TTK4155

Industrial and Embedded Computer Systems Design

NTNU – Trondheim
Norwegian University of
Science and Technology

Lab lecture 7

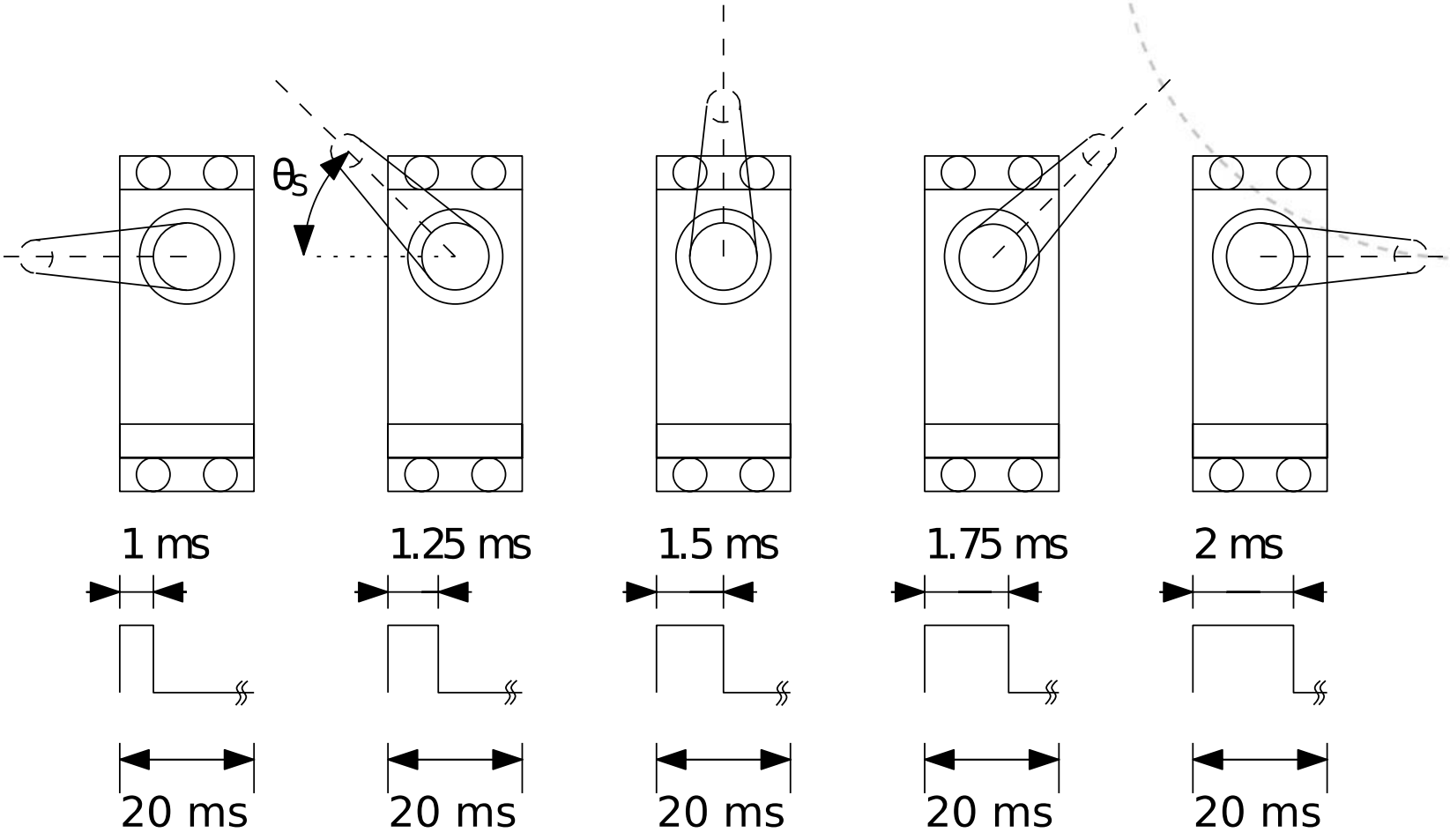
- Controlling a servo
- Detecting lost ball

Exercise 7: Controlling servo and IR

- In this exercise you will
  - Use the Arduino to generate a PWM signal to control a servo.
  - Use the joystick to control the pulse width of the PWM.
  - Read the signal from an IR photo-diode.
  - Detect when the IR-beam is blocked and count the score.

Controlling the servo

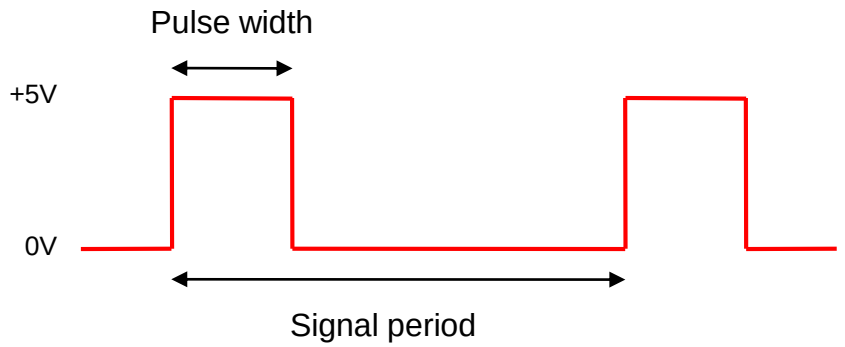
- The servo is controlled using a 50 Hz signal (signal period = 20 ms) (T = 1/f).
- The angle of the servo is determined by the ON pulse width (1.5 ms corresponds to centre position).
- By varying the pulse width we can control the angle.
- The pulse width must never be outside the range 0.9-2.1 ms.



Pulse Width Modulation (PWM)

- A way for the MCU to generate an analog voltage
- Used a lot within motor control since it enables combination of maximum torque with low speeds

- Signal period
- Pulse width
- Duty cycle



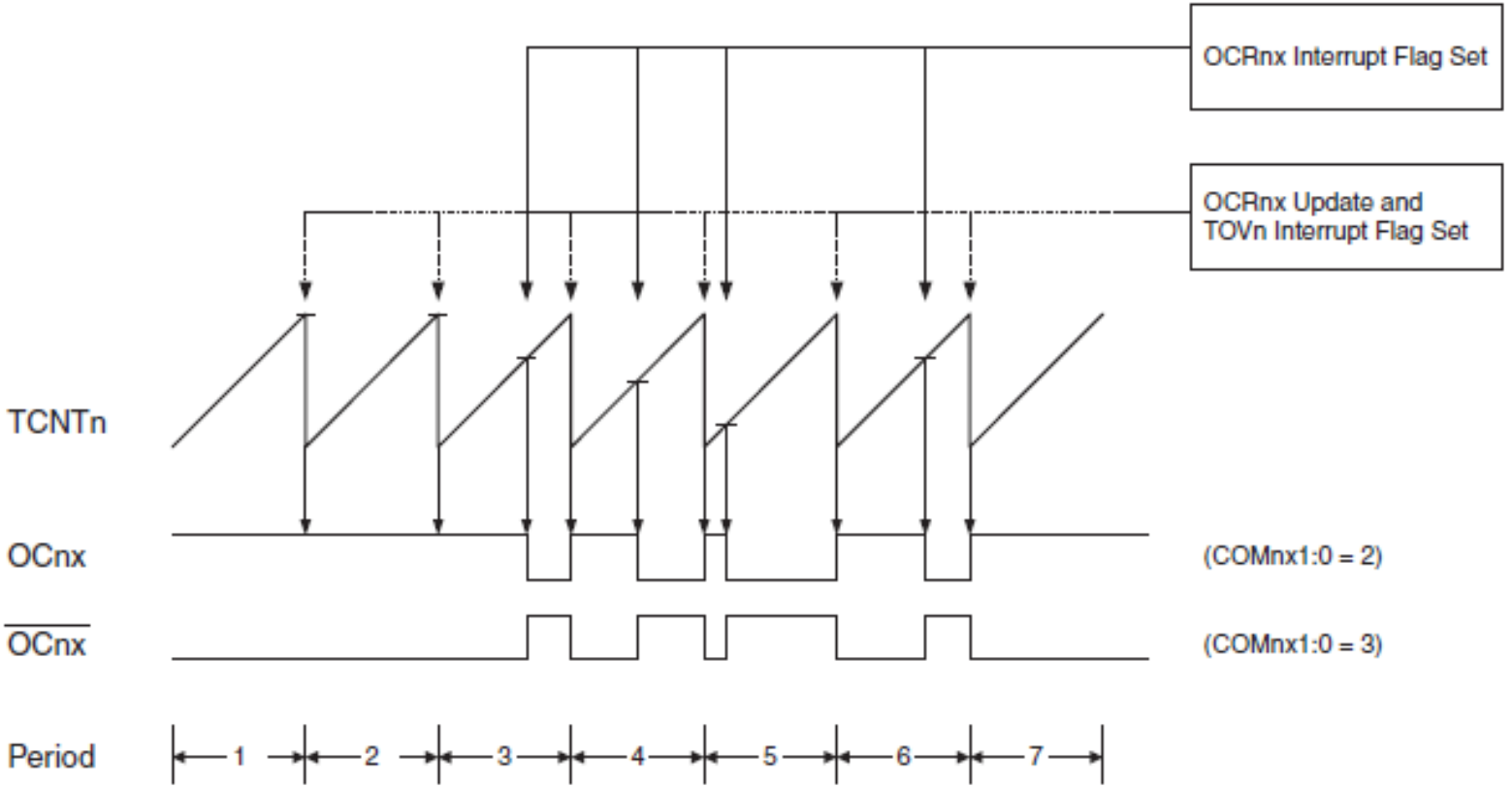
Duty cycle = Pulse width/Signal period

- Average voltage = 5V · Duty cycle

How to generate a PWM signal

- 8-/16-bit timers/counters clocked internally via pre-scalers.
- These count independently of the processor.
- One of many possible applications is PWM.
- Different types of PWM signal can be generated.
  - Fast and Phase correct in case of AVR.

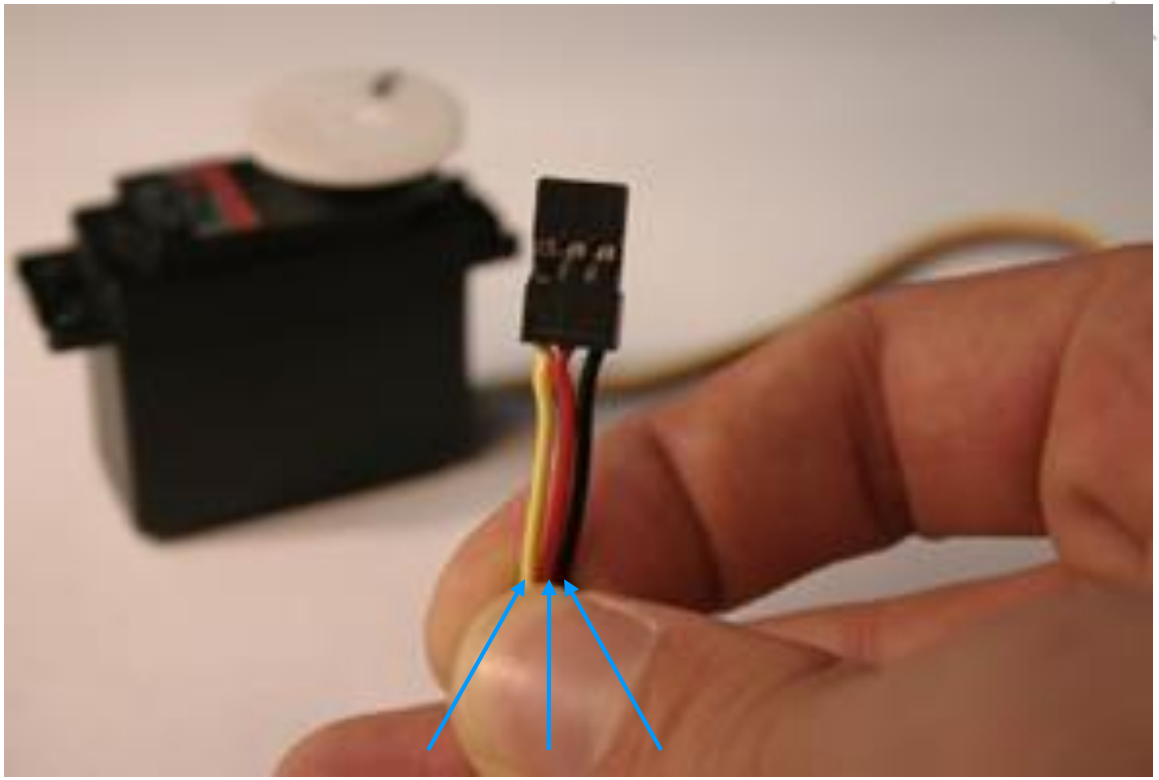
PWM


OCRnx Interrupt Flag Set
OCRnx Update and
TOVn Interrupt Flag Set
TCNTn
OCnx
(COMnx1:0 = 2)
OCnx
(COMnx1:0 = 3)
Period
1
2
3
4
5
6
7

Get the right pulse width and period

- Manual timer setup:
  - TCCRxA -> timer mode and o/p pin control.
  - TCCRxB -> CLK pre-scaling and WGM bits.
  - OCRnA/B for freq. and duty cycle values.
- Why use prescaling?
  - 16 bit timer → largest value in top register = $2^{16} - 1 = 65\,535$
  - Choose a pre-scaler and a top value that gives a period > 0.020 s
- Use “Fast PWM” mode.
- Make sure you don’t use a timer that is already utilized by your code.
- Make sure pulse 0.9 <ON period< 2.1 (ms) before connecting the servo.

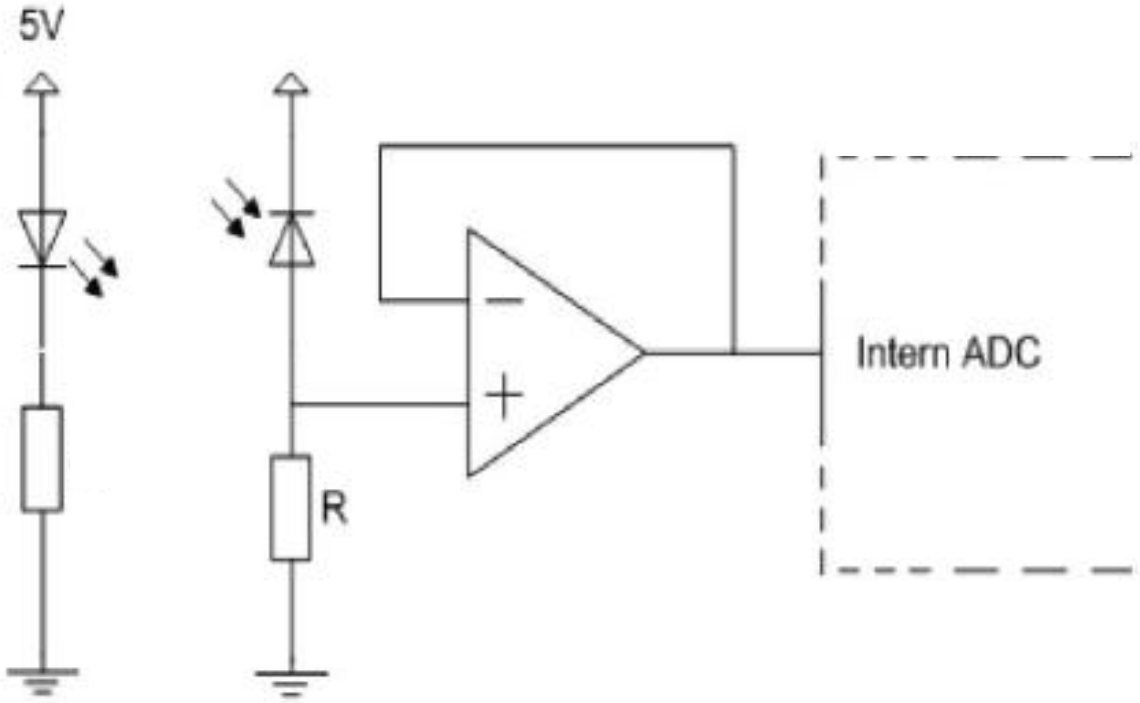
Connecting the servo



IR

- IR diode (sender).
- IR photo-diode (receiver).
- The current through the photo-diode is dependent on the amount of light received.
- To measure the current, we need a resistor to measure the voltage over.
- Choose the resistor value based on the information given in the photo-diode's datasheet.

Schematic



Connecting the IR circuitry

- Use the small bread-board.
- Use a voltage follower (op-amp).
- Connect it as close to the photo-diode as possible to increase immunity to noise.
- Use the internal ADC on the ATmega2560.
  - You can use vacant pins on PORT F.
- Check that the IR diode is working using a digital camera (e.g. mobile phone).

Internal ADC

- 10-bit resolution.
- 8 available channels on Port F.
- Manual configuration:
  - Use AVCC as reference.
  - Use single conversion.
  - The ADC's operation is well described in the datasheet.
- Use digital filtering ($y_n = ¼(x_n + x_{n-1} + x_{n-2} + x_{n-3})$).

Common issues with CAN

- Not understanding SPI waveform.
  - State of MISO signal is/maybe undefined during 1st and 2nd transaction.
- Hardware issues.
  - Use CLK pin to make sure CAN controller is working.
- Bit modify function/instruction.
  - Double check registers before and after using this function.
- Termination resistors.
  - Between CANH and CANL. No GND/Vdd connection.
- One AVR/C project/program => one main function.
  - Use two AVR projects for two nodes.
  - Files can be shared or copied to two different folders.

Extras….

- Keep extras inside embedded realm.
- Some ideas;
  - Well documented code e.g. UML.
  - Dual buffer and creative use of OLED e.g. animations etc.
  - Online tuning of PID controller.
  - Use of cell phone's sensor e.g. gyro, accelerometer (requires extra comm. module)
  - Explore & use modules from USB card e.g. buzzer, CAN etc.
  - More ideas => at the end of lab compendium text.
- In short, extras can be simple and still earn points.

Questions?