Soirée Pratique
Build your own robot

Session 7: Brains (Programming)

slides + extra information:
www.ieee-sb-leuven.be/programming2017
www.ieee-sb-leuven.be/soireepratique

IEEE Student Branch Leuven

Schedule sumo robot competition

- First semester: Arduino, sensors, motors, frame

- Monday 27/2 at 19h30:
  - integration session: frame + motors + sensors combined
- Monday 6/3 at 19h30:
  - building + testing session
- Monday 20/3 at 19h30:
  - programming session: the “brains” revisited
- Monday 27/3 at 19h30:
  - building + testing session
- (Easter break)
- Monday 24/4:
  - final building + testing session
- Wednesday 26/4: Competition Leuven
- (Wednesday 3/5: Competition Gent)

Rules of competition

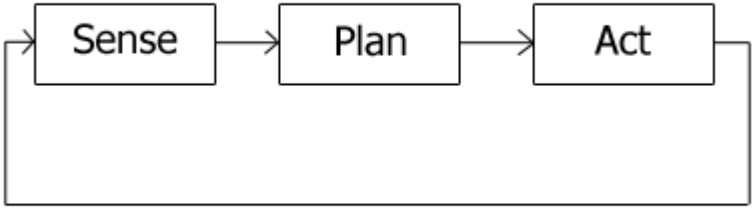
- rules:
  - http://www.ieee-sb-leuven.be/sites/default/files/eventfiles/Reglement_Sumo_Robot_Competition.pdf
- specifications:
  - **robot** should fit at the start of the round within a box of following dimensions:
    - width: 24cm
    - length: 24cm
    - height: unlimited
    - weight: less than 1.5kg
  - **field** (doyho)
    - wooden circular plate of about 1.5m diameter
    - surface is black with a white border
    - border width: about 8cm wide

Competition course

1. robots are put in two opposing quadrants on dohyo with at least 20cm distance between them
2. robots are powered
3. the **5 second countdown procedure** is activated on both robots at the same time (by releasing button). The countdown is **indicated by means of leds**.
4. everyone steps back from dohyo
5. robots start moving after 5 second countdown procedure
6. robots have **2 minutes to push contestant out of the dohyo**
7. round is over after one of the robots is pushed out or if 2 minutes are over (draw)
8. winner gets +3 points, loser +0 points, draw +1 point

Programming a robot

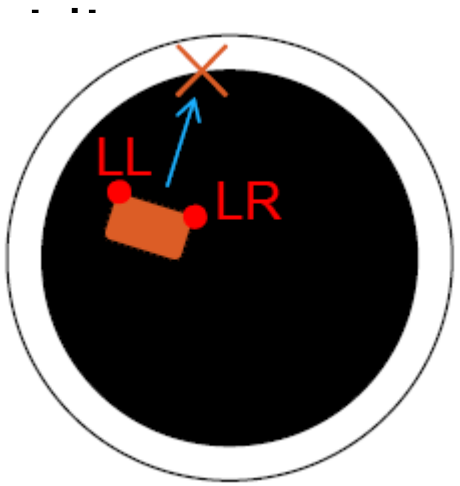
- Basic sensor-based robot control:
  - sense-plan-act framework



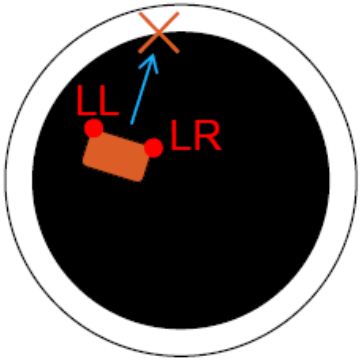
  - **sense** the environment with sensors
  - **plan** the next course of action
  - **act** according to the plan
  - **iterate** to be reactive to changes

Example: line detection + avoidance

- See code template
- drive forward on the Dohyo while av
  white border
  - sensors
    - line detection sensor on left: LL
    - line detection sensor on right: LR
  - if LL detects line
    - turn maneuver to the right for 1 sec
  - if LR detects line
    - turn maneuver to the left for 1 sec



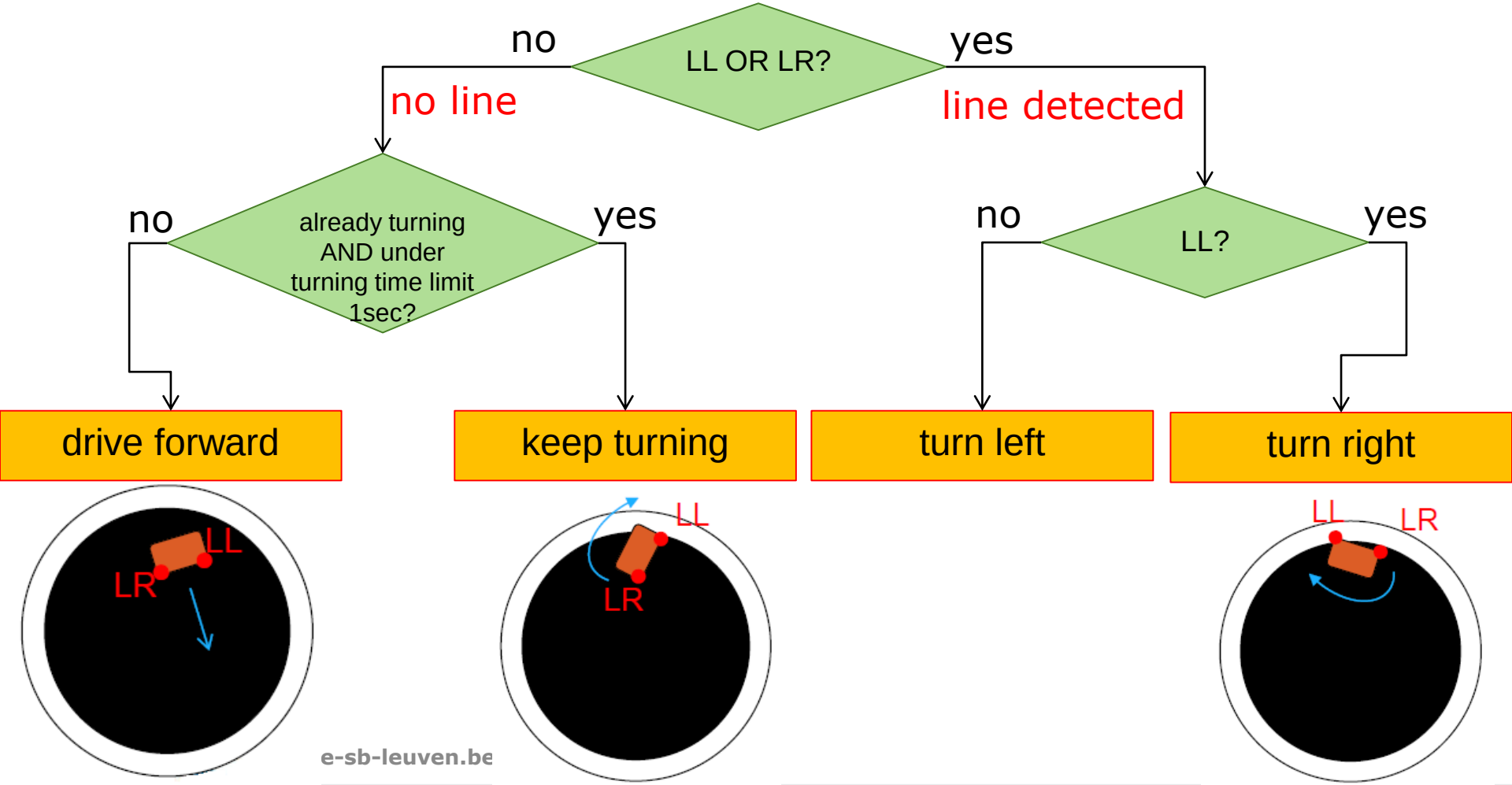
Example: line detection + avoidance



- *sense:*
  - read line detection sensors LL and LR
- *plan:*
  - determine action to perform
    - e.g. by using a **decision tree** (next slide)
- *act:*
  - actions can be seen as discrete states in a **finite state machine**
  - transition to new state determined by decision tree outcome
  - possible states here:
    - move forward
    - turn maneuver right
    - turn maneuver left

Example: line detection + avoidance

- *plan* step using a **decision tree**:



Example: line detection + avoidance

- implementation in Arduino (see code attachment)

sense

```
void loop() {
  /*<...>*/
    /*Line sensors
    ********************************/
    LL=digitalRead(LL_pin);
    LR=digitalRead(LR_pin);
  /*<...>*/
```

plan

```
    if (LL || LR) { // line detected
        lineDetected = 1;
        if (LL){
          state = 21;
          turn_manoever_left = 0;
        }
        else {
          state = 22;
          turn_manoever_left = 1;
        }
    }
    else {  // no line detected
        if ((state_duration < turn_maneuver_time_limit) && lineDetected) {
          if (turn_manoever_left) {
            state = 22;
          }
          else {
            state = 21;
          }
        }
        else {
          state = 11;
          lineDetected = 0;
        }
    }
  /*<...>*/
```

act

```
switch(state) {
  /*******************************************/
                /* NORMAL
  *******************************************/
  // DRIVE FORWARD
  case 11:
      digitalWrite(motLforward, slow_speed);
      digitalWrite(motRforward, slow_speed);
      digitalWrite(motLbackward, 0);
      digitalWrite(motRbackward, 0);
  break;

  /*********************************/
          /* TURN MANEUVERS
  *********************************/
  // Turn right
  case 21:
      digitalWrite(motLforward, slow_speed);
      digitalWrite(motRforward, 0);
      digitalWrite(motLbackward, 0);
      digitalWrite(motRbackward, slow_speed);

    break;
  // Turn left
  case 22:
      digitalWrite(motLforward, 0);
      digitalWrite(motRforward, slow_speed);
      digitalWrite(motLbackward, slow_speed);
      digitalWrite(motRbackward, 0);
    break;
 }
 delay(1); //1ms delay in loop to allow state to be executed
}
```

Programming sumo robot

- Seems complicated? No worries!

Check **example code template** for line detection and border avoidance can be found here using the standard frame

- framework is extendable with other sensors:
  - long range distance sensor
  - ultrasonic sensor
  - ...

- implement different behaviors:
  - border avoidance strategy
  - search opponent strategy
  - attack strategy

Next Soiree Pratique

- Monday 27/3 at 19h30:
  - building + training session