Stage 1: Fundamentals (1-2 Weeks)

Goal: Understand the basics of robotic motion and kinematics.

1. **Review Inverse and Forward Kinematics**:
    - Learn how to translate leg trajectories (x, y, z) into joint angles (θ1, θ2, θ3).
    - Verify your kinematics equations with simulations or simple test programs.
    - Tools: MATLAB, Python, or OpenAI Gym.
2. **Understand Gait Cycles**:
    - Learn about common gaits: crawl, trot, gallop.
    - Research how leg timing and phasing maintain stability.
3. **Learn Basics of Interpolation**:
    - Study linear and cubic spline interpolation.
    - Explore how interpolation smooths transitions between points.
4. **Familiarize Yourself with ESP32 Basics**:
    - Learn ESP32 programming with Arduino IDE or PlatformIO.
    - Experiment with basic tasks (blinking LEDs, reading sensors).

Stage 2: Motion Planning (2-3 Weeks)

Goal: Develop skills for trajectory generation and smooth leg movements.

1. **Generate a Walking Trajectory**:
    - Use MATLAB or Python to design a 100-point walking cycle in Cartesian space.
    - Precompute trajectories or develop interpolation functions.
2. **Test Kinematics Equations**:
    - Simulate leg movements by solving inverse kinematics for each point.
    - Visualize the leg's path in 3D to ensure it follows the trajectory.
3. **Write Basic ESP32 Code**:
    - Develop a script to move one motor to a target position using interpolation.
    - Experiment with controlling multiple motors simultaneously.

Stage 3: Multi-Motor Synchronization (2-3 Weeks)

Goal: Control multiple motors and coordinate leg movements.

1. **Control Multiple Servos**:
    - Use libraries like **PCA9685** or ESP32 PWM to control 3 motors per leg.
    - Write code to send commands to multiple motors for a single trajectory.
2. **Synchronize Motion**:
    - Use time-based control (millis or delays) to ensure all motors start and stop simultaneously.
    - Implement linear interpolation to generate smooth joint angles.
3. **Test Leg Movement**:
    - Test moving one leg through its full trajectory while keeping the others stationary.
    - Add smoothing by using trapezoidal velocity profiles or splines.

Stage 4: Full Gait Implementation (2-3 Weeks)

Goal: Coordinate all legs to create a walking gait.

1. **Implement Gait Phases**:
    - Assign time offsets for each leg based on the chosen gait (e.g., trot or crawl).
    - Use your trajectory points to lift, swing, and place each leg in sequence.
2. **Balance the Robot**:
    - Use an IMU (gyroscope/accelerometer) to monitor the robot's balance.
    - Write basic feedback loops to adjust leg positions dynamically.
3. **Test Basic Walking**:
    - Test simple forward walking by coordinating leg movements.
    - Debug and refine timing for stability.

Stage 5: Advanced Motion (2-4 Weeks)

Goal: Add refinement, adaptability, and efficiency to your robot's motion.

1. **Smooth Transitions**:
    - Implement cubic splines for joint interpolation to achieve smoother transitions.
    - Experiment with velocity scaling for speed control.
2. **Adapt to Terrain**:
    - Use sensors (e.g., ultrasonic or force sensors) to adjust leg placement dynamically.
    - Incorporate real-time IK to adapt walking to uneven surfaces.
3. **Optimize Gait**:
    - Experiment with different gaits for efficiency and speed.
    - Optimize power consumption by minimizing unnecessary movements.

Stage 6: Final Refinements and Additions (2+ Weeks)

Goal: Fine-tune performance and add advanced features.

1. **Energy Efficiency**:
    - Fine-tune your servos or motors for smooth operation.
    - Avoid rapid changes in angles to reduce strain.
2. **Obstacle Avoidance**:
    - Use computer vision (e.g., OpenCV on your ESP32 or Jetson) to detect obstacles.
    - Adjust trajectories to avoid collisions.
3. **Implement User Control**:
    - Add remote control functionality using Bluetooth/Wi-Fi (e.g., an app or joystick).
    - Include pre-programmed gaits and manual overrides.
4. **Testing and Debugging**:
    - Perform extensive testing to ensure consistent walking across surfaces.
    - Fix any issues with stability, timing, or motor synchronization.

Tools and Resources

Hardware:

- ESP32 board
- PCA9685 servo driver
- IMU sensor (e.g., MPU6050)
- RC servos or brushless motors with encoders

Software:

- Arduino IDE for ESP32
- MATLAB/Python for simulations
- PlatformIO for advanced ESP32 projects
- Libraries: Adafruit_PWMServoDriver, AccelStepper, or custom PWM control

Learning Resources:

- **Kinematics**: "Introduction to Robotics" by Craig
- **Programming**: Arduino and ESP32 guides
- **Gait Research**: Online papers/videos on robotic quadrupeds

---

What to Focus on First

1. **Simulate Your Trajectories**: Ensure your 100-point cycle works perfectly in simulation.
2. **Control a Single Leg**: Get one leg moving through the trajectory on the ESP32.
3. **Add Synchronization**: Coordinate all legs for your chosen gait.

By following this roadmap, you'll systematically achieve smooth and coordinated walking for your robotic dog. Let me know if you need help with any specific step!