Elective 2 – Robotics Technology
SY 2024-2025, 2nd Semester

LABORATORY ACTIVITY 1
Virtual Robotics Simulation

Submitted to:

Engr. Rojay A. Flores

Elective 2 Instructor

Submitted by:

Rey Vincent Q. Conde

BSECE – 4

KEY COMPONENTS OF THE ROBOT

- **Actuators** – Motors control the robot's joints for movement.
- **Motion Control System** – Controls smooth and natural robot movements.
- **Webot Simulation** – Provides function to control the robot in simulation.

HOW ARE THE COMPONENTS OF THE ROBOT INTER-RELATED?

- **Actuators & Motion Control System** – The motors (actuators) move the robot's joints, while the Motion Control System ensures smooth and precise movement by gradually adjusting motor positions.
- **Motion Control System & Webots Simulation** – The Motion Control System relies on Webots functions to update motor positions and advance the simulation.
- **Actuators & Webots Simulation** – Webots provides commands to control the motors, ensuring they execute movements correctly in the simulation environment.

IN YOUR OPINION, EXPLAIN WHERE COULD BE THIS KIND OF ROBOT CAN BE USED FOR?

I believe that this robot can be utilized for military purposes like snooping and transporting supplies, as well as for search and rescue operations to locate individuals in hazardous environments. It can also examine for issues in difficult-to-reach places in manufacturing. It can be used by farmers to monitor their livestock and crops. Additionally, it can be a comforting robotic pet or assist those with disabilities. It is capable of walking over uneven terrain to gather data and analyze worlds in space exploration.

THE PROGRAM USED WITH COMMENTS ON THE INSTRUCTION YOU EDITED OR ADDED.

```
#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

// Initialize the robot's information
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"};


static void step() {
  const double time_step = wb_robot_get_basic_time_step();
  if (wb_robot_step(time_step) == -1) {
    wb_robot_cleanup();
    exit(0);
  }
}


// Movement decomposition
static void movement_decomposition(const double *target, double duration) {
  const double time_step = wb_robot_get_basic_time_step();
  const int n_steps_to_achieve_target = duration * 1000 / time_step;
  double step_difference[NUMBER_OF_JOINTS];
  double current_position[NUMBER_OF_JOINTS];


  for (int i = 0; i < NUMBER_OF_JOINTS; ++i) {
    current_position[i] = wb_motor_get_target_position(motors[i]);
    step_difference[i] = (target[i] - current_position[i]) / n_steps_to_achieve_target;
  }


  for (int i = 0; i < n_steps_to_achieve_target; ++i) {
    for (int j = 0; j < NUMBER_OF_JOINTS; ++j) {
      current_position[j] += step_difference[j];
      wb_motor_set_position(motors[j], current_position[j]);
    }
    step();
  }
}


static void lie_down(double duration) {
  const double motors_target_pos[NUMBER_OF_JOINTS] = {-0.40, -0.99, 1.59,    // Front left leg
                                                       0.40,  -0.99, 1.59,   // Front right leg
                                                       -0.40, -0.99, 1.59,   // Rear left leg
                                                       0.40,  -0.99, 1.59};  // Rear right leg
  movement_decomposition(motors_target_pos, duration);
}


static void stand_up(double duration) {
  const double motors_target_pos[NUMBER_OF_JOINTS] = {-0.1, 0.0, 0.0,   // Front left leg
                                                       0.1,  0.0, 0.0,   // Front right leg
                                                       -0.1, 0.0, 0.0,   // Rear left leg
                                                       0.1,  0.0, 0.0};  // Rear right leg


  movement_decomposition(motors_target_pos, duration);
}


static void sit_down(double duration) {
  const double motors_target_pos[NUMBER_OF_JOINTS] = {-0.20, -0.40, -0.19,   // Front left leg
                                                       0.20,  -0.40, -0.19,  // Front right leg
```

```
                                                    -0.40, -0.90, 1.18,   // Rear left leg
                                                    0.40,  -0.90, 1.18};  // Rear right leg

  movement_decomposition(motors_target_pos, duration);
}

static void give_paw() {
  // Stabilize posture
  const double motors_target_pos_1[NUMBER_OF_JOINTS] = {-0.20, -0.30, 0.05,   //
Front left leg
                                                    0.20,  -0.40, -0.19,  // Front right leg
                                                    -0.40, -0.90, 1.18,   // Rear left leg
                                                    0.49,  -0.90, 0.80};  // Rear right leg

  movement_decomposition(motors_target_pos_1, 4);

  const double initial_time = wb_robot_get_time();
  while (wb_robot_get_time() - initial_time < 8) {
    wb_motor_set_position(motors[4], 0.2 * sin(2 * wb_robot_get_time()) + 0.6);   //
Upperarm movement
    wb_motor_set_position(motors[5], 0.4 * sin(2 * wb_robot_get_time()));         //
Forearm movement
    step();
  }
  // Get back in sitting posture
  const double motors_target_pos_2[NUMBER_OF_JOINTS] = {-0.20, -0.40, -0.19,  //
Front left leg
                                                    0.20,  -0.40, -0.19,  // Front right leg
                                                    -0.40, -0.90, 1.18,   // Rear left leg
                                                    0.40,  -0.90, 1.18};  // Rear right leg
  movement_decomposition(motors_target_pos_2, 4);
}


int main(int argc, char **argv) {
  wb_robot_init();


  const double time_step = wb_robot_get_basic_time_step();




  // Get the motors (joints) and set initial target position to 0
  for (int i = 0; i < NUMBER_OF_JOINTS; ++i)
    motors[i] = wb_robot_get_device(motor_names[i]);

  while (true) {
    lie_down(1.0);
    stand_up(0.1);
    give_paw();
    lie_down(1.0);
    stand_up(1.1);
    give_paw();
    stand_up(2.0);
    lie_down(1.0);
    stand_up(1.0);


  }


  wb_robot_cleanup();
  return EXIT_FAILURE;
```

```
}
```

1. **Removed LEDs and Cameras**
2. **Changed the Default Execution**