Final Challenge

Motor control – closed loop controller

{Learn, Create, Innovate};

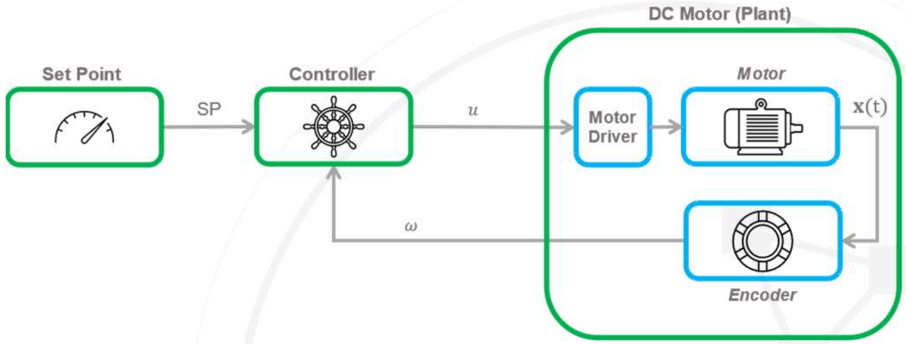
Final Challenge

Introduction

This challenge is intended for the student to review the concepts introduced in this course.

The activity consists in controlling the speed of a DC Motor.

- The motor speed, must be controlled using an external computer, a microcontroller, and a motor driver.
  - See following slide for requirements.



Final Challenge

- The “/Controller” node must run on the MCU.
- It should receive a signal in the [-1,1] interval. Such signal is the duty cycle.
  - The setpoint should be written in a /setpoint topic.
  - The sign represents the direction of the motor, i.e., (+) CCW rotation, and (-) CW rotation of the motor.
  - The controller can be “P”, “PI” or “PID” controller (other controllers can be accepted upon agreement with the professor.).
  - The duty cycle percentage (%) range [0,100]% →[0,1].
- The node should publish:
  - The estimated in in the range $[-\omega_{max}, \omega_{max}]$ in $\left[\frac{rad}{s}\right]$ /angular_speed.
  - The sign (+.-) represents the direction of rotation obtained from the dual encoder channel
  - The ω is the angular speed, obtained by the pulses.

Final Challenge

- The /Setpoint node runs in a computer generates an input signal in the form of a
  - Constant, sine, square or sawtooth waves.
  - NOTE: Beware of abrupt changes in the current. It might hurt your H-Bridge
- The control node, must use a parameter file, for all the required signal generation variables.
- The sampling time and rate must be defined by the student.
- It is encouraged to use original and disrupting solutions in Python and MATLAB.
- You must be able to plot the reference and actual signal in rqt_plot.

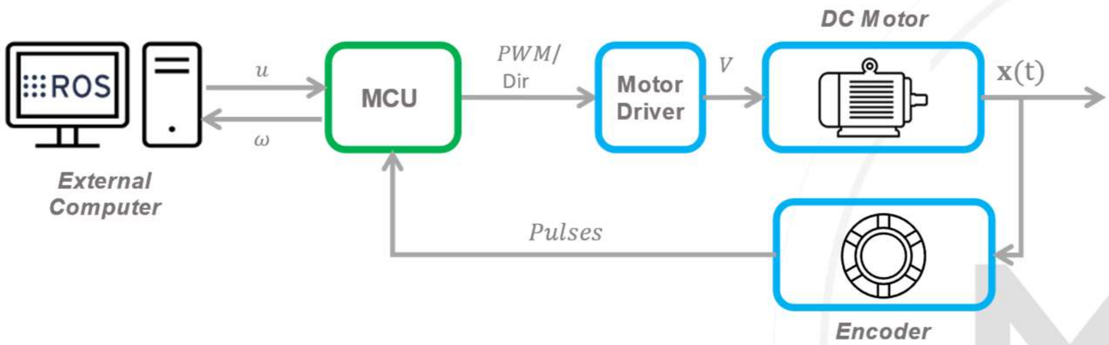
Closed loop motor control

- Develop a closed loop motor speed control.
- Modify the previous minichallenges to implement this challenge.
- You will need a motor with an encoder, and a power supply or battery pack.
- Determine an appropriate sampling time depending on your encoder.
  - Encoders can be read by certain communication protocols or by GPIO interruptions.
- Use different timers to publish and to sample data.

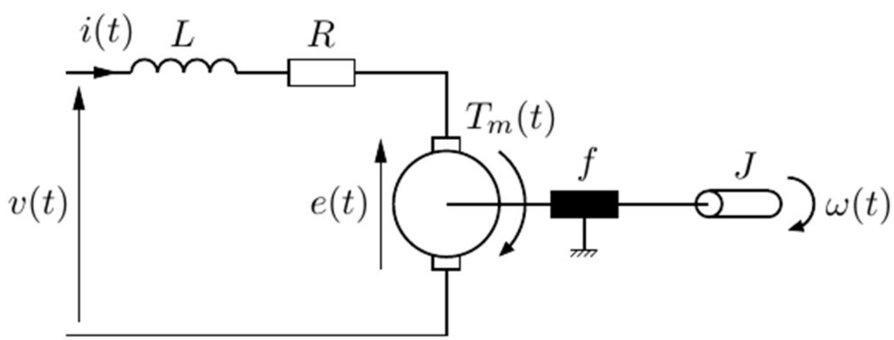
Final Challenge

Structure

- To perform this task, at least two nodes must be developed: /Setpoint, and /Controller.
- The /Setpoint nodes must run in the **external computing unit**, whilst the / Controller node must run in the **microcontroller**.

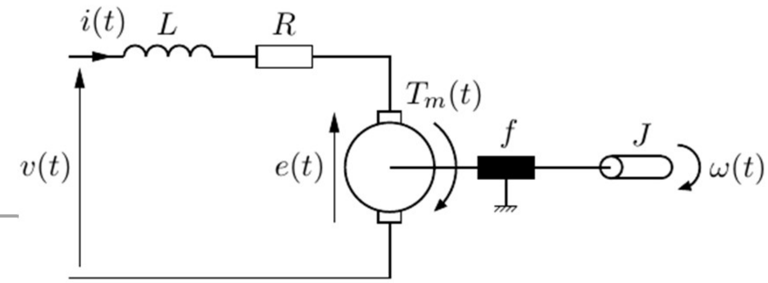


DC Motor Model



- R – resistor
- L – inductance
- ω – angular velocity
- J – moment of inertia
- β – friction coefficient
- k_m – torque/power ratio
- e – coil voltage

DC Motor Model



$$e = k_m\omega$$

$$\tau = k_m i$$

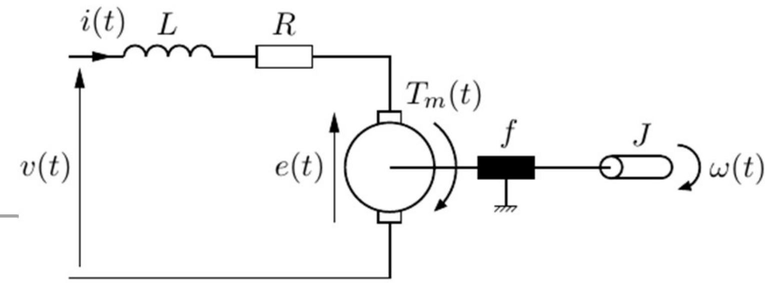
$$v = L\frac{di}{dt} + Ri + e$$

$$\tau = J\frac{d\omega}{dt} + \beta\omega$$

$$v = L\frac{di}{dt} + Ri + k_m\omega$$

$$k_m i = J\frac{d\omega}{dt} + \beta\omega$$

DC Motor Model



$$v = L\frac{di}{dt} + Ri + k_m\omega$$

$$k_m i = J\frac{d\omega}{dt} + \beta\omega$$

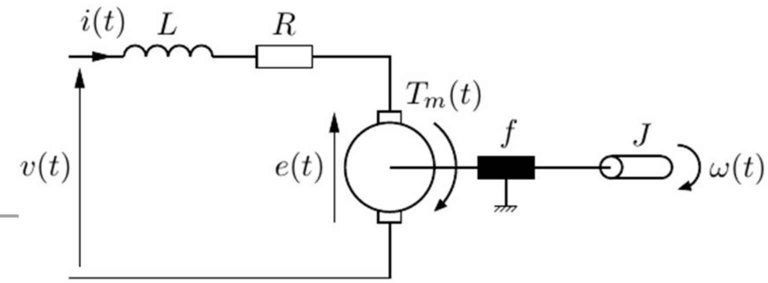
$$\mathcal{L}$$

$$\mathcal{L}$$

$$v = Lsi + Ri + k_m\omega$$

$$k_m i = Js\omega + \beta\omega$$

DC Motor Model



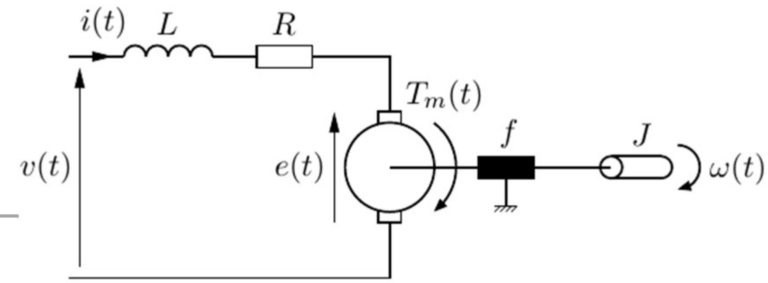
$$v = Lsi + Ri + k_m\omega$$

$$v = i(Ls + R) + k_m\omega$$

$$i = \frac{\omega}{k_m}(Js + \beta)$$

$$v = \frac{\omega}{k_m}(Js + \beta)(Ls + R) + k_m\omega$$

DC Motor Model

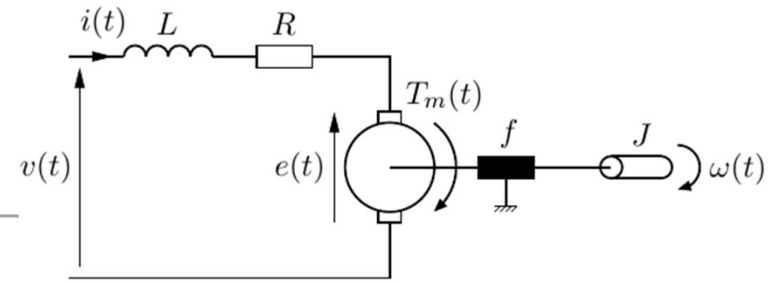


$$v = \frac{\omega}{k_m}(Js + \beta)(Ls + R) + k_m\omega$$

$$v = \frac{\omega}{k_m}\left((Js + \beta)(Ls + R) + k_m^2\right)$$

$$\frac{v}{\omega} = \frac{(Js + \beta)(Ls + R) + k_m^2}{k_m}$$

DC Motor Model

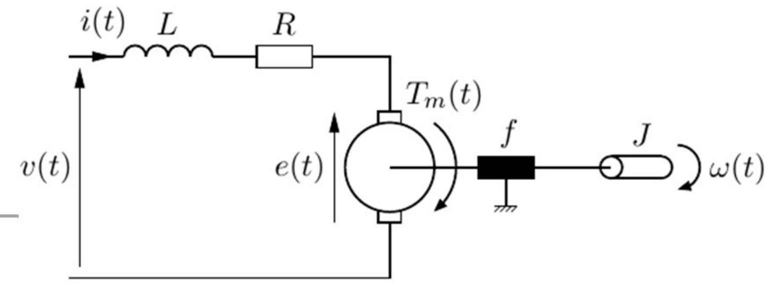


$$\frac{\omega}{v} = \frac{k_m}{(Js+\beta)(Ls+R)+k_m^2}$$

$$\frac{\omega}{v} = \frac{k_m}{JLs^2 + s(\beta L + JR) + RB + k_m^2}$$

$$\frac{\omega}{v} = \frac{k_m}{JL}\frac{1}{s^2 + \left(\frac{\beta}{J} + \frac{R}{L}\right)s + \frac{RB + k_m^2}{JL}}$$

DC Motor Model



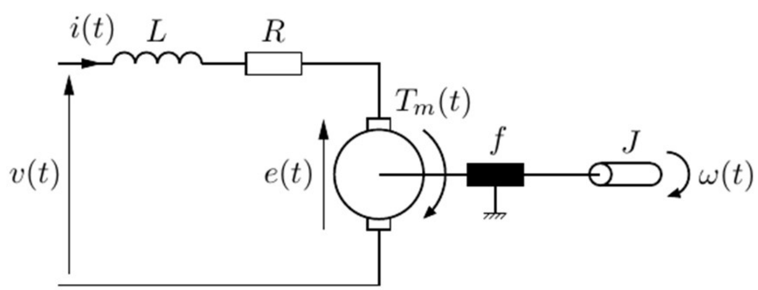
$$\frac{\omega}{v} = \frac{k_m}{JL} \frac{1}{s^2 + \left(\frac{\beta}{J} + \frac{R}{L}\right)s + \frac{RB + k_m^2}{JL}}$$

$$\frac{R}{L} \gg \frac{\beta}{J}$$

$$RB \ll k_m^2$$

DC Motor Model

$$\frac{\omega}{v} = \frac{k_m}{JL} \frac{1}{s^2 + \frac{R}{L}s + \frac{k_m^2}{JL}}$$



Hints

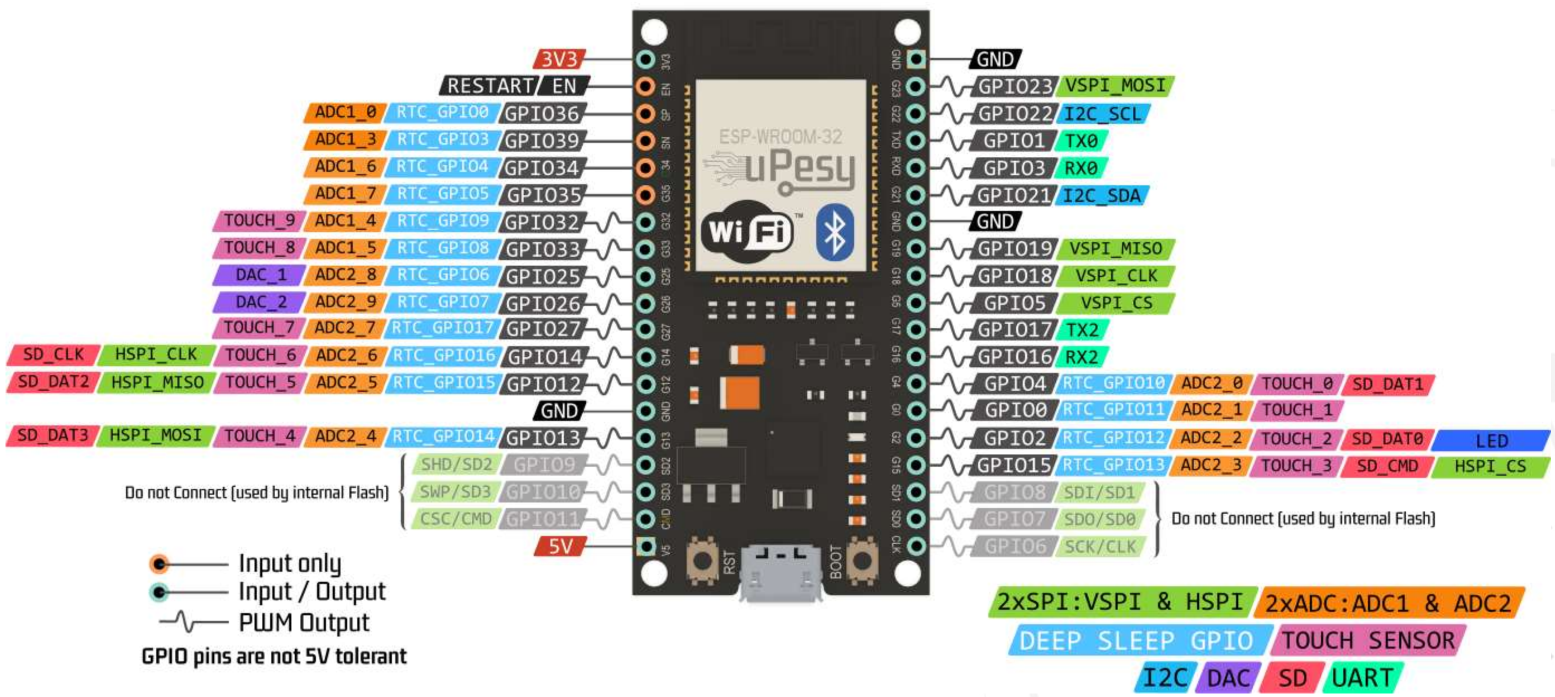
- [Configuring & Handling ESP32 GPIO Interrupts In Arduino IDE (lastminuteengineers.com)]
- Double check to perform operations with the same units.
- Check your data types
- Validate that there is no overflow in your PWM registry.
- Error

$$\varepsilon_n = v_{ref} - v_{medido}$$

- PID Controller

$$G_n = K_p \varepsilon_n + \frac{K_D}{T_s}(\varepsilon_n - \varepsilon_{n-1}) + K_I T_s(\varepsilon_n + \varepsilon_{n-1})$$

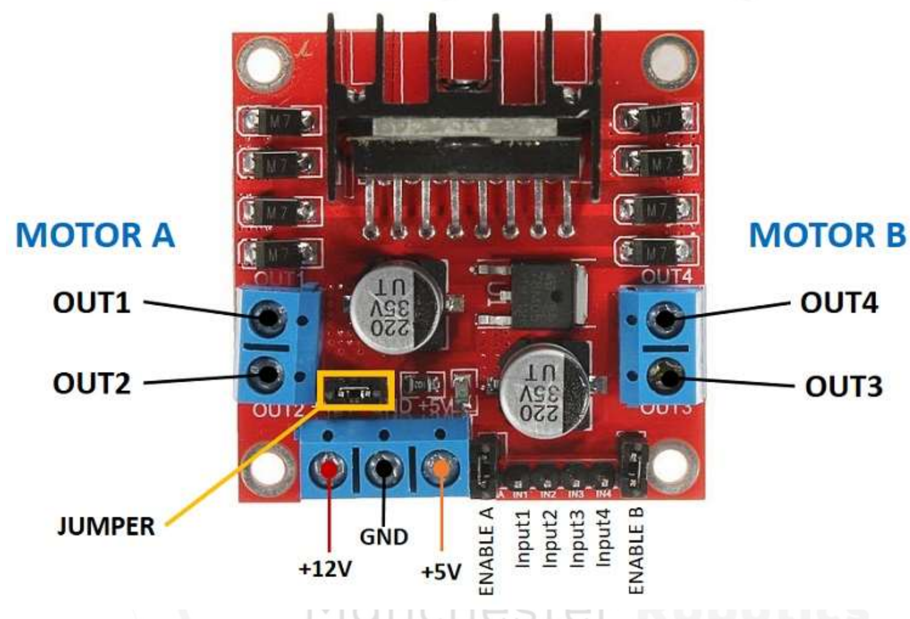
ESP32 Wroom DevKit Full Pinout



Micro-ROS Activity

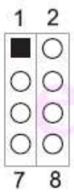
- Connect a H-bridge (L298) and a DC motor to the previous activity. The behavior of the motor should mimic the LED.
- Check the proper connections of the H-Bridge and the motor

- OUT1: DC motor A + terminal
- OUT2: DC motor A – terminal
- OUT3: DC motor B + terminal
- OUT4: DC motor B – termina
- IN1: Input 1 for Motor A
- IN2: Input 2 for Motor A
- IN3: Input 1 for Motor B
- IN4: Input 2 for Motor B
- EN1: Enable pin for Motor A (PWM)
- EN2: Enable pin for Motor B (PWM)



Micro-ROS Activity - Extra

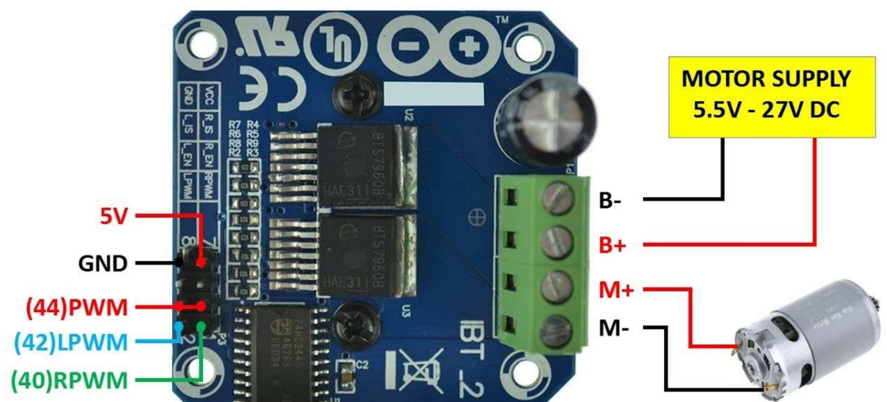
Input port



1、RPWM : Forward level or PWM signal input, active high
2、LPWM :Inversion level or PWM signal input, active high
3、R_EN :Forward drive enable input , high enable , low close
4、L_EN :Reverse drive enable input , high enable , low close
5、R_IS : Forward drive –side current alarm output
6、L_IS : Reverse drive –side current alarm output
7、VCC : +5 V power input,connected to the microcontroller 5V power supply
8、GND : Signal common ground terminal

Usage one:
VCC pick MCU 5V power supply, GND connected microcontroller GND
R_EN and L_EN shorted and connected to 5V level, the drive to work.
L_PWM, input PWM signal or high motor forward
R_PWM, input PWM signal or high motor reversal

Usage two:
VCC pick MCU 5V power supply , GND connected microcontroller GND
R_EN and L_EN short circuit and PWM signal input connected to high–speed
L_PWM, pin input 5V level motor is transferred
R_PWM, pin input 5V level motor reversal



PWM	1-255	1-255	1-255	1-255
RPWM	HIGH	LOW	LOW	HIGH
LPWM	LOW	HIGH	LOW	HIGH
	CW	CCW	STOP	BURN

Motor with encoder

Thank you

{Learn, Create, Innovate};

Manchester **Robotics**

T&C

Terms and conditions

{Learn, Create, Innovate};

Terms and conditions