

# Lab 1

---

# Driving and Controller

This lecture is part of the RACECAR-MN introductory robotics course.  
You can visit the course webpage at [mitll-racecar-mn.readthedocs.io](http://mitll-racecar-mn.readthedocs.io).



# Objectives

---

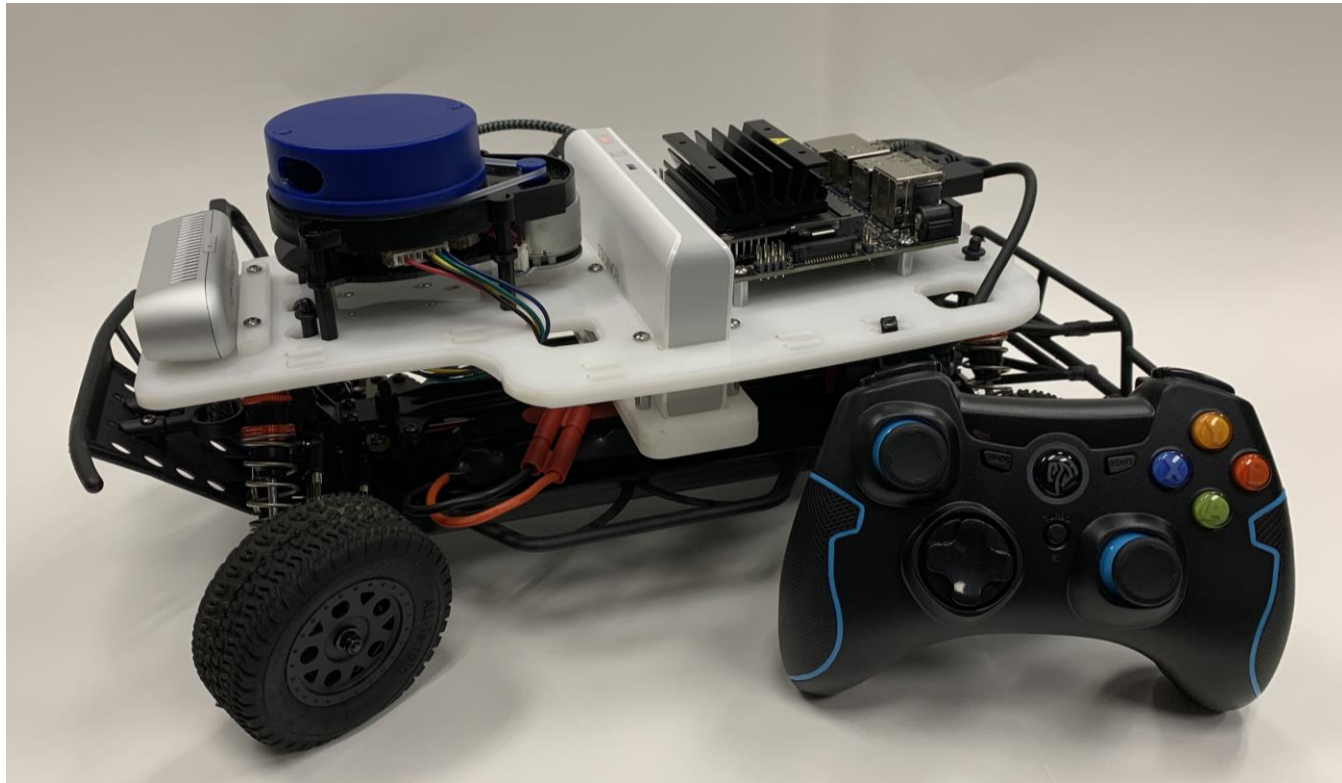
**Main Objective:** Program the car to respond to controller input and drive in predefined shapes

## Learning Objectives

- Use the start-update paradigm to create a program which can run on the car
- Use the drive module to move the car
- Use the controller module to respond to input from the Xbox controller in real time

# About the RACECAR-MN

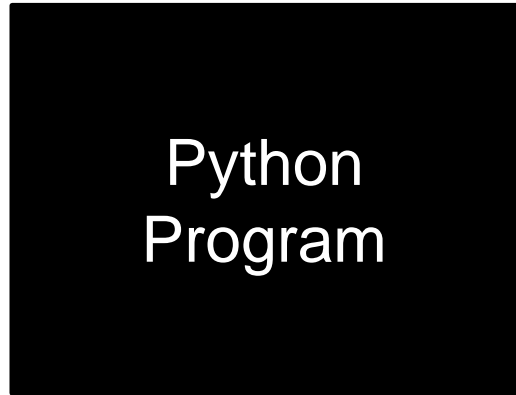
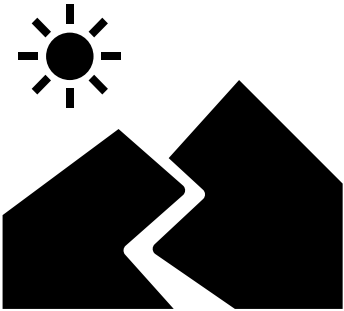
---



# About the RACECAR-MN

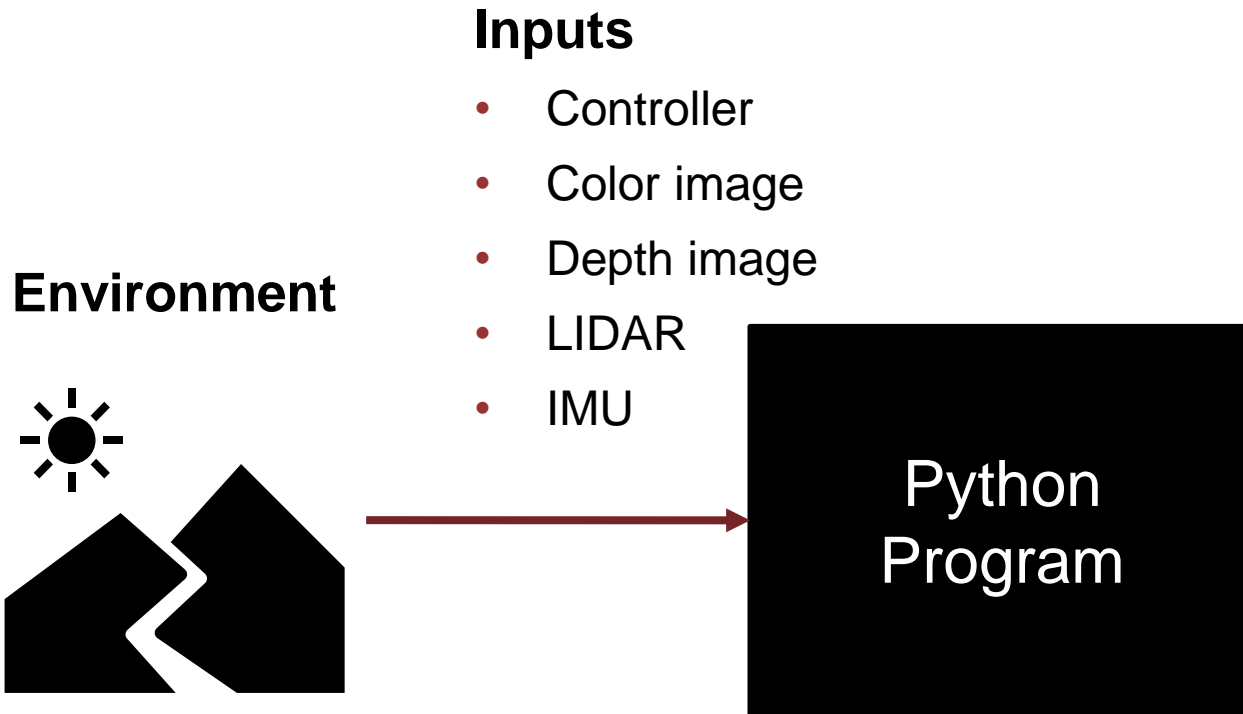
---

## Environment



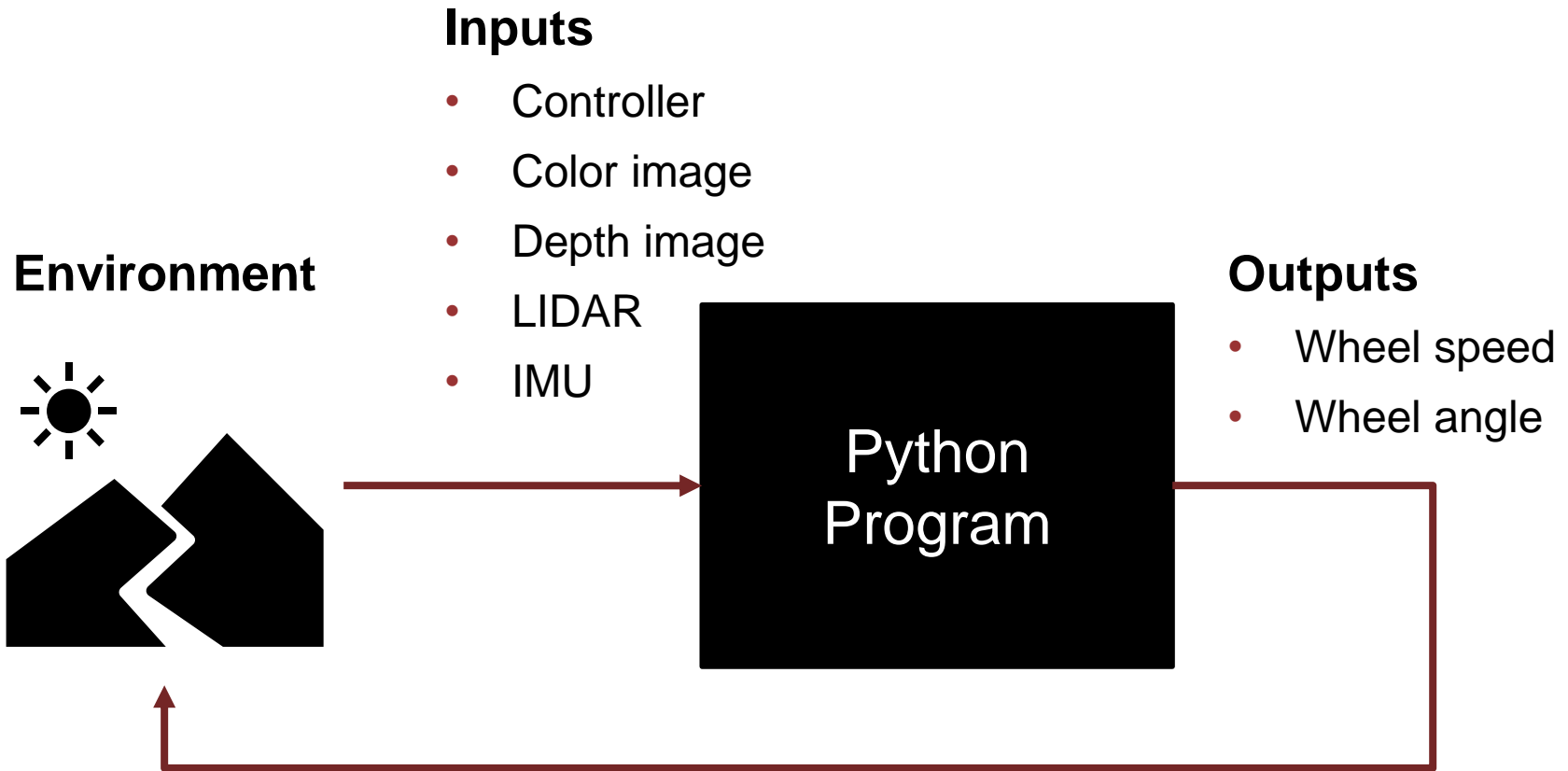
# About the RACECAR-MN

---



# About the RACECAR-MN

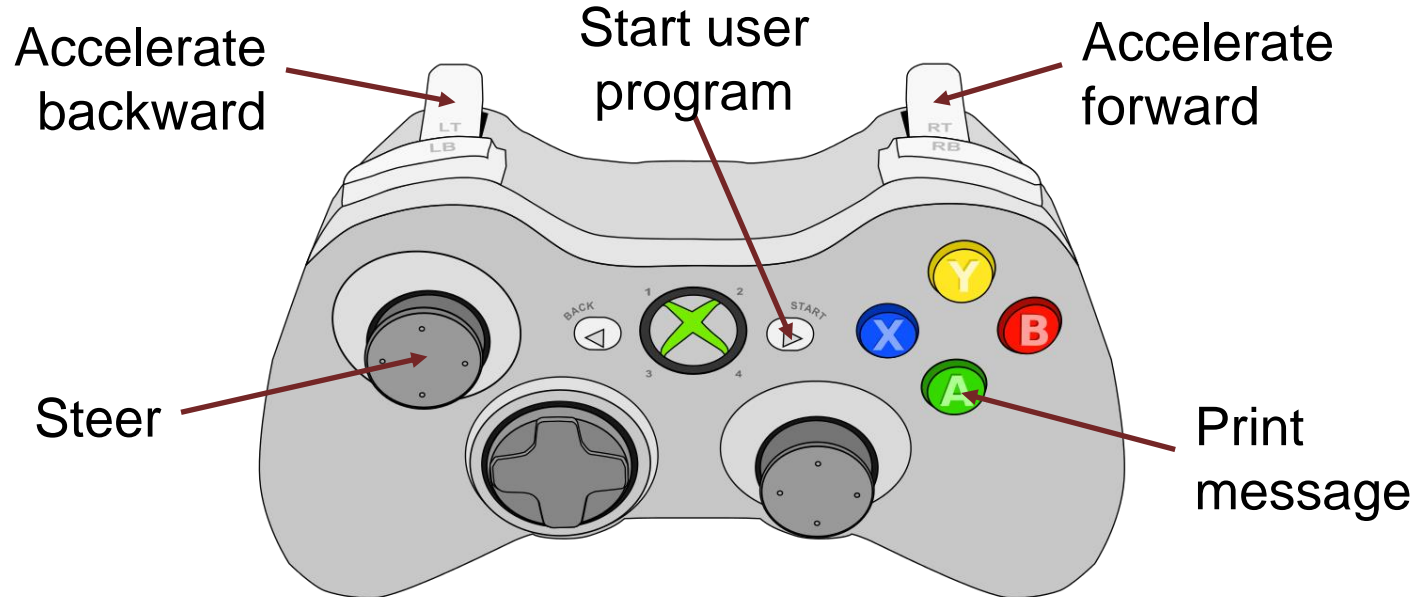
---



# Car Modes – Default Drive

---





- When you launch a RACECAR program, the car starts in **default drive** mode so you can easily drive around



# Car Modes – User Program

---

- In **user program** mode, the car is controlled by a Python script using the start-update paradigm
  - **Start:** Run once when the program begins
  - **Update:** Run 60 times per second

Button		Effect
Start		Enter user program mode
Back		Enter default drive mode
Start + Back	 	Exit the program



# Start-Update Paradigm

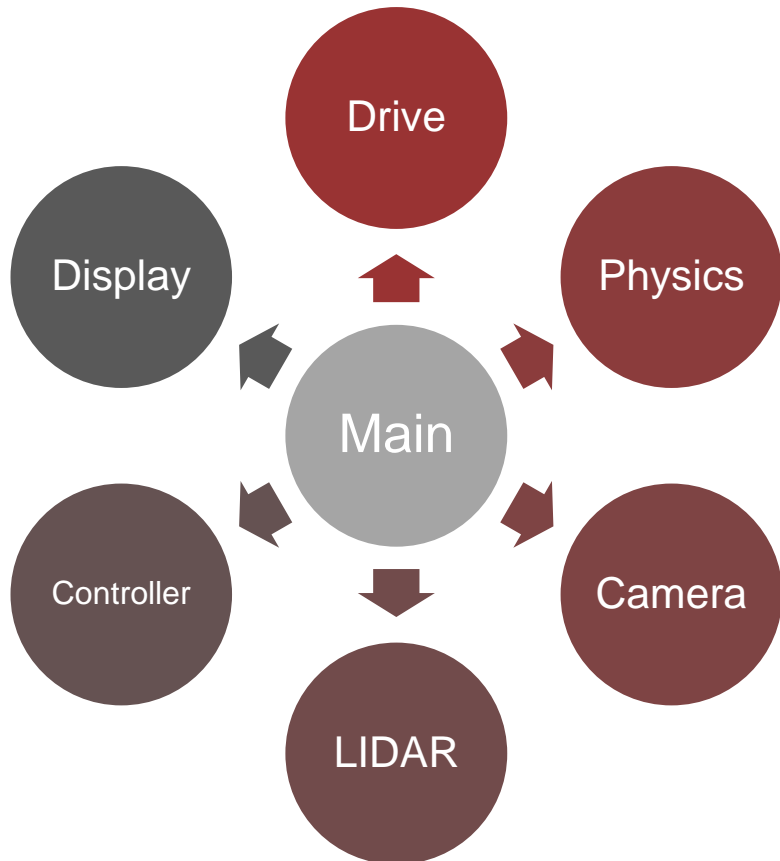
---

Example: Demo.py

# racecar\_core Library

---

- Exposes easy access to the car's hardware
- Under the hood: ROS
- Documentation
- Implementation



# racecar\_utils Library

---

- Helper functions to process inputs/outputs of racecar\_core
- You will write everything in racecar\_utils during the Jupyter notebook activates throughout the course
- Documentation
- Implementation

# Drive Module

---

- Controls the car's speed and wheel angle
- Public Interface
  - `set_speed_angle(speed, angle)`
  - `stop()`
  - `set_max_speed(max_speed)`

# Drive Module - Example

---

```
def update():
    global counter
    if counter < 1:
        rc.drive.set_speed_angle(1, 0)
    elif counter < 2:
        rc.drive.set_speed_angle(1, -0.5)
    else:
        rc.drive.stop()
    counter += rc.get_delta_time()
```

# Controller Module

---

- Measures input from the Xbox controller
- Public Interface
  - Button, Joystick, and Trigger enums
  - `is_down(button)`
  - `was_pressed(button)`
  - `was_released(button)`
  - `get_trigger(trigger)`
  - `get_joystick(joystick)`

# Controller Module - Example

---

```
def update():  
    if rc.controller.is_down(rc.controller.Button.RB):  
        speed = rc.controller.get_trigger(rc.controller.Trigger.RIGHT)  
        rc.drive.set_speed_angle(speed, 0)  
  
    if rc.controller.was_pressed(rc.controller.Button.B):  
        print("Kachow!")
```

# Lab 1 – Driving in Shapes

---

- Objectives
  - Implement default-drive style controls
  - A button = drive in a circle
  - B button = drive in a square
  - X button = drive in a figure eight
  - Y button = drive in a shape of your choice

lab1.py

