Università
della
Svizzera
italiana

Facoltà
di
scienze
informatiche

Edge Computing in the IoT

I/O in Embedded Systems

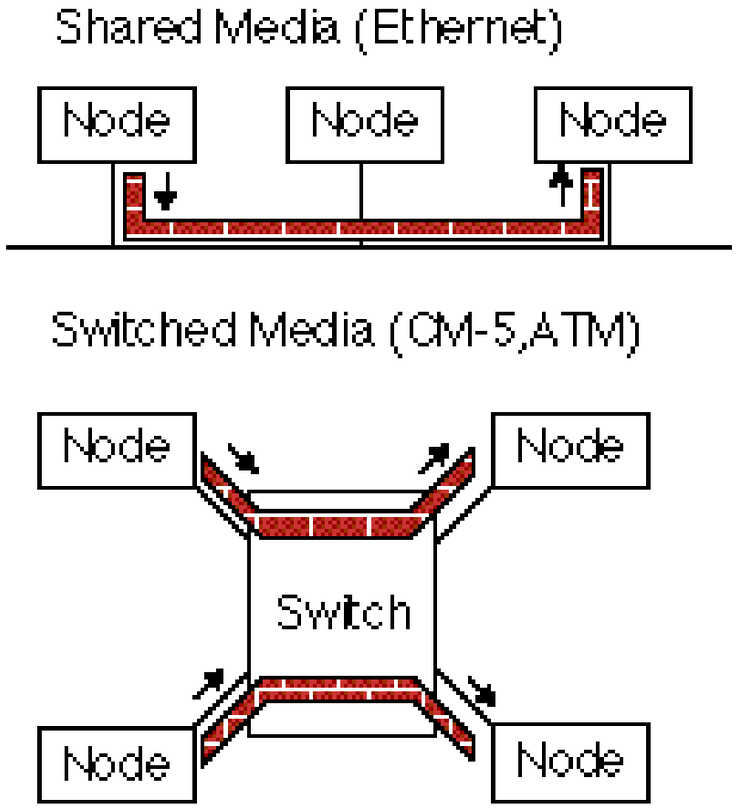
Alberto Ferrante (alberto.ferrante@usi.ch)

Interconnections

- On-chip networks
  - Interconnect microarchitectural functional units, register files, caches, IP cores within chips or in multichip modules
- System/storage Area Networs (SANs)
  - Inter-processor and processor-memory communication for multiprocessors as well as for high-performance storage and I/O
- Local area networks (LANs)
  - Interconnect autonomous computing systems in localized environments (e.g., building)
- Wide area networs (WANs)
  - Interconnect autonomous computing systems distributed across the globe

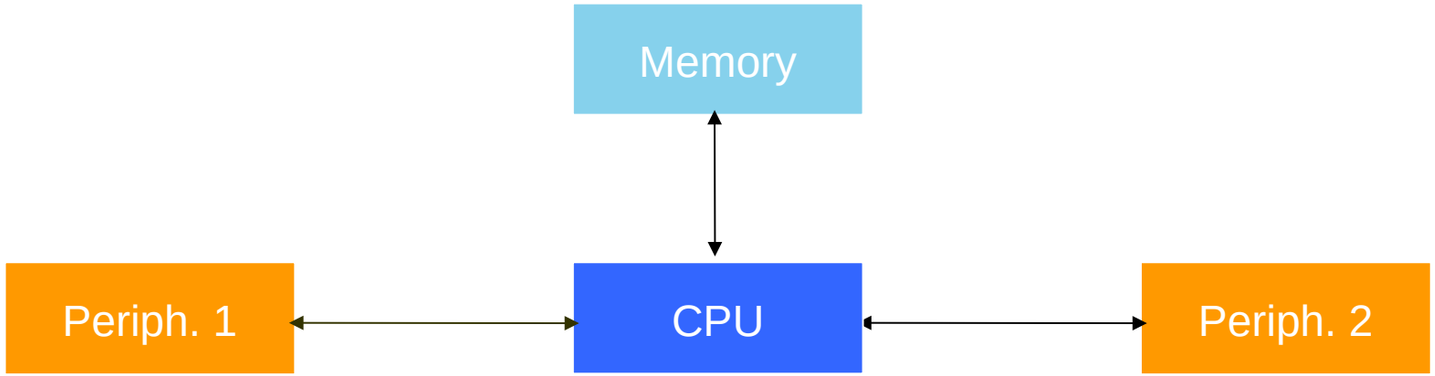
Types of interconnections

- Half-duplex/Full-duplex
- Shared-media
  - Require a mechanism to *arbitrate* the shared resource
  - Do not scale with the number of connected devices and arbitration further limits scalability
- Switched-media
  - Passive point to point links between active *switch* components that dynamically connect source-destination pairs
  - Multiple pairs are allowed to communicate simultaneously
  - May require arbitration for paths, routing, ...



Point to Point Connection

- It is expensive
- It does not scale
- In computer architectures, all the communications need to pass through the CPU



Bus

- A bus is a communication means shared by the components/devices
- The components can communicate through one or more buses
- Bus performance have a great impact on system performance
- Components share a common “set of wires”:
  - Medium access needs to be controlled to guarantee exclusive access:
    - One *bus arbiter* granting bus access
    - A Medium Access Control method
- Single bus / Hierarchy of busses

Bus Drawbacks

- The bus is a shared resource:
  - it may become a bottleneck for I/O:
    - Bus throughput
    - Bus contention: only one unit at a time can access the bus!
- Constraints over physical maximum length/speed of the bus;
- Constraints over the maximum number of connected peripherals.

I/O

- Connection of peripherals (sensors, actuators, …) to the CPU can be handled by using a number of different protocols
- Every peripheral has its own specific characteristics
  - The choice of the protocol depends on the characteristics (e.g., required throughput)
- Whenever analog I/O pins are provided on a development board, an internal ADC/DAC handles the conversion from/to digital to/from analog
  - Otherwise, we need an ADC/DAC connected to the digital pins

I/O Modes and Protocols

- General purpose IO
- Serial
- SPI
- I^2C

General Purpose I/O Pins

- No predefined protocol, used for simple sensors/actuators
  - E.g., a switch
  - The device (and the programmer) defines how the communication happens
- They can be
  - Inputs
  - Outputs
  - Configurable: can be switched between input or output
- The application manages the PIN directly or by means of the operating system

Serial connection

- Used for
  - Data communication between different parts of the system
  - Communication with external devices
  - Downloading firmware to keep the embedded systems up to date
  - Providing a debugging or console interface
- Electronic Industries Association (EIA) standard RS-232 (1969) is one of the serial protocols available
  - Electrical characteristics
  - Specifications regarding connection hardware and pin-outs
  - Transmission speeds vary between 20Kbps up to 115.2Kbps
- In modern devices, the serial port is mapped over USB, that is also a serial connection

Serial Peripheral Interface - SPI

- Communication across short distances
- Master-Slave (Multiple Slaves, Single Master)
  - **Traditional names, being dropped in favor of *Controller* and *Peripheral***
- Developed By Motorola
- Four-wire protocol
- Serial communication
- Synchronous: data transfer clocked with clock Signal
- Data Rate: 10Mbps
- Full Duplex Protocol
- No hardware-slave acknowledgement
- Data is exchanged: no device can be a transmitter-only or receiver-only

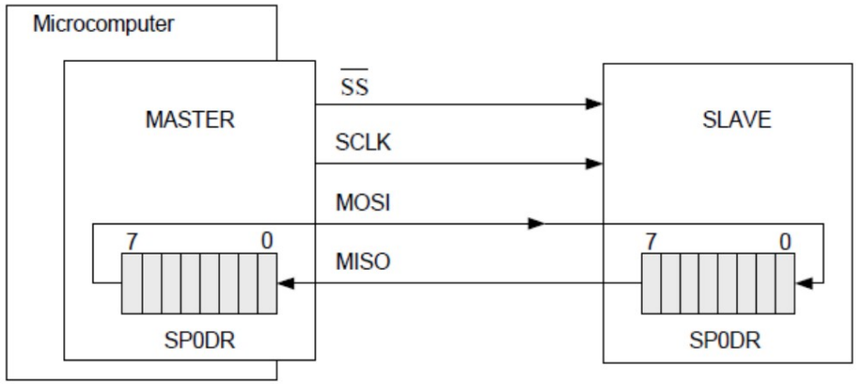
Serial Peripheral Interface – SPI Signals

- Master Out Slave In (MOSI)
  - Carries data out of Master to Slave
- Master In Slave Out (MISO)
  - Carries data from Slave to Master
- Serial Clock (SCLK)
  - Divider on the SPI clock: 2, 4, 8, 16, 32, 64, 128 or 256
- Slave Select (SS)

Serial Peripheral Interface - SPI

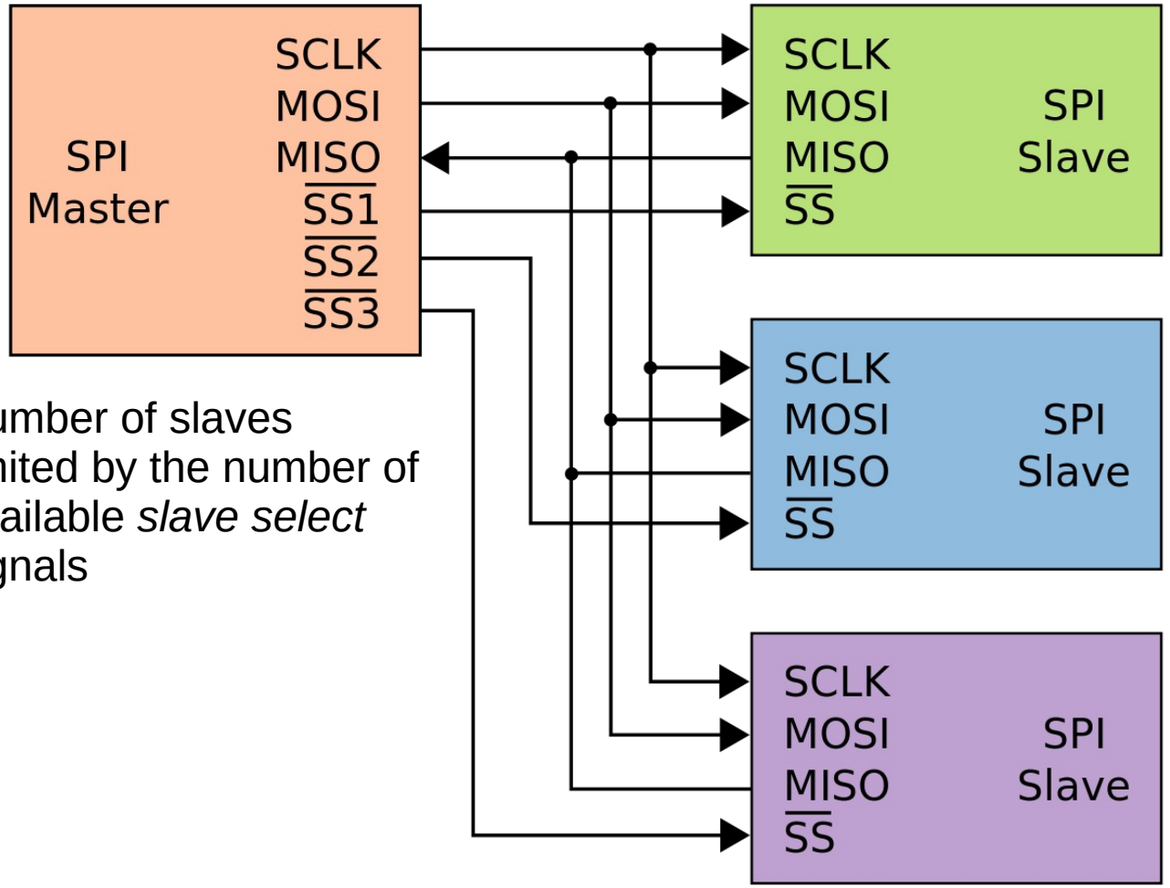
- The Master device controls the clock (SCK)
  - All slaves are controlled by the master clock
- No data is transferred unless a clock signal is present

Serial Peripheral Interface - SPI



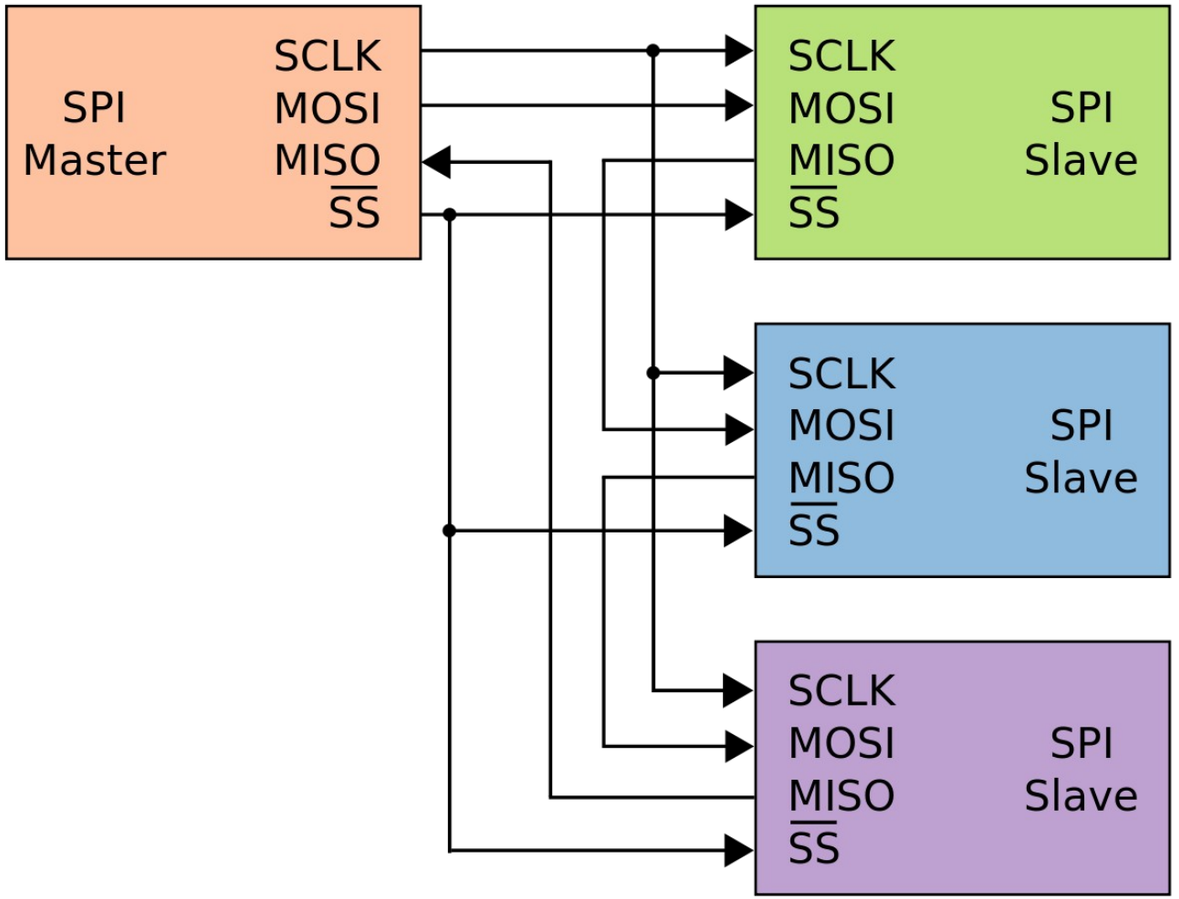
- Master sets *Slave Select* low (i.e., it enables the slave)
- Master generates clock
- Shift registers shift in and out data

SPI – Multiple Slaves

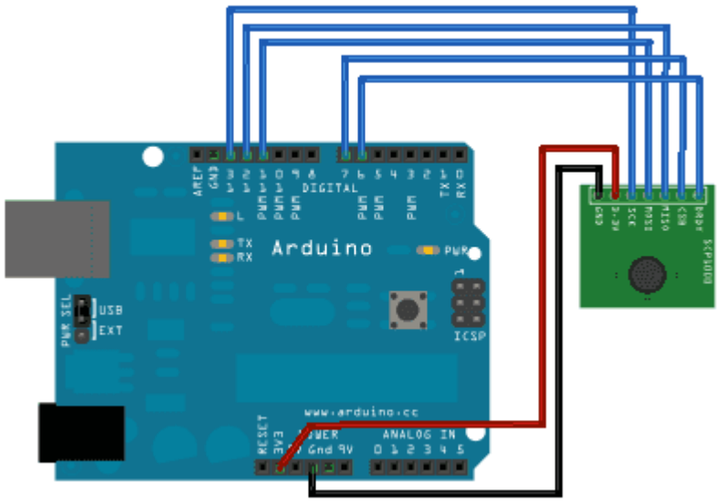


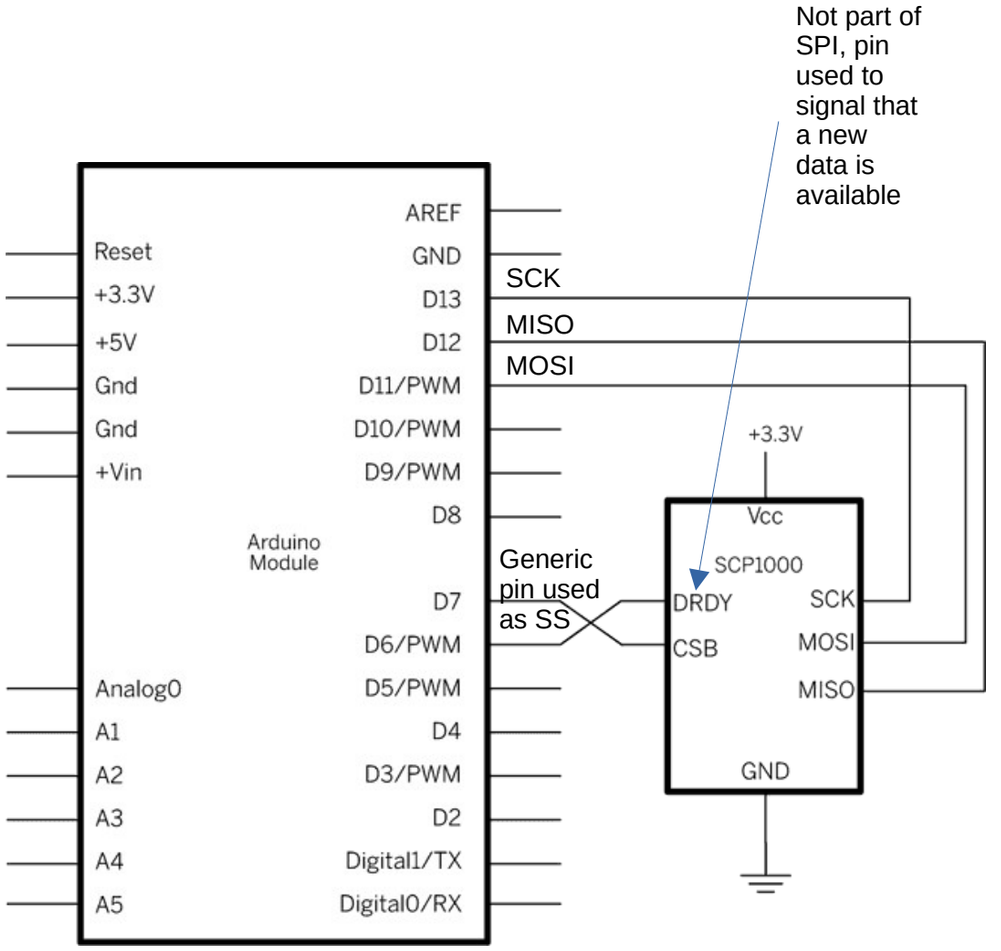
- Number of slaves limited by the number of available *slave select* signals

SPI – Daisy Chained Slaves



SPI – Example





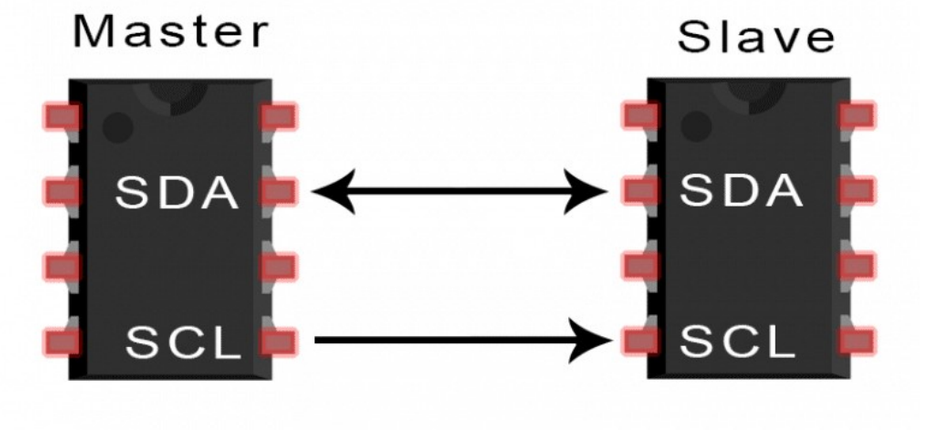
https://docs.arduino.cc/tutorials/communication/BarometricPressureSensor

I^2C - Inter-Integrated Circuit Bus

- Standard digital communication bus designed to interconnect integrated circuits belonging to the same board
- It has been introduced by Philips to interconnect integrated circuits in TV-sets in the ’80s

I²C - Inter-Integrated Circuit Bus

- Bus architecture
  - Can connect multiple slaves to a single master
- Serial bidirectional communication
- Master: controls the bus and has the responsibility of starting a communication
  - Only one master can be active at each specific time
- Slave: all the other devices which respond to master solicitations

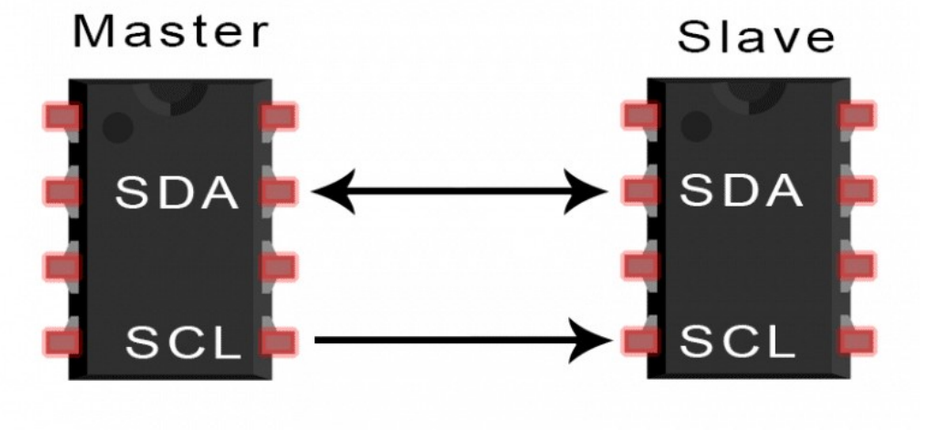


https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

I²C - Inter-Integrated Circuit Bus

- **SDA**: Serial DAta, bidirectional:
  - data bits, serial
- **SCL**: Serial Clock:
  - undirectional from master to slaves
- Synchronous interface

Wires Used	2
Maximum Speed	Standard mode= 100 kbps
	Fast mode= 400 kbps
	High speed mode= 3.4 Mbps
	Ultra fast mode= 5 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Max # of Masters	Unlimited
Max # of Slaves	1008



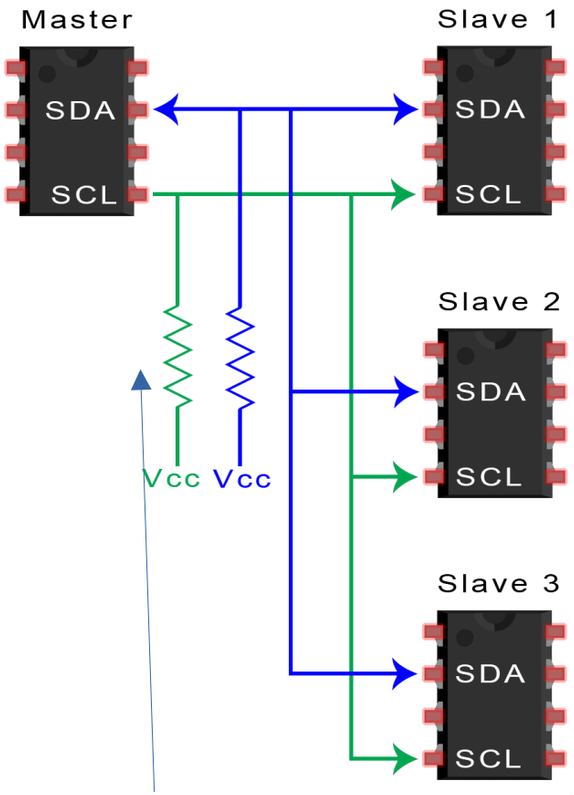
https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

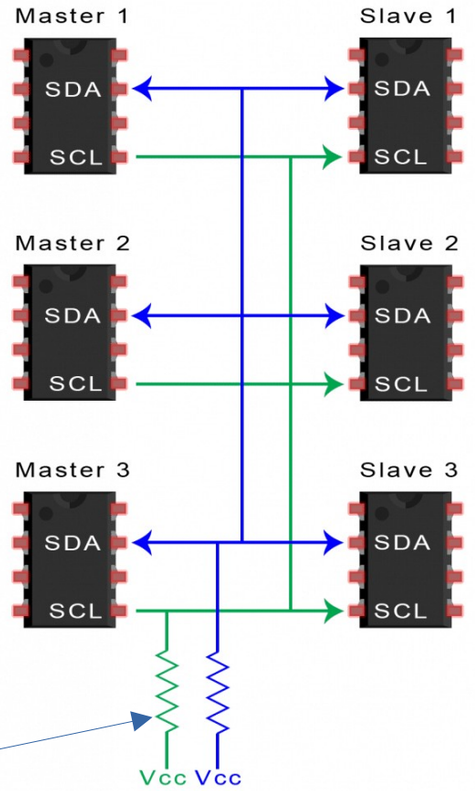
I²C - Inter-Integrated Circuit Bus

- Each slave has an address
  - 7-bit, widely used
  - 10-bit, used only in some special cases
- Each slave device has a register map
  - Each register
    - holds an 8-bit value
  - Each register is mainly used to:
    - Configure the device
    - Send commands to the device
    - Hold data
- Each register can be read or written by the master
- While SCL is driven by the master
  - Address frame is sent by the master
  - The data is sent on SDA by either the master or the slave, depending on whether the R/W bit indicated a read or write operation
  - The number of data frames is arbitrary, and most peripheral devices will auto-increment the internal register, meaning that subsequent reads or writes will come from the next register in line

https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

I²C – Single or Multiple Masters





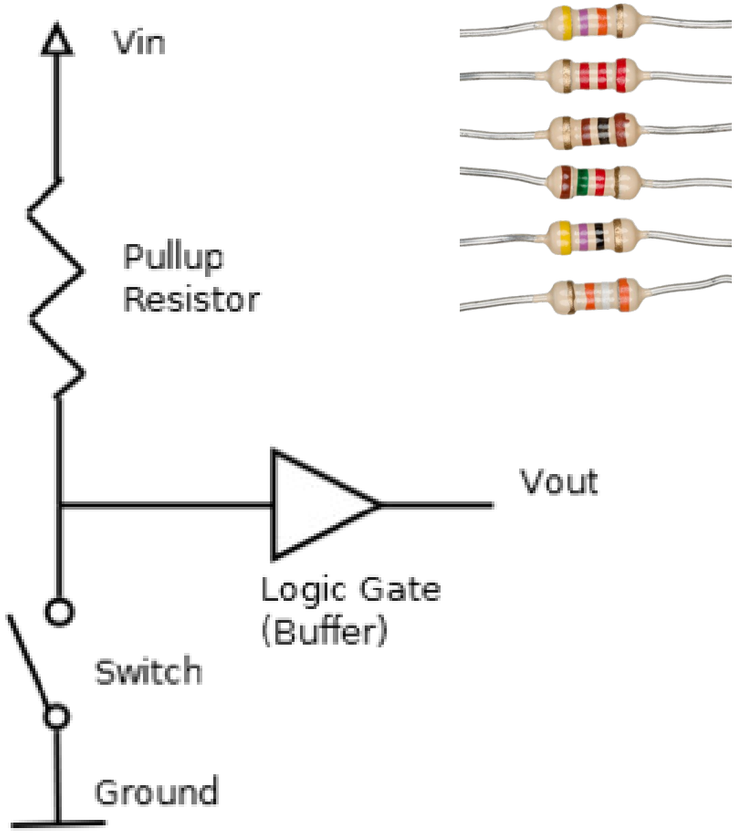
Multi-master bus:

- Arbitration logic. If two devices start to communicate at the same time the one writing more zeros to the bus (or the slower device) wins the arbitration and the other device immediately discontinues any operation on the bus
- Bus busy detection. Each device must detect an ongoing bus communication and must not interrupt it. This is achieved by recognizing traffic and waiting for a stop condition to appear before starting to talk on the bus.

https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

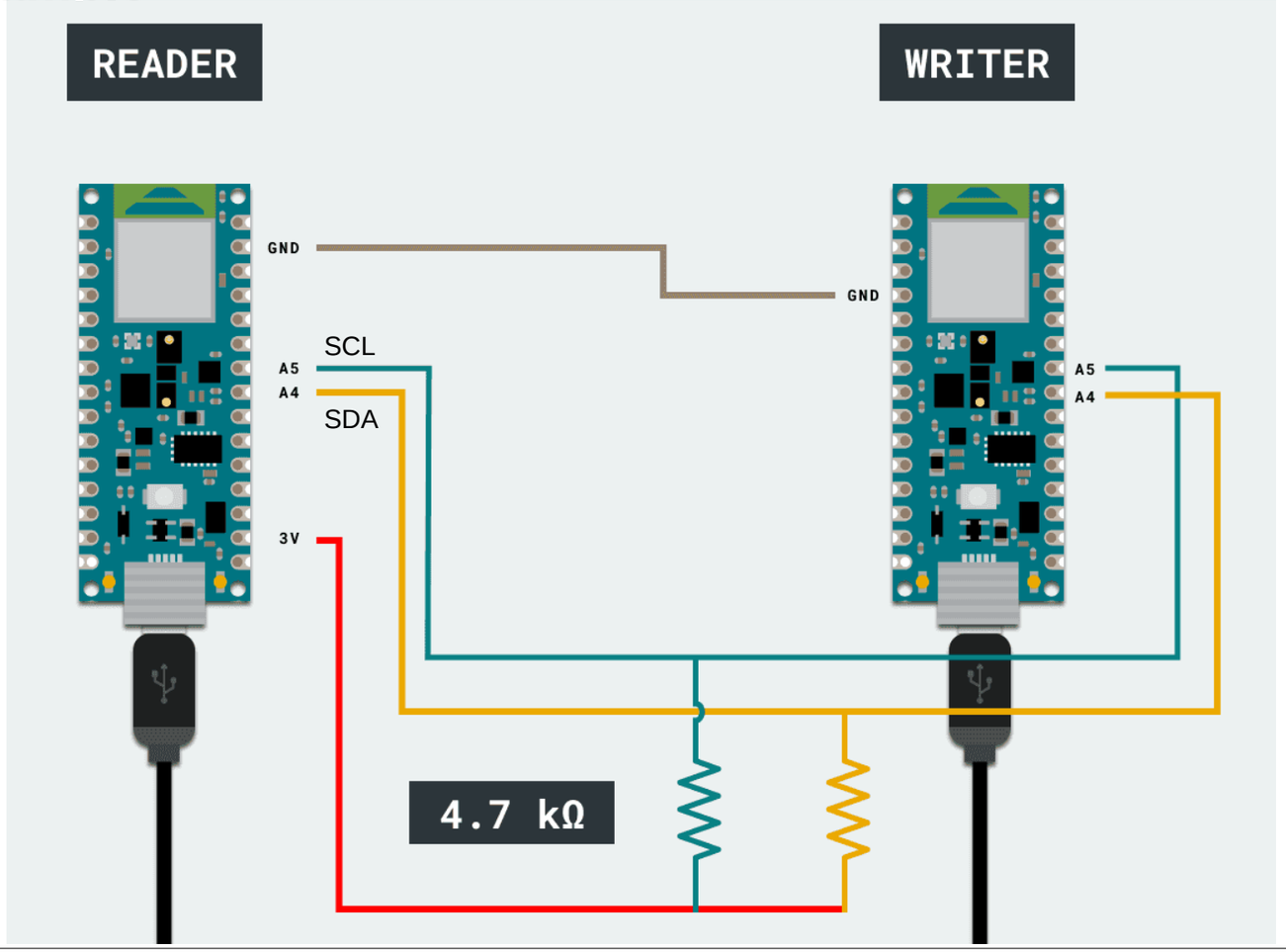
Pull-up and Pull-down Resistors

- When a known state for a signal is required, a pull-up resistor or a pull-down resistor is needed
  - E.g., pin connected to a switch: when the switch is open, the pin is floating
  - If no resistor is used, when the switch is closed a short circuit is obtained!
  - Without pull-up/down the signal would remain floating (i.e., unknown value)
- Typical value: 10kohm
  - For I²c, 4.7kohm
- Some devices already include pull-up or pull-down resistors



https://en.wikipedia.org/wiki/Pull-up_resistor

I²C – Example



https://docs.arduino.cc/tutorials/nano-33-ble-sense/i2c

Bit Banging

- Some microcontroller boards might not provide support for a protocol required for communicating with another device (e.g., SPI or I^2C)
  - In some cases, these protocols can be emulated in software
- Bit banging is a technique for serial communication in which the whole communication process is handled via software instead of dedicated hardware
  - The software application will control the transmission/reception bit by bit with the correct timing
  - It can only be done for protocols that are somewhat tolerant to jitter

In Summary...

- I/O modes and protocols:
  - GPIO, Serial, I^2C, SPI
- Pull-up/down resistors