**Elective 2 – Robotics Technology** February 20, 2025

SY 2024-2025, 2nd Semester

PIGAO, PORTIA VERA Y. BSECE 4-A

**LABORATORY - ACTIVITY 1**

(Virtual Robotics Simulation)

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

* The key components of the robot dog Spot in the project are:
* **Rotation Motor** – Controls the rotation of joints, allowing movement in different directions.
* **Abduction Motor** – Controls the outward movement of the legs, providing stability.
* **Joints** – Connect the legs and allow flexible movement.
* Controller – The brain of the robot that executes programmed instructions.

2. How are the components interrelated?

* The motors, joints, and controller work together to control the robot’s movement. The **rotation motor** moves the joints, while the **abduction motor** provides stability and flexibility. The **controller** sends signals to the motors, ensuring precise and coordinated actions.

3. Where could this type of robot be used?

* In my opinion, this kind of robotic dog can be used for:
* Entertainment – It can be used in amusement parks, robotic pet exhibitions, and interactive experiences.
* Small-Space Navigation – Suitable for exploring narrow or confined areas where wheeled robots struggle.
* Surveillance & Security – Equipped with cameras, it can be used for remote monitoring, especially in hazardous environments.

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

#include <webots/motor.h>

#include <webots/robot.h>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#define NUMBER\_OF\_JOINTS 12

// Initialize motors

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"

};

// Robot time step function

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);

}

}

// Smooth movement function

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();

}

}

// Crouch (store energy for jump)

static void crouch(double duration) {

const double motors\_target\_pos[NUMBER\_OF\_JOINTS] = {

-0.6, 0.0, 1.2, // Front left leg bent deep

0.6, 0.0, 1.2, // Front right leg bent deep

-0.6, 0.0, 1.2, // Rear left leg bent deep

0.6, 0.0, 1.2 // Rear right leg bent deep

};

movement\_decomposition(motors\_target\_pos, duration);

}

// Jump

static void jump(double duration) {

const double motors\_target\_pos[NUMBER\_OF\_JOINTS] = {

-0.05, 0.0, -0.1, // Front left leg extended

0.05, 0.0, -0.1, // Front right leg extended

-0.05, 0.0, -0.1, // Rear left leg extended

0.05, 0.0, -0.1 // Rear right leg extended

};

movement\_decomposition(motors\_target\_pos, duration);

}

// Small pause in air (simulate airtime)

static void mid\_air(double duration) {

step();

for (double t = 0; t < duration; t += 0.02) {

step();

}

}

// Landing

static void land(double duration) {

const double motors\_target\_pos[NUMBER\_OF\_JOINTS] = {

-0.4, 0.0, 0.6, // Front left slightly bent

0.4, 0.0, 0.6, // Front right slightly bent

-0.4, 0.0, 0.6, // Rear left slightly bent

0.4, 0.0, 0.6 // Rear right slightly bent

};

movement\_decomposition(motors\_target\_pos, duration);

}

// Standing position

static void stand(double duration) {

const double motors\_target\_pos[NUMBER\_OF\_JOINTS] = {

-0.2, 0.0, 0.2, // Front left relaxed

0.2, 0.0, 0.2, // Front right relaxed

-0.2, 0.0, 0.2, // Rear left relaxed

0.2, 0.0, 0.2 // Rear right relaxed

};

movement\_decomposition(motors\_target\_pos, duration);

}

int main(int argc, char \*\*argv) {

wb\_robot\_init();

// Get motors

for (int i = 0; i < NUMBER\_OF\_JOINTS; ++i)

motors[i] = wb\_robot\_get\_device(motor\_names[i]);

// Main loop (continuous jumping)

while (wb\_robot\_step(wb\_robot\_get\_basic\_time\_step()) != -1) {

crouch(0.8); // Prepare

jump(0.3); // Jump!

mid\_air(0.2); // Pause in air

land(0.8); // Absorb landing

stand(1.0); // Return to neutral

}

wb\_robot\_cleanup();

return EXIT\_FAILURE;

}