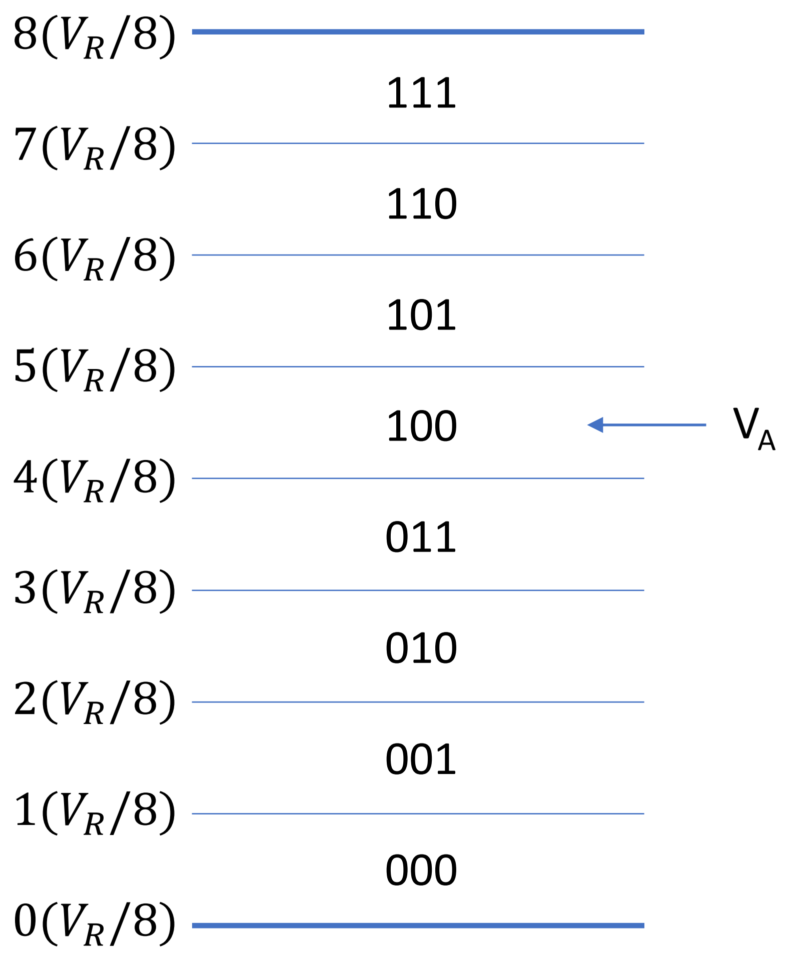
Lecture 4 – ADC and Digital signals

Dr. Aftab M. Hussain,

Assistant Professor, PATRIoT Lab, CVEST

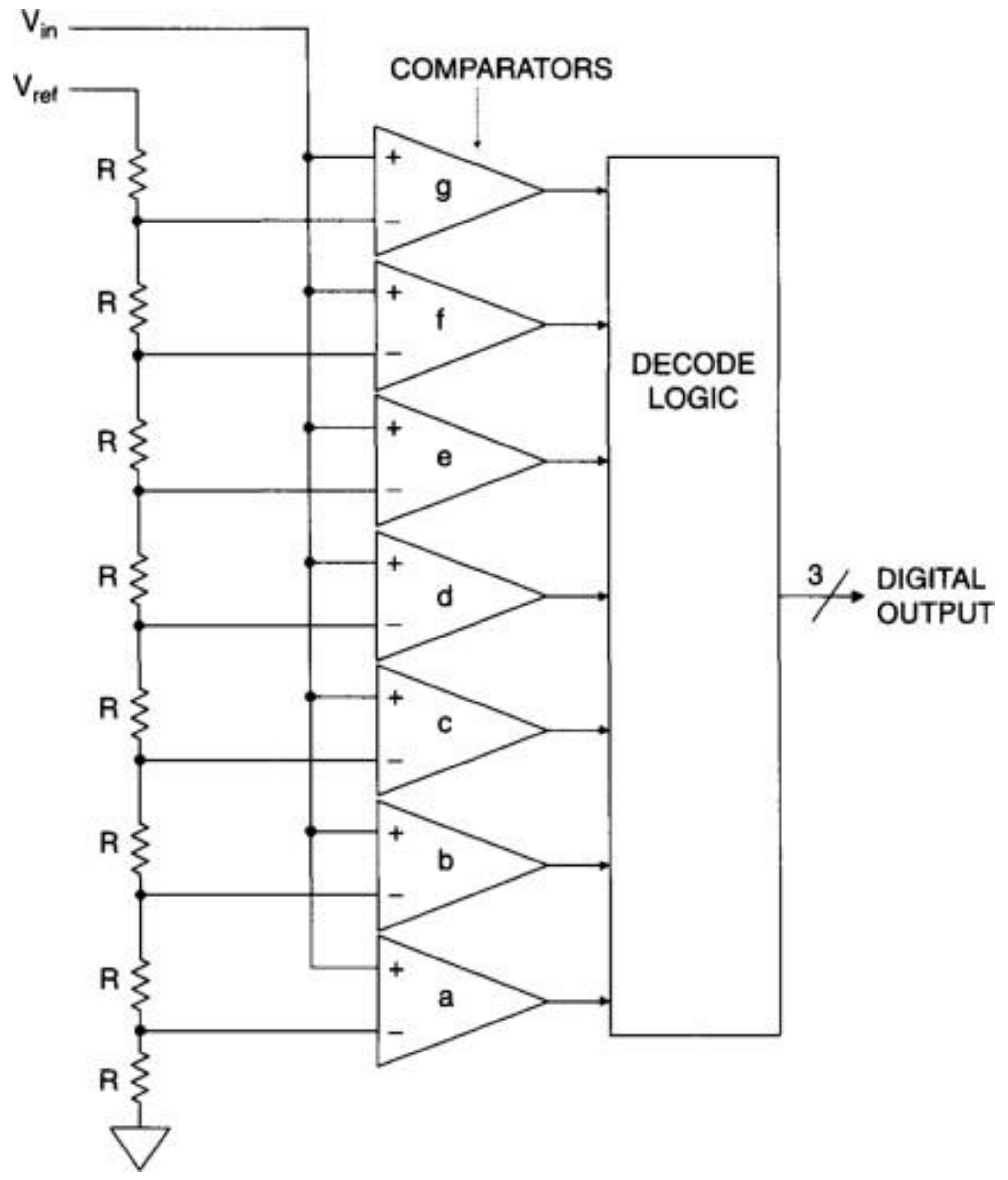
ADCs

- To convert from analog to digital, we may think of dividing the reference voltage by 2^n and consider each voltage interval (corresponding to 000, 001, etc.) as a bin
- If the input voltage V_A falls in the 100 bin; therefore, the output of the ADC would be 100
- Thus, the basic idea behind an ADC is simple:
  - Generate reference voltages V_1, V_2, etc.
  - Compare the input V_A with each of V_i to figure out which bin it belongs to
  - If V_A belongs to bin k, convert k to the binary format



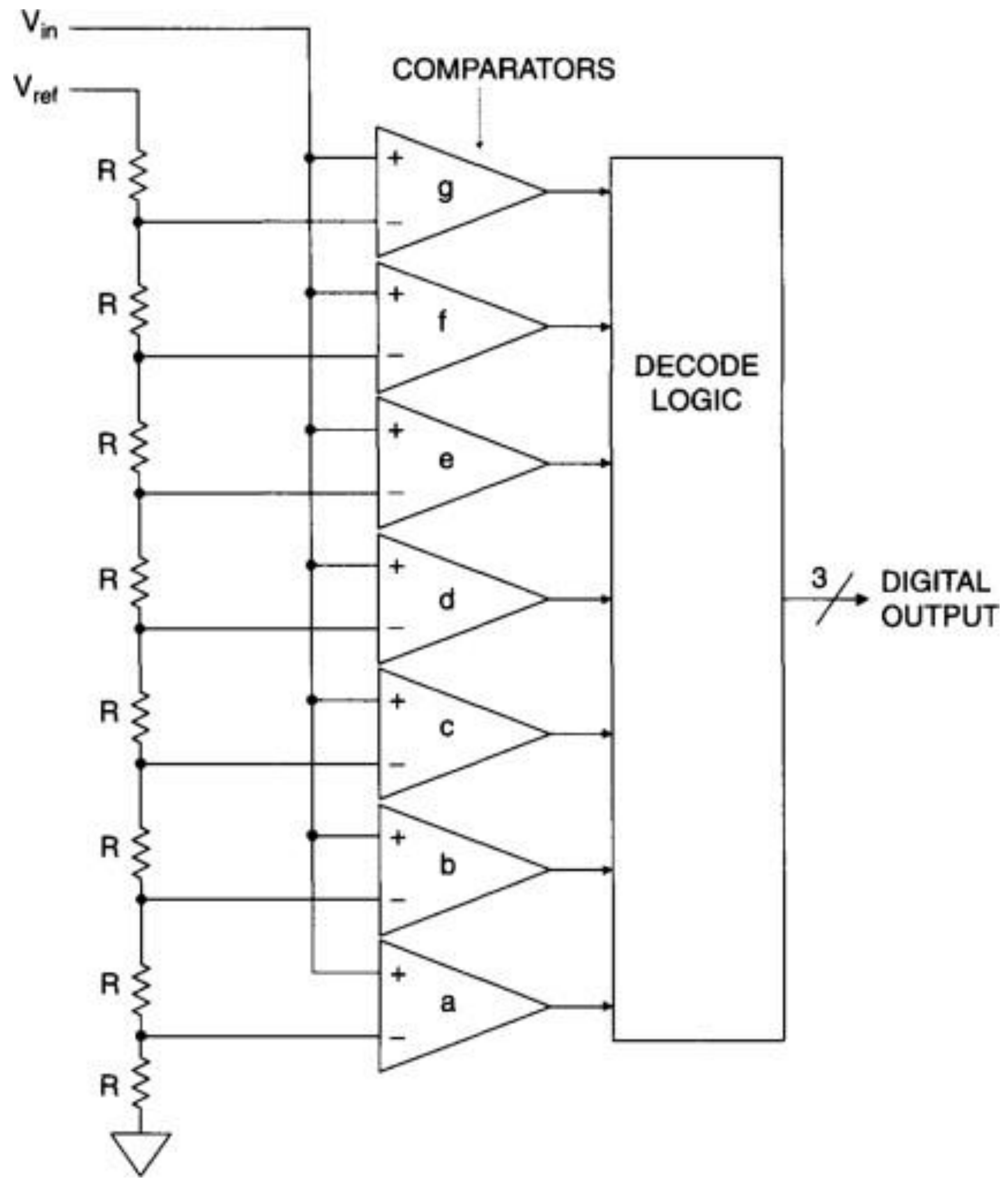
ADC – parallel/flash

- In case of the parallel ADC, the input voltage is compared with the V_{ref} divided into bins using a voltage divider
- The output of the comparators depends on the level of the input voltage with respect to these bins
- This output is decoded into the digital output



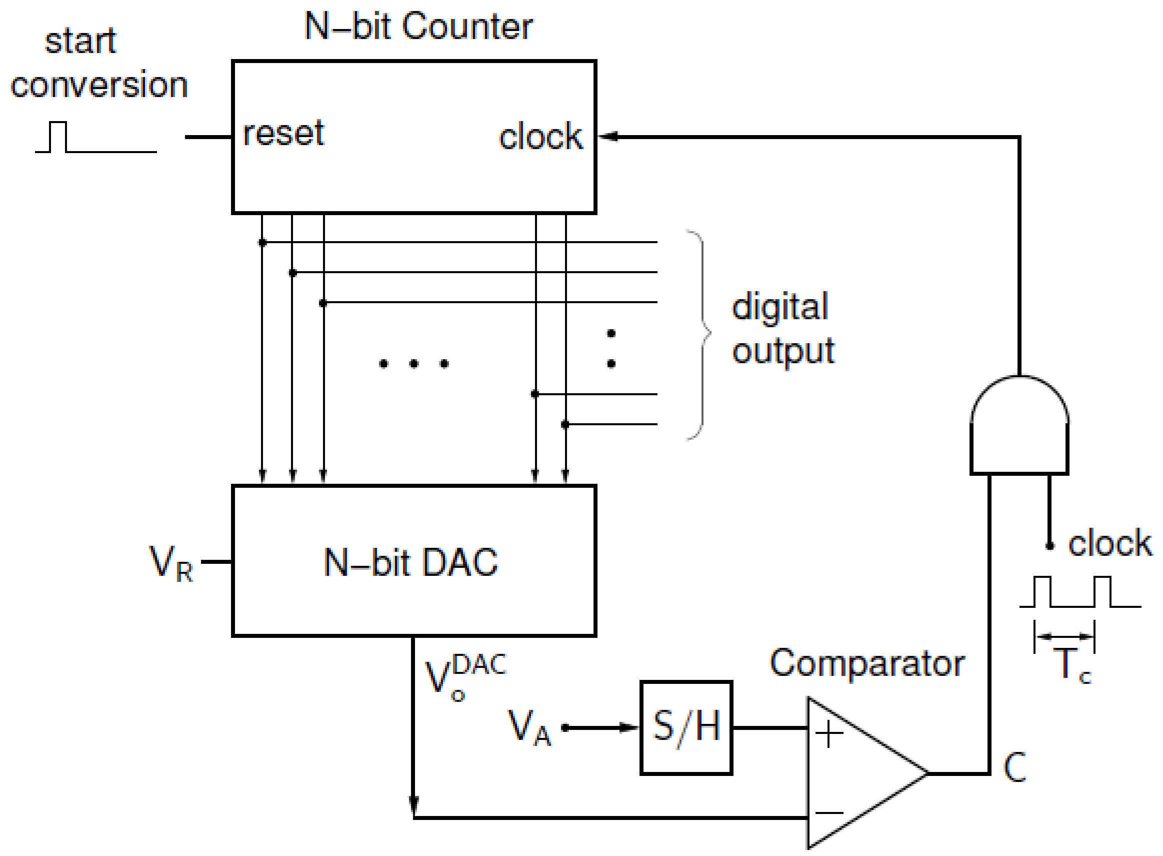
ADC – parallel/flash

- Advantages:
  - Speed – the ADC is not called flash for nothing! Flash ADCs handling 10+ Gsps are commercially available
- Disadvantages:
  - Number of comparators and resistors is 2^n
  - Static power is consumed because V_{ref} is continuously subjected to voltage divider
  - The comparators may not settle to the correct output value together for a changing input – leading to error in output



ADC – Ramp type (or counting)

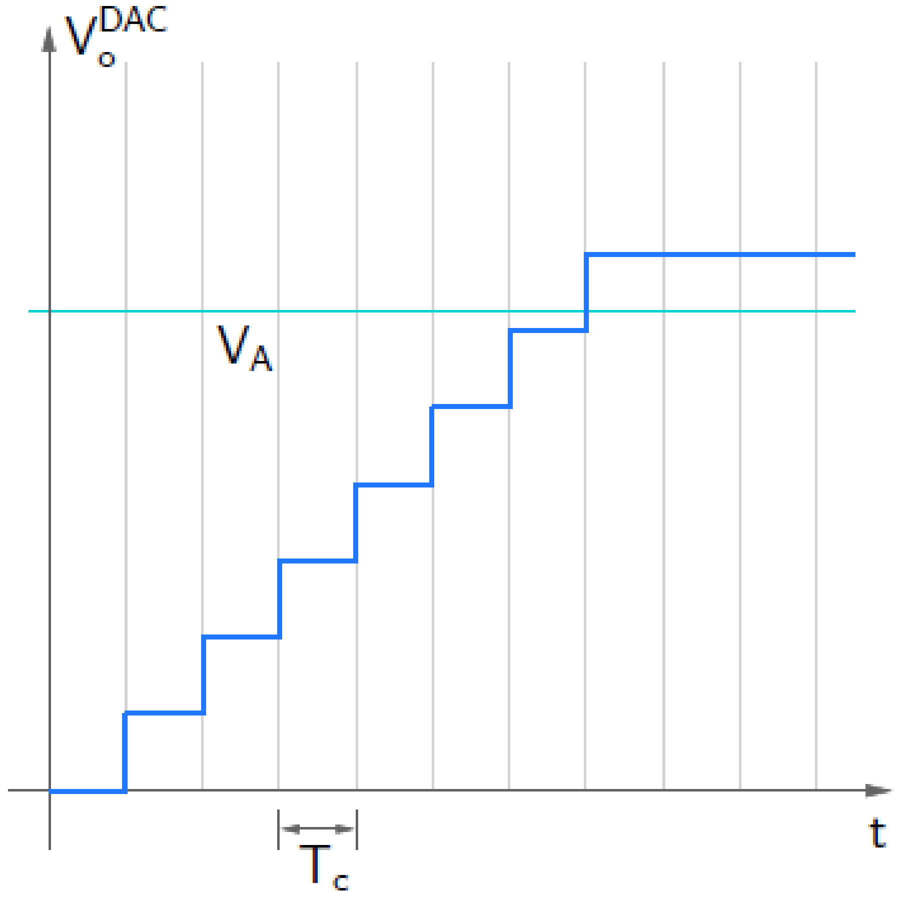
- An interesting way of making an ADC is using an internal DAC to compare the output with the input signal
- We start a digital counter at the start of every conversion
- The digital counter output is converted into analog and compared with the input signal
- When the comparison just becomes high, the counter is stopped
- The output of the counter is the digital equivalent of the analog input



ADC – Ramp type (or counting)

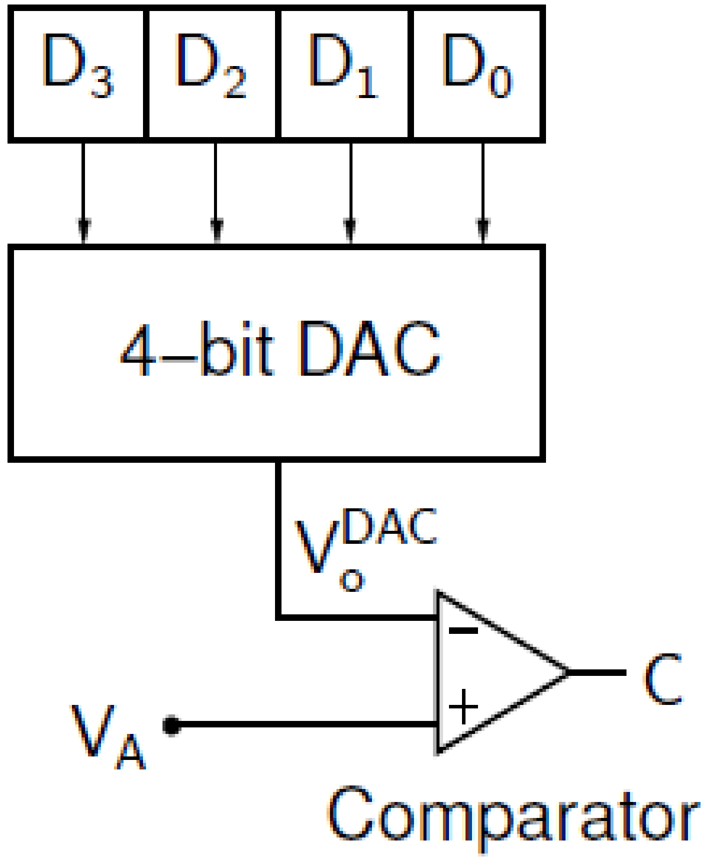
- Advantages:
  - Simpler circuit compared to the flash ADC for large value of n

- Disadvantages:
  - Requires a DAC
  - Is very slow – in worst case, requires 2^n clock cycles to complete. On average 2^{n-1}. This reduces the sampling frequency

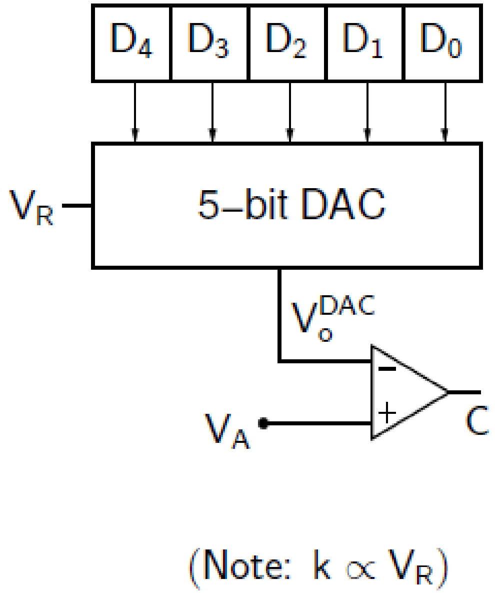


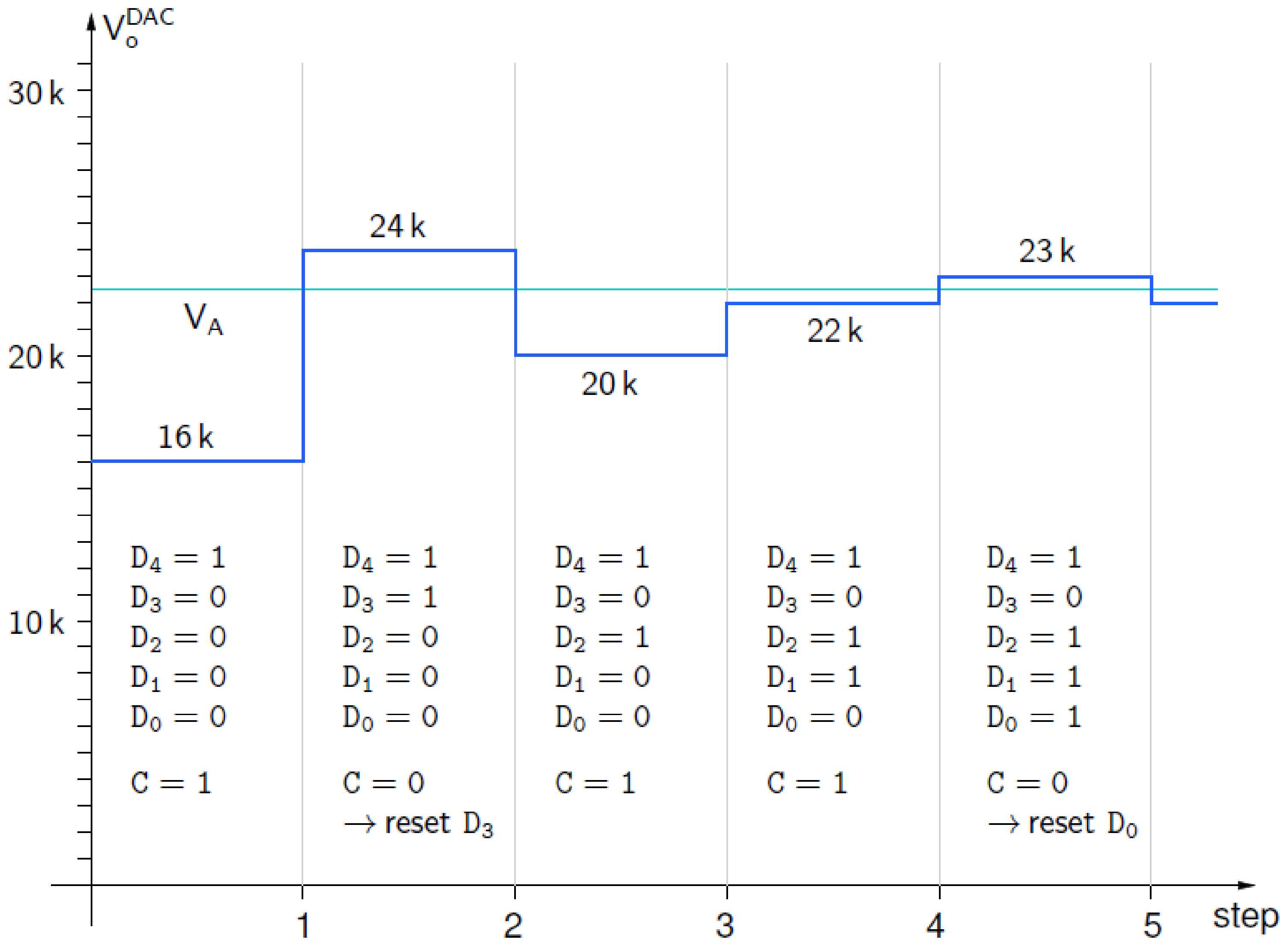
ADC – Successive approximation

- We can use a DAC and adjust one bit at a time to obtain the correct digital output
- Suppose we have a 4-bit DAC
  - Start with $D_3D_2D_1D_0$ = 0000
  - Set MSB to 1 (D_3 = 1) keeping other bits unchanged
  - If $V_{DAC} > V_A$, set D_3 back to 0, else keep D_3 at 1
  - Repeat these steps for successively lower bits
- At the end of four steps, the digital output is given by $D_3D_2D_1D_0$



ADC – Successive approximation





Sensors outputs - digital

Dr. Aftab M. Hussain,

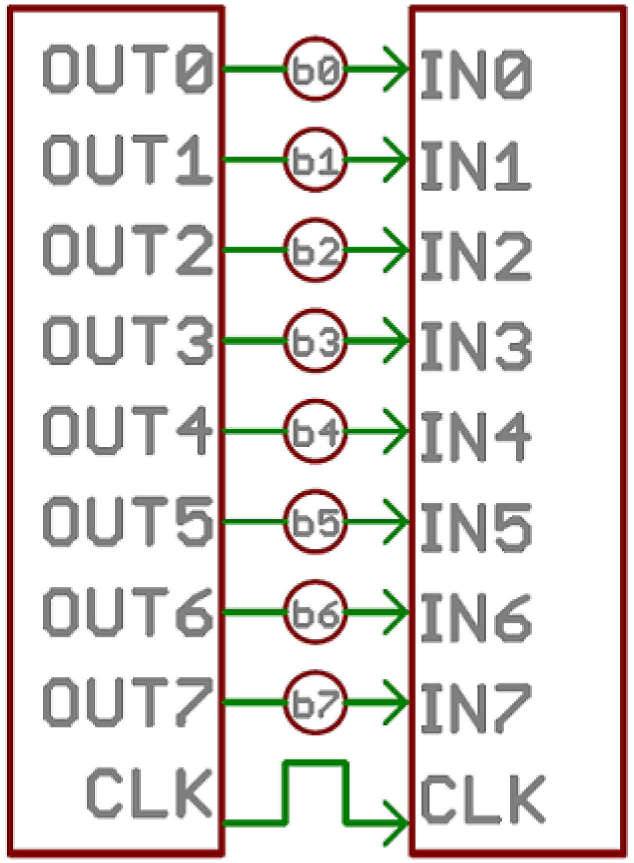
Assistant Professor, PATRIoT Lab, CVEST

Sensor outputs – Digital

- Digital communication is preferred over analog because it is less susceptible to noise
- There are multiple ways in which you can obtain digital
- Parallel – with each bit on a separate wire
- Serial – with bit transmitted one after the other
- In serial communication we can different protocols:
  - UART (asynchronous)
  - SPI (synchronous)
  - I2C (synchronous)

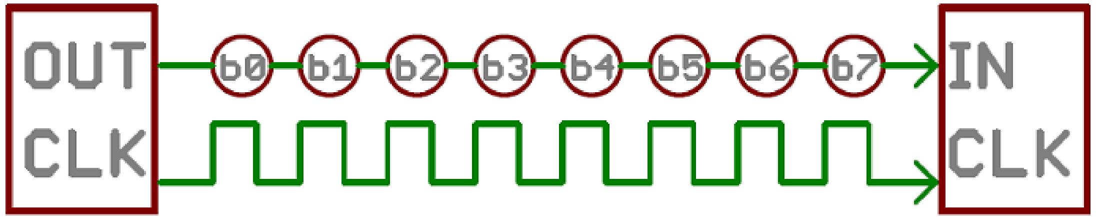
Sensor outputs – Digital – Parallel

- Parallel interfaces transfer multiple bits at the same time
- They usually require **buses** of data - transmitting across eight, sixteen, or more wires
- Advantages:
  - Very high data rates (single clock transfer)
  - Easy to implement
- Disadvantages:
  - Large number of data lines required, specially if number of peripherals are large



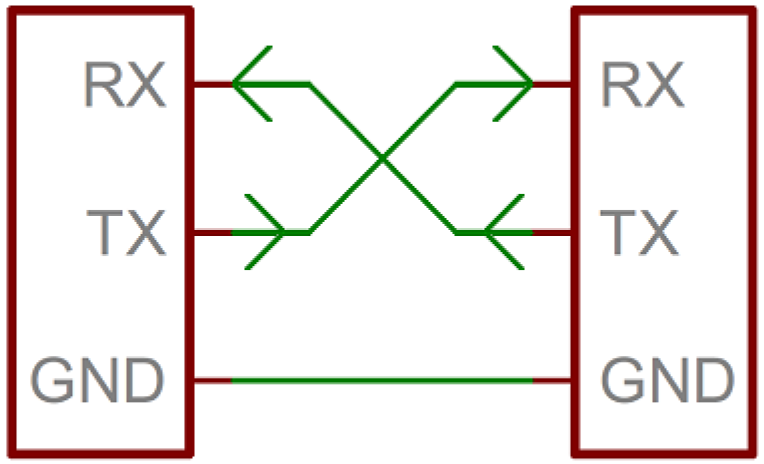
Sensor outputs – Digital – Serial

- Serial interfaces stream their data, one single bit at a time
- These interfaces can operate on as little as one wire
- Serial interfaces can be synchronous and asynchronous
- A synchronous serial interface always pairs its data line(s) with a clock signal, so all devices on a synchronous serial bus share a common clock
- Asynchronous means that data is transferred without support from an external clock signal



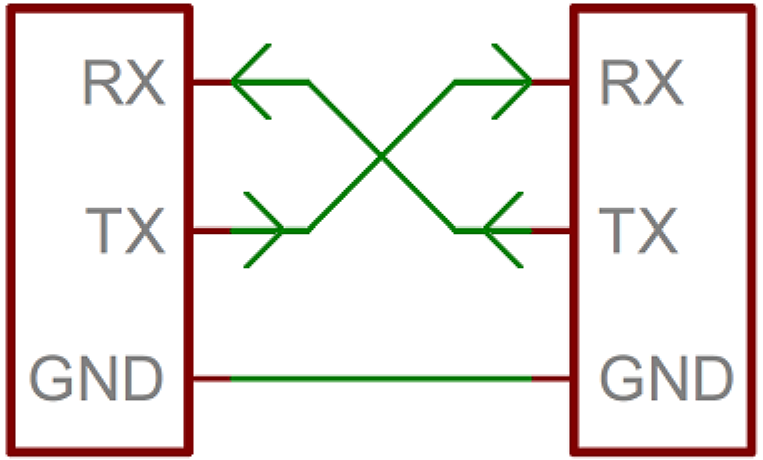
Sensor outputs – Digital – UART

- A universal asynchronous receiver/transmitter (UART) is a serial communication protocol that employs two lines Tx and Rx for communication
- UART support is commonly found inside microcontrollers
- For example, the Arduino Uno - based on the "old faithful" ATmega328 - has just a single UART, while the Arduino Mega - built on an ATmega2560 - has a whopping four UARTs
- NodeMCU has two UARTs



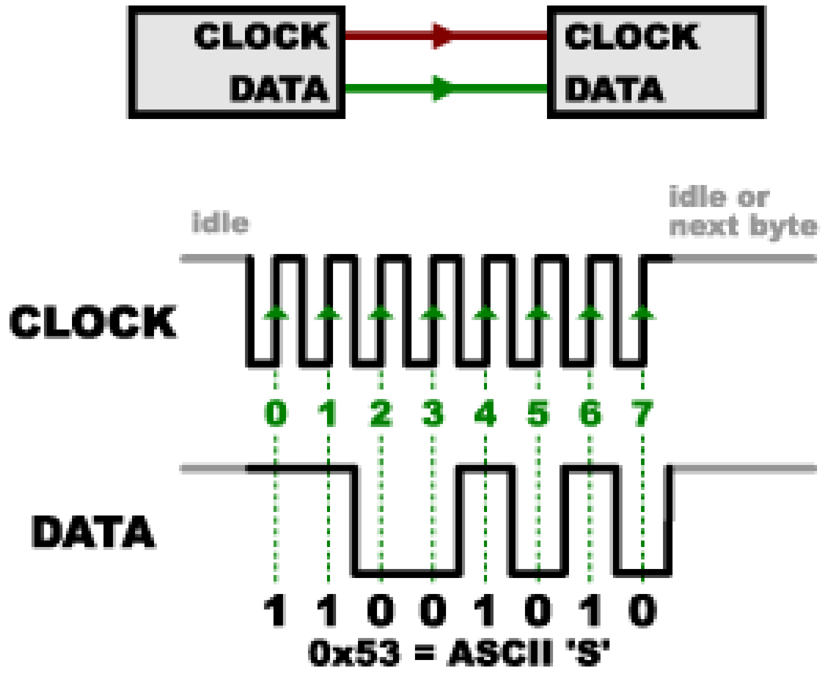
Sensor outputs – Digital – UART

- Advantages:
  - Two line communication
  - Simple to implement in software
  - Legacy protocol
- Disadvantages:
  - No synchronization means we have to make “baud rates” equal manually before communication
  - Low data rate – general baud rate is 9600 bits per second
  - Hardware implementation is complex
  - Needs start and stop bits to sync – which can be wasteful
  - Rx and Tx pins can be very confusing!



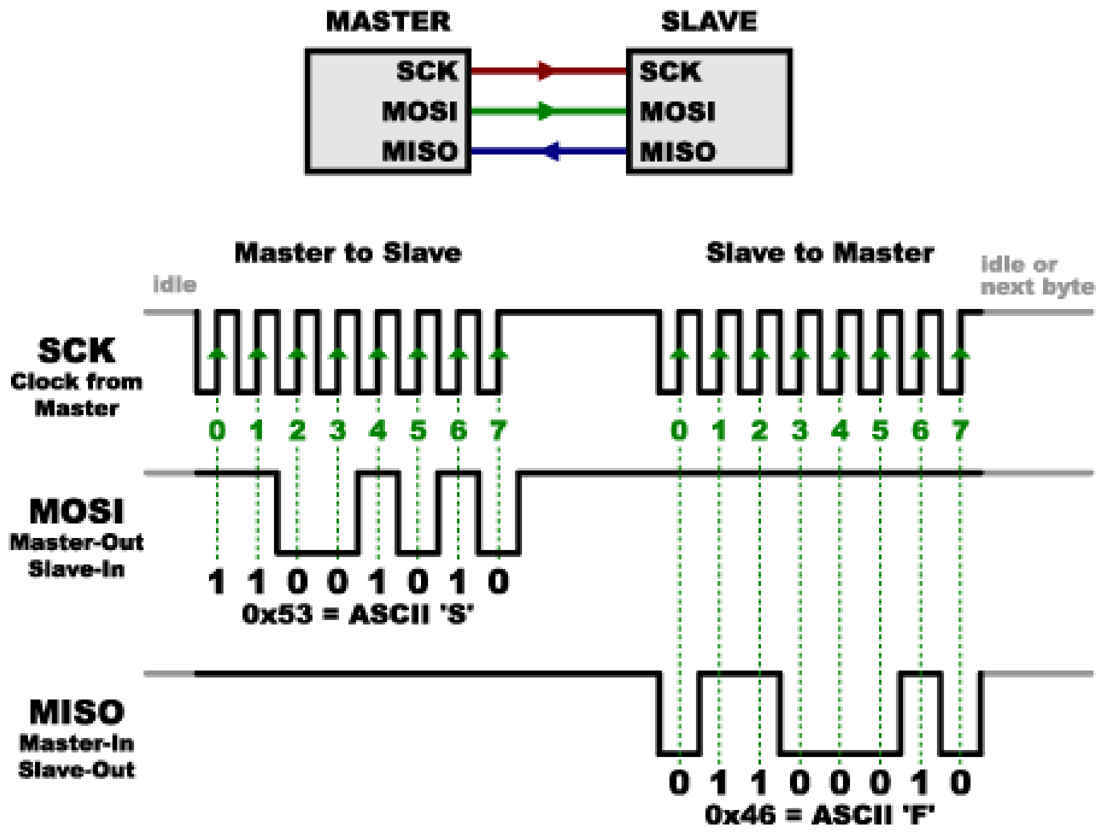
Sensor outputs – Digital – SPI

- SPI is serial peripheral interface
- It's a "synchronous" data bus, which means that it uses separate lines for data and a "clock" that keeps both sides in perfect sync
- The clock is an oscillating signal that tells the receiver exactly when to sample the bits on the data line
- When the receiver detects that edge, it will immediately look at the data line to read the next bit
- Because the clock is sent along with the data, specifying the speed isn't important, although devices will have a top speed at which they can operate



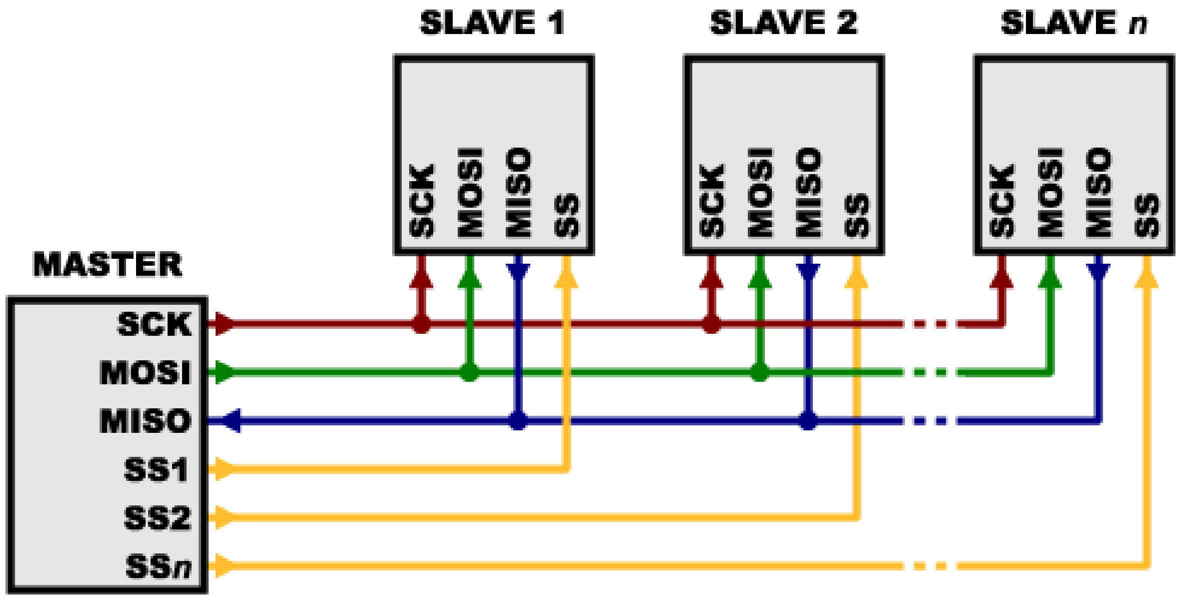
Sensor outputs – Digital – SPI

- We can also configure SPI for duplex communication
- In SPI, only one side generates the clock signal (usually called CLK or SCK for Serial ClocK)
- The side that generates the clock is called the "master", and the other side is called the "slave"
- When data is sent from the master to a slave, it's sent on a data line called MOSI, for "Master Out / Slave In"
- If the slave needs to send a response back to the master, the master will continue to generate a prearranged number of clock cycles, and the slave will put the data onto a third data line called MISO, for "Master In / Slave Out"



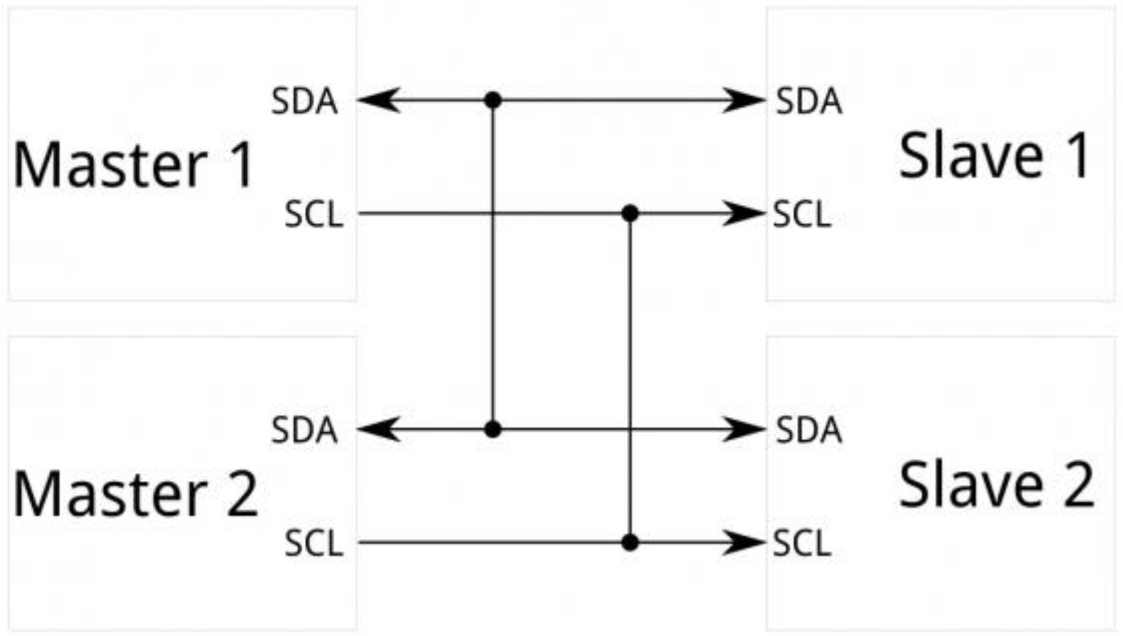
Sensor outputs – Digital – SPI

- Lastly, we can configure SPI for multiple slaves using the same lines for Sclk, MOSI and MISO, but different “slave select” lines
- With this, the slave with its slave select that is enabled will communicate with the master on the same bus, while the others await their turn
- SPI has lots of advantages:
  - Its synchronous so no prearranged baud rates and no start/stop bits
  - Multiple devices on a single bus
- Disadvantages:
  - Too many lines in case of many slaves
  - Only one master per bus



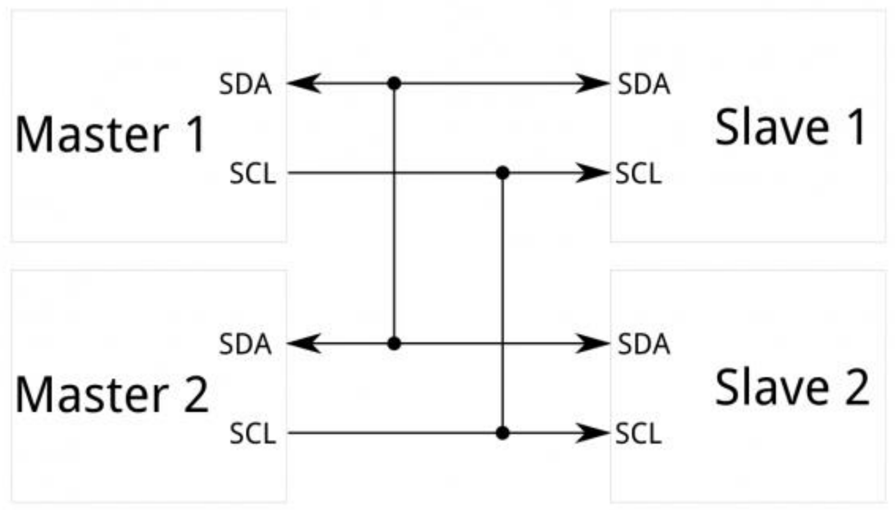
Sensor outputs – Digital – I2C

- The Inter-integrated Circuit (I^2C or I2C) Protocol is a protocol intended to allow multiple slaves to communicate with one or more "master" chips
- I^2C requires a mere two wires, like asynchronous serial, but those two wires can support up to 1008 slave devices
- Also, unlike SPI, I^2C can support a multi-master system, allowing more than one master to communicate with all devices on the bus



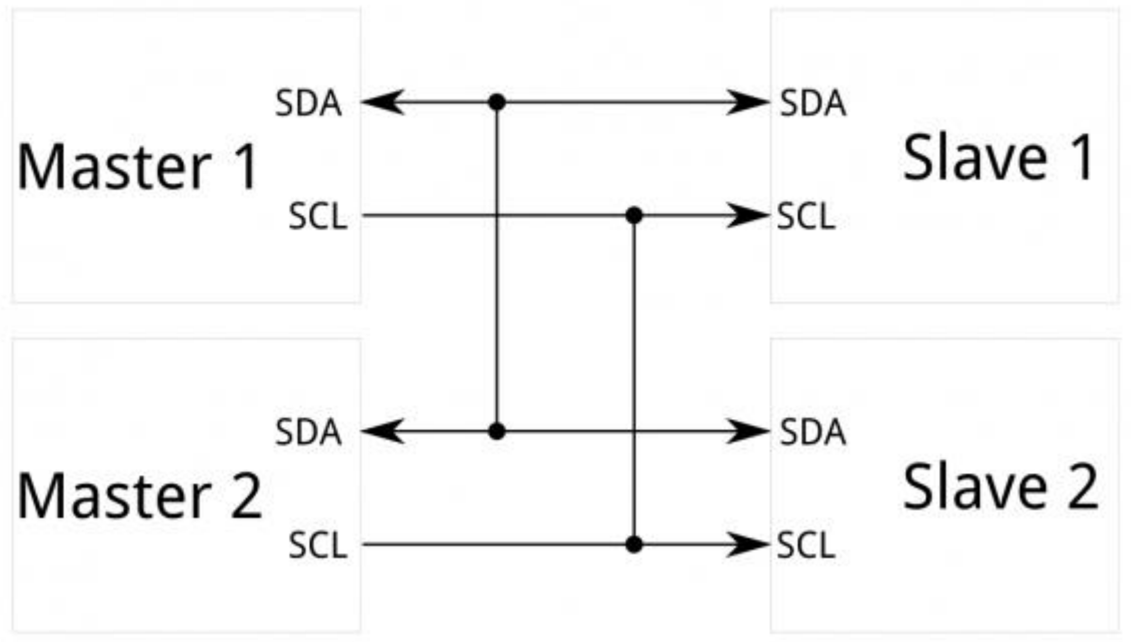
Sensor outputs – Digital – I2C

- Each I^{2}C bus consists of two signals: SCL and SDA. SCL is the clock signal, and SDA is the data signal
- The clock signal is always generated by the current bus master
- Unlike UART or SPI connections, the I2C bus drivers are "open drain", meaning that they can pull the corresponding signal line low, but cannot drive it high
- Thus, there can be no bus contention where one device is trying to drive the line high while another tries to pull it low
- Each signal line has a pull-up resistor on it, to restore the signal to high when no device is asserting it low



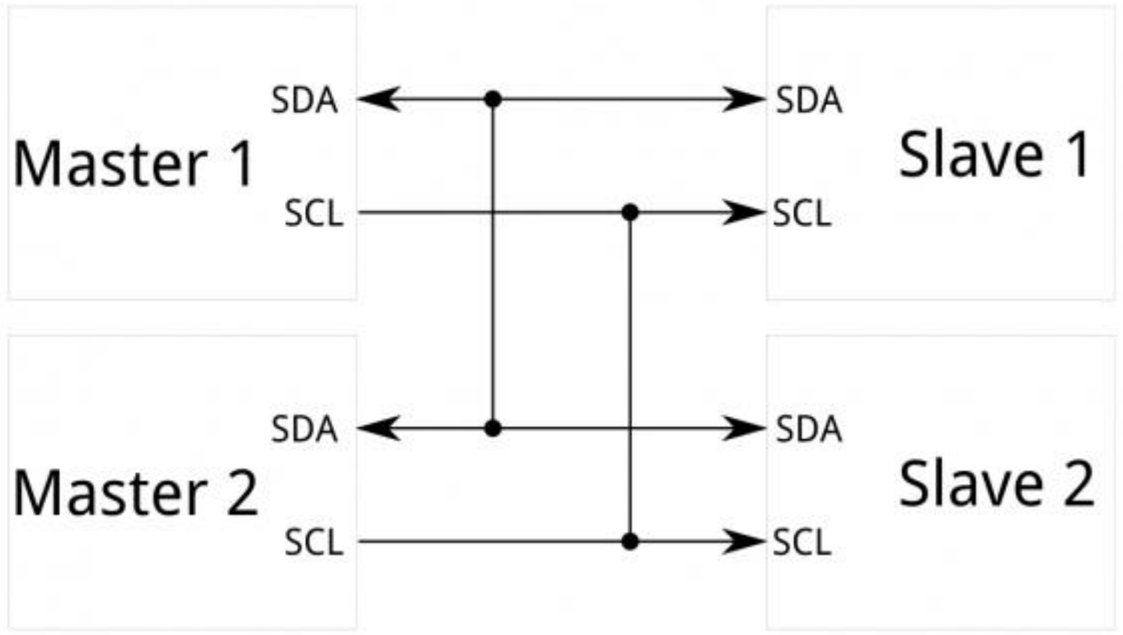
Sensor outputs – Digital – I2C

- Because the devices on the bus don't actually drive the signals high, I^{2}C allows for some flexibility in connecting devices with different I/O voltages
- In general, in a system where one device is at a higher voltage than another, it may be possible to connect the two devices via I^{2}C without any level shifting circuitry in between them
- The trick is to connect the pull-up resistors to the lower of the two voltages
- Although this only works in cases where the lower of the two system voltages exceeds the high-level input voltage of the the higher voltage system - for example, a 5V Arduino and a 3.3V peripheral



Sensor outputs – Digital – I2C

- In practice, most I2C peripherals have a defined address – or changeable address based on some external hardware pins
- The device address is first put on the SDA after the SCL is activated so that the correct slave can listen and respond
- Devices are addressed using a 10-bit address with a total of 1008 addresses possible
- In practice, if more than one I2C peripheral is to be connected, make sure there is only one pull up resistance for the complete bus


Master 1
SDA
SCL
Slave 1
SDA
SCL
Master 2
SDA
SCL
Slave 2
SDA
SCL