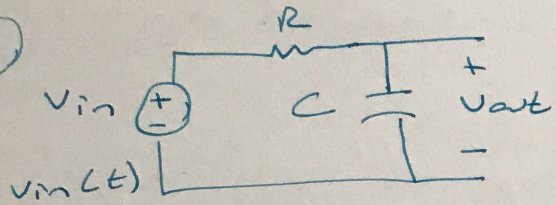
CSE 2037, Week 13

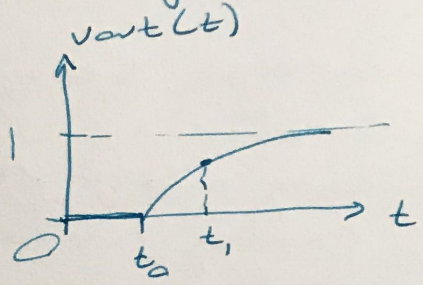
Conversion Between Analog and Digital Values

- Analog and Digital Values
  - Analog Values in Physical Systems
  - Digital Values in Embedded Systems
  - Digital Values in Code
- Analog to Digital Conversion (ADC)
  - Sapling
  - Quantization
  - ADC under Mbed
- Digital to Analog Conversion (DAC)
  - Zero-order Hold
  - Pulse Width Modulation
  - DAC under Mbed
- Mbed Exaple Codes

Analog Values in Physical Systems

(I)





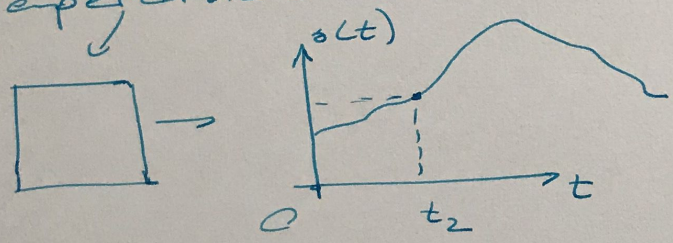
Vout(t) { ① defined for all time values.
② amplitude at any specific time is "read" valued.

① → We have infinite number of values

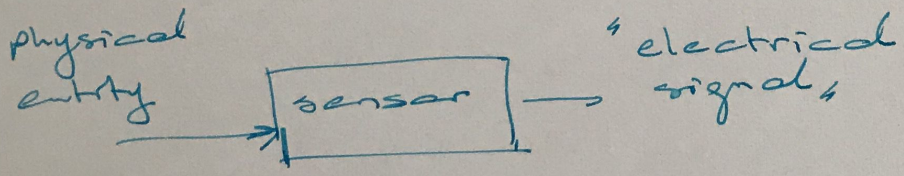
② → The amplitude at any time can take infinite number of values.

In our microcontroller, we do not have infinite storage space.
It is also true for any embedded system.

(II) Temperature sensor



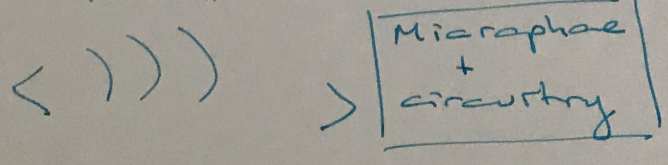
① also hold here.
②



(III) Microphone

Audio signals

"I am speaking now"



① also
② hold here.

How can I represent of process analog values in my microcontroller?

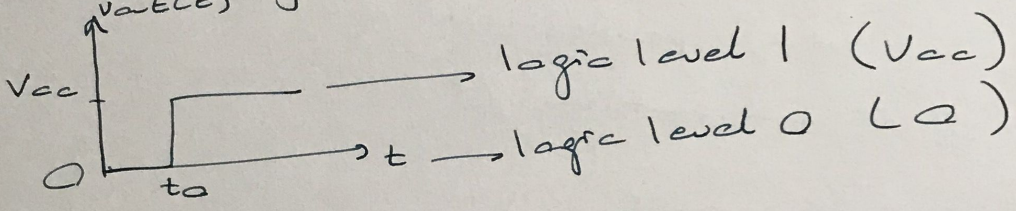
→ Microcontroller has limited memory.

→ I need "another" representation for this purpose.

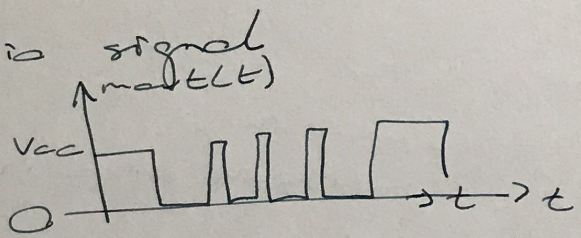
⇒ Digital value.

→ The simplest digital form is a "bit" which can take two values as logic level 0 or 1.

RC - circuitry



Audio signal



→ We use "group" of bits, formatted to represent a value in embedded system.

→ In C code, "formatted values"

char name; → 8-bits, (-128, 127) integer.

int name; → 32-bits, (2^{-31}, $2^{31}-1$) integer.

float name; → 32-bits; ⇒ floating point representation integer, fractional parts.

⇒ Digital form

Analog Value (Physical) —ADC→ Digital Value (Embedded system, Microcontroller)

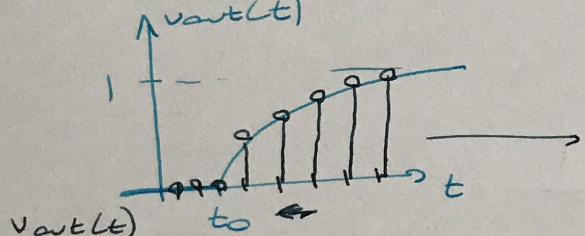
Digital Value (Embedded system, Microcontroller) —DAC→ Analog Value (Physical)

Analog to Digital Conversion

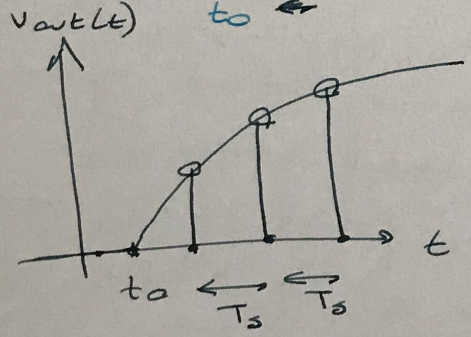
(1) We have to limit the number of samples in our analog signal.
⇒ Sampling (*)

(2) We also have to limit the analog value by using "predefined" number of bits.
⇒ Quantization (*)

(1) RC circuit

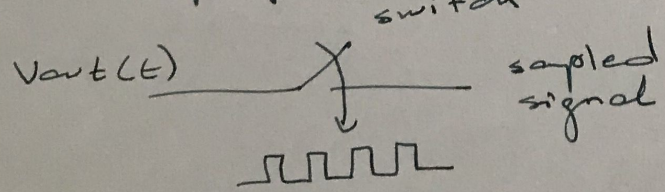


sampling certain time instants.



I sample the analog signal periodically.
T_s → sampling period
$f_s = \frac{1}{T_s}$ → sampling frequency

Sampling operation "simplified"

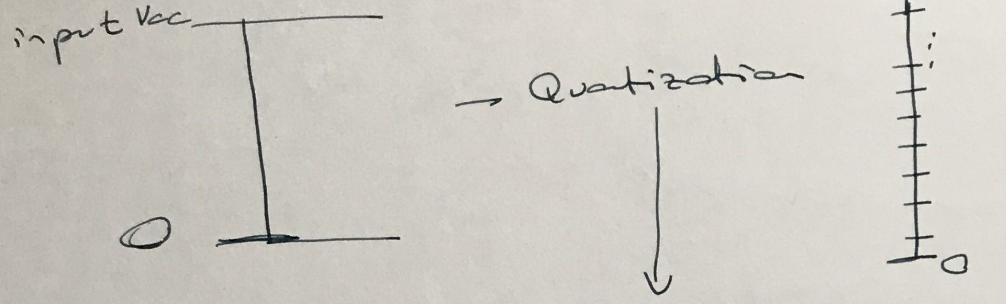


Index values to our sampled signal.
1. sample
2. sample
3. sample

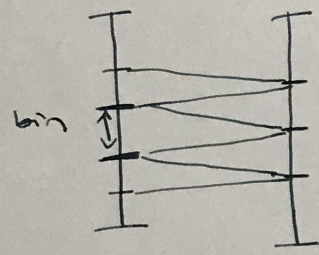
x[n]
↳ index ∈ I
"Array" in C language.

I have "sapled" the values, however they still have infinite precision.

Quantization



N-bit quantization.



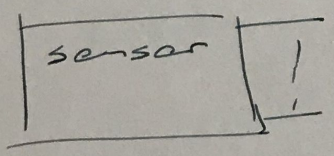
a range in input analog value is represented by a single number.

"12-bit" ADC module ⇒ 0, Vcc volts input range is represented by digital values $0 \to 2^{12}-1$ ⇒ In C language unsigned int.

sensor 24-bits

⇒ It has its own "ADC" module.

or it uses, digital com. module for operation UART, SPI, I^2C ——



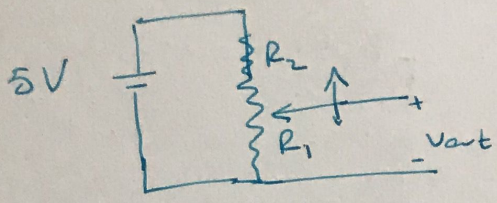
analog → ADC → digital

1. sapling
2. quantization

→ ADC under Mbed

⇒ My microcontroller "mbed" have the ADC module within it.

Potentiometer (Analog voltage source)



$$V_{out} = \frac{5 R_1}{R_1 + R_2}$$

$V_{out} \in 0 - 5V$

Mbed "adc-exaplc1.cpp"

AnalogIn ain(p15);

ain.read()

5 → 0

→ "float 32-bit output," 0-1

ain.read_u16() 0x0400

5 → unsigned integer "16-bits"

0 → 0x0000

ADC → 16-bits

unsigned int 32-bits

16-bits 16-bits

→ under mbed

16-bits

0

unsigned char 8-bits

24-bits

16-bits 8-bits

X

⇒ Quantization.

wait_ms(200)

→ sampling at every 200 msec.

Digital to Analog Conversion

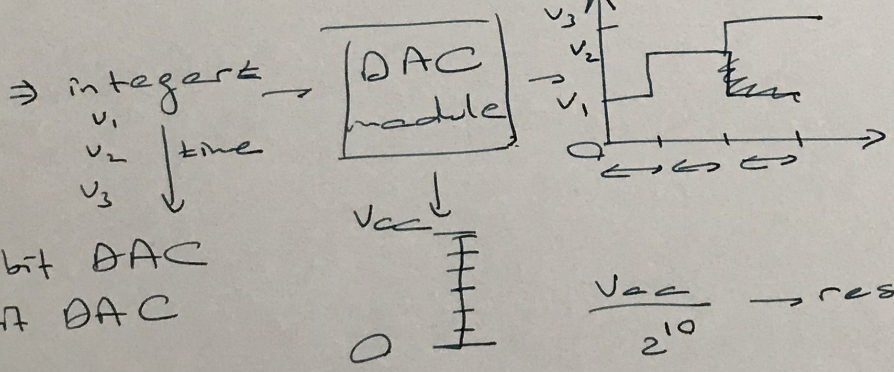
Digital value (int) → DAC → analog voltage (voltage)

(I) The microcontroller has a specific Digital to Analog Converter (DAC) module.
⇒ This is not mandatory.
If we have the DAC module, ZoH circuit.

→ Pulse Width Modulation (PWM)
(II) PWM signal can be generated by timers.
⇒ This option is more popular.

(I) I have the DAC module.
ZoH (Zero Order Hold) setup.

⇒ integers → DAC module → dac output

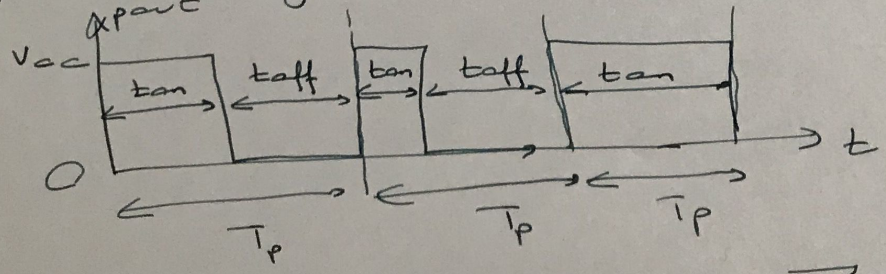


staircase like output.

12-bit DAC
10-bit DAC

$\frac{V_{cc}}{2^{10}}$ → resolution.

(II) PWM signal



$\frac{t_{on}}{T_p}$ = Duty cycle

→ We also need a smoothing filter.

Average output voltage

$V_{avg} = \frac{t_{on}}{T_p} V_{cc}$