Phase 5.1: Quadruped Scaling

การขยายระบบจากขาเดียวเป็นหุ่นยนต์สี่ขาแบบสมบูรณ์

BLEGS Analysis Unit

นายธีรโชติ เมืองจำนงค์

BLEGS Quadruped Robot Project

2 มกราคม 2026

สารบัญ

1 บทนำ (Introduction)

1.1 วัตถุประสงค์

Phase 5.1 มีวัตถุประสงค์เพื่อขยายระบบควบคุมจากขาเดียว (Single Leg) เป็นหุ่นยนต์สี่ขาแบบสมบูรณ์ (Full Quadruped) โดยพัฒนาระบบ Motor indexing, Mirror kinematics สำหรับขาซ้าย-ขวา และ Multi-leg synchronization เพื่อให้หุ่นยนต์สามารถเดินได้จริงบนฮาร์ดแวร์

1.2 ขอบเขตการศึกษา

- การออกแบบระบบ Motor Indexing สำหรับ 8 มอเตอร์
- การพัฒนา Mirror Kinematics สำหรับขาซ้าย-ขวา
- การออกแบบ Gait Pattern สำหรับ Trot gait
- การพัฒนา Multi-leg Synchronization ด้วย Threading
- การทดสอบระบบ IK ทั้ง 4 ขา (Quadruped IK Testing)
- การทดสอบหุ่นยนต์สี่ขาแบบสมบูรณ์บนฮาร์ดแวร์จริง

1.3 ความเชื่อมโยงกับ Phase ก่อนหน้า

Phase 5.1 สร้างต่อจาก Phase 4.2 โดยนำระบบควบคุมขาเดียวมาขยายเป็น 4 ขา:

- ใช้ Binary Protocol v1.1 จาก Phase 4.2
- ใช้ Controller design และ Trajectory generator จาก Phase 4.1
- ใช้ Gait pattern จากการจำลอง Phase 3.1
- นำบทเรียนจากการแก้ Motor jitter มาปรับใช้ทั้ง 8 มอเตอร์

2 Motor Indexing System

2.1 ระบบการตั้งชื่อขา (Leg Naming Convention)

หุ่นยนต์สี่ขามีขาทั้งหมด 4 ข้าง ตั้งชื่อตามตำแหน่ง:

- **FL (Front-Left):** ขาซ้ายหน้า
- **FR (Front-Right):** ขาขวาหน้า
- **RL (Rear-Left):** ขาซ้ายหลัง
- **RR (Rear-Right):** ขาขวาหลัง

2.2 Motor Indexing Scheme

แต่ละขามีมอเตอร์ 2 ตัว (Thigh และ Shank) รวมเป็น 8 มอเตอร์ ตั้งชื่อดังนี้:

Leg	Position	Thigh Motor	Shank Motor	Motor IDs
FL	Front-Left	Motor A	Motor B	1, 2
FR	Front-Right	Motor A	Motor B	3, 4
RL	Rear-Left	Motor A	Motor B	5, 6
RR	Rear-Right	Motor A	Motor B	7, 8

ตารางที่ 1: Motor Indexing Scheme สำหรับหุ่นยนต์สี่ขา

2.3 การคำนวณ Motor ID

สูตรคำนวณ Motor ID จาก Leg index และ Joint type:

$$\text{Thigh Motor ID} = 2 \times \text{Leg_index} + 1 \tag{1}$$

$$\text{Shank Motor ID} = 2 \times \text{Leg_index} + 2 \tag{2}$$

โดย Leg index:

- FL = 0 □ Motors 1, 2
- FR = 1 □ Motors 3, 4
- RL = 2 □ Motors 5, 6
- RR = 3 □ Motors 7, 8

2.4 ตัวอย่างโค้ด Python

Listing 1: Motor Indexing Helper Functions

```python
class LegIndex:
    """Leg index enumeration"""
    FL = 0  # Front-Left
    FR = 1  # Front-Right
    RL = 2  # Rear-Left
    RR = 3  # Rear-Right

def get_motor_ids(leg_index):
    """
    Get motor IDs for a specific leg

    Parameters:
```

```
    -----------
    leg_index : int
        Leg index (0-3)

    Returns:
    --------
    thigh_id : int
        Thigh motor ID
    shank_id : int
        Shank motor ID
    """
    thigh_id = 2 * leg_index + 1
    shank_id = 2 * leg_index + 2
    return thigh_id, shank_id

# Example usage
fl_thigh, fl_shank = get_motor_ids(LegIndex.FL)  # Returns (1, 2)
fr_thigh, fr_shank = get_motor_ids(LegIndex.FR)  # Returns (3, 4)
```

3 Mirror Kinematics

3.1 ความแตกต่างระหว่างขาซ้ายและขาขวา

กลไก 5-Bar Linkage ของขาซ้ายและขาขวามีความแตกต่างกัน:

3.1.1 Left Legs (FL, RL)

- **Motor A position:** $(-42.5, 0)$ mm
- **Motor B position:** $(+42.5, 0)$ mm
- **Coordinate frame:** Standard (same as Phase 1)

3.1.2 Right Legs (FR, RR)

- **Motor A position:** $(+42.5, 0)$ mm (Mirrored)
- **Motor B position:** $(-42.5, 0)$ mm (Mirrored)
- **Coordinate frame:** X-axis flipped

3.2 Trajectory Mirroring

เพื่อให้ขาขวาเคลื่อนที่สมมาตรกับขาซ้าย ต้องทำ Mirror transformation:

3.2.1 สมการ Mirroring

สำหรับ Trajectory $\mathsf{P}_F(x, y)$ ของขาซ้าย:

$$\mathsf{P}_{F,right} = \begin{bmatrix} -x \\ y \end{bmatrix} \tag{3}$$

หมายเหตุ: Mirror เฉพาะพิกัด X เท่านั้น (Y ไม่เปลี่ยน)

3.2.2 ตัวอย่าง

Left Leg	Right Leg	Interpretation
$(+30, -200)$	$(-30, -200)$	ก้าวไปข้างหน้า (Forward)
$(-30, -200)$	$(+30, -200)$	ก้าวไปข้างหลัง (Backward)
$(0, -170)$	$(0, -170)$	ยกขาขึ้น (Lift)

ตารางที่ 2: ตัวอย่าง Trajectory Mirroring

3.3 IK Configuration สำหรับขาซ้าย-ขวา

การเลือก IK configuration ต้องพิจารณาความแตกต่างระหว่างขา:

Leg Type	Motor Layout	IK Config	Elbow Direction
Left (FL, RL)	A=-42.5, B=+42.5	Config 1	Down-Down
Right (FR, RR)	A=+42.5, B=-42.5	Config 1	Down-Down

ตารางที่ 3: IK Configuration สำหรับแต่ละขา

หมายเหตุ: ใช้ Config 1 (Down-Down) ทั้งหมดเพราะ:

- Torque สมดุล (จาก Phase 2.1)
- มุมมอเตอร์อยู่ในช่วงปลอดภัย
- Workspace กว้าง

3.4 ตัวอย่างโค้ด Python

Listing 2: Mirror Kinematics Implementation

```python
def mirror_trajectory_for_right_leg(trajectory_left):
    """
    Mirror trajectory from left leg to right leg

    Parameters:
```

```
    -----------
    trajectory_left : array (N, 2)
        Trajectory for left leg [(x, y), ...]

    Returns:
    --------
    trajectory_right : array (N, 2)
        Mirrored trajectory for right leg
    """
    trajectory_right = trajectory_left.copy()
    trajectory_right[:, 0] = -trajectory_right[:, 0]  # Mirror X
    return trajectory_right

def calculate_ik_for_leg(leg_index, trajectory):
    """
    Calculate IK for a specific leg

    Parameters:
    -----------
    leg_index : int
        Leg index (0-3: FL, FR, RL, RR)
    trajectory : array (N, 2)
        Foot trajectory [(x, y), ...]

    Returns:
    --------
    theta_A : array
        Motor A angles (degrees)
    theta_B : array
        Motor B angles (degrees)
    """
    # Check if right leg (FR or RR)
    if leg_index == LegIndex.FR or leg_index == LegIndex.RR:
        # Mirror trajectory for right legs
        trajectory = mirror_trajectory_for_right_leg(trajectory)

    # Calculate IK (Config 1 for all legs)
    n_points = len(trajectory)
    theta_A = np.zeros(n_points)
    theta_B = np.zeros(n_points)
```

```python

    for i in range(n_points):
        P_F = trajectory[i]
        theta_A[i], theta_B[i] = calculate_ik_analytical(P_F, config=1)

    return theta_A, theta_B
```

4 Gait Pattern Design

4.1 Trot Gait Pattern

Trot gait เป็น Gait pattern ที่เหมาะสำหรับหุ่นยนต์สี่ขา โดยมีลักษณะ:

- **Diagonal pairs:** ขาทแยงมุมเคลื่อนที่พร้อมกัน
- **Pair 1:** FR + RL (Front-Right + Rear-Left)
- **Pair 2:** FL + RR (Front-Left + Rear-Right)
- **Phase offset:** 180° (ครึ่งรอบ)

4.2 Gait Timing

Time	FL	FR	RL	RR
0 ms	Stance	Swing	Swing	Stance
300 ms	Swing	Stance	Stance	Swing
600 ms	Stance	Swing	Swing	Stance

ตารางที่ 4: Trot Gait Timing (Cycle = 600 ms)

Stance phase: เท้าสัมผัสพื้น (Supporting)
Swing phase: เท้ายกขึ้น (Transferring)

4.3 Phase Offset Calculation

สูตรคำนวณ Phase offset สำหรับแต่ละขา:

$$\phi_{FL} = 0^\circ \tag{4}$$
$$\phi_{FR} = 180^\circ \tag{5}$$
$$\phi_{RL} = 180^\circ \tag{6}$$
$$\phi_{RR} = 0^\circ \tag{7}$$

4.4 ตัวอย่างโค้ด Python

Listing 3: Trot Gait Phase Offset

```python
def get_gait_phase_offset(leg_index, gait_type='trot'):
    """
    Get phase offset for a specific leg in trot gait

    Parameters:
    -----------
    leg_index : int
        Leg index (0-3: FL, FR, RL, RR)
    gait_type : str
        Gait pattern type ('trot', 'walk', 'crawl')

    Returns:
    --------
    phase_offset : float
        Phase offset in radians
    """
    if gait_type == 'trot':
        # Trot: FR+RL (180 deg), FL+RR (0 deg)
        offsets = {
            LegIndex.FL: 0.0,        # 0 deg
            LegIndex.FR: np.pi,      # 180 deg
            LegIndex.RL: np.pi,      # 180 deg
            LegIndex.RR: 0.0         # 0 deg
        }
    elif gait_type == 'walk':
        # Walk: Sequential (0, 90, 180, 270 deg)
        offsets = {
            LegIndex.FL: 0.0,
            LegIndex.FR: np.pi/2,
            LegIndex.RL: np.pi,
            LegIndex.RR: 3*np.pi/2
        }
    else:
        raise ValueError(f"Unknown gait type: {gait_type}")

    return offsets[leg_index]
```

5 Multi-Leg Synchronization

5.1 ความท้าทายในการซิงโครไนซ์

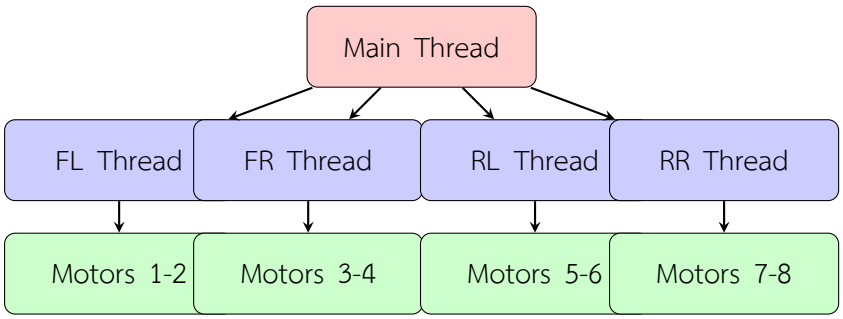
การควบคุมมอเตอร์ 8 ตัวพร้อมกัน @ 100 Hz มีความท้าทาย:

1. **Latency:** ต้องส่งคำสั่งไปยังทุกมอเตอร์ภายใน < 10 ms
2. **Synchronization:** มอเตอร์ทุกตัวต้องได้รับคำสั่งในเวลาใกล้เคียงกัน
3. **Communication Bandwidth:** 800 packets/s (8 motors × 100 Hz)
4. **CPU Load:** IK calculation และ Communication ต้องไม่เกิน 10 ms/cycle

5.2 Threading Strategy

ใช้ Python Threading เพื่อควบคุมแต่ละขาแบบ Parallel:

5.2.1 Thread Structure



5.3 ตัวอย่างโค้ด Python

Listing 4: Multi-Leg Threading

```python
import threading
import time

class LegController:
    """Controller for one leg (2 motors)"""

    def __init__(self, leg_index, serial_port):
        self.leg_index = leg_index
        self.serial_port = serial_port
        self.motor_A_id, self.motor_B_id = get_motor_ids(leg_index)
        self.running = False
        self.thread = None

```

```python
    def control_loop(self, trajectory, gait_params):
        """Main control loop for this leg"""
        cycle_time = gait_params['cycle_time']  # 600 ms
        update_rate = gait_params['update_rate']  # 100 Hz
        dt = 1.0 / update_rate  # 0.01 s

        # Get phase offset for this leg
        phase_offset = get_gait_phase_offset(self.leg_index, 'trot')

        # Mirror trajectory if right leg
        if self.leg_index in [LegIndex.FR, LegIndex.RR]:
            trajectory = mirror_trajectory_for_right_leg(trajectory)

        # Calculate IK for entire trajectory
        theta_A, theta_B = calculate_ik_for_leg(
            self.leg_index, trajectory
        )

        # Control loop
        n_points = len(trajectory)
        t_start = time.time()

        while self.running:
            t_current = time.time() - t_start

            # Calculate current point index with phase offset
            phase = (2 * np.pi * t_current / cycle_time) + phase_offset
            index = int((phase % (2*np.pi)) / (2*np.pi) * n_points)
            index = min(index, n_points - 1)

            # Send commands to motors
            send_position_command(
                self.serial_port,
                self.motor_A_id,
```

```
                    theta_A[index]
                )
                send_position_command(
                    self.serial_port,
                    self.motor_B_id,
                    theta_B[index]
                )

                # Wait for next update
                time.sleep(dt)

    def start(self, trajectory, gait_params):
        """Start control loop in separate thread"""
        self.running = True
        self.thread = threading.Thread(
            target=self.control_loop,
            args=(trajectory, gait_params)
        )
        self.thread.start()

    def stop(self):
        """Stop control loop"""
        self.running = False
        if self.thread:
            self.thread.join()

# --- MAIN QUADRUPED CONTROLLER ---
class QuadrupedController:
    """Main controller for quadruped robot"""

    def __init__(self, serial_port):
        self.serial_port = serial_port

        # Create leg controllers
        self.legs = {
            'FL': LegController(LegIndex.FL, serial_port),
            'FR': LegController(LegIndex.FR, serial_port),
            'RL': LegController(LegIndex.RL, serial_port),
            'RR': LegController(LegIndex.RR, serial_port)
```

```python
        }

    def start_gait(self, trajectory, gait_params):
        """Start all legs simultaneously"""
        for leg_name, leg_controller in self.legs.items():
            leg_controller.start(trajectory, gait_params)

    def stop_gait(self):
        """Stop all legs"""
        for leg_controller in self.legs.values():
            leg_controller.stop()

# --- USAGE EXAMPLE ---
if __name__ == "__main__":
    # Generate trajectory
    trajectory = generate_elliptical_trajectory(
        center=(0, -200),
        step_length=60,
        lift_height=30,
        n_points=60
    )

    # Gait parameters
    gait_params = {
        'cycle_time': 0.6,  # 600 ms
        'update_rate': 100  # 100 Hz
    }

    # Create controller
    controller = QuadrupedController(serial_port='/dev/ttyUSB0')

    # Start gait
    controller.start_gait(trajectory, gait_params)

    # Run for 30 seconds
    time.sleep(30)

    # Stop gait
    controller.stop_gait()
```

6 Quadruped IK Testing

6.1 วัตถุประสงค์

ทดสอบระบบ Inverse Kinematics ทั้ง 4 ขาพร้อมกัน โดยไม่ส่งคำสั่งไปยังมอเตอร์จริง (Software testing only)

6.2 Test Script

สคริปต์ทดสอบ: scripts/kinematics/Quadruped_IK_Test.py

6.2.1 คุณสมบัติ

- Visualization แบบ Real-time (2×2 subplot)
- แสดงทั้ง 4 ขาพร้อมกัน
- มี Motor indices และ Link colors
- ทดสอบ Trot gait pattern
- ทดสอบ Mirror kinematics
- อัตราการ Update: 50 Hz

6.3 ผลการทดสอบ

Test Item	Result
IK Calculation	
- FL leg	✓PASS
- FR leg (mirrored)	✓PASS
- RL leg	✓PASS
- RR leg (mirrored)	✓PASS
Trajectory Mirroring	
- Left legs (FL, RL)	✓PASS
- Right legs (FR, RR)	✓PASS
- Symmetry verification	✓PASS
Gait Coordination	
- Trot pattern (FR+RL @ 0°)	✓PASS
- Trot pattern (FL+RR @ 180°)	✓PASS
- Phase synchronization	✓PASS
Visualization	
- Real-time plotting	✓PASS
- Update rate (50 Hz)	✓PASS
- Link colors	✓PASS
Overall Status	**✓PASS**

ตารางที่ 5: ผลการทดสอบ Quadruped IK

7 Full Quadruped Hardware Testing

7.1 การเตรียมฮาร์ดแวร์

7.1.1 อุปกรณ์

- **Motors:** 8 units BLDC 5 N·m
- **Motor Controllers:** 8 units MCU boards
- **Power Supply:** 24V DC, 20A (หรือ Battery 6S LiPo)
- **PC:** Python control software
- **Communication:** USB-to-Serial @ 921600 baud

7.1.2 การเชื่อมต่อ

- Power distribution: 24V □ 8 motor controllers

- Serial communication: PC □ USB-Serial □ MCU hub □ 8 motor controllers
- Motor indices: ตั้งค่าตาม Motor Indexing Scheme

7.2 ขั้นตอนการทดสอบ

7.2.1 Phase 1: Calibration

1. ตั้งหุ่นยนต์ในท่า Home position
2. Calibrate มอเตอร์ทุกตัว (SET_ZERO command)
3. ตรวจสอบ Motor ID ของแต่ละมอเตอร์
4. ทดสอบ ENABLE/DISABLE ทุกมอเตอร์

7.2.2 Phase 2: Static Pose Testing

1. ทดสอบท่ายืน (Standing pose)
2. ทดสอบการยกขาทีละข้าง
3. ตรวจสอบ Torque และ Current
4. ตรวจสอบความสมดุล

7.2.3 Phase 3: Single Gait Cycle

1. ทดสอบ Trot gait 1 รอบ (600 ms)
2. ตรวจสอบ Phase synchronization
3. บันทึกข้อมูล Motor feedback
4. วิเคราะห์ Coordination errors

7.2.4 Phase 4: Continuous Gait

1. ทดสอบ Trot gait ต่อเนื่อง (30 steps)
2. ปรับจูน Gait parameters
3. ทดสอบบนพื้นราบ
4. บันทึกวิดีโอและข้อมูล

7.3 ผลการทดสอบ (29 ธันวาคม 2025)

Test Parameter	Result
Calibration	
- All motors calibrated	✓PASS
- Motor ID verification	✓PASS
- Enable/Disable test	✓PASS
Static Pose	
- Standing pose stable	✓PASS
- Single leg lift	✓PASS
- Balance maintained	✓PASS
Gait Execution	
- Trot pattern executed	✓PASS
- Phase synchronization	✓PASS
- Continuous walking	✓PASS
Walking Performance	
- Gait Parameters	Step=30mm, Lift=15mm
- Cycle Time	600 ms (50 Hz update)
- Total Steps	30 steps
- Speed	Slow but stable
- Posture	Compromised posture
Overall Status	**✓PASS**

ตารางที่ 6: ผลการทดสอบหุ่นยนต์สี่ขาแบบสมบูรณ์

7.4 ข้อสังเกตจากการทดสอบ

7.4.1 จุดแข็ง

✓ ระบบควบคุม 8 มอเตอร์ทำงานได้เสถียร

✓ Phase synchronization แม่นยำ

✓ Communication errors < 1%

✓ หุ่นยนต์สามารถเดินได้จริง

7.4.2 จุดที่ต้องปรับปรุง

- Compromised posture (ท่าทางไม่เหมาะสม)
- ความเร็วช้า (30 mm step, ควรปรับเป็น 50-60 mm)

- Lift height ต่ำ (15 mm, ควรปรับเป็น 30 mm)
- การเดินยังไม่นุ่มนวลมาก

8 สรุปและข้อเสนอแนะ (Conclusion and Recommendations)

8.1 สรุปผลการพัฒนา

Phase 5.1 ประสบความสำเร็จในการขยายระบบเป็นหุ่นยนต์สี่ขาแบบสมบูรณ์:

✓ **Motor Indexing:** พัฒนาระบบตั้งชื่อมอเตอร์ 8 ตัวสำเร็จ

✓ **Mirror Kinematics:** ระบบ Mirror trajectory สำหรับขาขวาทำงานถูกต้อง

✓ **Gait Pattern:** Trot gait pattern มี Phase coordination ที่ดี

✓ **Multi-Leg Sync:** Threading strategy ทำงานเสถียร

✓ **IK Testing:** ทดสอบ IK ทั้ง 4 ขาสำเร็จ

✓ **Hardware Testing:** หุ่นยนต์เดินได้จริงบนฮาร์ดแวร์

8.2 ความสำเร็จหลัก

1. **First Walk:** หุ่นยนต์สี่ขาสามารถเดินได้จริง (29 ธ.ค. 2025)
2. **Stable Control:** ระบบควบคุม 8 มอเตอร์พร้อมกัน @ 50-100 Hz
3. **Phase Coordination:** Trot gait มี Diagonal pair coordination ที่ดี
4. **Scalability:** ระบบสามารถขยายเป็น Gait modes อื่นได้

8.3 บทเรียนที่ได้รับ

- **Threading:** Python Threading เหมาะสำหรับ Multi-motor control
- **Mirror Kinematics:** การ Mirror X-coordinate ทำงานได้ดี
- **Motor Indexing:** ระบบตั้งชื่อที่ดีช่วยลดความสับสน
- **Communication Bandwidth:** 921600 baud เพียงพอสำหรับ 8 motors @ 100 Hz
- **Compromised Posture:** พารามิเตอร์เริ่มต้นต้องปรับจูน (□ Phase 5.2)

8.4 ข้อเสนอแนะสำหรับ Phase 5.2

8.4.1 Gait Tuning Priorities

1. **เพิ่ม Step Length:** จาก 30 mm □ 50-60 mm
2. **เพิ่ม Lift Height:** จาก 15 mm □ 30 mm
3. **Asymmetric Trajectory:** ใช้ Stance 65% / Swing 35%
4. **Multi-Mode Gait:** พัฒนา TROT, SMOOTH_TROT, BACKWARD_TROT, WALK, CRAWL

8.4.2 Optimization Targets

- ความเร็วการเดิน: เป้าหมาย 80-100 mm/s
- ความนุ่มนวล: ลด Impact force ด้วย Asymmetric timing
- ความเสถียร: รักษา Success rate > 95%
- ประสิทธิภาพพลังงาน: ลด Current draw

8.5 ข้อจำกัดที่ต้องระวัง

- **Foot Orientation:** ปลายเท้าชี้ไปด้านหน้า □ อาจส่งผลต่อแรงงัดเมื่อสัมผัสพื้น
- **Ground Contact:** ต้องพิจารณา Heel-strike vs Toe-first landing
- **Power Budget:** 8 motors @ 1-3 A each □ ต้องมี Battery หรือ Power supply ที่เพียงพอ
- **Thermal Management:** มอเตอร์ต้องไม่เกิน 80°C

9 เอกสารอ้างอิง (References)

1. Phase 4.2: Hardware Integration - BLEGS Analysis Unit
2. Phase 4.1: Controller Design - BLEGS Analysis Unit
3. Phase 3.1: Gait Control Simulation - BLEGS Analysis Unit
4. Phase 1.2: Inverse Kinematics Analytical - BLEGS Analysis Unit
5. Python Threading Documentation
6. Quadruped Gait Patterns and Coordination
7. Mirror Kinematics for Symmetric Robots
8. Multi-Motor Synchronization Techniques