Articoli references notes

# A Miniaturised, Fully Integrated NDIR CO2 Sensor On-Chip

* CO2 long term exposure by humans can affect the organism: cognitive abilities, bones demineralization and kidney calcification
* Ventilation air quality dictated by sensors
* Also a low cost sensor can sometimes make the effect (electromechanical sensors and NDIR ones)
* ELECTROMECHANICAL SENSORS: measurement of gas concentration by measuring the change in electrical properties
  + Resistance
  + Capacitance
  + Electric potential

Induced by the absorption of a gas.

* ELECTROMECHANICAL SENSORS:
  + advantageous because of easy of fabrication
  + low cost
  + high sensitivity to wide range of compounds
* ELECTROMECHANICAL SENSORS: poor long-term stability and cross-sensitivity to other gases = less attractive in the CO2 sensor market
* NDIR CO2: superior long-term stability and high gas specificity: significant absorption strength of CO2 in mid-IR region.
* NDIR CO2: very high absorption coefficient of CO2 in mid-infrared: a path length of few centimeters is sufficient to detect small changes in CO2 concentration
* NDIR CO2: the 83% of the total advanced CO2 sensors are based on the NDIR technique
* NDIR CO2: are also bulky: long (several cm) interaction length is required for achieving ppm detection
* NDIR CO2: the cost is high because they are typically based on discrete co-assembled optical elements… in last years price and dimension changed giving them more interest in industry and academia
* NDIR CO2: decreasing the cost depending on the design of miniaturized ones, which can use a multi-pass cell or an optical cavity with various shapes.
* NDIR CO2: pre-concentrator coatings can be employed to effectively amplify the gas concentration in the vicinity of the optical field  the optical path length can be reduced
* NDIR CO2: broadband infrared source + reflective gas tube + 2 optical detectors that are sensing and reference … the overall mechanism allows for autocalibration
* NDIR CO2: mid-IR led has a relatively narrow emission spectrum such that the need for optical filters can be eliminated without introducing notable cross-sensitivity by other gases
* NDIR CO2: why integrating cylinder? Multiple reflections can be experienced by the sensor before reaching the detector and thus a long effective path length on a small sensor footprint
* NDIR CO2: the response time of the sensor usually depends on the level to measure in ambient (usually few hundreds of ppm), usually it could be long but there’s not a big problem for applications such as air quality monitoring
* NDIR sensor accuracy: can vary depending on the presence of water molecules (it depends on relative humidity RH in the air): the water vapor has a broad absorption in the infrared wavelength range that can be significant for some specific wavelength depending on the spectral overlap between the absorption band of the target gas and the water molecules
  + Different water interference characteristics
  + It is negligible in common ranges
* All sensors: characteristics curves used for determine some points of concentrations dependent on T and RH (that are made variable during the measurement)
* Stability of a sensor: deals with the degree to which the sensor’s characteristics remain constant over time. The drift can be attributed to factors such as temperature fluctuations and component aging.
* Sometimes it could happen that the reference signal and the sensing signal change in opposite directions during this drift (led photodiode characteristics): strong correlation between the ambient temperature and the sensing signal whilst the worse correlation of the reference channel is not yet fully understood

# Quartz-Enhanced Photoacoustic Spectroscopy: A Review

* Quantitative and qualitative gas sensors categorization: analytical sensors (gas-chromatography and spectrometry), electrochemical, semiconductor and laser optical absorption sensors  they are mainly classified based on the physical mechanism used
* Analytical techniques: no real time response, costly, invasive and with a large spatial footprint
* Electromechanical: relatively specific to individual gases, with usable resolution of less than one part per million of gas concentration and operate with a very small amount of current
  + Suited for portable, battery powered instruments
  + But there’s the influence of hysteresis and water humidity
  + Slow time response: fluctuations of gas and power up can make minutes for the sensor to reach 90% of tis final output value
* Laser absorption spectroscopy (LAS): response is < 1s and they are faster and part per quadrillion detection sensitivity (small traces detection)
  + Molecular absorption principle: transitions that an electromagnetic wave cause in a chemical species
  + If a molecule is irradiated by infrared light, it is excited to a rotational vibrational energy level manifold.
* Semiconductor lasers as light source for sensors: limited by the available optical power
* Mid-IR and previous analyzed NDIR are the common choice
  + Quantum cascade lasers (QCLs)
  + Interband cascade lasers (ICLs)

High output power

* LAS-based techniques: excellent sensitivity and selectivity and long effective optical pathlengths, compactness, mechanical stability, versability and cost effectiveness
* cavity ring down spectroscopy (CRDS): optical cavity with low loss and high reflectivity (>99.9%): long optical path of up to several kilometers. Multiple reflections occur on mirrors after a pulse
* Cavity enhanced absorption spectroscopy (CEAS): it is a modification of CRDS: radiation is injected at a very small angle respect to the cavity axes. It results in the formation of a dense structure of a weak optical axial modes that makes the system more robust against instability in both cavity and laser spectrum.
* Cavity output spectroscopy (ICOS): similar to CEAS: measurement procedure is the comparison between signal amplitude both at input and output of cavity.
* Photo-acoustic spectroscopy (PAS): one of the most robust and sensitive trace-gas optical detection techniques. Extremely high detection sensitivities with compact and relatively low cost absorption detection module (ADM)
* PAS: is based on optical absorption procedure as ICOS, CEAS and CRDS as well. It differs in the physical phenomenon used for the detection of the absorption signal.
* PAS: .when light at a specific wavelength is absorbed by the gas sample, excited molecules will subsequently relax to the ground state either through emission of photons or by the means of non-radiative process: it produces localized heating in the gas which in turn results in an increase of the local pressure. If incident light intensity is modulated, generation of thermal energy in the sample will also be periodic and a pressure wave (a sound wave) will be produced having the same frequency of the light modulation.
* PAS signal: can be amplified by tuning the modulation frequency to one of the acoustic resonances of the gas sample cell.
* PAS: in this strategy no optical detector is required and the resulting sound waves can be detected by a commercial hearing aid microphone.
* PAS: In the adopted logic of calculus, there’s a linear relationship between the sample concentration and the photo-acoustic signal.
* PAS: also the noise can affect measurements as well. This noise is assumed to be independent from the optical excitation.
* PAS: the same technology of lasers and optical parameters oscillators of the near-IR and QCLs in mid-IR have been successfully applied to PAS
* PAS and 3 main noise sources:
  + caused by the radiation that is incident upon the walls of the PAS absorption cell
  + non selective absorption of the gas cell window
  + external acoustic noise
* PAS and signal to noise ratio improvement (SNR): different designs for PAS cells have been proposed and implemented including a resonant cell with acoustic buffers, windowless and a differential cell
* PAS and differential cell: it includes 2 acoustic resonators equipped with microphones having same responsivity at the resonance frequency of the cell. Since the laser light excites only one of the 2 resonators, difference between the two signals removes noise components that are coherent in both resonators
* PAS and trace gas sensing applications: atmospheric chemistry, volcanic activity, agriculture, industrial processes, workplace surveillance and medical diagnostics.
* PAS and other substances than CO2: nitric oxide (NO) from vehicle exhaust emissions, medicine and drug diffusion rates in skin and to detect trace concentrations of disease biomarkers such as ethylene (C2H4), ethane (C2H6) and pentane (C5H12) which are emitted by UV-exposed skin. NH3 for monitoring respiratory emission from cockroaches as well as detection of intake prohibited substances by athletes
* PAS low cost detectors on market: smoke detectors, toxic monitoring gas and oil sensors for monitoring hydrocarbons in water.
* Quartz-enhanced photoacoustic spectroscopy (QEPAS): alternative approach to photo acoustic detection of trace gas utilizing a quartz tuning fork (QTF) as sharply acoustic transducer. It detects weak photo acoustic excitation and allowing the use of extremely small volumes
* QEPAS: restrictions imposed on the gas cell by the acoustic resonance conditions are removed.
* QEPAS: the quartz crystal is low-loss and low-cost piezoelectric material
* QEPAS: QTF is a quadrupole which provides good environmental noise immunity
* QEPAS: the excitation laser beam passes through the gap between the prongs without touching them
* QEPAS: if a rotational-vibrational state is excited, a collision-induced vibrational to translation (V-T) relaxation follows with a time constant that for a particular molecule is dependent on the presence of other molecules and intermolecular interactions. This process is more sensitive for this technology compared with the PAS one.
* QEPAS: wavelength of laser is varied by changing the driving current when the temperature of the laser is fixed. DFB QCL is the light source applied in this case, EC-QCL: is used when both temperature and current are fixed and the optical frequency can be scanned by applying a modulated voltage to a piezoelectric translator attached to the diffraction grating element of the EC-QCL.
* QEPAS: sometimes wavelength modulation (WM) is employed to improve the QEPAS SNR, minimizing external acoustic noise for a QEPAS based sensor system (those are the WM QEPAS)
* QEPAS (WM): the wm description is based on an intensity representation of an optical wave, so that only the absorption of the sample is considered and dispersion effects due to the sample can be neglected. Those results are obtained after various modulation steps.
* Amplitude Modulation (AM) QEPAS: introduced because the vibrational spectra of most molecules consisting of more than five atoms are so dense that infrared absorption spectra consist in 100-200 cm-1 broad bands and spectroscopic identification of these species requires laser excitation sources with a wide spectral coverage
* Amplitude Modulation (AM) QEPAS: on a laser radiation. This is operated at f0 by means of square wave amplitude current modulation and QEPAS signals are detected by a lock-in amplifier at the same f0 frequency.
* Amplitude Modulation (AM) QEPAS are not background free. Residual radiation absorbed inside the gas cell produce a sound at the TF resonant frequency thus generating a coherent background.
* Amplitude Modulation (AM) QEPAS: background subtraction can be applied because generally it is stable over several hours. This is done also by the mean of normalization of signal and background for every spectral point and in post processing.
* Various architectures have been designed and implemented for the QEPAS realization (OB On-Beam QEPAS and Off-Beam QEPAS, which are characterized respectively by a perpendicular and parallel laser tube with respect to the QTF plane probing the acoustic vibration excited in the gas contained inside the ADM)
* And the fiber coupled QCL-QEPAS:
* QEPAS ranges of application in conclusion: they are demonstrated to be effective and mature for numerous real-world applications:
  + environment monitoring (CO, CO2, CH4, H2CO, C2HF, C2HF5, N2O, NO2)
  + industrial emission measurements such as at combustion sites and gas pipelines (Hcl, CO2, CH4, CO, Nox, CH2O)
  + urban emission coming from automobile traffic (Nox, Sox)
  + rural emission such as a horticultural greenhouse and fruit storage (C2H6, C2H4, CH4, N2o)
  + control for manufacturing processes (SF6, Hcl)
  + detection of medically important molecules (NO, CO, NH3, C2H6, H2S)
  + toxic gases (CH2o, Hcl, HCN, N2H4 etc.)
  + planetary science (H2O, CH4, CO, CO2, N2H4, C2H2)
  + environmental monitoring
* QEPAS have been for instance installed in a mobile laboratory to perform atmospheric CH4 and N2O detection near 2 urban landfill sites located in Houston: they recorded concentration values in a very good agreement (<5% difference) with those measured by the Aerodyne Research Inc. “QCL mini monitor” multi-pass optical sensor having a CH4 detection sensitivity of 300 ppt and N2O detection sensitivity of 60ppt, both in 1 s which demostrates the precision, stability and applicability of the QEPAS sensing technique.

# Analytical determination of load resistance value for MQ-series gas sensors: MQ-6 as case study

* MQ sensors: high sensitivity and low cost
* MQ sensors: process of load resistance selection is a matter not well studied
* MQ sensors: parametrical investigation of load resistance and power dissipation on LPG
* MQ sensors: metal oxide (MOX) semi-conductor gas sensors: wide applications in gas concentration sensing and detection because of their high sensitivity and low cost
* MOX sensors: consists of a micro AL2O3 ceramic tube, a sensitive layer of tin dioxide (SnO2) and Nickel-Chromium alloys which serve as a heater coil
* MOX sensors: 6 pins, 4 of which are for signal and electrodes, remaining 2 for heating coils
* MOX sensors: tin dioxide (SnO2) semiconductor is the sensor gas sensitive portion with a low conductivity in clean air
* MOX sensors and their principle: based on variation of their resistance when they come in contact with the gas to be sensed. Magnitude of the sensor output signal depends on the concentration and nature of the gas and the type of metal oxide used for the sensor sensing surface
* MOX sensors: made up of 2 elements, namely the heating and the sensing elements. These elements are normally powered independently either form the same or separate voltage surface
* MOX sensors: heater voltage will allow it to generate the required heat for maintaining the sensor in the active state while sensor voltage will allow the sensor to convert the sensed gas concentration to an appropriate voltage level across the load resistor connected in series with the sensing element
* MOX sensors: because of the characteristic of the sensing element a simple electrical equivalent circuit can be used to convert the sensed gas concentration to a corresponding signal usually voltage across the load resistor
* MOX sensors and their calibration: those parameters must be known:
  + Vcc: sensor calibration voltage
  + VRL: sensor electrical equivalent circuit output voltage
  + R0: sensor resistance for referent gas concentration and environmental conditions (temperature and humidity)
  + RS: sensor resistance
  + RL: load resistance
* MOX sensors and R0: this resistance is not explicitly given in the datasheet and has to be determined experimentally
* MOX sensors and R0: it must be determined for every sensor to be used because it is practically impossible to have similar gas sensors with the same value of R0
* MOX sensors: the value of R0 affects the results in general. This is because it is impossible to secure the reproducibility and stability of this class of sensors as a result of the impossibility of keeping the consistency of the manufacturing environment. It assures some variation in the sensor behaviour from one sensor to another and from one production batch to another.
* MOX sensors: after the R0 has been determined, it is possible to determine the sensor resistance at different gas concentrations value for various gases and different conditions (temperature and relative humidity (RH))
* MOX sensors: the sensor circuit sensitivity and the sensor power dissipation are both functions of the load resistance.
* MOX sensor, example of the MQ2 sensor: the 20 kohm load resistance as given in the sensor datasheet is used
* MOX sensor MQ sensors and selection of load resistance: it should be selected in such a way to optimize the alarm threshold value and keep the sensor power dissipation below the maximum allowable value. In some datasheet the manufacturer provides data on the value of load resistance to be used so that the resolution would be sufficient around the alarm point
* MOX sensor: the load resistance should be selected wisely: a lower value will result in less sensitivity while a higher value will give less accuracy

# Application of MQ-Sensors to Indoor Air Quality Monitoring in Lab based on IoT

* Indoor air quality: refers to the building’s residents’ air quality. State of bad or good content inside a building (residence, hospital, lab) that can impact the occupants’ health, comfort, performance and physical reactions
* Better indoor air quality is known as chemical (gaze) or biological (bacteria and fungi) or physical contaminants such as dust and the concentration of air pollutants in building does not exceed the environmental threshold.
* 2 categories for air pollutants: primary and secondary categories. Primary: polluting elements directly generated by air contamination. An example is CO because is produced by combustion
* Secondary contaminants: created when main pollutants react in the atmosphere: photochemical smog produces ozone, which is one example of secondary contamination
* IoT technology: can access information about the availability of indoor air quality monitoring in lab (IAQML) in real time by using Wireless Sensor Network (WSN) based monitoring systems.

A new approach to realize UART

* UART protocol: it is asynchronous as the name reports
* UART: can be customized for be synchronized, talking about some other protocols
* UART protocol: usually includes start bit, parity bit, stop bit and idle state
  + START BIT: the beginning of transmission: when the transmitter sends a character data, a logic “0” signal is firstly send: start bit, time width is a baud rate clock cycle
* UART protocol: data bits after the start can be usually from 5 to 8
* UART protocol: data bit from least significant bit (LSB) begin to sent
* UART protocol: data bits can be parity bit (odd or even parity) as well as no parity bit
* UART protocol: after the parity bit some stop bits can occur: they are logic “1” signal containing 1, 1.5 or 2 bits. Those bits are the end of data
* UART protocol: idle state is a logic “1”. This data format is adopted by the start and stop bit to achieve character synchronization
* UART: one internal configuration register: the user here can set the data bits, whether there is parity bit as well as the type of parity and stop bits.
* UART includes the transmitter, the receiver and a baud rate generator. Transmitter performs parallel-to-serial conversion
* UART baud rate generator generates the required clock signal
* UART serial transmitter section consists of an 8-bit transmitter hold register (THR) and transmitter shift register (TSR)
* UART: parallel data is stored in THR which received form the CPU, then it is transferred to the shift register and send out in the serial data, at same time parity bit is generated and transmitted by TSR, when the whole character is removed from TSR, CPU Interrupt signal is generated
* UART: serial receiver section also contains an 8-bit receiver buffer register (RBR) and receiver shift register (RSR)
* UART: serial data received is stored in the RSR and when it receives the whole character, automatically sent to RBR status register will be set and generate CPU interrupt signal
* UART: parallel data will be transmitted to the CPU in the read command of the CPU, serial-to-parallel conversion is performed
* UART: data latches four parts are comprised in the architecture, along with the already mentioned elements
* UART: data is latched by flip-latch and the input and output data is controlled through the door control and output enable pins
* UART: when the door control signal is effectively, the data is latched into the data device and when output enable signal is effectively, data is output
* UART: the baud rate clock of which the baud rate of 16 times of baud rate block frequency are generated by the baud rate generator
* UART: ~~transmitter~~ performs parallel-to-serial conversion and the receiver performs seral-to-parallel conversions which are realized by state machine
* ~~UART transmitter: realized by a state machine of 6 states. When the UART is reset by the reset pin, the transmitter will be reset to start state, waiting until the start bit will be asserted.~~
* ~~UART: start bit is asserted as soon as the THR is not empty~~
* ~~UART: once a low SOUT (start bit) is asserted, the FSM will switch to shift state~~
* ~~UART: when the FSM is in shift state, it is simply waiting for the last (most significant) data bit to be shifted out~~
* ~~UART: after the last data bit is shifted out, the FSM will switch to parity state if parity is enabled, otherwise it is switched to stop\_1bit state. When switching to this state the last data bit is still in transmission~~
* ~~UART: when the transmission is completed, the FSM will assert the parity bit~~
* UART: once parity bit is asserted, FSM switch to stop\_1bit and this is independently to the stop bit configured to be 1, 1.5, or 2 bits.
  + The FSM is here waiting for a baud rate clock cycle and then assert the stop bit(s)
  + 1 STOP BIT - UART: for 1 stop bit, the FSM switches back to start state and waits to assert the start bit of another frame
  + 1.5 STOP BIT: the switch is for stop\_half bit state and stays there for just half the baud clock cycle before switching to start state
  + 2 STOP BIT: it switched to stop\_2bit state then switches back to start state.
* UART: the stop bit(s) are asserted at the time when the FSM is leaving the stop\_1\_bit state
* UART: stop\_halfbit: is for 5 bit data bits with 1.5 stop bit. The FSM will stay in this state for only half baud clock cycle and then switch to start state
* UART stop\_2bit: the first stop bit is in transmission, there’s the waiting for a baud clock cycle then asserts the second stop bit and switches to the start state
* UART: stop bit is the ending of data, when stop bit is asserted an interrupt signal will be sent and the pin will be low for one baud rate clock cycle

# Design and Simulation of UART Serial Communication Module Based on VHDL

* UART – universal asynchronous receiver transmitter: kind of serial communication protocol
* UART: used for short-distance, low speed low-cost data exchange between computer and peripheral
* UART: 3 kernel modules are included: the baud rate generator, receiver and transmitter
* UART: asynchronous serial communication has advantages of less transmission line, high reliability and long transmission distance 🡪 this makes it widely used in data exchange between computer and peripherals. This is also implemented by UART
* UART: full duplex communication in serial link: widely used in data communication and control system
* UART: often implemented in FPGA using VHDL
* UART (basic): only 2 lines communication: RXD and TXD to complete full-duplex data communication
* UART TXD: it is the transmit side, the output of UART
* UART RXD: receiver side, the input of UART
* UART: 2 states in the signal line, using logic 1 (high) and logic 0 (low)
* UART: when the transmitter is idle, the data line is in the high logic state, when a word is given to the UART for asynchronous transmission, a bit called “start bit” is added to the beginning of each word that is to be transmitted
* UART: start bit: used to alert the receiver that a word of data is about to be sent and force the clock in the receiver into synchronization with the clock in the transmitter
* UART: the 2 clocks of transmitter and receiver must not be a frequency drift higher than 10% during the transmission of the remaining bits in the word.
* UART: after the start bit, the individual data bits of the word are sent with the least significant bit (LSB) being sent first
* UART: each bit in the transmission is transmitted for exactly the same amount of time as all of the other bits
  + The receiver looks at the wire at approximately halfway through the period assigned to each bit to determine if the bit is a 1 or 0
  + For example, if it takes 2 seconds to send each bit, the receiver will examine the signal to determine if it is a 1 or a 0 after one second has passed, then will wait 2 seconds and then examine the value of the next bit and so on
* UART: after the entire data word has been sent, the transmitter may add a parity bit that the transmitter generates, this will be used by the receiver to perform simple error checking
* UART: at least one stop bit is sent by the transmitter
* UART: when the receiver has received all of the bits in the data word, it may check the parity bits (both sender and receiver must agree on whether a parity bit is to be used)
* UART: then the receiver looks for the stop bit, if stop bit does not appear when it is supposed to, the UART consider the entire word to be garbled and will report a framing error to the host processor when the data word is read
* UART and usual cause of a framing error: sender and receiver clocks were not running at same speed or that the signal has been interrupted
* UART: in both correct reception or not cases, start parity and stop bits are automatically discards and not passed to the host
* UART: if another word is ready for transmission, the start bit for the new word can be sent as soon as the stop bit for the previous word has been sent. Because asynchronous data are self-synchronizing, if there’s no data to transmit the transmission line can be idle.

A diagram of a data

Description automatically generated

* Implementation of UART of the paper
  + Top down strategy with the UART serial communication divided in 3 sub-modules
    - Baud rate generator
    - Receiver module
    - Transmitter module
* UART baud rate generator: used to produce a local clock signal which is much higher than the baud rate to control the UART receive and transmit
* UART receiver module: receive the serial signal at RXD and convert them into parallel data
* UART transmit module: converts the bytes into serial bits according to the basic frame format and transmits those bits through TXD

A diagram of a person's relationship

Description automatically generated

* In the rest of the paper there’s the model of those 3 components
* In general programming VHDL for FPGA and UART is significant in the field of electronic design, where SOC technology has recently become increasingly mature, this design shows great significance.

# UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver / Transmitter

* UART : Device to device communication protocol
* UART : when properly configured it can work with many different types of serial protocols that involve transmitting and receiving data
* UART: serial communication protocols: data is transferred bit by bit using a single line or wire
* 2 way communication: two wires are used for successful serial data transfer
* UART and serial communications positive aspect: depending on the application they need less circuitry and wires which reduces the cost of implementation
* UART: it is used by embedded systems, microcontrollers, computers for a device-to-device hardware communication protocol
* UART: it uses only 2 wires for its transmitting and receiving ends
* UART: not fully optimized all the time. Properly implementation of frame protocol is commonly disregarded when using the UART module inside the microcontroller
* UART: hardware communication protocol that uses asynchronous serial communication with configurable speed and asynchronous means there is no clock signal to synchronize the output bits from the transmitting device going to the receiving end
* Example of 2 UART directly communicating with each other:

A blue and black line with a cross

Description automatically generated with medium confidence

* UART and 2 signals: transmitter and receiver: they are hence used to transmit and receive serial data intended for serial communication
* UART and data bus connection:

A diagram of a computer network

Description automatically generated

* Transmitting UART is connected to a controlling data bus that sends data in a parallel form
  + Data are then transmitted on the transmission line (wire) serially bit by bit to the receiving UART
  + The receiving UART in turns will convert the serial data into parallel for receiving device
* UART and baud rate: as for most serial communication, it needs to be set the same on both the transmitting and receiving device
* BAUD RATE: the rate at which information is transferred to a communication channel
* Serial port context: set baud rate will serve as the maximum number of bits per second to be transferred
* Summary UART

A close-up of a number

Description automatically generated

* UART: instead of a clock signal, the transmitter generates a bitstream based on its clock signal while the receiver is using its internal clock signal to sample the incoming data
* UART and point of synchronization: having the same baud rate at both sides
* UART: if not, there could be discrepancies during data handling. The difference of baud rate again is of 10%
* UART data transmission: in the form of packets
* UART and packet example:

A blue rectangular sign with black text

Description automatically generated

* UART and START BIT: when not transmitting the UART transmission line is at high voltage level, to start the transfer of data, the transmitting UART pulls the transmission line from high to low for 1 clock cycle
* UART and the receiving device: when it detects the high to low voltage transition, it begins reading the bits in the data frame at the frequency of the baud rate

A blue rectangle with black text

Description automatically generated

* UART and data frame: it contains the actual data being transferred. It can be 5 to 8 bits if parity bit is used, if not the data frame can be 9 bits. In most cases it is sent with the least significant bit first (LSB)

A blue rectangle with black text

Description automatically generated

* Parity describes the evenness or oddness of a number and it is used for receiving UART to tell if any data has changed during transmission.
* Bits can be changed by electromagnetic radiation, mismatched baud rate or long distance data transfer
* UART reads the data frame after the receiving and counts the number of bits with a value of 1 and checks if the total is even or odd number. If the parity bit is 0 (even number), the 1 or logic-high bit in the data frame should total to an even number. If the parity bit is a 1 (odd parity) the 1 bit or logic highs in the data frame should total to an odd number
* UART: when the parity bit matches the data, UART knows that the transmission is free of errors.

A blue rectangle with black text

Description automatically generated

* UART and stop bit: to signal the end of the data packet, sending UART drives the data transmission line from a low voltage to a high voltage for one (1) or two (2) bits duration
* STEPS OF UART TRANSMISSION
  + 1. The transmitting UART receives data in parallel from the data bus

A diagram of data flow

Description automatically generated

* + 2. The transmitting UART adds the start bit, parity bit and the stop bits to the data frame

A diagram of a device

Description automatically generated

* + 3. The entire packet is sent serially starting from start bit to stop bit from the transmitting UART to the receiving UART. The receiving UART samples data line at the preconfigured baud rate

A line drawing of numbers

Description automatically generated

* + 4. Receiving UART discards the start bit, parity bit and stop bit from the data frame

A diagram of a data frame

Description automatically generated

* + 5. The receiving UART converts the serial data back into parallel and transfers it to the data bus on the receiving end

A diagram of a data center

Description automatically generated

* UART and frame protocol: that is the introduction of added value for security and protection on each device (not fully introduced yet)
* UART and importance of added bits: for instance when 2 device use the same UART frame protocol, there are tendencies that when connecting to the same UART without checking the configuration, the device will be connected to different pins that may cause malfunctions in the system
* UART: on the other end the implementation of this secure stuff ensures security as well because of the need to parse the information received in alignment with the design frame protocol. Each frame protocol is specifically designed to be unique and secure
* UART: in designing a frame protocol, designers can set the desired headers and trailers, including CRC to different devices
* Sample UART frame protocol:

A blue rectangle with black text

Description automatically generated

* Header 1 (H1 is 0xAB) and Header 2 (H2 is 0xCD): it is a unique identifier that determines if we are communicating with the correct device
* Command (CMD) selection: it depends on the list of command designed to create the communication between 2 devices
* Data length (DL) per command: it will be based on the command chosen. We can maximize the length of data depending on the command chosen, so it can vary based on the selection. In that case, the data length can be adjusted
* Data n (varying data): the payload to be transferred from devices
* Trailer 1 (T1 is 0XE1) and trailer 2 (T2 is 0XE2): they are data that are added after the transmission is ended. Just like the header they can be uniquely identified
* Cycling redundancy checking (CRC Formula): it is an added error detecting mode to detect accidental changes to raw data. CRC value of the transmitting device must always be equal to the CRC computation on the receiver’s end
  + It is advisable to add security by implementing frame protocols for each UART device. The frame protocol needs identical configurations on both the transmitting and receiving devices
* UART OPERATIONS
  + FIRST: checking the data sheet interface of the device and getting the UART address in memory map for that device
  + SECOND: checking the specific details for the UART port, such as the operation mode, data bits length, the parity bit and stop bits
    - A sample port details in datasheet
  + THIRD: checking the UART operation details, including the baud rate computation, sample formula (it can varies depending on the device):
    - (getting parameters for the Raspberry and Arduino)
  + Some concepts for getting the BAUD RATE formula

A yellow text on a white background

Description automatically generated

* + - OSR (oversample rate): varies from 0 to 3
    - DIV (baud rate divider): 1 to 65535
    - M (DIVM fractional baud rate M): 1 to 3
    - N (DIVM fractional baud rate M): 0 to 2047
  + FOURTH: for the baud rate, make sure to check the peripheral clock (PCLK) to use
  + FIFTH: check the detailed registers for UART configuration. Taking a look at the parameters in computing the baud rate such as UART\_LCR2, UART\_DIV and UART\_FBR
  + SIXTH: under each register check the details and substitute the values to compute for the baud rate, then start implementing the UART
* Those operational concepts are important because of robust, quality-driven products development
* UART is one of the most commonly used hardware communication protocol, this knowledge can enable design flexibility in future designs
* UART and Use cases:
  + Debugging: early detection of system bugs is important during development. Adding UART can help in this scenario by capturing messages from the system
  + Manufacturing function-level tracing: logs are very important in manufacturing. They determine functionalities by alerting operators to what is happening on the manufacturing line
  + Customer or client updates: software updates are highly important. Having complete, dynamic hardware with update-capable software is important to having a complete system
  + Testing / verification: verifying products before they leave the manufacturing process helps deliver the best quality products possible to customers.

# A flexible Hardware architecture for slave device of I2C bus

* I2C: division between protocol level and application level for the paper
* I2C protocol level: implements the basic operations of I2C procol
* I2C application level: bases on protocol level aiming to the demand of various applications
* I2C inter integrated circuit: communication protocol very suitable for communications between on-board peripherals to transfer low/medium speed data
* I2C: widely used in various controllers, sensors and some other integrated circuits
* I2C: no central server to resolve data conflicts, that are hence resolved by the wired-and configurations of the serial data (SDA) signal and serial clock (SCL) signal
* I2C: in addition to SDA and SCL there’s also the acknowledgement signal sent by the receiver when every byte is transferred. This prevents the data loss of the SDA signal
* I2C protocol: defines the time sequence of writing or reading one or several data, every byte must be accurately specified
* I2C can be implemented in hardware with the utilization of FPGA
* I2C adopted protocols in the reference: protocol level signal level and interface level in the proposed methods
* I2C protocol level: signal and state defined by basic I2C protocol
* I2C: there could be the communication with many devices in the same bus network and each device is recognized by its unique address
* I2C: data is transmitted by 2 wire, bidirectional serial bus (SDA and SCL) of I2C bus network
* I2C: for the transfer start and end there’s proper signals (start and stop)

A black and white image of a rectangular object

Description automatically generated

* I2C: negative and positive edge of SDA in the high level of SCL represent the start signal and the stop signal respectively
* I2C: the signal of SDA would not reverse in the high level of SCL when the other signals are transmitted
* I2C: ACK: it is the acknowledgment bit and NACK must be transmitted after transmitting the data of one byte in the I2C protocol
* I2C messages: the address frame and the data frame
* I2C address frame: it is transmitted after the start signal, the higher seven bits of the address frame represent the address of slave device, the last one bit the operator (high level and low level represent the operator of read and write
* I2C data frames: follow the address frame closely. This phase will transmit several bytes continuously until the stop signal is coming and the meaning of every byte would be different in various applications

A diagram of a computer hardware architecture

Description automatically generated

* I2C designed hw architecture of protocol level for the paper in question: registers for SCL and SDA in which the signal is sampled by CLK signal,
* I2C: this CLK signal frequency needs not to change for different band rate, because much higher than the band rate of I2C bus
* I2C SCL and SDA sampled signals are buffered by registers, then the shift-register is used to obtain the posedge and negative edge of SDA signal and SCL signal
* I2C and SCL and SDA edge signals for computing additional modules:
  + Com Start: for generating the start signal depending on SCL and SDA
  + Com Stop: transfer ended
  + Sel Device: module used to compare the device address sent by master and the local device address to determine if this slave device is selected or not and that’s also the determination of the type of operation (reading operation or writing operation). Result of this module is seint to the modules of Tran. Data Recv Data and the application level for the following operations
  + Tran Data: when local slave device is selected and the reading operation is enabled. This module sends the data provided by application level to the master device. Data must be converted to the serial data before sending to the output buffer. State of this module should be provided to the application level for determining the next operation by application level
  + Recv data: enabled with the writing operation enabled. This module receives the data sent by the master. When the transfer is completed, the module sends a completing signal to the application level. This last one would obtain the received byte
  + Tran Data and Recv Data could be executed circularly until received stop signal
* I2C and designed HW architecture of application level: implementation based on protocol level and by the FSM:

A diagram of a computer system

Description automatically generated

* Idle state: local device listens to SDA and SCL signals until the device receives the start signal
* After the start signal there’s the judgement on the provided address with the following possibilities
  + Current Address Reading (CAR): read the data of several bytes from the beginning of the first address of register files. Operation would complete when the slave device receives a NACK signal and then the stop signal after it (both are sent by the master device)
  + Random sequential reading (RSR): read several bytes from the beginning of subaddress which is a 16 bits data. Subaddress is written by the master firstly, secondly the master will send the start signal and the local device address again. Then there’s the transfer of corresponding bytes one by one by the slave until it receives again the NACK and stop signals
  + Random sequential writing (RSW): master writes the data of one or several bytes from the beginning of subaddress to the slave device in this category. The master also writes
  + the subaddress to slave device firstly. The slave would receive the corresponding bytes sent by master until the stop signal is received
* I2C application protocol: there could some mechanisms of reset to idle state when the circuit enters the error state and over the time threshold. This module is also introduced for security reasons and for the I2C protection
* I2C slave device in bus: one of the most vital components in lots of chips
* I2C: while this protocol defines uniquely the basic operations: start, stop, writing several bytes and reading several bytes, various applications can customize these operations in sort of combinations of basic operations. Different applications for this needs different architecture and a flexible hw architecture is urgently needed

# Review of I2C protocol

* I2C: easy communication without data loss, with excellent speed compared to other protocols
* I2C: only 2 wires for communication needed, high speed communication and control in parameters with its development
* I2C: used in data surveillance for accuracy and efficiency with a design method developed in VHDL, simulated on MODELSIM or Xilinx and can be implemented on FPGA board.
* Serial interfaces: they allow processors to communicate without the need for shared memory and the problems they can create. There are several serial communication protocols like UART, CAN, USB, SPI, Inter IC
* Serial interfaces: USB, SPI and UARTS are all just one type to point type protocol. USB uses multiplexer to communicate with other devices
* Serial Interfaces: only I2C and CAN protocols use software addressing but ONLY i2c is very simple to design and easy to maintain.
* Serial interfaces examples: sensors communication with personal computers
* This table summarizes the prons and cons of the various protocols utilization:

A white table with black text

Description automatically generated

* I2C physical size and power requirement get reduce over the years, the main reason for it is more number of transistors can be integrated into smaller size and less number of interconnections wire present in between ICs can be possible

A diagram of a logic

Description automatically generated

* I2C bus: physically consists of two active wires and a ground connection. The 2 active wires are Serial Clock (SCL) and serial data (SDA).
* I2C wires: bidirectional half duplex in nature which carry information between the devices connected to the bus
* I2C connected devices: acknowledged by a unique address whether it is a microcontroller, LCD driver, memory or keyboard interface and can operate as either transmitter or receiver depending on its function.
* I2C device connection possibility: it is wide, devices can be added or removed easily which is very useful for low maintenance and control application in embedded system

A diagram of a system

Description automatically generated

* I2C proposed system:
  + I2C master top comprises of
    - Prescale register: used to reduce high frequency electrical signal to lower frequency by integer division
    - Command register
    - Status register: it is where the data comes in the origin and depending on this data it is issued the command register
    - Transmit register
    - Receive register

Those last 2 registers are for deciding whether to transmit or receive the data and this data is transmitted parallel to data I/O register

* + I2C master byte controller: byte command controller and data I/O shift register. The byte command controller is the heart of I2C communication traffic at the byte level and is a state machine that generates different states of I2C byte operations based on the command register bits. The data I/O shift register is a component which contains and deals with the data associated with the present I2C write and read transactions.
  + I2C master bit controller involves clock generator and a bit command controller. During the transmission, data is shifted bit by bit into the command bit controller and from there it is transferred to SDA. During reception data comes on SDA and then to bit controller

A diagram of a computer system

Description automatically generated

* I2C working – start and stop conditions: start condition issued on bus before the transaction.
* I2C start condition: acts as a signal to all connected ICs that something is about to be transmitted
* I2C stop condition: after a message is completed. This is a signal for all the devices on the bus that the bus is available again (idle).
* I2C: if a chip was accessed and has received data during the last transaction it will now process this information (if not already processed during the reception of the message)

A white rectangular object with black text

Description automatically generated

* I2C and transmitting a byte to a slave device: after start condition a byte can be transmitted to a slave by the master. The first byte identifies the slave on the bus (address) and the mode of operation. Meaning of all following bytes depends on slave:

A screenshot of a computer

Description automatically generated

* I2C and receiving a byte from a slave device: once the slave has been addressed and the slave has acknowledged this, a byte can be received from the slave if the R/W bit in the address was set to READ (1)

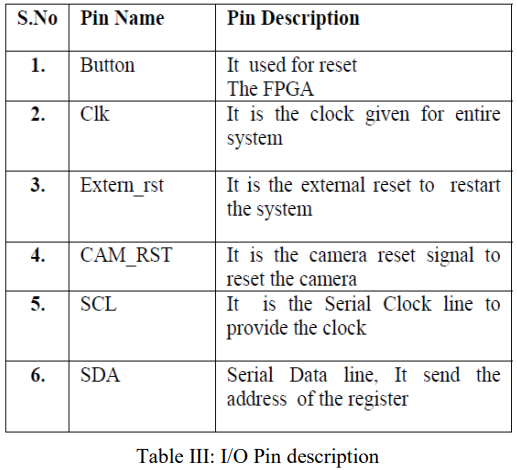
A blue line with black text

Description automatically generated

* Getting acknowledge ACK from a slave device: when address or data byte has been transmitted onto the bus it must be acknowledged by the slave(s): In case of an address if the address matches its own then that slave and only that slave will respond to the address with an ACK
* Giving acknowledge (ACK) from a slave device: upon reception of a byte from a slave the master must acknowledge this to the slave device. If there’s no data left to receive the master will send a not-acknowledge (NACK) signal and will stop the data transaction

A diagram of a computer system

Description automatically generated



* Features of I2C:
  + Multimaster serial ended computer bus
  + I2C has 2 wire bi-directional serial bus
  + It is a simple and efficient method for data exchange
  + I2C protocol have low bandwidth
  + It is a short distance protocol
* Advantages of I2C
  + Used for security sensitive applications like sensor connections, RFID, biometric devices etc
  + Common communication standards between microcontrollers and sensors
  + Each device is recognized by its unique address and can operates as either a transmitter or receiver, depending upon the function of the device
  + It provides enhance security system
  + Compatible with FPGA
* I2C in an ideal surveillance architecture: following characteristics: high performance, flexibility, easy upgradability, low development cost and migration path to lower cost as the application matures and volume ramps.

# Design and implementation of a high speed serial peripheral interface

* SPI: designed for both inter-chip and intra-chip low/medium speed data-stream transfers
* SPI: used for communicate between a microcontroller and other devices like external EEPROMs, DACs ADCs etc
* Ethernet, USB and SATA are meant for “outside the box communications” and data exchanges between the whole systems
* SPI: aptly suited for communication between integrated circuits for low/medium data transfer speed with on-board peripherals
* SPI: master and slaves, the master provides a clock signal to attain synchronization
* SPI: data transfer happens only when there is a clock manipulation, incoming data must be read before an attempt to transmit again, the clock is the SCK line and the data exchange between the devices are controlled by SCK clock line
* SPI: a device cannot be just a transmitter only device or a receiver only device
* SPI: working is essentially based on the contents of an eight bit serial shift register present in both the master and slave
* SPI transmission: takes place based on clock signal which is generated by the master: data are placed in the shift register both by the master and the slave
* SPI: with 8 clock pulses generated, bits contained in the master’s shift register are transferred by means of the MOSI line to the Slave’s shift register, the slave transfers its shift register content by means of the MISO signal line back to the master, having the content of the 2 shift registers to be exchanged
* SPI: the following signals for transmissions across the interface are used
  + SS: slave select line. When it goes high the corresponding slave device will be selected, slave selected line is used by master device to select which slave to initiate communication with the master
  + SCK: stands for serial clock. This signal synchronises the transmissions taking place across the bus
  + MOSI: serial single bit data line, it is generated by the master based on internally shifted value of the master data register
  + MISO: serial single bit data line, it is the line SPI slave communication with the master. It sends out the serially shifted out bits from the slave data register
* SPI functionalities of sub-block:
  + Data register and eight bit shift register: main part of the SPI system. When the SPI transfer takes place, a bit of data gets shifted out of the SPI master’s data register and subsequently the srial data which comes from the slave’s data register is serially shifted into the master’s data register
  + Time one SPI transmission completes, 16 clock cycles are used, the contents of master and slave will have been exchanged
* SPI control register: the user is provided preliminary control over the SPI operation. This register will be used to control the data transfer of the SPI. It includes enabling the SPI, configuring the SPI in master or slave mode, setting the sampling of data, configuring clock polarity etc.
* SPI: based on clock polarity (cpol) and clock phase (cpha) values 4 different modes of operation are supported in the SPI:
* SPI Baud rate register: it consist of series of divider stages. Eight bits in the SPI baud rate register determine the value with which the bus clock is divided
  + Provision of choices of end user ranged between 2 or 128 divisor.
  + The baud rate generator is active only if the SPI is operating in the master mode

A diagram of a computer program

Description automatically generated

* SPI: low or medium (n MHz to 10n MHz) data transfer rate depending on implementation
* SPI: offering of multiple transfer rates based on the SPI master baud rate
  + This last one can be programmed by the user
* SPI can support multi slave operation
  + Master and slave can be transmitter or receiver based on its mode of operation
* SPI: is capable of receiving and transmitting on both rising and falling edges of the clock idipendently
* SPI: also can be implemented in VHDL and using FPGA

# Design and verification serial peripheral interface (SPI) protocol for low power applications

* Serial communication: the process of sending data one bit at a time, sequentially, over a link
* Serial communication: it requires fewer interconnecting cables (wires/fibers) and hence occupies less space
* FPGA (field programmable gate arrays): also uses SPI to interface as a slave to a host as master sensor
* SPI: designed to replace parallel interfaces so that we don’t have to route parallel bus around PCB and provides hgiht speed data transfer between the devices
* SPI: it is used also in MIC, power PC or ARM and microcontrollers such as PIC and AVR, this protocol can run both as master or slave mode on this chips
* SPI: implemented also in intel (low pin count) LPC bus, known as Enhanced Serial Peripheral Interface bus (eSPI)
* SPI: in the enhanced version the aim is to allow reduction in the number of pins required on motherboard compared to systems using low pin count
* eSPI: more throughput than LPC reduce the working voltage to 1.8 volts to facilitate smaller chip manufacturing processes, allow enhanced SPI to share serial peripheral interface flash devices with the host
  + whereas on the other hand low pin count (LPC) bus did not allow firmware hubs to be used by the LPC peripherals and enhanced serial peripheral interface also allow system designers to trade off cost and performance
* eSPI: saving pins is done either by sharing eSPI with SPI devices or be separate from the SPI bus to allow more performance

# serial protocols compared

* many serial communication interfaces compete for use in embedded system
* serial interfaces: used in the need to interface with a PC, during development and/or in the field
* serial interfaces: present of more PC: they have a sort of serial bus interface available to connect peripherals
* serial interfaces: it is often easier to use in embedded systems, easier with respect to the use of the ISA or PCI expansion bus
* serial interfaces: low pin counts, they can be performed with just one I/O pin, compared to eight or more for parallel communications
* serial interface: many common embedded systems peripherals, such as analog-to-digital and digital-to-analog converters, LCDs, temperature sensors, support serial interface
* serial interface (bus): they provide inter-processor communication-a network
  + large tasks that would normally require larger processors to be tackled with several inexpensive smaller processors
* serial interface: allow processors to communicate without the need for shared memory and semaphores, with the relative problems
* serial interface: it would not substitute the parallel buses for some cases
  + parallel buses: for operational fetches, address and data buses, other microprogram control are the clear winner
* memory mapping peripherals: technique commonly used for systems with address and data buses
  + allowing parallel access to off-chip peripherals
* memory mapping peripherals: however with many 8-bit microncontrollers (alone 8-pin) with no external address / data bus available for design, memory-mapping is not an option

# the FTT-CAN protocol: Why and How

* CAN protocol: supports the time-triggered communication in flexible way and being an efficient combination of both time and event-triggered traffic with temporal isolation
* CAN: two complementary subsystems: the Synchronous and the Asynchronous Messaging System, the protocol can convey real-time traffic of either type
* Flexibility on fieldbus systems: dynamic communication requirements with the online addition, removal and adaptation of message streams must be supported, sometimes stringent timing constraints arising from control and monitoring requirements are also needed… flexibility and timeliness have typically been considered separately and the most filedbuses available today favor either one aspect or the other
* Fieldbus systems: another requirement is the capacity to deliver both time and event trigger communication services under timing constraints
  + Time communication: well suited for conveying periodic updates of state data
  + Event trigger communication: well suited for conveying alarms and management data
  + In every case one against to other is often privileged: In systems eminently time triggered, event triggered services are either non existing or handled inefficiently in terms of either response time or network utilization, on the other hand, in systems eminently event -triggered, interesting properties of time triggered services such as composability with respect to the temporal behavior are normally lost
* Paradigms aim to reach time and event triggered service in an efficient manner … this is done by the FTT-CAN protocol: the flexible time triggered communication on controller area network
* Static vs dynamic traffic scheduling
  + Static scheduling: communication requirements are fixed throughout all the system operation
    - Release and transmission times are known at pre-run time
    - Timeliness is supported: complex offline schedulability analysis to be carried out are possible… but the level of flexibility is low
  + Dynamic scheduling: the communication requirements may change at run time
    - Flexibility is guaranteed
    - But for timeliness guarantee an online admission control based on an adequate schedulability analysis must be used
    - Timing constraints but with no timeliness guarantees
  + In the rest of scratched section some other paradigms related to the static and dynamic scheduling are reported
* FTT-CAN: dual phase elementary cycle concept in order to combine time and event triggered communication with temporal isolation
* FTT-CAN: a master node schedules online and centrally the time-triggered traffic: events are kept centrally in one local table
* FTT-CAN: with online admission control the protocol supports the time-triggered traffic in a flexible way under guaranteed time (dynamic planning-based scheduling algorithm)
* FTT-CAN: time triggered traffic is controlled without collisions either through master-slave transmission control or through control of transmission instants

# CAN+: A new backward-compatible Controller Area Network (CAN) protocol with up to 16x higher data rates∗

* CAN: the facto standard for filedbuses in many domains, especially automotive
* CAN physical setup: terminated 2 wired bus where nodes can be connected in any order
* CAN: bits are in the Non return to zero method logic
  + Logical 0: represented by a voltage difference between the 2 wires. It is the dominant logic
  + Logical 1: no difference in voltage. It is the recessive logic, applied when both bits are pending
  + Size of this voltage difference depends on the environment of application
  + It may results in long sequences of 1s or 0s where missing edges makes synchronization impossible
  + Bit stuffing is hence used: bit inverted inserted between 5 consecutive similar bits, this additional bit is filtered by the receiver
* AlCAN: length of a bit is depending on the bit rate used (eg 1 microsecond at 1Mbit/s), the signal is not perfect and delay also needs to be considered
  + A bit transmission interval is divided into 8 to 25 time segments
  + After about 2/3 (depending on settings) of the bit time, the signal is sampled
* CAN: synchronization is done after each edge by resetting the counter of the time segments, the length of the bit time is adjusted if the detected edge is offset by more than the predefined resynchronization jump witdth
* CAN: message oriented protocol where each node can read all the messages
* CAN and four types of messages:
  + Data message is for the information exchange
    - Data variations are form 0 to 8 bytes
    - The content is described by a unique identifier
      * Besides the content description the identifier also defines the priority by which access to the bus is granted
* CAN: bus access is decided by CSMA/BA (Carrier sense multiple access with bitwise arbitration):
  + Each node that wants to have access to the bus starts sending its message as soon as the bus is idle for 3 bit times
  + Every sent bit is also watched: when the sent bit differs from the watched, then a message with higher priority (lower identifier) also is pending and transmission is stopped
  + After sending the identifier, only the message with the highest priority is left and has exclusive bus access
* CAN: non destructive priority based access with up to 2032 priorities is guaranteed
* CAN and error handling: one of the most reason why it is successful
* CAN and error handling: as soon as any node detects an error, it sends an error message which consists of six consecutive dominant bits, the sequence is unique and can be immediately detected
  + There are 3 states in which a node can be depending on number of error counted:
    - Error active: the node is working as intended
    - Error passive: the node is still monitoring bus traffic, but don’t send error messages and defer to error active nodes
    - Bus off: not allowed to have any influence on the bus
* CAN and main reasons of speed limit: output resistance, propagation delay and oscillator inaccuracy
  + Output resistance: it only limits the length of the bus depending on the power used
  + Oscillator inaccuracy: overcome when more expensive HW is used, can be expensive
  + Propagation delay: 2 components
    - Signal delay on cable
    - Delay within the controller and transceiver
    - Propagation delay: reason why CAN has bit rate limitations depending on the length of the bus
      * In this fashion error occurs if during the arbitration phase a sending node cannot observe if it is overwritten until the end of the bit time (time interval from the beginning of transmission of a single bit to the end)
      * Propagation delay is not allowed to be longer than the time from the beginning of the bit time to the sampling point Wikipedia – cotroller area network

# All links

* https://www.mdpi.com/1424-8220/21/16/5347
* https://www.mdpi.com/1424-8220/14/4/6165
* http://telkomnika.uad.ac.id/index.php/TELKOMNIKA/article/view/17427/9978
* <https://ieeexplore.ieee.org/abstract/document/9615333>
* <http://coecsl.ece.illinois.edu/se423/datasheets/SerialProtocolsCompared.pdf>
* <https://ieeexplore.ieee.org/abstract/document/1097741>

others

* <https://medium.com/@sarakadam18/the-most-popular-serial-communication-protocols-c191e34be8ac>
* <https://en.wikipedia.org/wiki/CAN_bus>