Lab 1 – Driving in Shapes

MIT Beaver Works Racecar Curriculum

https://matthewcalligaro.github.io/RacecarWebsite/

Lab Objectives

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

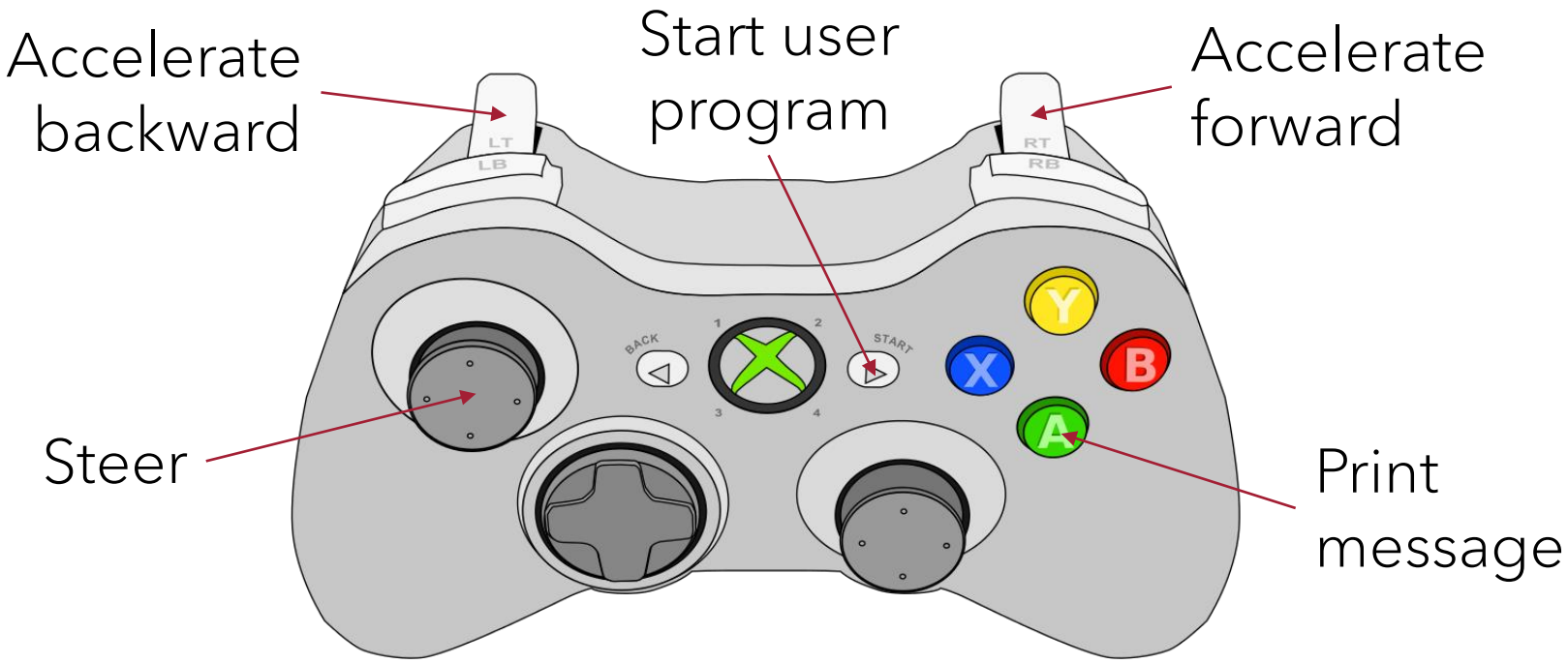
Learning objectives

- 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

Finished Car

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

- You can then tell the car to execute your Python program, which consists of a **start** and **update** function
  - Start: run once when you enter user program mode
  - Update: run 60 times per second in user program mode

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

Car Modes – Starter Code

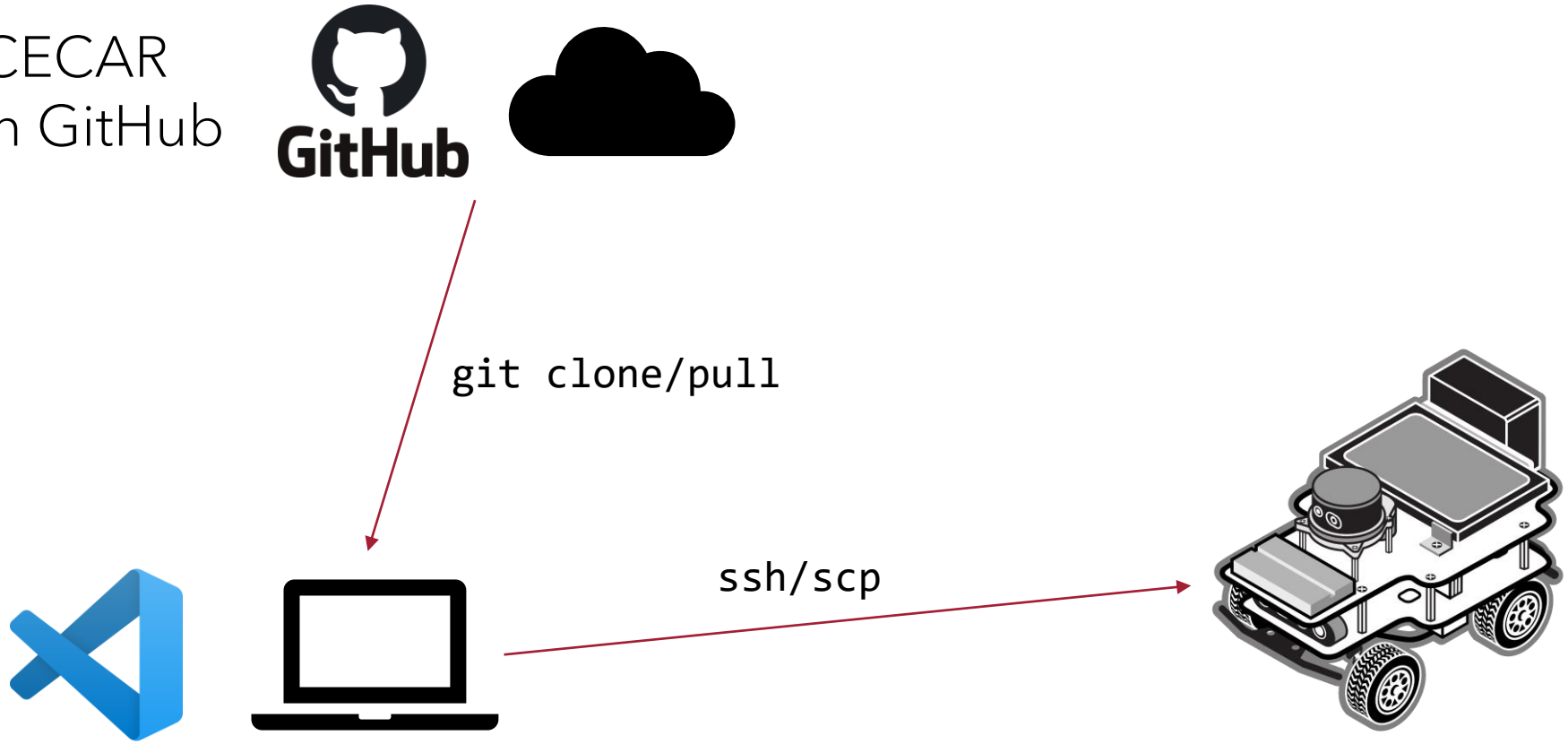
- For each lab, you will be provided with a Python skeleton consisting of a blank start and update function
- You will complete each lab by filling the start and update function
- You may wish to add additional helper functions and global variables

```python
##############################################################################
# Imports
##############################################################################

import sys
sys.path.insert(0, '../../library')
from racecar_core import *
rospy.init_node('racecar')


##############################################################################
# Global variables
##############################################################################

rc = Racecar()


##############################################################################
# Functions
##############################################################################

def start():
    """
    This function is run once every time the start button is pressed
    """
    pass

def update():
    """
    After start() is run, this function is run every frame until the back button
    is pressed
    """
    pass


##############################################################################
# Do not modify any code beyond this point
##############################################################################

if __name__ == "__main__":
    rc.set_start_update(start, update)
    rc.go()
```

Example starter code

Lab Logistics

1. The RACECAR files live on GitHub



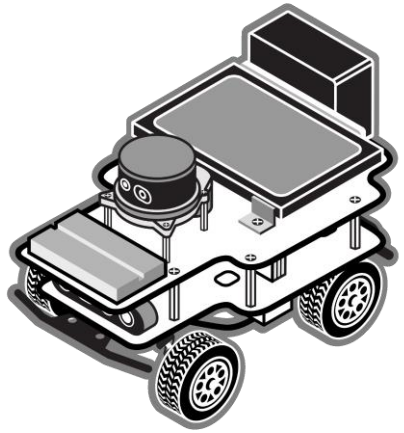
Lab Logistics

1. The RACECAR files live on GitHub

2. Clone the RACECAR files to your computer

`git clone/pull`

`ssh/scp`



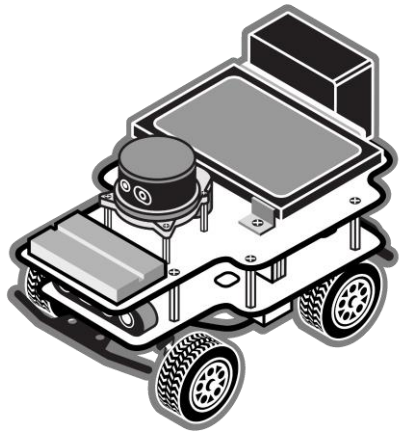
Lab Logistics

1. The RACECAR files live on GitHub

2. Clone the RACECAR files to your computer

`git clone/pull`

`ssh/scp`



3. Edit the starter code on your computer

Lab Logistics

1. The RACECAR files live on GitHub

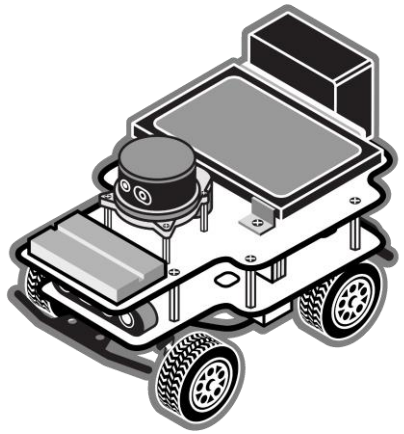
2. Clone the RACECAR files to your computer

`git clone/pull`

3. Edit the starter code on your computer

`ssh/scp`

4. Copy the files to you RACECAR



Lab Logistics

1. The RACECAR files live on GitHub

2. Clone the RACECAR files to your computer

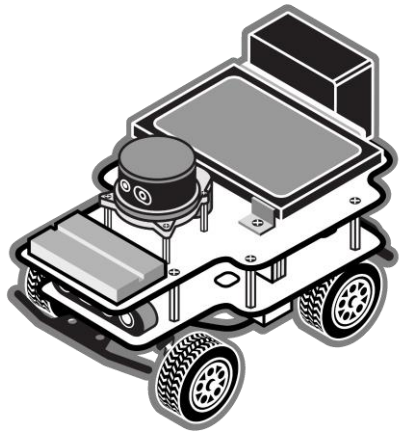
`git clone/pull`

3. Edit the starter code on your computer

`ssh/scp`

4. Copy the files to you RACECAR

5. Run the program on the RACECAR



Key Modules – Drive

- The **Drive** module allows us to move the car by setting its speed and the angle of the front wheels
- Public interface
  - `set_speed_angle(speed, angle)`
  - `stop()`

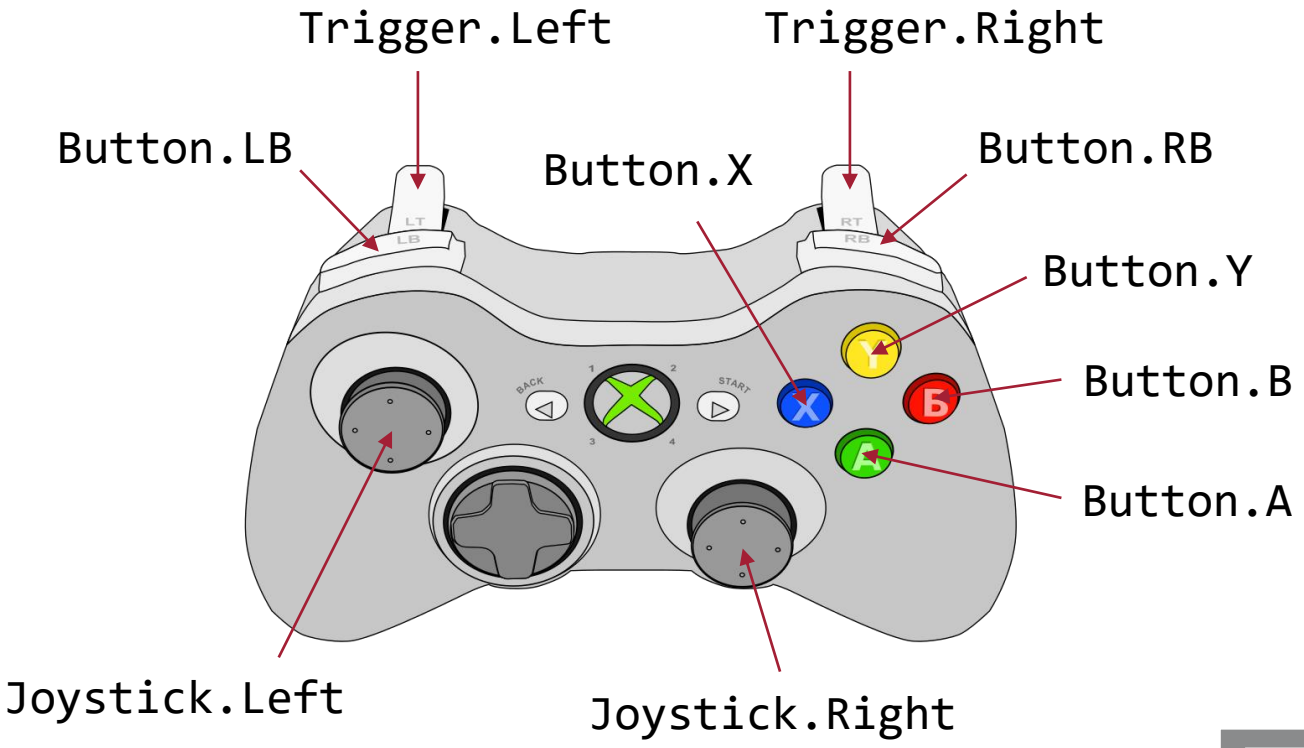
Key Modules – Drive Example

- This update function will cause the car to drive forward for 1 second, drive to the left for 1 second, and then stop

```
def update():
    global counter
    if counter < 1:
        # Drive forward at full speed
        rc.drive.set_speed_angle(1, 0)
    elif counter < 2:
        # Drive to the left at full speed
        rc.drive.set_speed_angle(1, -0.5)
    else:
        rc.drive.stop()
    counter += rc.get_delta_time()
```

Key Modules – Controller

- The **Controller** module allows us to detect input from the Xbox controller
- Public interface
  - `is_down(Button)`
  - `was_pressed(Button)`
  - `was_released(Button)`
  - `get_trigger(Trigger)`
  - `get_joystick(Joystick)`



Key Modules – Controller Example

- This update function uses input from the controller to decide how to move and when to print a message to the command line

```
def update():
    # Enter this if block if RB is currently pressed
    if rc.controller.is_down(rc.controller.Button.RB):
        # Read the value of the right trigger to determine speed
        speed = rc.controller.get_trigger(rc.controller.Trigger.RIGHT)
        rc.drive.set_speed_angle(speed, 0)

    # Print a message every time B is pressed
    if rc.controller.was_pressed(rc.controller.Button.B):
        print("The B button was pressed")
```

Good luck and have fun!

Let us know if you need any help during the lab