Aalto University
School of Electrical
Engineering

Sensors & Buses

Protopaja / Protocamp / ELEC-D0301

Shahram Barai
9.6.2022

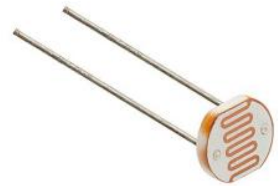
Sensors

- Sensor can detect an spect of the physical environment and turn it into useful information

Sensors classification:

- Active sensor
    - requaire an external signal
- Passive sensor
    - work without any external signal
- Analog sensor
    - produce an analog output, ie a continuous signal
- Digital sensor
    - work with discreate, digital data

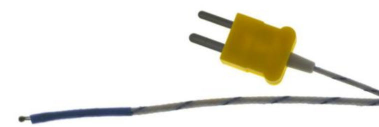
Sensors




- Light, Colour
  - LDR (photoresistor), Phtodiodes
- Sound
  - Microphone, Piezo
- Temperature
  - Thermistor, Termocouple, PT100, LM35…
- Humidity, Moisture
  - e.g. DHT11
- Pressure
  - e.g. MPX5100AP
- Gas Contents
  - e.g. MQx (for different gases), MQ3 (for athanol)
- Force Sensors
  - Force Sensing Resistor, Load Cell
- Tilt Sensors
  - Accelerometer

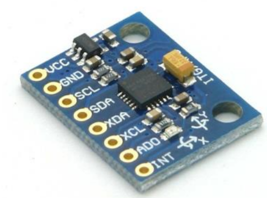










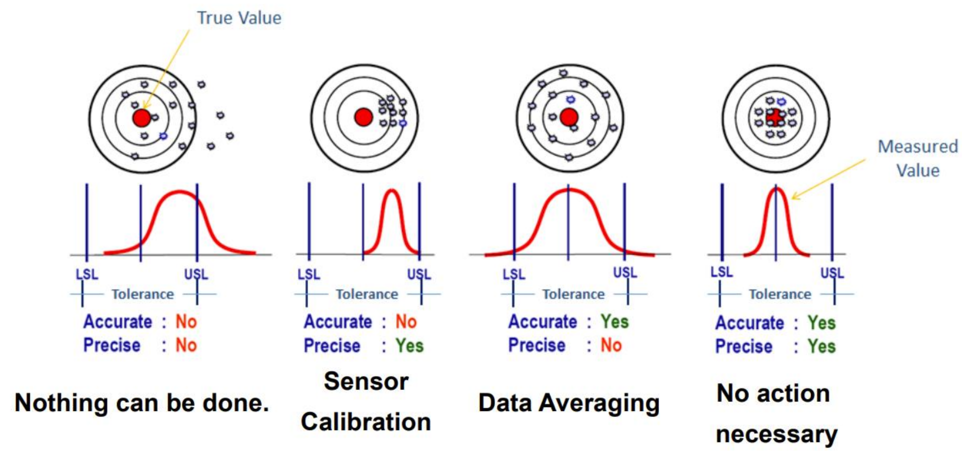


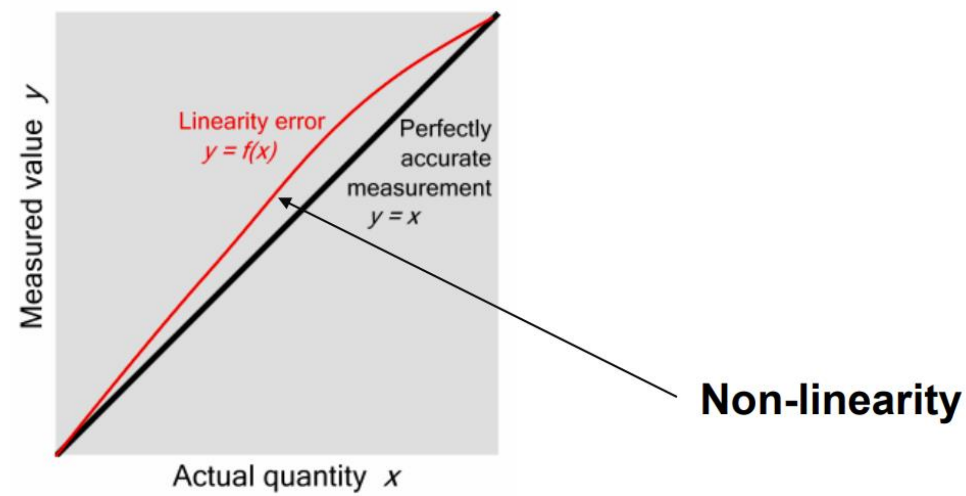
Calibration

- Errors is Sensor Measurement
  - Improper Zero Reference
  - Shift in Sensor Range
  - Mechanical Damage

- Calibration Methods
  - One point calibration
  - Two point calibration
  - (Multi-point Curve Fitting)

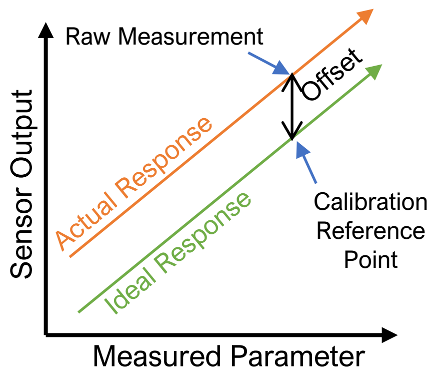
- Characteristic curve
  - Offset
  - Sensitivity or Slope
  - Linearity

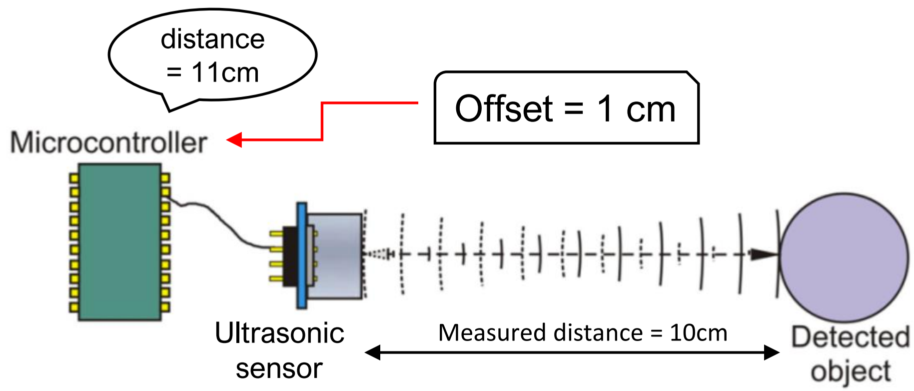




One Point Calibration

- One point calibration is used to correct the sensor offset errors when accurate measurement of only a single level is required and the sensor is linear.

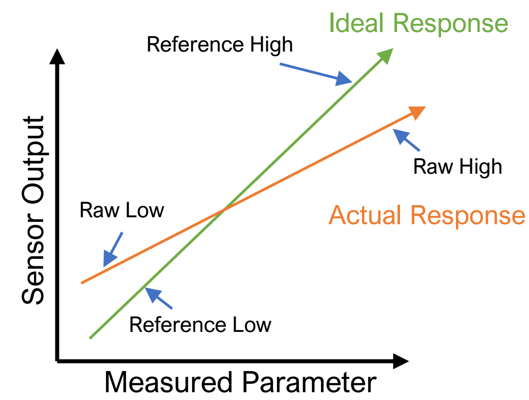




- How to do it:
  1. Take a measurement with your sensor.
  2. Compare that measurement with your reference standard.
  3. Subtract the sensor reading from the reference reading to get the offset.
  4. In your code, add the offset to every sensor reading to obtain the calibrated value.

Two Point Calibration

- Sensor output -> linear over the measurement range.
- It can be applied to either raw or scaled sensor outputs.
- It is used to correct both slope and offset errors.

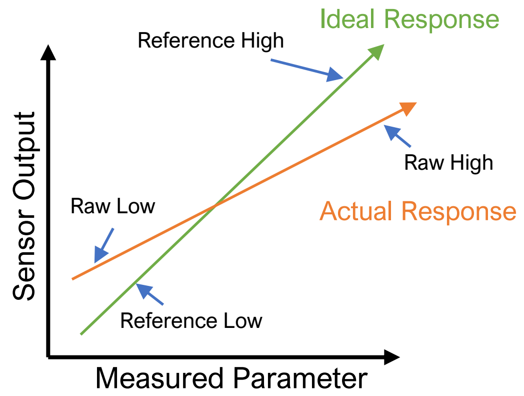


How to do it:

1. Take two measurements with your sensor: One near the low end of the measurement range and one near the high end of the measurement range. Record these readings as "RawLow" and "RawHigh“
2. Repeat these measurements with your reference instrument. Record these readings as "ReferenceLow" and "ReferenceHigh“
3. Calculate "RawRange" as RawHigh – RawLow.
4. Calculate "ReferenceRange" as ReferenceHigh – ReferenceLow
5. In your code, calculate the "CorrectedValue" using the formula below:

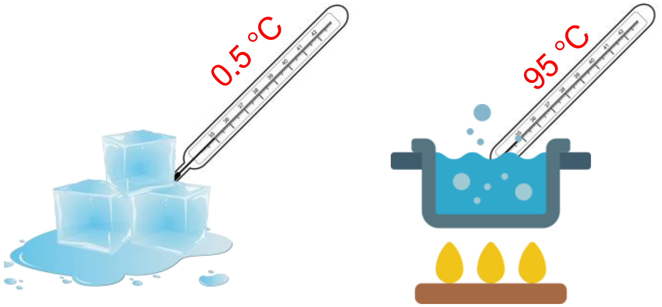
CorrectedValue = (((RawValue – RawLow) * ReferenceRange) / RawRange) + ReferenceLow

Two Point Calibration

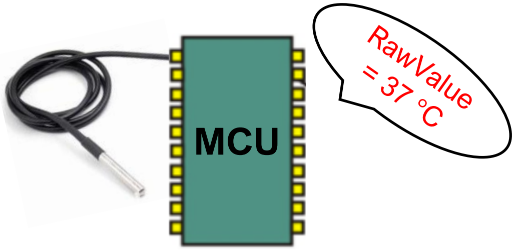


Temperature sensor:

- Physical standards -> normal sea-level atmospheric pressure
  - **ReferenceLow** = 0°C
  - **ReferenceHigh** = 100°C
  - **ReferenceRange** = 100°C



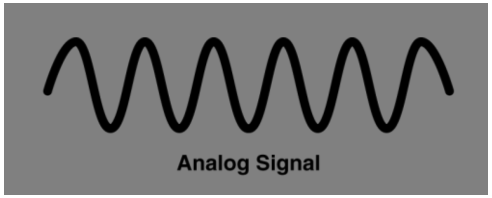
- Raw reading are:
  - **RawLow** = 0.5°C (ice-water bath)
  - **RawHigh** = 95°C (boiling water)
  - **RawRange** = 95.5°C

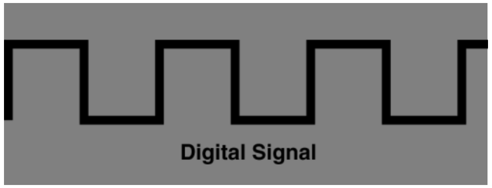


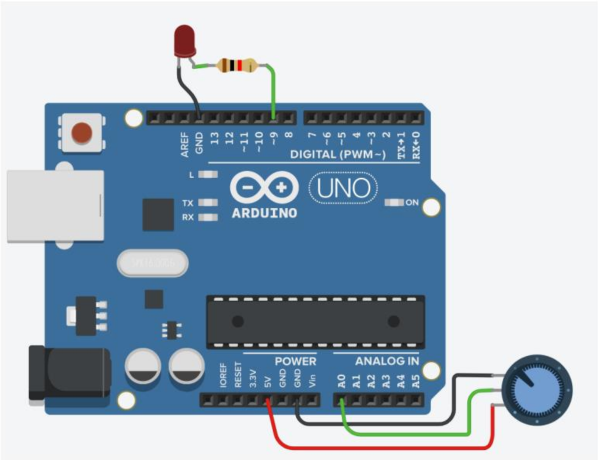
CorrectedValue = (((37 + 0.5) * 100) / 95.5) + 0.0 = **39.3°C**

Analog & Digital Sensor Output

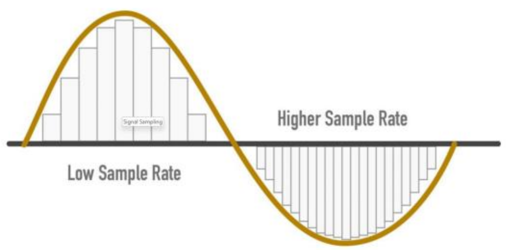
- If analog sensor data is processed by digital hardware it must be converdet.
- Most sensors have A/D converters built in
- High preformance sensors usually have only analog inputs



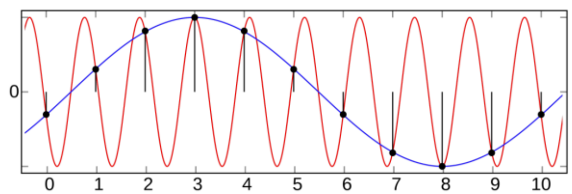




A/D Converters



- Convert analog data to digital through quantisation
  - Leads to some data loss
- Sampling Frequency must be higher than 2 x maximum frequency being sampled.
  - With low sampling frequency aliasing may occur



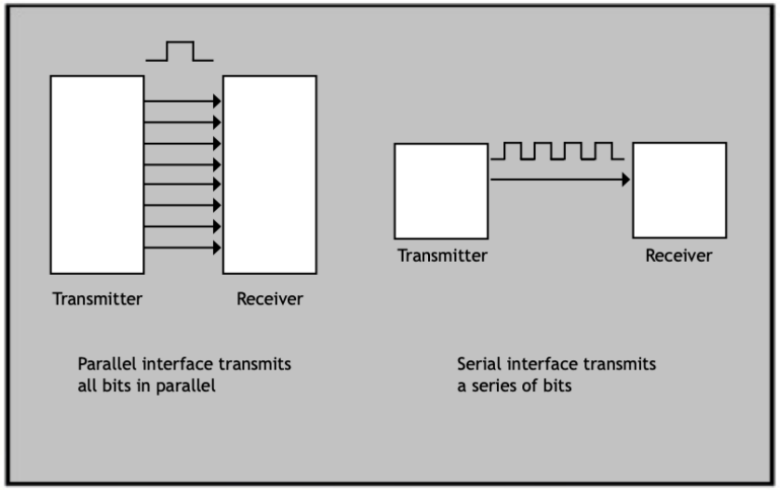
- MCU’s usually have builtin A/D’s
  - E.g. in Arduino UNO whenever we use analog pins (analog_read(___)) MCU’s A/D unit is used (sampling frequency 1kHz & Bit depth 10 bit).
- Noise considerations:
  - Component noise
  - ADC saturation: amplify signal before feeding it to the ADC.

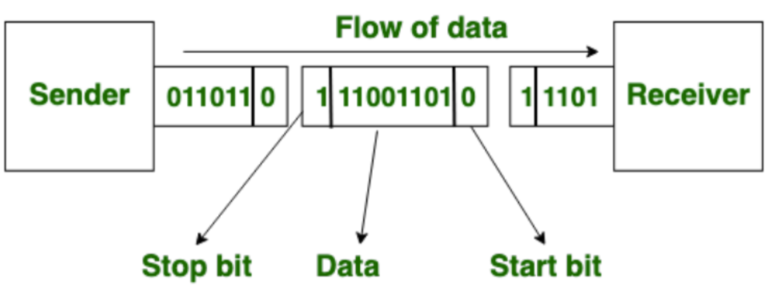
Buses

- Needed to communicate to sensors & other devices
- Implemented with hardware or software drivers
- Operate according to protocols
    - SPI, I2C, UART

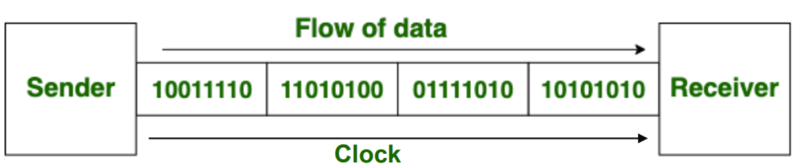
- Parallel vs Serial

- Asynchronous vs Synchronous





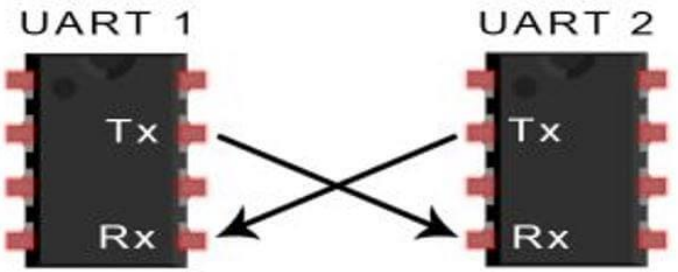
Asynchronous Transmission



Synchronous Transmission

UART

- Stands for **Universal Asynchronous Reception and Transmission**
    - Only GND, TX and RX, no separate clock signal
    - Simple, Easy to use
- Protocol not defined, several standard electrical interfaces
- Usually used for specific peripherals, E.g. Bluetooth transmitters, GPS, GSM
- Arduino library: (Serial, SoftwareSerial)



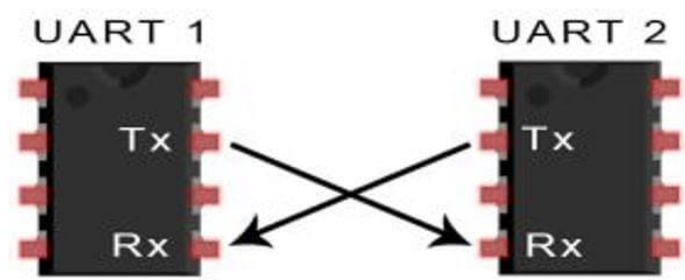
UART

Advantages of using UART

- Simple to operate, well documented as it is a widely used method with a lot of resources online
- No clock needed
- Parity bit to allow for error checking

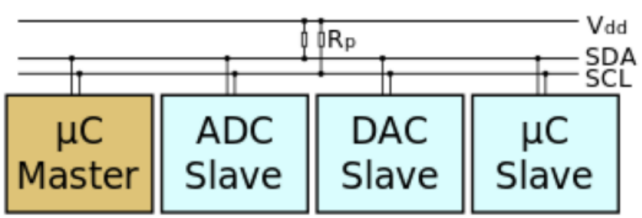
Disadvantages of using UART

- Size of the data frame is limited to only 9 bits
- Cannot use multiple master systems and slaves
- Baud rates of each UART must be within 10% of each other to prevent data loss.
- Low speed



I2C

- Inter-Integrated Circuit, Display Data Channel, System Management Bus…
  - Low speed: 400 / 100 kHz usually, but higher speed devices available (>1 MHz)
  - Developed, Patented & Controlled by Philips Semiconductors
- Master initiated, half-duplex
  - SDA (SerialData), SCL (SerialClock)
  - Several devices can share same bus, (each has 7-bit unique address)
  - Devices interface open-collector/open-drain (pull-up resistors)
  - Available at VGA, DVI, HDMI-Connectors
  - Used in PCI, DIMM etc. for identification & configuration
- Easy to use with Arduino (Wire Library)



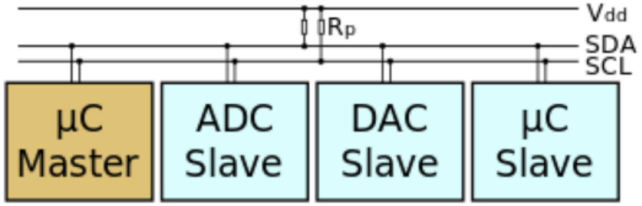
I2C

Advantages of using I2C

- Has a low pin/signal count even with numerous devices on the bus
- Flexible, as it supports multi-master and multi slave communication.
- Simple as it only uses 2 bidirectional wires to establish communication among multiple devices.
- Adaptable as it can adapt to the needs of various slave devices.
- Support multiple masters.

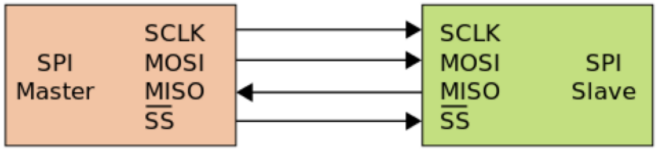
Disadvantages of using I2C

- Slower speed as it requires pull-up resistors rather than push-pull resistors used by SPI. It also has an open-drain design = limited speed.
- Requires more space as the resistors consume valuable PCB real estate.
- May become complex as the number of devices increases.



SPI

- Serial Peripheral Interface Bus
  - High speed (up to > 10 MB/s), full duplex capable
- Master initiated, simultaneous bidirectional data transfer capable
  - MISO (master in slave out), MOSI (master out slave in), SCK (serial clock), SS/CS (slave / chip select)
- Easy to use with Arduino libraries (SPI Library)



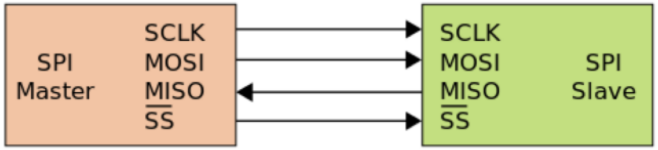
SPI

Advantages of using SPI

- The protocol is simple as there is no complicated slave addressing.
- It is the fastest protocol compared to UART and I2C.
- No start and stop bits unlike UART which means data can be transmitted continuously without interruption
- Separate MISO and MOSI lines which means data can be transmitted and received at the same time

Disadvantages of using SPI

- More Pin ports are occupied, the practical limit to a number of devices.
- There is no flow control specified, and no acknowledgement mechanism confirms whether data is received unlike I2C
- Uses four lines – MOSI, MISO, NCLK, NSS
- No form of error check unlike in UART (using parity bit)
- Only 1 master



Other

1-Wire:

- Low speed single datawire bus by Dallas/Maxim
- Several devices can share same data bus
- E.g. used in DS18x20 digital interfave temperature sensors
- Arduino library (OneWire)

MIPI:

- Camera & Display Serial interface, HD resolutions
- Requires driver to work
- Found on Raspberry Pi platforms

USB:

- Hard, complicated protocol
- Always requires a driver, usually it is easier to use a general Serialover-USP that emulates traditional serial port (UART)
- Supplies power, max 500 mA