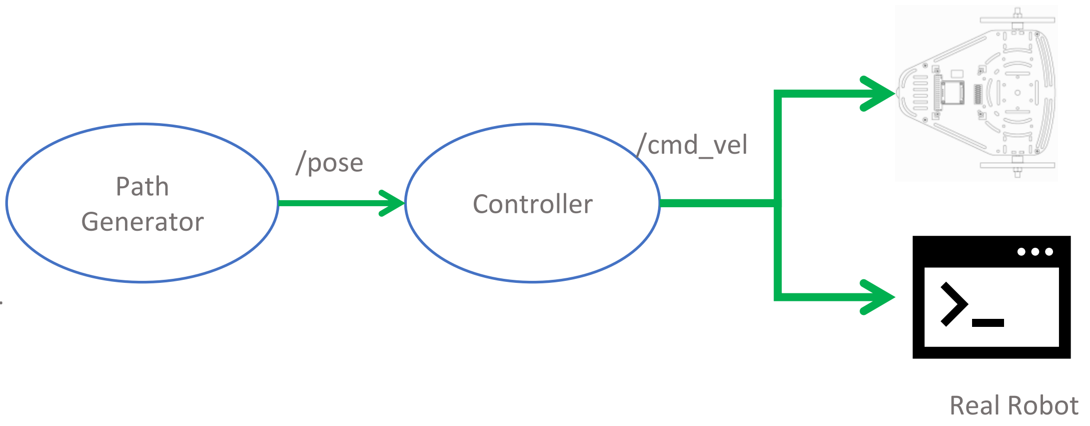
{Learn, Create, Innovate};

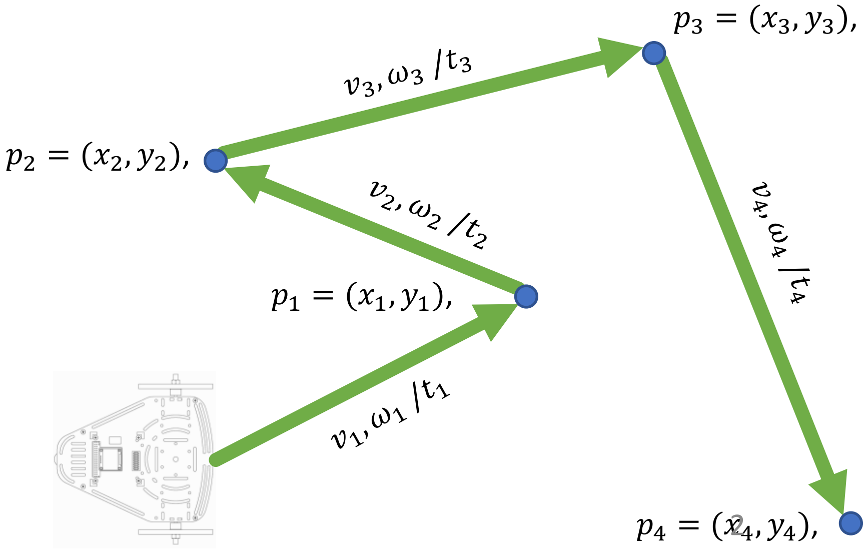
Closed Loop Control

Mobile Robot Control

Mini challenge 1

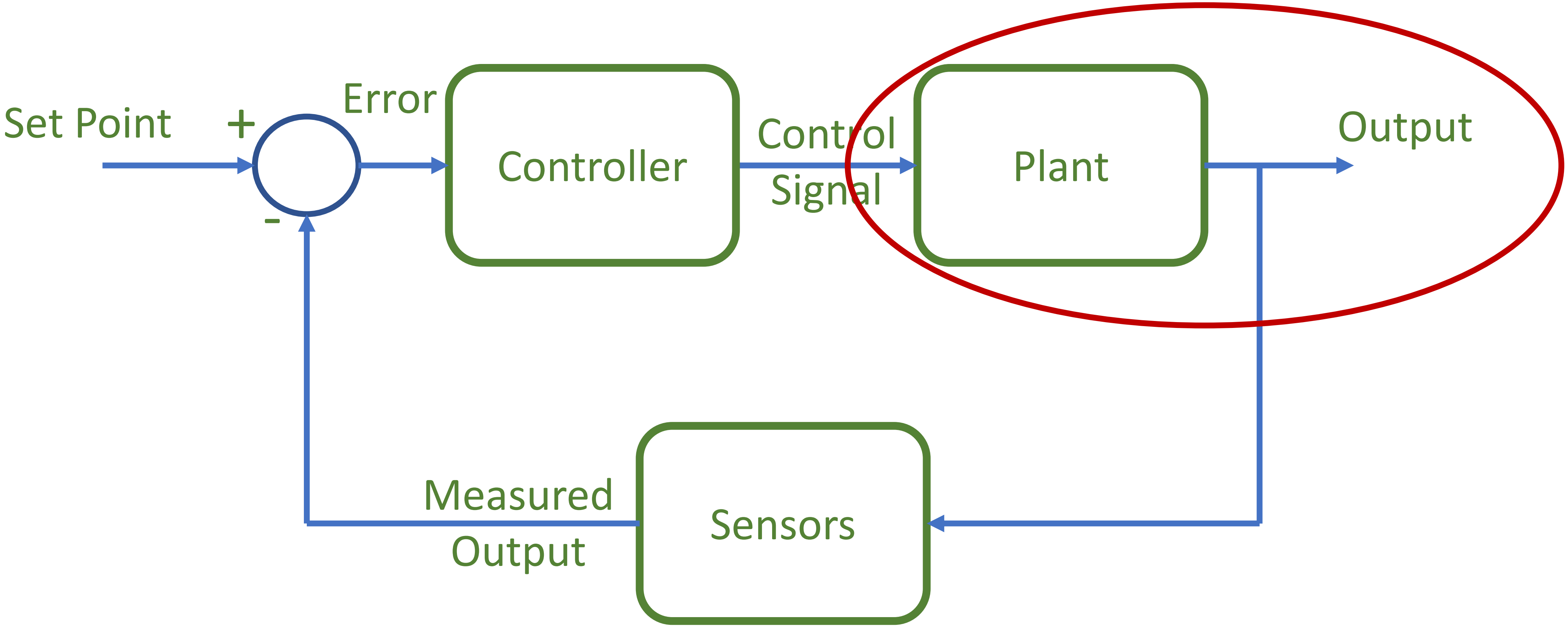
- Create a node that generates different paths according to a user.
- The path must be defined in the parameter files by the user. (3 or 4 are enough)
- The path must be defined by different points $p_k = (x_k, y_k)$, velocities of the robot, or a time $(v_1, \omega_1)/t$ (depend on the user).
    - For each point, the node must estimate the linear and rotational speeds in case a time is given by the user or estimate the time in case the velocities are provided.
- The node must let the user know if the point is reachable according to the dynamical behaviour of the mobile robot and the parameters that were input by the user $(v_1, \omega_1)/t$.
- The student must define what is robustness for this case, and the controller must take that into consideration.



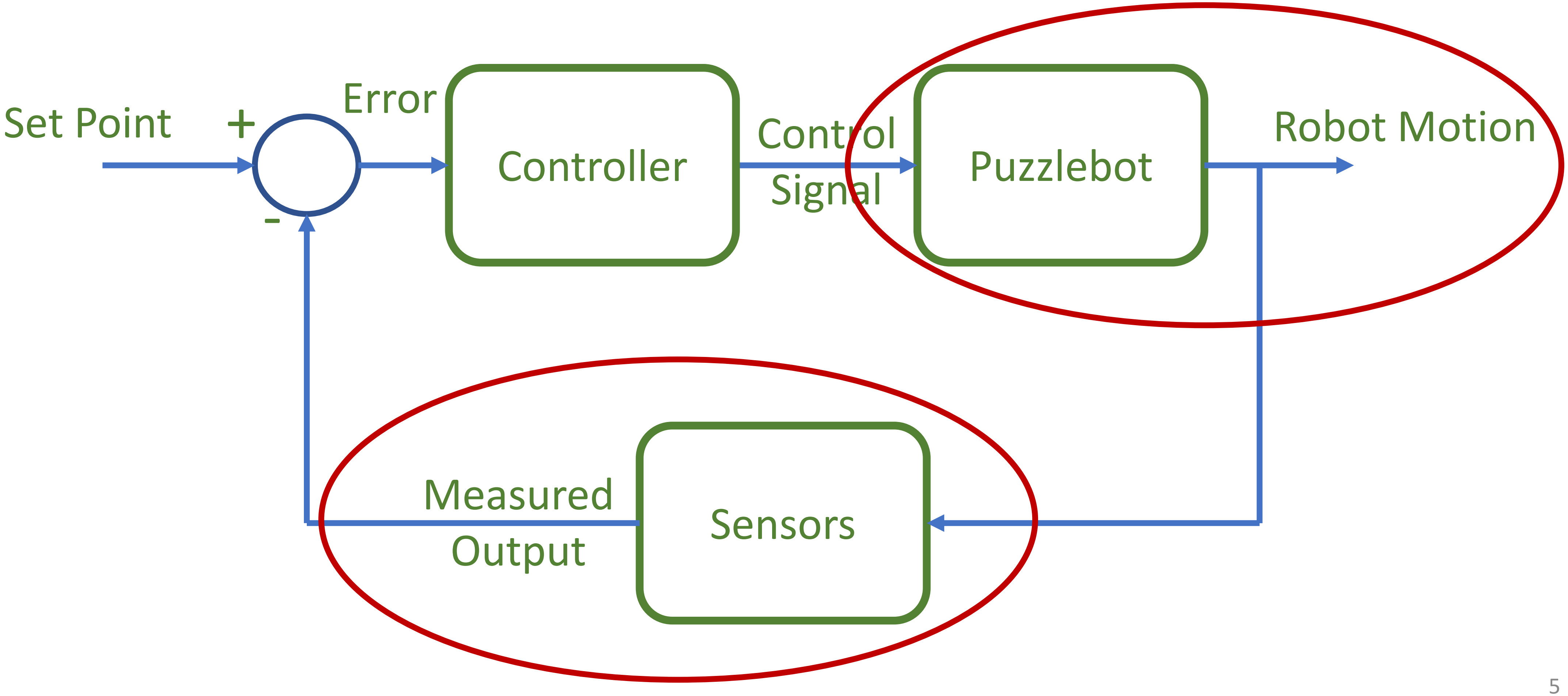


The Task

The Control System

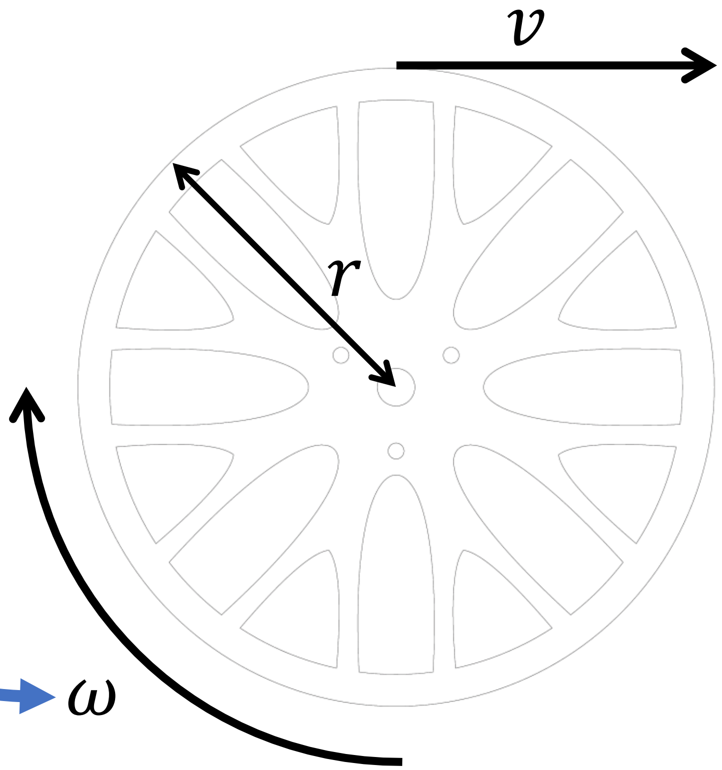


The Control System



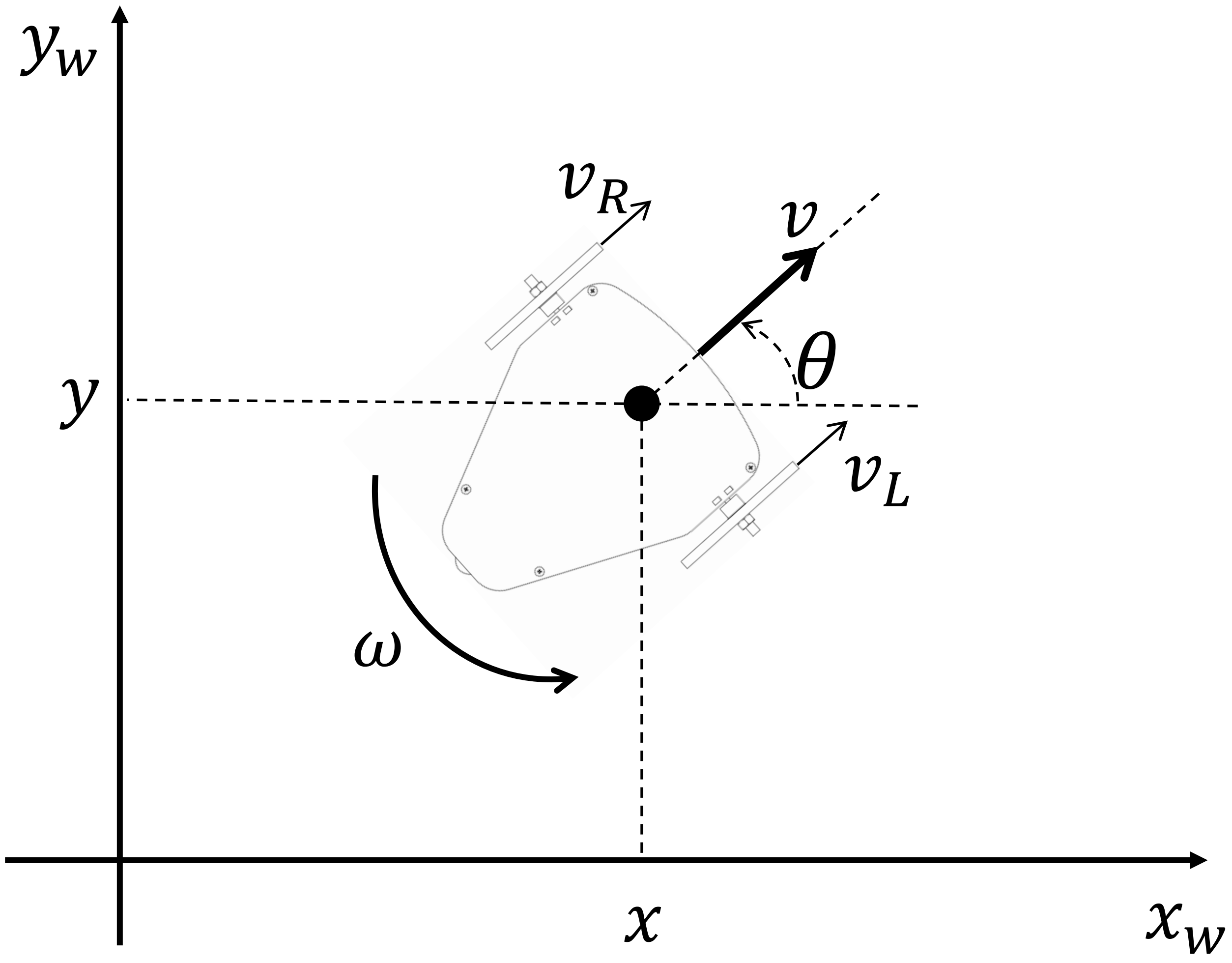
Determining the Robot Position

```
/cmd_vel
/diagnostics
/rosout
/rosout_agg
/wl
/wr
```



$$v_{Wheel} = r\omega_{Wheel}$$

Determining the Robot Position

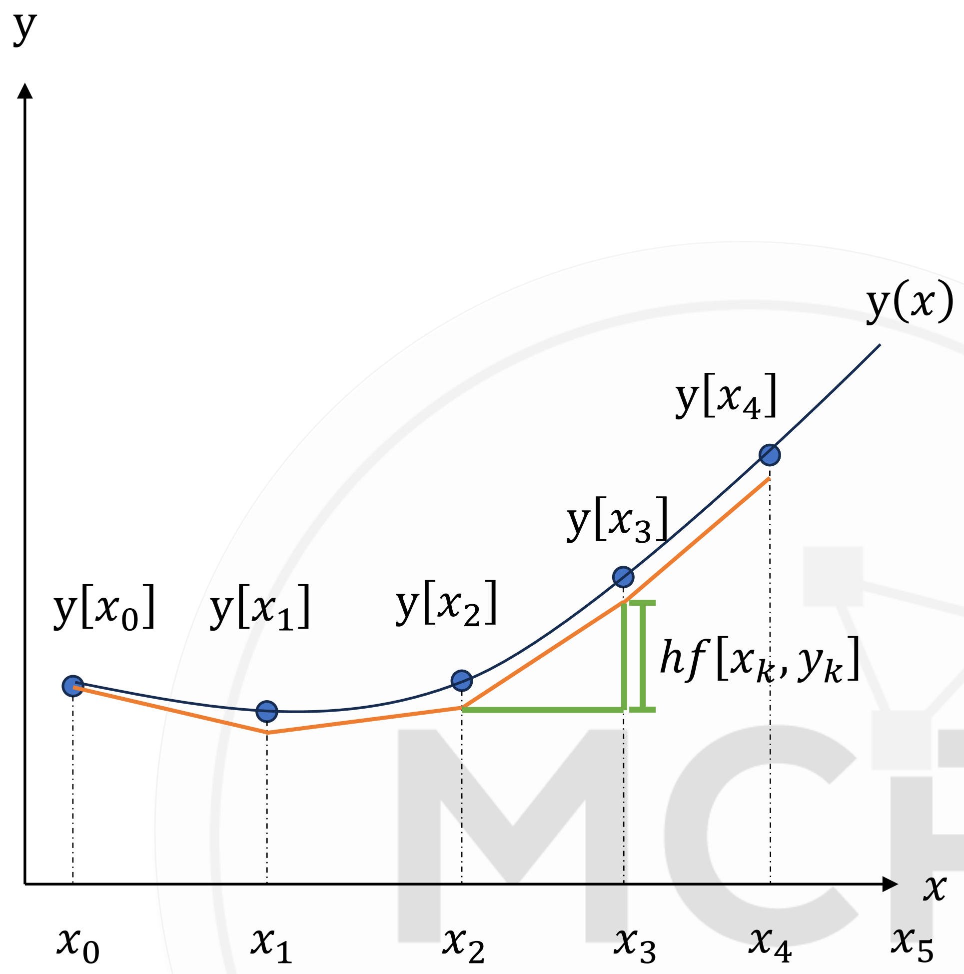


$$v_{Robot} = \frac{v_R + v_L}{2} = r\frac{\omega_R + \omega_L}{2}$$

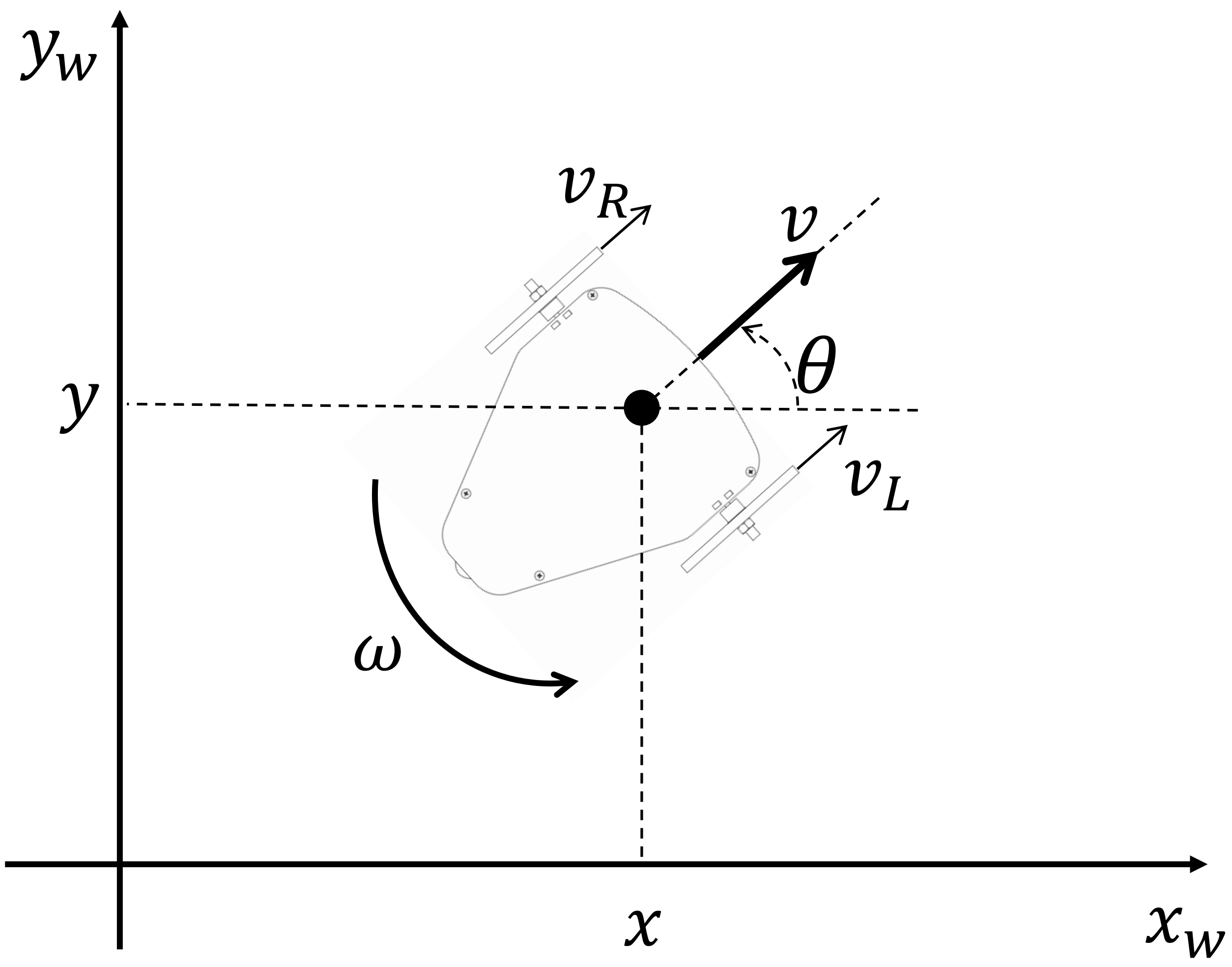
$$\omega_{Robot} = \frac{v_R - v_L}{l} = r\frac{\omega_R - \omega_L}{l}$$

Forward Euler method

- First order numerical procedure for ODEs
- Most basic method for numerical integration
- $y_{k+1} = y_k + \mathrm{h}f[x_k, y_k]$
- y_k - initial value



Determining the Robot Position



$$v_{Robot} = \frac{v_R + v_L}{2} = r\frac{\omega_R + \omega_L}{2}$$

$$\omega_{Robot} = \frac{v_R - v_L}{l} = r\frac{\omega_R - \omega_L}{l}$$

Determining the Robot Position

$$\boxed{\theta_{k+1}} = \boxed{\theta_k} + \boxed{r}\,\frac{\boxed{\omega_R - \omega_L}}{\boxed{l}}\,\boxed{dt}$$

$$\boxed{x_{k+1}} = \boxed{x_k} + \boxed{r}\,\frac{\boxed{\omega_R + \omega_L}}{2}\,\boxed{dt}\cos\boxed{\theta_k}$$

$$\boxed{y_{k+1}} = \boxed{y_k} + \boxed{r}\,\frac{\boxed{\omega_R + \omega_L}}{2}\,\boxed{dt}\sin\boxed{\theta_k}$$

Robot Location:
(x_k, y_k, θ_k): Pose of the robot at timestep k (m, m, rad).
Stored in memory, initial value 0

Robot Constants:
r: Wheel radius = 0.05 m
l: Distance between robot wheels = 0.19 m

Measured variables
$(\omega_\mathrm{R}, \omega_\mathrm{L})$: Wheel velocity (rad/s)
dt: Time between samples (s)

Values of θ must be contained within a single circle:
Either:

$$-\pi \leq \theta < \pi$$

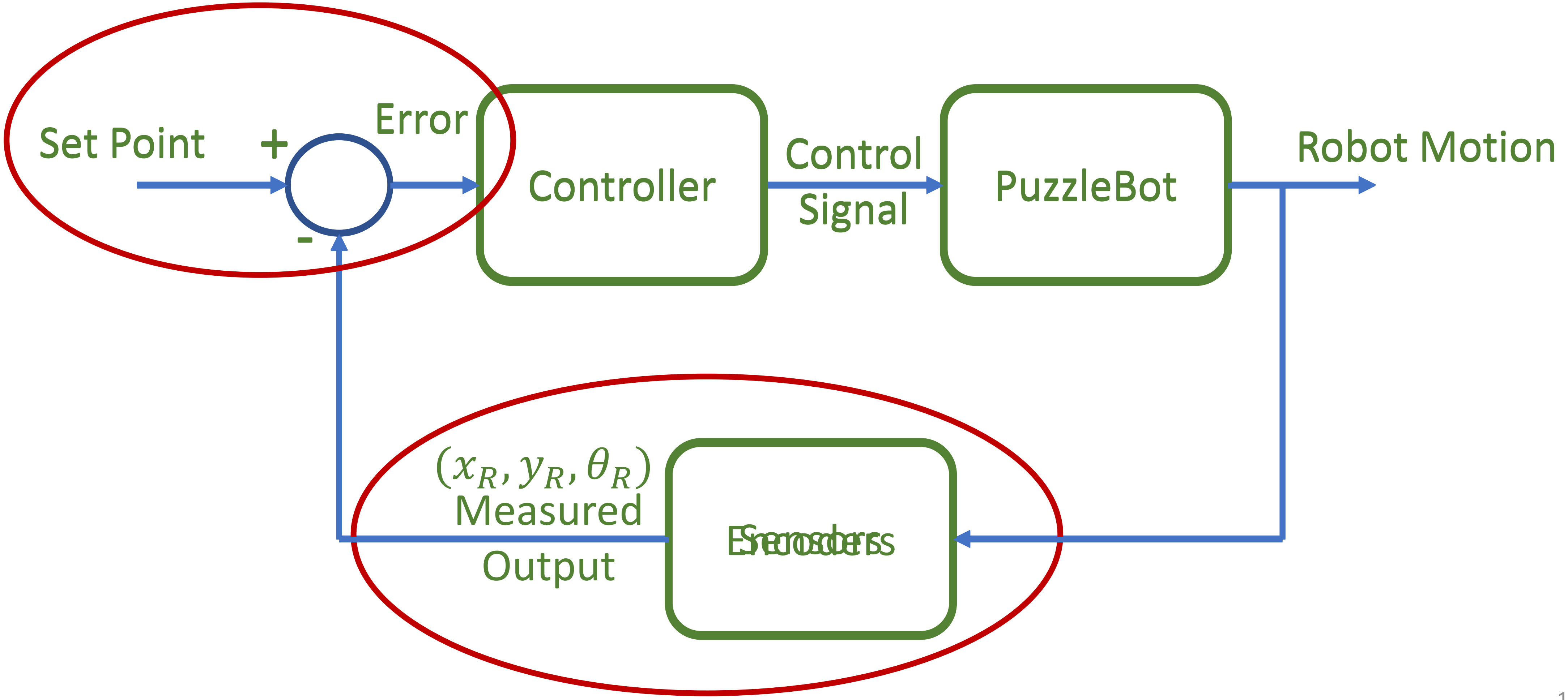
Or:

$$0 \leq \theta < 2\pi$$

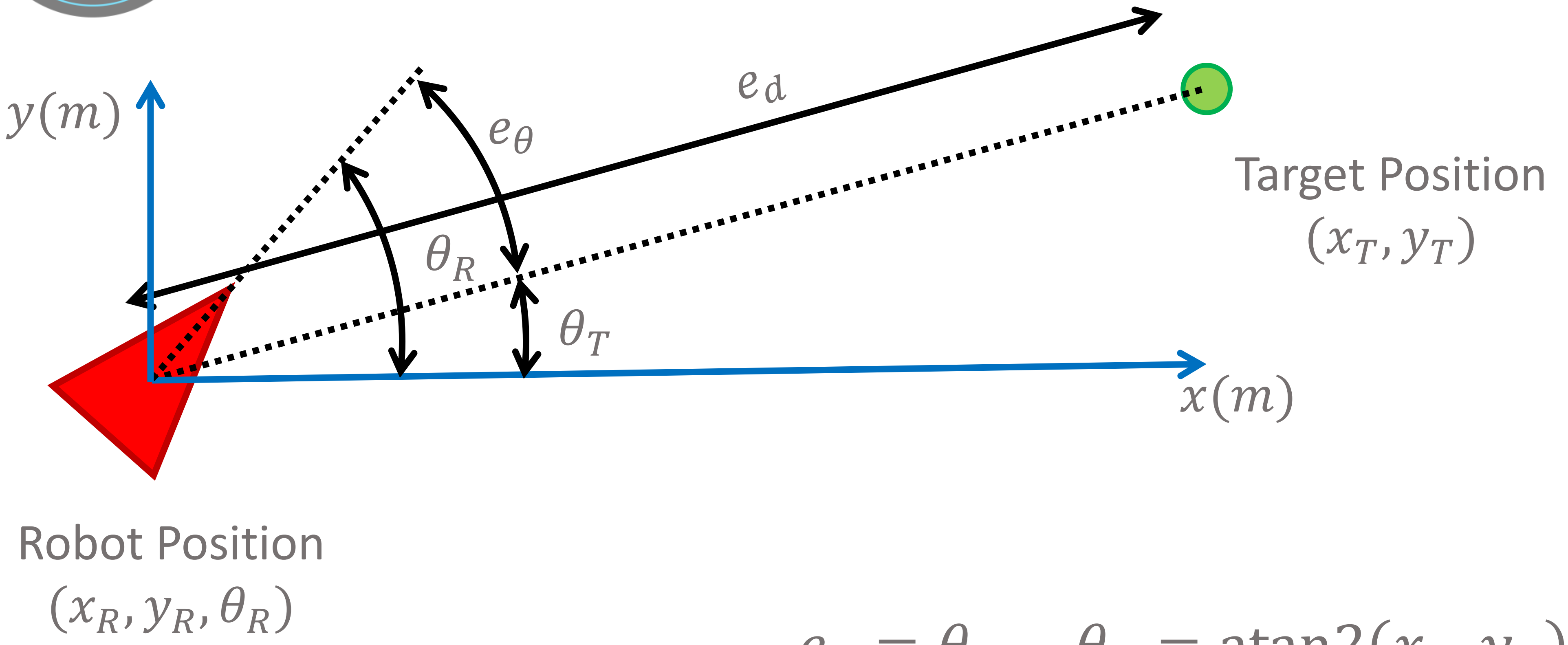
Activity 1

- Implement a ROS node that computes the robot location using the encoder data
  - It should subscribe to `/wl` and `/wr`, and publish the data to a suitable set of topics
  - The published messages could be a Pose2D message

The Control System



The Error

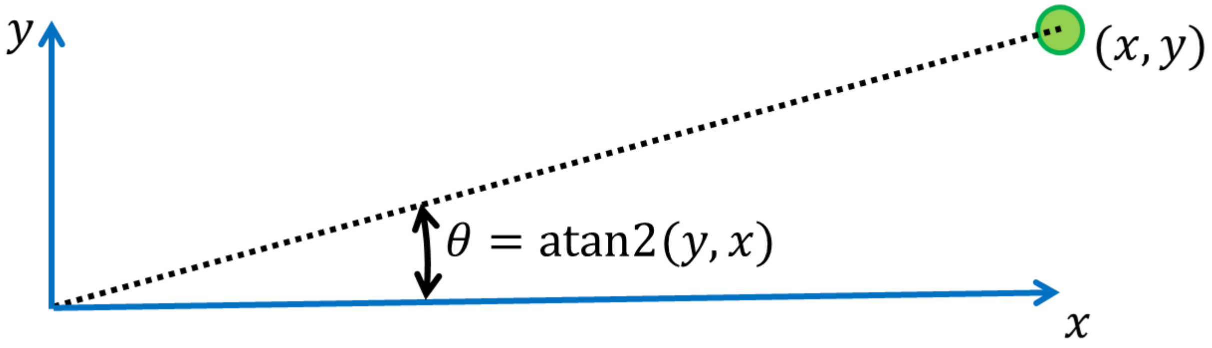


$$e_\theta = \theta_T - \theta_R = \text{atan2}(x_T, y_T) - \theta_R$$

$$e_d = \sqrt{(x_T - x_R)^2 + (y_T - y_R)^2}$$

atan2

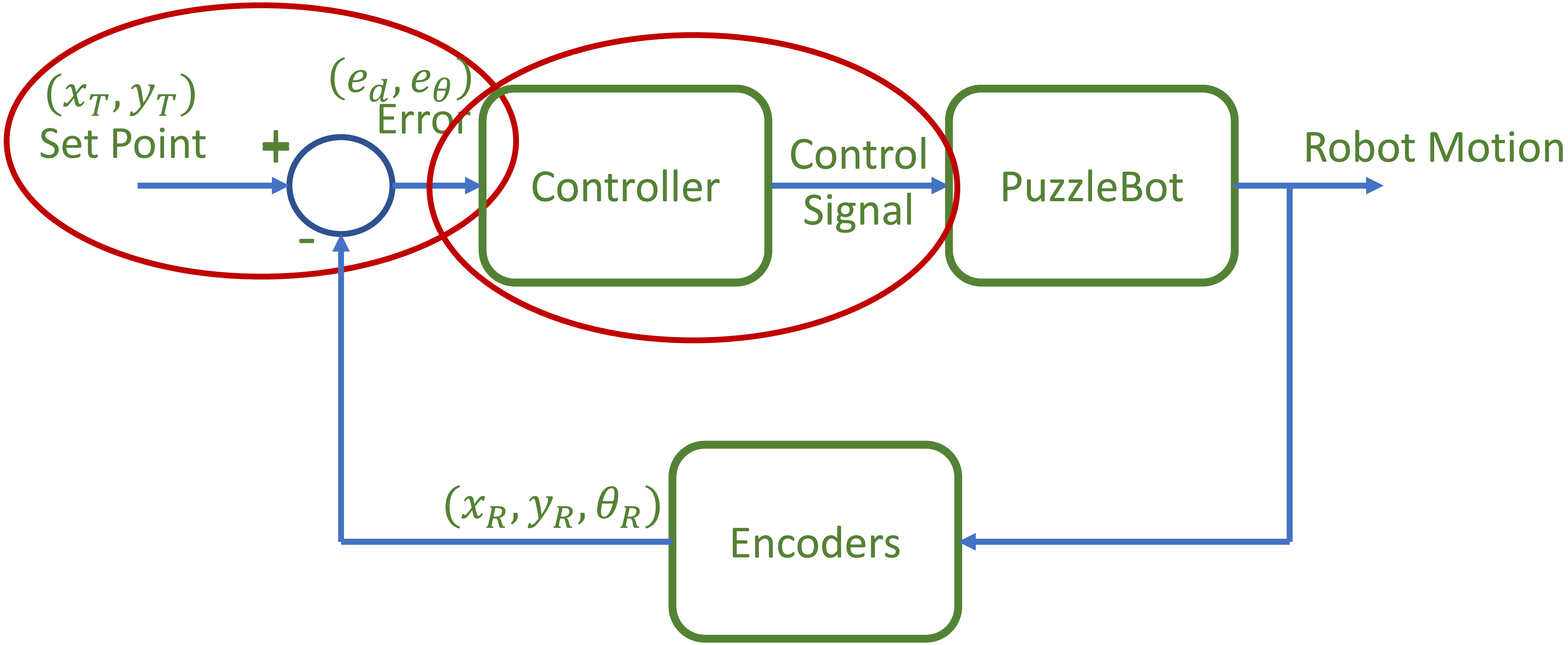
- The atan2 function is a special form of arctan or $\tan^{-1}$.
- It takes two arguments, y and x, and returns the angle to the x axis:



- It is included in most maths libraries, but it is recommended to use numpy, as numpy will be necessary later on in the course

```
import numpy
theta = numpy.arctan2(y,x)
```

The Control System



The Controller

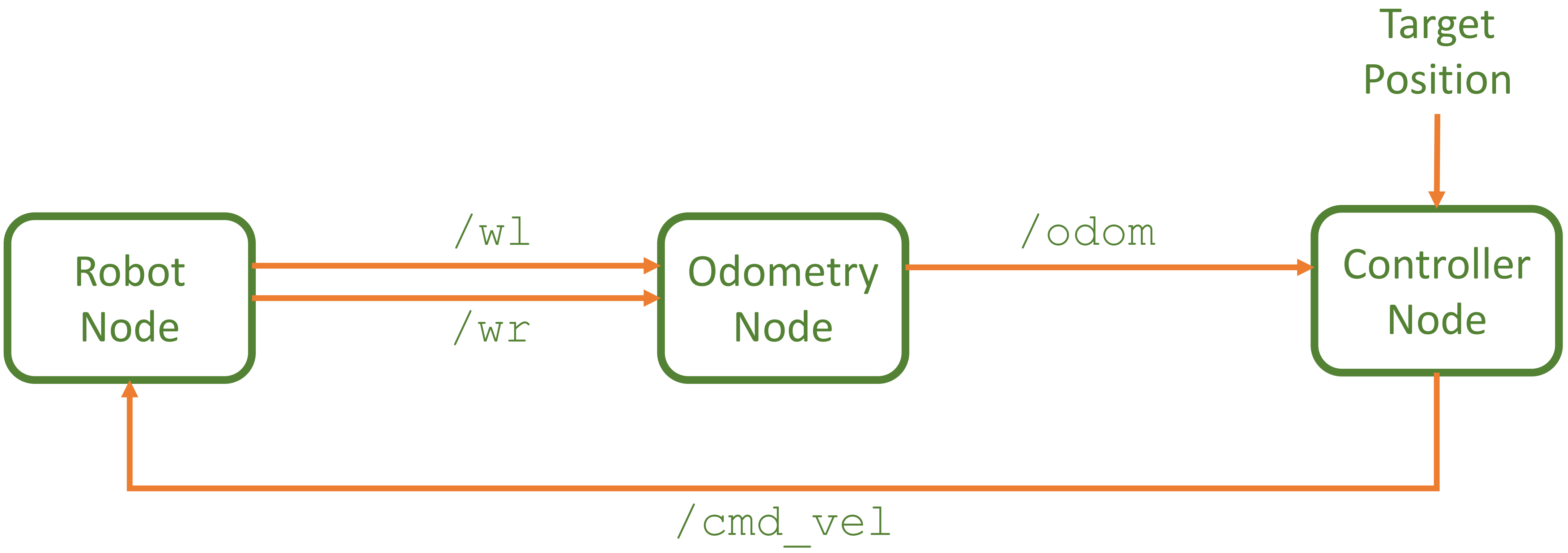
- Since the robot is inherently stable, a simple PID scheme should be sufficient.
- Start with a pair of proportional controllers:

$$v = K_v e_d$$

$$\omega = K_\omega e_\theta$$

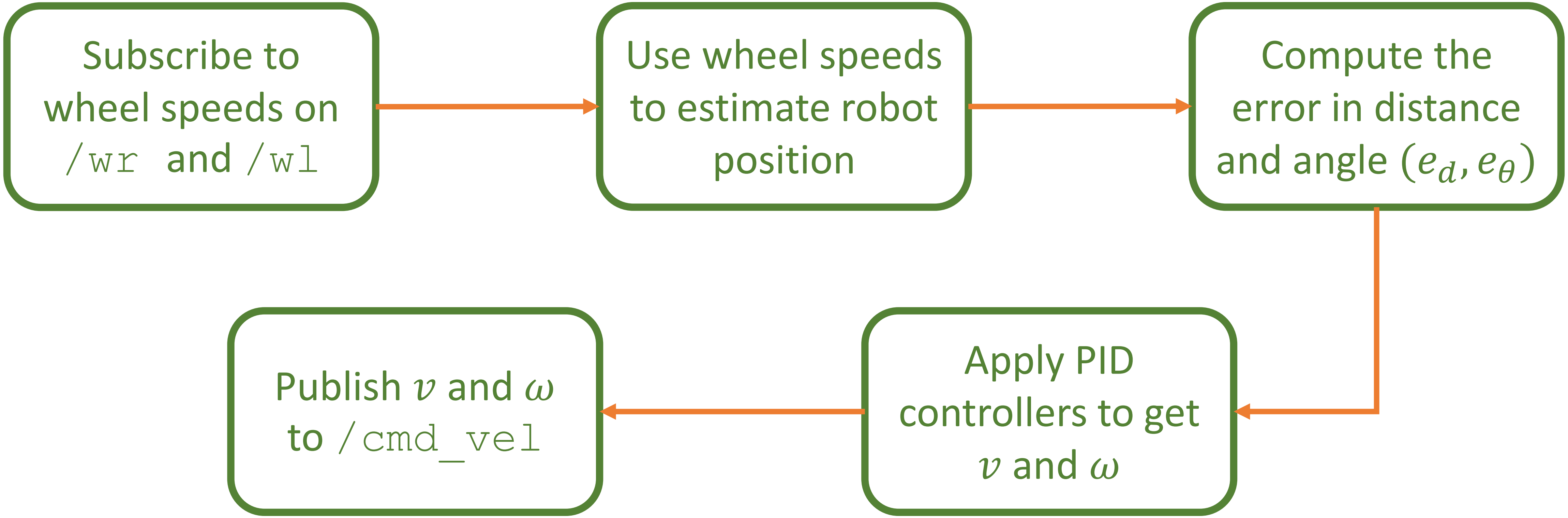
… and add integral and derivative elements if necessary.

ROS Setup



Target
Position
Robot
Node
/wl
/wr
Odometry
Node
/odom
Controller
Node
/cmd_vel

ROS setup



Subscribe to wheel speeds on /wr and /wl
Use wheel speeds to estimate robot position
Compute the error in distance and angle (e_d, e_θ)
Apply PID controllers to get v and ω
Publish v and ω to /cmd_vel

Activity 2

- Modify the previous node to publish e_d and e_θ.
- Set a target, and drive the robot around, checking that the angle to the target and the distance from the target are updated correctly
- Remember to wrap all angles to within 1 circle

Tips and Tricks

- Write and test your node with the PuzzleBot off the ground:
  - Use this to check the basics of your code are working correctly, such as the sign (+/-) of your controller parameters K_v and K_ω
  - Does the robot turn towards the goal?
  - Does the robot move towards or away from the goal?
- Tune one of the controllers at a time. You may find it easier to tune K_ω first, while setting your robot to move with a fixed forward speed.
- If in doubt, *lower* the value of the control constants.
- You may find it helpful to use a launch file to load your controller constants in from a config.yaml file.

Accuracy

- It will not be possible to tune the controllers such that the robot moves perfectly into position.
  - You will need a threshold after which your algorithm decides it has successfully arrived.
  - Suggested initial threshold: 10 cm
- Additionally, if you measure the position of the robot, it will likely not match up with the measurement computed from the encoders.
  - This is inevitable due to additive noise in the encoder readings.
  - The solution to this is to use sensors that can measure the position of the robot relative to its environment.

The Task

Thank You

Robotics For Everyone

{Learn, Create, Innovate};

MCR2 Manchester Robotics

T&C

Terms and conditions

{Learn, Create, Innovate};

Terms and conditions

- *THE PIECES, IMAGES, VIDEOS, DOCUMENTATION, ETC. SHOWN HERE ARE FOR INFORMATIVE PURPOSES ONLY. THE DESIGN IS PROPRIETARY AND CONFIDENTIAL TO MANCHESTER ROBOTICS LTD. (MCR2). THE INFORMATION, CODE, SIMULATORS, DRAWINGS, VIDEOS PRESENTATIONS ETC. CONTAINED IN THIS PRESENTATION IS THE SOLE PROPERTY OF MANCHESTER ROBOTICS LTD. ANY REPRODUCTION, RESELL, REDISTRIBUTION OR USAGE IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF MANCHESTER ROBOTICS LTD. IS STRICTLY PROHIBITED.*
- *THIS PRESENTATION MAY CONTAIN LINKS TO OTHER WEBSITES OR CONTENT BELONGING TO OR ORIGINATING FROM THIRD PARTIES OR LINKS TO WEBSITES AND FEATURES IN BANNERS OR OTHER ADVERTISING. SUCH EXTERNAL LINKS ARE NOT INVESTIGATED, MONITORED, OR CHECKED FOR ACCURACY, ADEQUACY, VALIDITY, RELIABILITY, AVAILABILITY OR COMPLETENESS BY US.*
- *WE DO NOT WARRANT, ENDORSE, GUARANTEE, OR ASSUME RESPONSIBILITY FOR THE ACCURACY OR RELIABILITY OF ANY INFORMATION OFFERED BY THIRD-PARTY WEBSITES LINKED THROUGH THE SITE OR ANY WEBSITE OR FEATURE LINKED IN ANY BANNER OR OTHER ADVERTISING.*