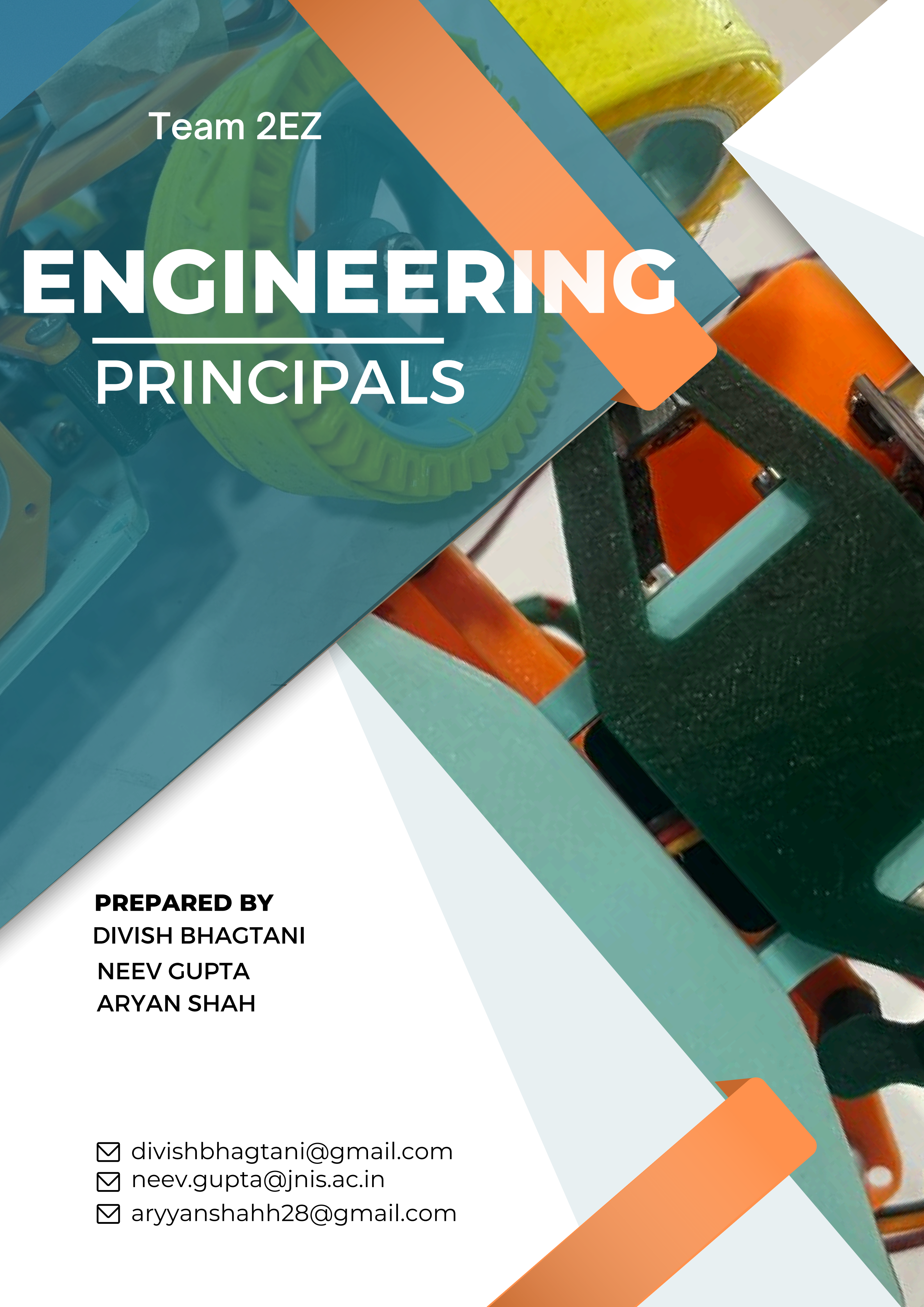


Team 2EZ

ENGINEERING PRINCIPALS

PREPARED BY
DIVISH BHAGTANI
NEEV GUPTA
ARYAN SHAH

divishbhagtani@gmail.com
neev.gupta@jnis.ac.in
aryyanshahh28@gmail.com

1. Mechanical Engineering Principles

1.1 Single Axle Drive with 1:1 Gear Ratio (Kinematics)
The rear-wheel propulsion system is based on a single axle drive with a 1:1 gear ratio, meaning that both rear wheels rotate at the same speed. The principle of kinematics applies here, particularly in terms of understanding how motion (speed and direction) is controlled. By using a single axle, the bot is set up for simple, uniform motion along straight paths.

- **Straight-Line Motion: The constant speed rotation of both rear wheels, as driven by the single axle, simplifies movement along straight paths. The 1:1 gear ratio ensures no differentiation in wheel speed, which aligns well with forward or backward motion.**
- **Turning Radius: Since the bot cannot adjust the speeds of its rear wheels individually (unlike a differential system), turning is entirely dependent on the front-wheel steering. The steering angle provided by the front servo motor defines the bot's turning radius. A sharper angle results in tighter turns, which becomes critical when navigating around obstacles on the track.**

1.2 Load Distribution and Torque (Statics and Dynamics)

In this design, torque is transferred from the DC N20 motor to the wheels, and the load distribution is crucial for effective movement.

- **Torque and Traction: The rear wheels, being driven by the motor, must generate enough torque to propel the bot forward while maintaining traction with the surface of the track. The bot's weight distribution, along with the placement of the motor and servo, impacts how the torque is applied and how stable the bot is during movement.**
- **Stability During Turns: Turning stability is affected by the weight of the bot, its center of mass, and the distribution of load across the wheels. The MG90S servo must apply enough force to change the front-wheel direction without causing instability.**

2. Electrical Engineering Principles

2.1 Pulse Width Modulation (PWM) for Speed Control

The DC N20 motor and MG90S microservo both rely on Pulse Width Modulation (PWM) to control their speed and position. PWM involves varying the duration of electrical pulses to modulate power, which in turn controls the motor's speed or servo's position.

- Motor Speed Regulation: By sending PWM signals to the DC N20 motor, the system can adjust its speed based on track conditions. This ensures that the bot can speed up during straight paths and slow down when approaching obstacles or turns.
- Servo Angle Control: The MG90S servo uses PWM to adjust the angle of the front wheels. For example, the angle changes when the bot detects a red or green obstacle (red for right, green for left), allowing it to make the appropriate turn.

2.2 Voltage Regulation (Buck Converters and Power Management)

The bot uses LM2596 buck converters to step down the voltage from the 11.1V LiPo battery to appropriate levels for different components. This ensures that the components receive stable power within their required operating voltage ranges.

- Power Efficiency: The use of buck converters allows the system to run efficiently by reducing energy losses during power conversion. The voltage is stepped down to 5V for the Arduino, sensors, and Pixy Cam 2, ensuring proper functionality.
- Protection and Safety: Proper power management prevents overheating or damage to sensitive electronics such as the microcontroller, sensors, and motors.

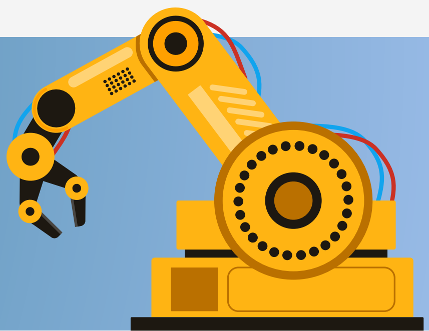
3. Software Engineering Principles

3.1 Sensor Data Integration and Decision-Making (Sensor Fusion)

The bot's ability to navigate the track depends heavily on sensor data. The Pixy Cam 2, ultrasonic sensors, and the IMU BNO055 are used to gather environmental data, which the bot's microcontroller processes to make decisions in real time.
Obstacle Detection and Response: The Pixy Cam 2 detects red or green obstacles. Based on this color detection, the system uses conditional programming (e.g., if statements) to decide whether to turn left or right. The ultrasonic sensors provide distance data from the front, left, and right, preventing collisions by signaling when an object is too close.
Data Filtering: To improve navigation accuracy, the IMU (BNO055) can help filter noise from the sensor data. For example, if the bot is moving on uneven ground, the gyroscope and accelerometer readings from the IMU can help stabilize the movement by making small adjustments in speed or steering.

3.2 Feedback Control System

The bot's navigation relies on a feedback control loop, where data from sensors (ultrasonics, Pixy Cam 2, IMU) is fed back into the system to adjust the bot's movement.
Closed-Loop Control: For instance, if the ultrasonic sensor detects an object in the bot's path, the feedback loop ensures that the motors slow down or stop until the obstacle is cleared. Similarly, the IMU helps maintain orientation, ensuring that the bot doesn't drift off course.
Error Correction: As the bot moves, errors (e.g., drifting due to uneven surfaces) are continuously corrected by comparing the desired state (e.g., the planned path) to the current state (e.g., actual position) and adjusting accordingly. This is vital for smooth turns and consistent movement on the track.



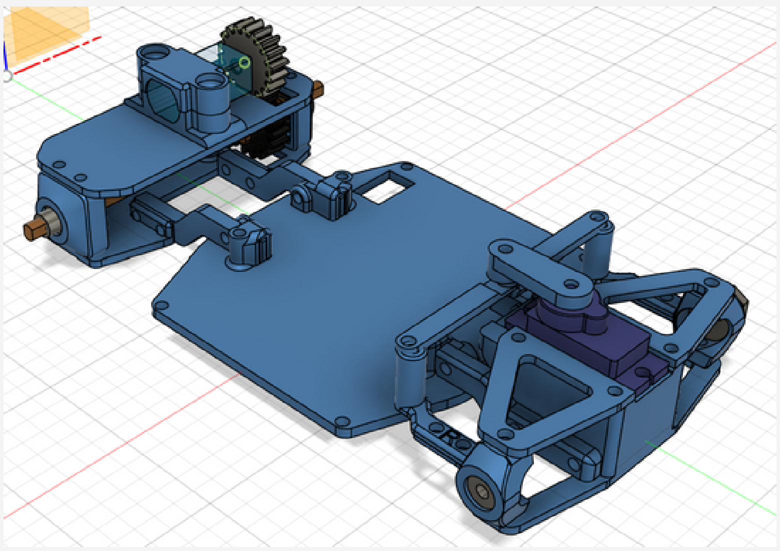
4. Control Systems and Mechatronics

4.1 Proportional-Integral-Derivative (PID) Control

The movement of the bot, especially during turns, can benefit from a PID control system. PID is a control loop feedback mechanism used to ensure that the bot moves smoothly and maintains its path without excessive oscillations or errors.
Proportional Control (P): Corrects the bot's steering or speed based on the difference between the desired position and the current position.
Integral Control (I): Takes the accumulated error over time into account, helping to reduce any long-term drift in the bot's path.
Derivative Control (D): Predicts future errors by calculating the rate of change of the error, allowing the system to react more quickly and avoid overshooting during turns or obstacle avoidance.

4.2 Path Planning and Navigation

While basic navigation depends on immediate sensor data, more advanced engineering can incorporate path planning algorithms. These algorithms use principles from mechatronics and robotics to decide the optimal path based on sensor readings, obstacle locations, and the bot's current speed and direction.
Reactive Path Planning: The bot reacts dynamically to obstacles by adjusting its movement in real time. For example, when the Pixy Cam 2 detects a red obstacle, the bot immediately adjusts its course to the right. The ultrasonic sensors ensure that the bot maintains a safe distance from surrounding objects while navigating.
Predictive Path Adjustment: Using data from the IMU and sensors, the bot could predict future obstacles or path deviations and preemptively adjust its speed and steering to avoid issues.



5. Systems Engineering Principles

5.1 Systems Integration

The bot's performance relies on the seamless integration of multiple subsystems: mechanical, electrical, and software. Systems engineering ensures that all components work together in harmony, from the power management (via buck converters) to motor control, sensor data processing, and decision-making.

Component Interfacing: Proper communication between sensors, microcontroller, motors, and actuators is essential. The Pixy Cam 2, ultrasonic sensors, and IMU need to interface correctly with the Arduino Uno to provide continuous feedback for navigation.

Modularity: The design follows the principle of modularity, where each subsystem (mobility, power, sensing) is developed separately but integrated to form a complete system. This makes it easier to debug, test, and upgrade individual subsystems without affecting the entire bot.

THANK

YOU