Elective 2 – Robotics Technology

LABORATORY ACTIVITY 1

Virtual Robotics Simulation

SY 2024-2025, 2nd Semester

Name:	**PAGATPAT, Reymark M.**	**Date:**	**02/25/2025**
Course & Year:	BSECE 4A		

1. What are the key components of the robot in the project?

➢ The simulated virtual robot in this project is built upon several key components. Its structure and body consist of a robust frame that acts as the skeleton, along with links and joints—both rotary and prismatic—that form its limbs and enable diverse movements. A range of sensors (including position sensors, force/torque sensors, and proximity sensors) supplies vital real-time feedback, ensuring that the robot can accurately monitor its joint positions and interact safely with its environment. Complementing the hardware is a sophisticated control system comprised of a processor, memory, and specialized software that orchestrates the robot's behaviors, such as transitioning smoothly between movements using the movement_decomposition() function. A reliable power system underpins the entire operation, delivering energy through a dedicated power supply and energy storage mechanisms.

2. How are components inter-related?

➢ These components are intricately inter-related. The robot's physical structure provides the mechanical foundation that the joints and links utilize to create motion. Sensors embedded within the structure continuously feed data into the control system, which processes this information to execute precise movements. For instance, the movement_decomposition() function ensures fluid transitions between joint positions, while the walk_move() function simulates a natural two-step gait repeated over three cycles. In contrast, the inspection_move() routine employs coordinated movements—crouching, oscillating the elbows to simulate a "sniff," and rotating the shoulders to "look"—before returning to a neutral stance. The main loop of the program alternates between walking and inspecting routines, demonstrating a highly coordinated interplay between hardware components and software commands.

3. In your opinion, explain where could be this kind of robot can be used for?

➢ In my opinion, this type of robot, capable of mimicking lifelike exploratory behavior, has a broad range of potential applications. Its ability to navigate and inspect environments with nuanced, human-like movements makes it ideal for educational demonstrations and research in robotics. Additionally, such a robot could be utilized in industrial inspection tasks or interactive exhibits, where dynamic movement and precise control are critical. The simulation of realistic walking and inspection routines not only showcases the robot's technical capabilities but also highlights its potential utility in environments where adaptability and subtle movement are paramount.

4. The program you used with comments on the instruction you edited or added.

```
/*
 * Spot Robot - Walk & Inspect Routine
 * The robot walks a few steps, stops to inspect, then resumes walking.
 */

#include <webots/camera.h>
#include <webots/device.h>
#include <webots/led.h>
```

```
#include <webots/motor.h>
#include <webots/robot.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#define NUMBER_OF_LEDS 8
#define NUMBER_OF_JOINTS 12
#define NUMBER_OF_CAMERAS 5

// Devices: motors, cameras, LEDs.
static WbDeviceTag motors[NUMBER_OF_JOINTS];
static const char *motor_names[NUMBER_OF_JOINTS] = {
  "front left shoulder abduction motor", "front left shoulder rotation motor", "front left elbow motor",
  "front right shoulder abduction motor", "front right shoulder rotation motor", "front right elbow motor",
  "rear left shoulder abduction motor", "rear left shoulder rotation motor", "rear left elbow motor",
  "rear right shoulder abduction motor", "rear right shoulder rotation motor", "rear right elbow motor"
};

static WbDeviceTag cameras[NUMBER_OF_CAMERAS];
static const char *camera_names[NUMBER_OF_CAMERAS] = {
  "left head camera", "right head camera", "left flank camera", "right flank camera", "rear camera"
};

static WbDeviceTag leds[NUMBER_OF_LEDS];
static const char *led_names[NUMBER_OF_LEDS] = {
  "left top led", "left middle up led", "left middle down led",
  "left bottom led", "right top led", "right middle up led",
  "right middle down led", "right bottom led"
};

// Take one simulation step.
static void step() {
  const double ts = wb_robot_get_basic_time_step();
  if (wb_robot_step(ts) == -1) {
    wb_robot_cleanup();
    exit(0);
  }
}

// Smoothly move joints to target positions.
static void movement_decomposition(const double *target, double duration) {
  const double ts = wb_robot_get_basic_time_step();
  const int steps = duration * 1000 / ts;
  double diff[NUMBER_OF_JOINTS], current[NUMBER_OF_JOINTS];
  for (int i = 0; i < NUMBER_OF_JOINTS; i++) {
    current[i] = wb_motor_get_target_position(motors[i]);
    diff[i] = (target[i] - current[i]) / steps;
  }
  for (int s = 0; s < steps; s++) {
    for (int i = 0; i < NUMBER_OF_JOINTS; i++) {
      current[i] += diff[i];
      wb_motor_set_position(motors[i], current[i]);
    }
    step();
  }
}

// Inspection: crouch, sniff, look around, then stand.
static void inspection_move() {
  // Crouch: lower front legs.
  const double inspect[NUMBER_OF_JOINTS] = {
    -0.3, -0.5, 0.8,  0.3, -0.5, 0.8,
    -0.1,  0.0, 0.0,  0.1,  0.0, 0.0
  };
```

```
  movement_decomposition(inspect, 2.0);

  // Sniff: oscillate front elbows.
  double startTime = wb_robot_get_time();
  while (wb_robot_get_time() - startTime < 3.0) {
    double posFL = 0.8 + 0.2 * sin(3 * wb_robot_get_time());
    double posFR = 0.8 + 0.2 * sin(3 * wb_robot_get_time() + M_PI);
    wb_motor_set_position(motors[2], posFL);
    wb_motor_set_position(motors[5], posFR);
    step();
  }

  // Look around: oscillate shoulder rotations.
  startTime = wb_robot_get_time();
  while (wb_robot_get_time() - startTime < 2.0) {
    double posL = 0.0 + 0.2 * sin(2 * wb_robot_get_time());
    double posR = 0.0 + 0.2 * sin(2 * wb_robot_get_time());
    wb_motor_set_position(motors[1], posL);
    wb_motor_set_position(motors[4], posR);
    step();
  }

  // Return to neutral.
  const double neutral[NUMBER_OF_JOINTS] = {
    -0.1, 0.0, 0.0,  0.1, 0.0, 0.0,
    -0.1, 0.0, 0.0,  0.1, 0.0, 0.0
  };
  movement_decomposition(neutral, 1.5);
}

// Walk: take a few steps.
static void walk_move() {
  // Neutral stance.
  const double neutral[NUMBER_OF_JOINTS] = {
    -0.1, 0.0, 0.0,  0.1, 0.0, 0.0,
    -0.1, 0.0, 0.0,  0.1, 0.0, 0.0
  };

  // Step A: left front & right rear forward; right front & left rear back.
  const double stepA[NUMBER_OF_JOINTS] = {
    -0.1,  0.2, 0.0,   0.1, -0.2, 0.0,
    -0.1, -0.2, 0.0,   0.1,  0.2, 0.0
  };

  // Step B: reverse of step A.
  const double stepB[NUMBER_OF_JOINTS] = {
    -0.1, -0.2, 0.0,   0.1,  0.2, 0.0,
    -0.1,  0.2, 0.0,   0.1, -0.2, 0.0
  };

  for (int i = 0; i < 3; i++) {
    movement_decomposition(stepA, 0.8);
    movement_decomposition(stepB, 0.8);
  }
  movement_decomposition(neutral, 0.8);
}

int main(int argc, char **argv) {
  wb_robot_init();
  const double ts = wb_robot_get_basic_time_step();

  // Init cameras.
  for (int i = 0; i < NUMBER_OF_CAMERAS; i++)
    cameras[i] = wb_robot_get_device(camera_names[i]);
  wb_camera_enable(cameras[0], 2 * ts);
```

```
  wb_camera_enable(cameras[1], 2 * ts);

  // Init LEDs.
  for (int i = 0; i < NUMBER_OF_LEDS; i++) {
    leds[i] = wb_robot_get_device(led_names[i]);
    wb_led_set(leds[i], 1);
  }

  // Init motors.
  for (int i = 0; i < NUMBER_OF_JOINTS; i++)
    motors[i] = wb_robot_get_device(motor_names[i]);

  // Loop: walk then inspect.
  while (true) {
    walk_move();
    inspection_move();
  }

  wb_robot_cleanup();
  return EXIT_FAILURE;
}
```