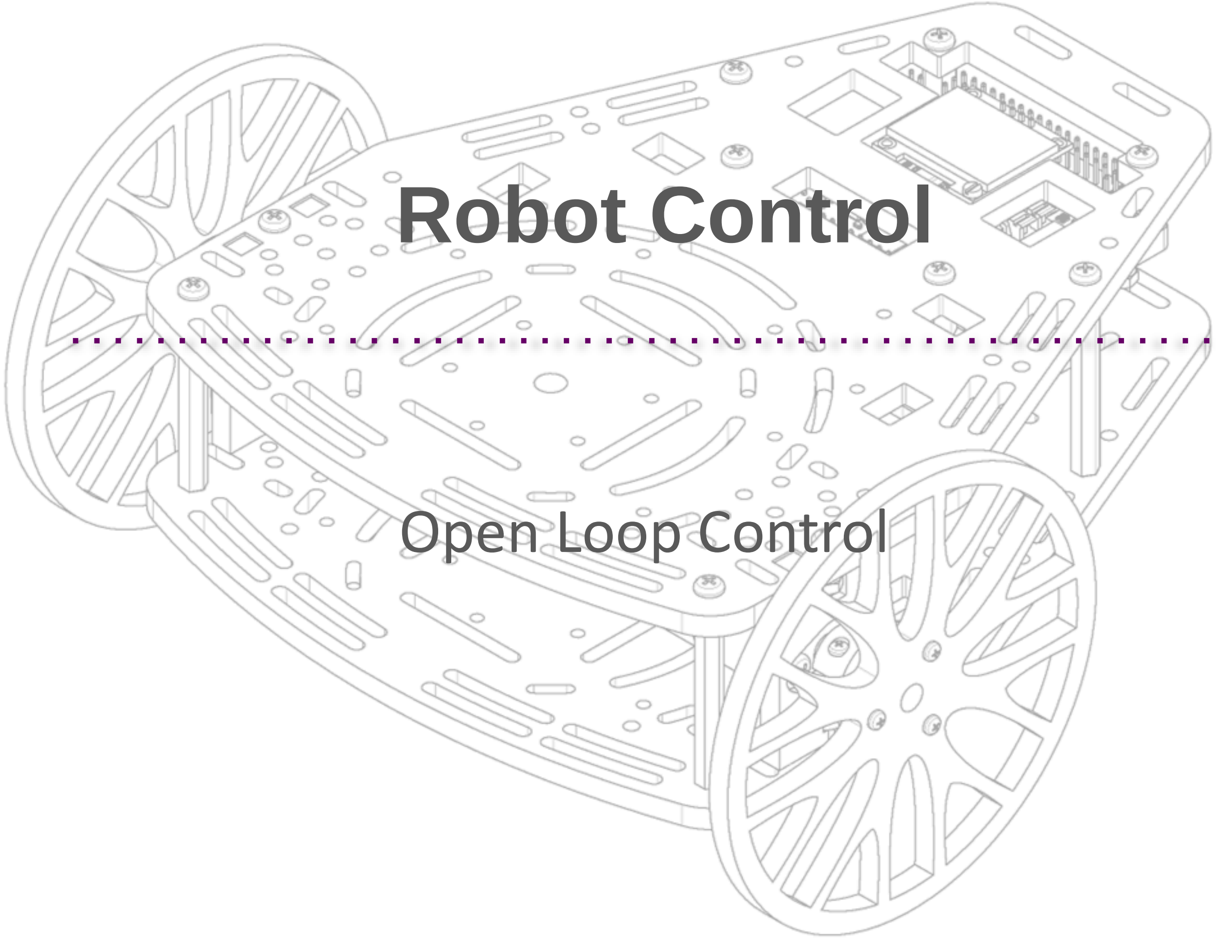


Robot Control

Open Loop Control

Schedule

- Day 4 : Robot Control : Introduction

I. Basic Control Theory

- Open loop systems
- Time based control

II. Activities (Rest of the class)

- *Activity 1: Create a node that sends commands to the robot gazebo simulation and move in a straight line for a period of time.*
- *Activity 2: Using the previous created node, expand it to create a square path.*

Autonomy In Robotics

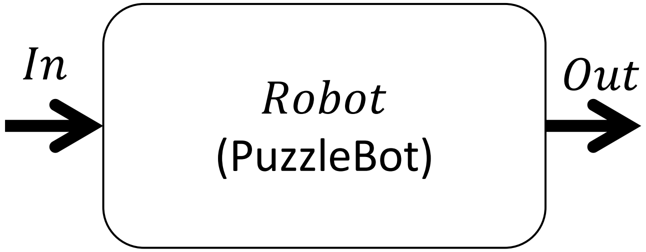
- As stated before, a robot is a dynamic system that is guided by a computer program to perform some specific tasks.

- The level of autonomy of a robot depends on the amount of information the robot requires from a human to perform its tasks.

Levels of Autonomy	
Human Operated : Human operator makes all the decisions.	Remote Control
Human Delegated : Robot can perform several functions independent of human control when delegated.	Semi-Autonomous
Human Supervised : The system can perform wide variety of activities when given top level permission or direction by a human.	
Fully Autonomous : The system receives goals form humans, translating them into tasks to be perfomed without human interaction.	Autonomous System

Autonomy In Robotics

- In this session we will address remote control and Semi-Autonomy.
- In other words, we will look at the robot from the perspective of an “open loop” control system.
- The robot will be seen as a black box system to be controlled with inputs and outputs.



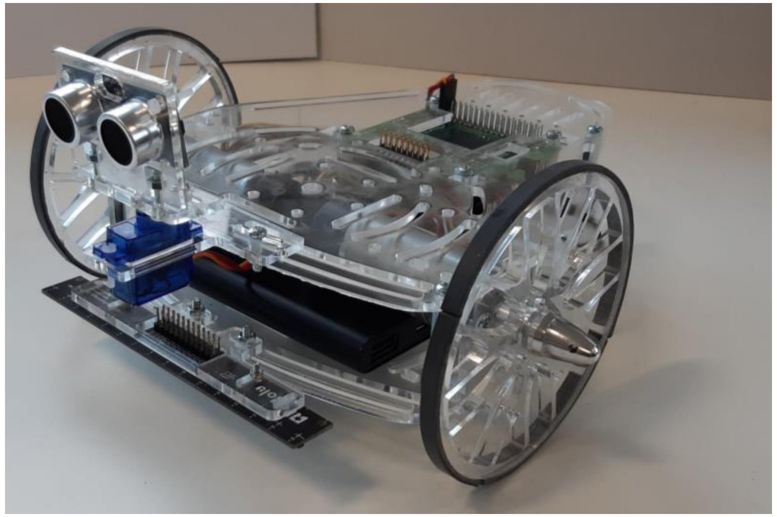
In

Robot

(PuzzleBot)

Out

Black Box Model



Differential-drive

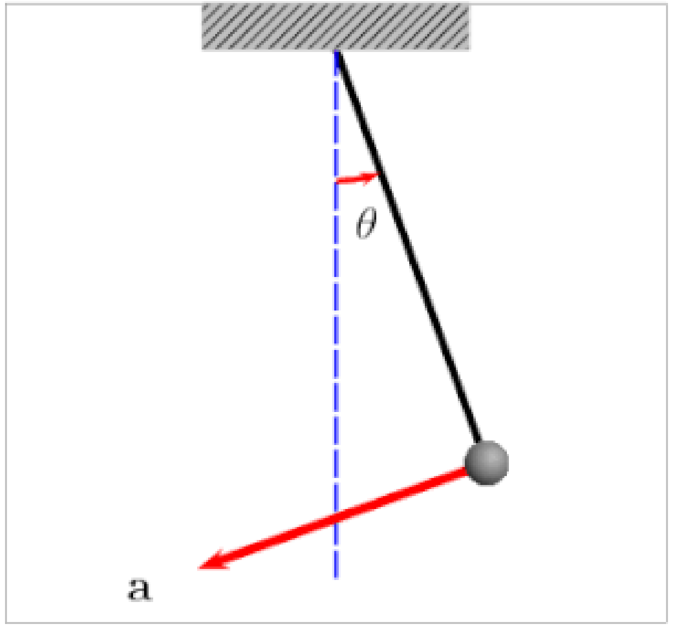
PuzzleBot ©.

System Fundamentals

- In control engineering, a system is an entity that consists of inputs, outputs and interconnected components, built with a desired purpose.

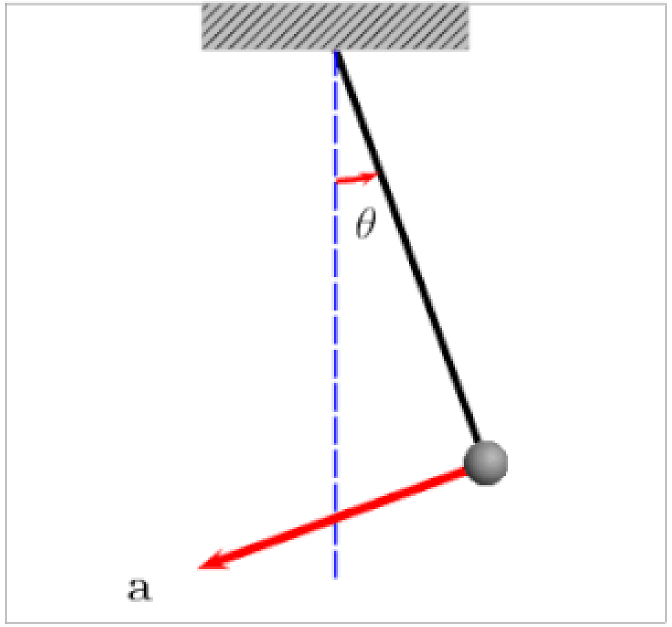
- The components that conform the system, are defined by their own particular set of variables called *states*. These *states* can completely define a system behavior at a given moment in time.
  - *For the case, of the differential drive robot. The states of such robot would be its position, velocity and acceleration.*

- Systems are inherently dynamic (present dynamical behavior), in other words, the change of a state of a system cannot be instantaneous due to energy and power limitations therefore requiring time.



Dynamical System Example (Pendulum)

Pendulum State Space

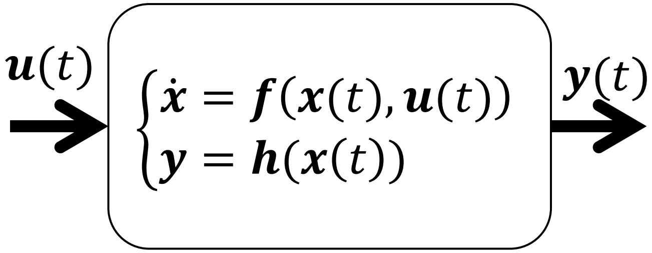


System Fundamentals

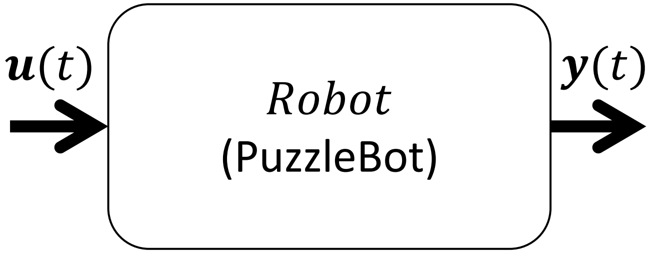
- If these state variations are measurable and analyzed with time and modeled mathematically, the system is said to be a “Dynamical System”.
- In general, physical Dynamical Systems, such as the differential drive robot, can be described / modelled using ODE of the form

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), & \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \\ \boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}(t)) \end{cases}$$

where $\boldsymbol{x}(t)$ is the state vector of the system, $\boldsymbol{x}_0$ are the initial conditions, $\boldsymbol{u}(t)$ is the input vector to the system at a given time, $\boldsymbol{f}$ is a well-behaved function that provides the information about the behavior / evolution of the system at a specific time and $\boldsymbol{h}$ is a function that represents the relationship between the states of the system and the measured variables in the output.


$\boldsymbol{u}(t)$
$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}(t)) \end{cases}$
$\boldsymbol{y}(t)$

Dynamical Model



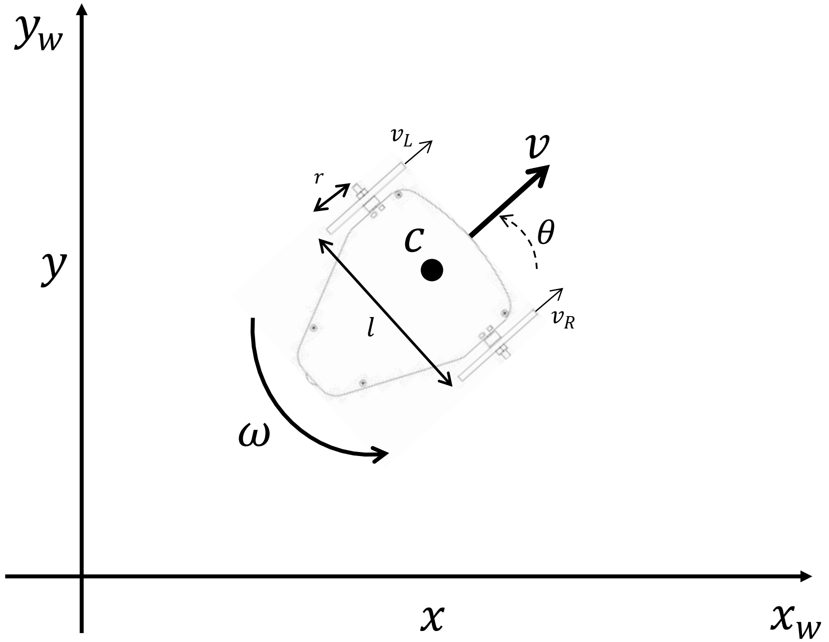
Black Box Model

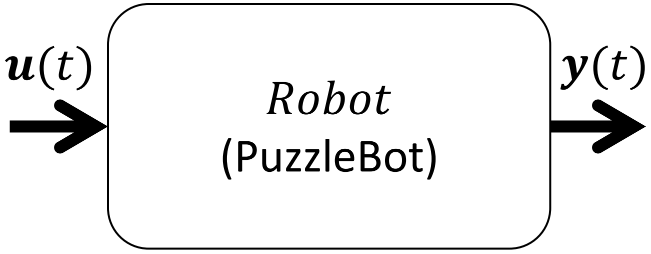
System Fundamentals

- For the case of the real system, “Puzzlebot” (differential drive robot) the inputs to the system $\boldsymbol{u}(t)$ would be *the linear and angular velocity of the robot** i.e., $\boldsymbol{u}(t) = [v, \omega]^T$, in $\frac{\mathrm{m}}{\mathrm{s}}$ and $\frac{\mathrm{rad}}{\mathrm{s}}$ respectively .
  - **The inputs to the robot can also be the wheel speed i.e., $\boldsymbol{u}(t) = [\omega_r, \omega_l]^T$ since it just a linear transformation that depends on the radius of the wheel and the wheelbase of the robot i.e.,*

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{l} & -\frac{r}{l} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

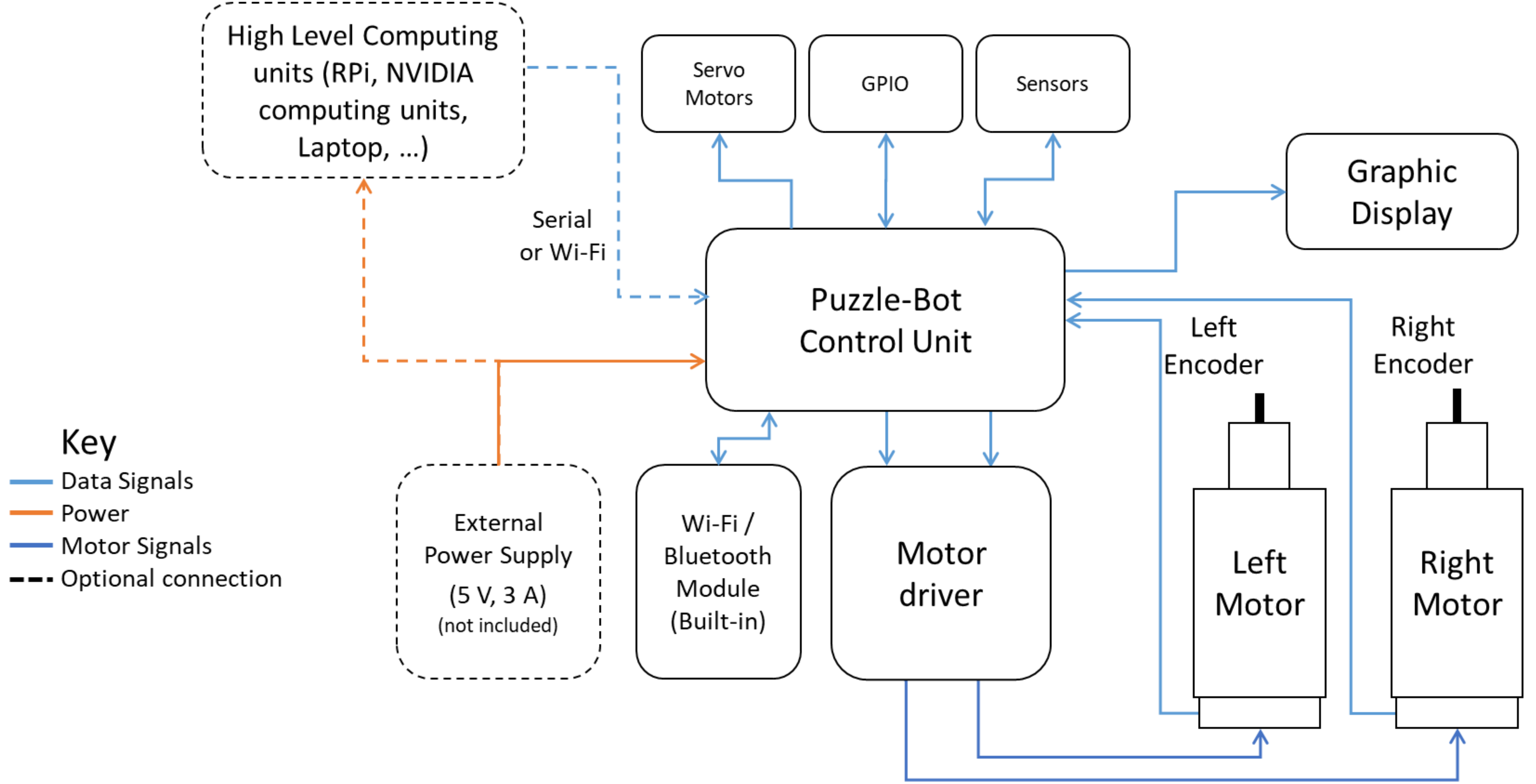
- The output of the system will be the right and left measured wheel velocities $\boldsymbol{y}(t) = [\omega_r, \omega_l]^T$ given in $\frac{\mathrm{rad}}{\mathrm{s}}$.
  - **The output of the robot can also be the linear and angular velocities i.e., $\boldsymbol{y}(t) = [v, \omega]^T$ by inverting the transformation shown previously.*
- Can we model the dynamical behaviour of the differential drive? Yes, we can but it can be a difficult task.





Black Box Model

Differential Drive Sensors and Actuators



Dynamic Model of a DDR

$$\bar{M}(q)\,\ddot{q} + \bar{V}(q,\dot{q})\,\dot{q} \; = \bar{B}(q\,)\tau$$

Where:

$$\bar{M}(q) = \begin{bmatrix} I_w + \frac{R^2}{4L^2}(mL^2 + I) & \frac{R^2}{4L^2}(mL^2 - I) \\ \frac{R^2}{4L^2}(mL^2 - I) & I_w + \frac{R^2}{4L^2}(mL^2 + I) \end{bmatrix},\; \bar{V}(q,\dot{q}) = \begin{bmatrix} 0 & \frac{R^2}{2L} m_c d\,\omega \\ -\frac{R^2}{2L} m_c d\,\omega & 0 \end{bmatrix},\;\; \bar{B}(q\,) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\; q = \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix}$$

I_w= moment of inertia of the wheels

I = total equivalent inertia $I = I_c + m_c d^2 + 2m_w L^2 + 2I_m$, where I_c=moment of inertia of the robot around the vertical axis, m_c= mass of the robot without the driving wheels and motors, I_m =moment of inertia of each driving wheel with a motor about the wheel diameter, d =distance between centre of mass and rotation, m_w = mass of the wheel with motor.

R= radius of the wheels

L= Distance between wheels

m= mass of the robot

ω = angular speed of the robot

Dynamic Model of a DDR

$$\begin{cases} V_a = R_a i_a + L_a \dfrac{di_a}{d_t} + e_a \\ e_a = K_b \omega_m \\ \tau_m = K_t i_a \\ \tau = N\tau_m \end{cases}$$

$where$:

V_a =armature voltage

i_a =armature current

R_a, L_a =resistance and inductance of the armature winding

e_a =back emf

K_t, K_b =torque and back emf constants

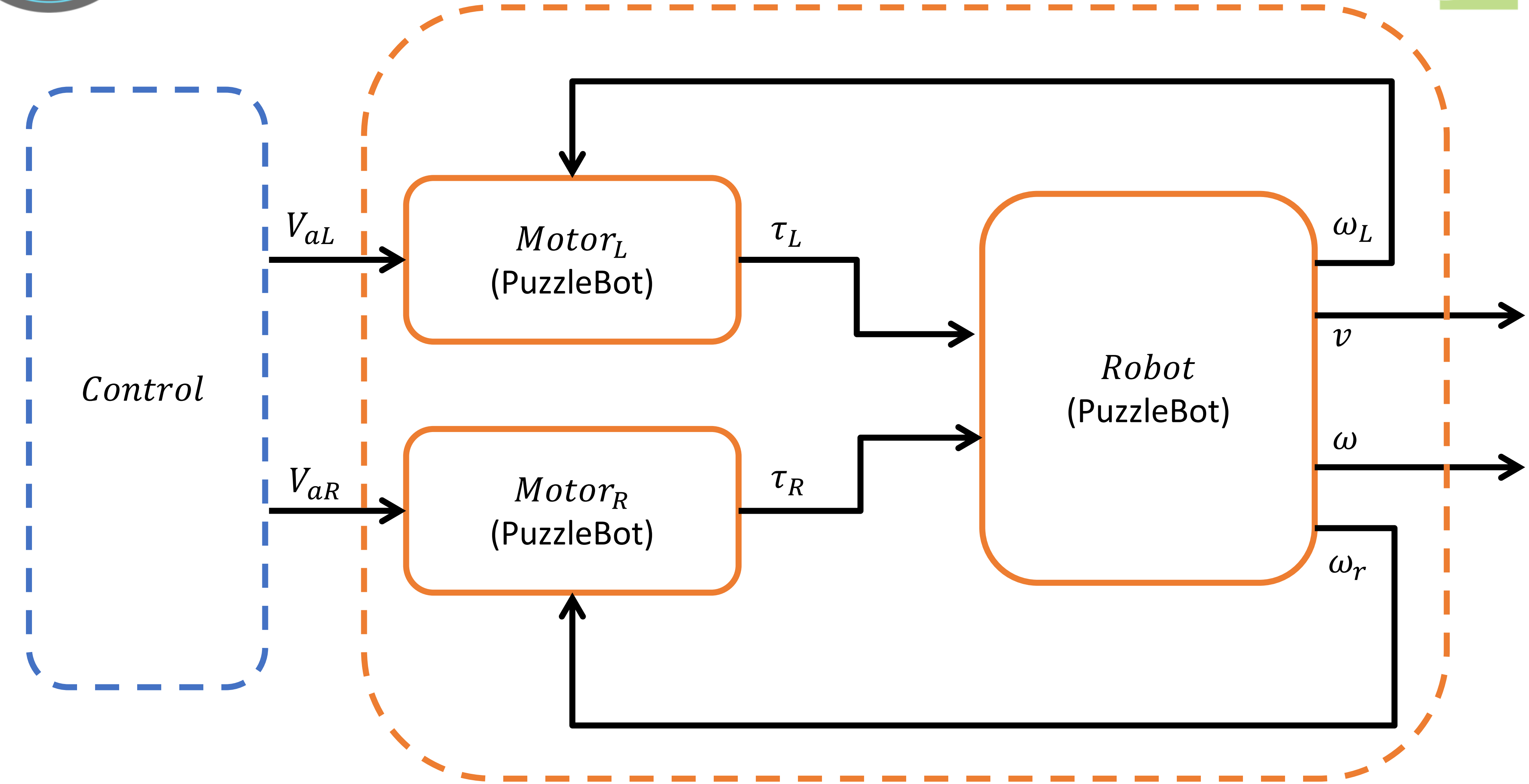
τ_m =motor torque

ω_m =motor angular speed (ω_r, ω_l for each motor respectively)

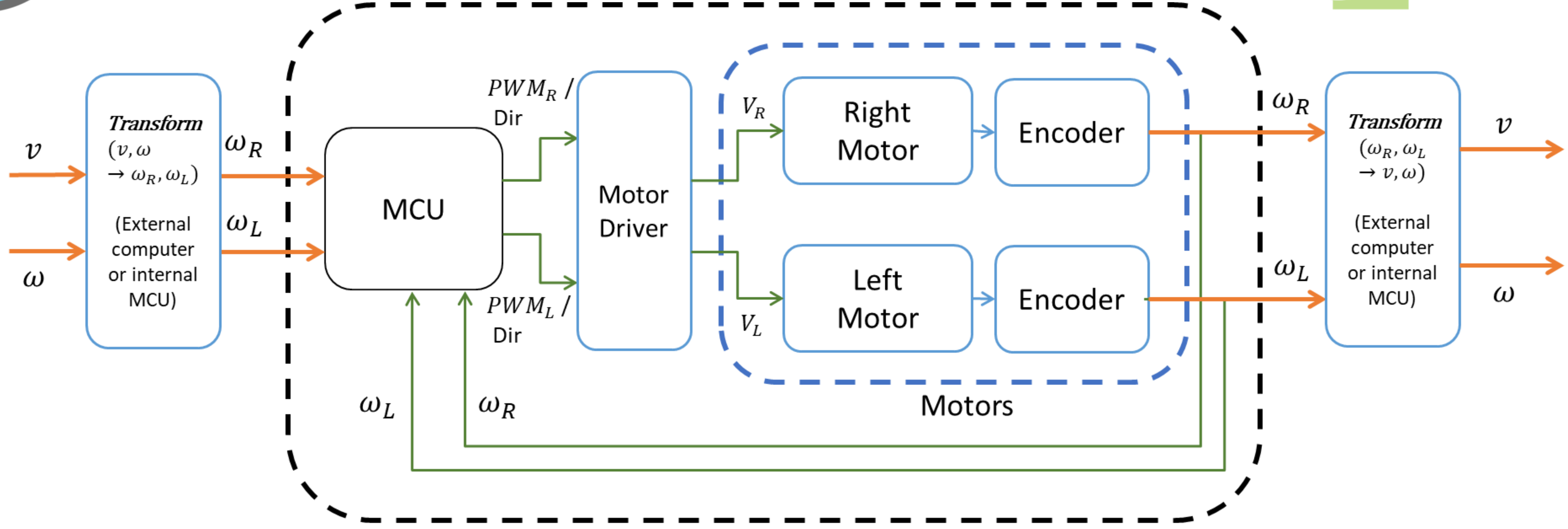
N =gear ratio

τ =output torque

Dynamic Model of a DDR

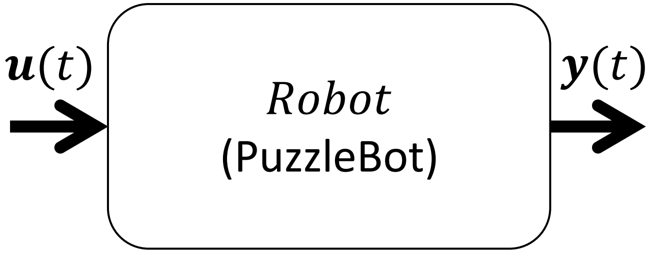


Differential Drive Sensors and Actuators



v
ω
Transform $(v, \omega \rightarrow \omega_R, \omega_L)$
(External computer or internal MCU)
ω_R
ω_L
MCU
PWM_R / Dir
PWM_L / Dir
Motor Driver
V_R
V_L
Right Motor
Left Motor
Encoder
Encoder
Motors
ω_L
ω_R
ω_R
ω_L
Transform $(\omega_R, \omega_L \rightarrow v, \omega)$
(External computer or internal MCU)
v
ω

Real Robot Diagram*



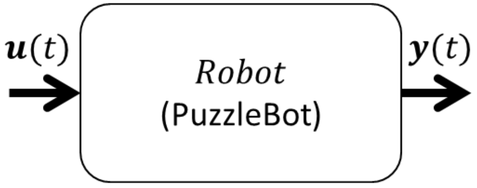
Black Box Model

System Fundamentals

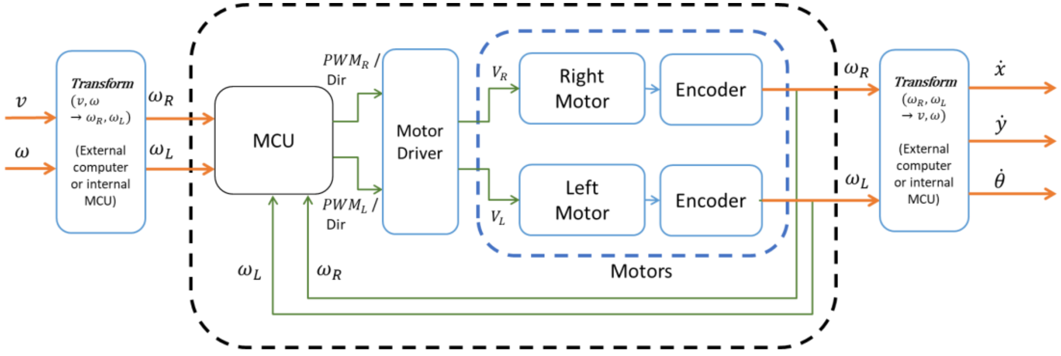
- In Engineering we there are different approaches to deal with complex systems.
- One of the most common ones is the black, white and gray box modelling / testing techniques.

Black Box	Data based
Gray Box	Data Based / Partly Physical Knowledge
White Box	Deterministic Equations, Detailed Sub-models and Full Physical Knowledge

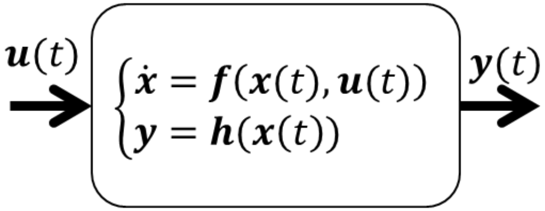
- It is possible to estimate the gray box model for the differential drive but for this activity, as stated before, we will use the black box approach.



Black Box Model



Real Robot Diagram*



$u(t)$
$\begin{cases}\dot{x} = f(x(t), u(t)) \\ y = h(x(t))\end{cases}$
$y(t)$

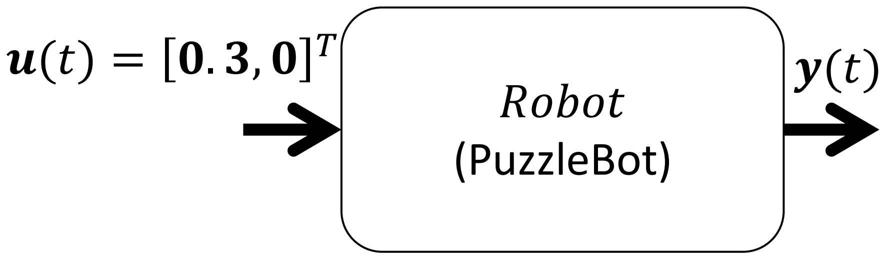
Dynamical Model

Open Loop Control

- Let the system (in this case the robot) to be represented as a black box and let only constant linear velocity of $v = 0.3\frac{m}{s}$ to be the input to the system (step input).

- It can be observed that the output of the system matches the input reference at steady state. Therefore, the gain of the system is said to be 1. **This gain is only for the robot system for any other system the system's gain must be estimated.*

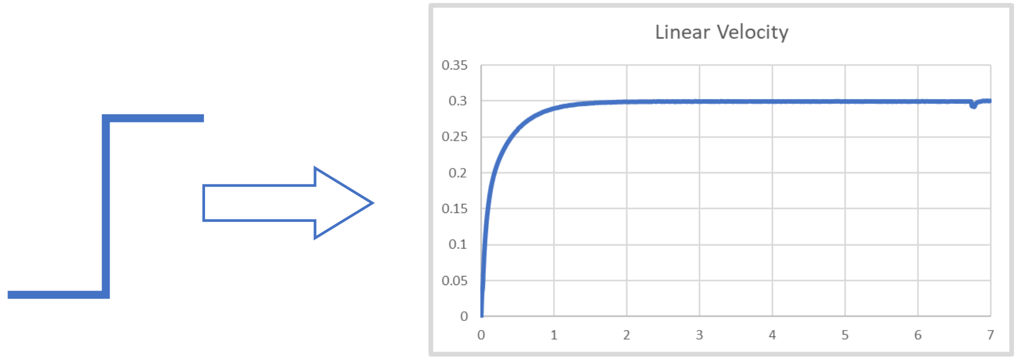
- A more in-depth analysis of this situation reveals that this is due to the inner controllers of the robot, allows the robot to follow the reference after some transient. The same thing happens with the angular speed.


$\boldsymbol{u}(t) = [\mathbf{0.3, 0}]^T$

Robot
(PuzzleBot)

$\boldsymbol{y}(t)$

Black Box Model



Input

Output

Open Loop Control

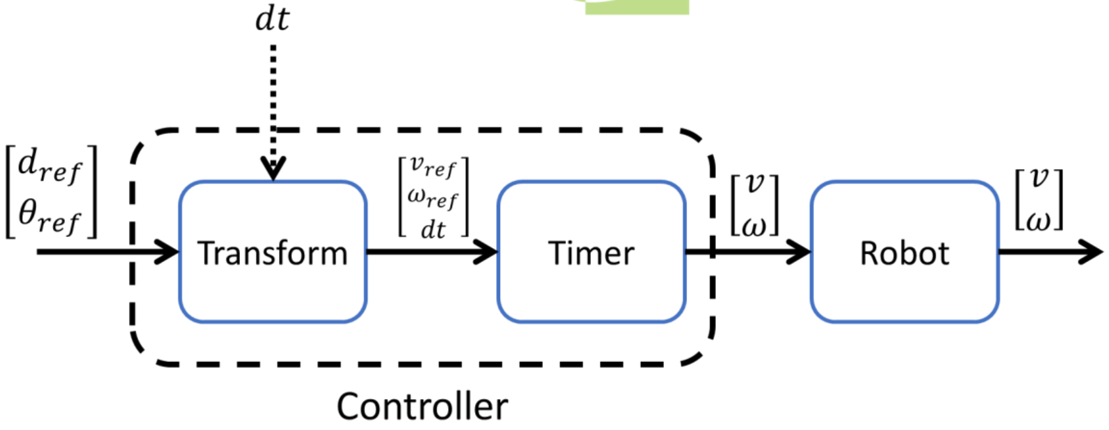
- Knowing that the system’s output will follow the input after a transient, a question arises. Can we use this information to move the robot a certain distance?
- Since we know that the output follows the input, it is possible to approximate the distance and angle traveled by using the following formula.

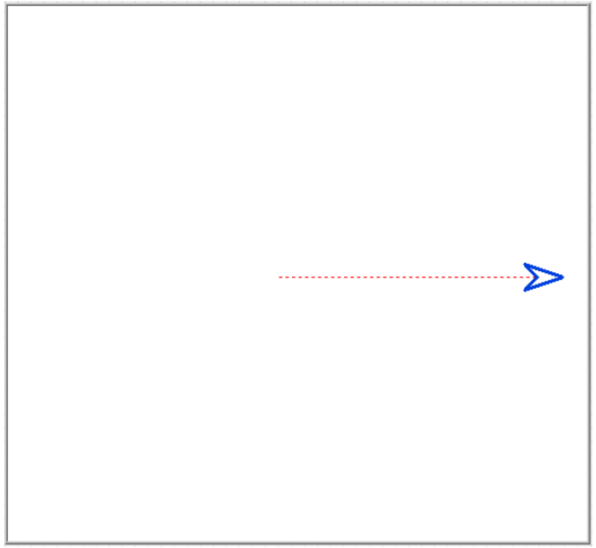
$$d = v \cdot dt = r\left(\frac{\omega_r + \omega_l}{2}\right) \cdot dt$$

$$\theta = \omega \cdot dt = r\left(\frac{\omega_r - \omega_l}{l}\right) \cdot dt$$

where dt is the time since the input was applied.

- Therefore, it is possible to use a timer to control the distance and angle travelled by the robot.
- This is called open loop control.


dt
$\begin{bmatrix} d_{ref} \\ \theta_{ref} \end{bmatrix}$
Transform
$\begin{bmatrix} v_{ref} \\ \omega_{ref} \\ dt \end{bmatrix}$
Timer
$\begin{bmatrix} v \\ \omega \end{bmatrix}$
Robot
$\begin{bmatrix} v \\ \omega \end{bmatrix}$
Controller



Open Loop Control

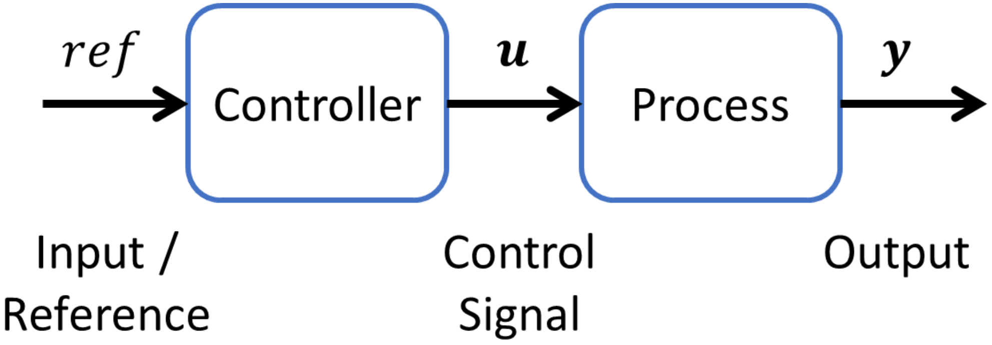
Open Loop Control

- Open Loop Control System is a system in which the control action is independent of the output of the system.

- In this type of control, the output is regulated by varying the input or reference.

- The output of the system is determined by the current state of the system and the inputs that are received from the controller.

- Some examples of this type of control systems are Windows, window blinds, washing machines, microwave ovens, hair drier, bread toaster, Door Lock System, some stepper motors, turning on/off lights, some remote-controlled applications, etc.



ref

Controller

u

Process

y

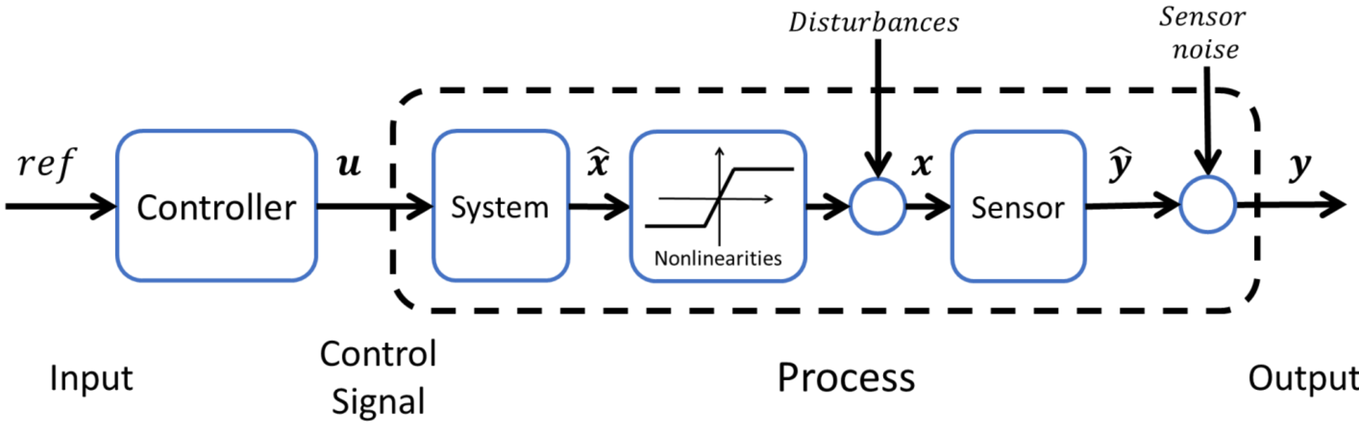
Input / Reference

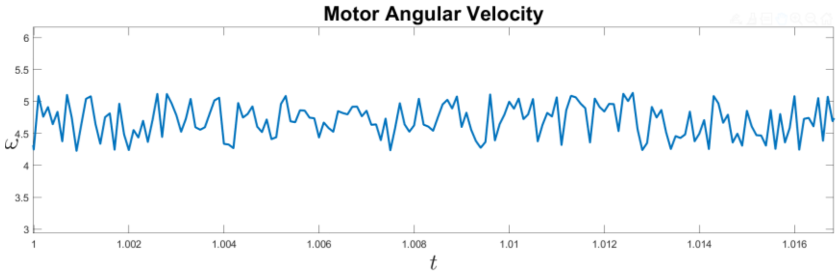
Control Signal

Output

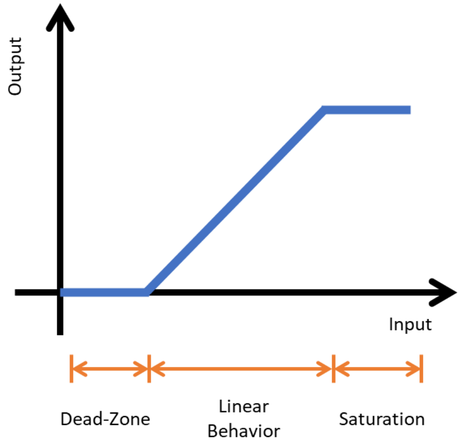
Open Loop Control

- In reality, every process present disturbances, nonlinearities and noise.
- Disturbances and noise come from the environment that surrounds the system.
  - On the differential drive robot, some disturbances can come from wheels slipping, obstacles on the environment, different types of terrain, etc.
  - On the other hand, the noise is present when reading the values of the motor encoders (Sensors).
- Nonlinearities are a intrinsic characteristic of a system in which the output does not linearly follow the input (output not proportional to the input).
  - For the case of the robot, the nonlinear behavior can be seen as a saturation and dead-zone in the motors and the robot itself.





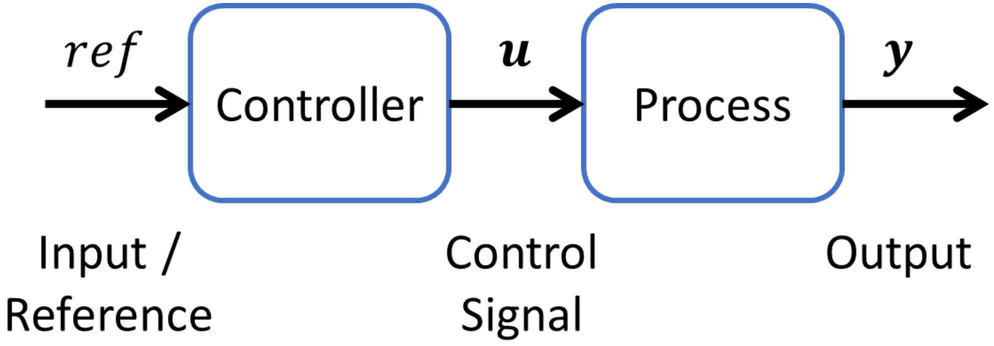
Noisy Signal (Angular velocity)



Nonlinear behavior

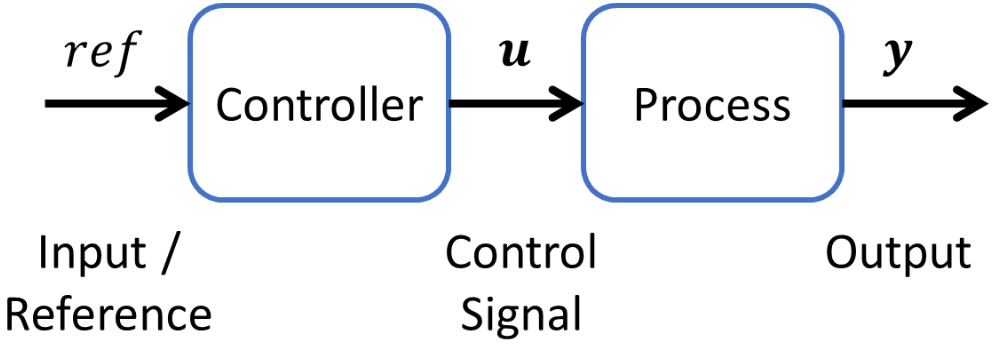
Open Loop Control

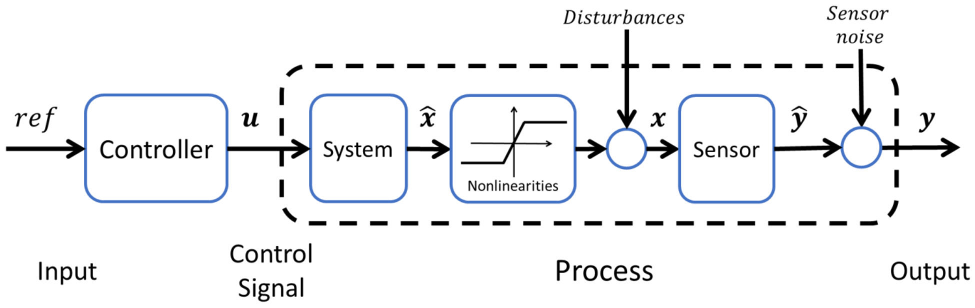
- Open Loop Advantages:
  - Simple to design and implement, provided that the user has some experience with the system.
  - The cost for design, implement and maintain is relatively low compared with other controllers.
  - Maintenance is considered simple, no high technical level required (for most of the controllers).
  - The behavior of the controller is quite stable, provided that the system is in a controlled environment, such that the process does not present big disturbances, or the process is not dangerous when unsupervised.



Open Loop Control

- Open Loop Disadvantages:
  - This type of control is not robust against disturbances (cannot correct the output in the presence of a disturbance).
  - An open-loop system has no self-regulation or control action over the output value.
  - Not reliable
  - The input depends on the experience of the user.
  - Each input determines a fixed operating point of the system.
  - Controller must be altered manually in case of an output disturbance or uncalibrated controller.
  - Requires re-calibration often.
  - Prone to errors in the output and control signal
  - Does not take into consideration changes on the process over time.
  - Also, there is no chance to correct the transition errors in open loop systems so there is more chance to occur errors.

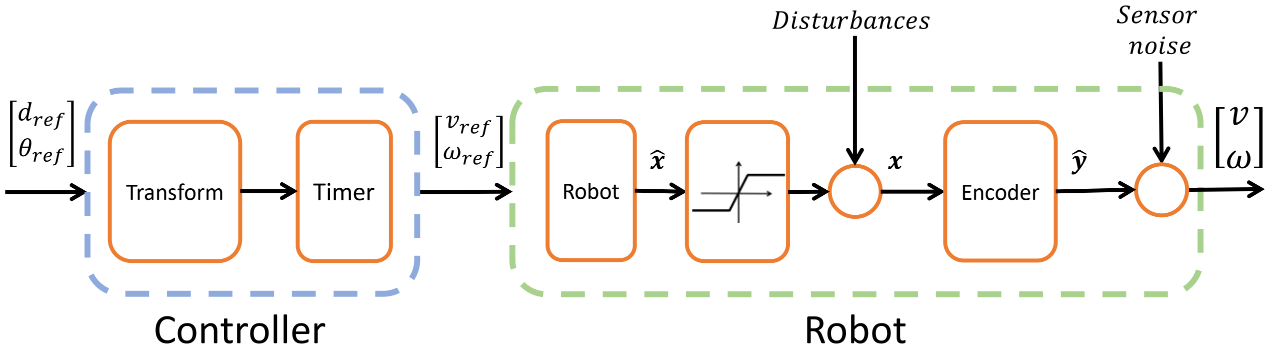




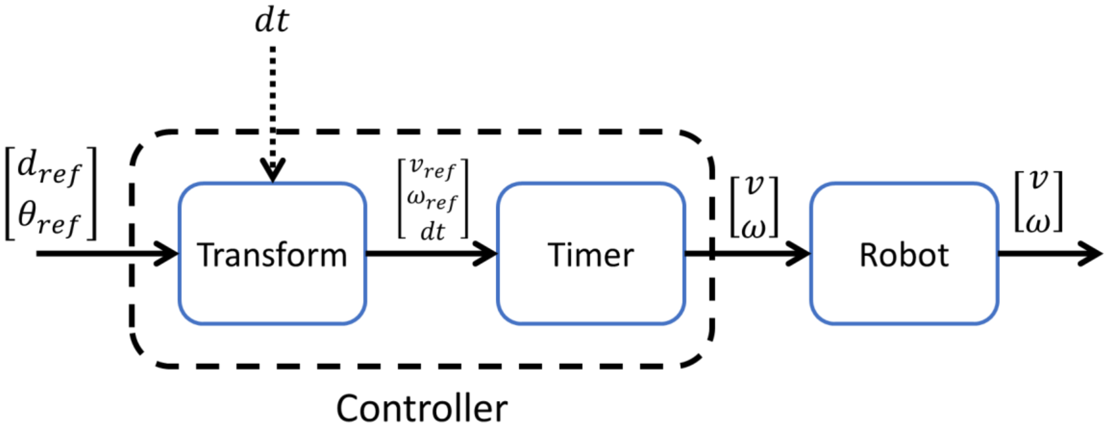
Open Loop Control: Mobile Robot

- As stated before, open loop control can be used to control de position of the robot.
- When designing this controller, the nonlinear behavior, perturbations and noise should be taken into consideration to make the controller work on the linear part of the robot and make it as robust as possible.
- In this case, the real mobile robot is a nonlinear system, due to the maximum and minimum velocities of the motors therefore it presents saturation and dead-zone.

  *Other nonlinear behavior are present, but they are not relevant in this case.
- The linear behavior is present within a region in which the actuators (motors) output are proportional to the input.


$\begin{bmatrix} d_{ref} \\ \theta_{ref} \end{bmatrix}$
Transform
Timer
$\begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix}$
Robot
$\hat{x}$
Disturbances
x
Encoder
$\hat{y}$
Sensor noise
$\begin{bmatrix} v \\ \omega \end{bmatrix}$
Controller
Robot

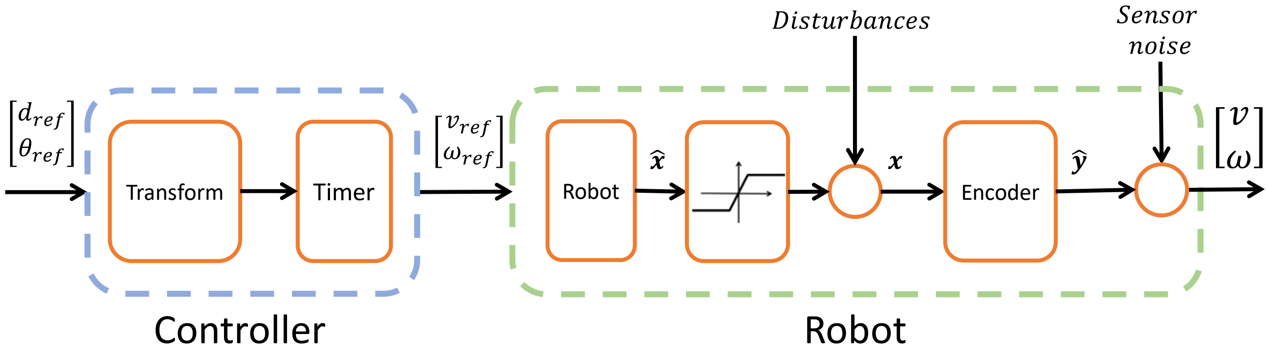
Real Robot


dt
$\begin{bmatrix} d_{ref} \\ \theta_{ref} \end{bmatrix}$
Transform
$\begin{bmatrix} v_{ref} \\ \omega_{ref} \\ dt \end{bmatrix}$
Timer
$\begin{bmatrix} v \\ \omega \end{bmatrix}$
Robot
$\begin{bmatrix} v \\ \omega \end{bmatrix}$
Controller

Simulated robot

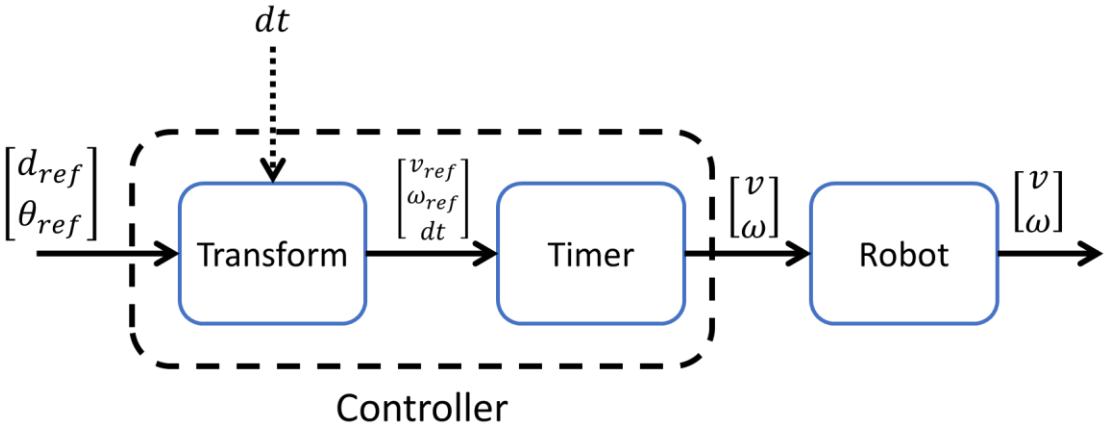
Open Loop Control: Mobile Robot

- Noise and perturbations can also be found in the robot.
- As said before, noise can be found in the encoder reading, the noise can be due to electromagnetic noise or radiation, mechanical unbalance of encoders, etc.
- Disturbances are due to the mechanical unbalance of the wheels, wheel slippage with the floor, defects of productions, changes in the environment, etc.
- Depending on the simulator, the dynamical behavior of the robot can be linear or nonlinear and can consider the noise and disturbances .



$\begin{bmatrix} d_{ref} \\ \theta_{ref} \end{bmatrix}$
Transform
Timer
$\begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix}$
Robot
$\hat{x}$
Disturbances
x
Encoder
$\hat{y}$
Sensor noise
$\begin{bmatrix} v \\ \omega \end{bmatrix}$
Controller
Robot

Real Robot



dt
$\begin{bmatrix} d_{ref} \\ \theta_{ref} \end{bmatrix}$
Transform
$\begin{bmatrix} v_{ref} \\ \omega_{ref} \\ dt \end{bmatrix}$
Timer
$\begin{bmatrix} v \\ \omega \end{bmatrix}$
Robot
$\begin{bmatrix} v \\ \omega \end{bmatrix}$
Controller

Simulated robot

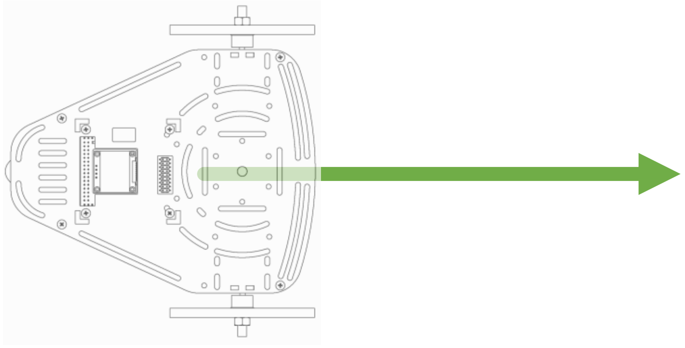
Now is your turn…

- *The following 2 activities will help you build an open loop controller for the Puzzlebot Gazebo simulation.*
  - *Activity 1: Create a node that sends commands to the robot gazebo simulation and move in a straight line for a period of time.*
  - *Activity 2: Using the previous created node, expand it to create a square path.*

Activity 1

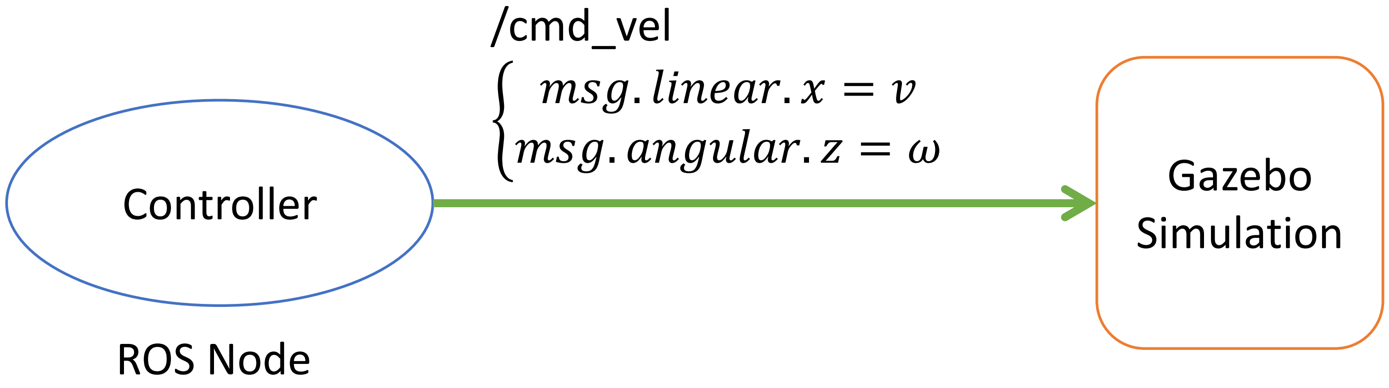
Moving a PuzzleBot

Straight line



Drive the robot 2 meters in a straight line.

- Objectives:
  - Create an open loop control as a ROS node that sends commands to the robot gazebo simulation and move in a straight line for 2 meters.



/cmd_vel

$$\begin{cases} msg.linear.x = v \\ msg.angular.z = \omega \end{cases}$$

Activity

Some tips and tricks

- One publisher `cmd_vel`
- `cmd_vel` – `from geometry_msgs.msg import Twist` – 3 linear and 3 angular velocities.
  - `msg.linear.x`, `msg.linear.y`, `msg.linear.z`
  - `msg.angular.x`, `msg.angular.y`, `msg.angular.z`
- Use the equations below to compute the distance moved and the angle turned by the robot

$$d = v \cdot dt = r\left(\frac{\omega_r + \omega_l}{2}\right) \cdot dt$$

$$\theta = \omega \cdot dt = r\left(\frac{\omega_r - \omega_l}{l}\right) \cdot dt$$

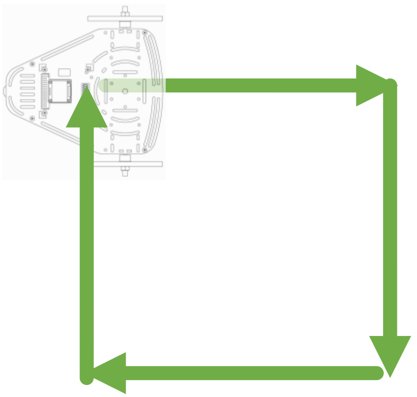
where r is the radius of the wheels ($0.05\ m$) and l is the distance between the wheels ($0.18\ m$)

- Use `rospy.get_time()` to measure the time dt between each loop
- If the robot is not moving, check your topics with `rostopic echo` and `rostopic pub`
- Ensure your python file is executable: `sudo chmod +x <path_to_file>.py`
- Ensure you source your file: `source devel/setup.bash`

Activity

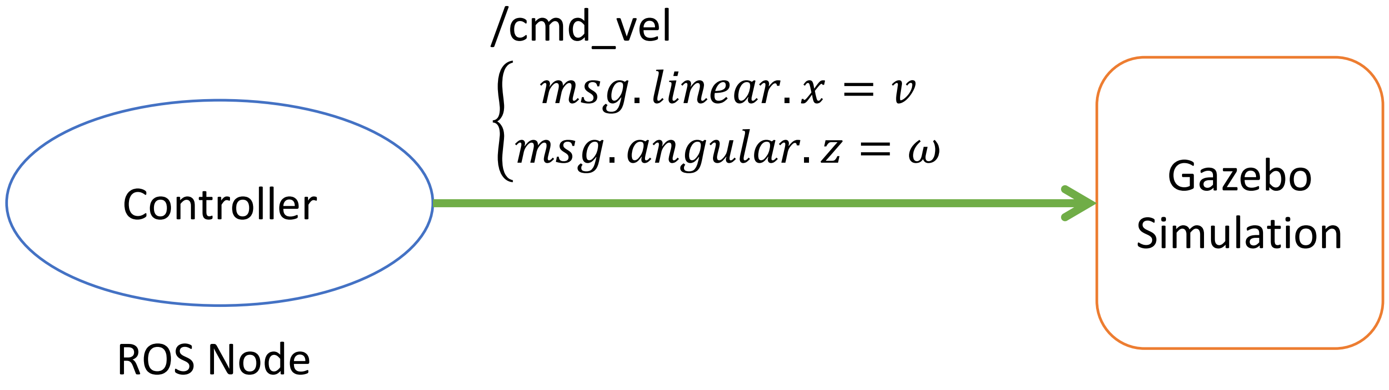
Moving a PuzzleBot

Square



Drive the robot making a square with a side length of 1m.

- Objectives:
  - Expand your previously created node or create a new node to make your robot drive in a square path of a side length 2 m.



/cmd_vel

$$\begin{cases} msg.linear.x = v \\ msg.angular.z = \omega \end{cases}$$