9 October 2015

INFO-0064 Embedded systems

Exercise session 3

Introduction to microcontrollers III

Kevin Hogan

kevin.hogan@ulg.ac.be

Planning for today

- Practical information
- ADC
- PWM
- Application example: Dimmer

Practical information

- Someone from each group, send me an email **today**, with every member of your group as copy.
- If you still don’t have any group, send me an email.

- Each group will have a small budget to buy components on **be.farnell.com** or **befr.rs-online.com**

PWM module

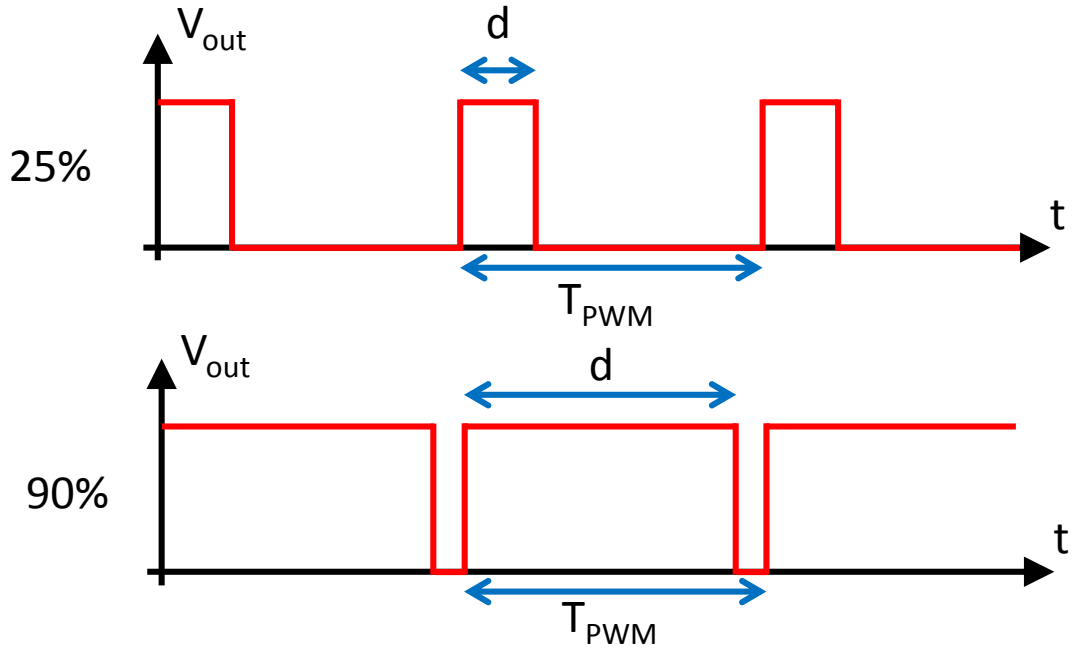
- Pulse Width Modulation

Two parameters:

➔ T_{PWM} : PWM period

➔ d : Duty cycle

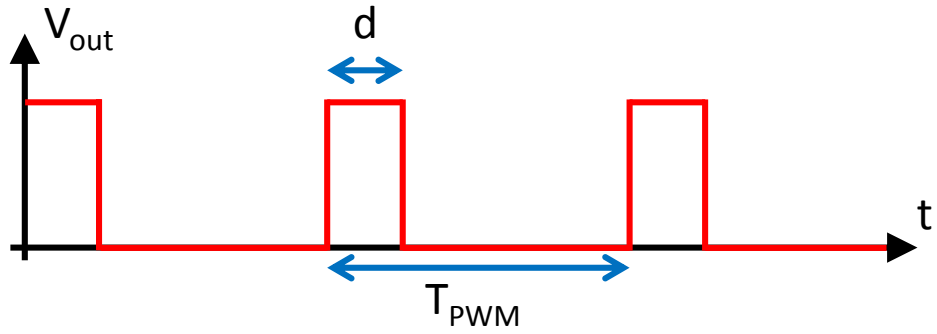
We have a modulation of the average value of the signal

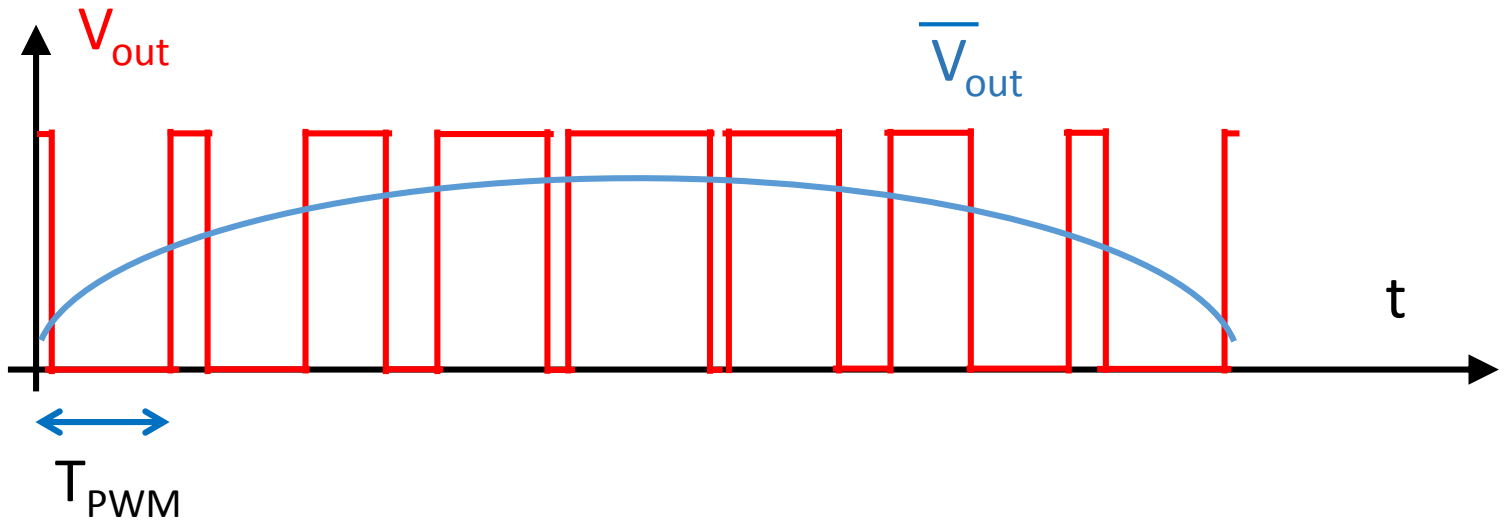


$$\overline{V}_{out} = \frac{1}{T_{PWM}} \int_0^{T_{PWM}} V_{out}(t)\, dt = d\, V$$

PWM module

- Pulse Width Modulation
  - ➔ T_{PWM} : PWM period
  - ➔ d : Duty cycle

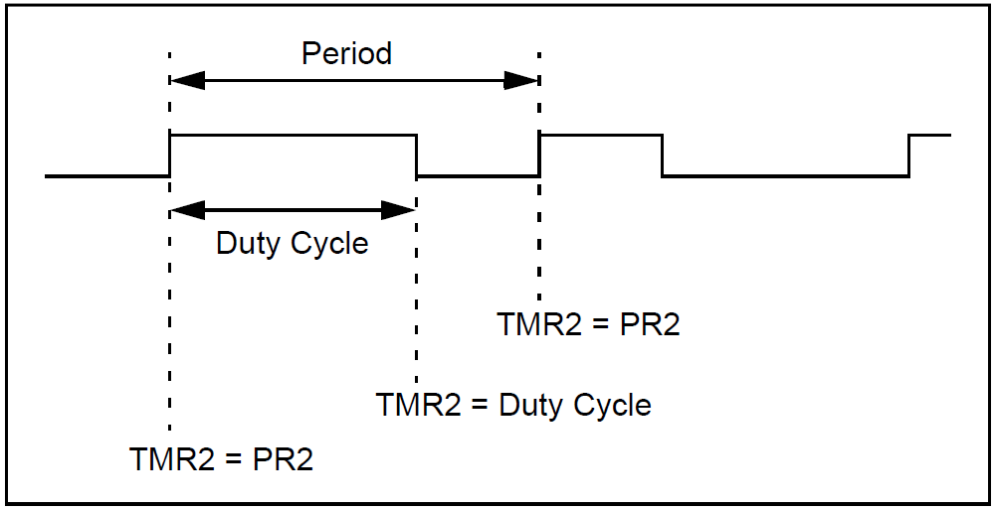




PWM module: PIC18F4620

- 10-bit resolution PWM output = 10-bit duty cycle [0..1023]
- Timer2 is used
- PWM period is set by PR2 register + TMR2 Prescaler
- PWM duty cycle is set by CCPRxL + CCPxCON<5:4>

FIGURE 15-4: PWM OUTPUT



EQUATION 15-1:

$$\text{PWM Period} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

EQUATION 15-2:

$$\text{PWM Duty Cycle} = (CCPRxL:CCPxCON\langle 5:4\rangle) \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

PWM module: PIC18F4620

15.4.4 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCPx module for PWM operation:

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPRxL register and CCPxCON<5:4> bits.
3. Make the CCPx pin an output by clearing the appropriate TRIS bit.
4. Set the TMR2 prescale value, then enable Timer2 by writing to T2CON.
5. Configure the CCPx module for PWM operation.

```
movlw  0xFF          ;
movwf  PIR2          ; set the PWM period

movlw  0x3F          ;
movwf  CCPR1L        ;
bcf    CCP1CON , 4   ; set the 10-bit PWM
bcf    CCP1CON , 5   ; duty cycle

bcf    TRISC , 3     ; RC2 = output

movlw  b'00000111'   ;
movwf  T2CON         ; configure TMR2

movlw  b'00001100'   ;
movwf  CCP1CON       ; set the PWM
```

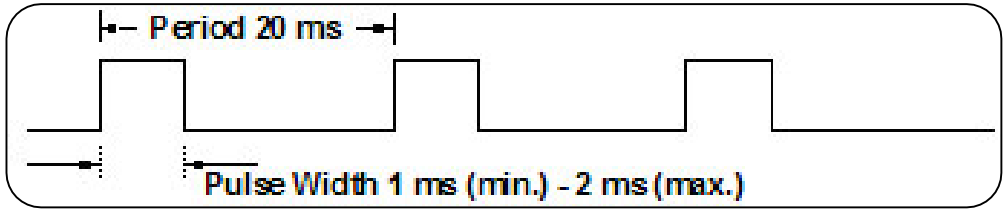
PWM exemple: Servomotors

- Servomotor ➔ send a given PWM signal to obtain a certain position
- 3 inputs: Vdd, GND, PWM

Exemple: Servo HS-85BB

- Power supply: 4,8V .. 6V DC
- PWM control:
  - Amplitude: 3V .. 5V
  - **Period = 20 ms**
  - **Duty cycle = [600 .. 2400] µs**

The period is out of rang! What can we do?



Figures taken from:
http://www.servocity.com/html/hs-85bb__mighty_micro.html

PWM exemple: Servomotors

- Servomotor control

➔PWM using timer

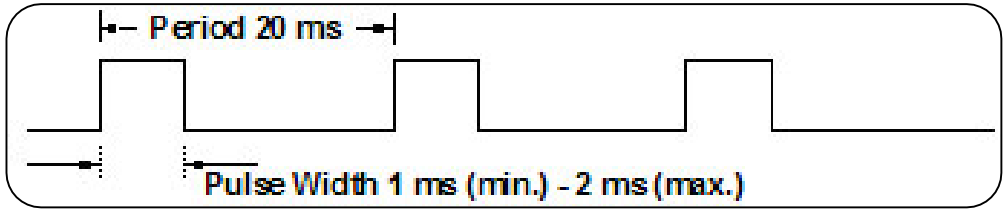
We have to configure the timer in such way that we have enough resolution to control the PWM.

Timer 16bits: 1,5 interrupts/period

Timer 8bits: 390 interrupts/period

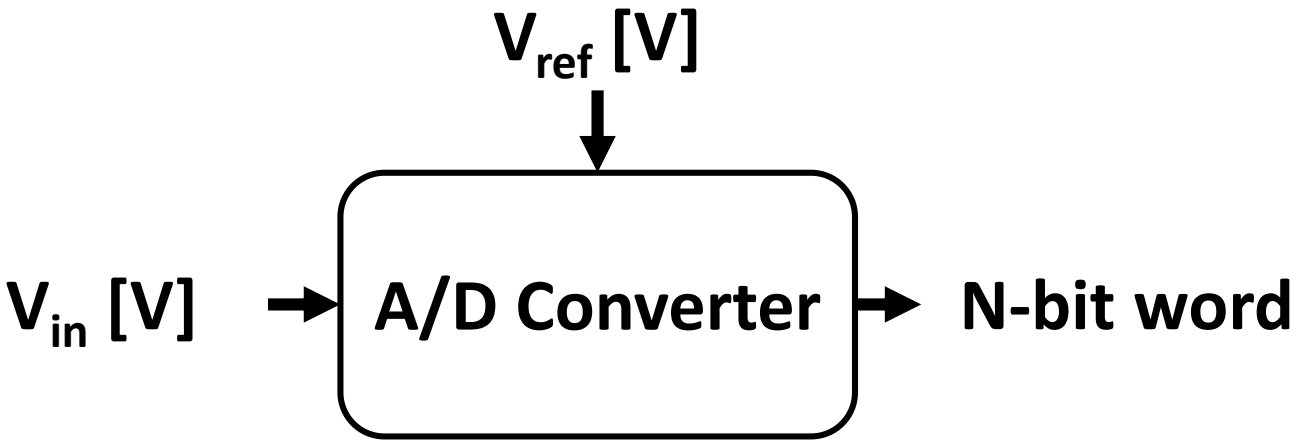
Timer 8bits + Prescaler 1:4:

➔ 97 interrupts/period



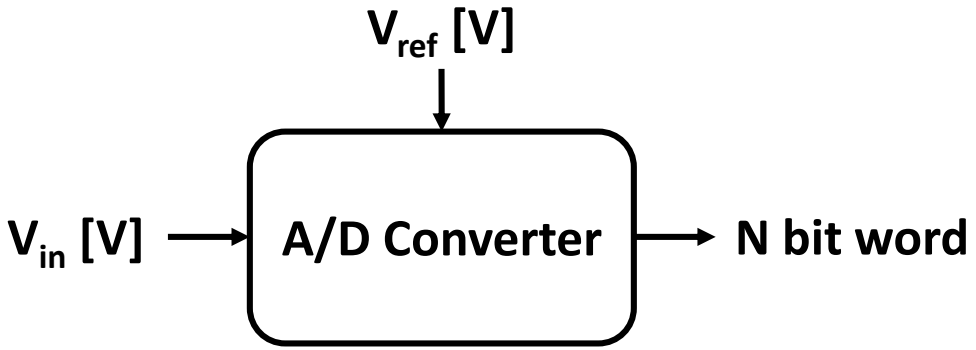
A/D Converter

- Concept



$$V_{meas} = \left(\frac{N\ bit\ word}{2^N}\right) V_{ref}$$

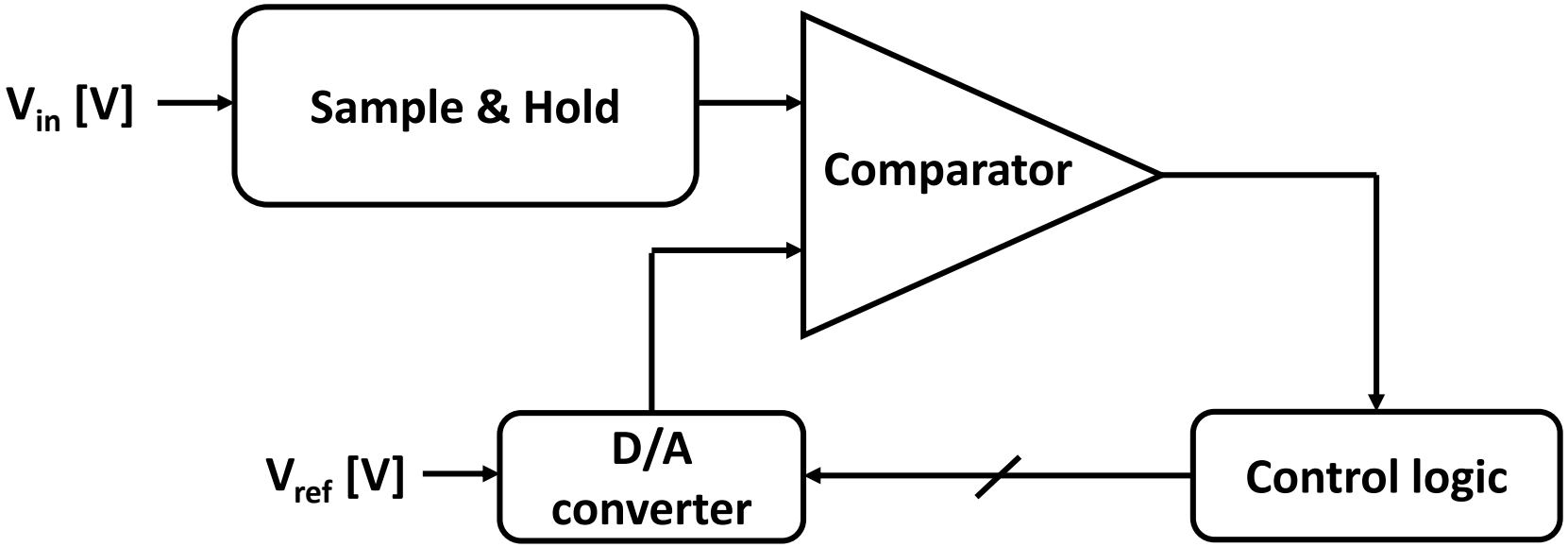
A/D Converter



How does it work?

➔ Successive approximation concept

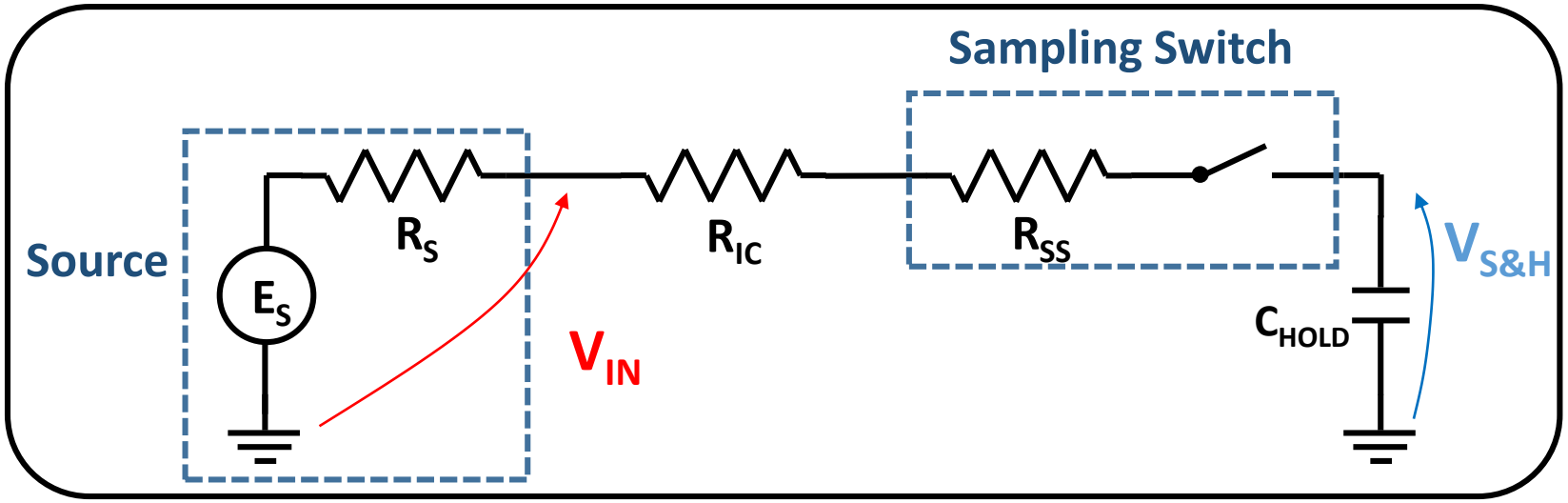
$$V_{meas} = \left(\frac{N\ bit\ word}{2^N}\right) V_{ref}$$

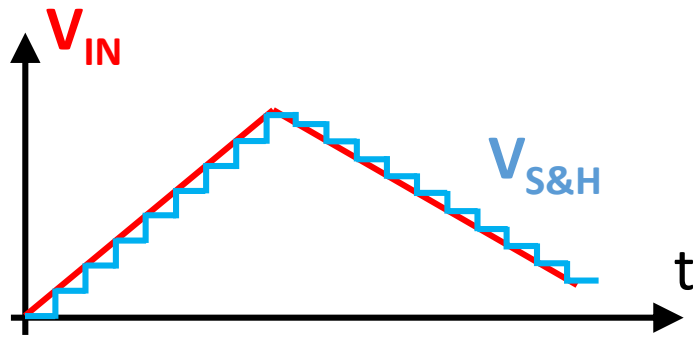


A/D Converter

- Sample & Hold circuit

➔ samples the input signal and holds the value of the voltage constant during the time of the conversion





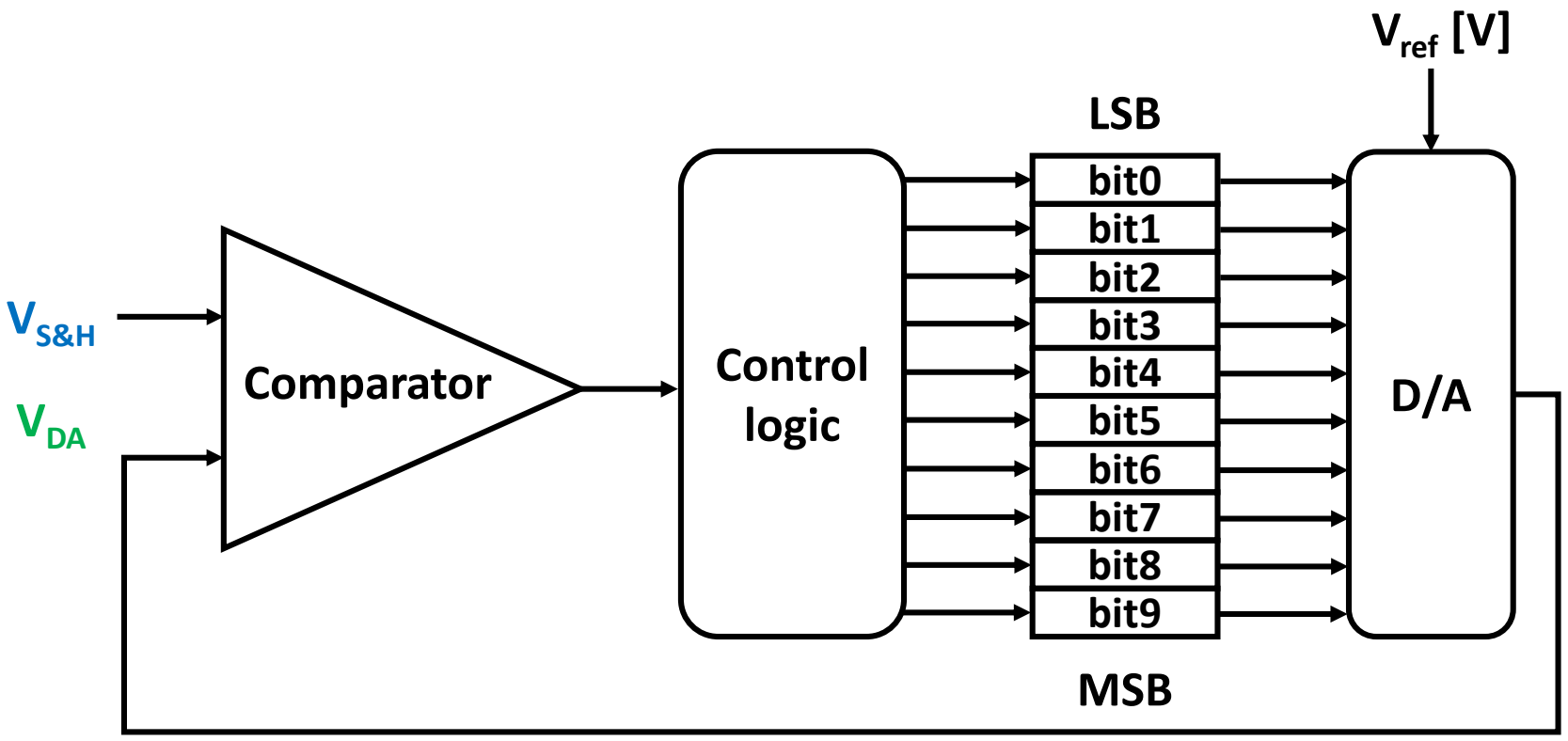
- Based upon a switch which samples the voltage and stays on long enough in order to let the capacitor fully charge.
- The minimum acquisition time ($t_{ACQ,MIN}$) is the time needed to charge the capacitor to the input channel voltage level. It is proportional to the RC circuit time constant $\tau = (R_s + R_{ic} + R_{ss}).C_{hold}$

In order to minimize the acquisition time, **the maximum recommended impedance for analog sources (R_S) is 2.5 kOhm** for the PIC18F4620! (cf. datasheet)

A/D Converter

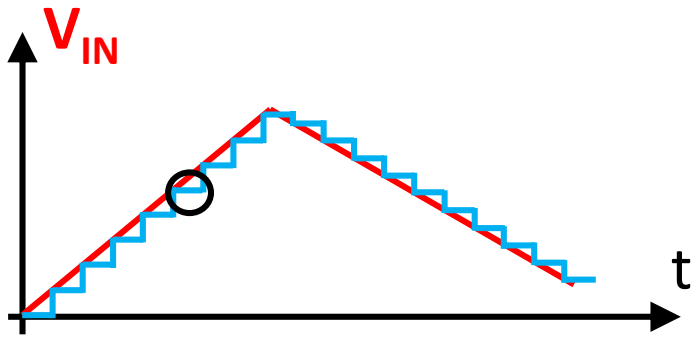
How does it work?

➔ Successive approximation concept

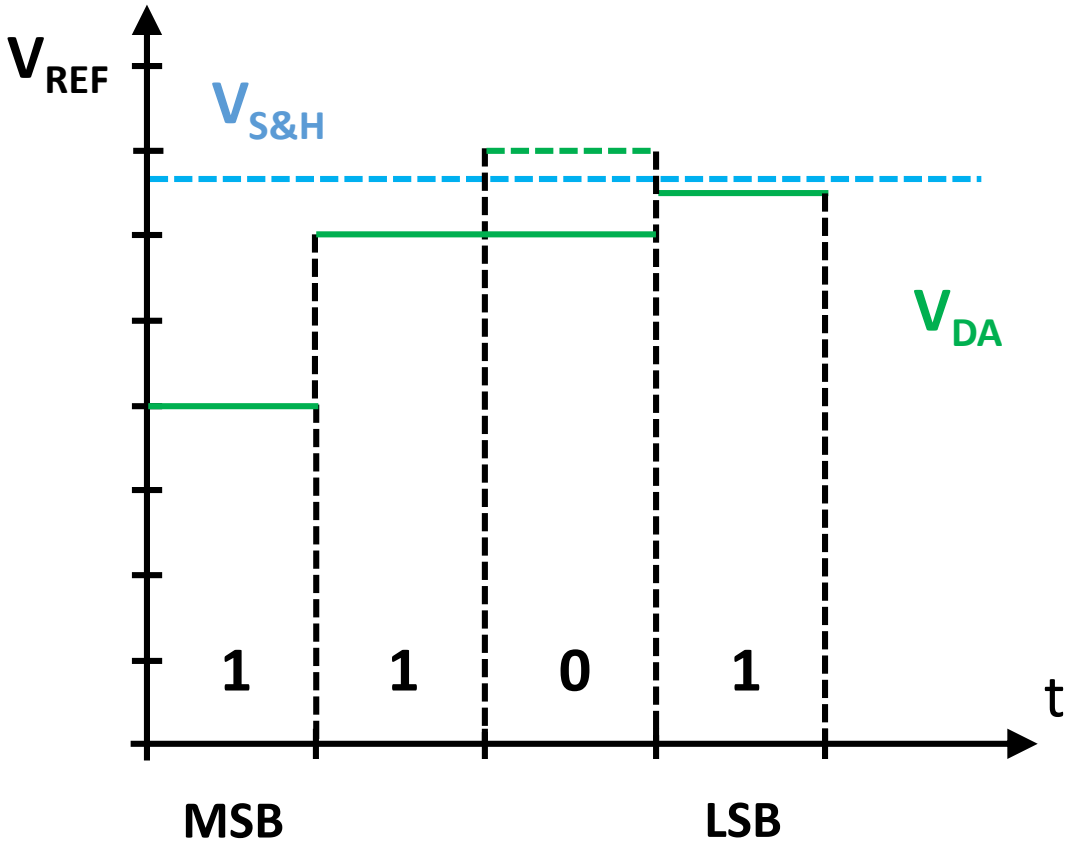


- V_{DA} = voltage corresponding to the value stored in the register:
  $V_{DA} = (register/ 2^{10}) * V_{REF}$
- Initially, the register is cleared (V_{DA} = 0 V)
- Starting from the MSB to the LSB:
  Current bit is set;
  If $V_{DA} < V_{S\&H}$, current bit stays set
  If $V_{DA} > V_{S\&H}$, current bit is clear
- At the end, the value stored in the register corresponds to the converted voltage

A/D Converter



4-bits register exemple



$V_{DA} = (register/2^4) * V_{ref}$

MSB			LSB	
1	0	0	0	→ $V_{DA} = V_{ref}/2$, $V_{S\&H} > V_{DA}$ → bit3 = 1
1	1	0	0	→ $V_{DA} = (3/4)*V_{ref}$, $V_{S\&H} > V_{DA}$ → bit2 = 1
1	1	1	0	→ $V_{DA} = (7/8)*V_{ref}$, $V_{S\&H} < V_{DA}$ → bit1 = 0
1	1	0	1	→ $V_{DA} = (13/16)*V_{ref}$, $V_{S\&H} > V_{DA}$ → bit0 = 1

A/D Converter: PIC18F4620

- Output = 10-bit digital number = [0 .. 1023]
- 3 SFR are used to control the A/D converter:
  - **ADCON0**: on/off, go/done, channel selection
  - **ADCON1**: Vref, A/D Port configuration bit
  - **ADCON2**: L/R justification, acquisition time & conversion clock
- The result is stored in two SFR: **ADRESH** (MSB) and **ADRESL** (LSB)

$$V_{meas} = \frac{result}{1024} V_{ref}$$

How to use it?

A/D Converter: PIC18F4620

The following steps should be followed to perform an A/D conversion:

1. Configure the A/D module:
   - Configure analog pins, voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D acquisition time (ADCON2)
   - Select A/D conversion clock (ADCON2)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time (if required).
4. Start conversion:
   - Set GO/$\overline{\text{DONE}}$ bit (ADCON0 register)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared

   OR
   - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit ADIF, if required.
7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2 TAD is required before the next acquisition starts.

A/D Converter: PIC18F4620

```
movlw  b'00001110'
movwf  ADCON1        ; configure ADCON1

movlw  b'00000000'
movwf  ADCON0        ; configure ADCON0

movlw  b'00000001'
movwf  ADCON2        ; configure ADCON2

movlw  b'00000001'
movwf  ADCON0        ; A/D converter on
```

```
; clear interrupt flag for ADC
bcf    PIR1 , ADIF
; enable interrupt for ADC
bsf    PIE1 , ADIE
```

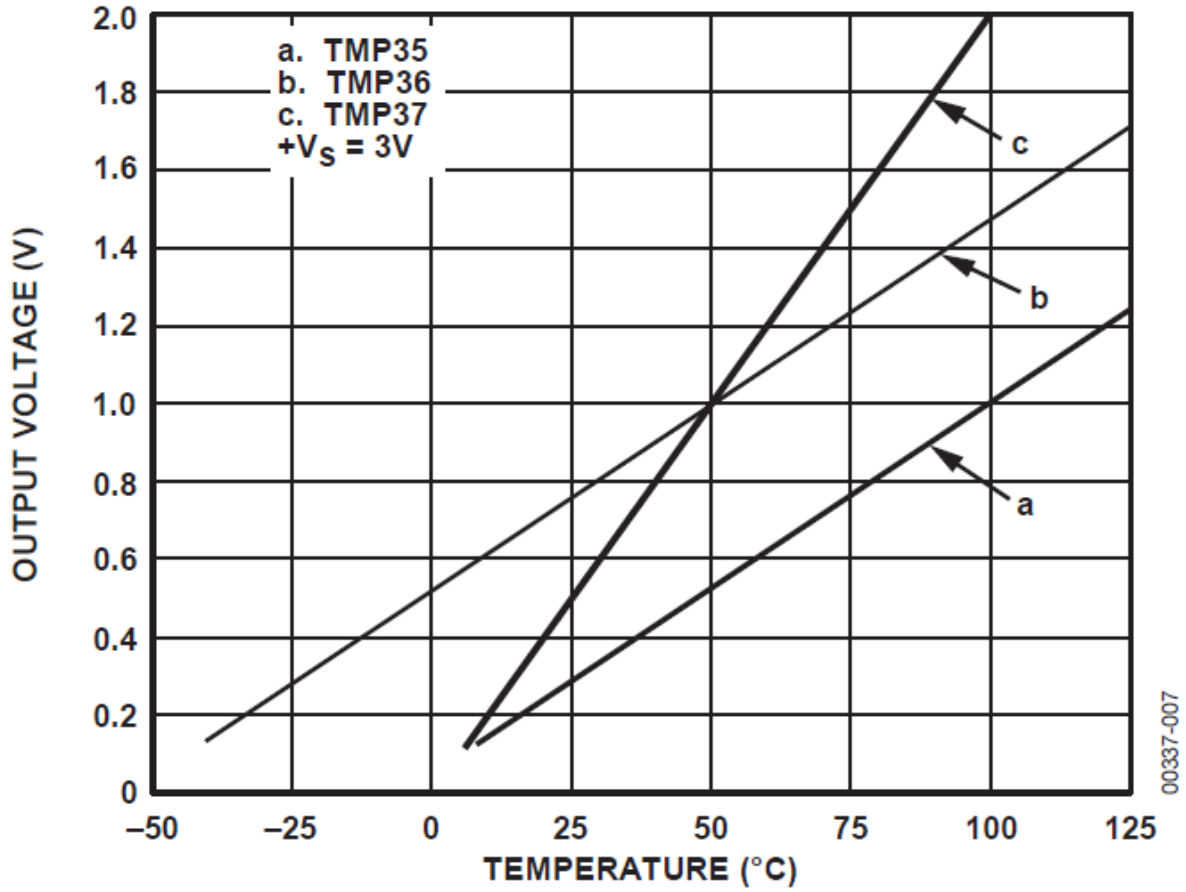
```
; start conversion
bsf    ADCON0 , GO
```

```
;clear the interrupt flag
bcf    PIR1 , ADIF

(use ADRESH and ADRESL)

return
```

A/D Converter: application exemples

Temperature sensor

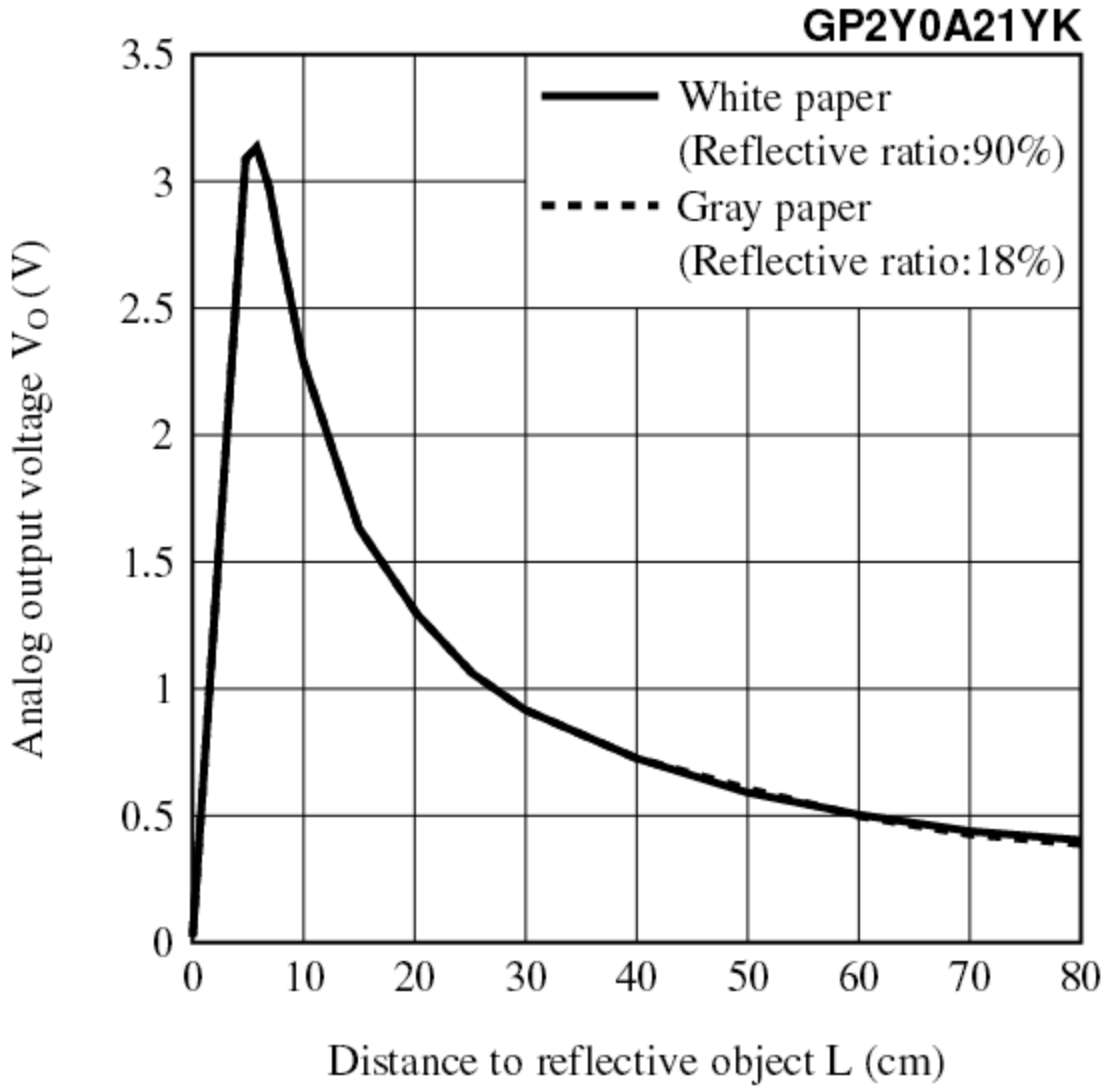
Exemple: TMP35



A/D Converter: application exemples

Proximity sensor

Exemple: Sharp GP2Y0A21YK



Application exemple: Dimmer

Goal: control the luminosity of a led with a potentiometer + blinky

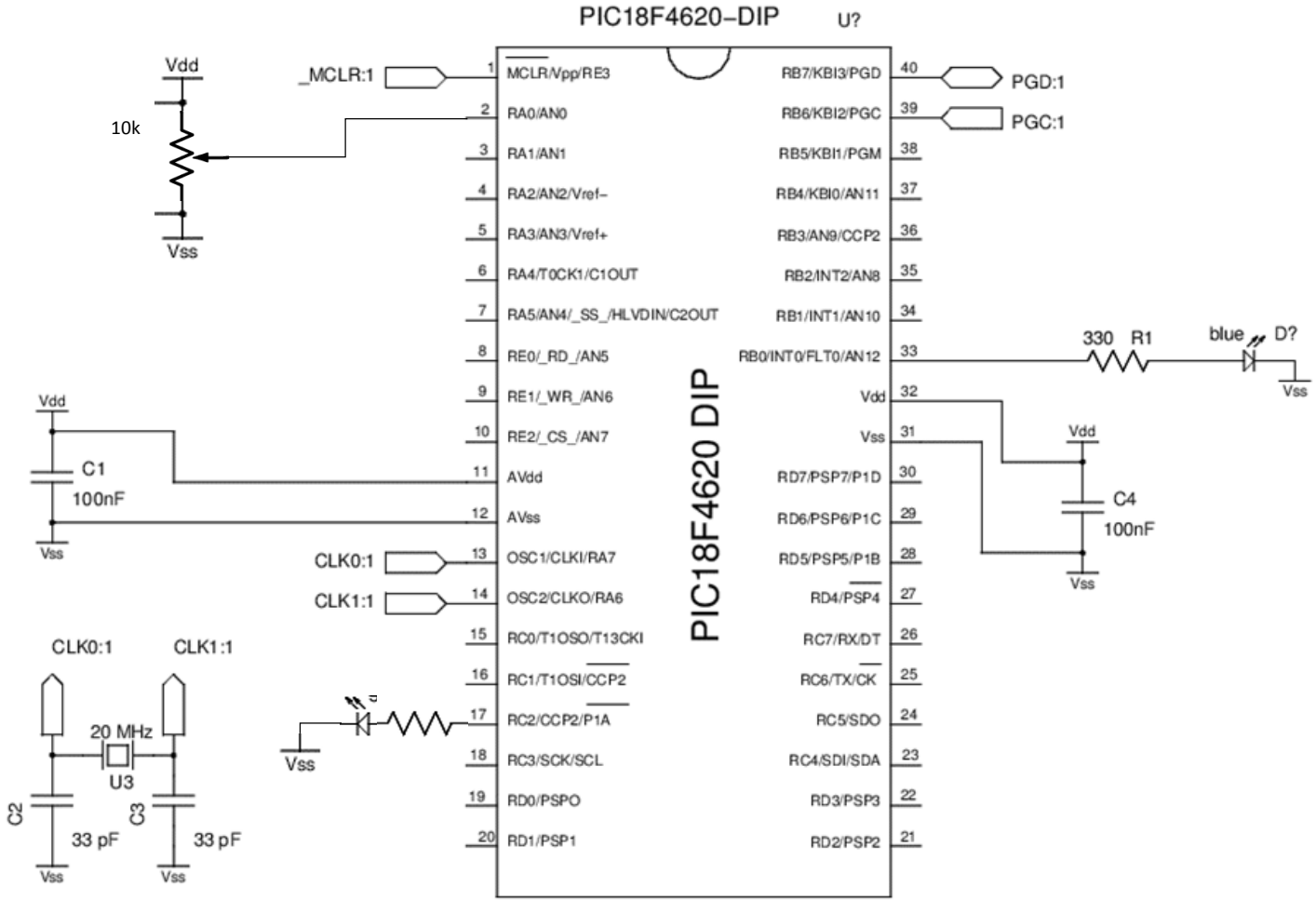
How can we implement that?

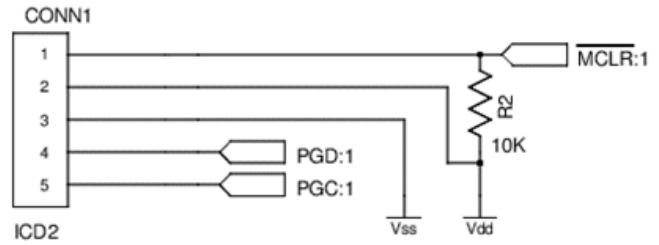
LED powered by a PWM signal with a varying duty cycle

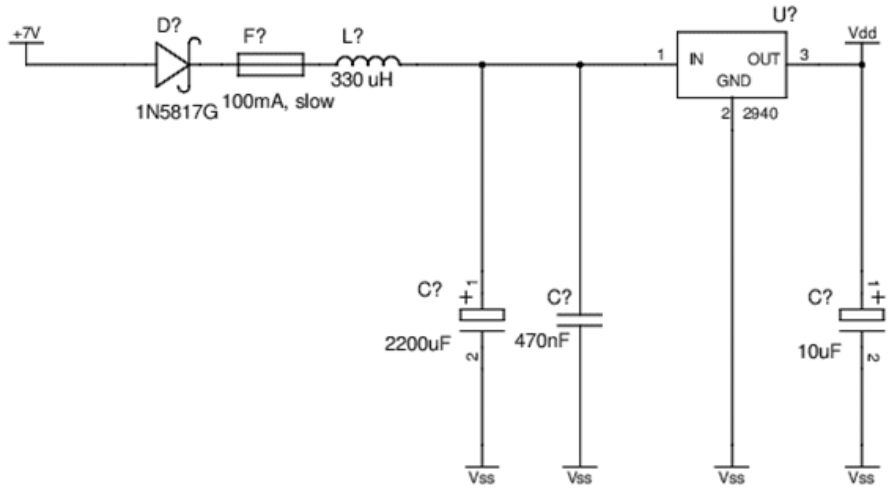
Voltage of the potentiometer ➔ ADC

Acquired voltage becomes the new duty cycle

Application exemple: Dimmer







Ideas for your project

- Wii Nunchuck
- Line tracking robot
- Spirograph
- Inverted pendulum
- Propeller clock