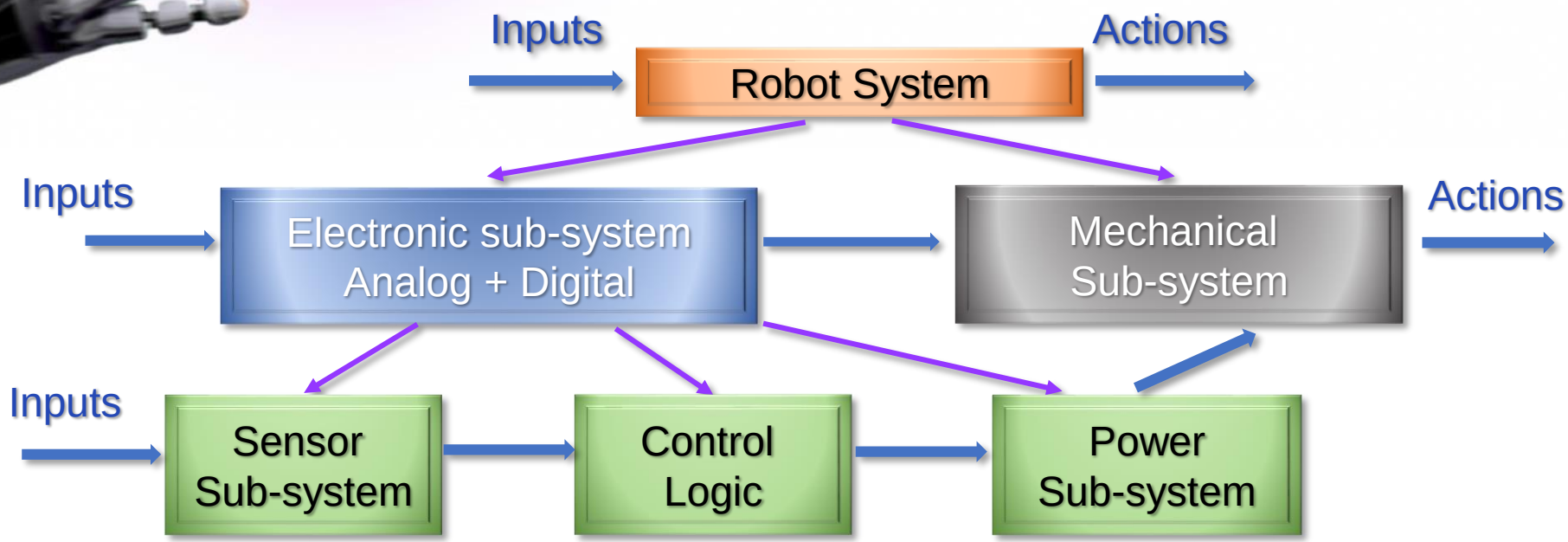
ELEC1100: Introduction to Electro-Robot Design

Lecture 16: Final Project

SONG Shenghui and MURCH Ross, Dept. of ECE, HKUST

ELEC1100 ROADMAP



Sensor Basics:
- Wk6: Sensor Basic – Sensor/Line/ADC

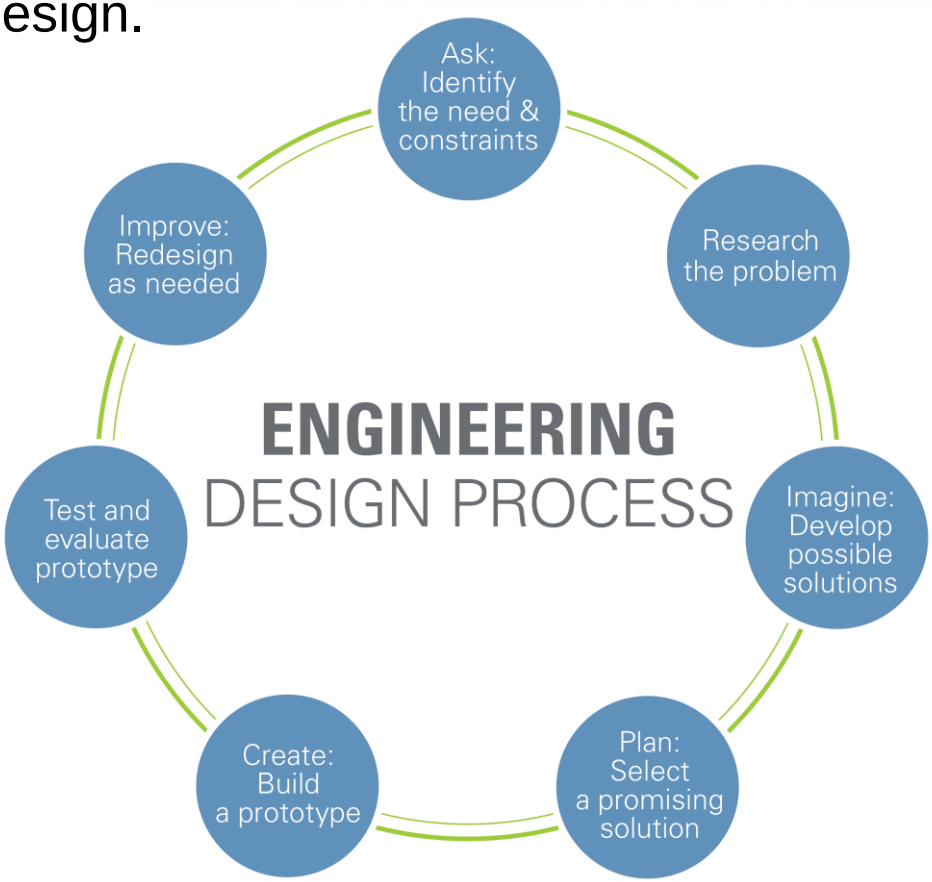
Combinational/Sequential Logic:
- Wk7: Robot Brain: Logic Gate and Logic Operation
- Wk8: MCU & Arduino
- Wk8-9: Programming Language

Basic electronics:
- Wk1: Basic Electronics - Charge/Current/Voltage/Resistor
- Wk2: Energy/Power and DC Sources

Motor Power Supply:
- Wk3: Pulse Signal and PWM Control
- Wk4: Transistor and H-Bridge

ENGINEERING DESIGN PROCESS

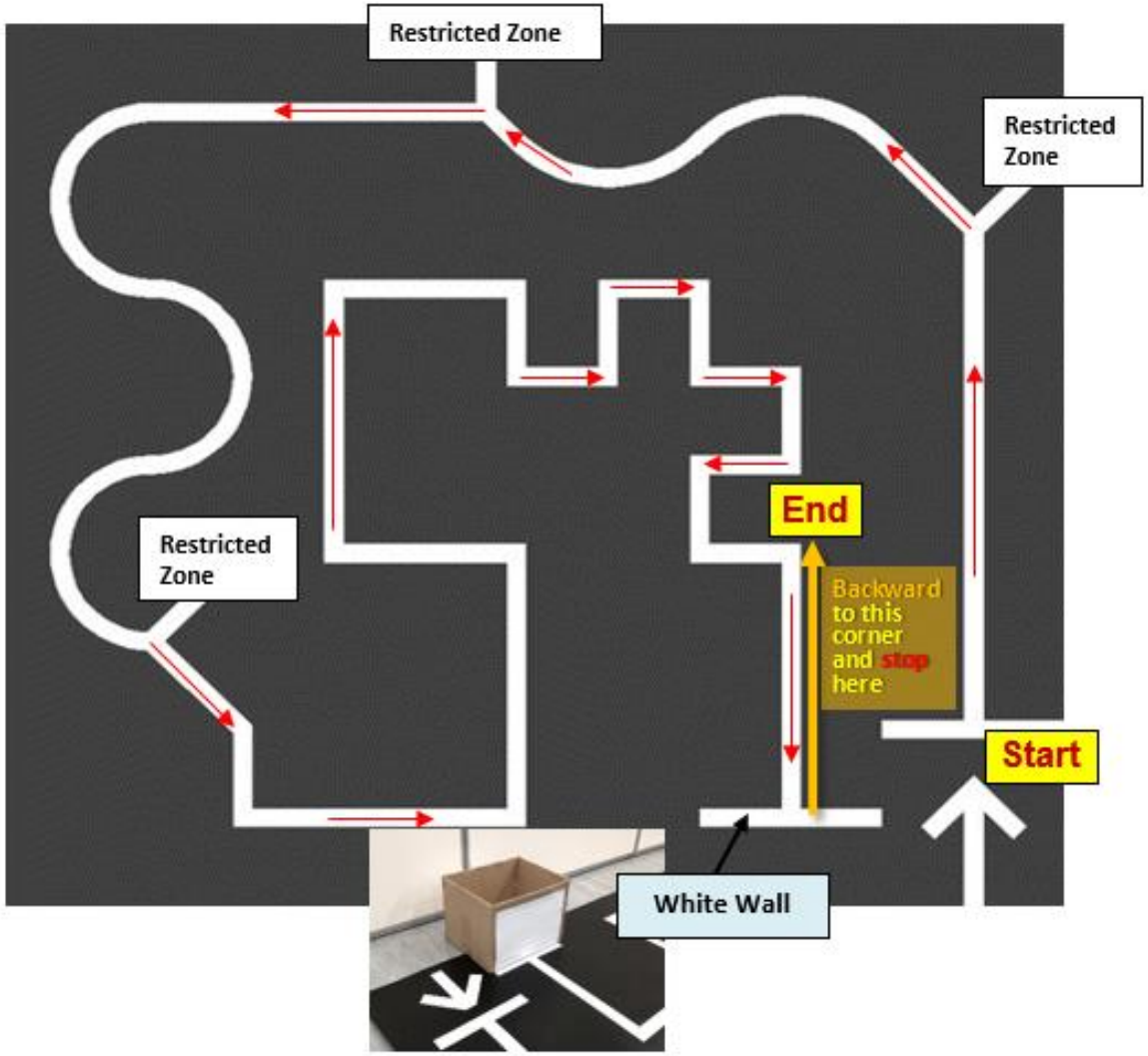
- Our project is not that complex, but it gives a good opportunity to learn engineering design.



ENGINEERING DESIGN PROCESS

- Read carefully about the project requirement.
- Think about possible solutions.
- Pick one and build a prototype.
- Test and evaluate.
- Improve and re-design as needed.

YOUR FINAL PROJECT



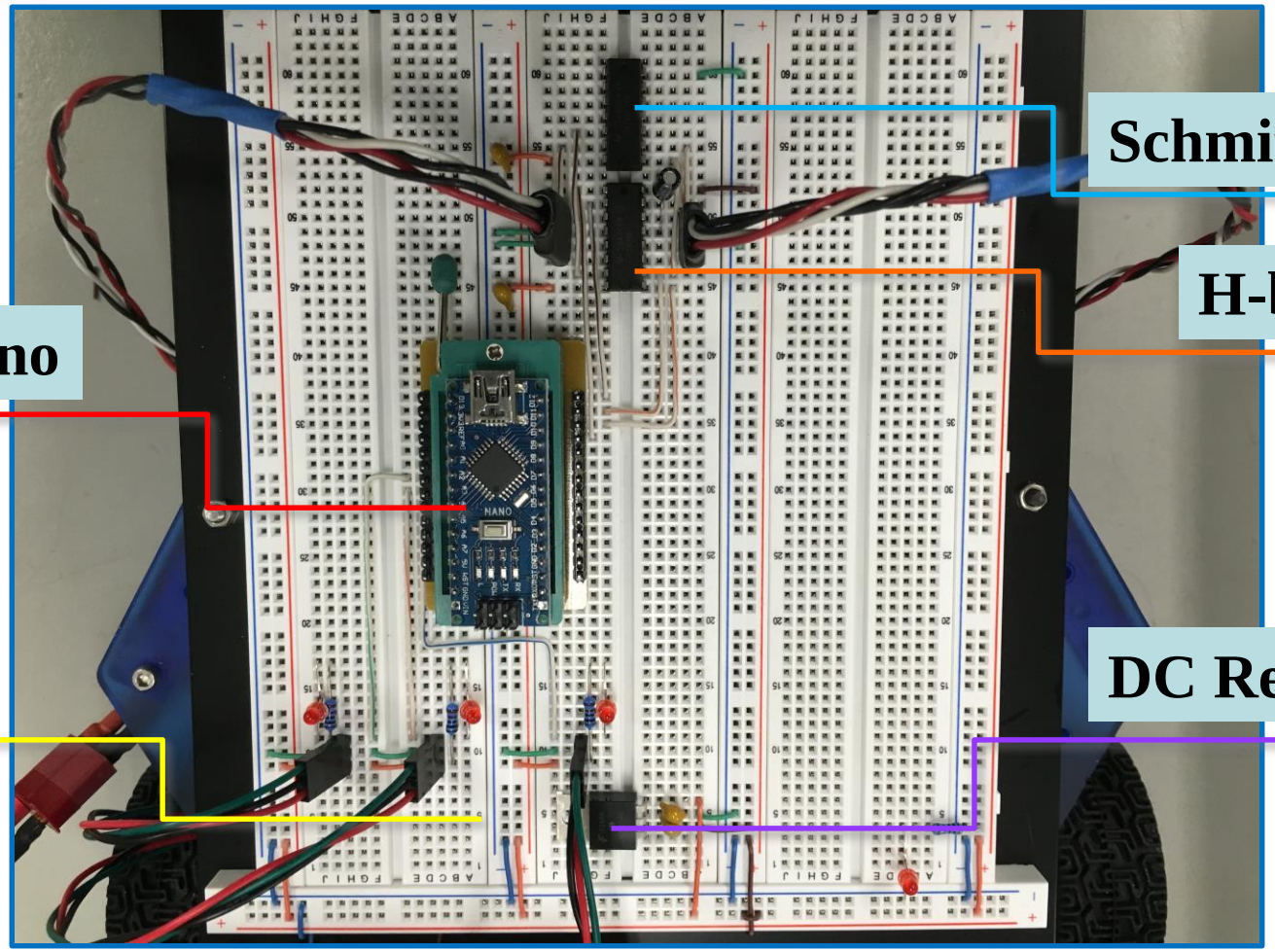
PROJECT ARRANGEMENT

- To allow the project to be completed in an online mode using Tinkercad, but provide a real hardware experience, we will download and run your Tinkercad code on a real robot car and track to determine how well your system can operate.
- At test & demo lab sessions (through ZOOM meeting), your TA will download your Arduino code from Canvas, upload to the robot car, and start the trials on the demo mat at physical lab for grading.

- Supporting document on your Canvas Lab (LA1/LA2/LA3) page:
  - Project Guide_2020s
  - Project Report Format_2020s

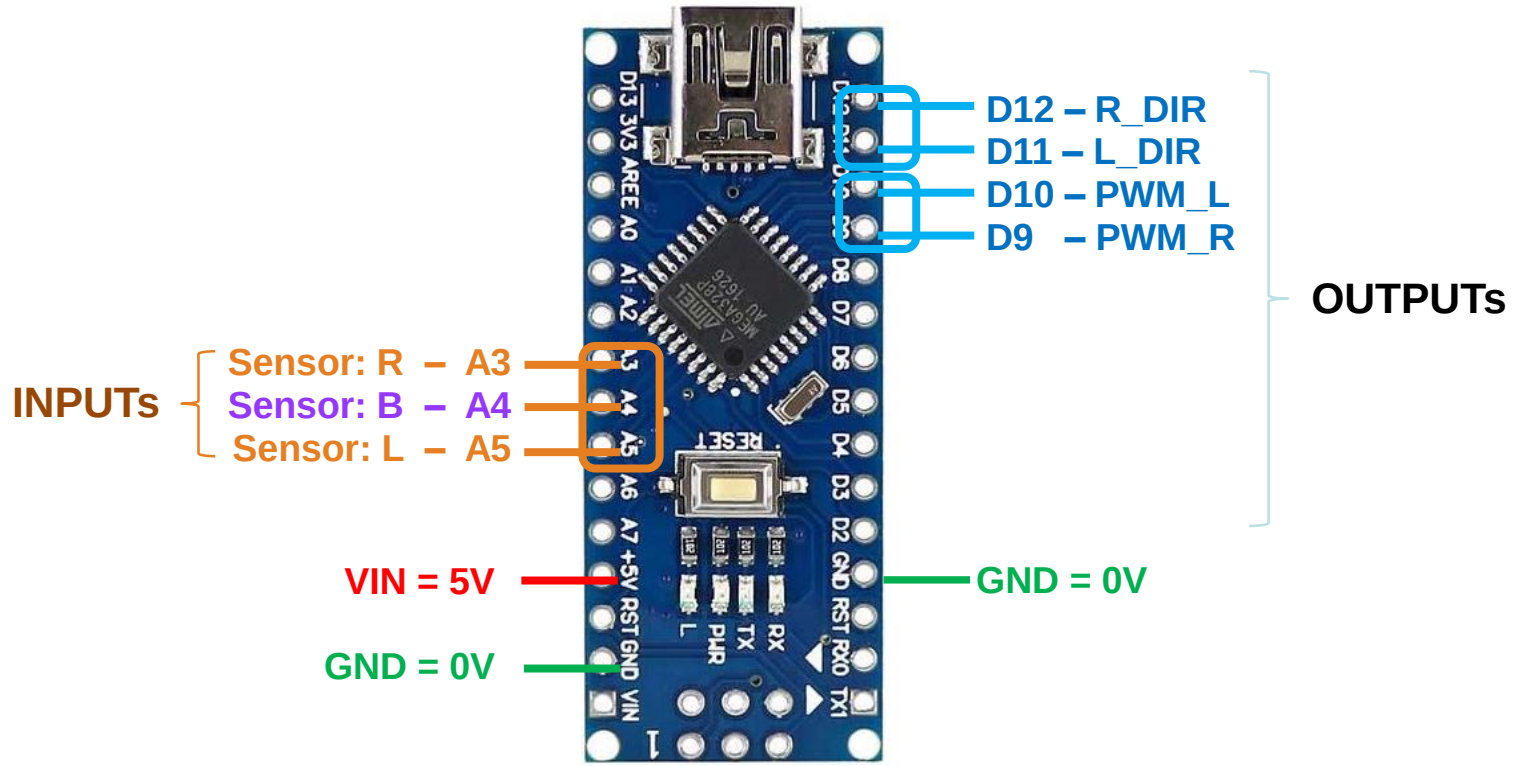
BREADBOARD LAYOUT

- The robot car will be provided by the teaching team



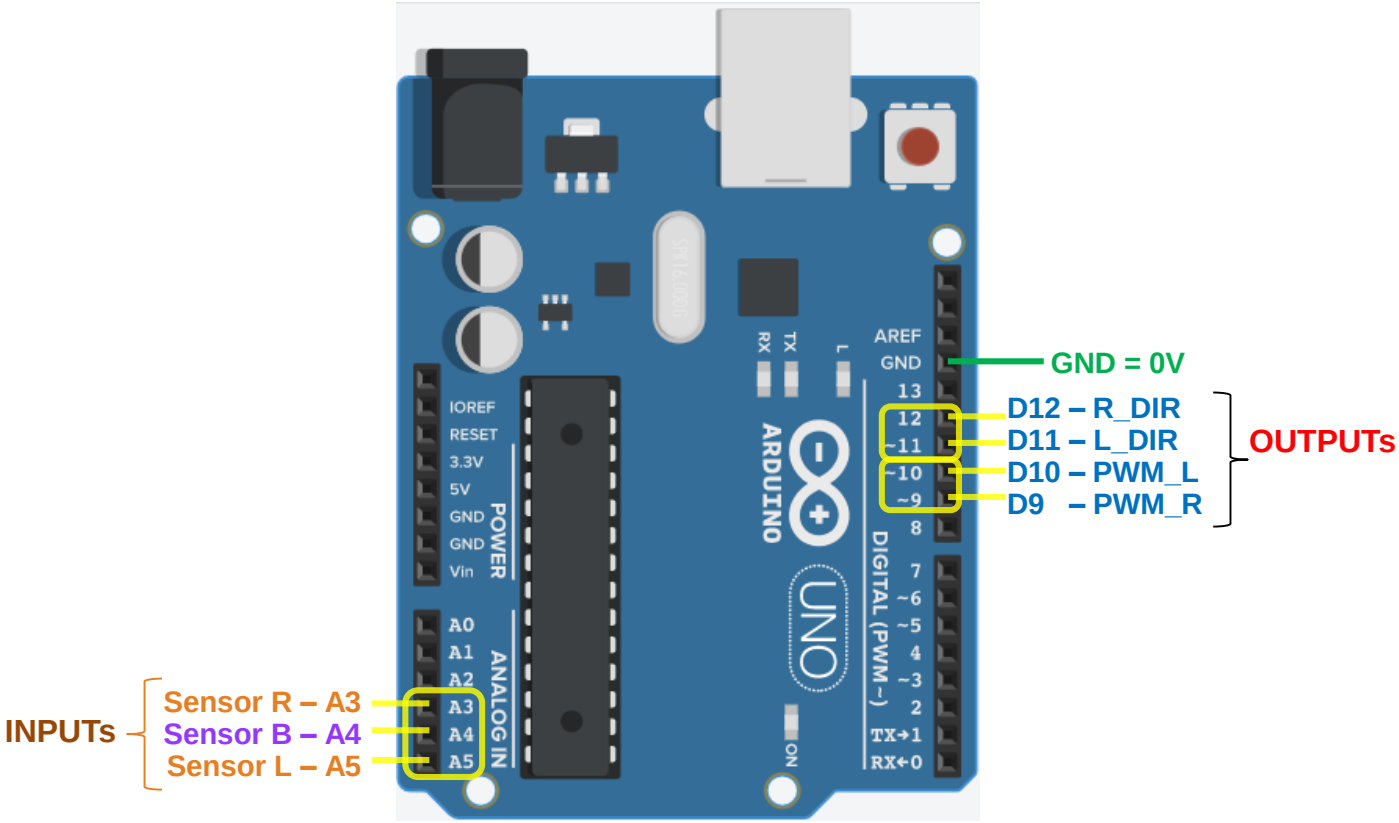
ARDUINO I/O PORTS ASSIGNMENT

- Nano-Board connection on the breadboard



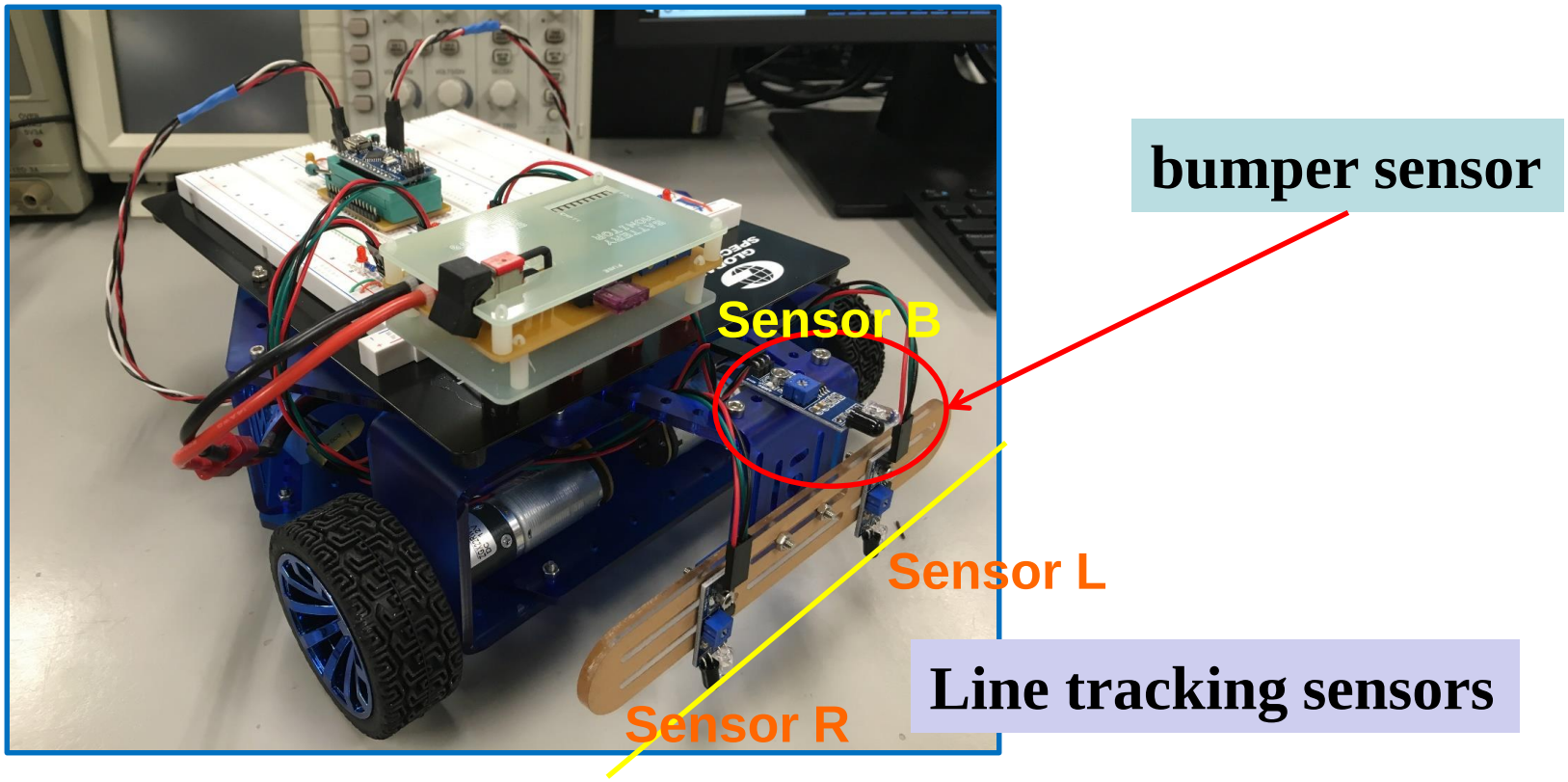
ARDUINO I/O PORTS ASSIGNMENT

- Uno-Board in your Tinkercad simulation model



YOUR ROBOT CAR SENSORS

- Line tracking sensors: 2 sensors in a line
- Bumper sensor: to be triggered at the start point & to sense the white wall

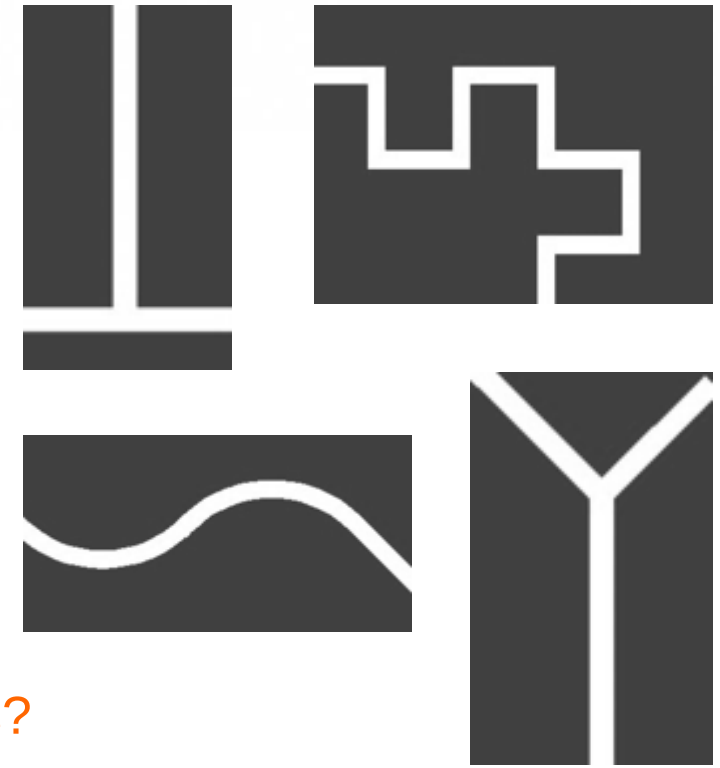


TRACK DEMO

- A “Perfect Run” within each trial:
  - After power up, the car wheels should not be started yet. The robot car is put at the START position such that the line tracking sensors are on the horizontal “Start” white line.
  - By using a white paper to trigger the bumper sensor, the car should start to run.
  - Once the car starts running, it tracks the white course, and navigates through all of the splits and turnings.
  - When the car arrives at the front of the white wall, the bumper sensor shall be triggered and the car go backward until it reaches the “End” corner and stop there (ensure the bar of line tracking sensors pass the horizontal line of the corner).

SOFTWARE DESIGN

- Things we cannot change
  - setup()
  - loop ()
- Freedoms we have:
  - Everything inside the loop()
  - The map has several sections, so



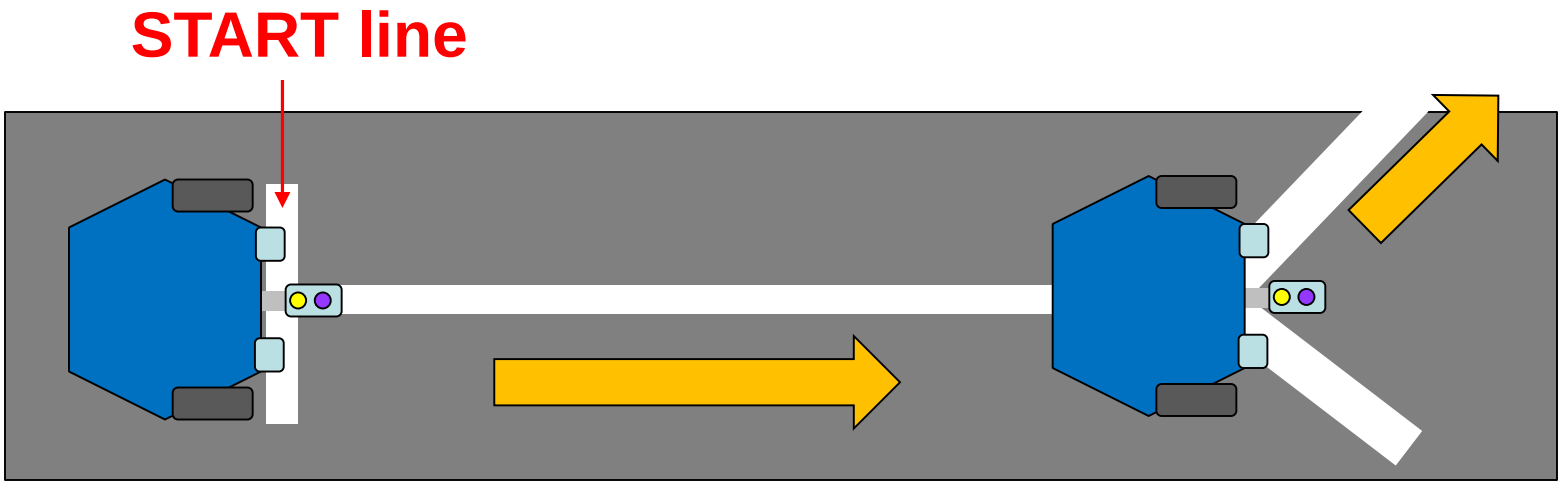
Should I use the same speed for all sections?

Should I use the same logic for all sections?

Are there any operations I should functionalize?

START TO RUN

- Use a white paper to trigger the bumper sensor, the car starts to run
  - How to guarantee the car wheels not start yet after power up?
  - How to guarantee the car to move forward tracking the white line?
  - What if the car is not moving straightly?
  - How to guarantee the car to sense the split and turn to correct direction?



SEQUENTIAL LOGIC

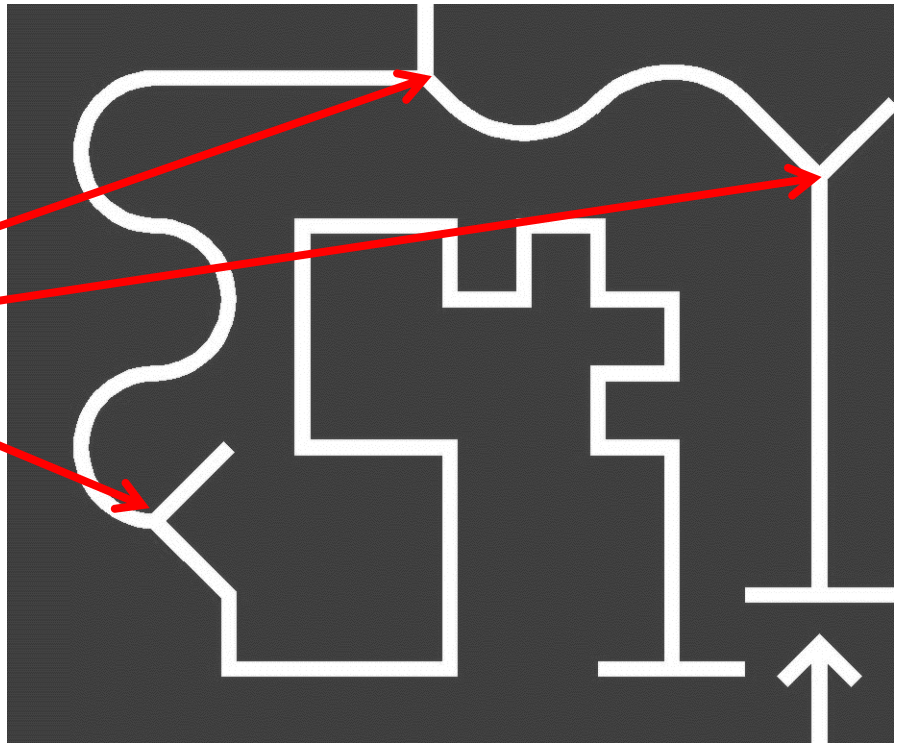
- Navigate the splits

- What are the values of Sensors L & R at each split?

- Does the combinational logic used in lab#06 works all the time for the project line tracking?

Which way?

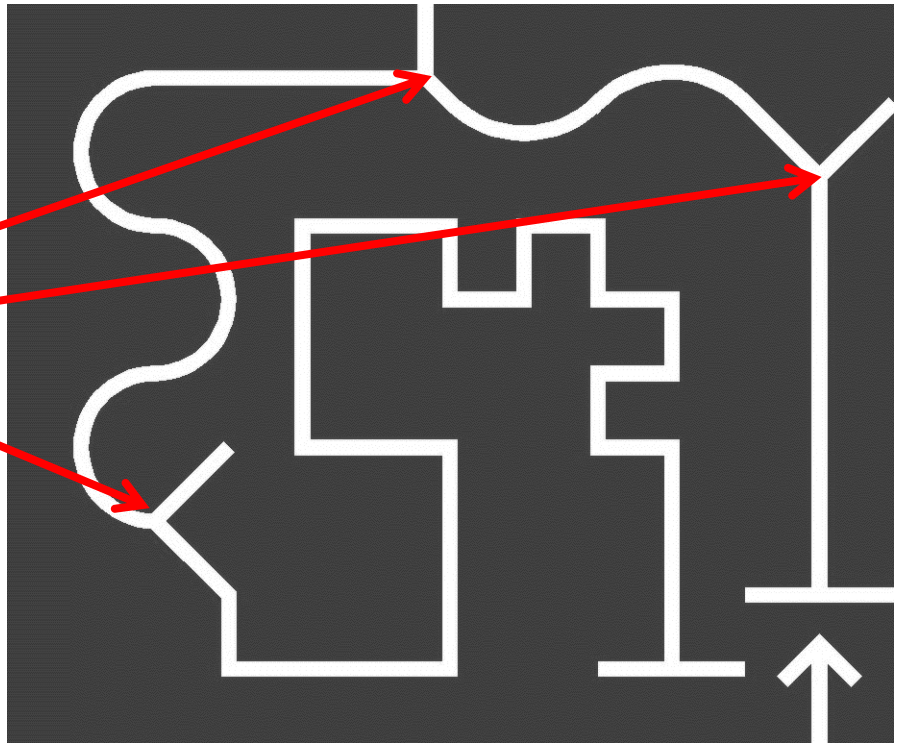
- How could your car sensors differentiate those three splits?



CONDITIONAL EXECUTION CODE

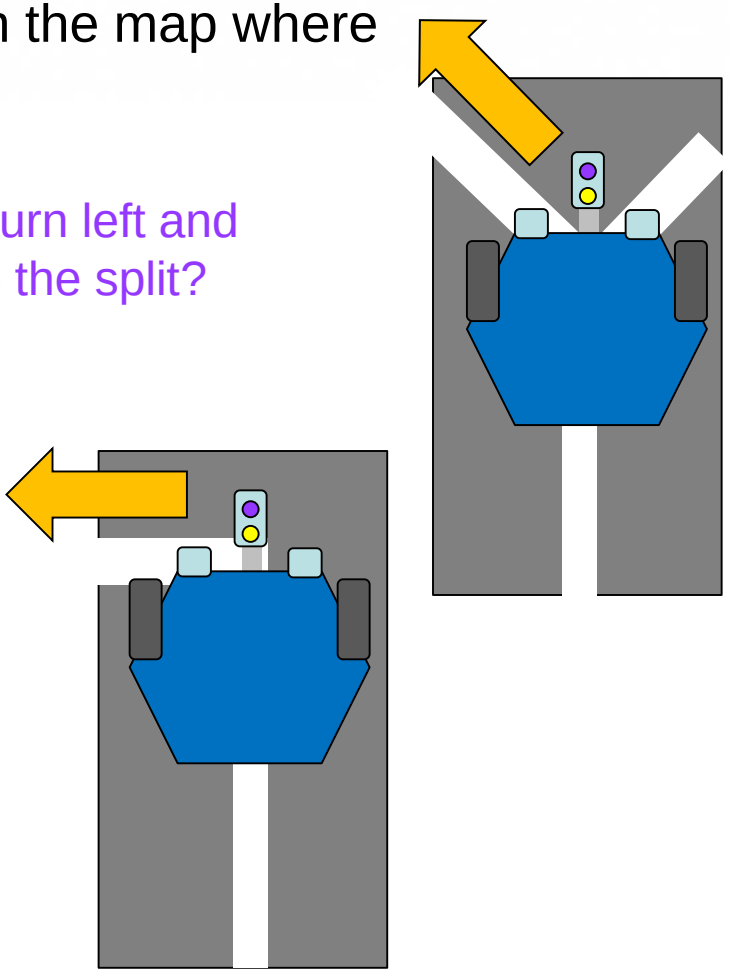
- if … else …
- for
- while
- switch
- break

Which way?



SPLITS AND TURNINGS

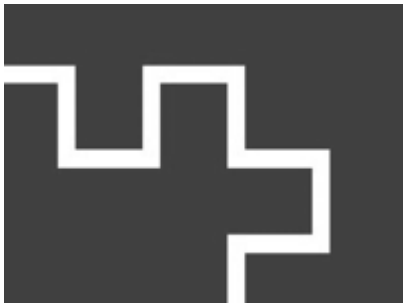
- There are several splits and turnings on the map where the car should turn.
  - How can we guarantee that the car will turn left and leave the split, i.e., not bouncing back to the split?
  - For the right angle turn, how can we guarantee that the car has sufficient time to turn before running out of the track?



SPEED CONTROL

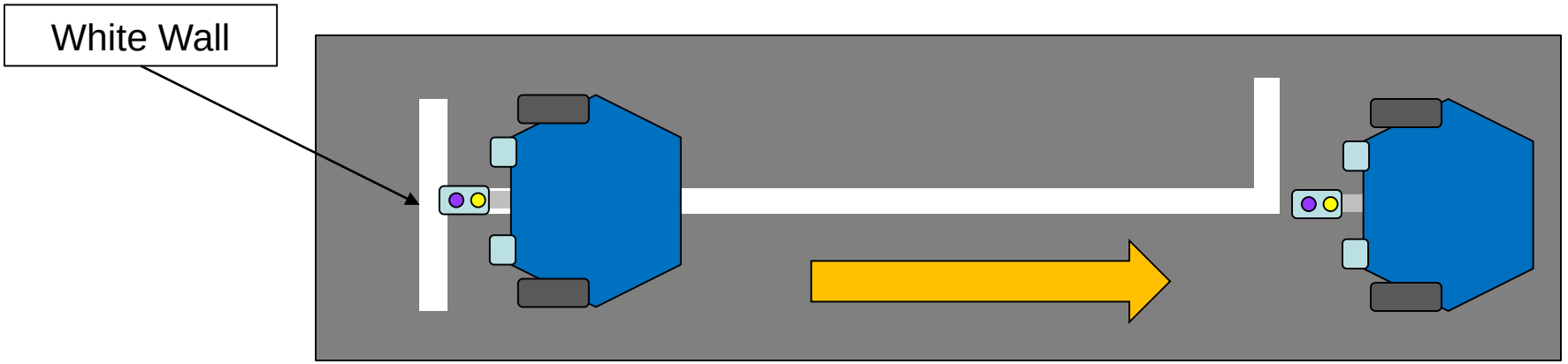
- Your Arduino code should send logic outputs through **PWM_L** and **PWM_R** to control the speeds of motors.

- The use of different speeds (optional).
  - You may need to use different speeds within the same trial so that your car can smoothly (with a lower speed) go through the gentle curves or the right angle curves.
  - This may require additional code section within your Arduino code to change **the duty cycle value** in function analogwrite () automatically at a particular task point.



BACKWARD

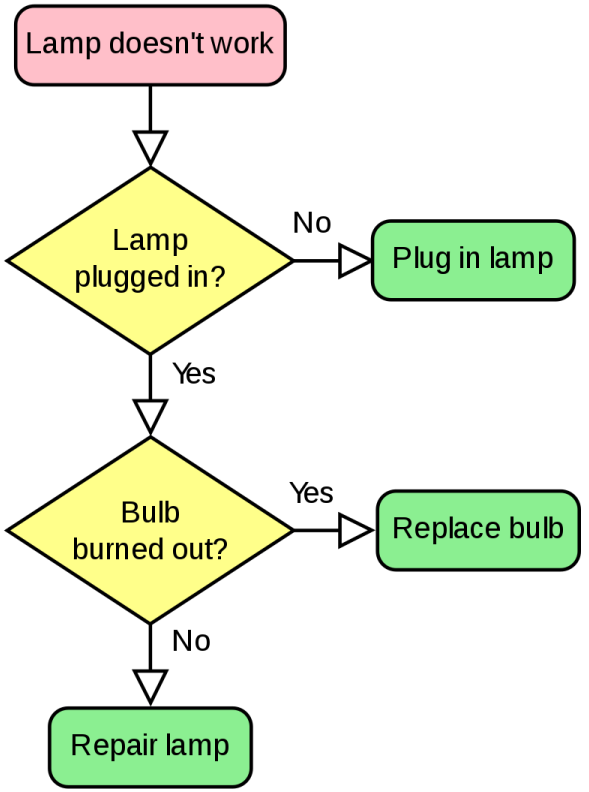
- The car needs to go backward after the bumper sensor triggered at the "white wall".
  - How to guarantee the car to move backward?
  - How to guarantee the car to move backward until the required corner?
  - How to guarantee the car to sense the corner and stop there?
  - What if the car is not moving straightly?



LOGIC FLOW CHART

- A **flowchart** is a type of diagram that represents a workflow or process, a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

  - The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.
  - You will need to draw your **project logic flow** chart and include it in your project report.



NEXT LECTURE

- Online Exam Review
  - Exam arrangement
  - Past paper: example questions

QUESTIONS?