Practical introduction to robotics

Concepts covered in class

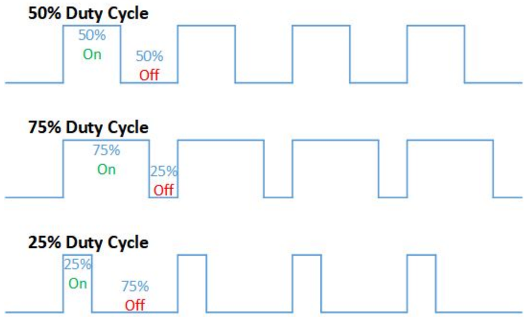
What is a robot?

An autonomous machine capable to physically act in the real world and to replace human action.

Concept 1: PWM

Problem: How to control the speed of a motor without changing the voltage ?

Solution: Turning the power ON and OFF very fast. The ratio of time spent ON will be proportional to the speed of the motor.



Key parameter: duty cycle = time_ON / (time_ON + time_OFF)

Key insight: It works the same for LEDs, and other electrical actuators.

How it is made: A dedicated circuit can generate PWM signals on the microprocessor. We simply have to pass the duty_cycle that we want. We do this via a register.

Code: https://github.com/jgrizou/CRI-UE-Robotics/tree/master/code/0_PWM

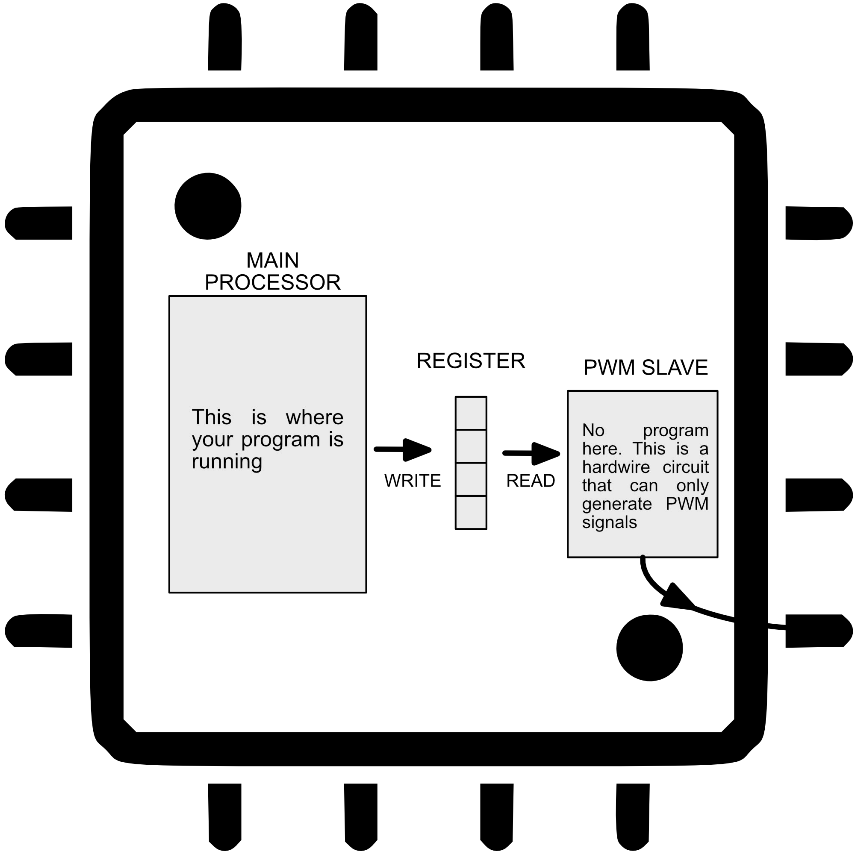
More info at: https://en.wikipedia.org/wiki/Pulse-width_modulation

Concept 2: Registers

Problem: How to communicate to the PWM dedicated circuit on the board?

Solution: Both the main processor and the PWM slave circuit share a few bits in memory. We call this dedicated shared memory space a register. The main processor can write to it and the PWM circuit change the duty_cycle according to the information in the register.

In class: Remember when a student could synchronise the classroom in playing music while drawing an elephant on the whiteboard.


MAIN PROCESSOR
This is where your program is running
WRITE
REGISTER
READ
PWM SLAVE
No program here. This is a hardwire circuit that can only generate PWM signals

Concept 2: Registers

More info at:

- https://en.wikipedia.org/wiki/Hardware_register
- http://www.robotoid.com/appnotes/programming-data-and-port-registers.html
- If you have a hard time with this, you should read more about von Neumann architecture that is the basis of all computers hardware organisation: https://en.wikipedia.org/wiki/Von_Neumann_architecture

Key insights:

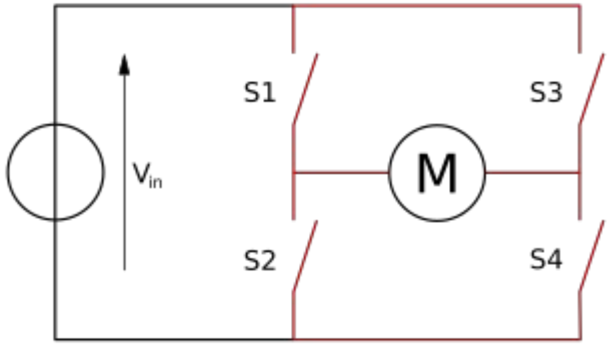
- Always remember that humans decided how to build these things. Registers are only bits of information that are either 0 or 1. How they are used has been defined by an engineer in the confine of its office.
- This is were reading the documentation helps to understand how they decided to encode the information. And what to write in those registers to get the correct duty cycle. For the PWM on the Arduino, this is all hidden from us because we use high-level functions. But you should understand the principle of a register nonetheless.

Concept 3: H-Bridge

Problem: How to change the direction of a motor without physically (by hand) changing its polarity ?

Solution: Using four transistor, we can control the direction of the current going through the motor. This circuit is called an H-Bridge because of its shape.

More info at: https://en.wikipedia.org/wiki/H_bridge



Key insights:

- It works the same for LEDs, and other electrical actuators.
- It is what you find on electronic component called: “DC motor controller”

Concept 4: Interruptions

Problem: How to track the change of state of a PIN without constantly having to look at it ?

Solution: Same as for the PWM, there is a dedicated circuit that only look at the PIN and trigger an interruption in your program when there is a change in value.

In class: Remember my analogy about asking a friend to tap your shoulder when a specific event happens.

More info at:

- https://en.wikipedia.org/wiki/Interrupt (concept of interrupt)
- https://en.wikipedia.org/wiki/Interrupt_handler (the function that is called on interruption)

Code: https://github.com/jgrizou/CRI-UE-Robotics/tree/master/code/1_Interrupt

Key insights:

- Interrupts function should be very fast to execute. Just a few operations should be executed. This is to avoid having another interruption happening while the first one is not finished which will explode the stack due to context switching.

Concept 5: Encoders

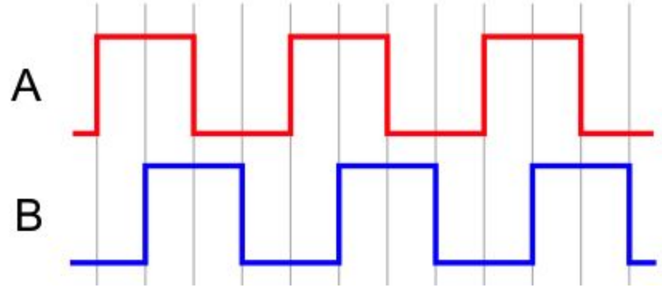
Problem: How to track the position of a motor ?

Solution: We put a sensor (mechanical, optical, magnetic, etc) on the axis that triggers Nth time per rotation of the wheel. We then can count the number of change state event and know how much the wheel has moved. We use interruptions to count this precisely.

Direction of rotation: With only one sensor we can known how much the motor moved but not the direction of rotation. For that we place two sensors A and B. If A is triggered before B then we know that the motor turns in one direction, otherwise it turns in the other.

More info at:

- http://www.creative-robotics.com/quadrature-intro



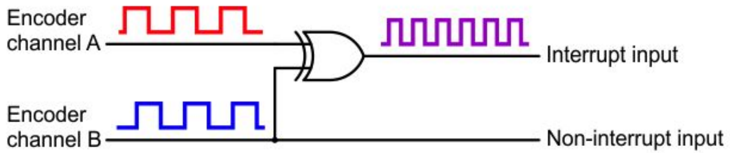
Key insights:

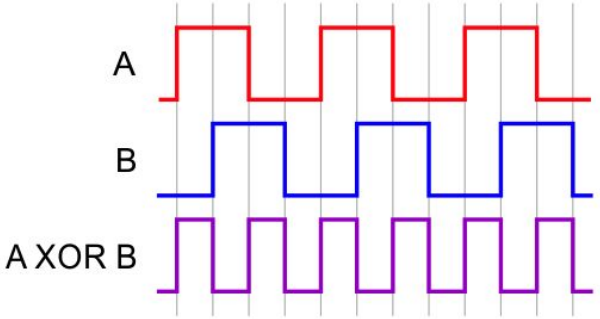
- Because robots act in the physical space, we are always down to physics to measure things. You can use whatever sensor you want but there is not choice than making a physical measurement.

Concept 5: Encoders

In practice: During the course, we saw that the construction of the robot used an XOR gate to combine A and B signals. They did that because of limitation in the number of interrupts on the microprocessor.

Code: https://github.com/jgrizou/CRI-UE-Robotics/tree/master/code/1_Interrupt

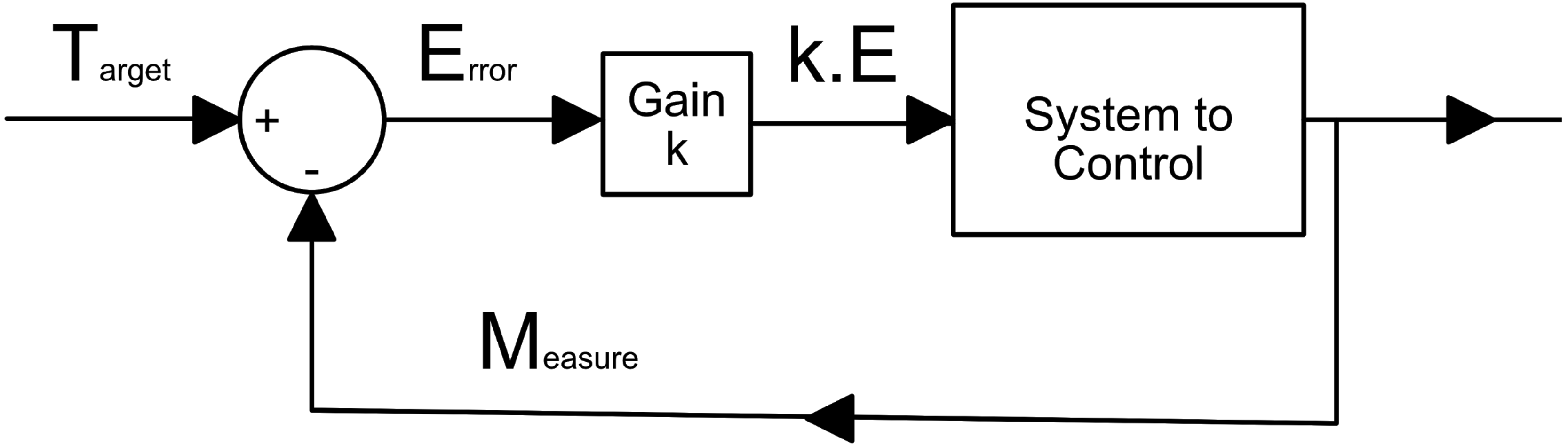




Concept 6: Feedback Loops

Problem: How to maintain the robot in a specific state? For example, keep the robot in the middle of a line, or keep the robot facing in one direction.

Solution: We condition the action of the motor on the values of the sensors. This is called a feedback loop. A lot of theory has been developed to build such control loops in the field called control theory: https://en.wikipedia.org/wiki/Control_theory



Concept 6: Feedback Loops

We have seen three examples in class:

1. Move the robot for 10cm. This is the simplest form of feedback loop. If the robot is has not made 10cm move forward, else stop.

Code: https://github.com/jgrizou/CRI-UE-Robotics/tree/master/code/1_Interrupt/4_TravelDistance

2. Follow a line with the robot. We designed a set of condition for the robot to come back on the line. If the line was on the left, the robot turns left. If on the right, the robot turns right. If on the line, move forward. Very simple set of instruction resulting in the robot following a line. This is a discrete feedback loop because we use a lot of "if" condition to select what action to do.

Code: https://github.com/jgrizou/CRI-UE-Robotics/tree/master/code/2_LineSensors

3. Keep the heading of the robot in one particular direction. This is a continuous feedback loop because there was no "if" condition but rather we used a continuous measure of the error between the target angular position and the measured angular position. Then the commands to the motors was simply proportional to the error. No need for "if" conditions, the robot will simply do nothing if the error is 0.

Code: https://github.com/jgrizou/CRI-UE-Robotics/tree/master/code/4_Gyro