Advanced Embedded System Design

Dr. Belal H. Sababha, Computer Engineering Dept.

Load Balancing Robot

Done By

Maissam khalill (20228185); mis20228185@std.psut.edu.jo

Omar Nemer (20228116); oma20228116@std.psut.edu.jo

Saad alzoubi (20238108); Saa20238108@std.psut.edu.jo

Abstract:

In this work. we designed a load-balancing robot using the HCS12 microcontroller to control the car's movement. A Bluetooth module receives the movement direction from the user (mobile app), and then the car moves according to the user's instructions. Also, we use an IR sensor to detect obstacles as the car moves. On the other hand, the load will be hung above the robot on a flat surface, and the Arduino will balance the load and keep it stable on the surface under any circumstances using a PID controller. We use an accelerometer to determine the orientation of the surface, and servomotors will control this orientation.

Introduction:

A load-balancing robot is designed to maintain equilibrium while carrying a load or traversing uneven terrain. These robots are equipped with sensors and control systems to detect changes in their orientation and adjust their movements to keep the load balanced. Load-balancing robots represent a remarkable fusion of mechanical engineering, control theory, and robotics. These robots have garnered significant attention due to their ability to navigate environments while carrying payloads without losing stability. They find applications in various fields, including logistics, manufacturing, healthcare, and entertainment. The core challenge they address is the dynamic stabilization of loads in real-time, making them invaluable for tasks requiring precise movement and control [1].

Background:

The concept of load-balancing robots traces back to the field of inverted pendulum systems, where the goal is to maintain the stability of an upright pendulum by continuously adjusting its base. This concept has been extended to mobile robotic platforms, where the "pendulum" represents the payload or the robot's body, and the base corresponds to the wheels or legs [2].

One of the earliest examples of load-balancing robots is the Segway Personal Transporter, introduced in the early 2000s. While not a traditional robot, the Segway's dynamic stabilization mechanism inspired researchers and engineers to explore similar principles in robotics [3]. Since then, significant advancements in sensor technology, computational power, and control algorithms have led to the development of more sophisticated load-balancing robots capable of handling heavier payloads and navigating challenging terrains.

Load-balancing robots typically use gyroscopes, accelerometers, encoders, and sometimes cameras to measure the robot's orientation, velocity, and position. This sensor data is processed by control algorithms, which calculate the necessary adjustments to maintain balance. These adjustments are then translated into motor commands to actuate the robot's movement [4].

Research in load-balancing robots continues to push the boundaries of what's possible, with ongoing efforts focused on improving stability, efficiency, and adaptability. As the technology matures, load-balancing robots are expected to play a crucial role in various industries, offering solutions for tasks that require precision, mobility, and robustness in dynamic environments [5].

Purpose of work:

A load-balancing robot can balance and stabilize its load under any circumstances, making it an essential application in transportation as the world moves towards automated processes. This robot can help transport sensitive things without breaking them. Also, it can increase the safety of the passengers and make the ride more comfortable for them.

Design

Hardware design

- The components used in this project are as follows:

1) **Arduino Uno**

This microcontroller was used in our project to control the balance surface of the robot. The Arduino was programmed using MATLAB Simulink Model-Based Development.

Figure 1. Arduino Uno

2) **HCS12**

This microcontroller was used in our project to control the robot's movement. The HCS12 was programmed using the C Language.

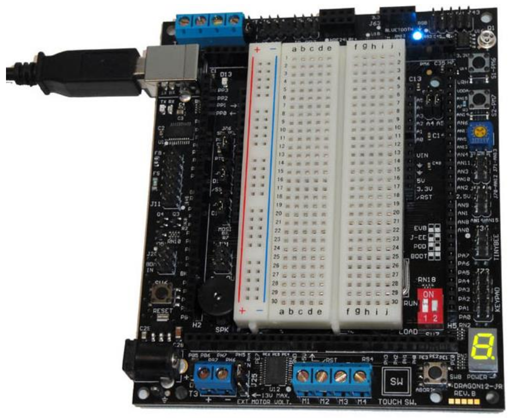


Figure 2. HCS12 Dragon 12 JR

3) **Servo Motor**

One servo motor is used in this project. This motor was used to control the orientation of the surface mounted on the robot; this motor received its signal from the output of the PID controller, which was implemented using model-based programming.

Figure 3. Servo Motor

4) Accelerometer Module

This project uses one GY-61 DXL355 3-axis Accelerometer module, and the GY-61 is a three-axis accelerometer sensor module based on an ADXL355 integrated circuit. This sensor was used in our project to measure the extent to which the surface mounted on the robot deviates from the balance point, and then the measured angle was sent to the Arduino PID.

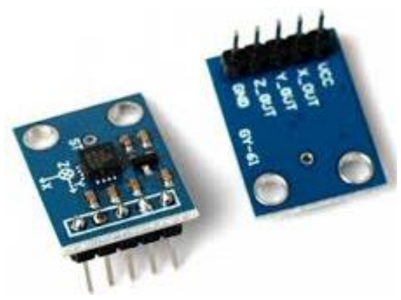


Figure 4. Accelerometer Module

5) Bluetooth Module

one serial communication connection is used in this project; this connection is between the HCS12 SCI0 and a Bluetooth module circuit HC-06; this Bluetooth module is used to transfer data from a mobile phone to the HCS12 to drive the robot.

Figure 5. HC-06 Bluetooth Module

6) IR Module

The IR Sensor Module has a built-in IR transmitter and IR receiver that sends out infrared light and looks for reflected infrared light to detect the presence of any obstacle in front of the sensor. If an obstacle is found, the IR Sensor will communicate with the HCS12 to prevent the robot from moving forward.

Figure 6. IR Sensor Module

7) DC to DC Buck Converter

This converter was used to step down the main supply power (12V DC) to 9V DC to supply the HCS12 and Arduino microcontrollers.

Figure 7. DC to DC Buck Converter

8) H-bridge Module

An H-bridge module is connected to two DC motors in the motor drive circuit. It controls the direction of each motor using two digital pins of the HCS12 for each motor.

Figure 8. H-bridge Module

9) DC Motor Gearbox Wheel and Tyer

In this project, we use two DC motors rated at 3-6V DC to drive the robot's movement.

Figure 9. DC Motor Gearbox Wheel and Tyer

Hardware design

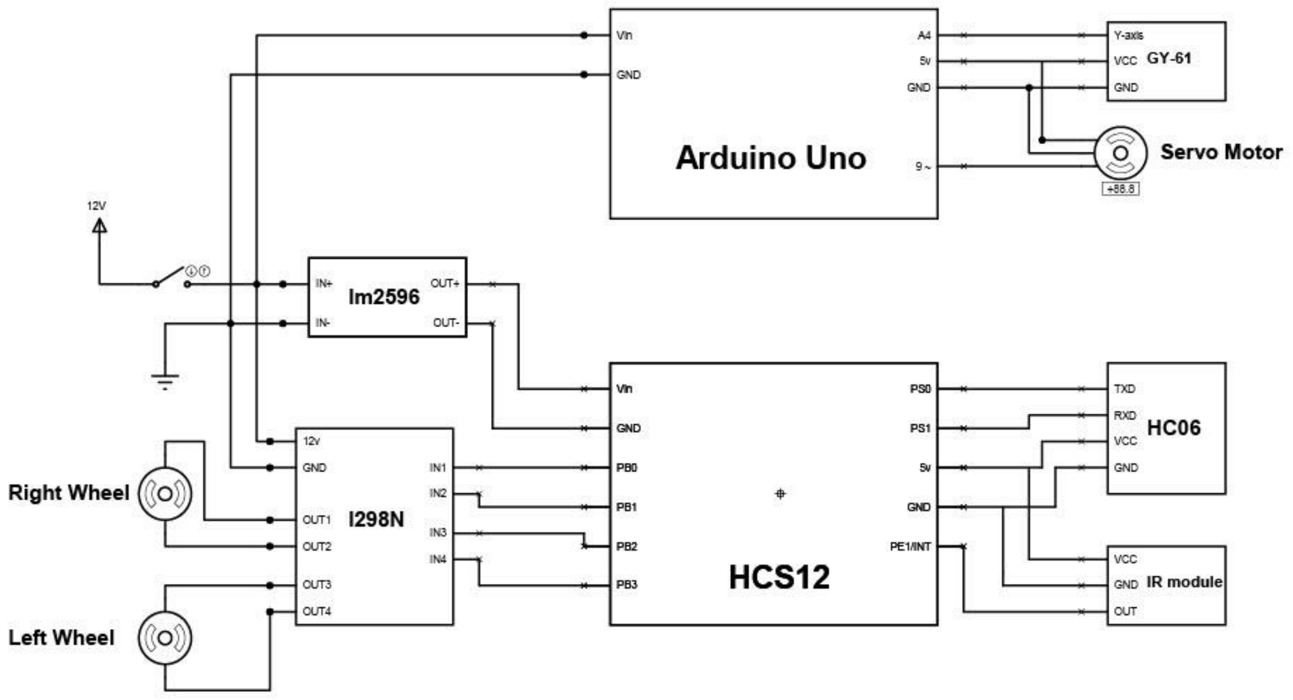


Figure 10. Schematic design

Software design

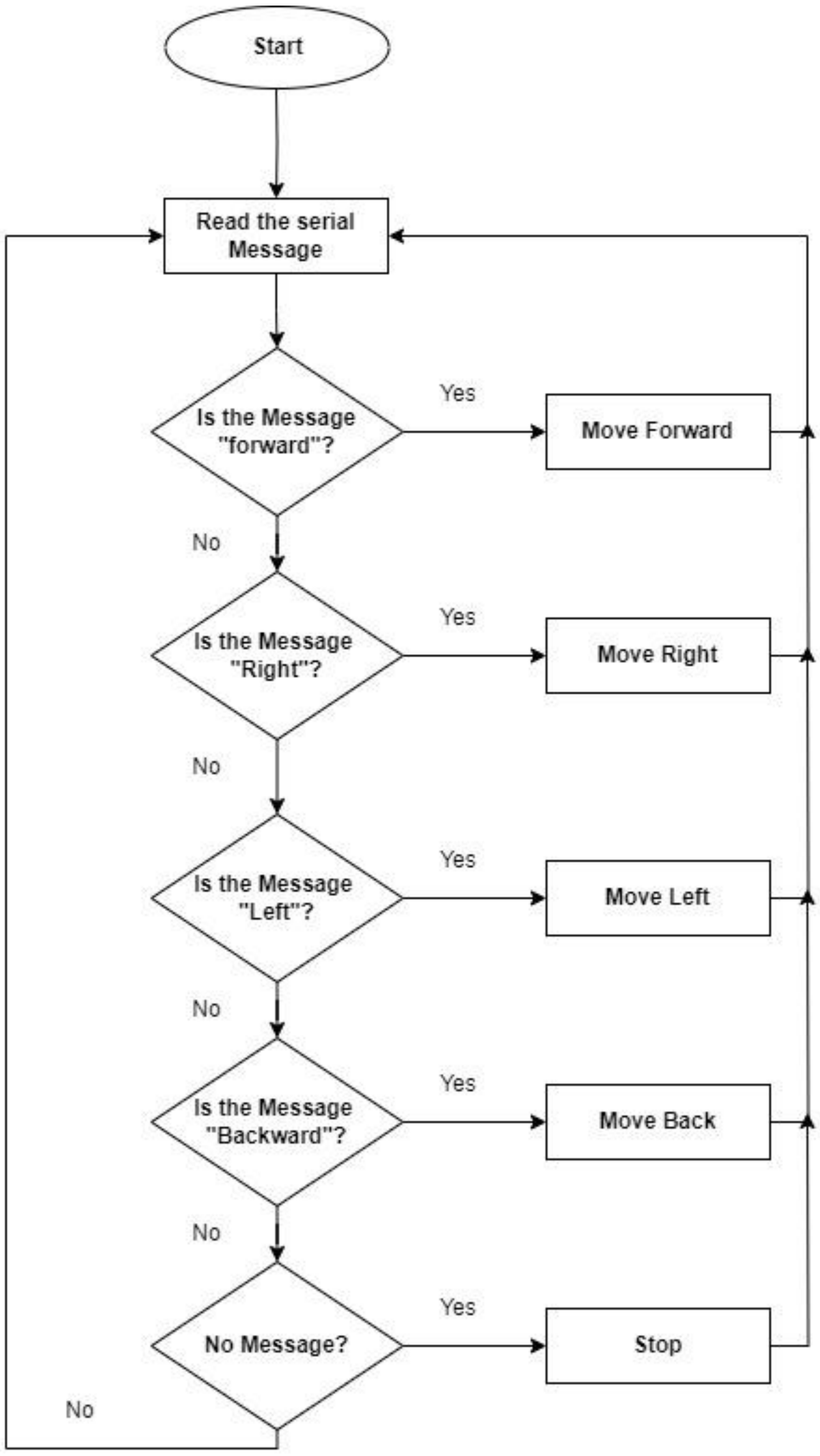


Figure11.software design of HSC12

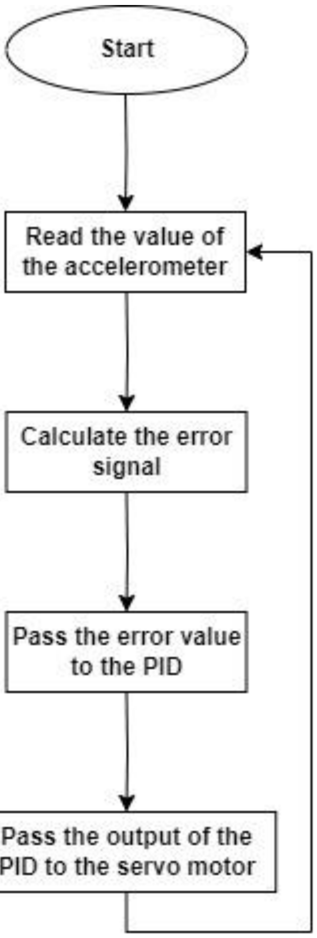

Start
Read the value of the accelerometer
Calculate the error signal
Pass the error value to the PID
Pass the output of the PID to the servo motor

Figure12.MATLAB block diagram

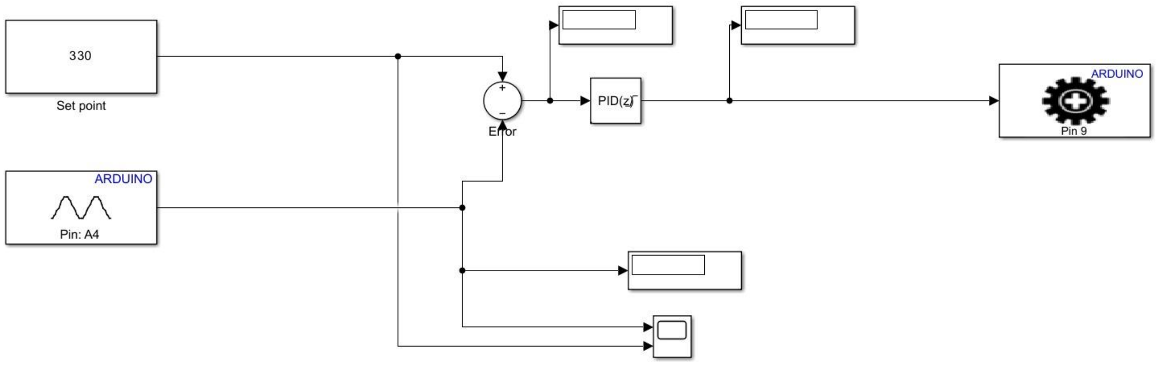


Figure13.design load balancing by MATLAB

PID Controller

A proportional-integral-derivative controller is a generic feedback controller. PID controller processes the "error" as the difference between a measured output and a desired given reference and tries to minimize the error by adjusting the control parameters.

Proportional Control (P)

The difference between the planned setpoint and the actual process variable is known as the current error value, and the proportional component of a PID controller generates an output that is precisely proportionate to this value. The proportionate reaction will be enormous if the mistake is significant, and vice versa. The controller's sensitivity to the error is determined by the proportional gain (Kp). However, P control alone cannot eliminate the steady-state error, which results in a permanent offset.

Integral Control (I)

The integral component eliminates the steady-state error, gradually accumulating the error and modifying the controller’s output. The mistake is added over time and multiplied by the integral gain (Ki). Ensuring that the total error is pushed to zero eliminates whatever offset the proportional control may have left behind. On the other hand, instability and oscillations might result from excessive integral activity.

Derivative Control (D)

Based on its rate of change, the derivative component forecasts *inaccuracy in the future. Increasing the derivative gain (Kd) by the error’s derivative over time produces a damping effect. Doing this makes the response smoother, and oscillations and overshooting are lessened. Derivative control, however, is susceptible to error signal noise and, if mishandled, may intensify it.

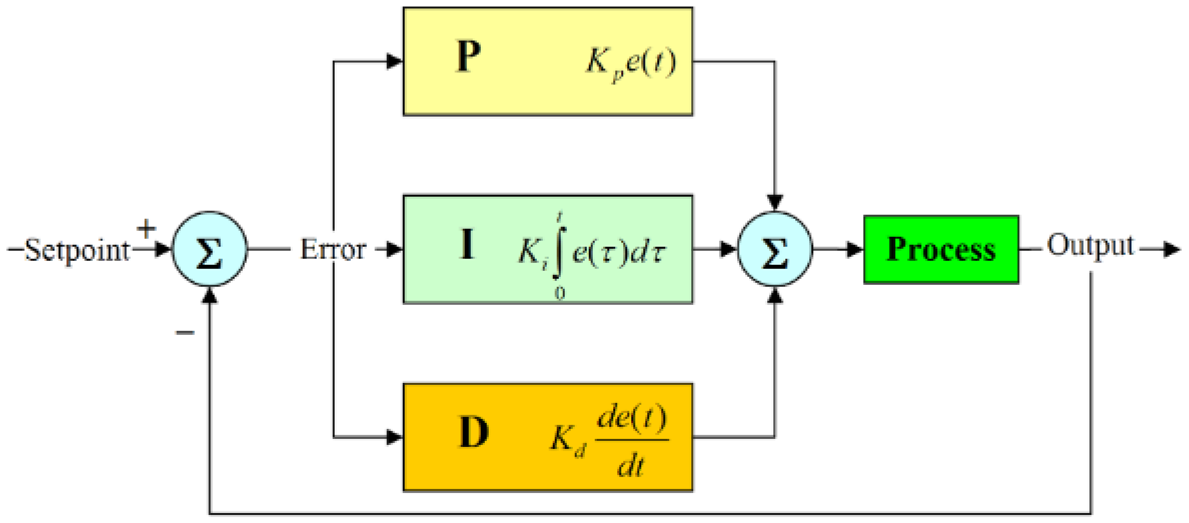
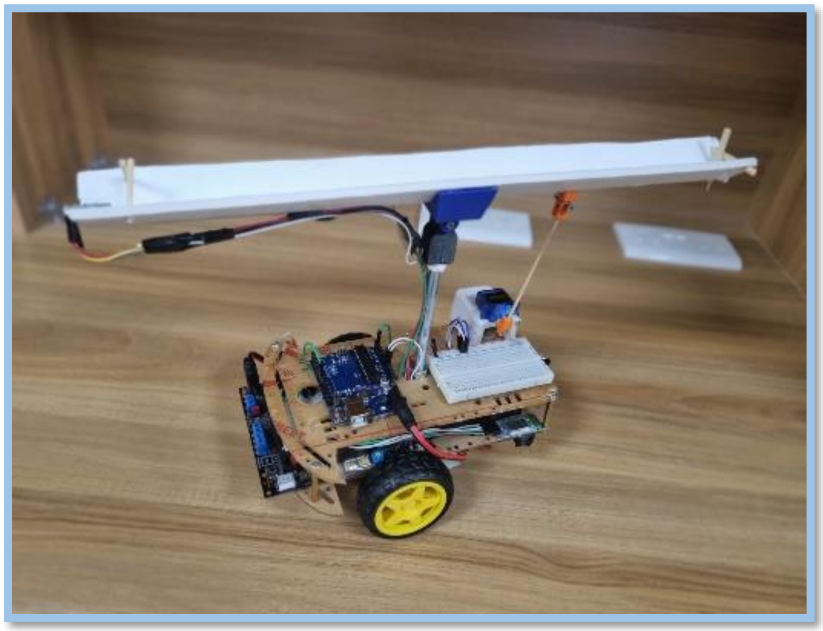


Figure14. PID Controller

final project



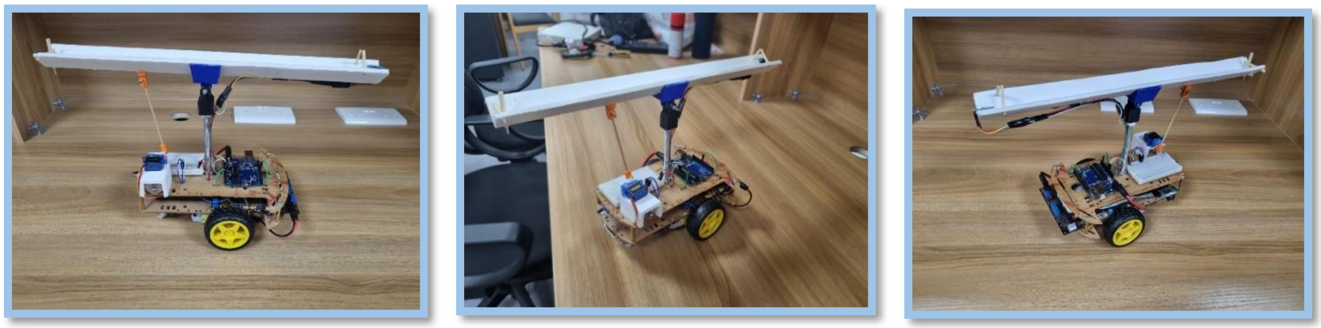


Figure 15. final project

Problems and recommendations

1. **power problem:**
   we use a 9-volt battery to operate HSC12 and Arduino, but that is insufficient; the battery drained quickly, causing the microcontroller to reset itself continually. so, to solve this problem, we utilized a buck converter connected to a lithium battery
2. **PID tuning:**
   The PID has been tuned using the Ziegler-Nichols method, but we could not find the ultimate gain of P, at which the control loop output has stable and consistent oscillations. For future work, lead-lag controllers can be used to solve this problem.

Conclusion

A load-balancing robot can balance and stabilize its load under any circumstances, making it an essential application in transportation. Self-balancing applications are widely popular and useful in scooters or other moving wheels, such as self-balancing trays, racks, or anything else. We use a Bluetooth module to receive the direction of movement and an IR sensor to detect obstacles as the car moves it is connected to HSC12. The Arduino will balance the load and keep it stable on the surface using a PID controller and an accelerometer to determine the orientation of the surface, and servomotors will control this orientation.

We could upgrade it to be self-balanced on two axes instead of just one for future work.

References

[1] Malpani, Shubham & Gupta, Shrishti & Agarwal, Shreyanshi & Alam, Mohd & Raj, Geetanjali. (2021). REVIEWING THE PID-BASED TECHNOLOGIES BEHIND THE WHEEL BALANCING ROBOT. International Journal of Technical Research & Science. Special. 56-60. 10.30780/specialissue-ICAASET021/011.

[2] Matesica, Iulian & Nicolae, Mihai & Barbulescu, Liliana & Margeruseanu, Ana-Maria. (2016). Self-balancing robot implementing the inverted pendulum concept. 1-5. 10.1109/RoEduNet.2016.7753230.

[3] R. Babazadeh, A. G. Khiabani, and H. Azmi, "Optimal control of Segway personal transporter," 2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA), Qazvin, Iran, 2016, pp. 18-22, doi: 10.1109/ICCIAutom.2016.7483129.

[4] Majczak, Michał & Wawrzyński, Paweł. (2015). Comparison of two efficient control strategies for a two-wheeled balancing robot. 10.1109/MMAR.2015.7283968.

[5] He, Wen & Pan, Kai & Li, Xun & Chen, An. (2014). Research on Adaptable Load Balancing of Task in Cloud Robots. Advanced Materials Research. 998-999. 1190-1194. 10.4028/www.scientific.net/AMR.998-999.1190.