Robotics Laboratory 1

ROBOTICS AND EMBEDDED SYSTEM

Seth Paul C. Babayson
BSECE-4A
College of Engineering
Samar State University

***Abstract*—This experiment focuses on simulating and improving the walking motion of a robot using the Webots simulator. The robot, based on the Spot model, was programmed using C to move more smoothly by adjusting its joint angles. Key components like position sensors and camera sensors were used to monitor and guide its movement. By replacing the original lie_down() function with a custom walk() function using trigonometric functions, the robot was able to move in a more natural, animal-like way. The simulation helped students understand how to control robot motion using sensor feedback and math functions like sine for smoother transitions. This hands-on activity supports learning in embedded systems and autonomous robotics by combining theory and practical application.**

I. Rationale

With the growing use of robotics in industries such as manufacturing, logistics, and even healthcare, it is essential for electronics engineering students to develop practical skills in designing and controlling robotic systems. This laboratory experiment introduces students to the basic principles of obstacle avoidance using an Arduino-based robot. The robot uses an ultrasonic sensor and a servo motor to detect obstacles and decide its movement, mimicking how real autonomous robots navigate their environment.

This activity bridges the gap between classroom theory and practical application by allowing students to build, program, and test their robot both in real life and through simulation using Webots. The simulation environment helps students visualize how the robot reacts to different obstacles, which reinforces learning and promotes experimentation without the risk of damaging physical components.

By engaging with core topics like sensor integration, motor control, and real-time decision-making, students gain a strong foundation in embedded systems and autonomous robotics. Previous research emphasizes the importance of project-based learning in robotics to improve student understanding and motivation (Benitti, 2012; Siciliano & Khatib, 2016). This experiment directly addresses this need by providing a hands-on, problem-solving experience that prepares students for more advanced robotics projects.

II. Objectives

A. To simulate robot movement based on the program.

The purpose of the simulation is to adjust the values of the robot's movement to obtain a smooth transition of movements.

Identify applicable funding agency here. If none, delete this.

III. Materials and Software

- Software:
  - Webots Simulator

IV. Procedures

- Download Webots software at the cyberbotics website .
- On the File Menu of the software open the project ”spotẇbt” under the directory of ”boston_dynamics” .
- In the right side of the Webot panel there is a program written in c language which is the current program for the project. Edit the program to modify the movement of the robot.
- Simulate the movement by clicking the play button found on the upper middle of the simulation environment panel.
- Observe how the robot moves based on the program and other key components that have edited and record (video) the movement for documentation.

V. Observations and Data Collection

In this section, the structure and movement of the robot were observed and adjusted during the testing process.

One of the key components of the robot(spot) in the simulation is the body structure, where it has rigid elements that connect its body parts such as the body, head, and legs (rotary joints). It allows for movement abilities, enabling the robot to perform various tasks. The other one is the position sensors that provide information about the robot's joint positions where it allows to obtain control and movement precisions. These sensors are represented as joints in the robot's body. As observed during the simulation, the robot has already built-in camera sensors for the robot's vision. It provides vision information to the robot for object recognition.

VI. Data Analysis

Since the program code is already given in the sample world of the Webots simulator, only the values of the robot's joints were adjusted.

As observed during testing, the robot's movement didn't have smooth transitions of movement when walking.

The walk() function makes the robot walk by repeatedly changing the joint angles of its legs in a wave-like pattern. Each leg moves slightly out of sync with the others to create a walking motion. The robot steps through this movement many times, creating a smooth walking animation.

```c
static void walk(double duration) {
  const double time_step = wb_robot_get_basic_time_step();
  const int n_steps = duration * 1000 / time_step;

  for (int i = 0; i < n_steps; ++i) {
    double phase = 10.0 * M_PI * i / n_steps;

    // Generate slow walking motion by modifying joint positions
    double positions[NUMBER_OF_JOINTS] = {
      -0.2, -0.2 * sin(phase), -0.13 * sin(phase + (3.1416/2)),   // Front left leg
      0.2, 0.2 * sin(phase), 0.13 * sin(phase + (3.1416/2)),  // Front right leg
      -0.2, 0.2 * sin(phase), 0.13 * sin(phase + (3.1416/2)),   // Rear left leg
      0.2, -0.2 * sin(phase), -0.13 * sin(phase + (3.1416/2))   // Rear right leg
    };

    for (int j = 0; j < NUMBER_OF_JOINTS; ++j) {
      wb_motor_set_position(motors[j], positions[j]);
    }
    step();
  }
}
```

Fig. 1. Modified walk() function for smooth transitions of movement(walking)

VII. Discussion and Interpretations

The main objective of this experiment was to simulate the robot's movements by modifying the values of its joints, and modifying movement functions for better transitions of movements.

All of the components are interrelated, as each sensor communicates, depending on the program, to perform certain tasks precisely and automatically. The robot's joints (or the position sensors) correspond with its visual inputs depending on what tasks the robot is programmed to do. In the position sensors alone, it requires a lot of time to determine the movement precision of the robot. If I want the robot to walk smoothly, it must consider parameters such as the speed and torques of each leg, how it balances the body of the robot to prevent falling, and the smoothness of how the robot walks.

The only function that has been removed is the lie_down() function, and it is replaced by a new function which is the walk() function. It shows the ability to walk similarly to a dog. The sequence of movement of the robot is walk, sit, wave, stand, and walk again. In the walk() function, each leg has joints that need to move in a synchronized way to create a proper walking motion. The movement of these joints is controlled using the sine function. The sine function is often used in robotics because it creates smooth, wave-like motions, similar to how animals walk.(See the Webots video simulation at the GitHub link attached in the Appendix)

VIII. Conclusion

This experiment successfully demonstrated how a robot can move like a real animal using joint control and sensor feedback. By adjusting the values in the robot's movement functions—especially the walk() function—we were able to improve how smoothly it walks. The robot's movements depend on precise coordination between its joints and sensors, and using mathematical functions like sine helps create natural, wave-like motions. The project showed how important it is to test and tweak movements carefully to get realistic and balanced motion. Overall, this simulation helped us understand how robots can perform lifelike tasks through careful programming and design.

References

[1] Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. Computers & Education, 58(3), 978–988. https://doi.org/10.1016/j.compedu.2011.10.006

[2] Siciliano, B., & Khatib, O. (Eds.). (2016). Springer Handbook of Robotics (2nd ed.). Springer. https://doi.org/10.1007/978-3-319-32552-1

IX. Appendix

https://github.com/seth-paul/Elective-2-Robotics-Technology/tree/main/Laboratory%201