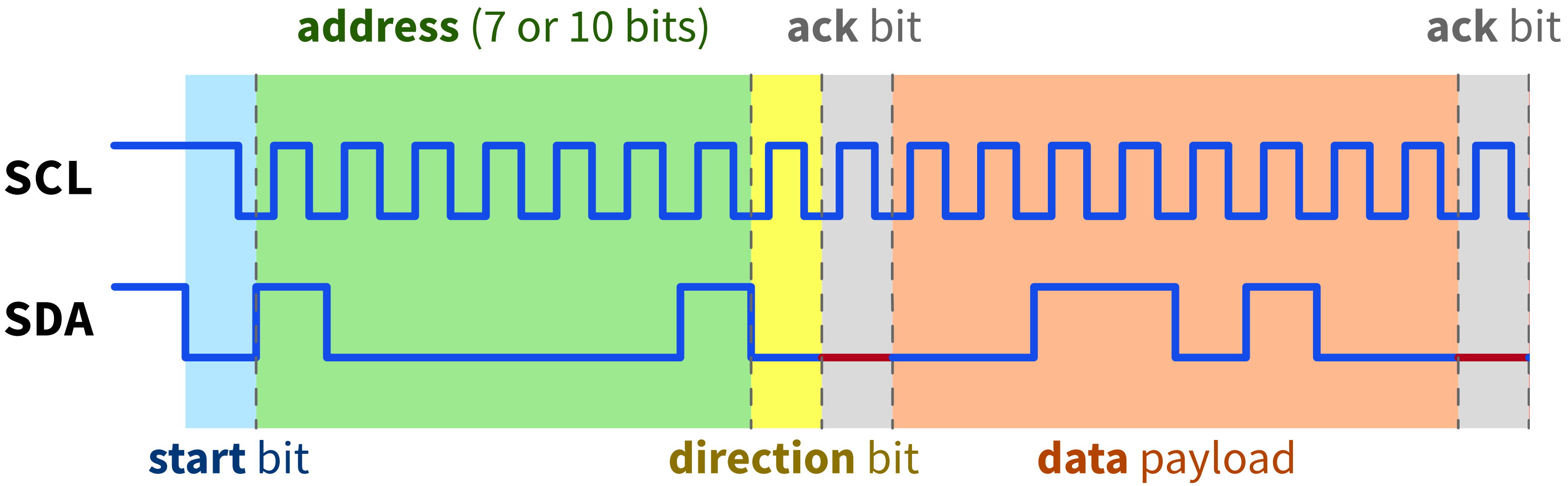
4. The **slave** then **acknowledges** reception:
   - by driving **SDA** to **0**
   - if not acknowledged, the transaction must be repeated by the master

I²C: Interface Protocol



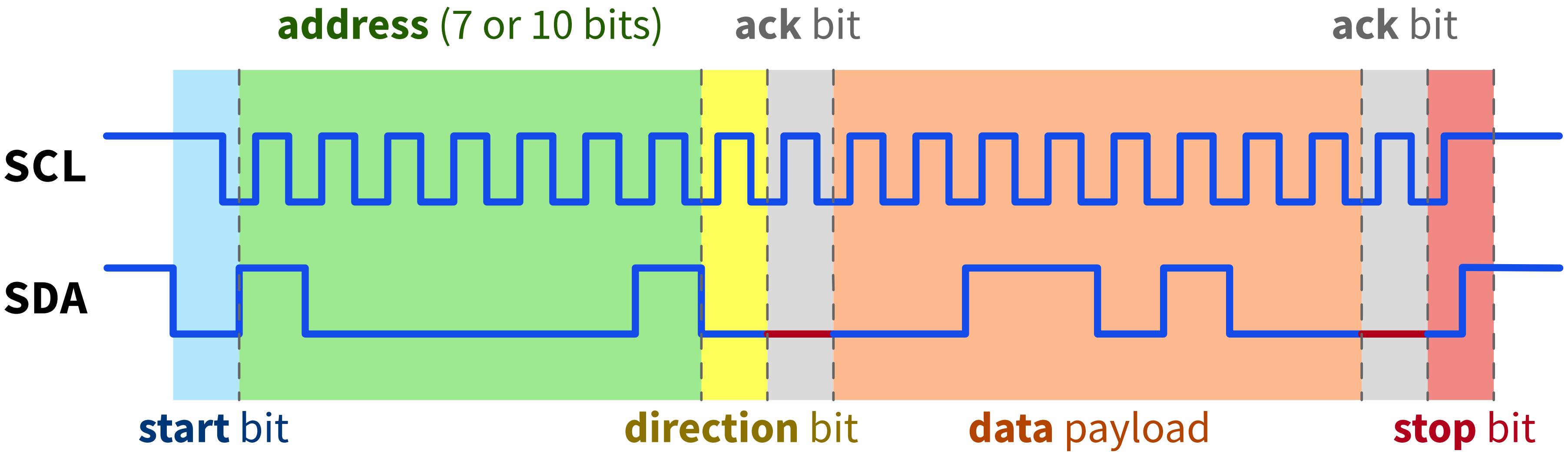
5. The **master** transmits its data payload:
   - each payload packet is **8 bits**
   - there might be more than one packet, depending on application
   - *in the example, data payload is 8'b00110100*

I²C: Interface Protocol



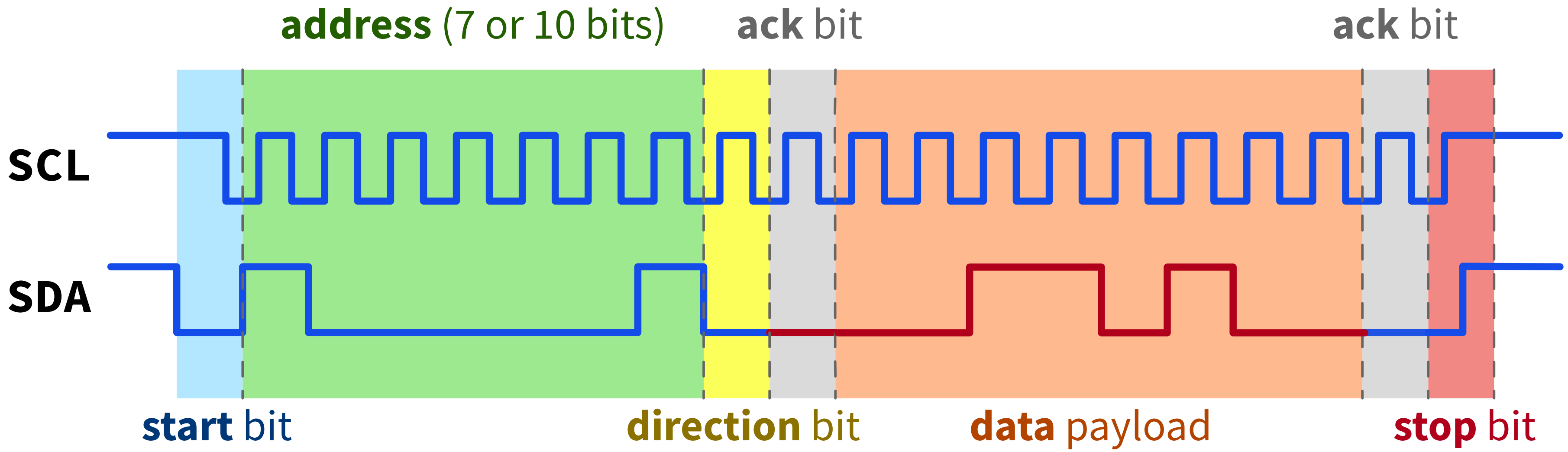
6. The **slave** acknowledges reception of the data packet:
   - **1 ack bit** every **8 payload bits**
   - slave must acknowledge each packet

I²C: Interface Protocol



7. At the end of the transfer, the **master** transmits a **stop bit**:
   - first, it sets **SDA** to **0**
   - then it releases **SCL** (i.e. it lets it go to **1**)
   - finally, it releases **SDA** which also goes to **1**

I²C: Interface Protocol

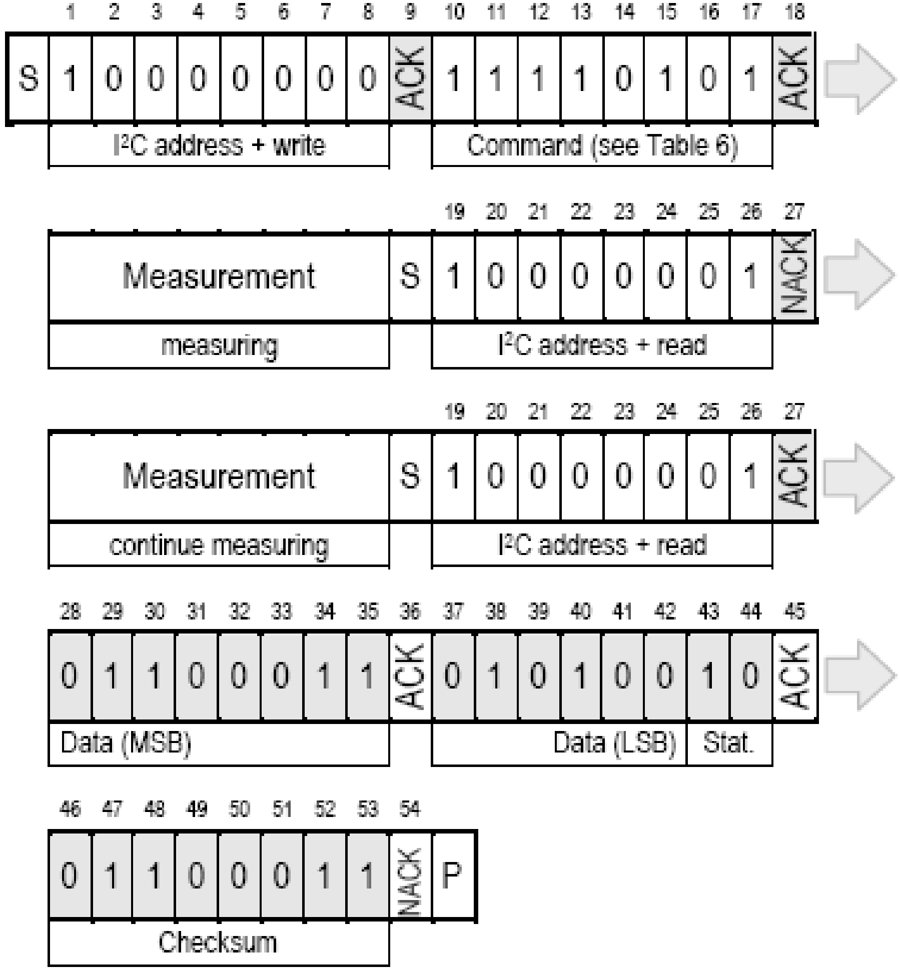


BUS sequence:

1. Master sends the start bit
2. Master sends the slave address (7 bits)
3. Master sends the write/read bit
4. Slave sends the ACK
5. Master sends the payload (8-bits)
6. Slave sends the ACK
7. Master sends the stop bit

I^2C: Interface Protocol

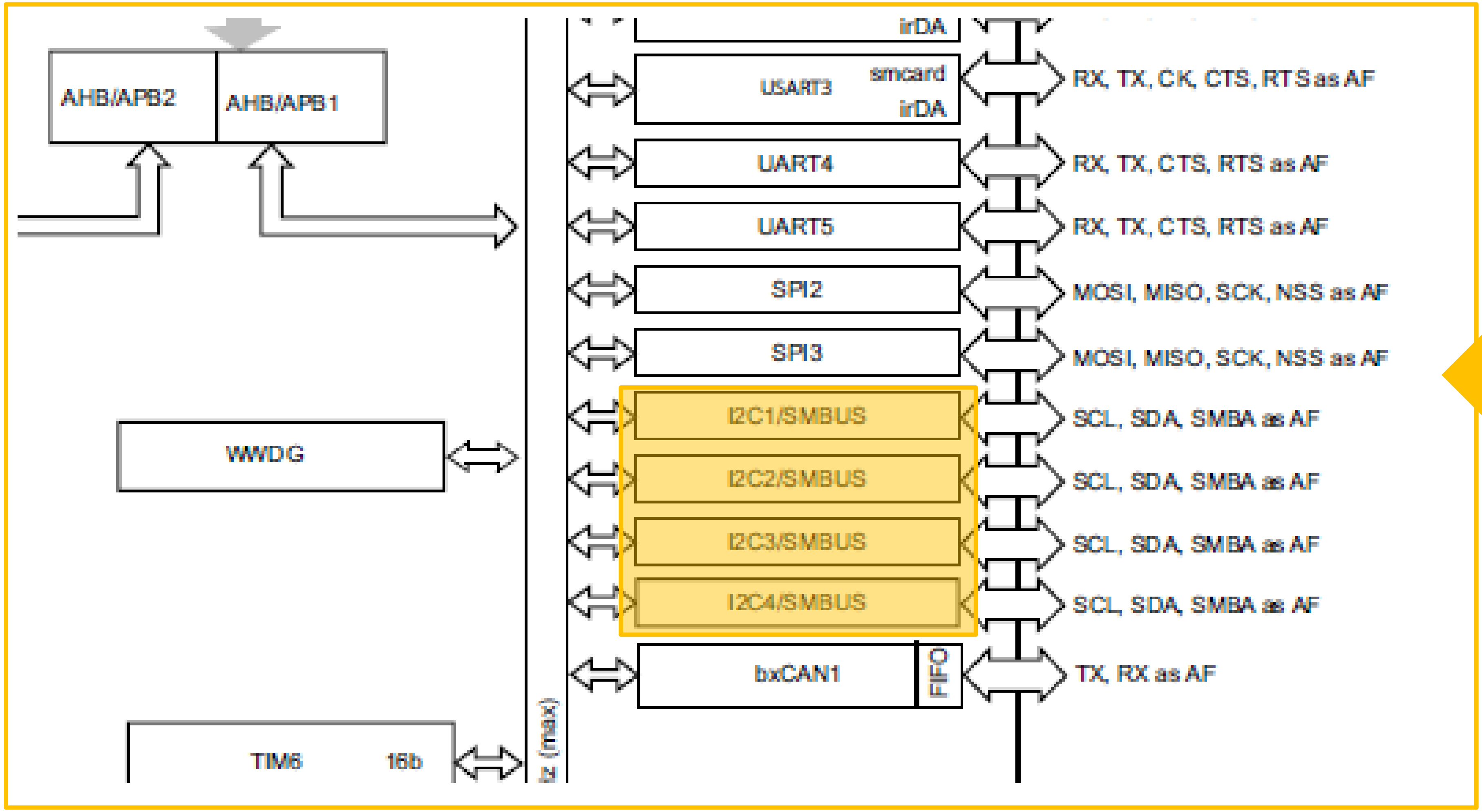
Example of **MCU – sensor** communication (data acquisition) via I^2C bus

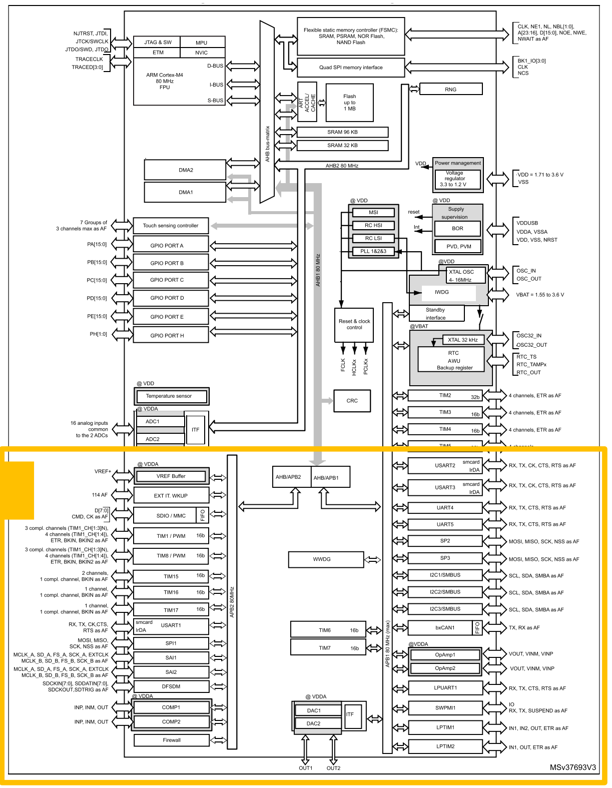


I²C – STM32F4xx

Datasheet STM32F401 – pag. 14

3 x I²C





I²C – Connected Sensors (Example)

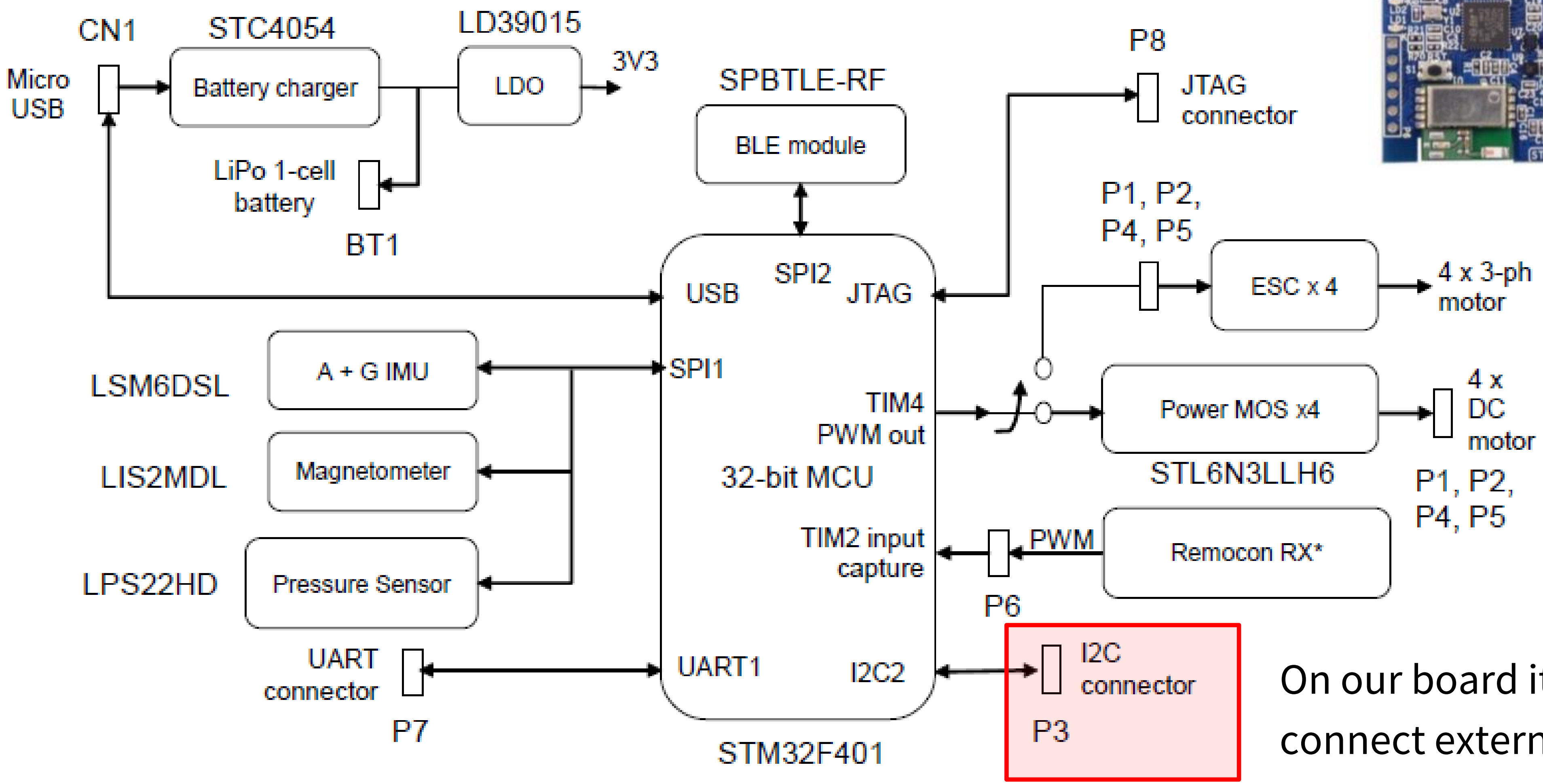
Humidity
Read=10111111(BFh)
Write=10111110(BEh)

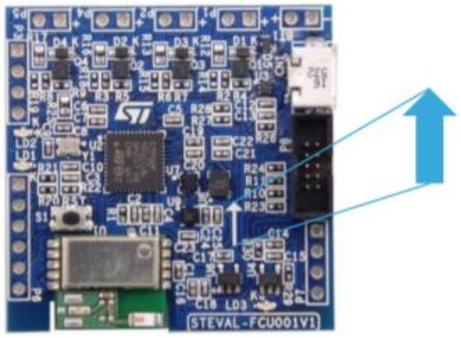
Temp. sensor
Read=10010001(91h)
Write=10010000(90h)

Magnetometer
Read=00111101(3Dh)
Write=00111100(3Ch)

Pressure
Read=10111011(BBh)
Write=10111010(BAh)

I²C – Connected Sensors Schematic





On our board it is used to connect external sensors

I^2C – Typical Datasheet - HTS221

Figure 1. HTS221 block diagram

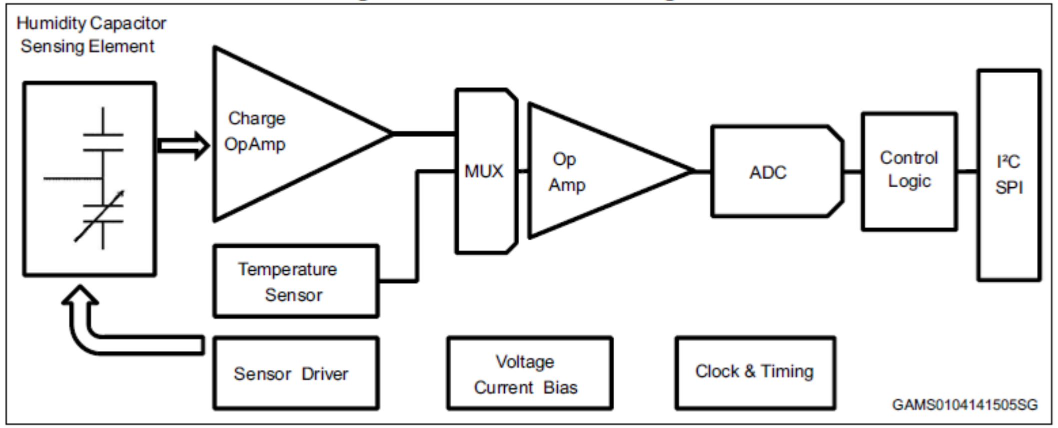


Figure 2. Pin configuration (bottom view)

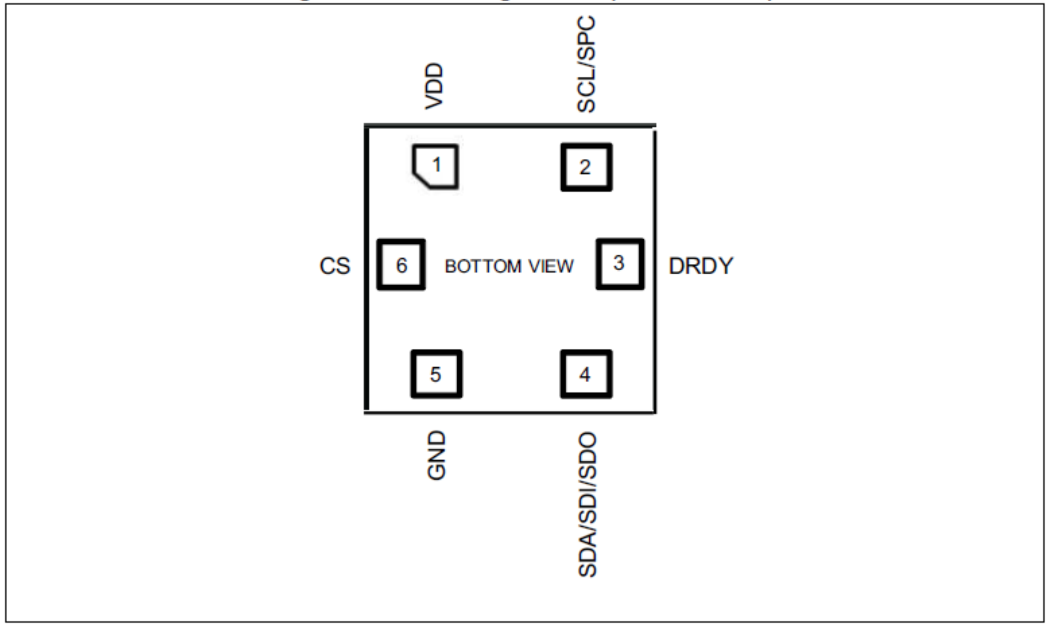


Table 10. SAD + Read/Write patterns

Command	SAD[6:0]	R/W	SAD+R/W
Read	1011111	1	10111111 (BFh)
Write	1011111	0	10111110 (BEh)

Address sometimes hidden in the text.

What happens if I want to use one sensor twice on the bus?

Table 2. Pin description

Pin n°	Name	Function
1	V_{DD}	Power supply
2	SCL/SPC	I²C serial clock (SCL) SPI serial port clock (SPC)
3	DRDY	Data Ready output signal
4	SDA/SDI/SDO	I²C serial data (SDA) 3-wire SPI serial data input /output (SDI/SDO)
5	GND	Ground
6	SPI enable	I²C/SPI mode selection (1: SPI idle mode / I²C communication enabled; 0: SPI communication mode / I²C disabled)

I^2C – Typical Datasheet - HTS221

Table 15. Register address map

Name	Type	Register address (hex)	Default (hex)
Reserved		00-0E	Do not modify
WHO_AM_I	R	0F	BC
AV_CONF	R/W	10	1B
Reserved		11-1C	Do not modify
CTRL_REG1	R/W	20	0
CTRL_REG2	R/W	21	0
CTRL_REG3	R/W	22	0
Reserved		23-26	Do not modify
STATUS_REG	R	27	0
HUMIDITY_OUT_L	R	28	Output
HUMIDITY_OUT_H	R	29	Output
TEMP_OUT_L	R	2A	Output
TEMP_OUT_H	R	2B	Output
Reserved		2C-2F	Do not modify
CALIB_0..F	R/W	30-3F	Do not modify

Configuration

Data

Calibration

WHO_AM_I (0Fh)

Device identification

7	6	5	4	3	2	1	0
1	0	1	1	1	1	0	0

This read-only register contains the device identifier, set to BCh

7.9 TEMP_OUT_L (2Ah)

Temperature data (LSB)

7	6	5	4	3	2	1	0
TOUT7	TOUT6	TOUT5	TOUT4	TOUT3	TOUT2	TOUT1	TOUT0

[7:0]	TOUT7 - TOUT0: Temperature data LSB (see TEMPERATURE_OUT_H)

7.10 TEMP_OUT_H (2Bh)

Temperature data (MSB)

15	14	13	12	11	10	9	8
TOUT15	TOUT14	TOUT13	TOUT12	TOUT11	TOUT10	TOUT9	TOUT8

[15:8]	TOUT15 - TOUT8: Temperature data MSB.

Temperature data are expressed as TEMP_OUT_H & TEMP_OUT_L as 2's complement numbers.

ETH zürich

SPI – Serial Peripherals Interface

SPI: Serial Peripheral Interface - 1

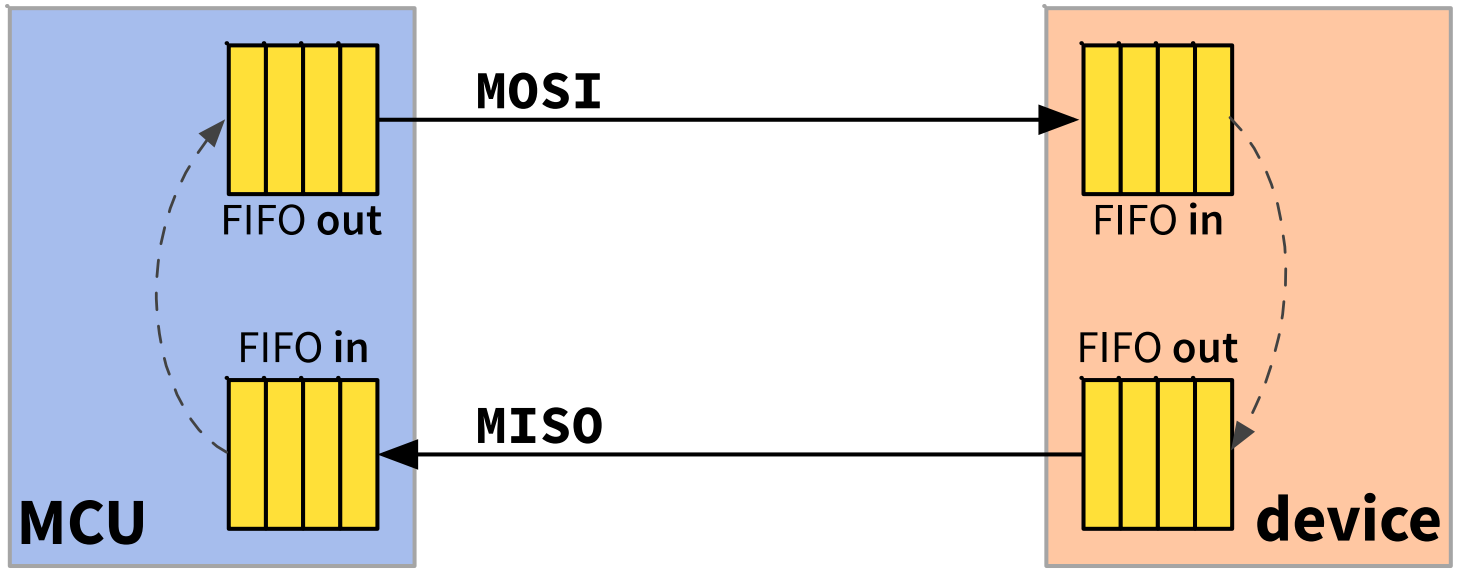
- Introduced by Motorola (now Freescale Semiconductors) for the MC68HCxx line of microcontrollers
- Use cases are generally similar to I2C
- Generally faster than I2C (up to several Mbit/s)
  - Short-distance (i.e. on printed circuit boards)
- Single-master, multiple slave
  - needs one chip select per slave device (no broadcast addressing)
- Full-duplex synchronous communication scheme
  - master drives the clock (SCLK or SCK)
  - clock polarity (i.e. write/read edges) and phase depend on specific application!

SPI: Serial Peripheral Interface - 2

- Based on two data and two control lines:
  - MISO (master-in, slave-out data)
  - MOSI (master-out, slave-in data)
  - SCK (clock)
  - CSN (chip select, one per slave – usually active low)
- Names are not standard, beware! Some possible alternatives:
  - SDI (SPI data in) instead of MISO
  - SDO (SPI data out) instead of MOSI
  - SCLK, CLK, SPC, … instead of SCK
  - CS, SS (slave select), SSN (slave select, active low) ... instead of CSN

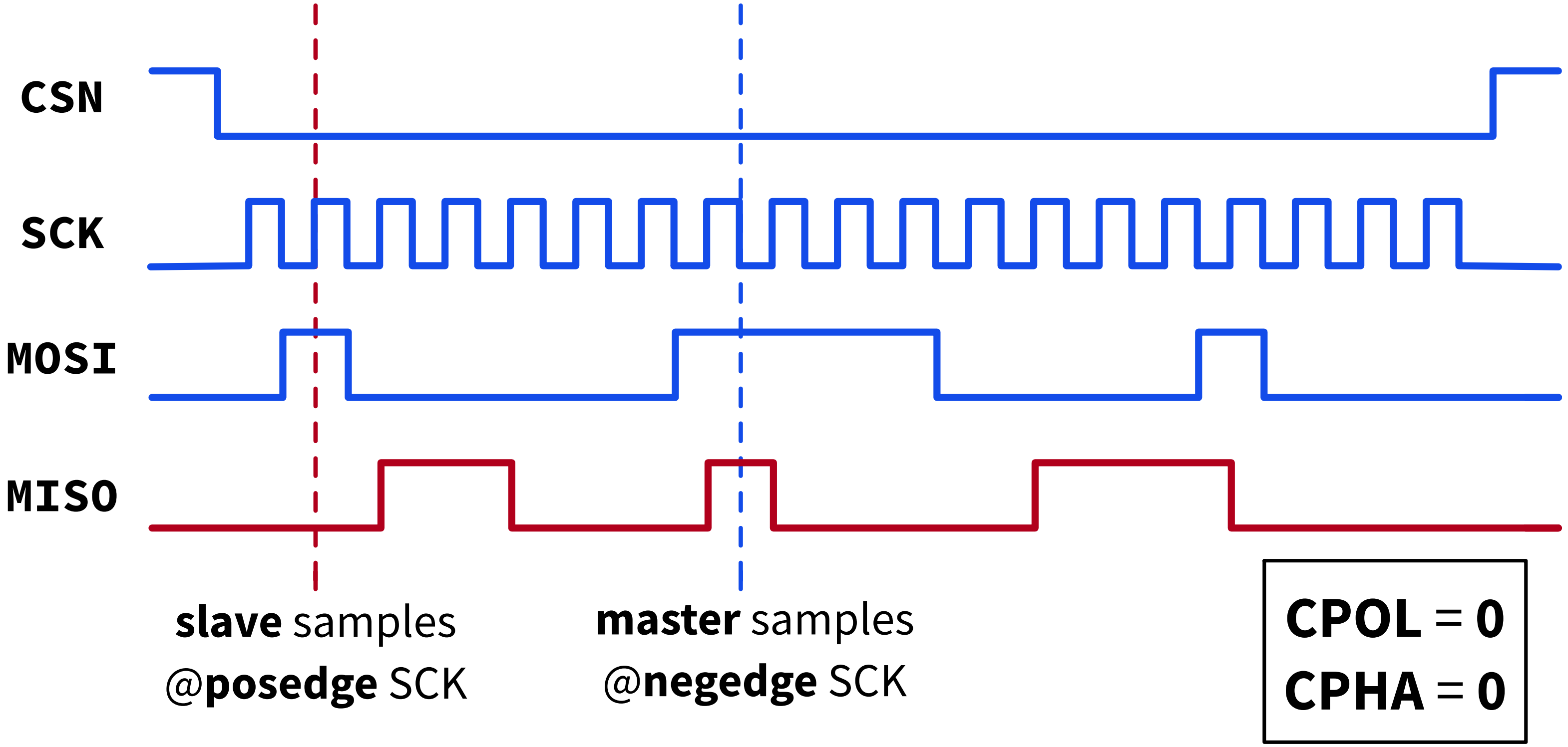
SPI: Serial Peripheral Interface - 3

- Full-duplex transfer: data is streamed between master and slave shift-registers / FIFO buffers:
  - the master pushes the content of its buffer to the slave via MOSI
  - the slave pushes the content of its buffer to the master via MISO
- Processing / sensing / … happens in between (dashed line)



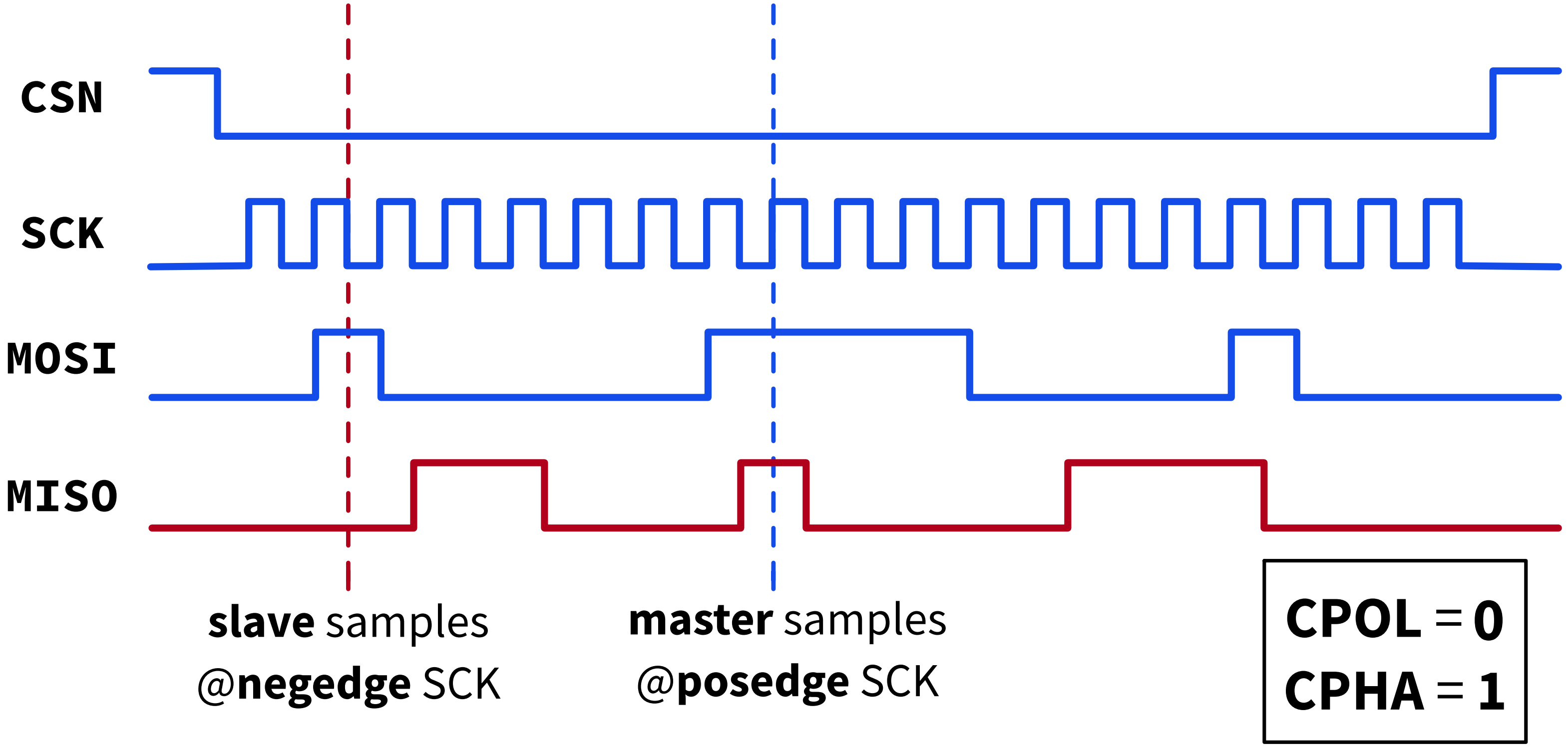
SPI: Interface Protocol - 1

- Four operating modes, varying by clock polarity (CPOL) and phase (CPHA):
  - polarity sets the initial value of the SPI clock signal
  - phase defines the edge at which MOSI is switched and the one at which MISO is sampled



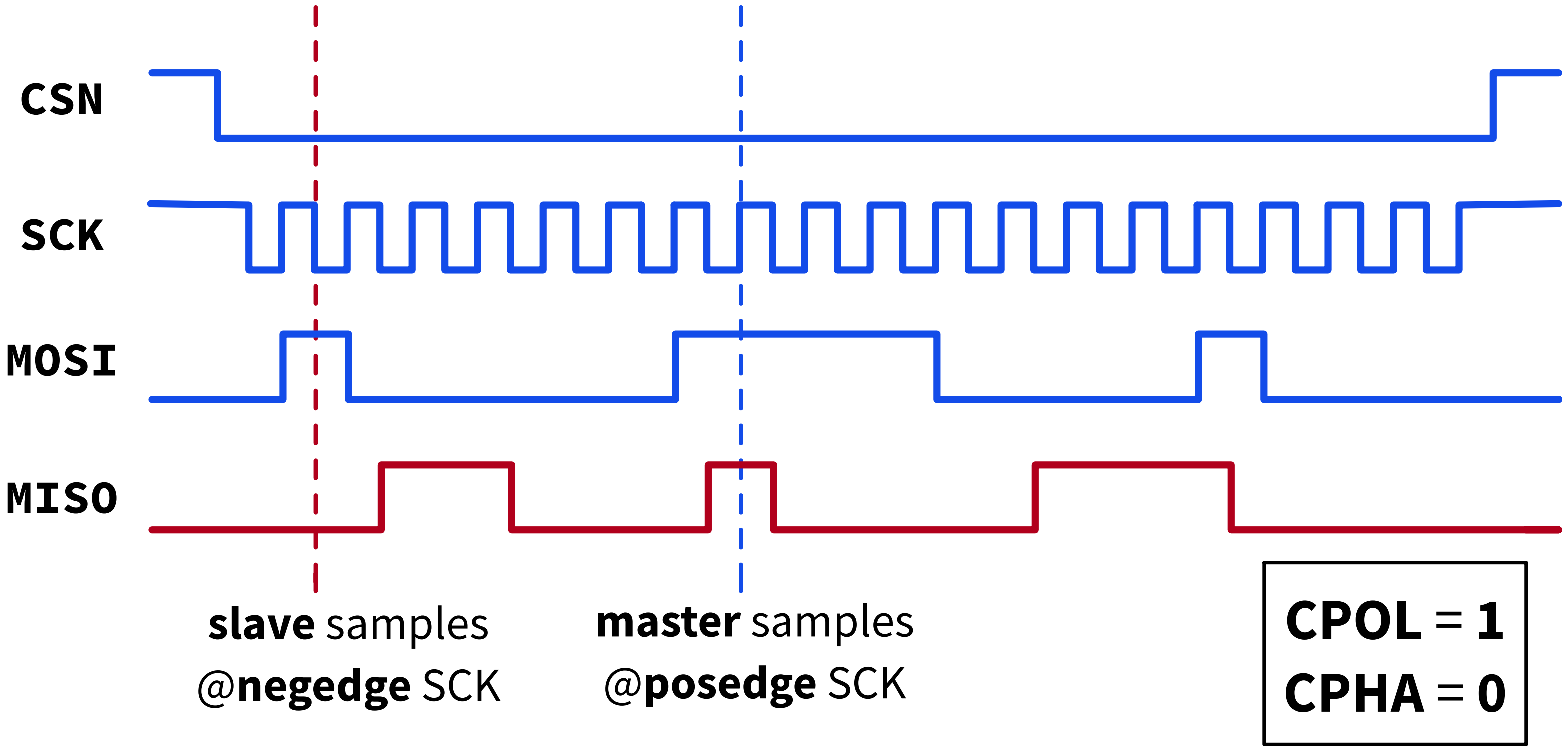
SPI: Interface Protocol - 1

- Four operating modes, varying by clock polarity (CPOL) and phase (CPHA):
  - polarity sets the initial value of the SPI clock signal
  - phase defines the edge at which MOSI is switched and the one at which MISO is sampled



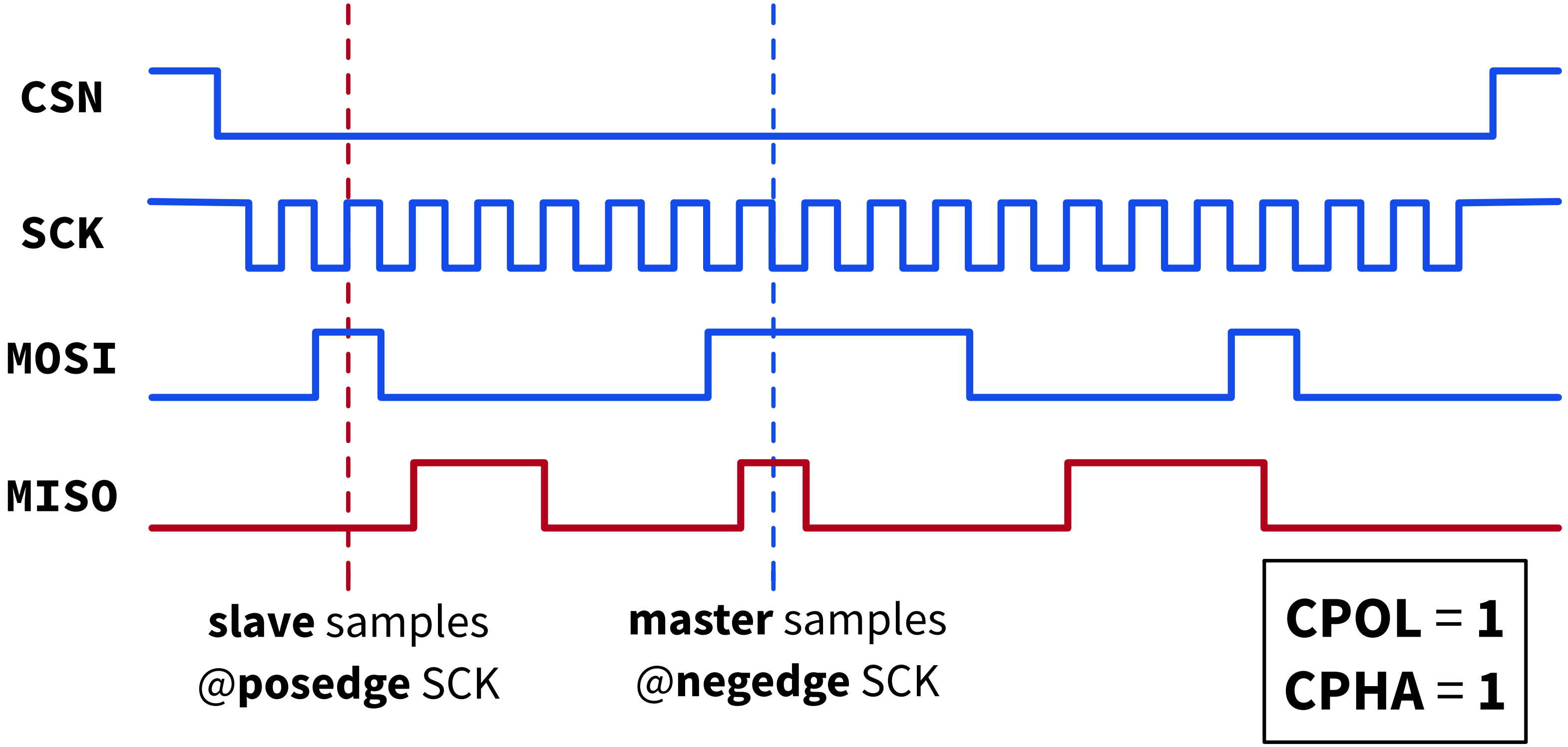
SPI: Interface Protocol - 1

- Four operating modes, varying by clock polarity (CPOL) and phase (CPHA):
  - polarity sets the initial value of the SPI clock signal
  - phase defines the edge at which MOSI is switched and the one at which MISO is sampled



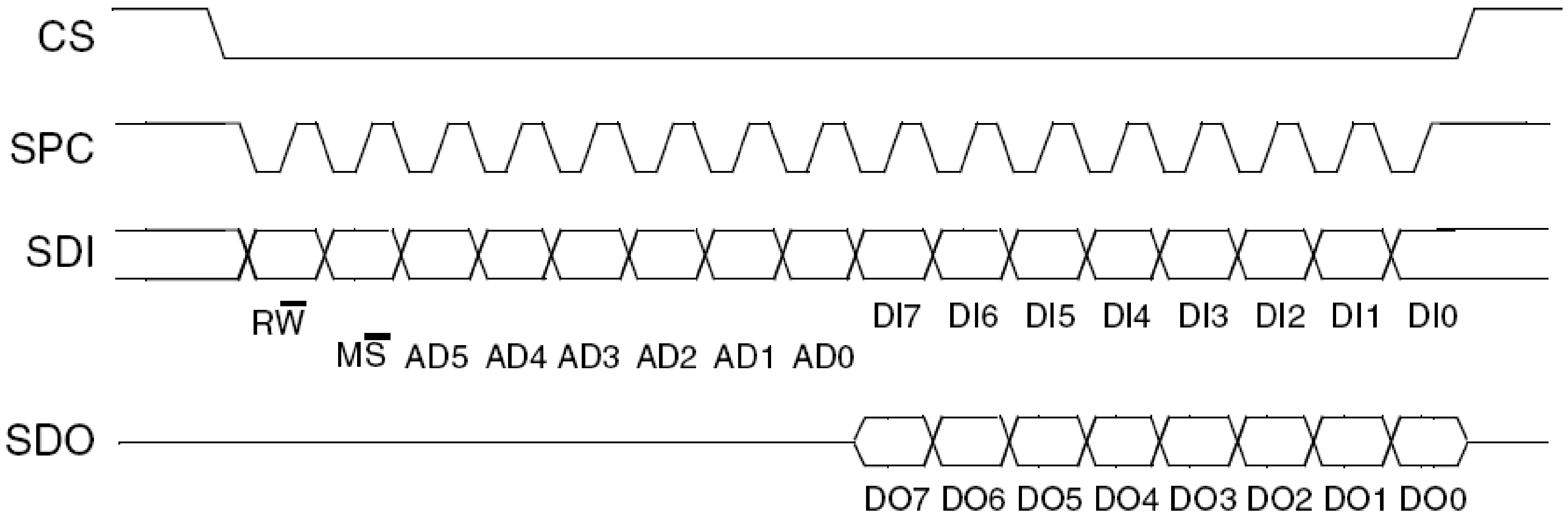
SPI: Interface Protocol - 1

- Four operating modes, varying by clock polarity (CPOL) and phase (CPHA):
  - polarity sets the initial value of the SPI clock signal
  - phase defines the edge at which MOSI is switched and the one at which MISO is sampled



SPI: Interface Protocol - 2

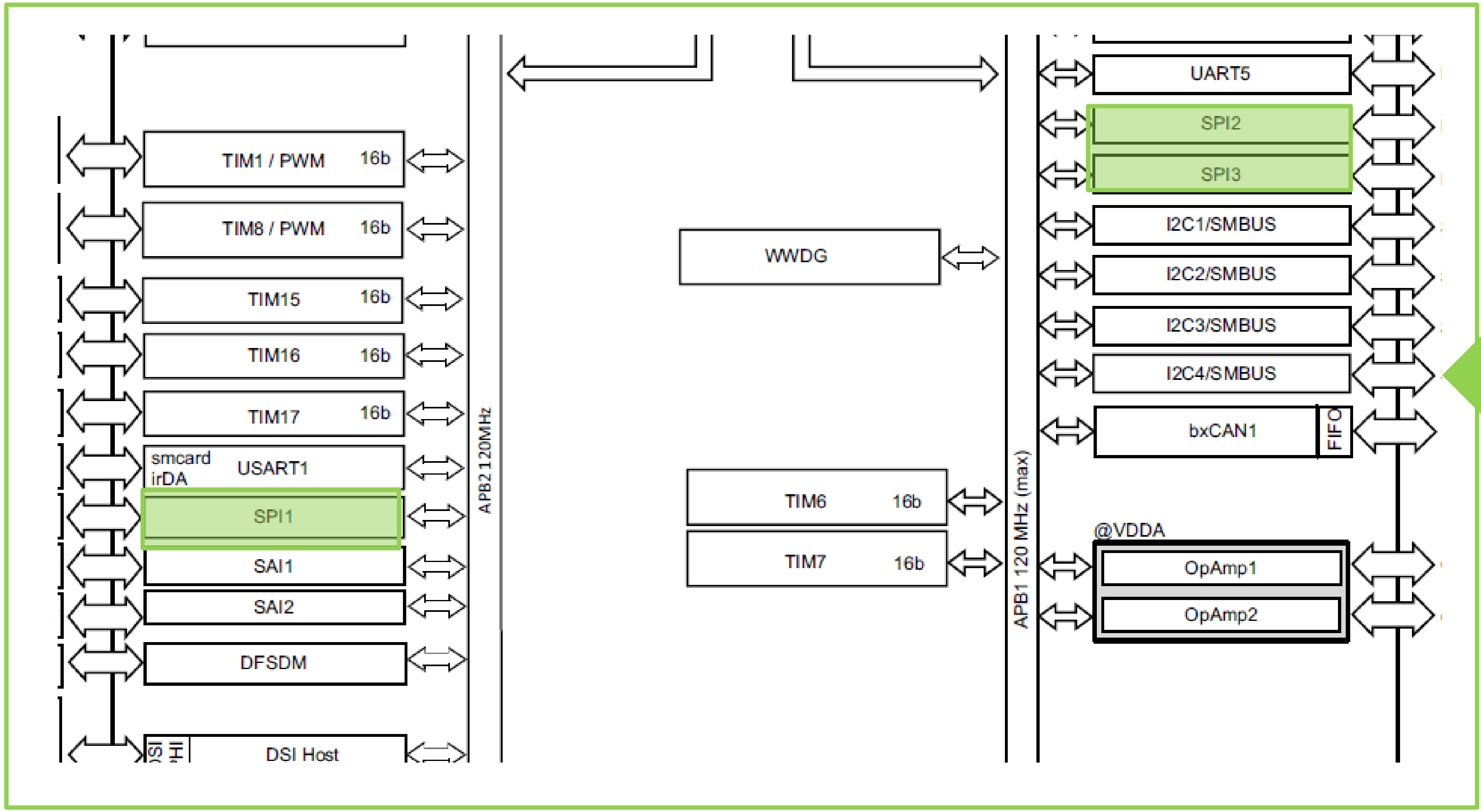
- Master completely in charge of transfer
  - no ack, no clock stretching contrarily to I2C
- More complex behavior than simple data streaming can be mapped on top of SPI protocol
  - e.g. command + address + data streaming

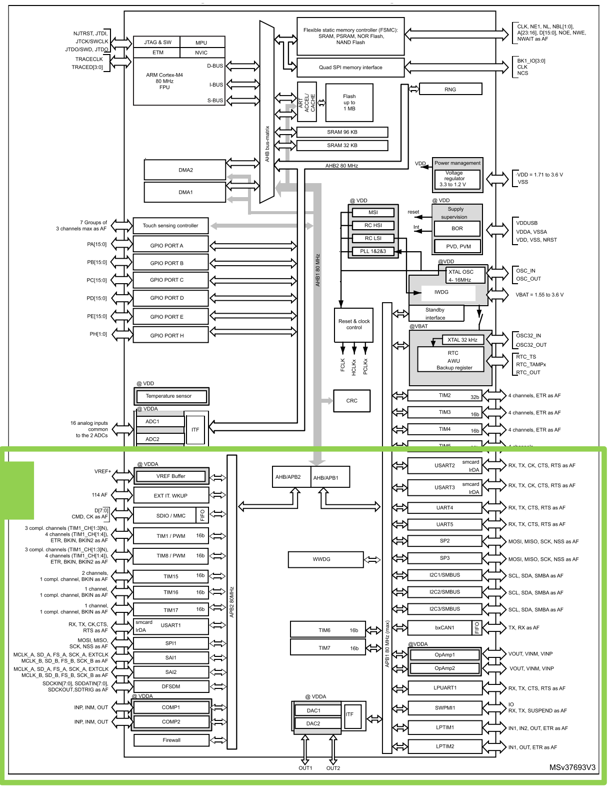


SPI – STM32F401xx

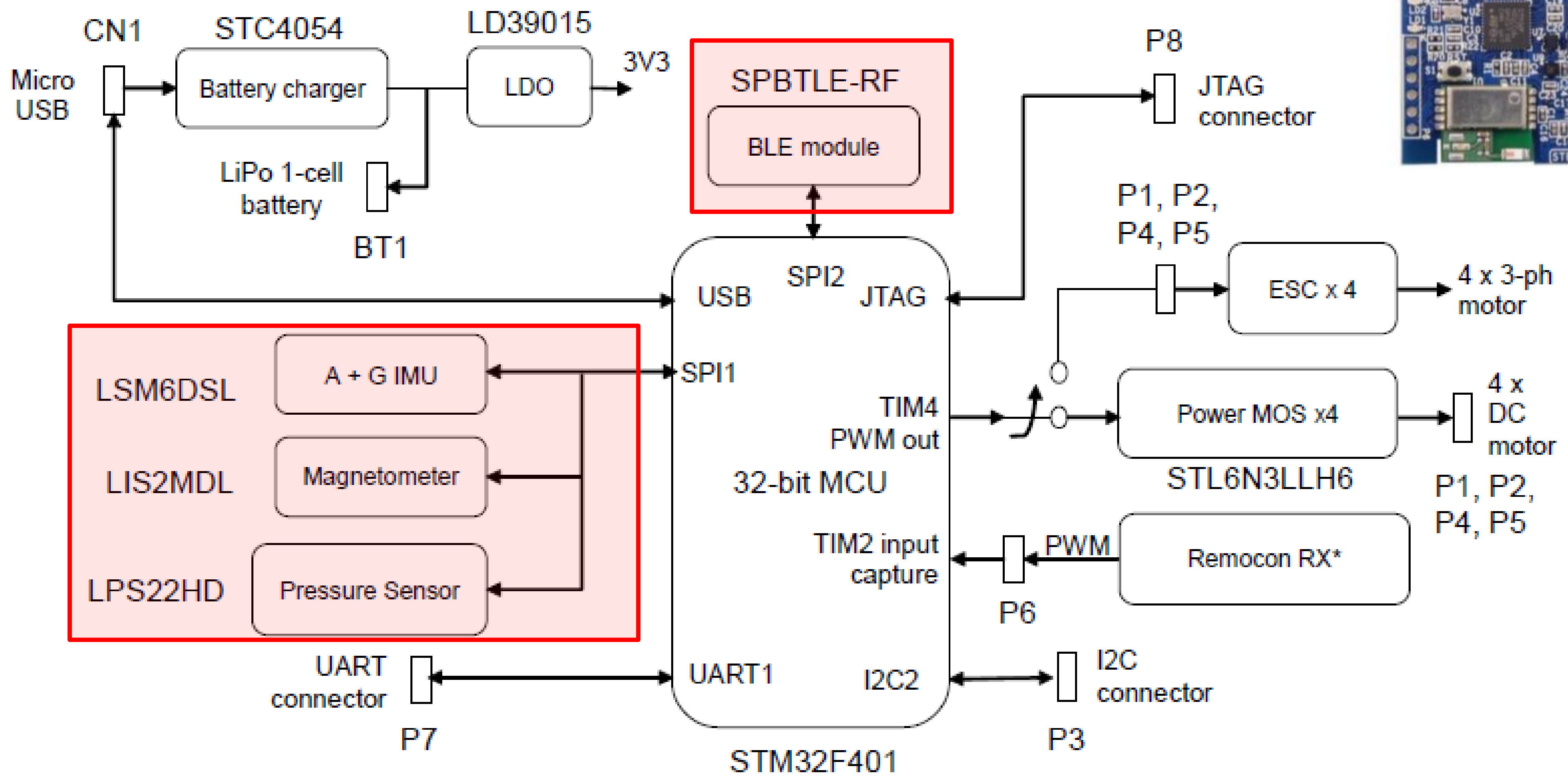
Datasheet STM32F401 – pag. 14

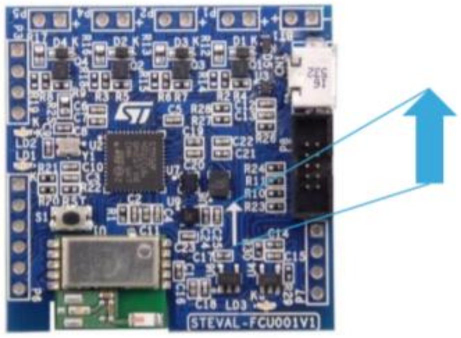
4 x SPI





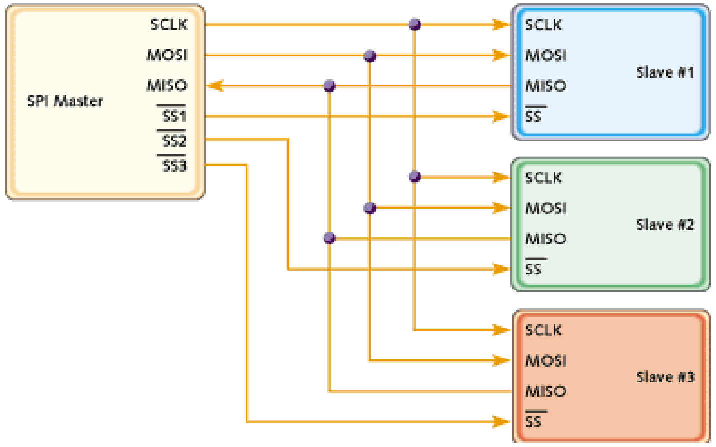
SPI – Connected Sensors



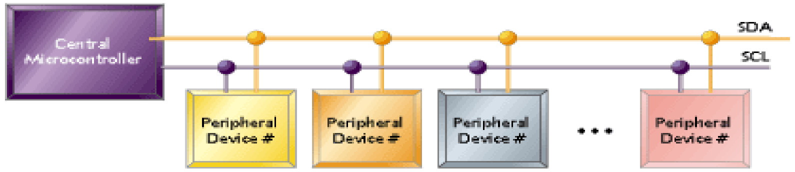


SPI vs I2C

- For **point-to-point**, SPI is simple and efficient
  - Less overhead than I2C due to lack of addressing, plus SPI is full-duplex.
- For multiple slaves, each slave needs separate slave select signal
  - SPI requires more effort and more hardware than I2C
- Quad-SPI also exists
  - 4x the bandwidth, often used by Flash drives



SPI



I²C

ETH zürich

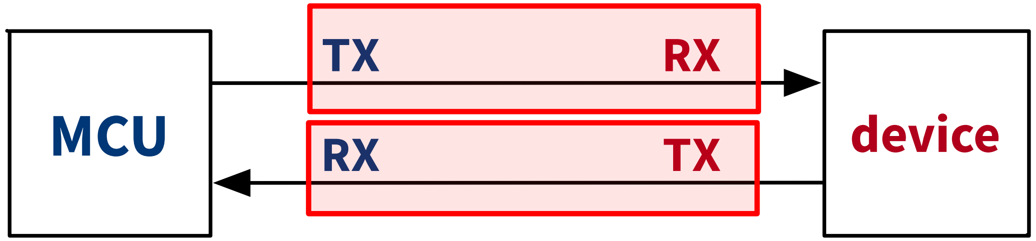
UART - Universal Asynchronous Receiver-Transmitter

UART - 1

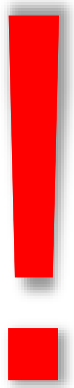
- Stands for Universal Asynchronous Receiver-Transmitter
  - sometimes also found as USART (Universal Synchronous-Asynchronous Receiver Transmitter)
- Used to interface MCUs with other computing devices:
  - Communication with other processors, a PC (e.g. a serial terminal)
  - Used to interface the microcontroller with others transmission bus as: RS232, RS485, USB, CAN BUS, KNX, LonWorks ecc.
  - Used to connect MCUs with modems and transceivers as telephone modems, Bluetooth, Wi-Fi, GSM/GPRS/HDPSA

UART - 2

- Essentially a parallel2serial (TX), serial2parallel (RX) converter couple
  - e.g. using shift registers for P2S conversion
- Asynchronous: no common clock shared
  - Each device has its own local clock, typically running faster than the bit rate (e.g. 8x faster)
  - The phase of the receiver clock is locked onto the edge of the transmitted data



- Highly configurable
  - parity / no parity
  - data framing (e.g number of stop bits, number of payload bits)
  - simplex, full-duplex or half-duplex



UART: “baud rate” vs “bit rate”

- UART communication speed is defined by its symbol rate measured in baud:
  - 1 baud = 1 **symbol per second**
  - in UART, a **symbol** has two values (0/1) -> **1 bit**
  - this number includes both data payload and protocol bits (e.g. parity, framing) – this number is also called “physical” or “gross” bit rate
- This can cause some confusion
  - Some people use “bit rate” for UART when referring only to payload bits
  - In some devices (e.g. modems) one symbol
  - might correspond to more bits -> baud rate is not the same as gross bit rate
  - Bottom line: to be 100% clear, **always talk of baud rate when referring to UART**, and remember that in UART **1 symbol = 1 bit**

Gross bit rate [edit]

In digital communication systems, the physical layer *gross bitrate*,[5] *raw bitrate*,[6] *data signaling rate*,[7] *gross data transfer rate*[8] or *uncoded transmission rate*[6] (sometimes written as a variable R_b[5][6] or f_b[9]) is the total number of physically transferred bits per second over a communication link, including useful data as well as protocol overhead.

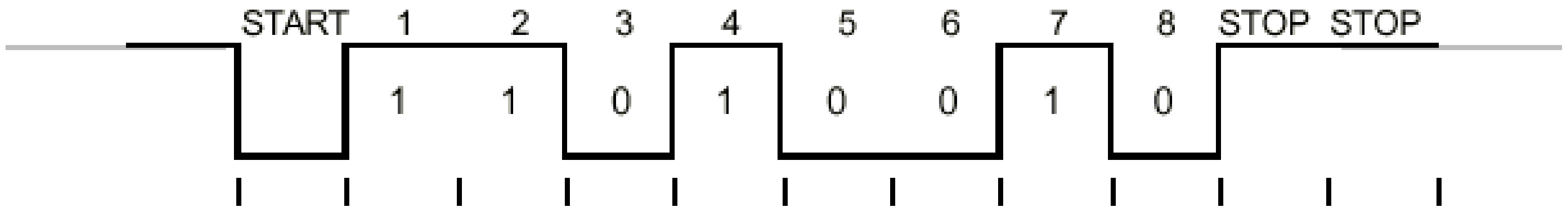
In case of serial communications, the gross bit rate is related to the bit transmission time T_b as:

$$R_b = \frac{1}{T_b},$$

The gross bit rate is related to the symbol rate or modulation rate, which is expressed in bauds or symbols per second. However, the gross bit rate and the baud value are equal *only* when there are only two levels per symbol, representing 0 and 1, meaning that each symbol of a data transmission system carries exactly one bit of data; for example, this is not the case for modern modulation systems used in modems and LAN equipment.[10]

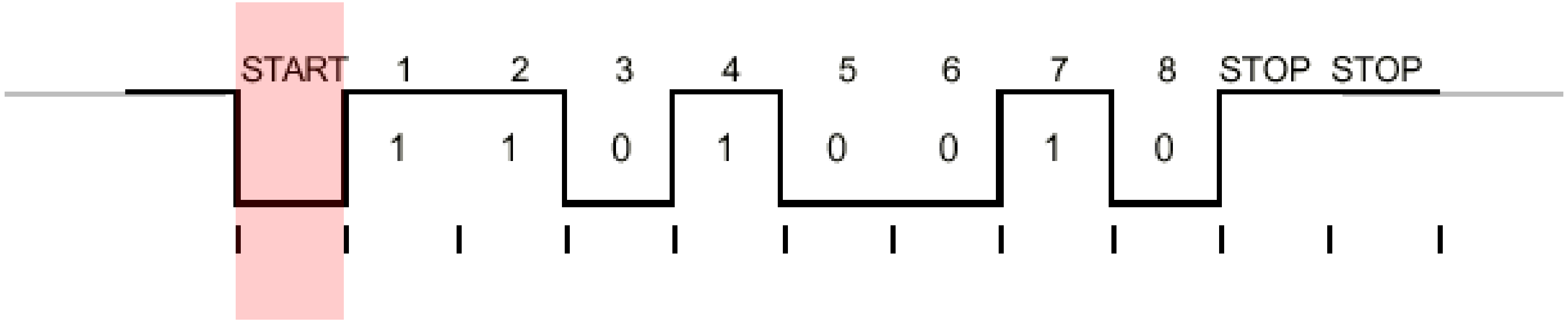
Ref. Wikipedia “Bit rate” page

UART: Interface Protocol



- In idle, the transmission line is driven to 1

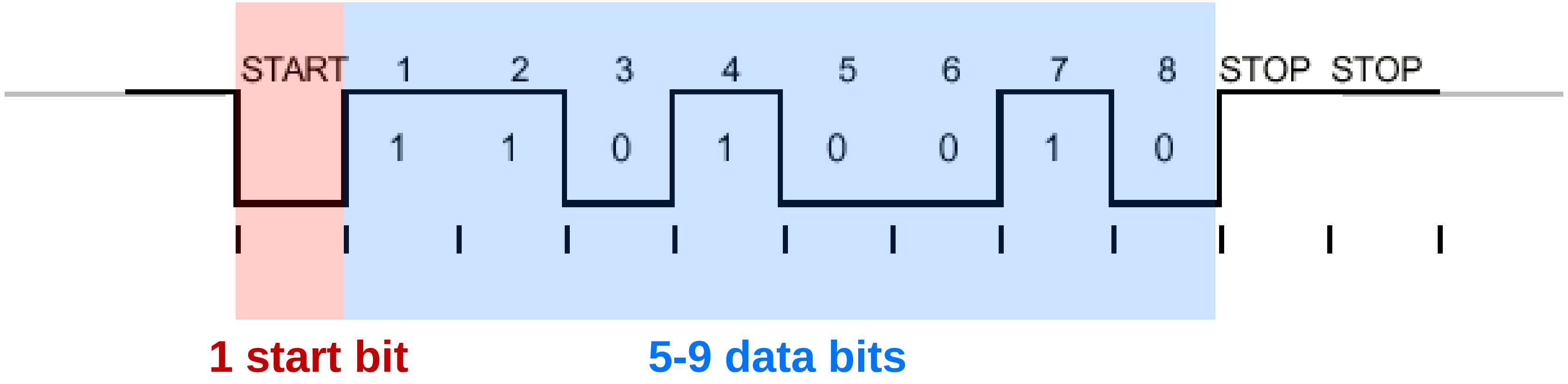
UART: Interface Protocol



1 start bit

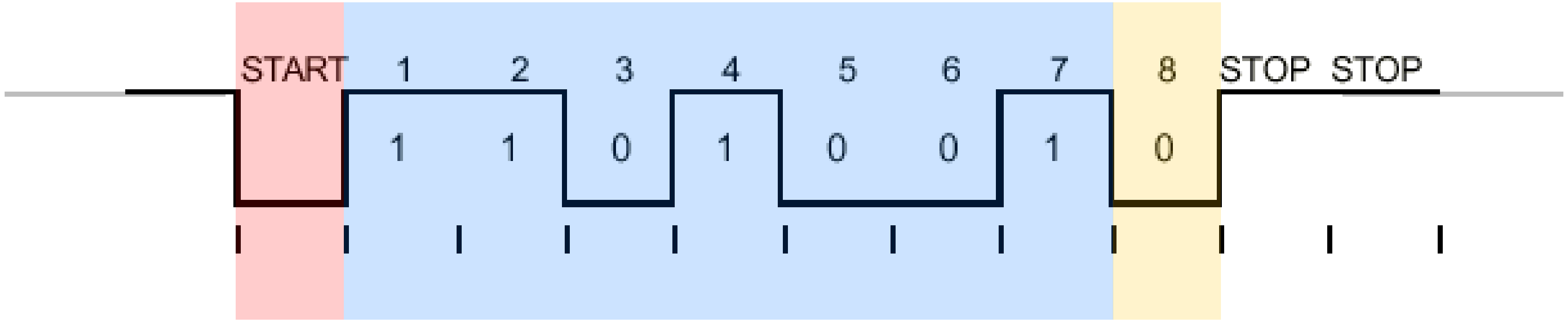
- The transfer begins with a start bit:
  - the transmission line is driven to 0

UART: Interface Protocol



- Then, a symbol of 5 to 9 bits is transmitted:
  - most often, 8 bits (1 ASCII character)
  - the symbol size is defined by the application and known a-priori with respect to the communication

UART: Interface Protocol



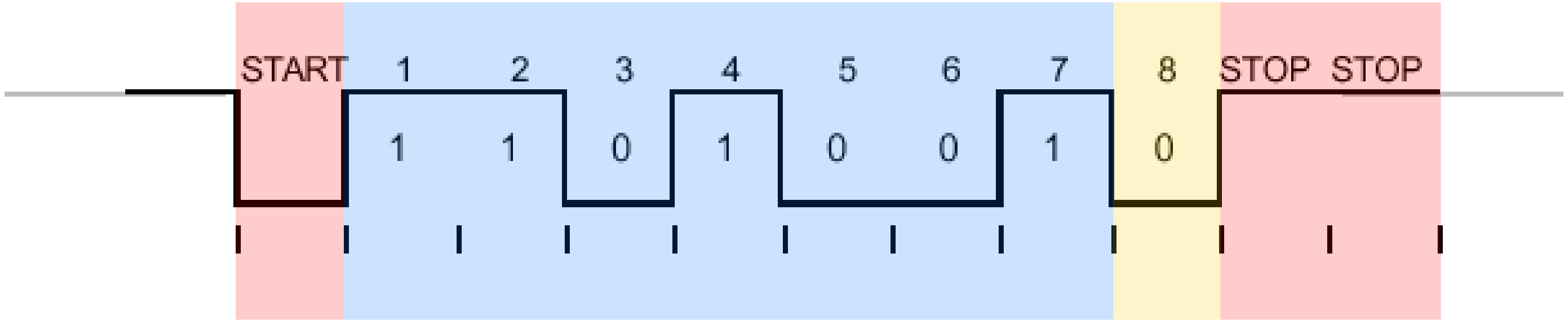
1 start bit

4-8 data bits

1 parity bit

- One of the data bits can be used for parity:
  - odd parity $p_{\text{odd}} = \mathop{\mathbf{XOR}}_{i=1}^{N} b_i$ even parity $p_{\text{even}} = \mathop{\mathbf{XNOR}}_{i=1}^{N} b_i$
  - in this case, 4-8 bits can be used for data

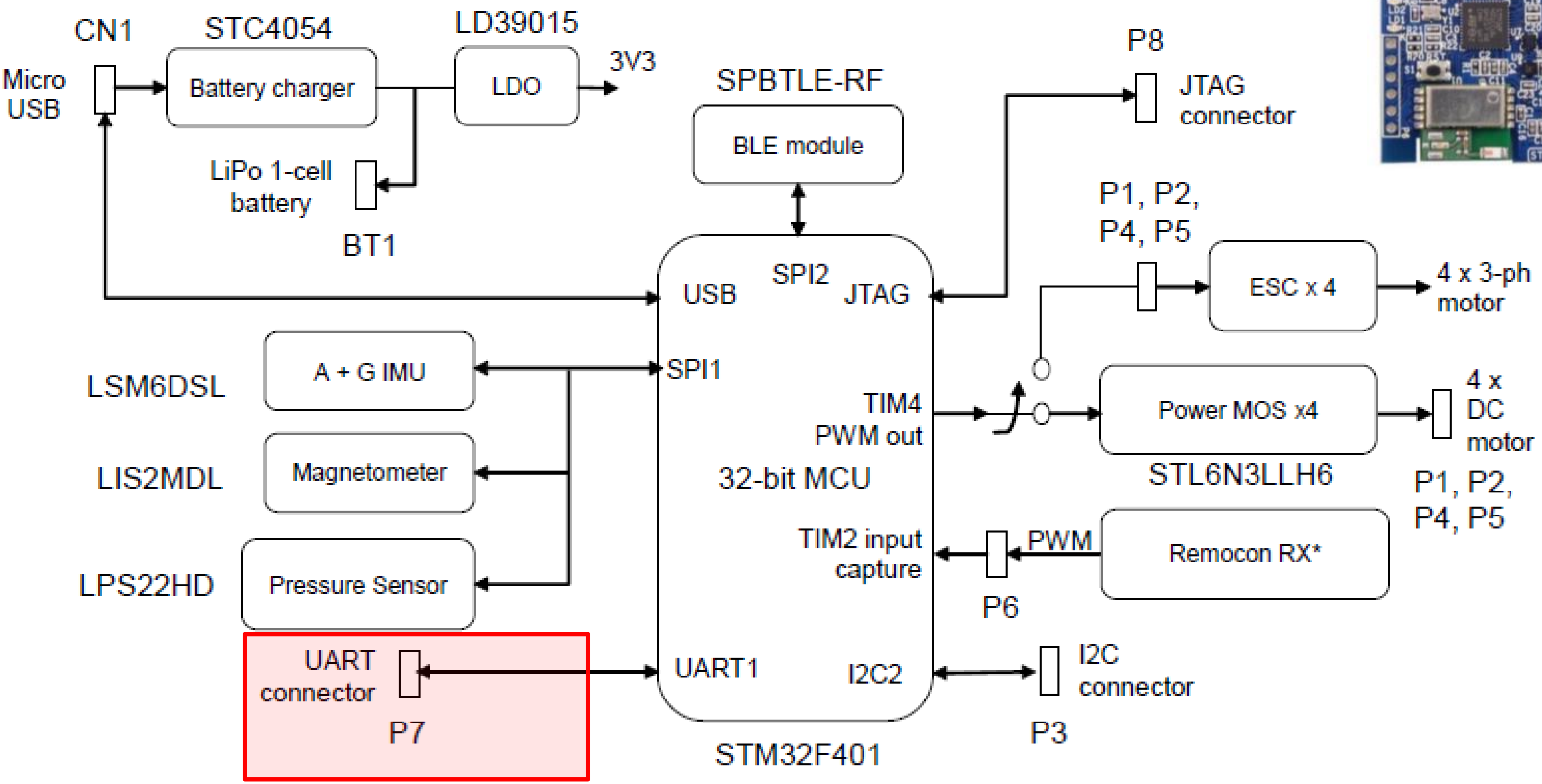
UART: Interface Protocol

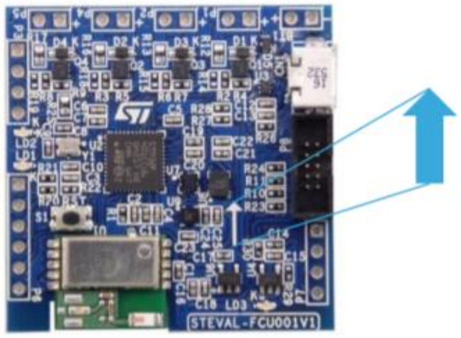


1 start bit **4-8 data bits** **1 parity bit** **1-2 stop bits**

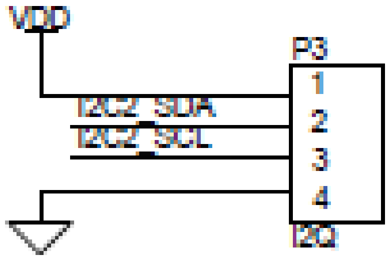
- Finally, 1-2 stop bits:
  - Transmission line brought back to 1
  - 1 or 2 stop bits depending on application

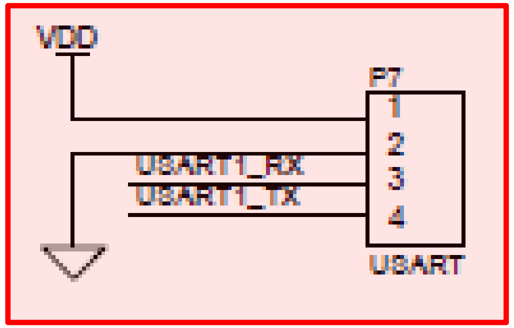
UART – Connected Sensors

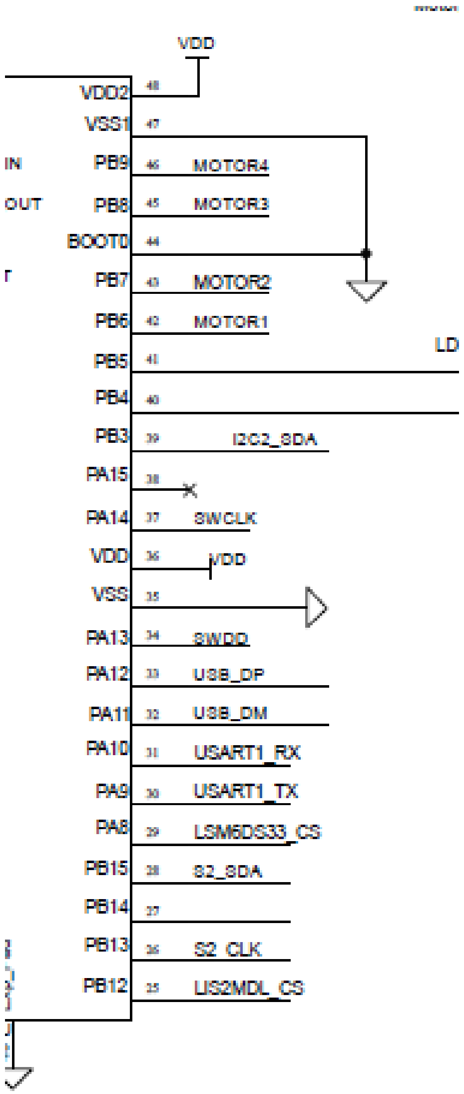


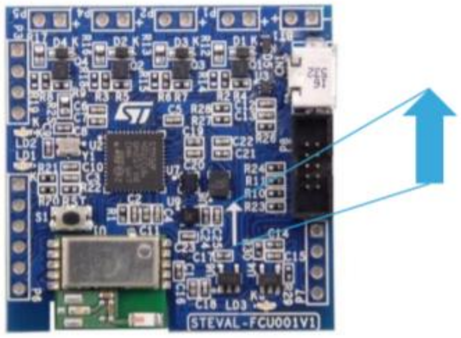


UART – Connected Sensors









- USART1_RX to **TDX**
- USART1_TX to **RDX**
- GND - **GND**

- Document: dm00434690.pdf

ETH zürich

IMU Inertial Measurement Unit

What is a MEMS?

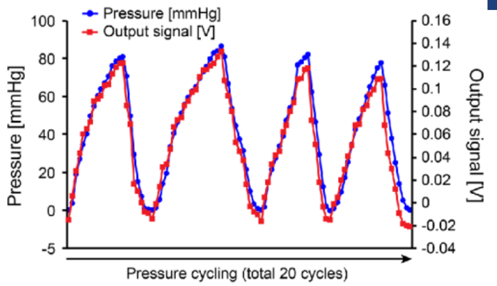
MicroElectroMechanical System = **MEMS**

MEMS, *also written as micro-electro-mechanical systems (or microelectronic and microelectromechanical systems) and the related micromechatronics and microsystems constitute the technology of microscopic devices, particularly those with moving parts. They merge at the nanoscale into nanoelectromechanical systems (NEMS) and nanotechnology.*

***MEMS** are made up of components between 1 and 100 micrometers in size (i.e., 0.001 to 0.1 mm), and MEMS devices generally range in size from 20 micrometres to a millimetre (i.e., 0.02 to 1.0 mm).*

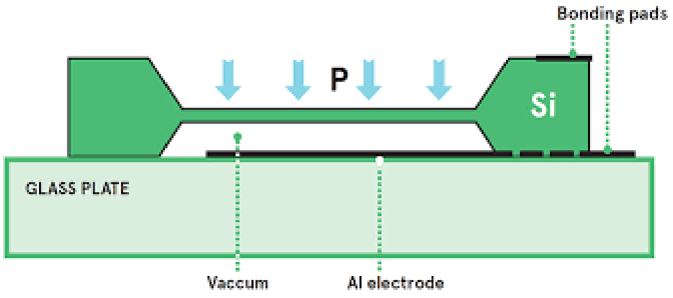
MEMS working principle

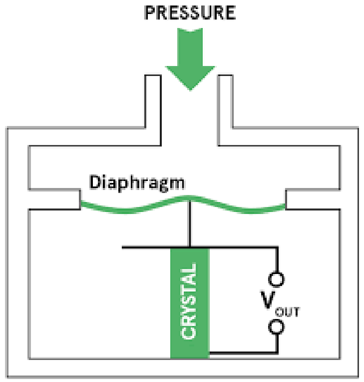
Output: Voltage
Current

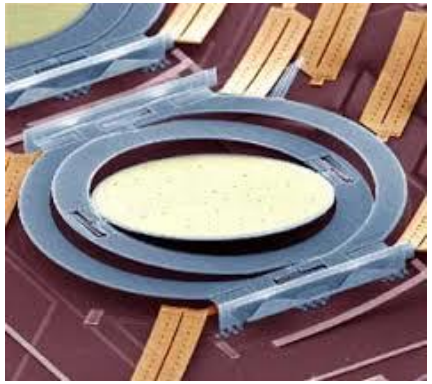


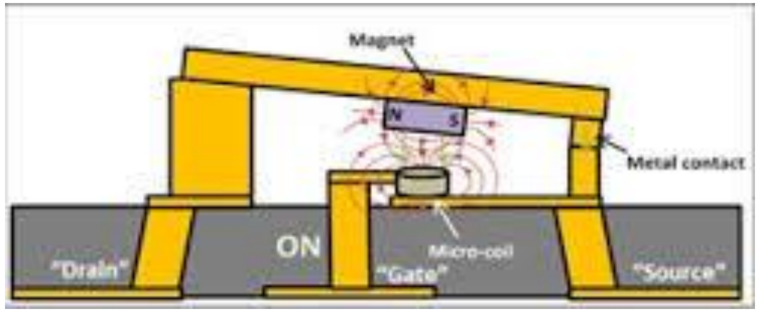
MEMS

- CAPACITIVE
- PIEZORESISTIVE
- OPTICAL (LASER)
- MAGNETIC

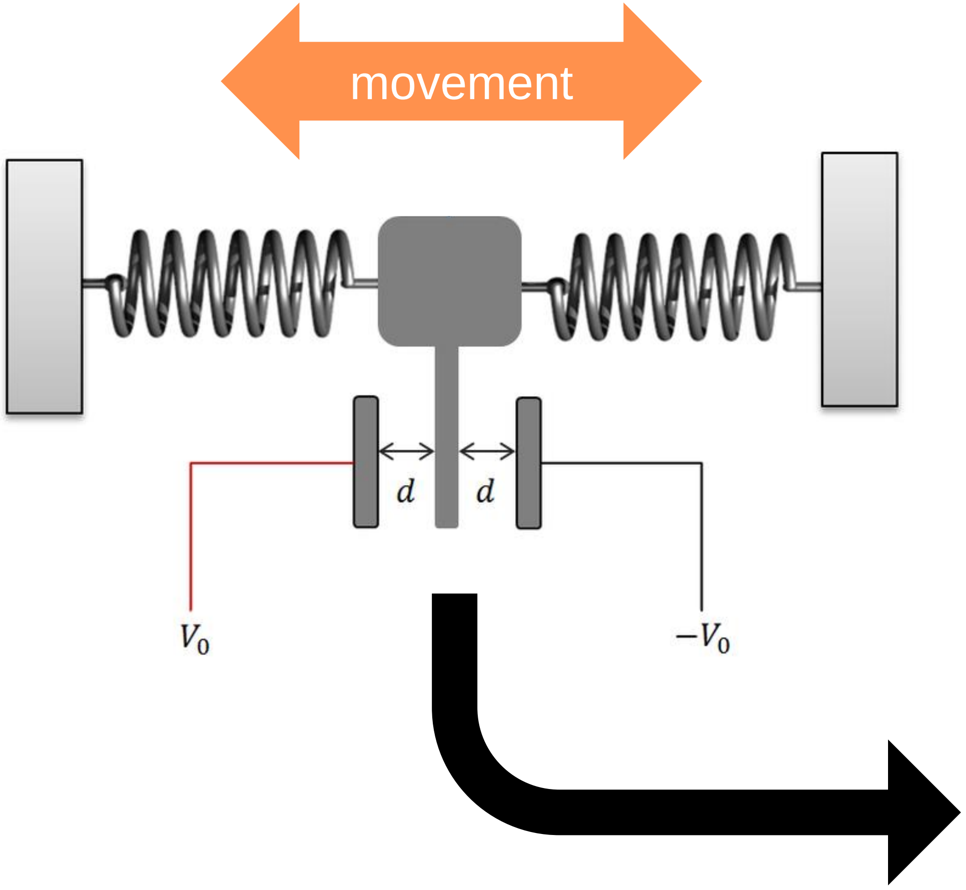


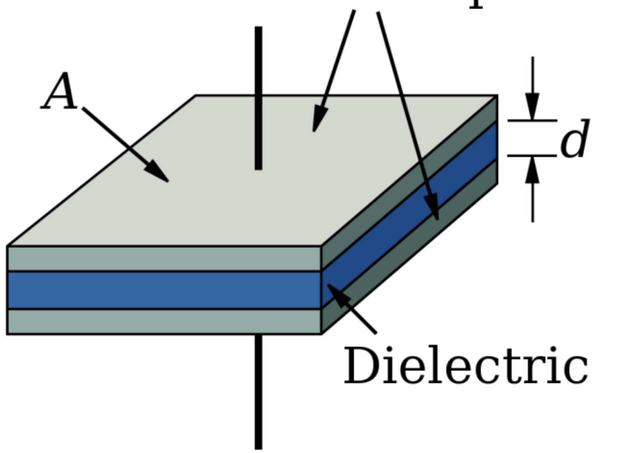






Capacitive MEMS





capacitance [F]

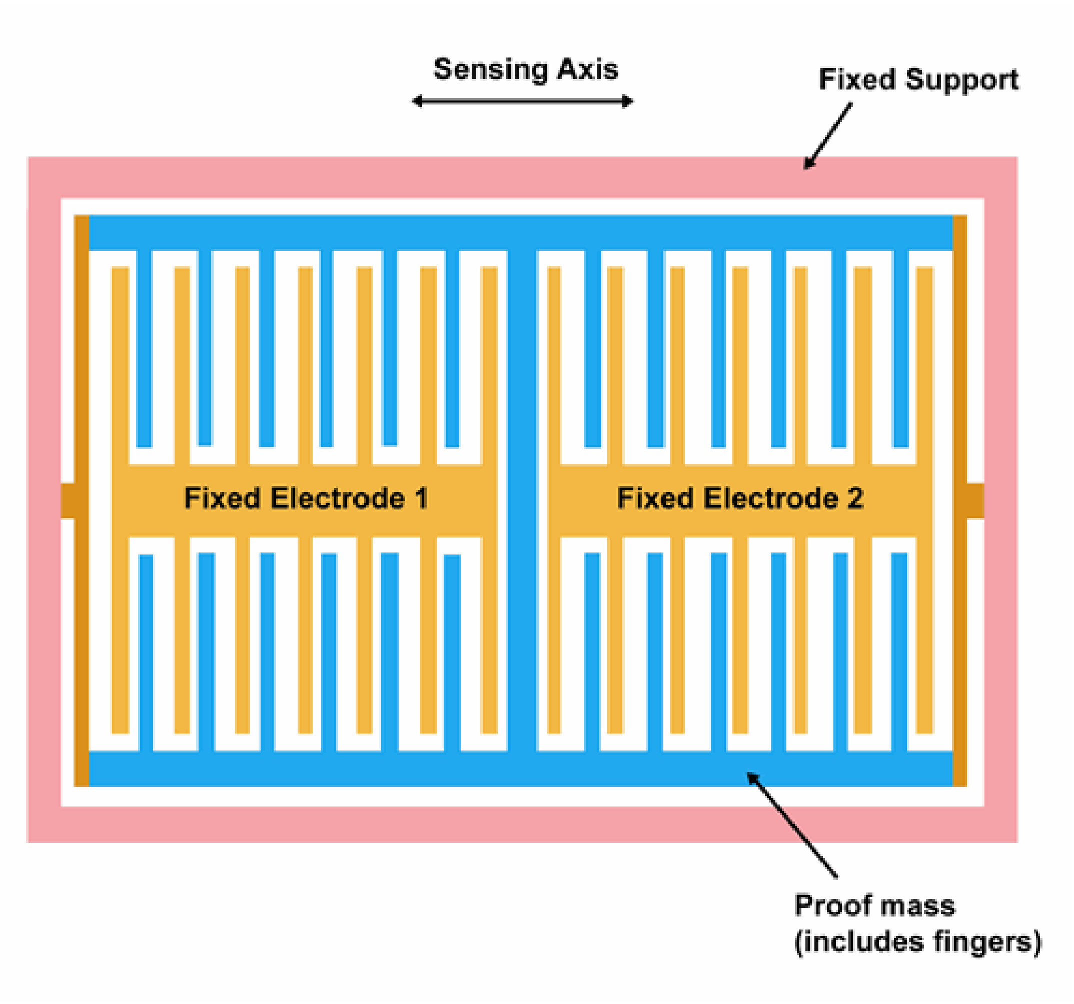
dielectric coefficient

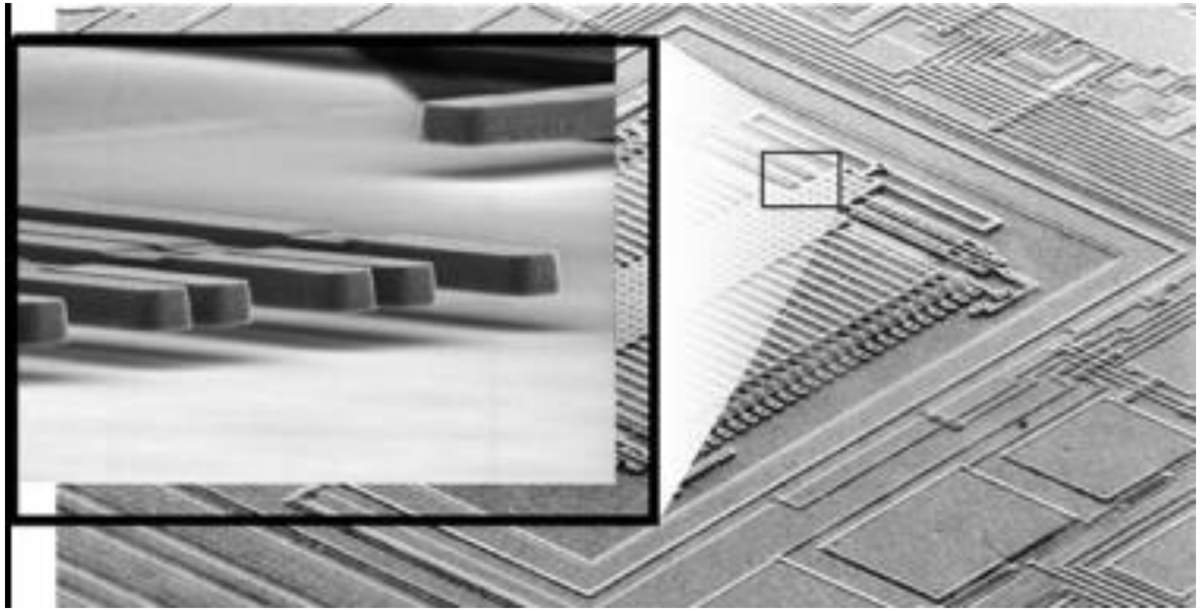
$$C = \frac{\varepsilon A}{d}$$

Area [m²]

Distance [m]

Capacitive MEMS - fabrication

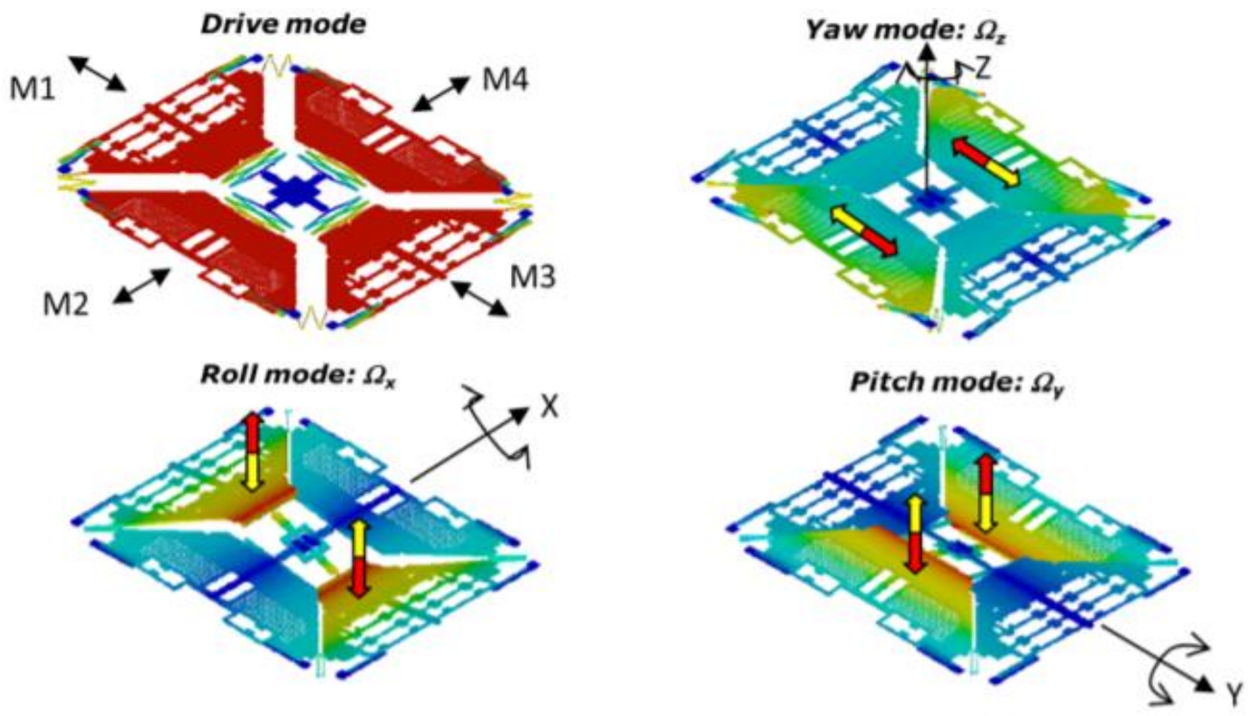


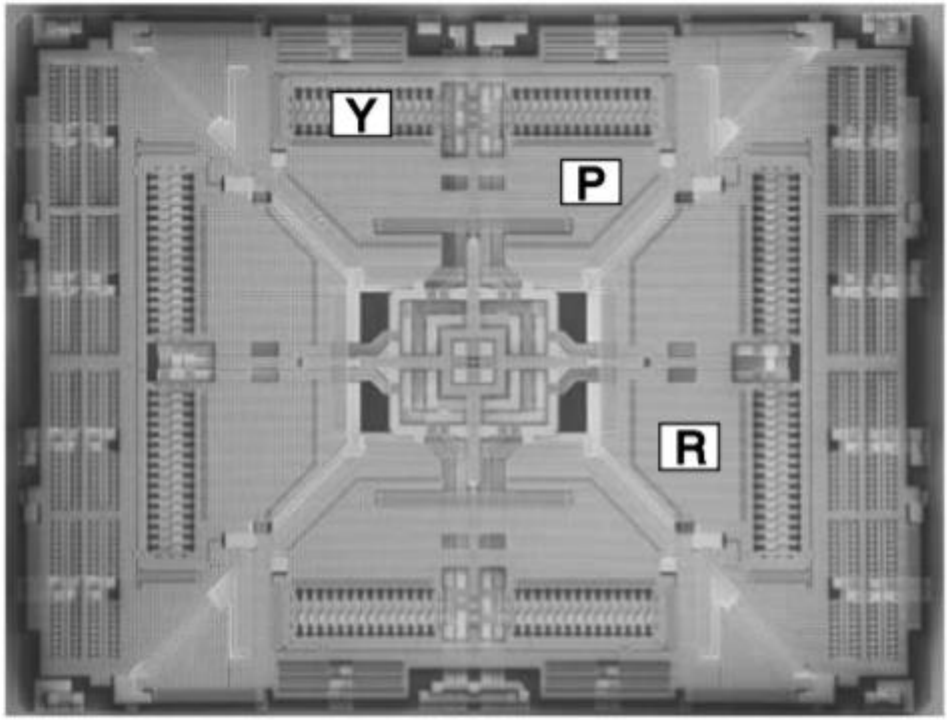


A scanning electron microscope (SEM) image of an inertial MEMS accelerometer. Polysilicon fingers are suspended in a depressurized cavity to enable movement and electrical capacitance proportional to acceleration is measured by adjacent signal conditioning electronics.

Courtesy of analog.com

Capacitive MEMS - fabrication



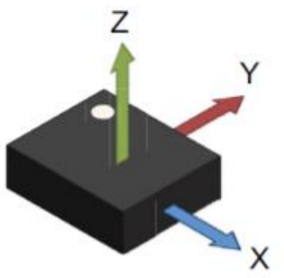


Y, P and R stand for the sensing masses for yaw, pitch and roll modes. The driving mass consists of 4 parts M1, M2, M3 and M4. When an angular rate is applied on the Z-axis, due to the Coriolis effect, M2 and M4 will move in the same horizontal plane in opposite directions as shown by the red and yellow arrows. When an angular rate is applied on the X-axis, then M1 and M3 will move up and down out of the plane due to the Coriolis effect. When an angular rate is applied to the Y-axis, then M2 and M4 will move up and down out of the plane.

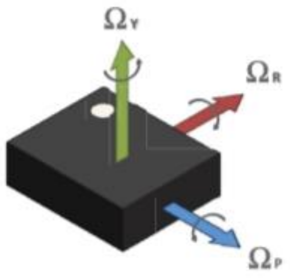
Courtesy of STMicroelectronics

LSM6DSL

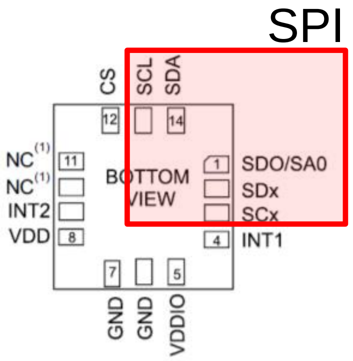
iNEMO 6DoF inertial measurement unit (IMU)

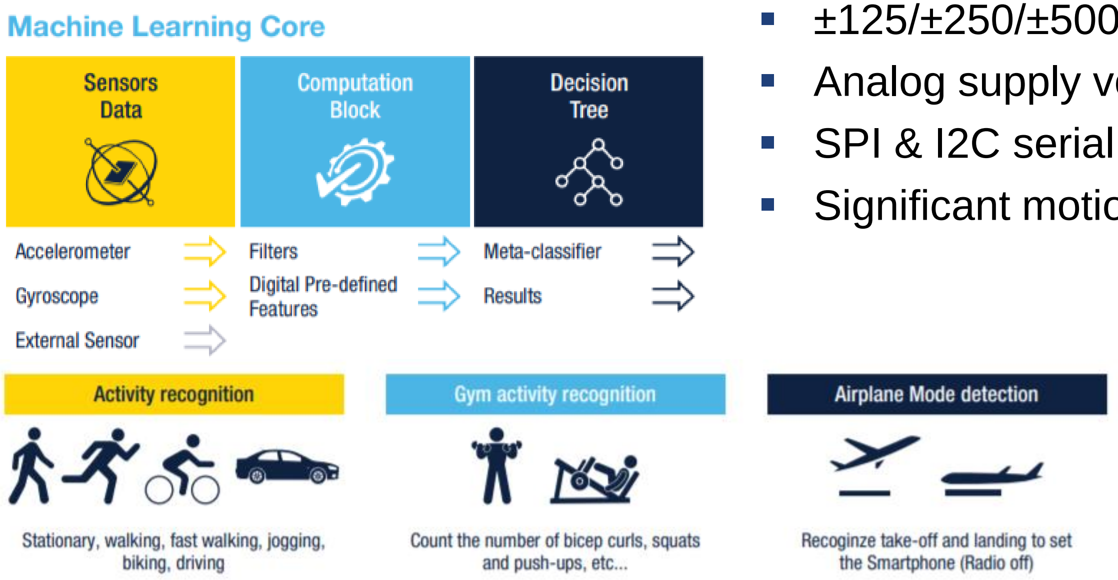


Direction of detectable acceleration (top view)



Direction of detectable angular rate (top view)

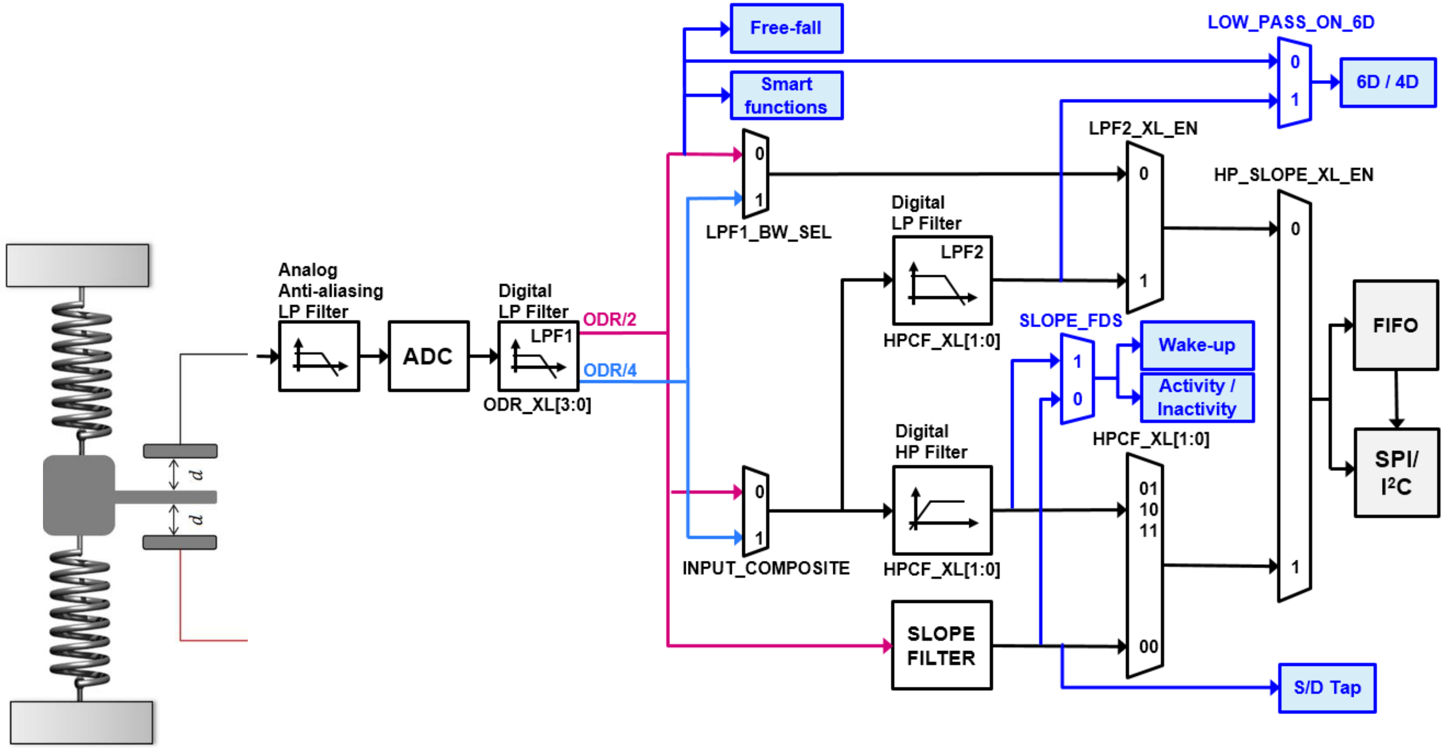




- Main features:
  - Power consumption: 0.4 mA in normal mode, 0.7 mA in high performance
  - ±2/±4/±8/±16 g full scale
  - ±125/±250/±500/±1000/±2000 dps full scale
  - Analog supply voltage: 1.71 V to 3.6 V
  - SPI & I2C serial interface
  - Significant motion and tilt function

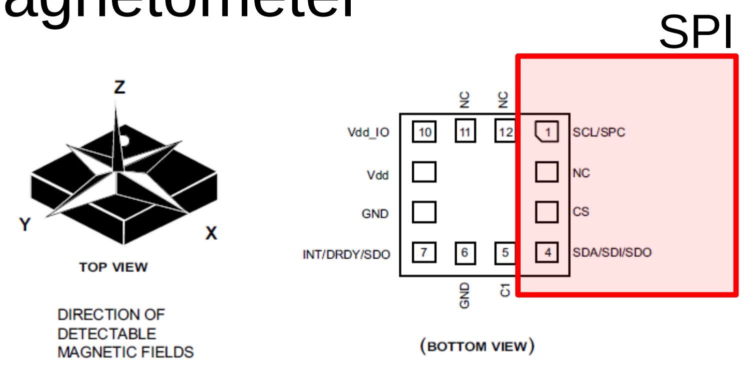
LSM6DSL main blocks

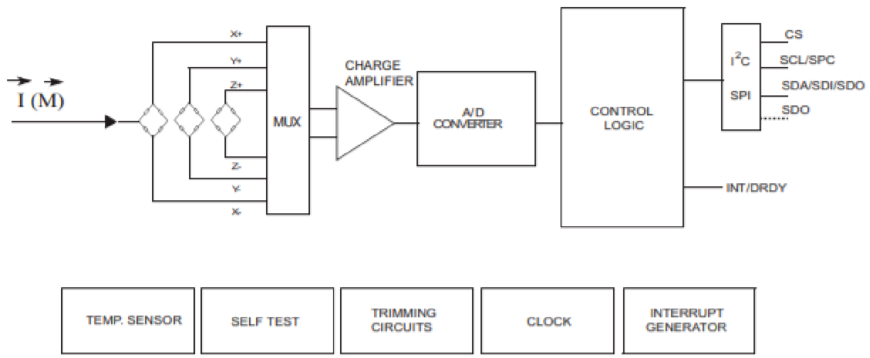
The accelerometer sampling chain is represented by a cascade of four main blocks: an analog anti-aliasing lowpass filter, an ADC converter, a digital low-pass filter and the composite group of digital filters. The analog signal coming from the mechanical parts is filtered by an analog low-pass anti-aliasing filter before being converted by the ADC



LIS2MDL

Digital output magnetic sensor: ultra-low-power, high-performance 3-axis magnetometer



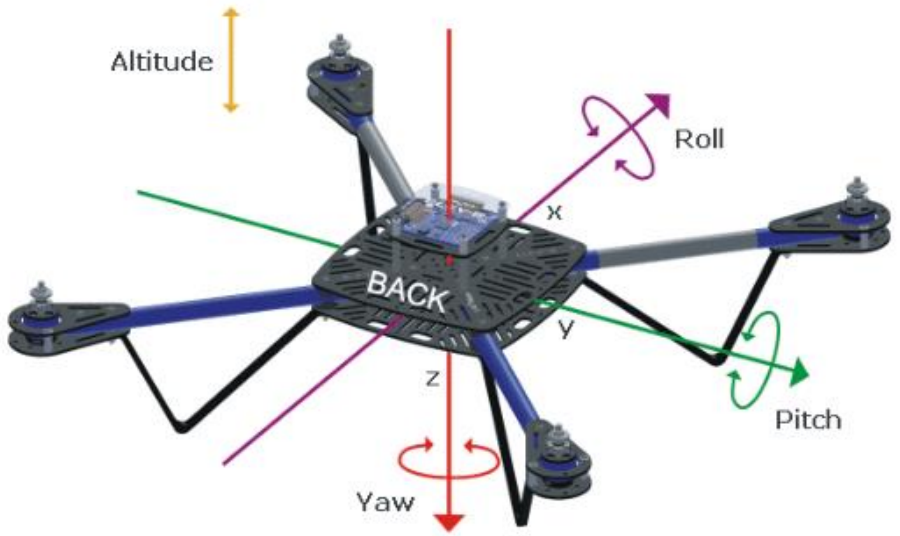


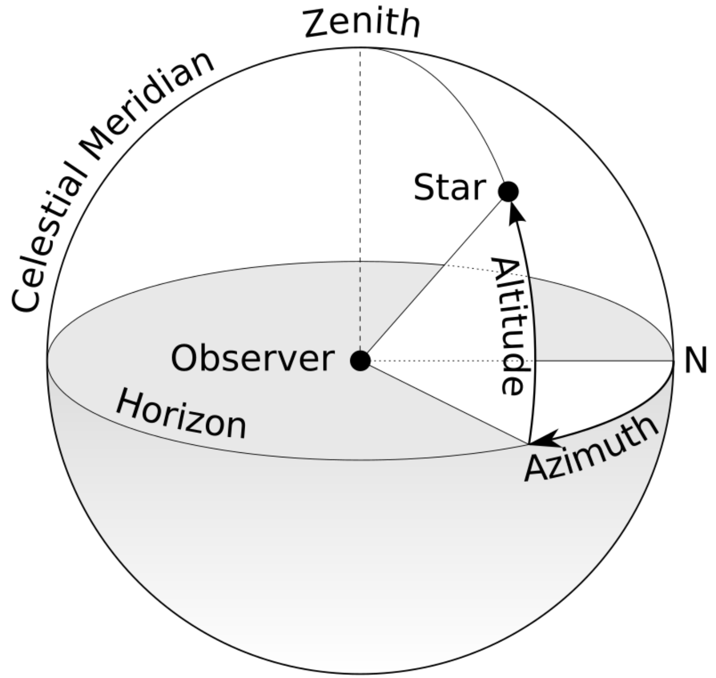
- Main features:
  - 3 magnetic field channels
  - Current consumption 200 µA
  - ±50 gauss dynamic range
  - Analog supply voltage: 1.71 V to 3.6 V
  - SPI & I2C serial interface
  - Significant motion and tilt function

Azimut - eCompass

The **azimuth** is an angular measurement in a spherical coordinate system. The vector from an observer (origin) to a point of interest is projected perpendicularly onto a reference plane; the angle between the projected vector and a reference vector on the reference plane is called the azimuth.

Used by the drone to calculate the flying direction (**Yaw**)



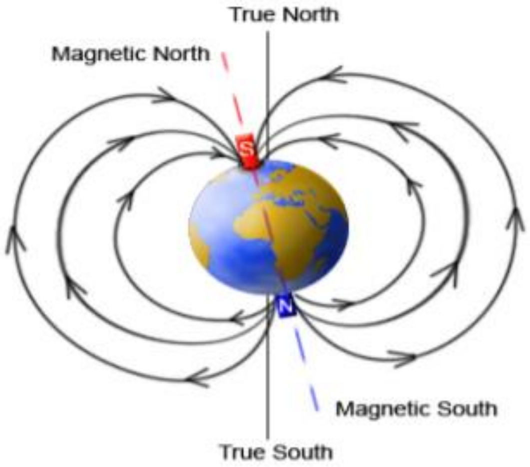


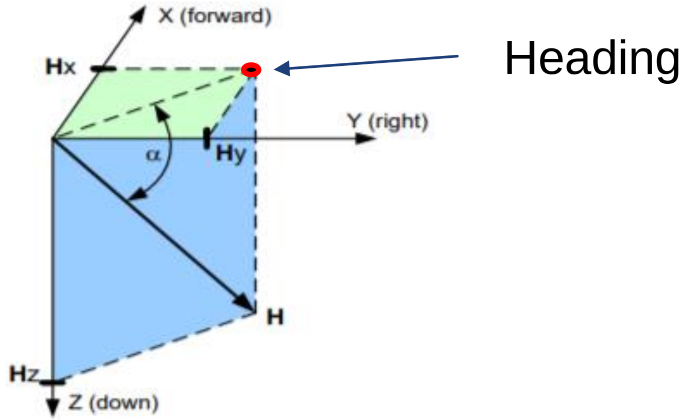
Magnetic Compass with Tilt Compensation

A common approach used to estimate the **heading** is to measure two orthogonal components of the magnetic vector (**Hx** and **Hy**)

$$Heading = arctan\left(\frac{H_y}{H_x}\right)$$

The angle between the magnetic vector and the horizontal plane, XY, is called the dip or inclination angle (**α**). This angle is mainly dependent on the geographical latitude.





This equation is correct only when the magnetic sensor is precisely levelled. If some tilt is present, the measured values Hx and Hy change and the resultant heading **is not exact**.

if the inclination angle is about 60° and the compass is tilted at 5° to level, the azimuth measurement error will be up to 8°

Magnetic Compass with Tilt Compensation

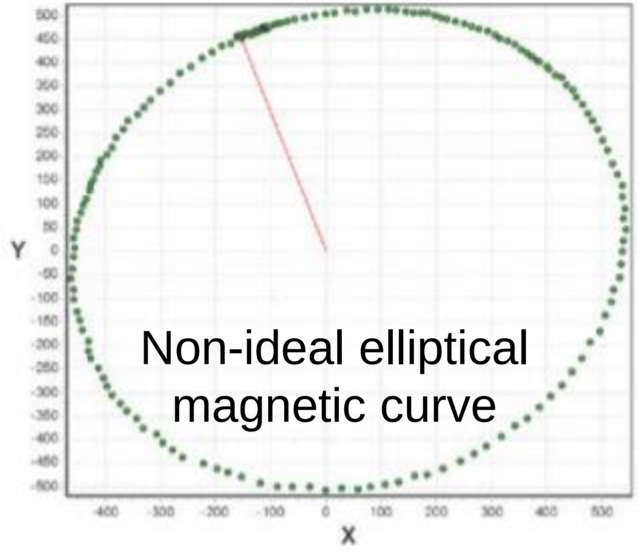
INTEGRATION BETWEEN ACCELEROMETER AND MAGNETOMETER

eCompass Tilt Compensation

rotation matrix to compensate the tilted platform

+

Compass calibration

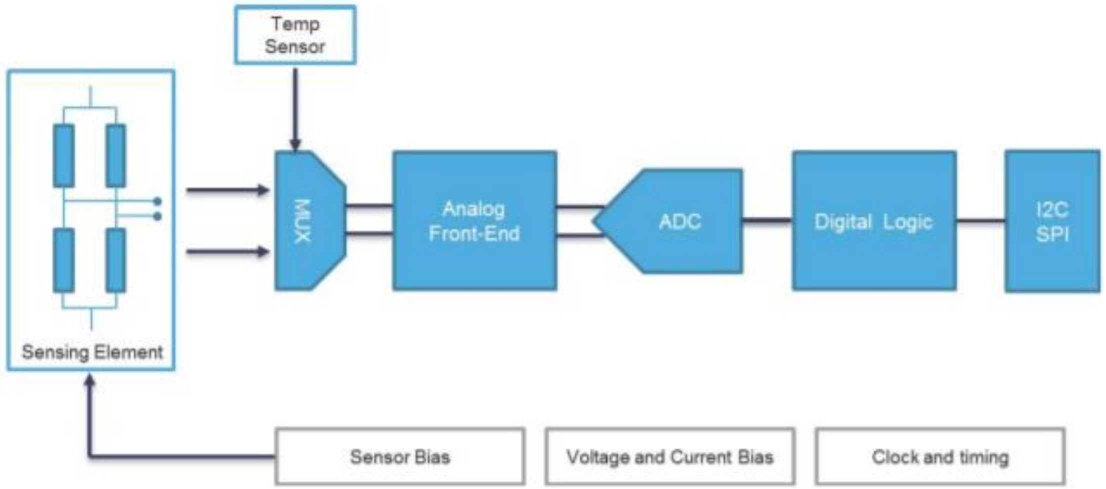


Suggested readings (Polybox):

1. Sensing Magnetic Compass with Tilt Compensation.pdf
2. Electronic Compass Tilt Compensation and Calibration.pdf
3. Computing tilt measurement and tilt-compensated eCompass.pdf

LPS22HD

The LPS22HD is an ultra-compact **piezoresistive** absolute pressure sensor which functions as a digital output barometer.



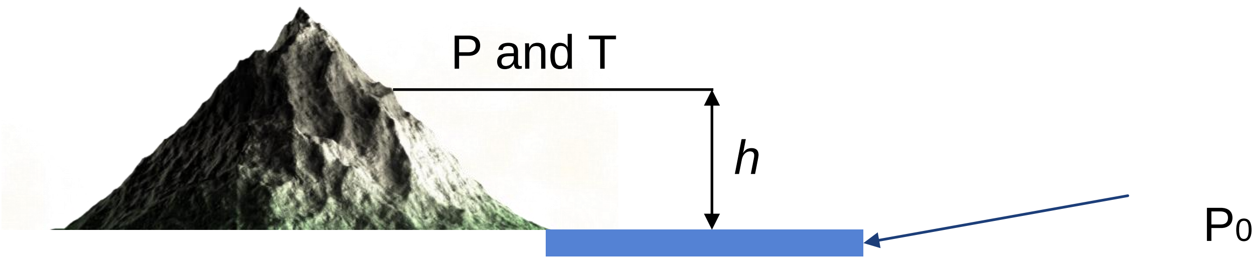
- Main features:
  - 260 to 1260 hPa absolute pressure range
  - Current consumption 3 µA
  - 24-bit precision, accuracy 0.1 hPa, **~ 20 cm**
  - Analog supply voltage: 1.71 V to 3.6 V
  - SPI & I2C serial interface
  - ODR from 1 Hz to 75 Hz

Hypsometric formula

Calculates the altitude at the present location from the atmospheric pressure and temperature, and sea-level pressure.

$$h = \frac{\left(\left(\frac{P_0}{P}\right)^{\frac{1}{5.257}} - 1\right)(T + 273.25)}{0.0065}$$

- Valid below 11 km
- Temperature rate compensation
- Sea-level reference
- No-intrinsic calibration



- P_0 sea-level pressure [hPa] = 1013.25
- P atmospheric pressure [hPa]
- Temperature T [°C]
- h altitude [m]