Robotics 311 : How to build robots and make them move

Prof. Elliott Rouse

GSI Yves Nazon MS

Fall 2022

Photo credit: Brenda Ahearn / U-M CoE

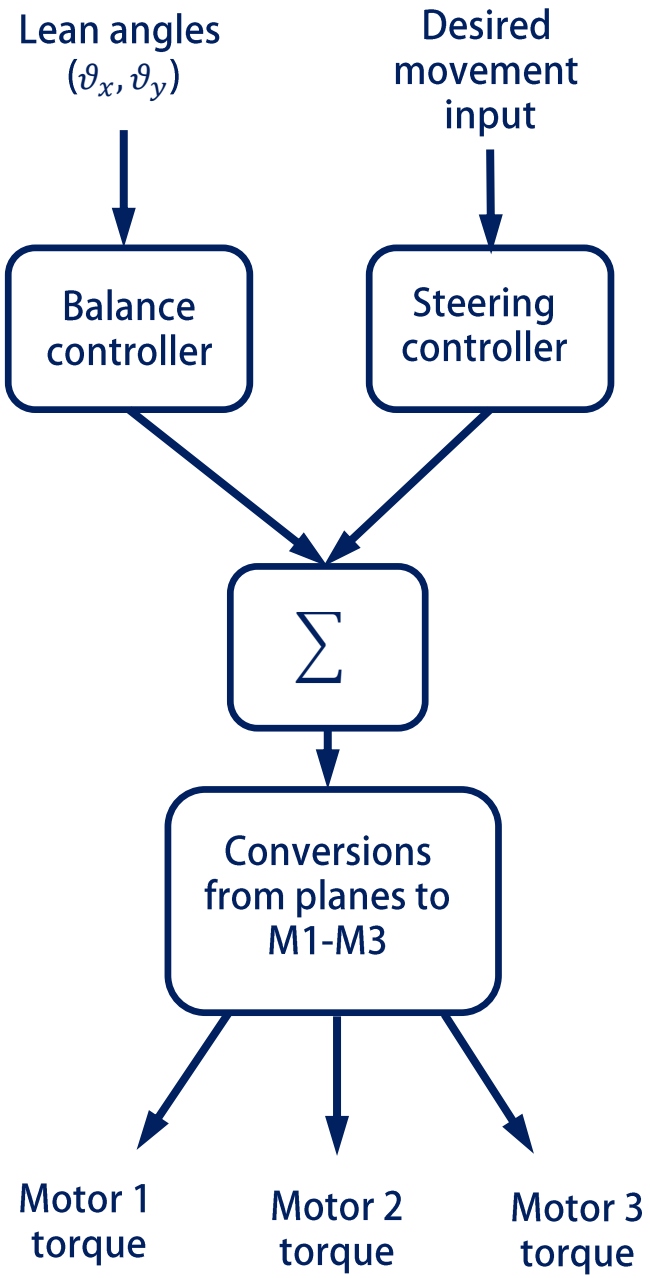
ROB 311 – Lab 11

- Continue with balance control
- Learn how to pair PS4 controller
- Begin steering control

- Announcements
  - If you haven't filled out the t-shirt form, do this ASAP

Balance and Steering Controllers

- We break the controller into the two planes
- Each plane will be handled independently
- Each plane has two controllers that run in parallel
  - Balance controller / steering controller
  - They will be separate but will run simultaneously
- There will be four total controllers in parallel
- We will superimpose the torques from the balance and steering controllers
- Simultaneous balance and steering
- We will begin with the **balance controller**
- Let's think about how this controller should be designed



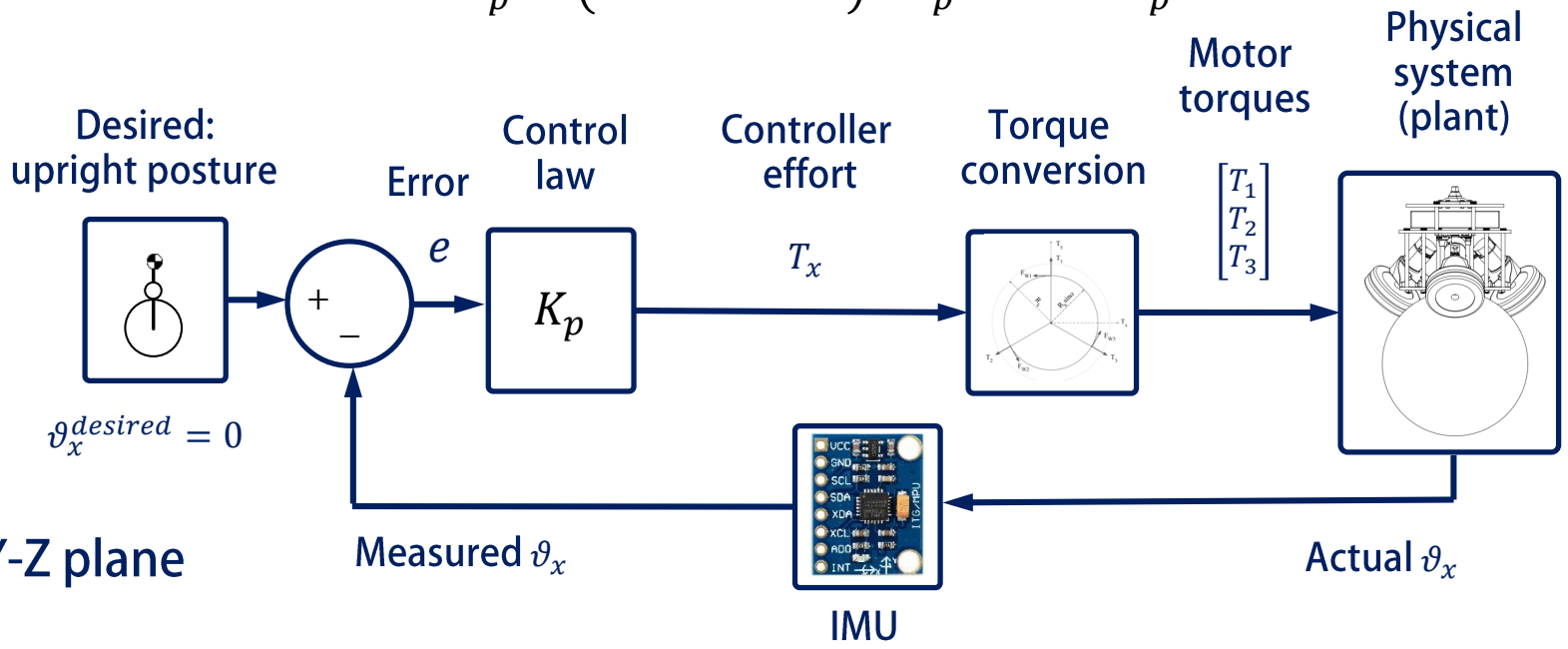
Balance Controller

- Let's build a basic feedback controller using chassis lean angle
- We know we want controller effort to be

$$T = -K_p \cdot \vartheta_{axis}$$

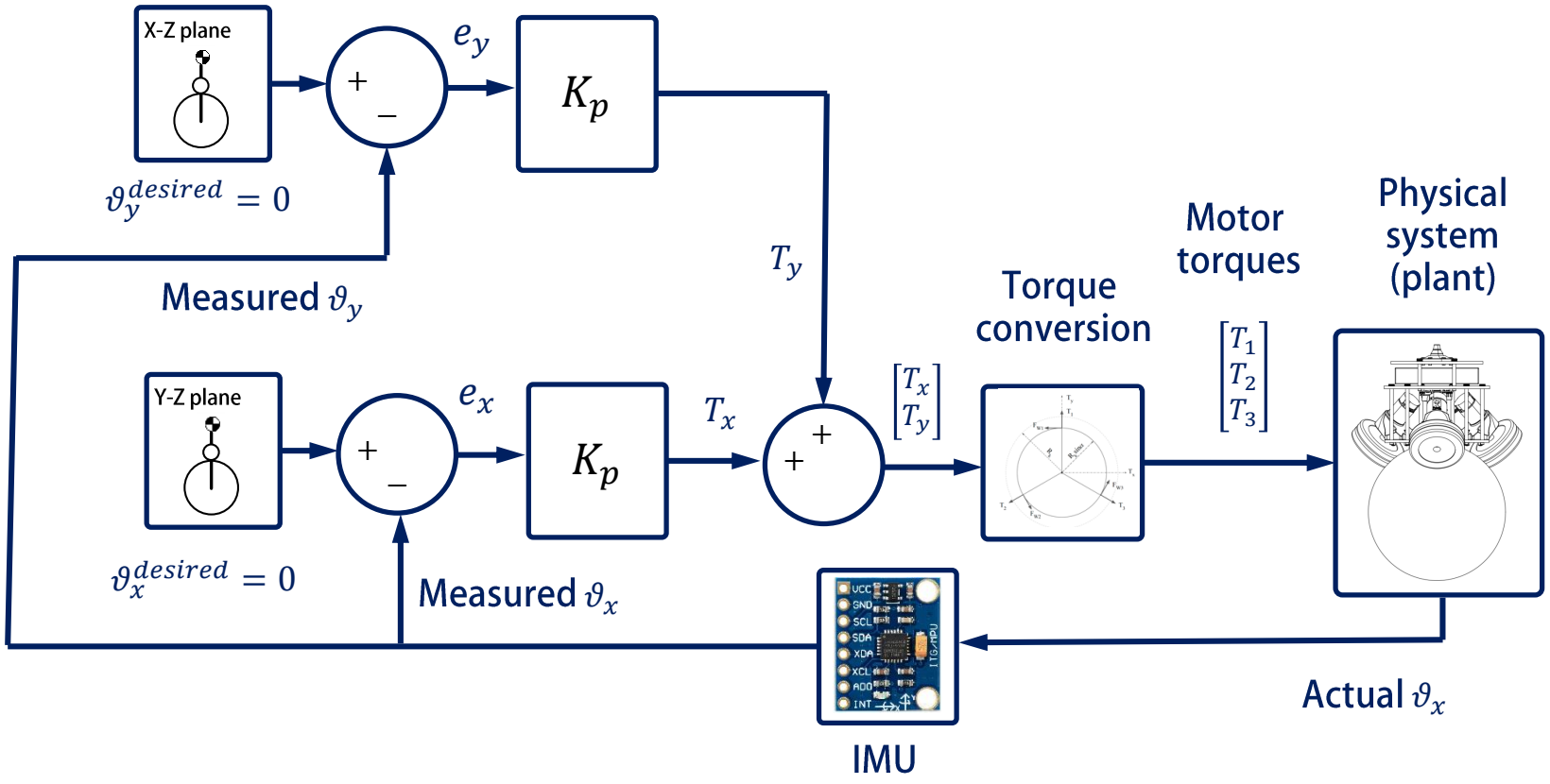
Controller gain
or 'proportional gain'

- Lets define a reference trajectory of upright posture ($\vartheta_x^{desired} = \vartheta_y^{desired} = 0$)
- By defining a reference trajectory, we can make our approach more general
- Control law: $T = e \cdot K_p = \left(\vartheta^{desired} - \vartheta\right) \cdot K_p = -\vartheta \cdot K_p$



Balance Controller

- How about for both planes?
- Control law: $T = e \cdot K_p = \left(\vartheta^{desired} - \vartheta\right) \cdot K_p = -\vartheta \cdot K_p$
- How do we choose K_p? This process is called 'tuning' a controller



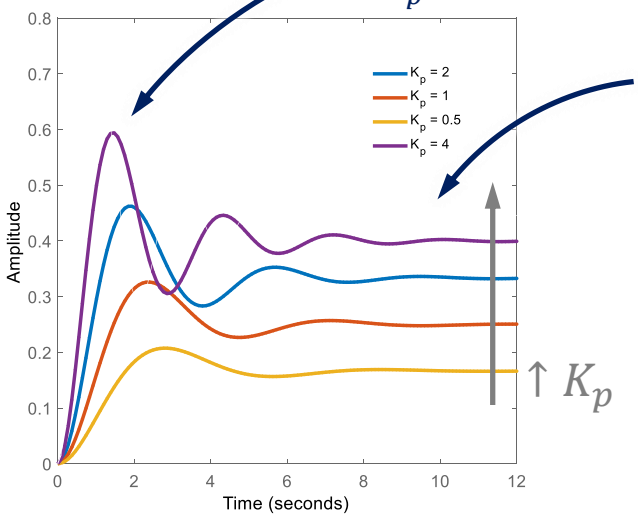
Tuning Your Controller

- A proportional controller is like a virtual spring (when controlling around position)
- The greater the K_p (stiffness), the greater the restoring torque to balance
- But it has to be selected carefully
- A controller gain that's too high will cause oscillations and instability
- Let's look at an example 2nd order system under proportional control trying to go from 0 to 1 at $t = 0$

More oscillations as K_p is increased

The responses get closer to the goal as K_p is increased

These systems are trying to reach the goal of 1 at $t = 0$



These responses are known as 'step responses'

Tuning Your Controller

- You will use code on Canvas/Lab 10 – `ROB311_stability_PS4.py`
- Code walk through
- You will need to iteratively adjust your controller gains

Adjust these gains

```
196  # ------------------------------------------------------------------------
197  ##############  THESE WILL NEED TO BE CAREFULLY ADJUSTED ##################
198  # ------------------------------------------------------------------------
199
200  # Proportional gains for the stability controllers (X-Z and Y-Z plane)
201
202  KP_THETA_X = 0.0                                # Adjust until the system balances
203  KP_THETA_Y = 0.0                                # Adjust until the system balances
204
205  # ------------------------------------------------------------------------
206  ##########################################################################
207
```

- Begin with a small number and increase until it starts to balance
- This will take time and you may need to make many adjustments
- Be careful and be ready to `ctrl+c` to exit if it begins doing something bad
- Tuning a controller can be very dangerous when working with powerful machines—for the ball-bot, we don't need to worry too much

PS4 Controller

- We will now add a controller to provide steering commands to the ball-bot
- We will use an off-brand PS4 controller
- It uses a Bluetooth connection already built into your RPi
- Python API developed that enables easy controller use
- We have access to information from each button press
- There are many options for how to control
- You will use the buttons to define a control law for rotation around the z-axis

We will use button presses to control your ball-bot

How to Pair and Use Your PS4 Controller

- To put your PS4 controller in pairing mode, hold down home + share buttons on the PS4 controller
- Then, from the RPi terminal execute the following commands

```
sudo bluetoothctl
```

- This will open a new command line, using the bluetoothctl

```
agent on
discoverable on
pairable on
default-agent
scan on
```

- This configures the RPi Bluetooth settings and gets it ready to pair with your controller
- Now, we need to tell it to pair with your specific PS4 controller
- We will use its MAC address to pair, a unique identifier related to the specific controller (anything with a radio has a MAC address, including your RPis)

How to Pair and Use Your PS4 Controller

- Example terminal showing Bluetooth commands
- Next we need to pair for your specific controller
- Find your controller's MAC address—should be labeled on the bottom
- In bluetoothctl, next execute

  `connect CONTROLLER_MAC_ADDRESS`

- The controller light should turn blue
- Now we want to trust this connection in the future

  `trust CONTROLLER_MAC_ADDRESS`

- Future connections: you will need to open `bluetoothctl` and `connect` with your MAC address

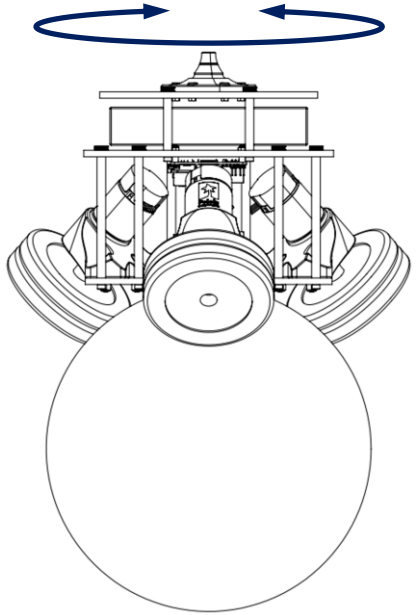
```
pi@raspberrypi:~ $ sudo bluetoothctl
Agent registered
[bluetooth]# agent on
Agent is already registered
[bluetooth]# default-agent
Default agent request successful
[bluetooth]# scan on
Discovery started
[CHG] Controller DC:A6:32:BA:33:FD Discovering: yes
[CHG] Device 60:16:53:0B:8F:C2 RSSI: -41
[CHG] Device 60:16:53:0B:8F:C2 ManufacturerData Key: 0x004c
[CHG] Device 60:16:53:0B:8F:C2 ManufacturerData Value:
```

Control Z-Axis Rotation

- Your goal is to use the controller to provide a torque command to rotate the bot around the z-axis
- Something like a steering wheel with a PS4 controller
- Triggers / joysticks provide a continuous signal
- Bumpers / buttons provide a logical signal (0 or 1)
- Torque must be added to balance control script (T_z)

Want to know more about connecting to your PS4 controller? Click here

You define the logic / use demos



PS4 Controller API

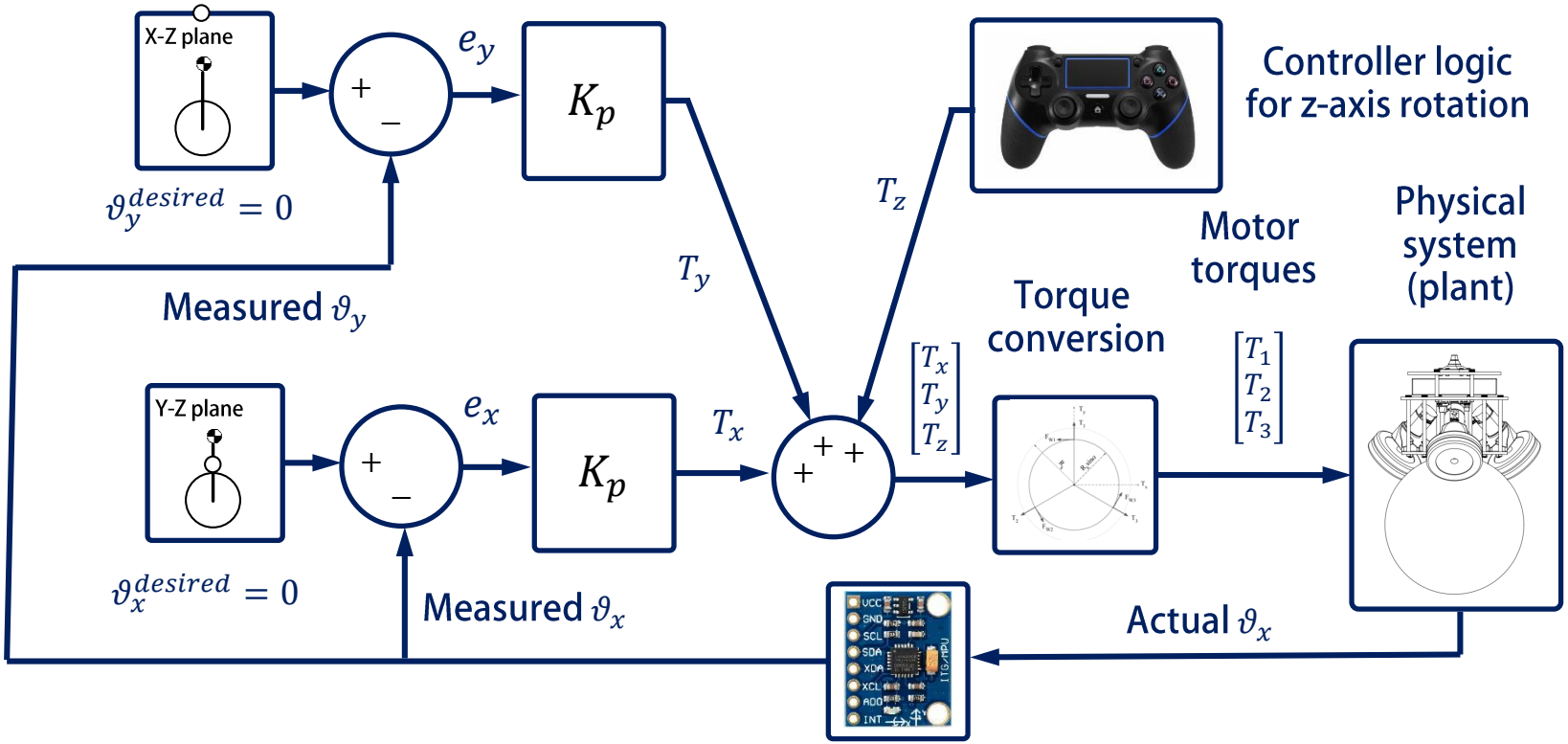
- You can download a controller demonstration script from Canvas

  `ps4_controller_api.py`

- This Python program runs a loop and watches the buttons on the controller
- It will print when some button is pressed
- Run this program to familiarize yourself with using the PS4 controller
- Then add the controller class / variables to your balance controller
- We have added 3 example options to provide z-axis torque using the controller
  - `rob311_bt_controller.tz_demo_1`
  - `rob311_bt_controller.tz_demo_2`
  - `rob311_bt_controller.tz_demo_3`
- Once you've familiarized yourself with the PS4 controller, you need to add all the missing commands from the PS4 script to your balance controller script
- You'll need to set the z-axis torque to the value from the PS4 controller (above)

Controller for Z-Axis Torque

- Now we will add the z-axis torque from button commands you choose
- Triggers provide continuous values and buttons provide binary values
- Create a torque function using the button presses and add to the torque commands (or use the demos)
- You will need to set the z-axis torque to the torque value from the PS4 controller



Add To Your Balance Control Script

- Add your logic for z-axis torque to the balance control script
- This script should have been modified to include the PS4 controller
- When you've determined your logic for axis rotation, set the z-axis torque
- To exit the control script, you can use `ctrl+c` like usual
- The balance controller, PS4 controller, and RPi/Pico communication all run on separate threads
- But to stop the PS4 controller, you also need to press the options button after you stop the your controller script

Press 'options' button to stop PS4 controller after you've stopped the balance controller