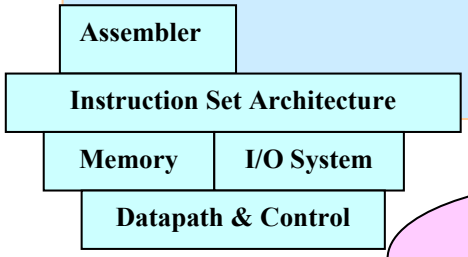
ELEC 3300
Introduction to Embedded Systems

Topic 8

Digital and Analog Interfacing, Motor Interfacing

Prof. Tim WOO

Course Overview



Introduction to Embedded Systems

More about Embedded Systems

Basic Computer Structure

MCU Main Board

Digital and Analog Interfacing

USART, IEEE1394, USB, I2C and SPI

Microcontroller Structure

A/D Port

Serial Port

CPU

External Memory Port

External Interrupt Port

Interrupt Organization

Timer and Counter

External Timer Port

Simple I/O Port

Buffering and Direct Memory Access (DMA)

Memory, Interfacing to Memory, Memory Timing and applications

Interfacing LCD

Motor Interfacing

In this course, STM32 is used as a driving vehicle for delivering the concepts.

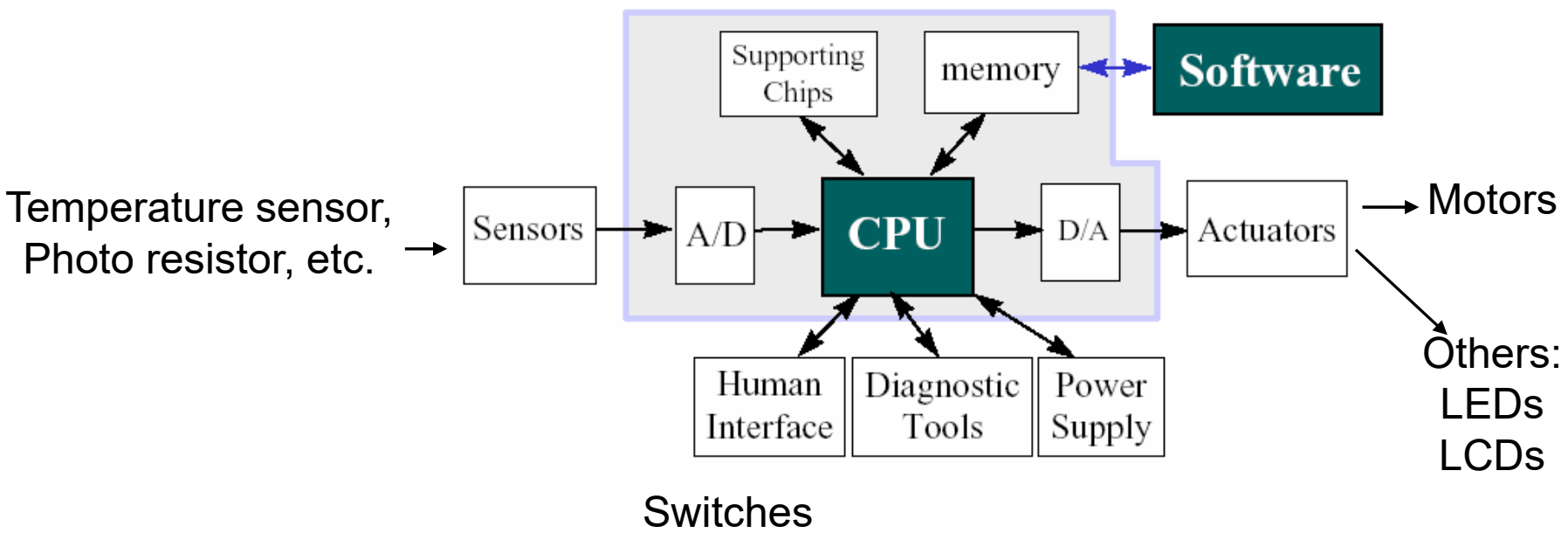
To be covered	In progress	Done

Expected Outcomes

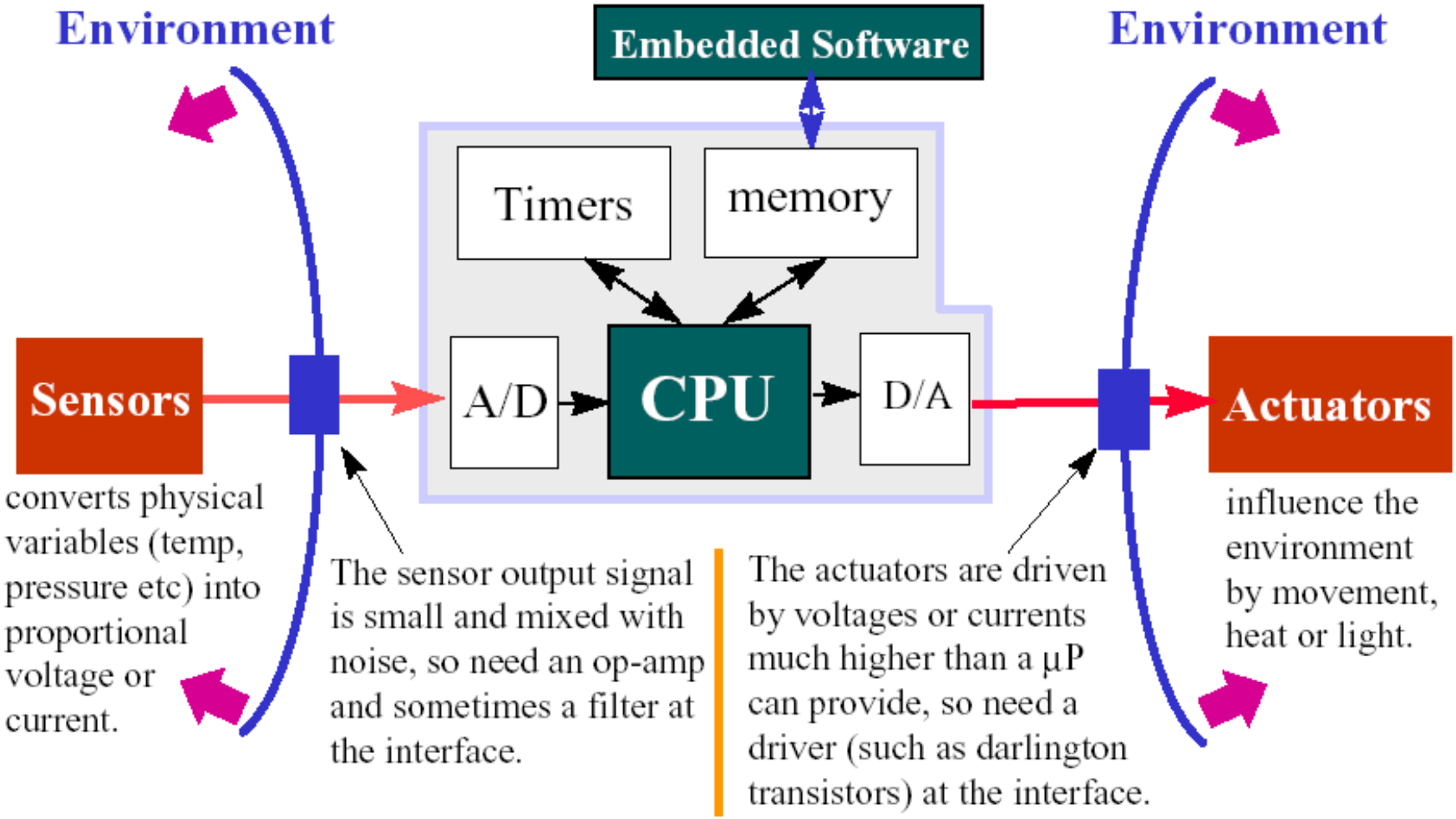
- On successful completion of this topic, you will be able to
  - Describe the basic concepts of Analog-to-Digital Conversion (ADC) and Digital-to-Analog Conversion
  - List the different types of motors and their characteristics
  - Implement the drivers for interfacing motors with microprocessor
    - Pulse Width Modulation (PWM) for driving DC motor without the use of DAC circuitry
    - Operation Principles for driving a stepper motor
    - Servo motor

Sensing and Controlling the Environment

- To sense and control the environment, we need to interface the microcontroller to peripheral devices
- Variables we want to measure and/or control
  - Light, temperature, sound, pressure, humidity, etc

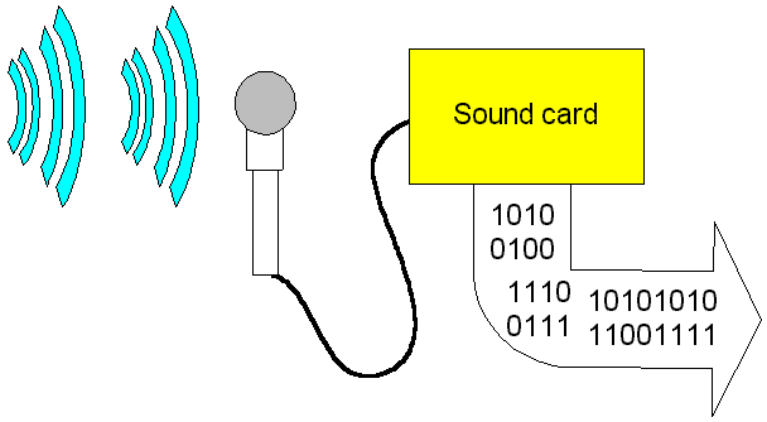


A Typical Embedded System



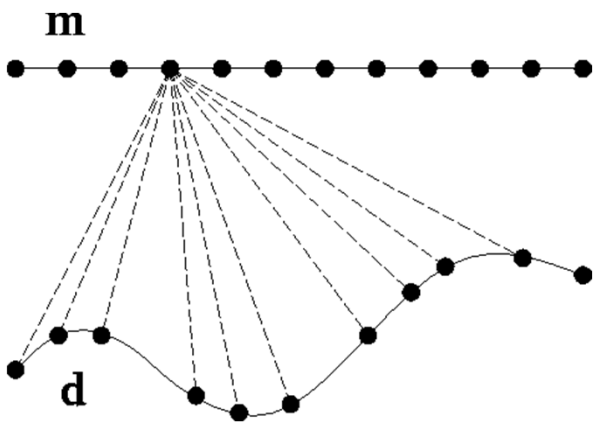
DAC output (STM32) = **20 mA** whereas the operating current of a typical the motor of a 1.5 ton Aircon is **7 A.** Starting current of a motor is about 2-2.5 times normal operating current (14A – 17.5A).

Considerations?



Operating range of the input signal

Regular / Irregular Sampling



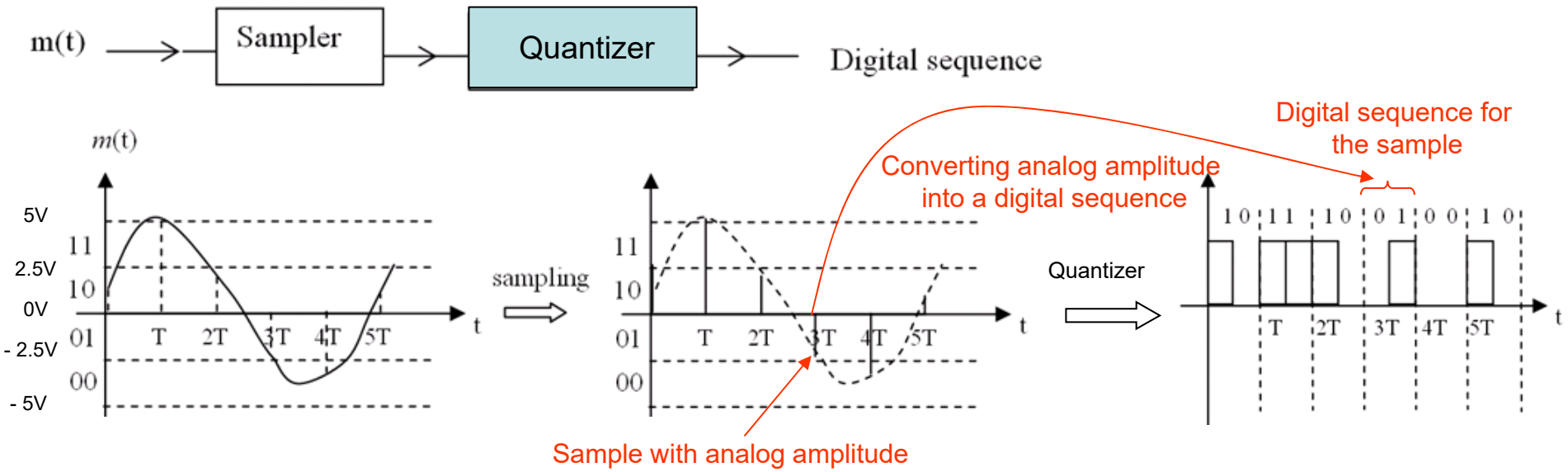
Sampling rate

Operating range of A/D converter

Resolution

Analog-to-Digital Conversion

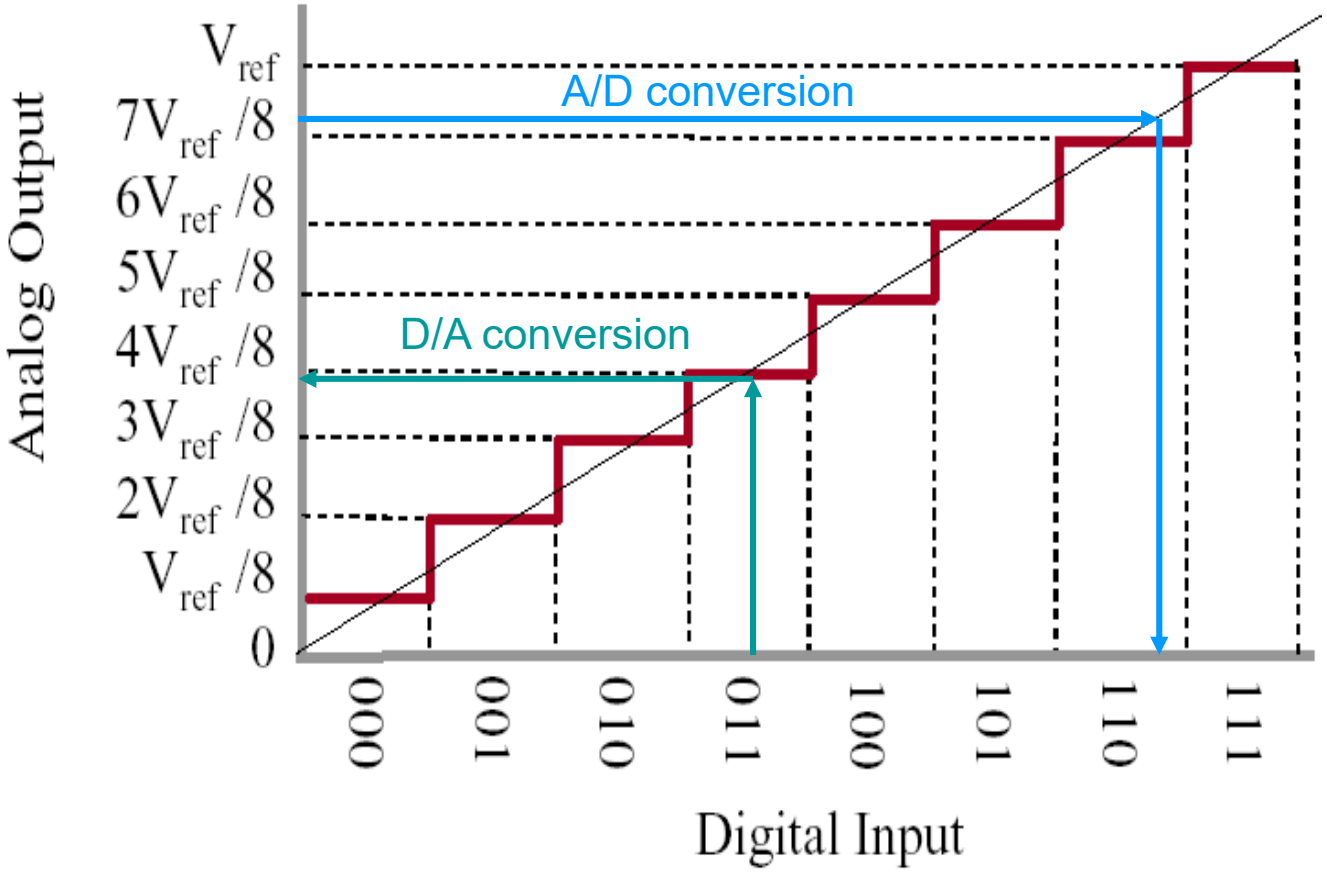
- Map analog inputs to a range of binary values
  - Consists of two buildings blocks : **Sampler** and **Quantizer**



Data rate = number of bits / total time

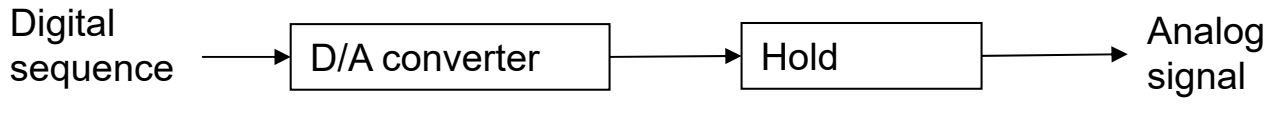
Concept of Analog-to-Digital Conversion

- Map analog outputs to binary values (linear quantization)



Digital-to-Analog Conversion

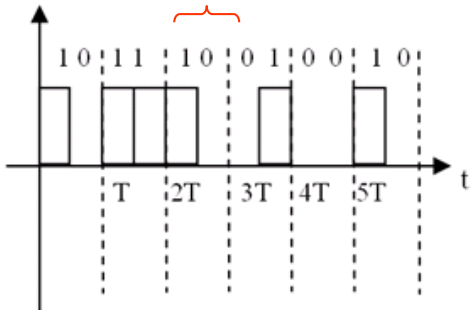
- Map digital inputs to a range of analog values, and then ….



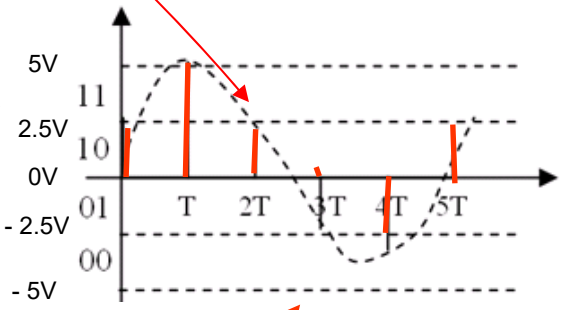
Zero order: y = c (reconstruct samples using horizontal lines)

Converting a digital sequence into analog amplitude

Digital sequence for the sample



A/D converter

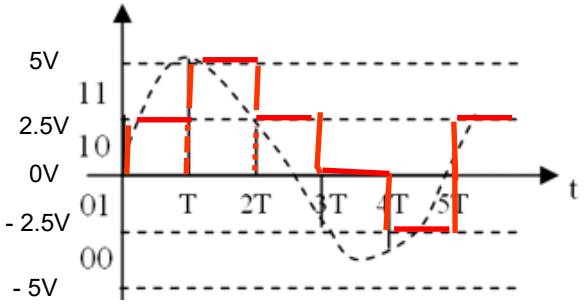


Sample with analog amplitude

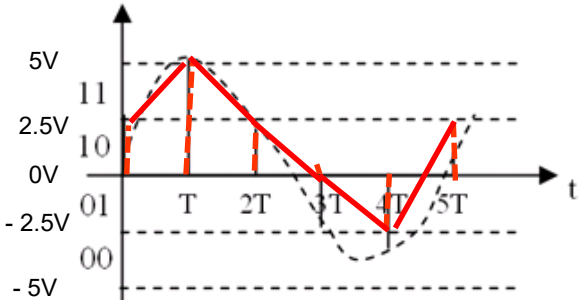
Digital value	Mapped Analog Voltage (e.g.)
11	5V
10	2.5V
01	0V
00	-2.5V

The quantization step = ?

Zero-order hold



First-order hold



First order: y = mx+c

Second order: y = a x^2+bx+c

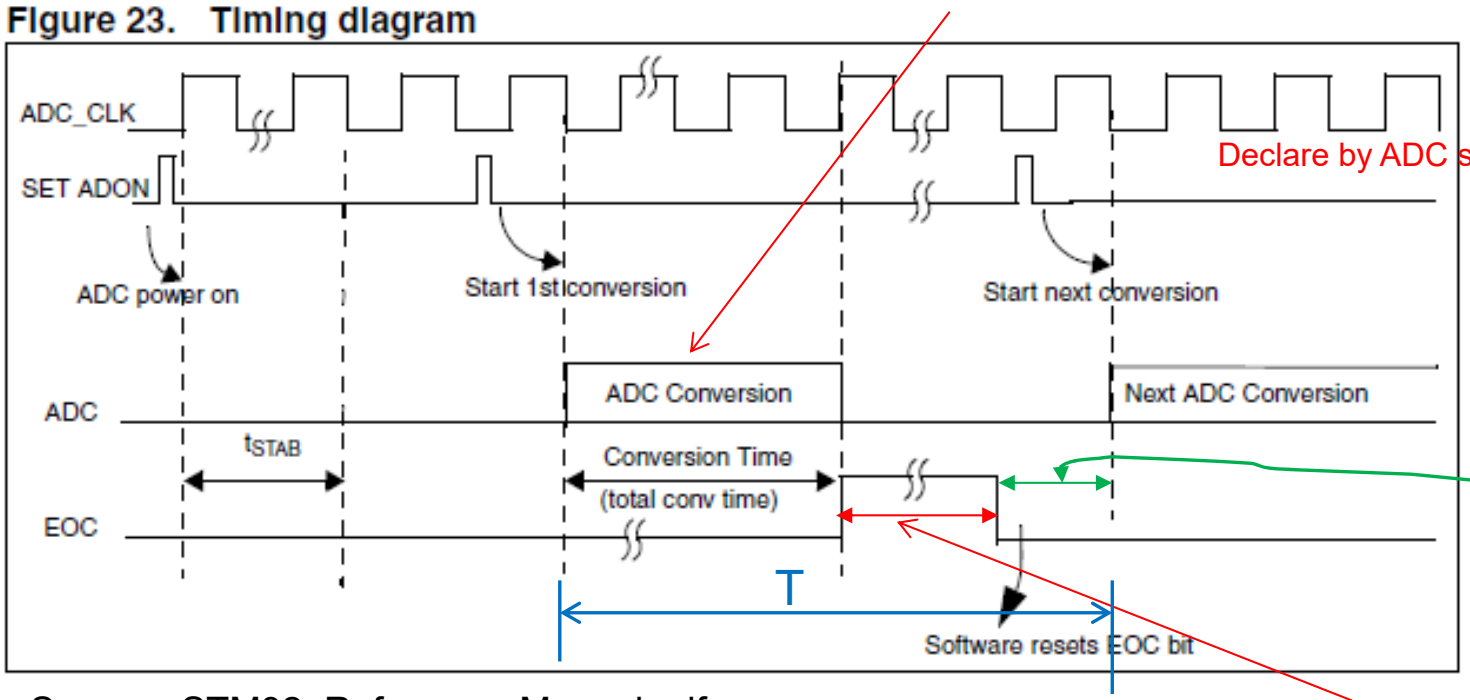
Analog-to-Digital Conversion

- A typical timing diagram of ADC process:

Remark: STM32's terminology is 'sampling time', which is not same as 'T'.

Total conversion time = conversion time + 12.5 cycles

Declare by ADC sample time register ≠ T



Source : STM32_Reference_Manual.pdf

time to start of next ADC conversion.

Amount of additional delay

Sampling time T should be greater than that of A/D conversion time

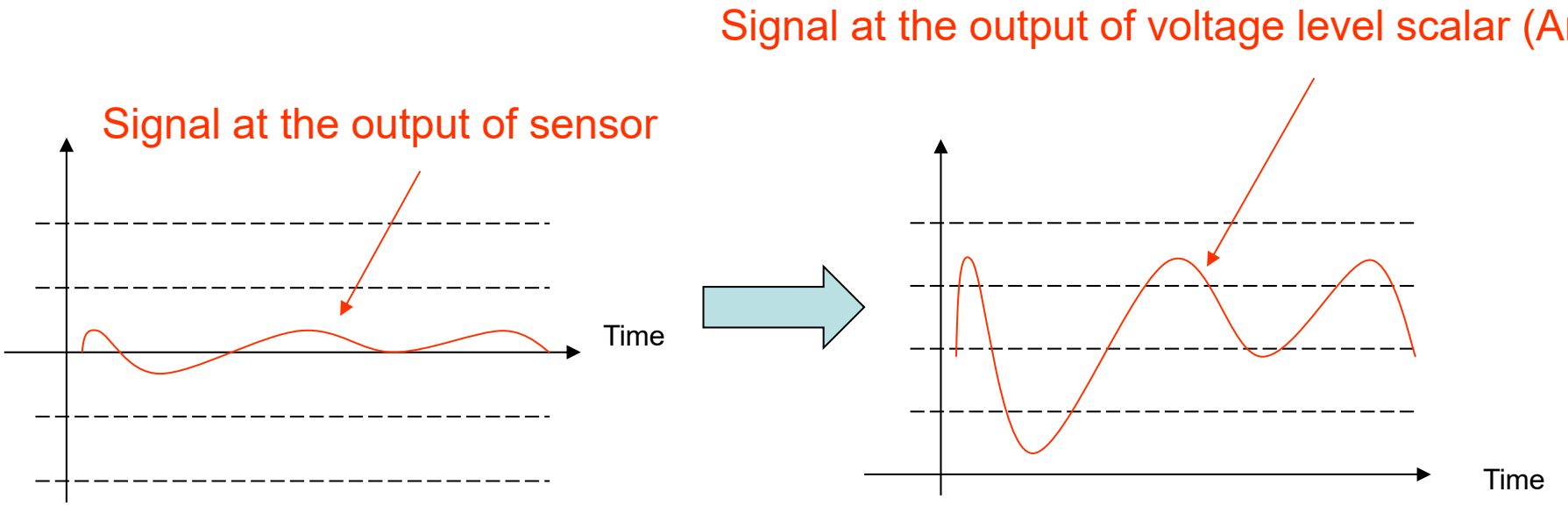
Additional delay (implemented using timer) 12.5 cycles = time accounted for interrupt activation, initiation of A/D conversion, pushing the values into registers, etc.

T = conversion time + delay (12.5 cycles) + time to start of next ADC conversion.

: p. 216 of STM32_Reference_Manual.pdf

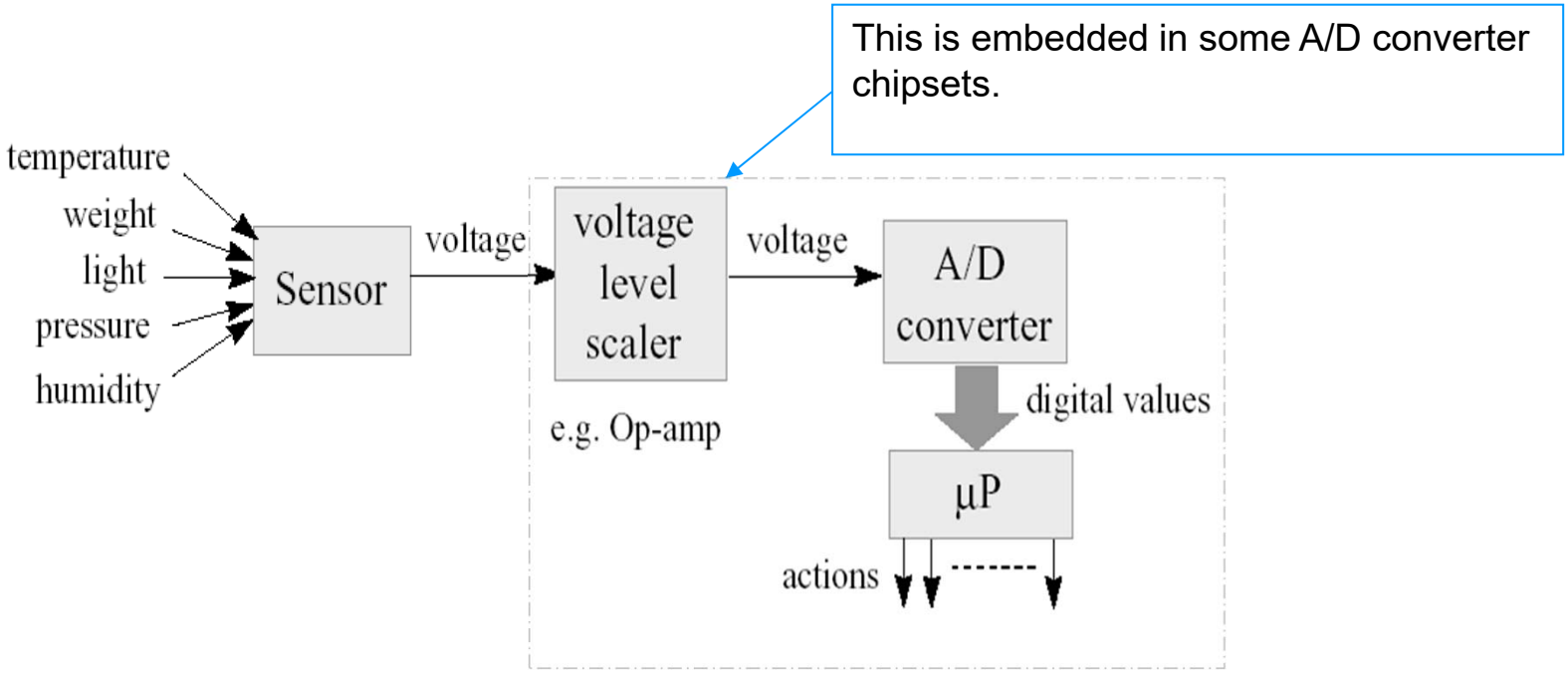
Analog-to-Digital Conversion

- Typical value of voltage at the output of sensor ranges from μV to mV
- Is it suitable to be the input signal of A/D converter ?

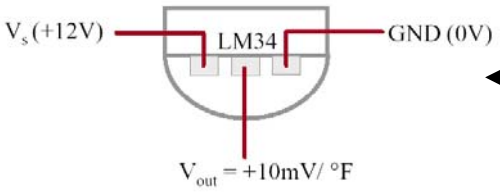


Analog-to-Digital Conversion

- Example: Voltage Level Scaler (Amplifier)

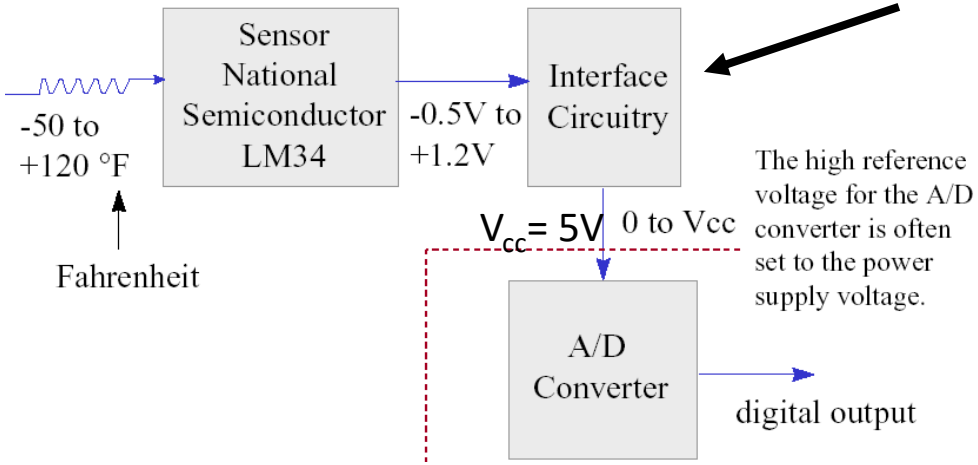


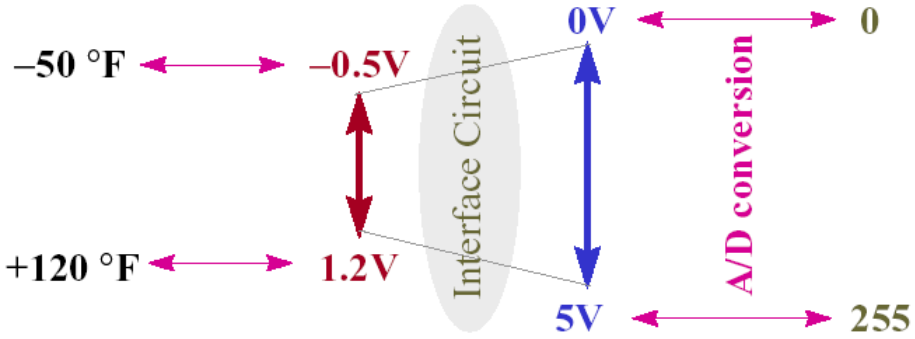
Example: Measuring the Temperature



- Senor output voltage increases 10mV for an increase of 1°F
- When 0V is for 0°F, the output voltage is from -0.5V to 1.2V for temperature range of -50 to +120°F

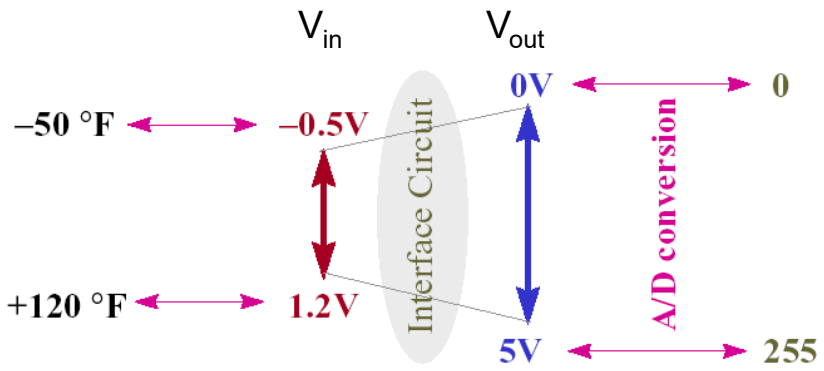
- At the interface, an analog circuit is needed to scale and shift the sensor output voltage of range -0.5V to 1.2V to between 0V to 5V

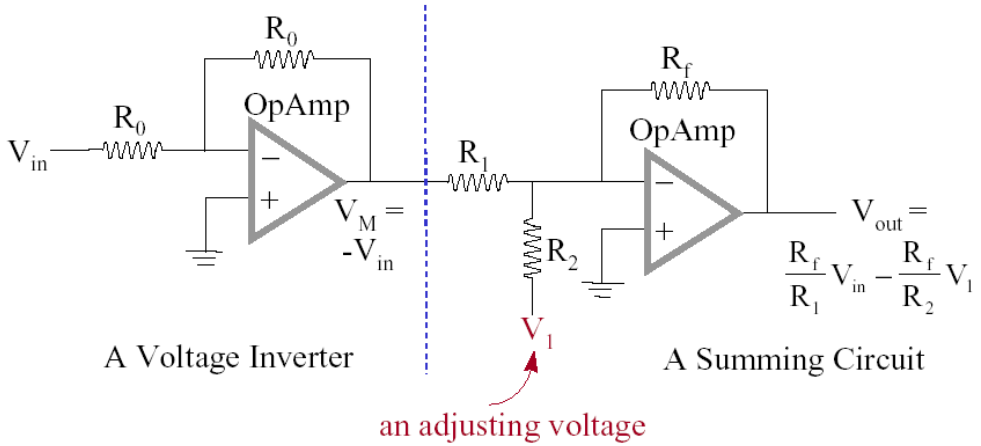


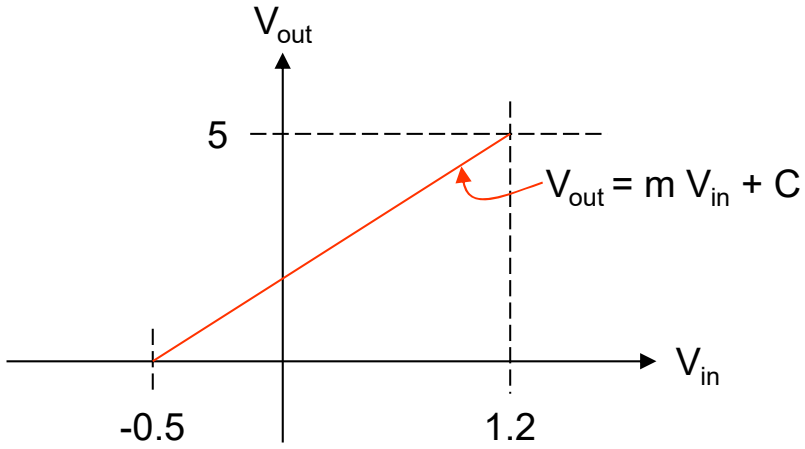


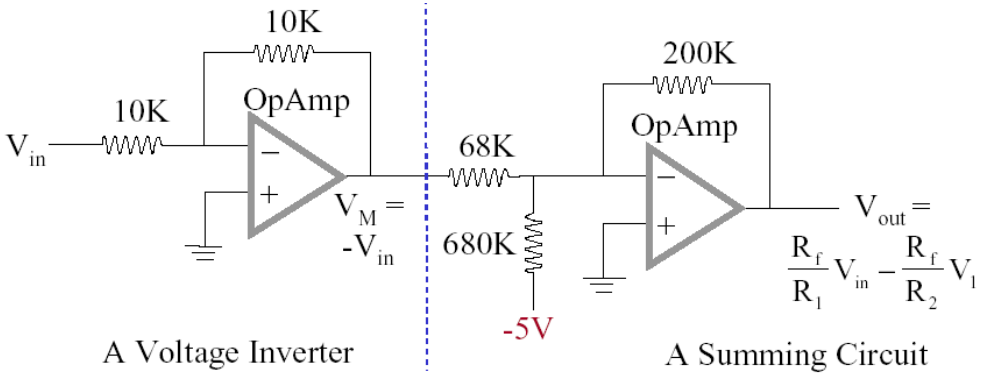
The Interface Circuitry - Schematic

- When on-chip voltage scaling and shifting is not supported in the A/D converter, the following circuitry is a way to achieve the task.





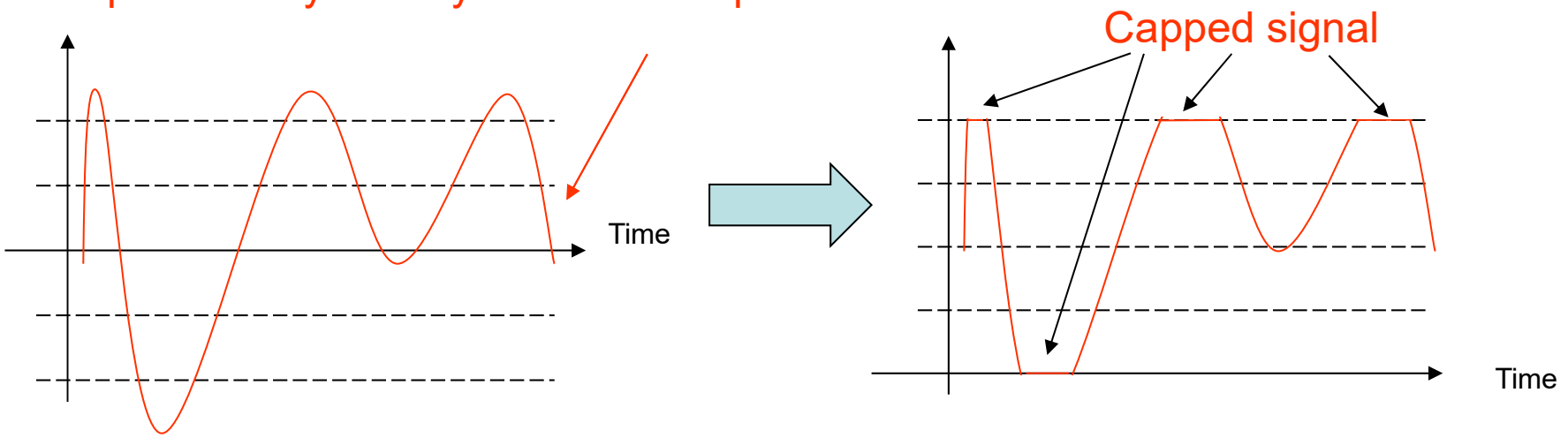




Analog-to-Digital Conversion

- If the amplified signal is too large, it may go beyond the operating range of ADC.
- Is it suitable to be the input signal of A/D converter ?

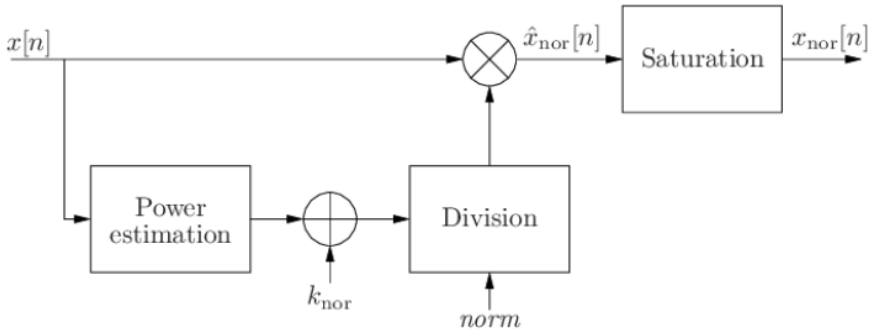
Signal at the input signal to sensors
e.g. You speak loudly when you use micro-phone.



Do you have any suggestions to avoid any distortion of signal?

Automatic Gain Control

- Objective: A device effectively reduces the volume if the signal is strong and raises it when it is weaker.
- Solution:
  - Automatic gain control (AGC) is a closed-loop feedback regulating circuit in an amplifier or chain of amplifiers, the purpose of which is to maintain a suitable signal amplitude at its output, despite variation of the signal amplitude at the input.
  - The average or peak output signal level is used to dynamically adjust the gain of the amplifiers, enabling the circuit to work satisfactorily with a greater range of input signal levels.



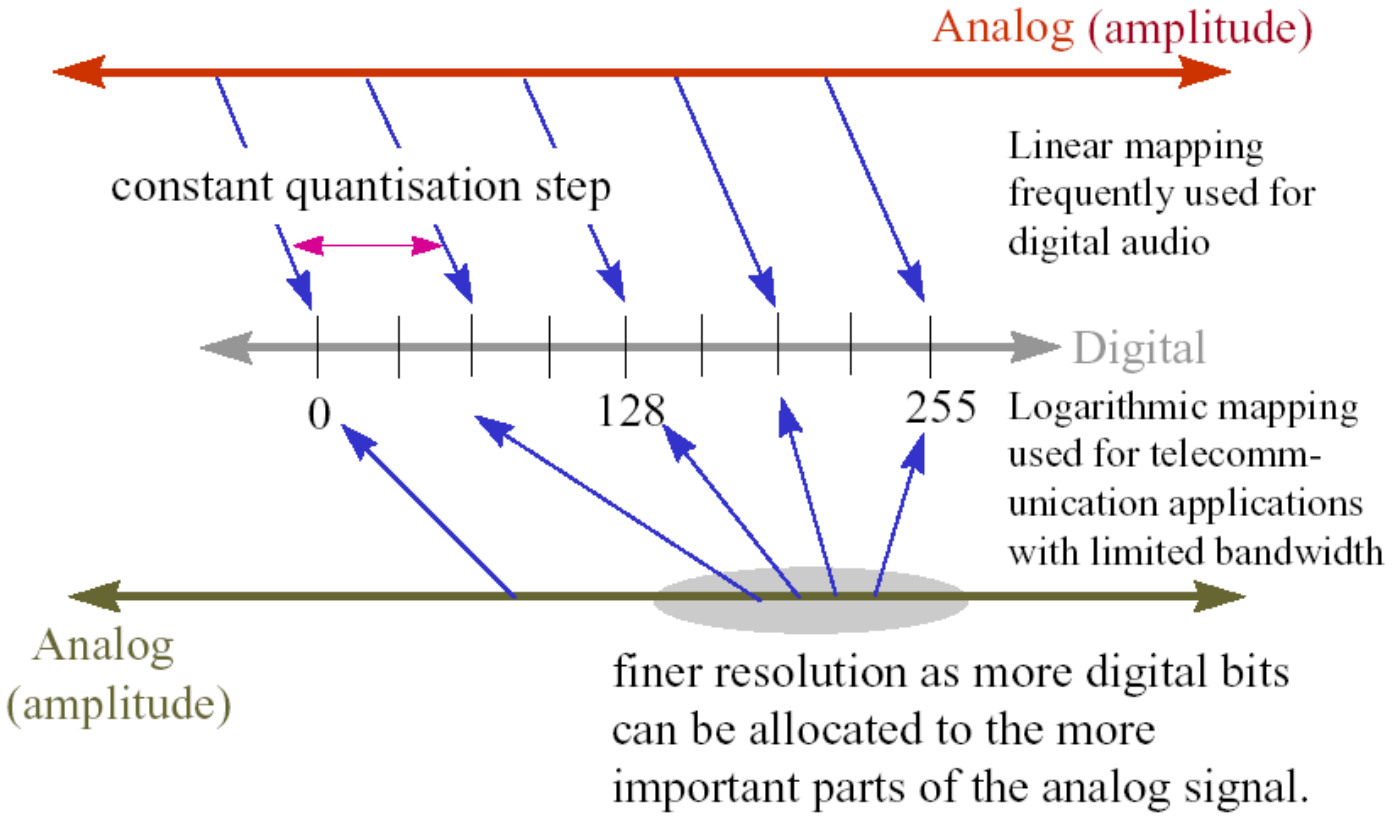
Source: https://www.researchgate.net/figure/Block-diagram-of-automatic-gain-control-AGC-system_fig3_221673769

Concept of Analog-to-Digital Conversion

- Two common mapping of the A/D conversion
  - Linear mapping : constant quantisation step
  - Logarithmic mapping : constant quantisation step in logarithmic scale

Log ADC:
Small analog amplitudes are quantized with fine resolution, while large amplitudes are quantized with coarse resolution.

Suitable for signals with a large dynamic range.



Main Types of ADCs

- Successive Approximation (SAR) ADC
- Delta-sigma (ΔΣ) ADC
- Dual Slope ADC
- Pipelined ADC
- Flash ADC

Refer to ‘Supplementary Slides’ (slides #54 to #57) for description of each ADC.

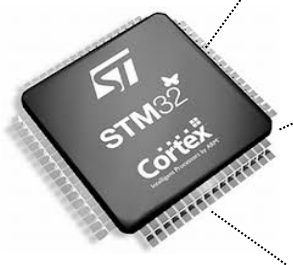
More information: https://dewesoft.com/daq/types-of-adc-converters#main-adc-types

ARM Microcontroller: High density pin definitions

Signal Type (Inside the MCU)

Physical Pin Assignment

Three Analog to Digital Converters



adc1

adc2

adc3

-ADC1_IN0
-ADC1_IN1
-ADC1_IN2
-…..
-ADC1_IN15

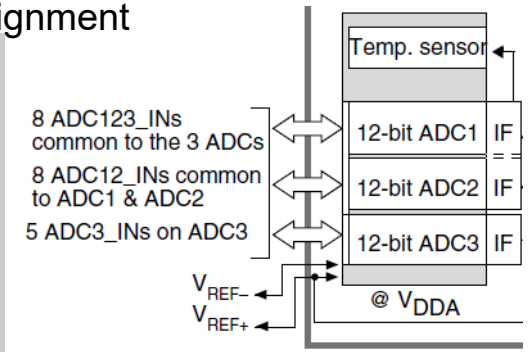
-ADC2_IN0
-ADC2_IN1
-ADC2_IN2
-…..
-ADC2_IN15

-ADC3_IN0
-ADC3_IN1
-ADC3_IN2
-ADC3_IN3
-ADC3_IN4
-ADC3_IN5
-ADC3_IN6
-ADC3_IN7
-ADC3_IN8
-…
-ADC3_IN13

ADC12 has 16 inputs
ADC3 has 13 inputs
analog signals

-PIN34
-PIN35
-PIN36
-PIN37
-PIN40
-PIN41
-PIN42
-PIN43
-PIN46
-PIN47
-PIN26
-PIN27
-PIN28
-PIN29
-PIN44
-PIN45

16 pins



12-bit ADC successive approximation ADC has up to 19 multiplexed channels, allowing measurement of up to 16 external sources and up to 3 internal sources.

Total:
21 pins are employed

-PIN18
-PIN19
-PIN20
-PIN21
-PIN22

5 pins in 144pin packages. Not available in 100pin package

ARM Microcontroller: High density pin definitions

Signal Type (Inside the MCU)

Physical Pin Assignment

Three Analog to Digital Converters

adc1

adc2

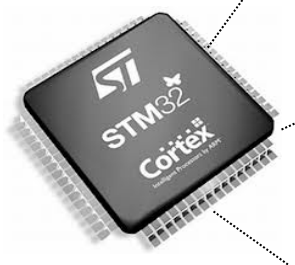
adc3

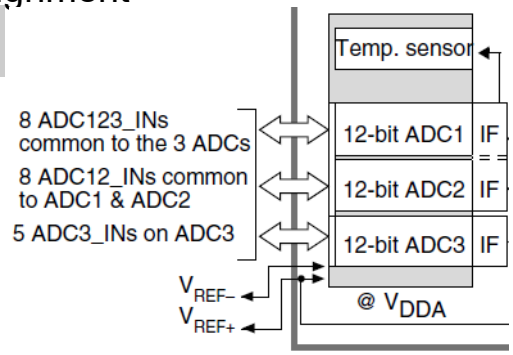
-ADC1_IN0

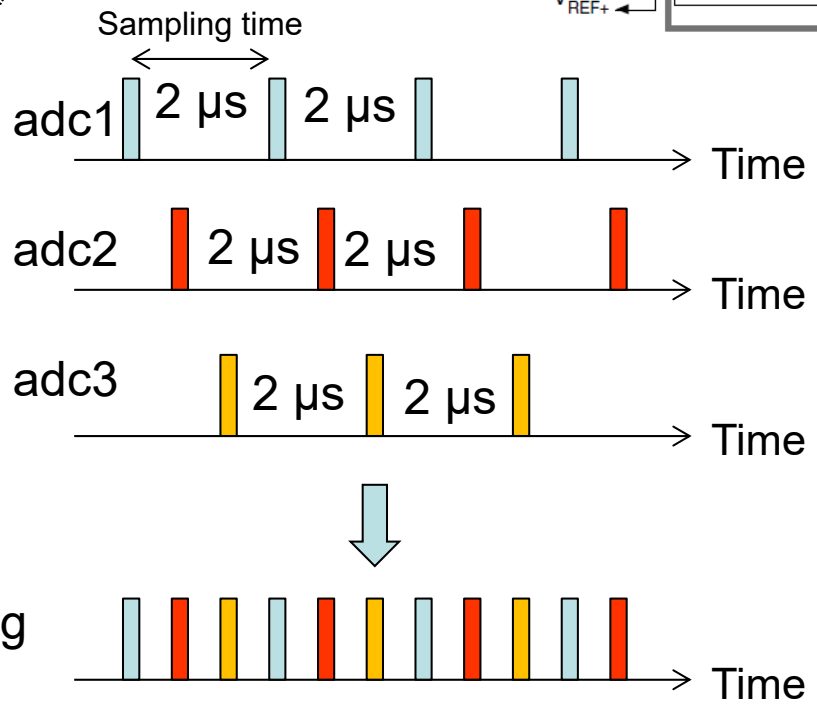
-ADC2_IN0

-ADC3_IN0

-PIN34







ADC12 has 16 inputs
ADC3 has 13 inputs
analog signals

Acquired 3 samples in 2 µs → Higher sampling rate.

ADC in STM32

Some parameters:

- ADC on-off control: ADON Bit

- ADC clock
- Channel mode selection
  - Regular Channel
  - Injected Channel
- Mode of conversion
  - Single conversion mode
  - Continuous conversion mode
- Self Calibration mode
- Data Alignment
  - Right alignment of data
  - Left alignment of data
- Channel-by-channel programmable sample time
- Conversion on external trigger

implementation

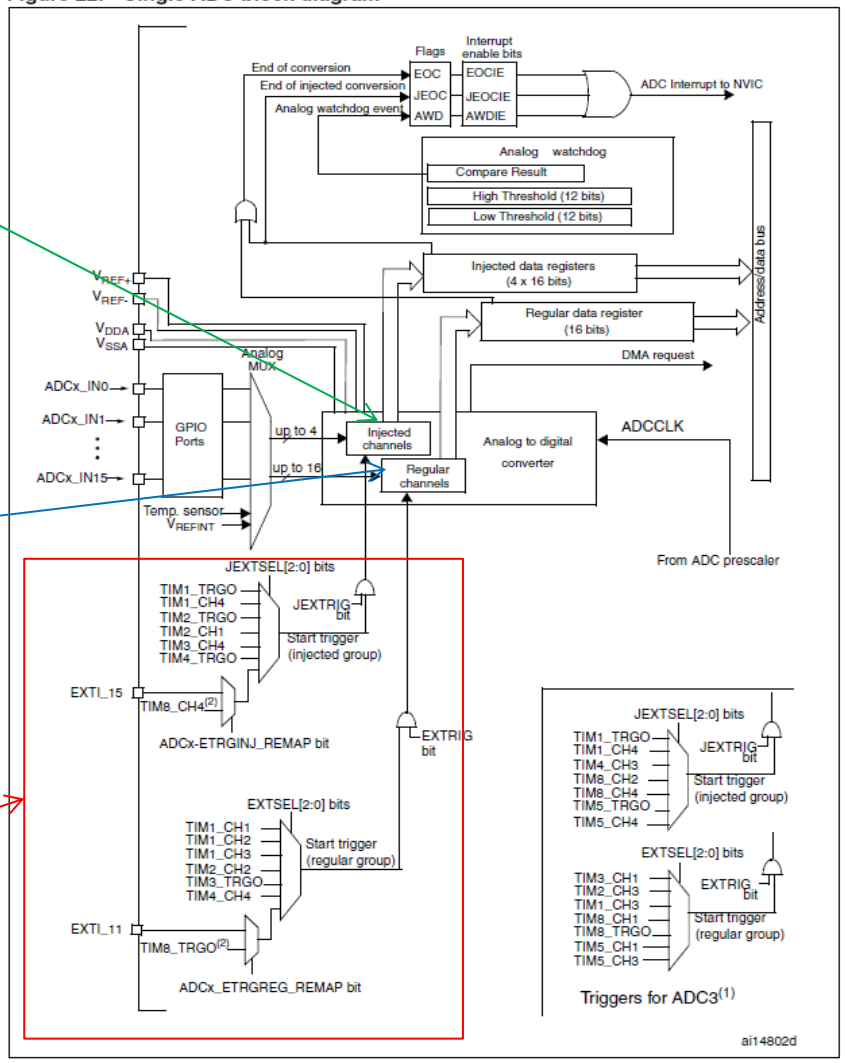
Injected Channel: up to 4 channels in any order.

Regular Channel: up to 16 channels in any order. e.g. Ch3, Ch8, Ch2, Ch2, CH0, Ch2, Ch2, Ch15

External trigger

Initialization

Figure 22. Single ADC block diagram



1. ADC3 has regular and injected conversion triggers different from those of ADC1 and ADC2.
2. TIM8_CH4 and TIM8_TRGO with their corresponding remap bits exist only in High-density and XL-density products.

ADC in STM32

Some parameters:

- ADC on-off control: ADON Bit

- ADC clock
- Channel mode selection
  - Regular Channel
  - Injected Channel
- Mode of conversion
  - Single conversion mode
  - Continuous conversion mode
- Self Calibration mode
- Data Alignment
  - Right alignment of data
  - Left alignment of data
- Channel-by-channel programmable sample time
- Conversion on external trigger

Regular Channels are channels that are being converted regularly.

Injected mode conversion is triggered by an external event or by software. An injected conversion has higher priority in comparison to a "regular" conversion and thus interrupts the regular conversions.

Single conversion mode: ADC stops after one conversion (CONT bit is set to 0).

Continuous conversion mode: ADC starts another conversion as soon as it finishes one (CONT bit is set to 1).

Self calibration: Auto calibration during the process of operation (to minimize the conversion error).

Data alignment: Register 16 bits, ADC 12 bits. – Right or left.

Channel-by-channel programmable sample time: Conversion time.

External trigger: Switch / Interrupt

ADC in STM32

Channel Selection

- 16 multiplexed channels.
- ADC conversions can be organized in two groups: **Regular** and **Injected.**
- A group consists of a sequence of conversions that can be done on any channel and in any order.

Regular Group:

- Composed of up to 16 conversions.
- Regular channels and their order in the conversion sequence must be selected in the ADC_SQRx registers.

Injected Group:

- Composed of up to 4 conversions.
- Injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register.

ADC in STM32

Data alignment

Right alignment of data

Injected group

SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Regular group

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Left alignment of data

Injected group

SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0

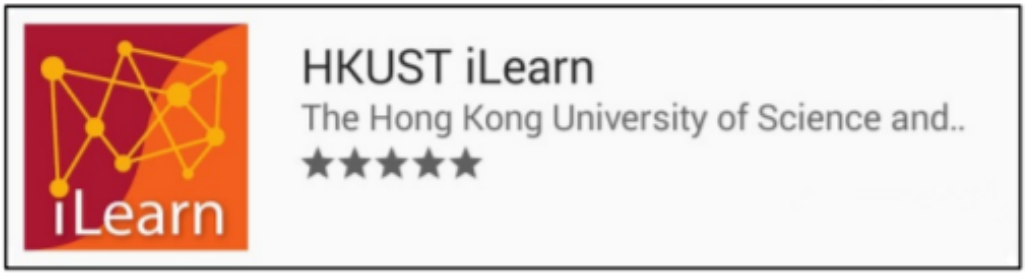
Regular group

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

SEXT: Sign-extended (extended sign bits)

In-class activities (Questions 1 – 3)

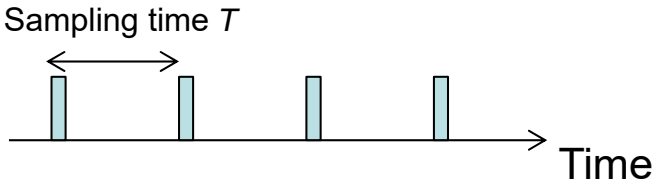
For Android devices, search **HKUST iLearn** at Play Store.



For iOS devices, search **HKUST iLearn** at App Store.

ADC in STM32

- Example: Design appropriate parameters for sampling the music from an audio CD.

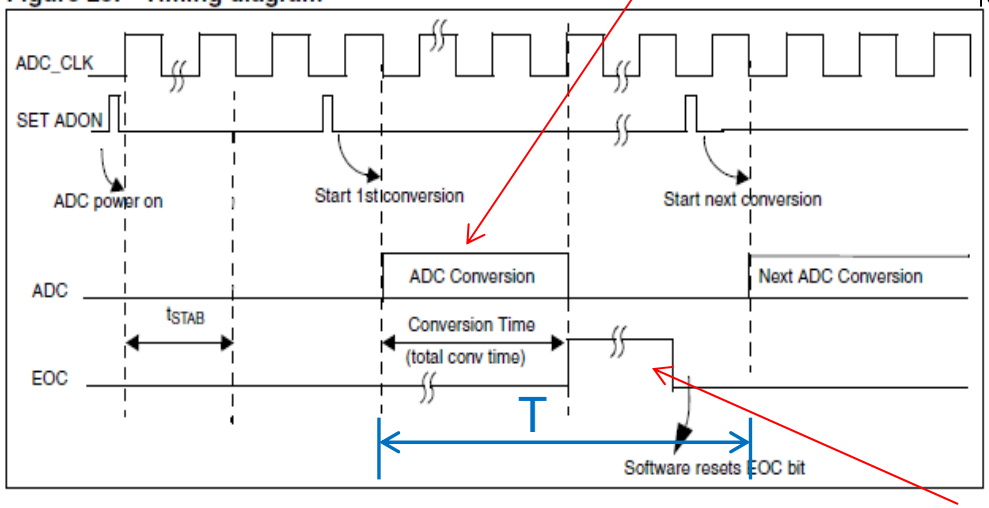


- Solution:
  - Typical sampling frequency: 44.1kHz => sampling time *T* = 1/(44.1k) = 22.675 μs

Total conversion time (Tconv) = conversion time + 12.5 cycles

Declare by ADC sample time register ≠ T

Figure 23. Timing diagram

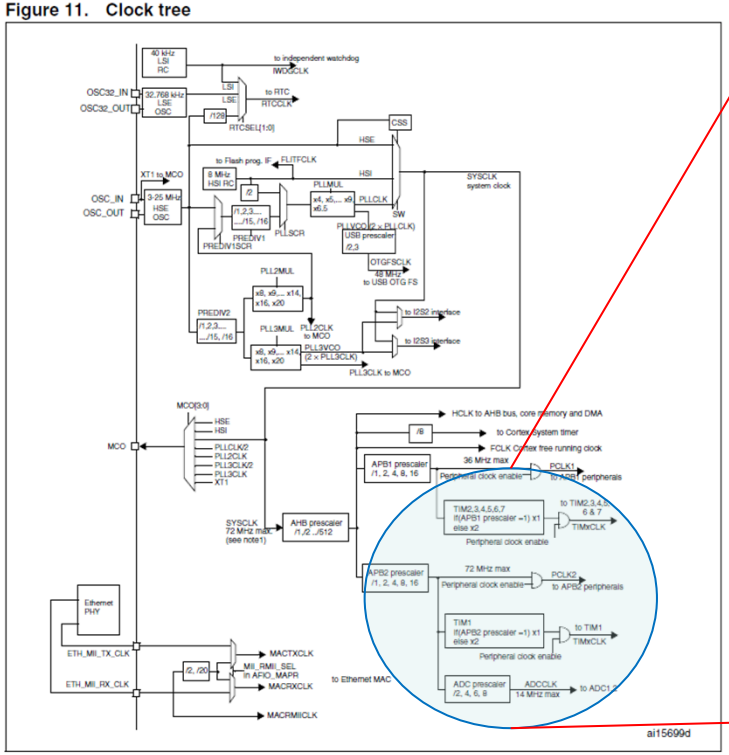


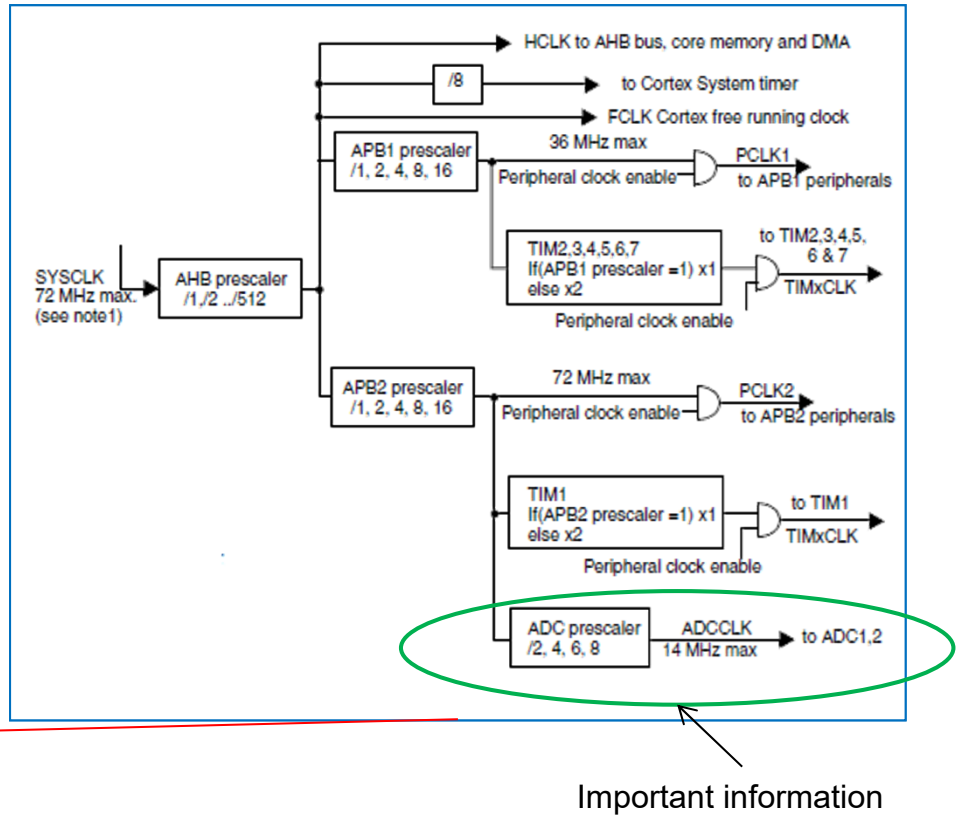
Amount of additional delay

ADC in STM32

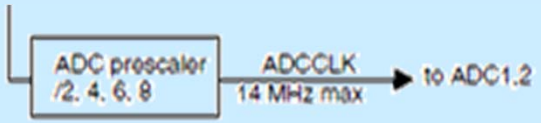
- Important relevant information (Reference Manual):
  - Figure 11 : Clock Tree
  - Section 11.6 Channel-by-channel programming (p.216)
  - ADC sample time register 1 and 2 (p.235 – p.236)

Figure 11. Clock tree





Important information



ADC in STM32

- To design the parameters in providing sampling time T = 22.675 µs

Select from Reference Manual

CLK (MHz) at the input at ADC Prescaler	ADC Prescaler (2/4/6/8)	ADCCLK (MHz) Max 14 MHz	ADC sample time register (cycles) (1.5 – 239.5)	Total conversion time Tcycle (cycles)	Tconv (µsec)	Additional delay (µsec)
28	2	14 = 28/2 (MHz)	1.5	14 = 1.5 + 12.5	1 = 14/14 (µs)	21.675 = 22.675 – 1 (µs)
	6	4.667	71.5	84	18	4.675
56	4	14	1.5			
	8	7	239.5	252	36	
72	6	12	1.5			
	6	12	239.5			

Total conversion time, Tcycle = sample cycles + 12.5 (conversion cycles)

Tconv = Tcycle / ADCCLK

*Reference information: STM32F103xx performance line devices: 1 µs at 56 MHz (1.17 µs at 72 MHz)

ADC in STM32

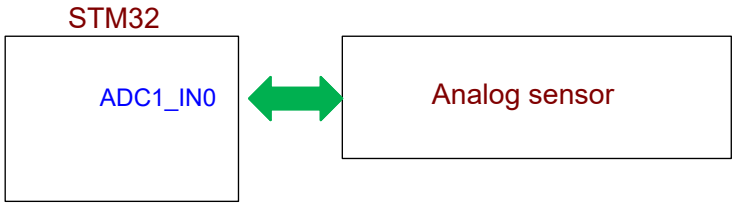
- Other design parameters can be selected as
  - Regular Channel, Single conversion mode
  - 12-bit data with right alignment (stored in 16-bit ADC Data register)
  - How to implement an additional delay in supporting sampling time T?
    - Timer ? Delay loop?
- Additional inquiry
  - How large of the memory do we need for keeping sampled data in 1 second?
    - In 1 second, we have 44,100 samples. If word length is 16 bits (i.e., 2 bytes),
    - Total memory = 44,100 samples x (2 bytes/ sample) = 88,200 bytes (to store 1 sec data)

How much memory would be needed to store 5 minutes of voice (speech)?
44,100 samples x 2 bytes/sample x 5 x 60 (sec) = 26,460,000 bytes.

Reference information: STM32F103xx performance line devices: 1 µs at 56 MHz (1.17 µs at 72 MHz)

Examples of ADC interface

- Task:
  - Take the ADC signal in at a sampling rate 44.1kHz



Initialization

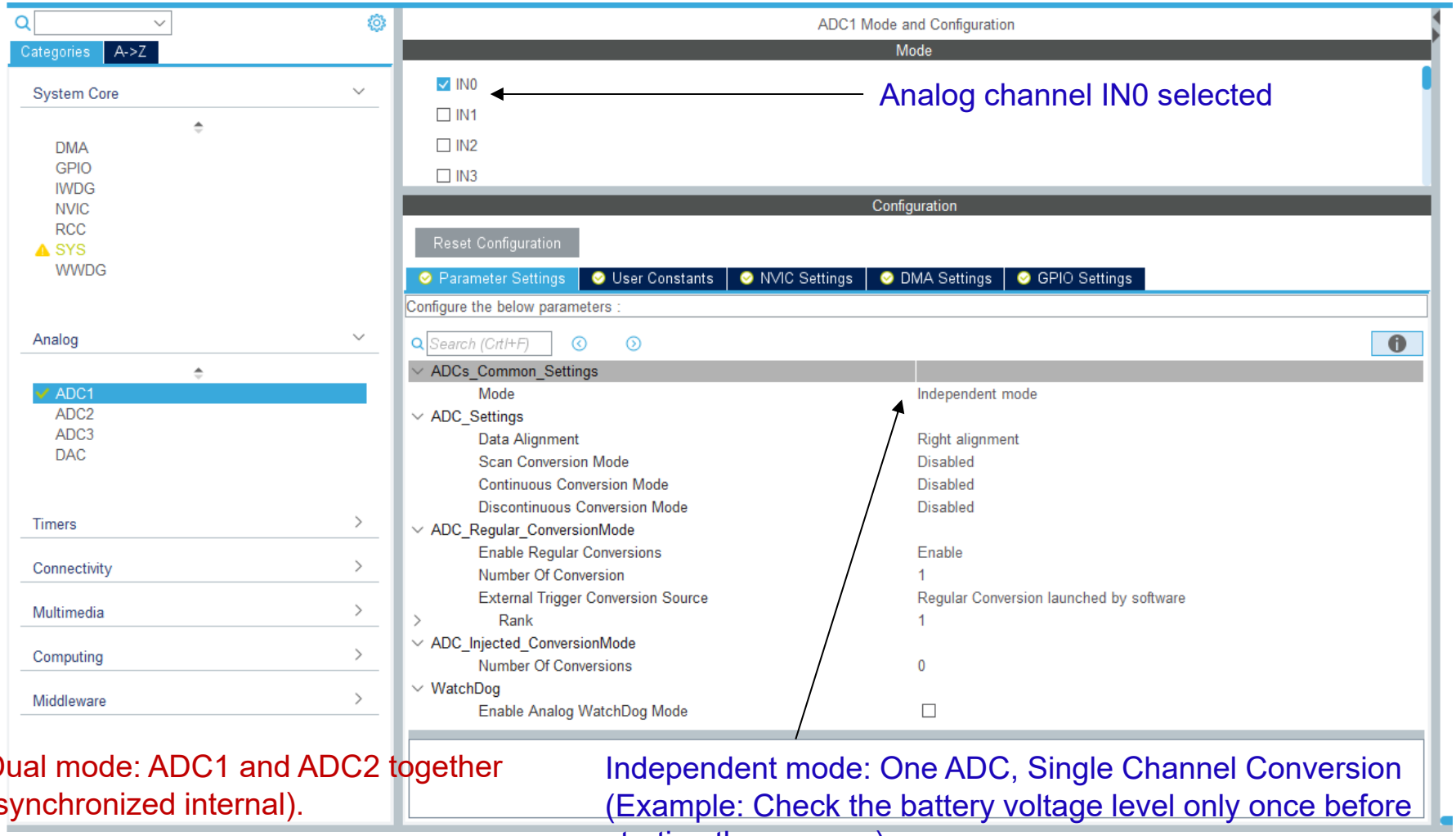
```
Void Main{
    Initialization of ADC                                   Single mode Conversion
     ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
     ADC_InitStructure.ADC_ScanConvMode = DISABLE;
     ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;                Just an example
     ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
     ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
     ADC_InitStructure.ADC_NbrOfChannel = 1;
     ADC_Init(ADC1, &ADC_InitStructure);
```

While loop

implementation

```
     ADC_RegularChannelConfig(ADC1, ADC_Channel_14, 1, ADC_SampleTime_55Cycles5);
     ADC_Cmd(ADC1, ENABLE);          Enable the function of ADC
     while {
          ADC_SoftwareStartConvCmd(ADC1, ENABLE);   Trigger the sampling
          Delay(value);                             Procedure for additional delay **
          D_Data = ADC_GetConversionValue(ADC1) ;    Display data at the specified location
     }
}
```

ADC in STM32



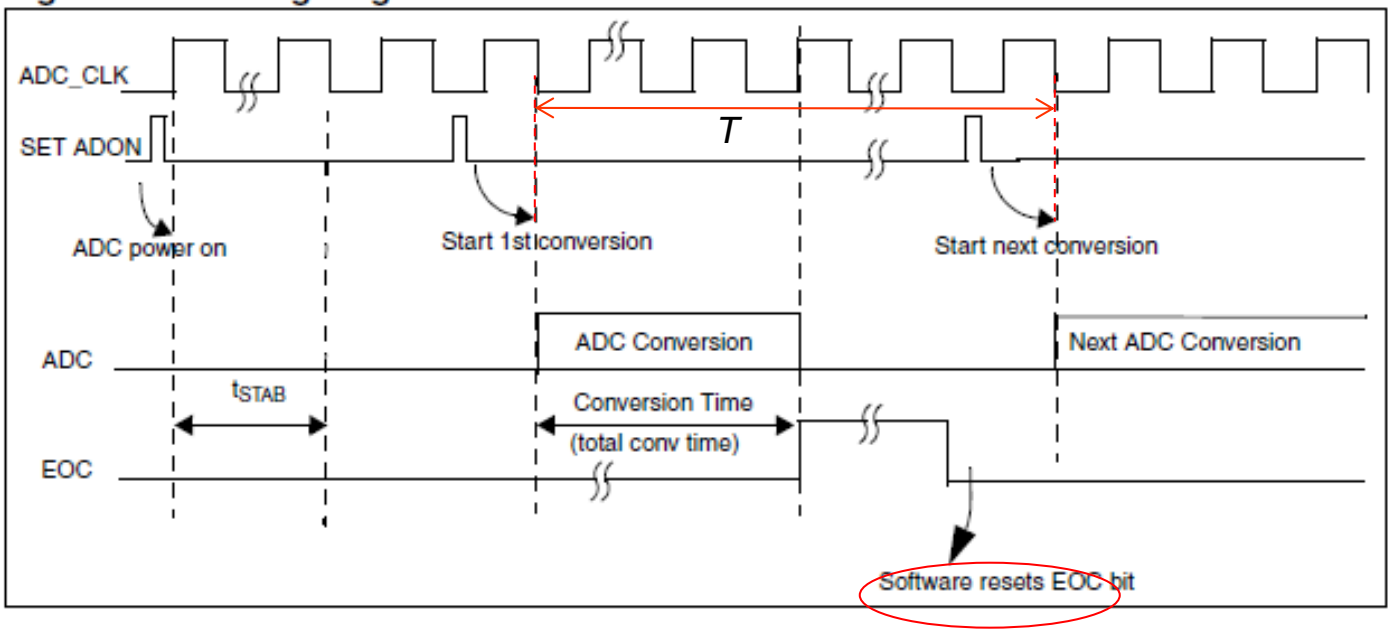
Dual mode: ADC1 and ADC2 together (synchronized internal).

Independent mode: One ADC, Single Channel Conversion (Example: Check the battery voltage level only once before starting the process).

ADC in STM32

- In the continuous mode.
  - After the start of ADC conversion and after 14 clock cycles, the EOC flag is set and the 16-bit ADC Data register contains the result of the conversion.

Figure 23. Timing diagram

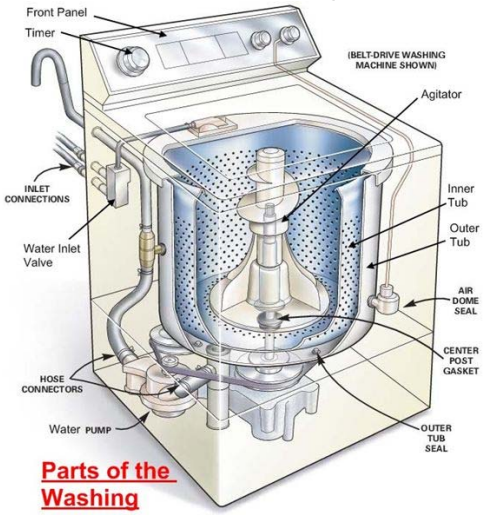


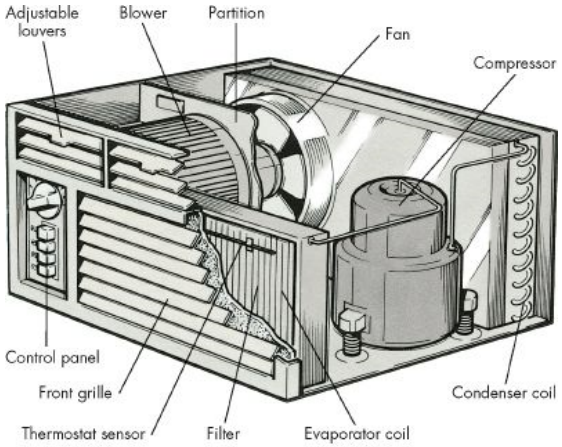
Laboratory Experiment

Description	Choices in this course
Abstract idea of project (Define the functionality of the system)	ADC
Data format / representation	12 bits
Programming Language	C-language
Communication Protocol	-
Physical connection (Pins assignment)	Pins for ADC1 and ADC2
Hardware devices (Microcontroller, Peripherals)	Microcontroller: STM32 ARM Platform LDR

Motors

- In many applications, motor is one of the commonly used components in embedded systems.





The embedded system in the DRONE communicates with the RC controller, which controls the speed of the MOTORS that spins the propellers.

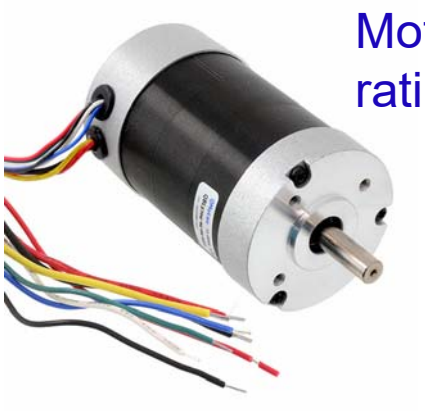
Motors

- They can be classified into several categories:
  - Types of Motors
  - Open-loop Vs Closed loop system

Power = I x V

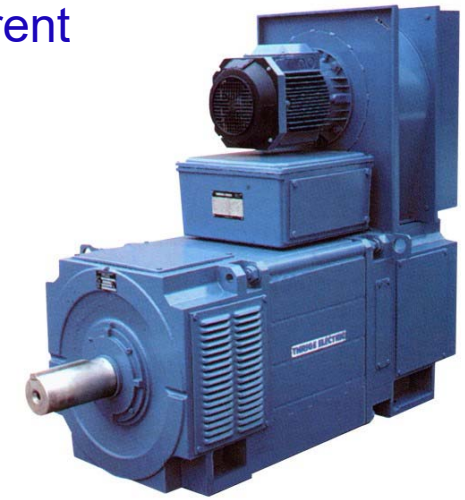
- **Types of motors : DC Motor (brushed and brushless)**

Motors of Different power, voltage and current ratings.



4000 RPM 134W 36 VDC
Shaft is 57mm in diameter
(Brushless motor)

11.5k RPM 6 VDC, 800mA
Shaft is 2mm in diameter
(Brushed motor)



0/45/45kW x 0/700/1000RPM for 117A x 460V DC

Shaft is 70mm diameter x 140mm long

It is an analog motor
The DC motor is available in almost any size that produce large torque

STM32 GPIO current is much lower than 800 mA. So, current amplifiers are needed to drive motors.

Motors

- **Types of motors : AC Motor**

Power = Torque x speed = T. ω (ω is rad/sec)

3000 RPM 375KW 380VAC
(3 phase induction motor)
Size: 41 x 35 x 51 cm

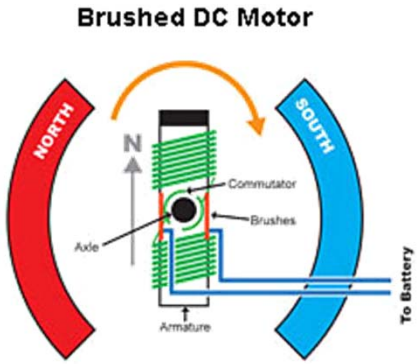
1150 RPM 70W 100-240VAC
(3 phase induction motor)
Shaft size: 8mm, 12mm

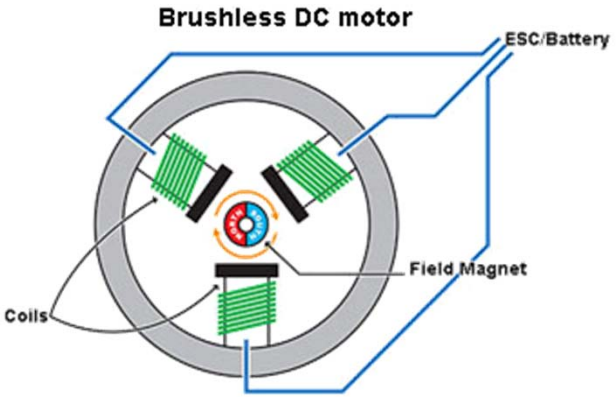
Motors

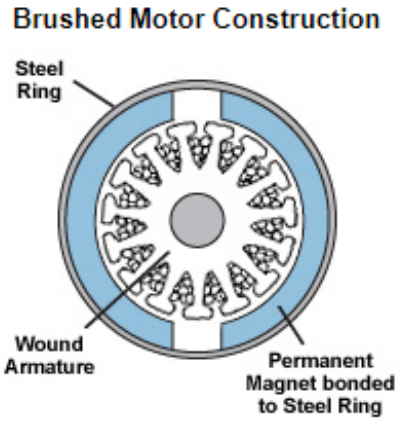
- Supplementary information

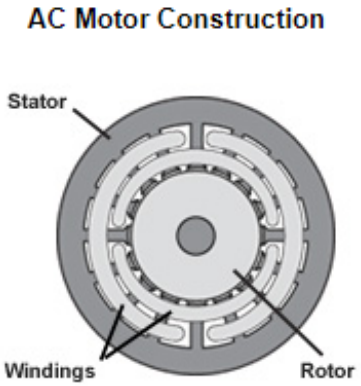
Brushless motor: Permanent magnets are used on the rotor while controllable electromagnets are used to spin the rotor. Eliminates commutator.

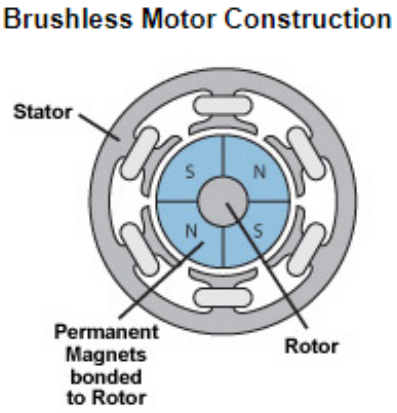
Brushed motor: Permanent magnets are used on the stator while controllable electromagnets is formed by the rotor. Commutator is needed to reverse the current.











Motors

- Types of motors : Stepper Motors

Number of steps = 360°/step-size

1.8 °, 5.6 A, 2.8 V

200 steps

3 V, 400 steps

42 Step Motor 1.7A for CNC

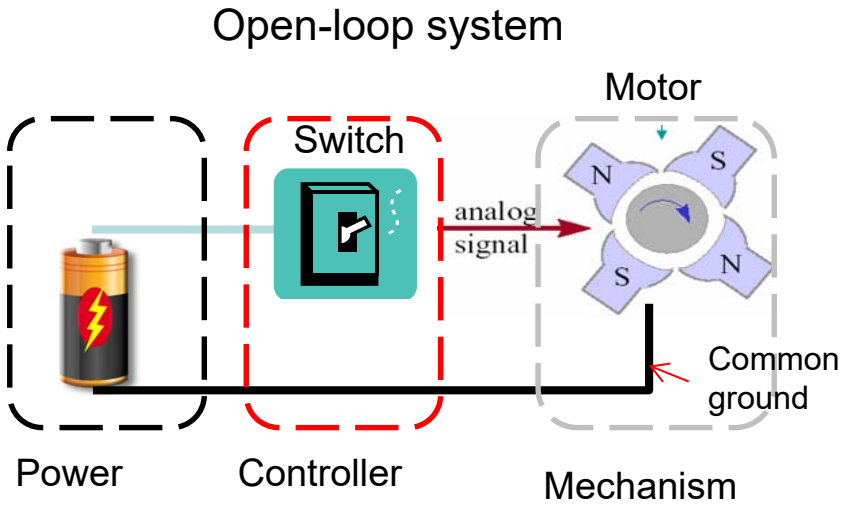
It is a digital motor
The stepper motor is available in small sizes that produce low torque

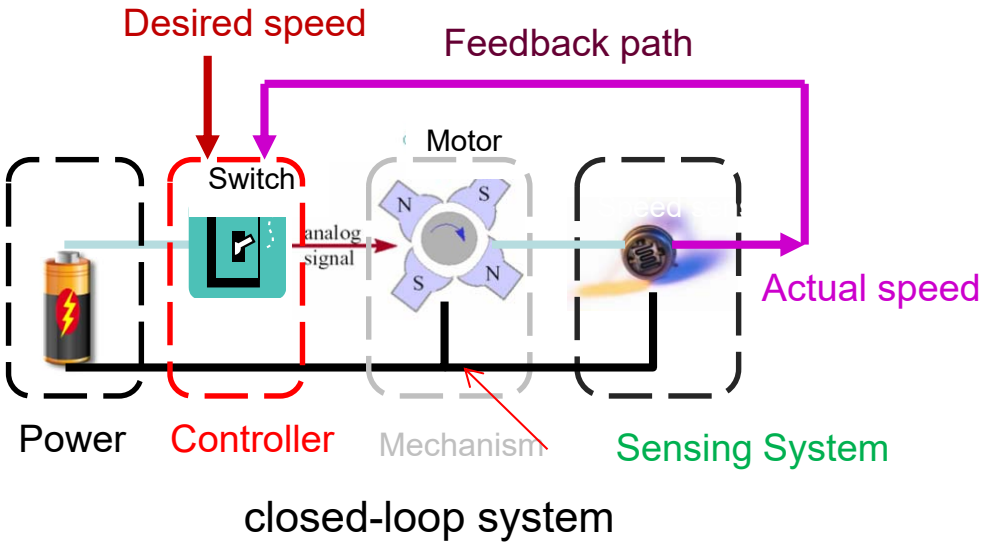
Motors

- Servo motor is a type of motor whose output shaft can be moved to a specific angular position by sending it a coded signal.
- A closed-loop control mechanism is used that incorporates positional feedback in order to control the rotational or linear speed and position.
- Types of servo motors
  - Electrical or partially electronic (most common)
  - hydraulics, pneumatics, or magnetic principles.

Interfacing of Motors with Microprocessor

- There are two control systems:
  - Open-loop Vs Closed loop system



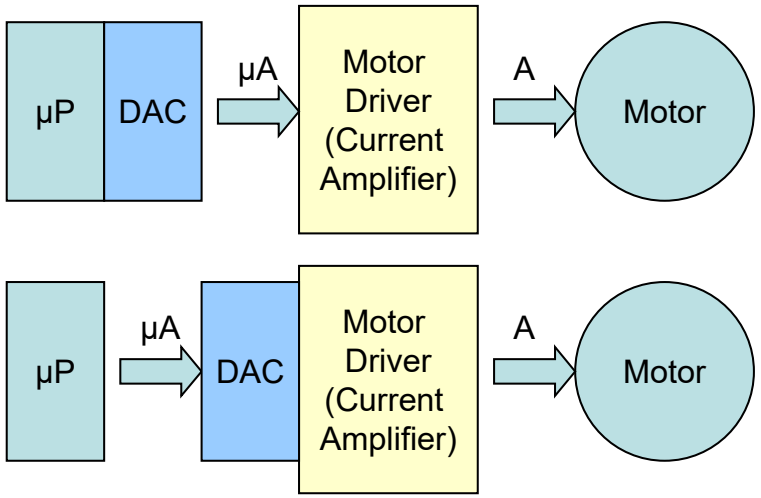


closed-loop system

Interfacing of Motors with Microprocessor

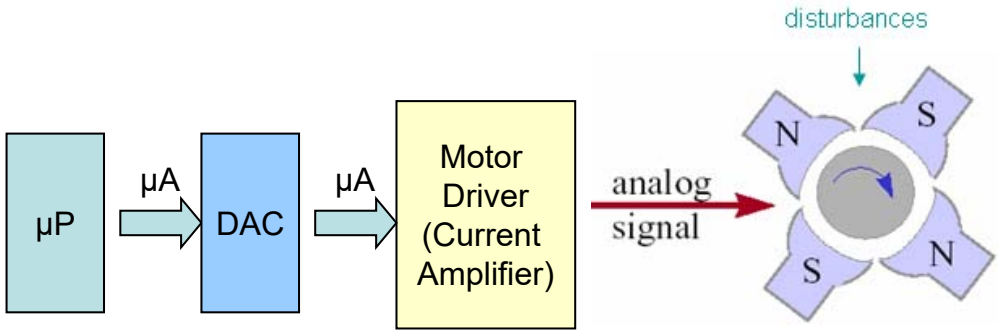
- The Microprocessor cannot generally supply the necessary current to drive a motor directly.

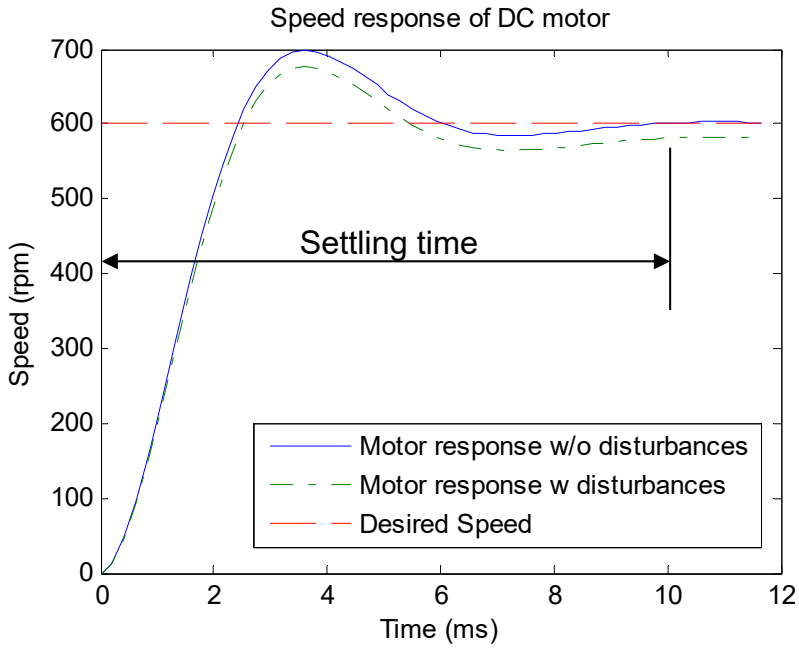
- Depending on the size of the DC motor, a driver must be used to take the output signals form microprocessor and deliver the necessary current to a motor.



Driving circuit of DC Motors: Analog Signal

- The speed and the torque of a DC motor can be controlled over a wide range
- Reversing the direction is done by changing the polarity of the voltage applied to motor
- The DC voltage determines the speed of rotation by a DAC

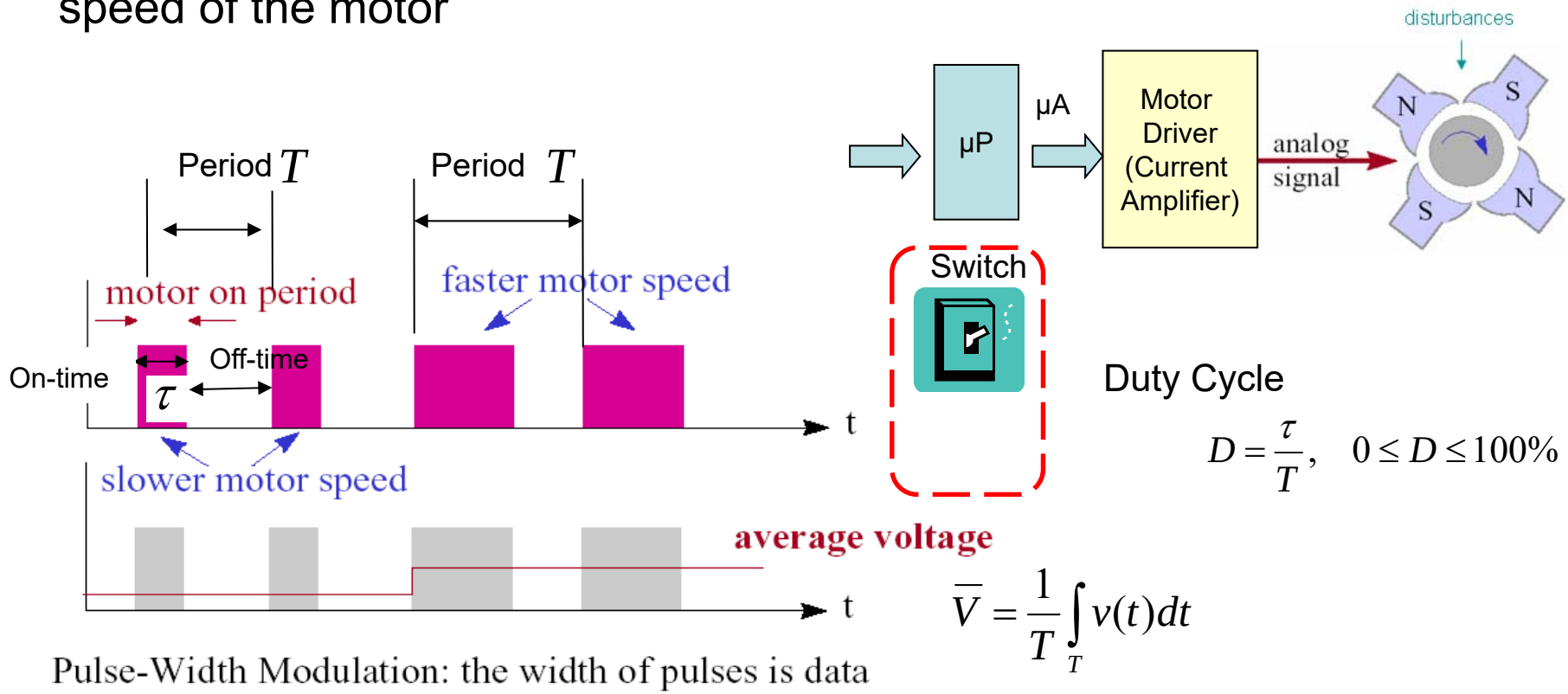




DAC conversion time is a bottleneck.

Driving circuit of DC Motors: Pulse-Width Modulation (PWM)

- An alternative way which does not need a DAC is to **vary the pulse width** of a digital signal input to the motor
- This varies the average voltage delivered to the motor and so does the speed of the motor



Duty Cycle

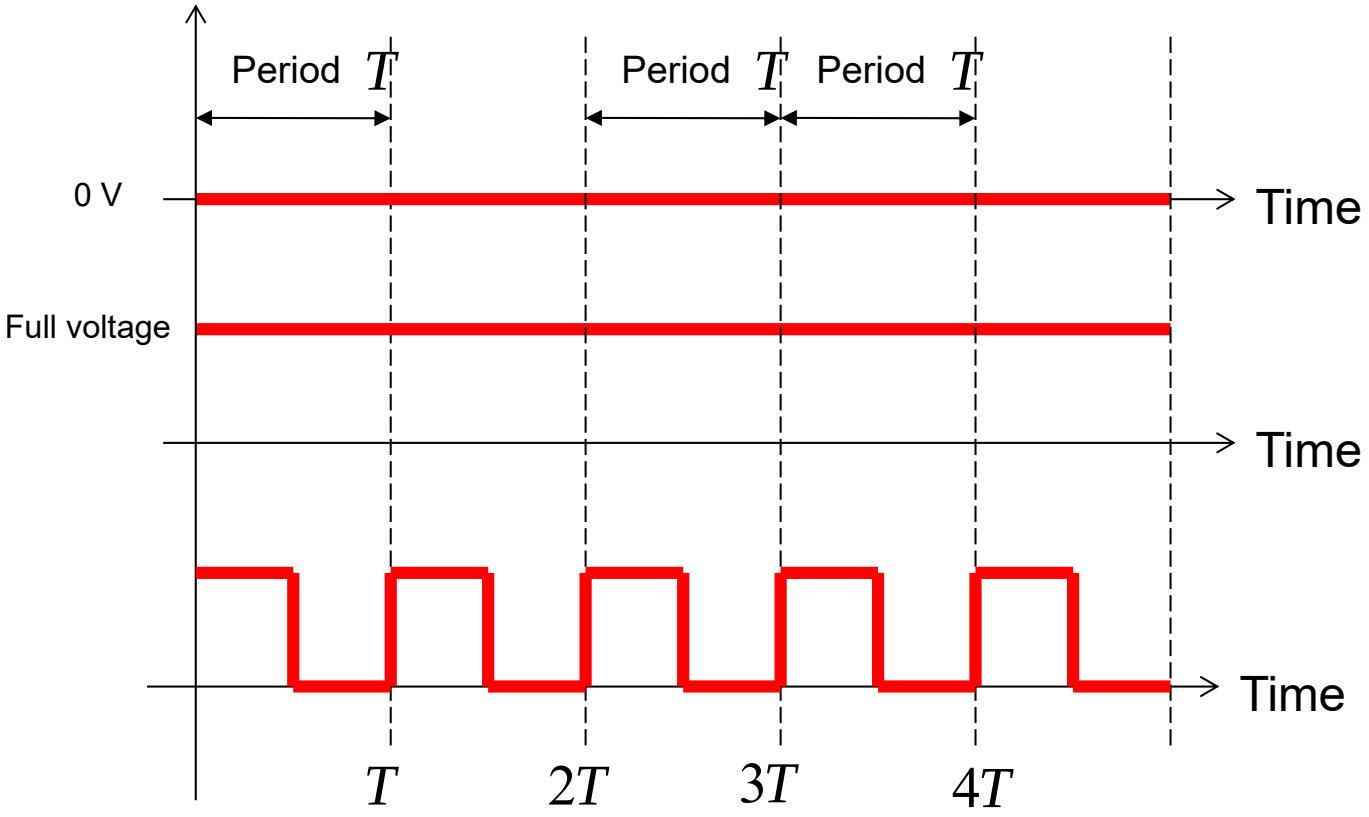
$$D = \frac{\tau}{T}, \quad 0 \le D \le 100\%$$

$$\overline{V} = \frac{1}{T}\int_{T} v(t)dt$$

Pulse-Width Modulation: the width of pulses is data

Driving circuit of DC Motors: Pulse-Width Modulation (PWM)

- What is the speed of the motor if following PWM signal is provided?



Duty cycle?

Speed can be controlled by controlling the Duty Cycle, i.e., ON (or HIGH) time and T.

How do you generate PWM signals of desired duty cycle in STM32?

Timer can be used to control the Duty cycle.

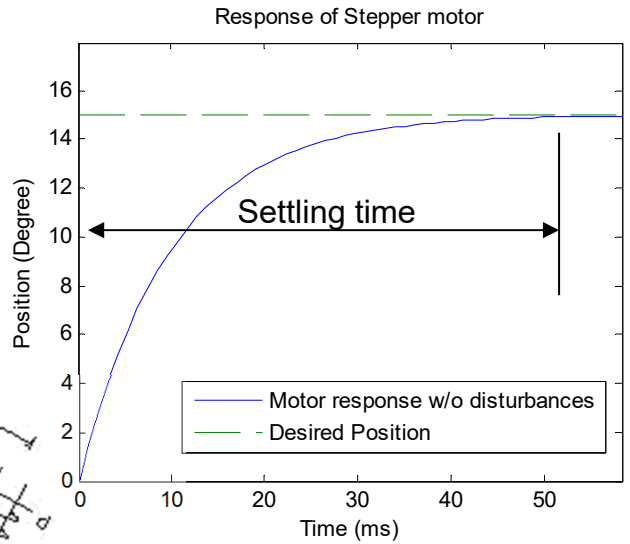
Timer's ARR and CCRx.

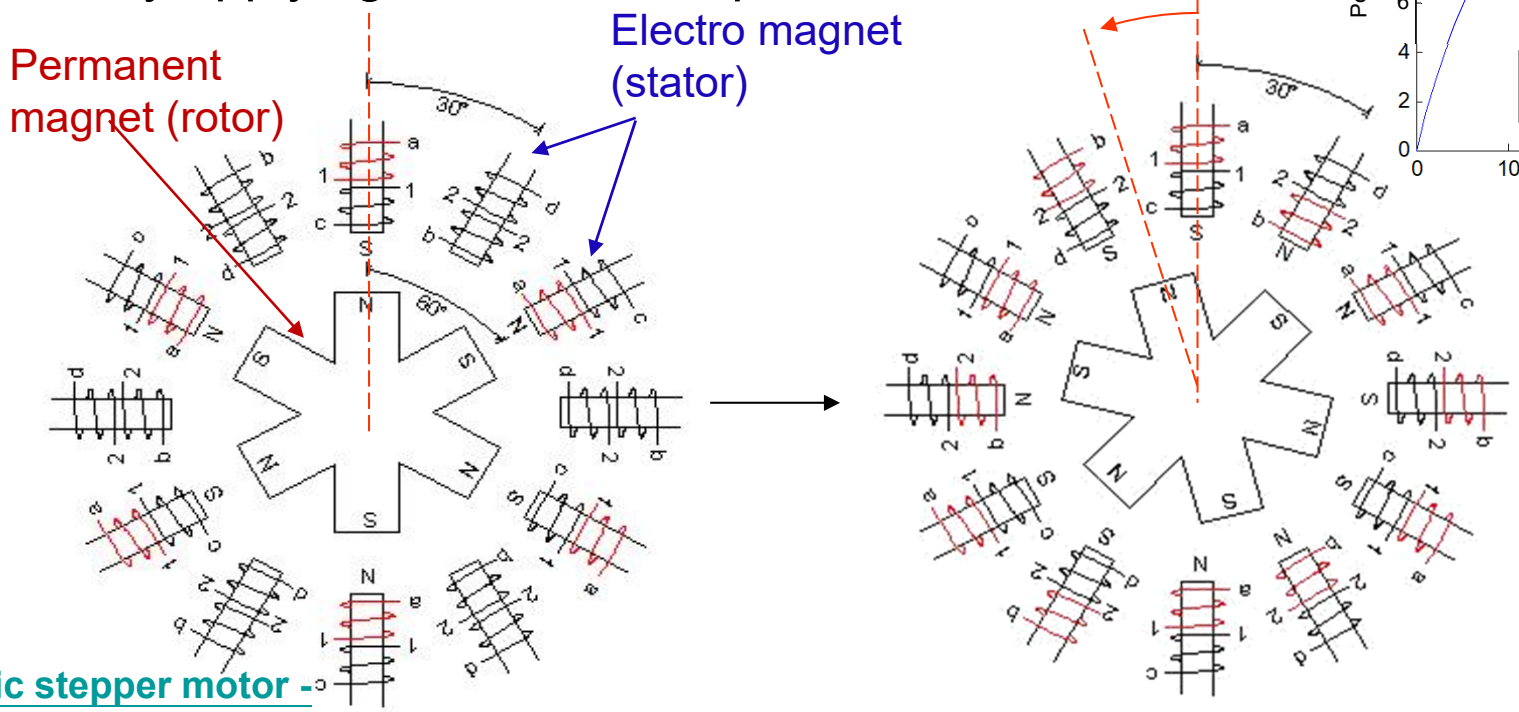
Duty cycle =

CCRx / ARR+1

Stepper Motor

- A stepper motor is a synchronous electric motor that can divide a full rotation in a larger number of steps.
- The motor is rotated a certain number of degree by applying current in the particular coils.





Basic stepper motor - Animation

Half-step movement

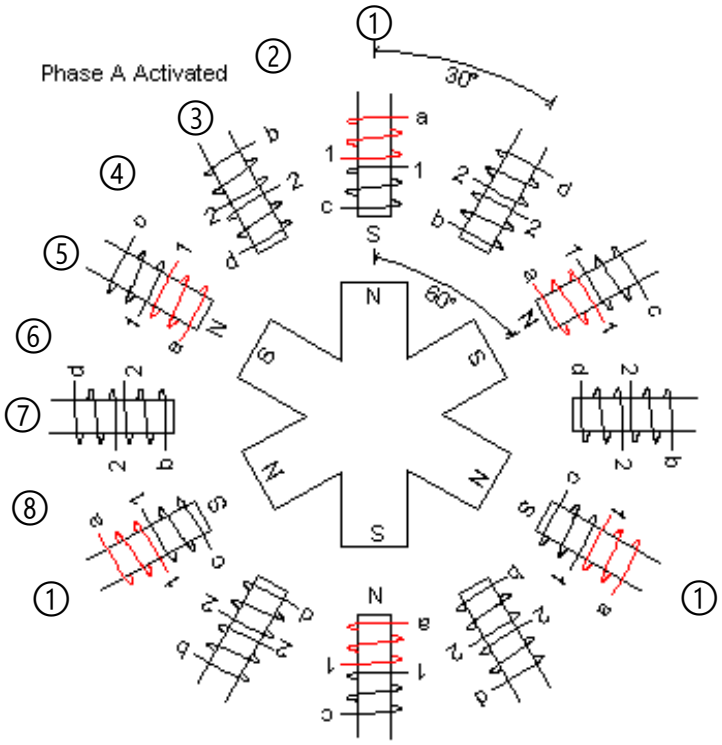
By exciting the appropriate electromagnetic coil, the rotor will turn clockwise or anticlockwise, depending on the direction of current flow (which will cause attraction/repulsion).

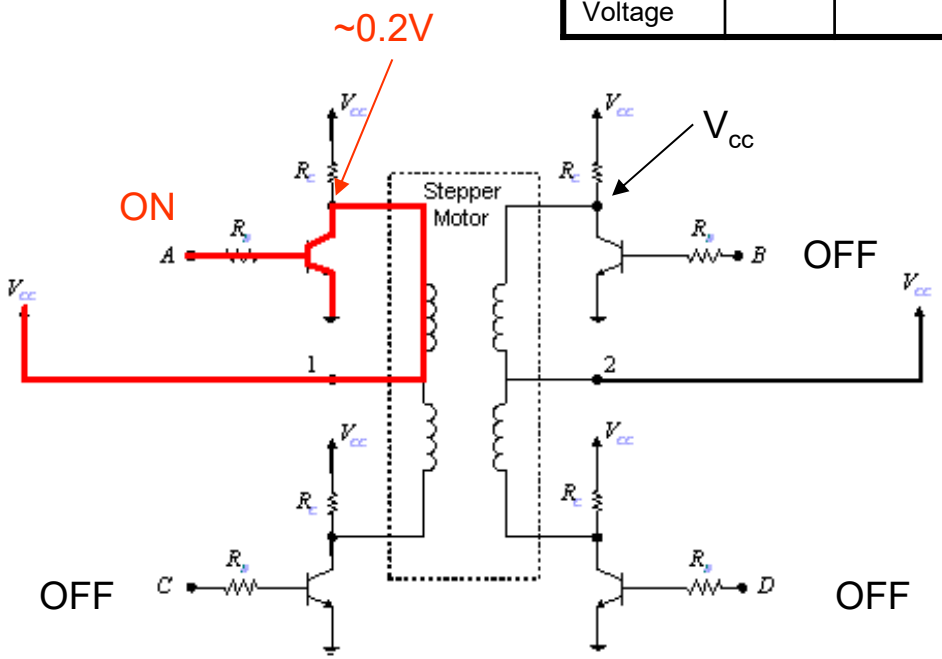
Stepper Motor

- The motor receives a rectangular pulse train and responds by rotating a certain number of degree as dictated by the number of pulses in the pulse train.

Position ①

	MSB A	B	C	LSB D
Status	ON	OFF		
Voltage				

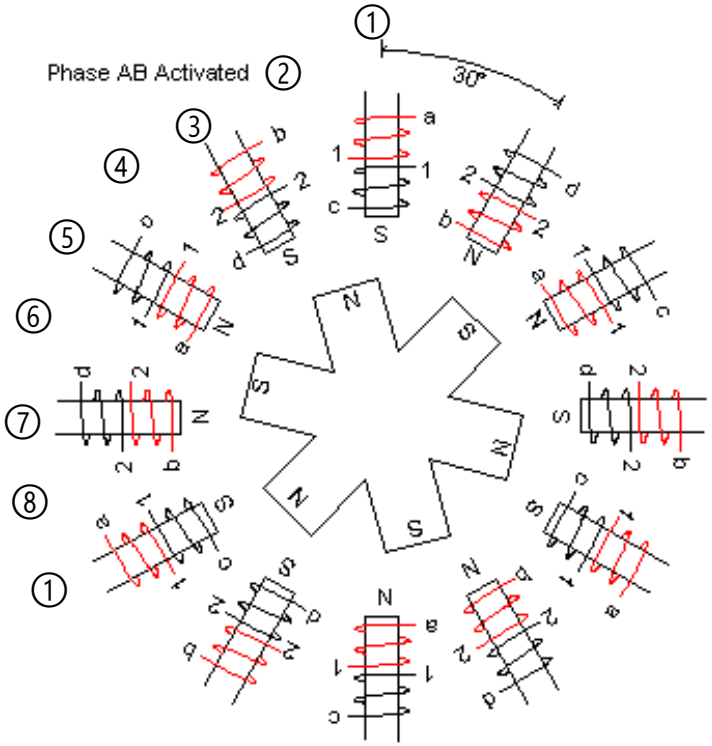


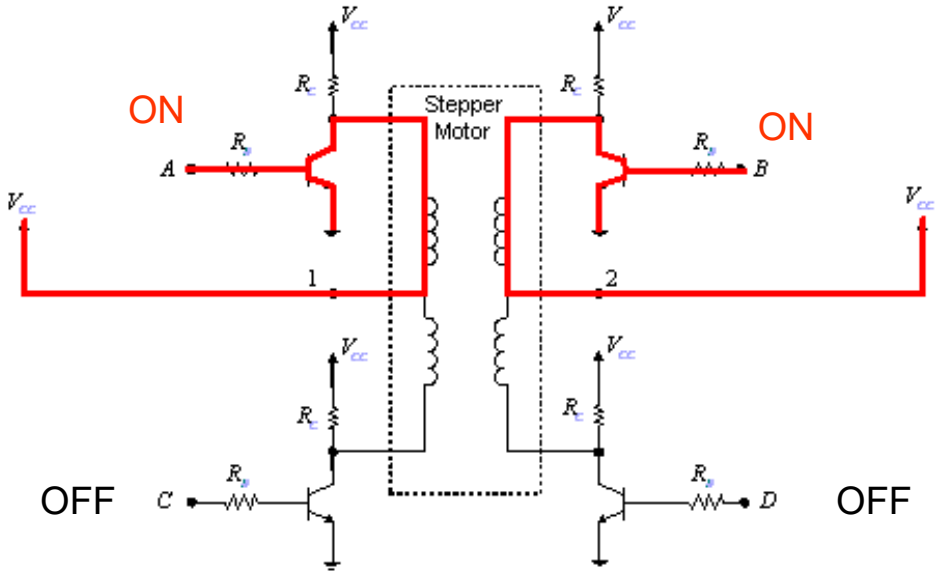


Stepper Motor

- The motor receives a rectangular pulse train and responds by rotating a certain number of degree as dictated by the number of pulses in the pulse train.

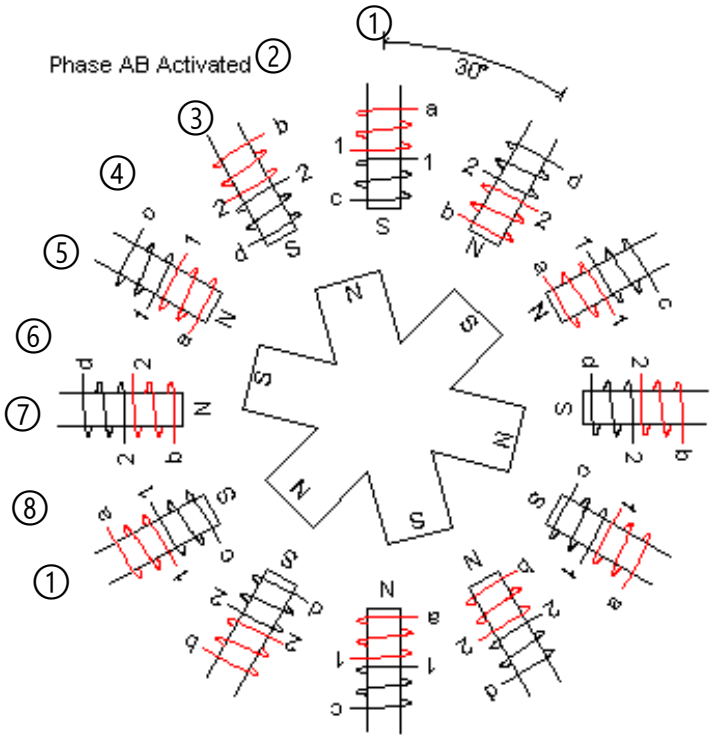
	MSB			LSB
Position ②	A	B	C	D
Status				
Voltage				





Stepper Motor

- The motor receives a rectangular pulse train and responds by rotating a certain number of degree as dictated by the number of pulses in the pulse train.



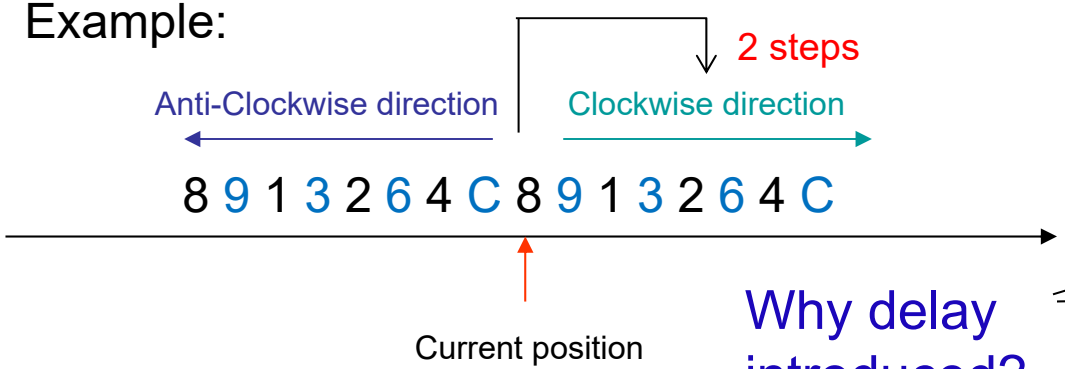
	MSB			LSB	
Position	A	B	C	D	
①	ON	OFF	OFF	OFF	8
②	ON	ON	OFF	OFF	C
③	OFF	ON	OFF	OFF	4
④	OFF	ON	ON	OFF	6
⑤	OFF	OFF	ON	OFF	2
⑥	OFF	OFF	ON	ON	3
⑦	OFF	OFF	OFF	ON	1
⑧	ON	OFF	OFF	ON	9

Anti-clockwise

Clockwise

Stepper Motor

- The motor can rotate clockwise and anti-clockwise, which is controlled by the sequence of pulses.
  - Example:



Why delay introduced?

Initialization

```
void main
{
    // Initialize port B as output port
    while ()
    {
      GPIOB→ODR = 0x08;
      Delay(>settling  time);
      GPIOB→ODR = 0x09;
      Delay(>settling  time);
      …..
    }
}
```

Implementation

Delay is to account for settling time.

- It is an excellent device for position control such as plotter, floppy disk driver, robot, etc.
- The lack of feedback though limits their performance, as the stepper motor can only drive a load that is well within its capacity, otherwise missed steps under load may lead to positioning errors.

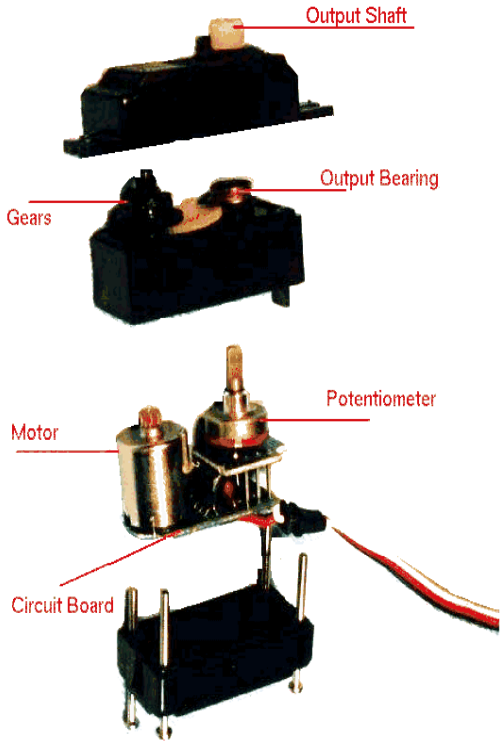
What will happen if no delay?
Motor will oscillate.

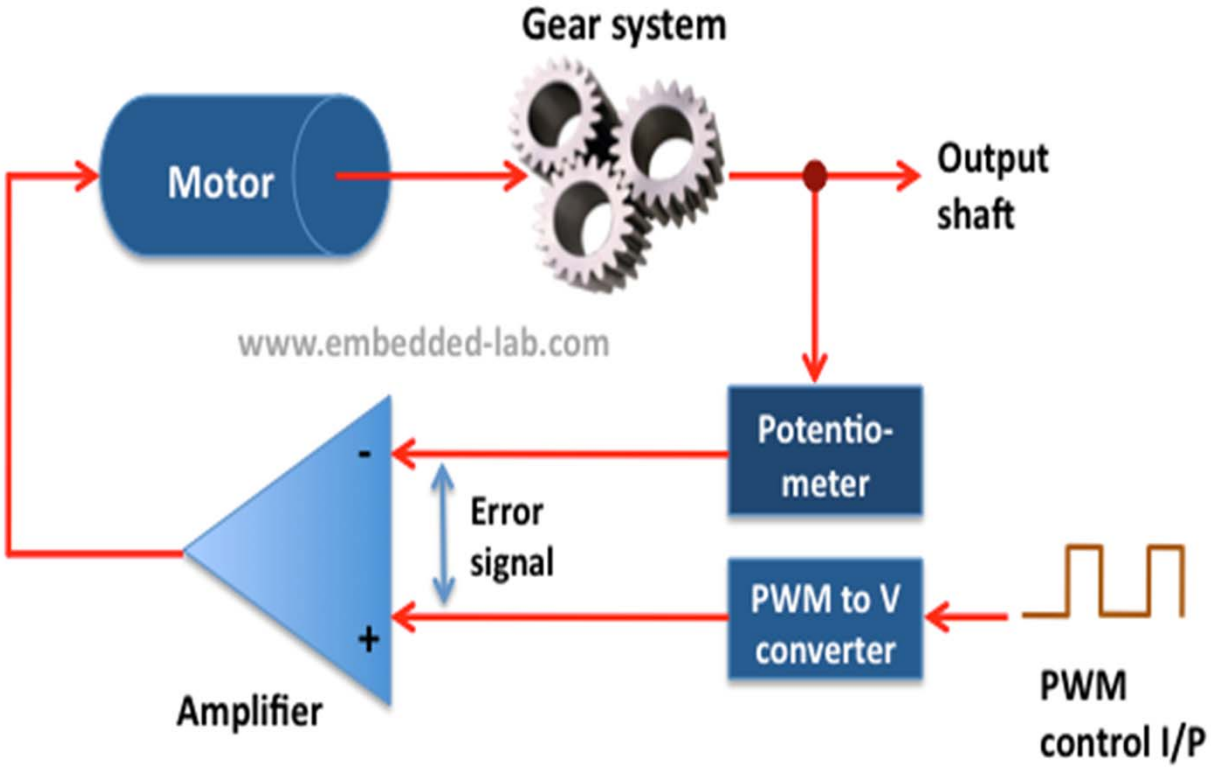
Servo Motor

- Servo motor is a type of motor whose output shaft can be moved to a specific angular position by sending it a coded signal.

- The servo motor will maintain the position of the shaft as long as you keep applying the coded signal.
- When you change the coded signal, the angular position of the shaft will change.
- Types of servo motors
  - Electrical or partially electronic (most common)
  - hydraulics, pneumatics, or magnetic principles.

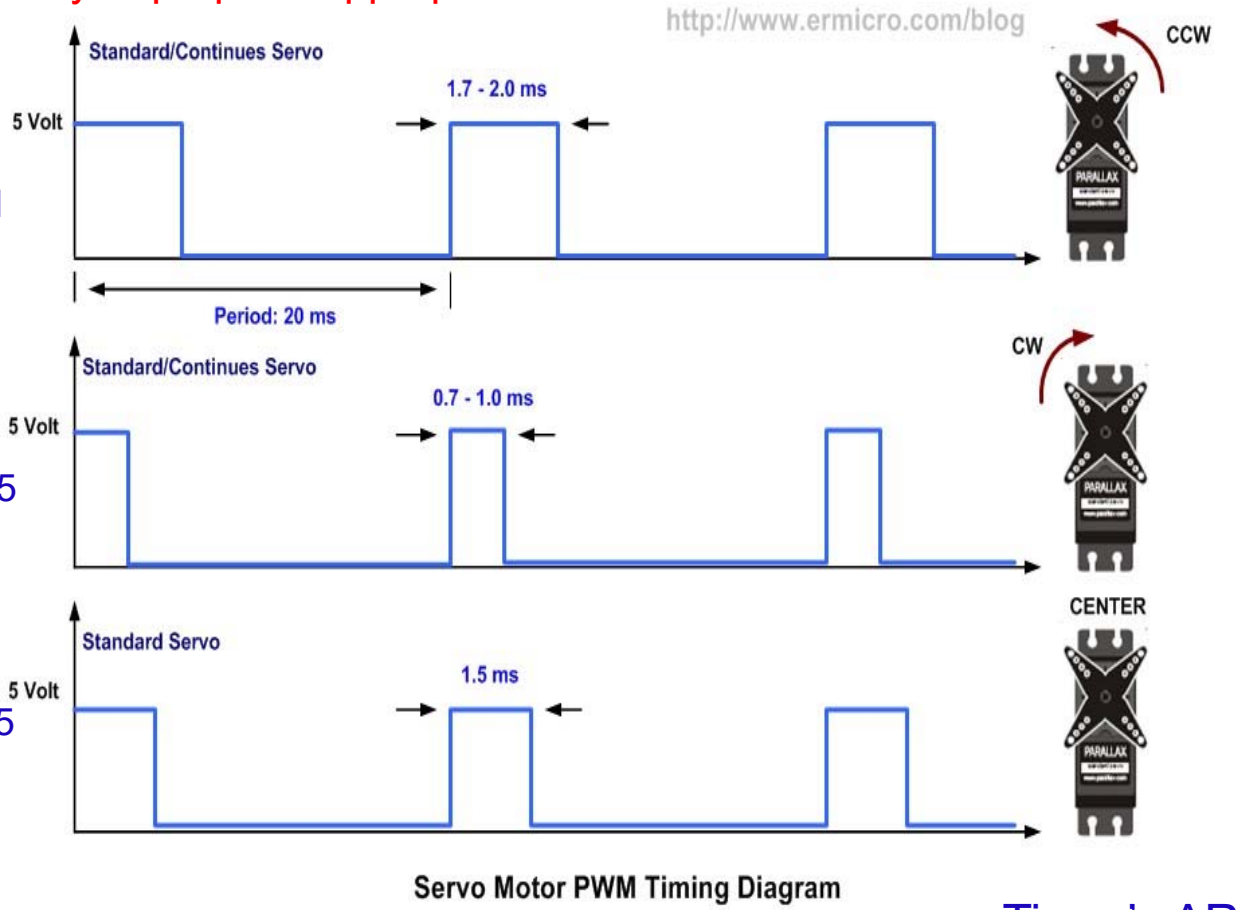
Servo Mechanism





Servo Motor PWM Timing

Can you propose appropriate ARR and CRRx values?



Servo Motor PWM Timing Diagram

Duty cycle =
2ms / 20ms=0.1

Duty cycle =
1ms / 20ms=0.05

Duty cycle =
1.5ms / 20ms=0.075

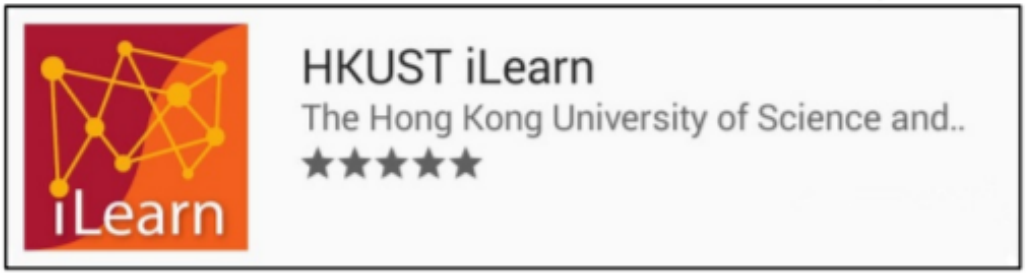
ARR = 39

Timer's ARR and CCRx.
Duty cycle = CCRx / ARR+1

How do you generate PWM signals in STM32 ?

In-class activities (Question 4 – 6)

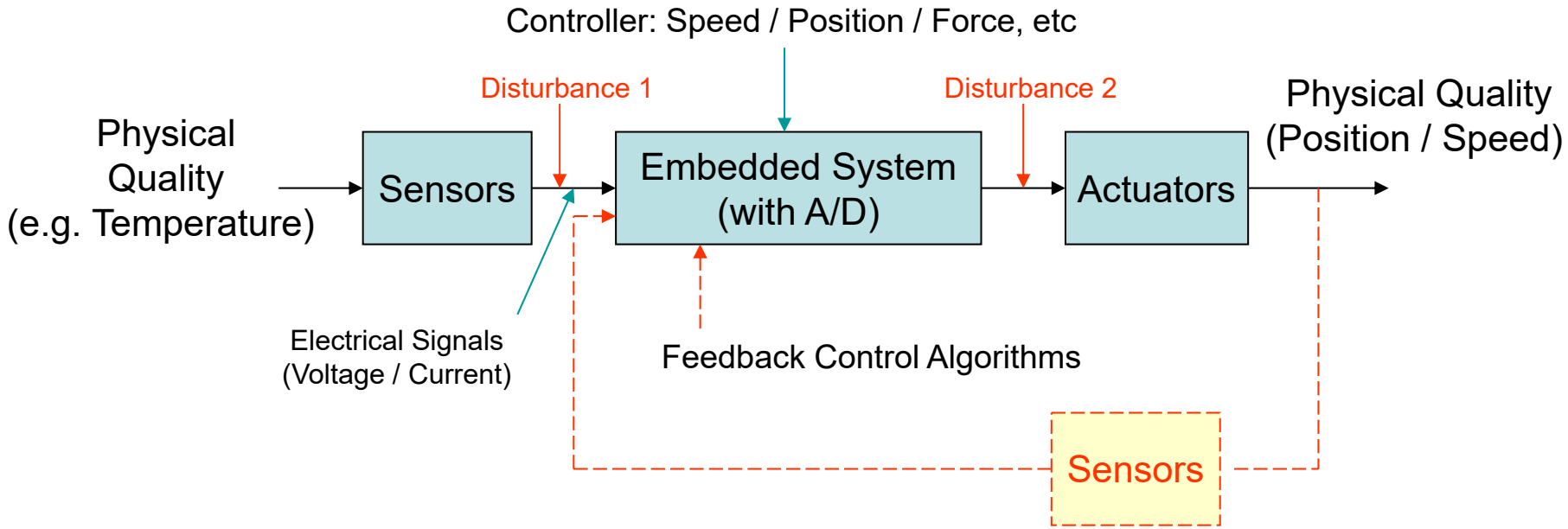
For Android devices, search **HKUST iLearn** at Play Store.



For iOS devices, search **HKUST iLearn** at App Store.

Advanced design for motor controlling system

- How do we minimize the error in the position / speed control due to disturbances?

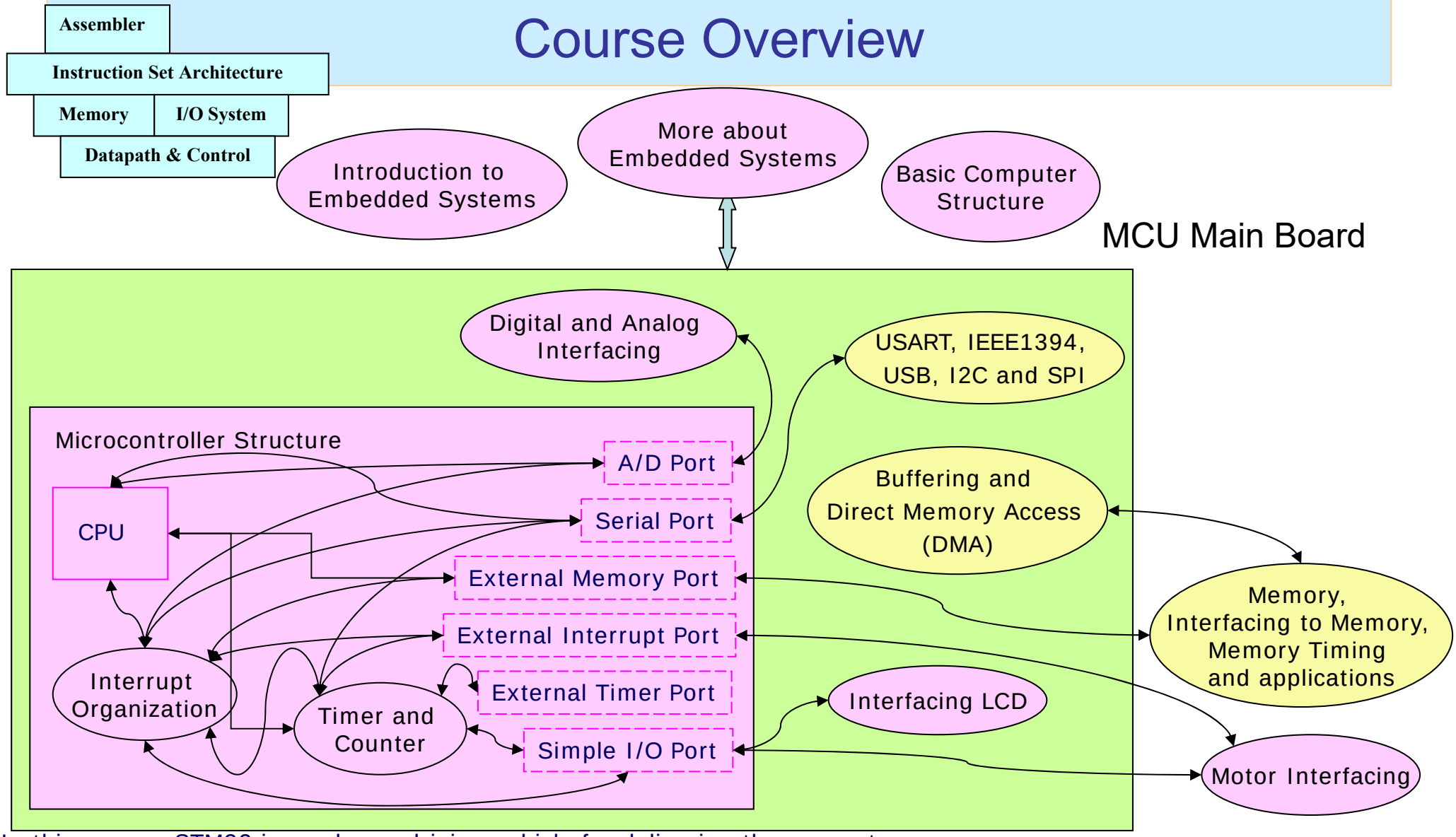


Some typical sensors for motor:
Encoder, optical tuning speed, variable resistor

Reflection (Self-evaluation)

- Do you
  - Understand basic concepts of Analog-to-Digital Conversion (ADC) and Digital-to-Analog Conversion ?
  - Understand characteristic and driving circuits of DC and Stepper motors ?
  - State the reason in using current amplifier in motor driving circuit?
  - Know how to write a PWM driver ?
  - Know how to generate a sequence in driving a stepper motor?

Course Overview



In this course, STM32 is used as a driving vehicle for delivering the concepts.

To be covered	In progress	Done

Main Types of ADCs (Supplementary Slides)

– Successive Approximation (SAR) ADC, Delta-sigma (ΔΣ) ADC,

Dual Slope ADC, Pipelined ADC and Flash ADC

Successive Approximation (SAR) ADC

Has a sample-and-hold circuit that takes in the analog voltage.

An on-board DAC creates an analog reference voltage equal to the digital code output of the sample and holds a circuit.

Both of these are fed into a comparator which sends the result of the comparison to the SAR.

This process continues for “n” successive times, with “n” being the bit resolution of the ADC itself, until the closest value to the actual signal is found.

Pros: High sampling rate, simple circuit

Cons: Low bit resolution and dynamic range

Main Types of ADCs

Delta-sigma (ΔΣ) ADC

Works by over-sampling the signals far higher than the selected sample rate (hundreds of times than the selected sample rate).

The DSP then creates a high-resolution data stream from this over-sampled data at the rate that the user has selected.

This approach creates a very high-resolution data stream (24-bits is common).

Pros: High Resolution output

Cons: Speed limitation

Main Types of ADCs

Dual Slope ADC

Uses an integrator.

The voltage is input and allowed to “run up” for a period of time.

Then a known voltage of the opposite polarity is applied and allowed to run back down to zero. When it reaches zero, the system calculates what the input voltage had been by comparing the run-up time with the run-down time, and by knowing what the reference had been.

The run-up and run-down times are the two slopes for which this technique has been named.

Pros: Precise and Accurate Conversion

Cons: Slow (due to ramp-up and ramp-down time)

Main Types of ADCs

Pipelined ADC

Cascade of several low resolution stages to obtain high overall resolution (e.g. 10bit ADC can be built with series of 10 ADCs each 1-bit only!).

Each stage performs coarse A/D conversion and computes its quantization error, or “residue”.

Pros: Faster than SAR and Delta-sigma ADCs

Cons: Latency due to serial pipelined conversion process.

Flash ADC

Converts analog to a digital signal by comparing it with known reference values. The more known references that are used in the conversion process, the more accuracy can be achieved.

Pros: Fastest ADC

Cons: Low resolution (8 bits)

More information: https://dewesoft.com/daq/types-of-adc-converters#main-adc-types