Engineering Project: Assembly Line for Autonomous Handicap Vehicle

This project involved designing and optimizing an assembly line for a LEGO vehicle to meet specific production goals and financial constraints over a two-year period. The project was supervised by Abdur Rafay, Hani Ahmed, Mohamed Noor, and Mrigendraa Vimal Kannan.

Project Summary

The team was tasked by the LEGO company to design and produce an autonomous, self-driving handicap service vehicle, with the goal of meeting monthly sales demands. The operational facility runs for 4 hours a day, 5 days a week, and 4 weeks a month.

Key Requirements & Constraints

- **Production Goal:** Meet the following monthly demand targets for a two-year project duration:
  - **January - March:** 3,550 cars per month
  - **April - August:** 5,750 cars per month
  - **September - December:** 5,100 cars per month
- **Design Specifications:** The vehicle must be built with a minimum of 50 LEGO pieces , and the total cost per vehicle cannot exceed $600. The team's chosen design uses 62 pieces, costing $372 per vehicle.
- **Operational Budget:**
  - Initial team of 4 human employees with an annual salary of $42,000 each.
  - Option to purchase up to 6 robots at a cost of $90,000 each, plus an $800 monthly operational fee per robot. Robots are 25% faster than human employees.
  - Cost to convert the facility to a production line: $40,000.
  - Hiring/firing costs: $1,200 to fire an employee and a $400 signing bonus to hire.
- **Logistics & Storage:**
  - Packaging costs $40 per car.
  - Initial storage capacity for 500 cars is provided at no additional cost. Any surplus storage costs $2 per car per month.

Assembly Line Details

The LEGO vehicle is assembled in three subassemblies. The total time to build one vehicle is 124 seconds.

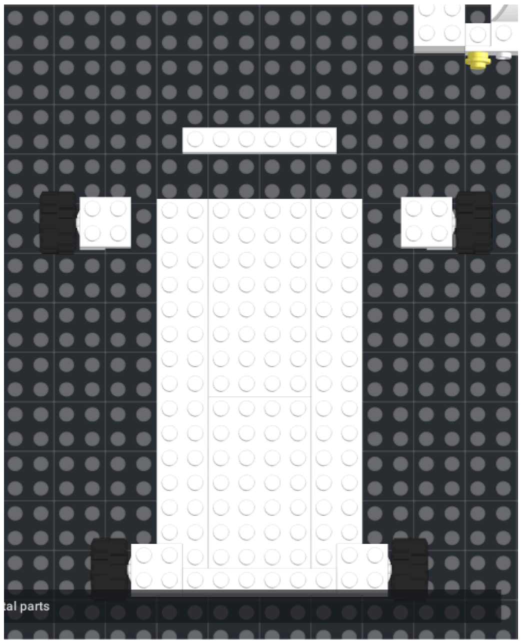
Subassembly	Time to build (s)
Subassembly 1 (base)	41s
Subassembly 2 (middle)	45s
Subassembly 3 (upper)	53s
Total Assembly Time	124s

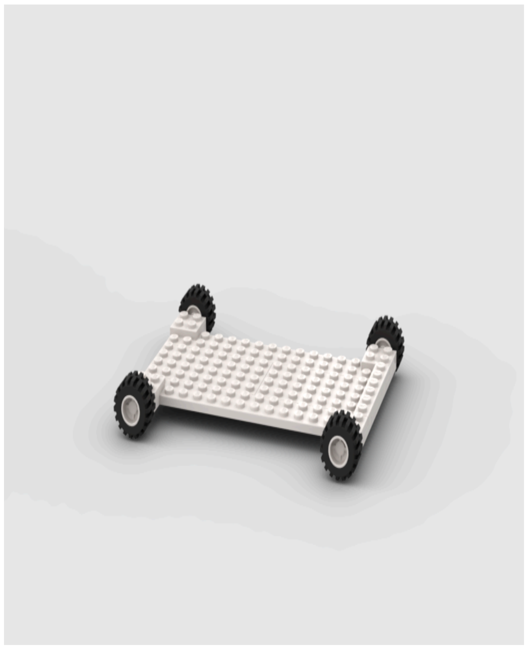
Production and Financial Analysis

Based on the current setup and a total assembly time of 124 seconds, the facility is capable of producing approximately 5,387 vehicles per month. This production rate allows the team to meet the yearly demand by producing 53,871 cars, exceeding the total annual demand of 50,200 cars.

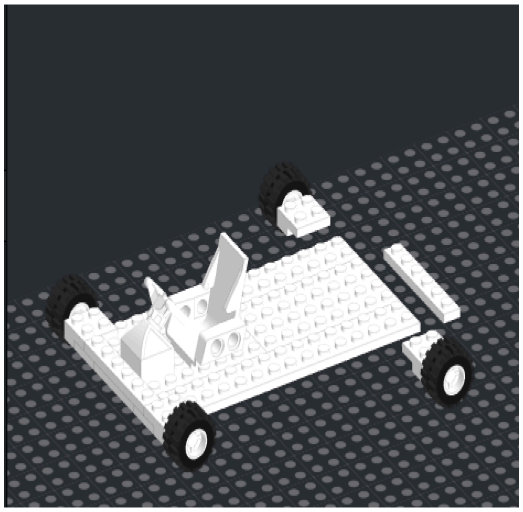
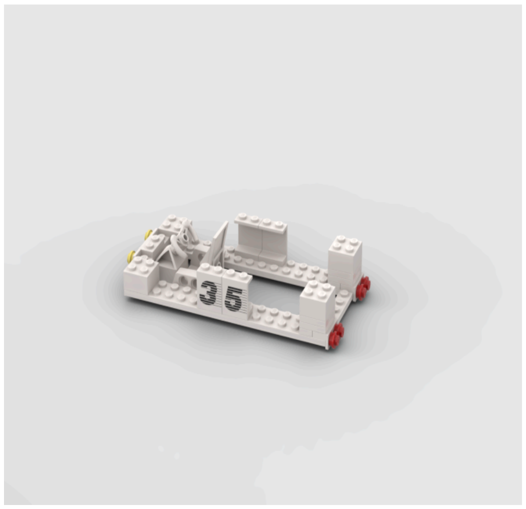
- **Quarterly Production & Surplus/Deficit:**
  - **January - March:** Produces 16,161 vehicles against a demand of 10,650, resulting in a surplus of 5,511 vehicles.
  - **April - August:** Produces 26,935 vehicles against a demand of 28,750, resulting in a deficit of 1,815 vehicles.
  - **September - December:** Produces 21,548 vehicles against a demand of 20,400, resulting in a surplus of 1,148 vehicles.
- **Storage Costs:** The surplus from January to March, after the initial 500 free storage spots, will incur storage fees. The cost for storing the excess 1,337 cars in January is calculated at $2,674 per month.

Subassembly 1 (base)



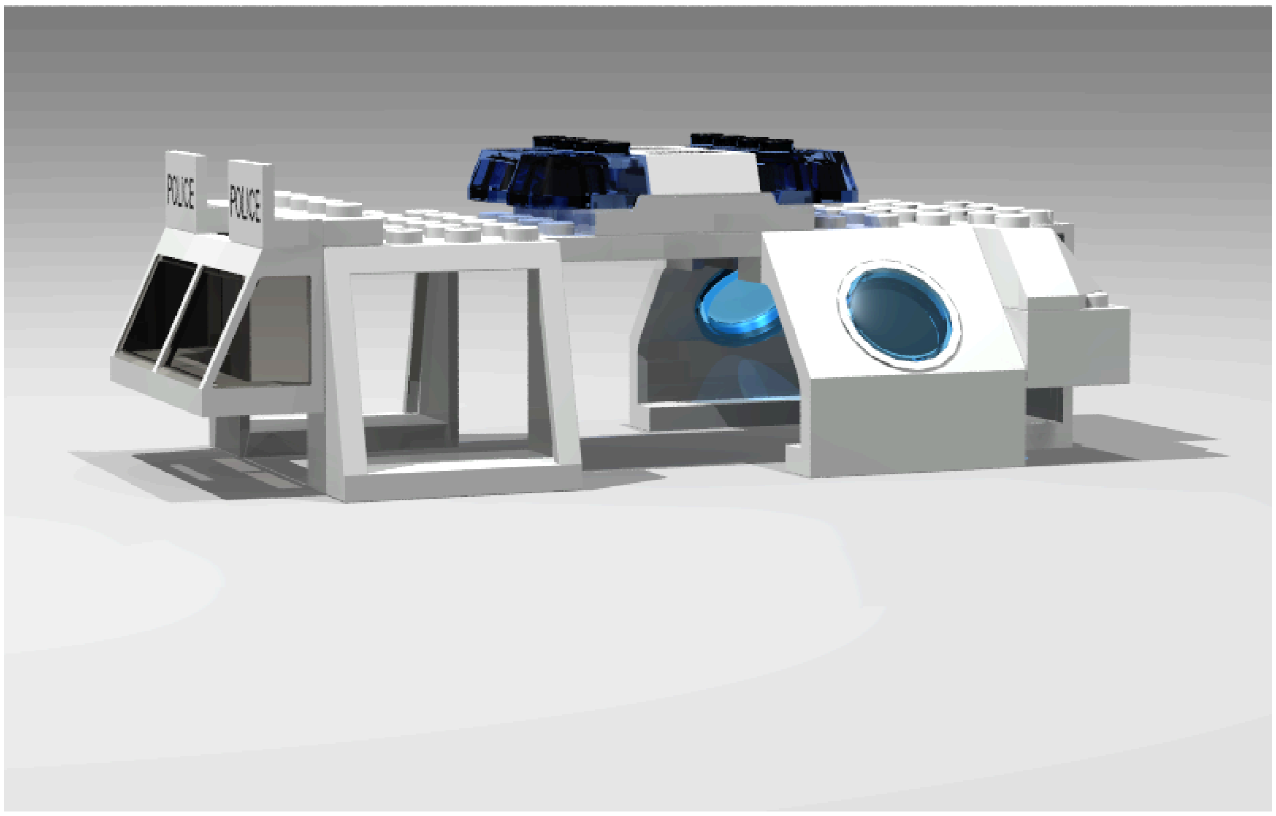

The vehicle's chassis and base, with wheels and steering assembly attached.

Subassembly 2 (middle)

The vehicle's base with the steering wheel holder and seat added, along with the middle section of the body frame and initial detailing.

Subassembly 3 (upper)



The upper section of the vehicle, which includes the windscreen, roof, and siren components.

Automated Vehicle Control System

We developed a robust MATLAB program to autonomously control the vehicle, leveraging the EV3 brick's processing power and a suite of integrated sensors to execute complex maneuvers.

In addition to its autonomous functions, the vehicle's remote-control capabilities were a key milestone. The MATLAB code enabled precise remote operation and programmed color-based responses:

- **Red:** The robot is programmed to stop for one second upon detecting red.
- **Blue:** The robot stops and beeps two times upon detecting blue.
- **Green:** The robot stops and beeps three times upon detecting green.
- **Yellow:** The program seamlessly transitions to a manual remote-control state, allowing us to manually steer the robot and operate the lifting mechanism to pick up the wheelchair. Once the robot exits the yellow zone, it autonomously reverts to its original programmed path.

Live Demonstration: Autonomous Handicap Vehicle Service :
https://youtu.be/BW-gpezHiJc

MATLAB Program

The code below was used for the final demonstration:

```
brick.SetColorMode(4, 2);
gyroPort = 2;
brick.GyroCalibrate(gyroPort);
% Define motor
leftMotor = 'D';
rightMotor = 'A';
% Initialize sensors
brick.SetColorMode(4, 2);
gyroPort = 2;
brick.GyroCalibrate(gyroPort);
% Define motor ports
leftMotor = 'D';
rightMotor = 'A';
% Navigation
forwardSpeed = -50;
turnSpeed = 30;
pauseDuration = 1;
openingThreshold = 50;
wallFollowingDistance = 20;
emergency = 5;
backupTime = 1.0;
% Main navigation
while true

    color = brick.ColorCode(4);
    distance = brick.UltrasonicDist(1);
    isTouching = brick.TouchPressed(3);
    gyroAngle = brick.GyroAngle(gyroPort);

    if color == 5

        brick.StopMotor(leftMotor, 'Brake');
        brick.StopMotor(rightMotor, 'Brake');
       pause(1);
```

```
elseif color == 2
    brick.StopMotor(leftMotor, 'Brake');
    brick.StopMotor(rightMotor, 'Brake');
    brick.beep();
    pause(0.3);
    brick.beep();
    pause(0.5);

elseif color == 3

    brick.StopMotor(leftMotor, 'Brake');
    brick.StopMotor(rightMotor, 'Brake');
    brick.beep();
    pause(0.3);
    brick.beep();
    pause(0.3);
    brick.beep();
    pause(0.5);

end

if isTouching

    brick.StopMotor(leftMotor, 'Brake');
    brick.StopMotor(rightMotor, 'Brake');
    pause(pauseDuration);

    brick.MoveMotor(leftMotor, -forwardSpeed);
    brick.MoveMotor(rightMotor, -forwardSpeed);
    pause(backupTime);

    brick.StopMotor(leftMotor, 'Brake');
    brick.StopMotor(rightMotor, 'Brake');
    pause(0.2);
```

```
        brick.MoveMotor(leftMotor, turnSpeed);
        brick.MoveMotor(rightMotor, -turnSpeed);
        pause(1);

        brick.StopMotor(leftMotor, 'Brake');
        brick.StopMotor(rightMotor, 'Brake');
    elseif distance > openingThreshold

     pause(1);
        brick.StopMotor(leftMotor, 'Brake');
        brick.StopMotor(rightMotor, 'Brake');
        pause(1);
                brick.MoveMotor(leftMotor, -turnSpeed);
        brick.MoveMotor(rightMotor, turnSpeed);
        pause(1);

        brick.StopMotor(leftMotor, 'Brake');
        brick.StopMotor(rightMotor, 'Brake');
        brick.MoveMotor(leftMotor, forwardSpeed);
        brick.MoveMotor(rightMotor, forwardSpeed);
        pause(2.5);
        elseif distance < emergency

        brick.MoveMotor(leftMotor, forwardSpeed*.9 );
        brick.MoveMotor(rightMotor, forwardSpeed);
    elseif distance < wallFollowingDistance

        brick.MoveMotor(leftMotor, forwardSpeed*.9);
        brick.MoveMotor(rightMotor, forwardSpeed );
    else
        brick.MoveMotor(leftMotor, forwardSpeed);
        brick.MoveMotor(rightMotor, forwardSpeed);
    end

    pause(0.1);
end
```