제 Ⅰ 장 서 론

1. 연구의 목적

본 연구의 목적은 6족 보행로봇(Hexapod)을 직접 제작하여 6족 보행로봇이 가지고 있는 메커니즘을 이해하고 연구하여, 다양한 로봇 동작제어를 구현하는 것이다. 기존 로봇의 주 이동수단으로는 바퀴나, 캐터필러등과 같이 차륜을 이용하여 이동을 하는데 반해, 2족 혹은 다족을 가지고 있어 이를 이용하여 보행을 하는 보행 로봇(Walking Robot)은 험한 지형에서도 이동이 가능하다는 점에서 과거부터 현재까지 많은 관심과 연구가 진행되고 있다. 특히나 이 보행로봇은 평탄한 장소가 아닌 건설, 광업, 해양, 군사 등에서 주로 작업을 하기위해 개발되었으며, 그 크기에 따라 초소형 보행로봇부터 시작하여, 사람이 탑승할 수 있는 대형 보행로봇까지 아주 다양하다.

그림 1-1) 탑승형 2족 보행로봇 '메소드-2'
(이미지 출처 : 한국미래기술)

그림 1-2) Boston Dynamics 사의 'BigDog'(이미지 출처 : gizmodo)

또한 2족 보행로봇은 대지 적응성이 높은데 반해, 불안정한 구조성으로 인해 자세 안정성이 낮아, 이를 보완하여 대지 적응성과 자세 안정도를 높힌 4족, 6족 보행 로봇이 현재 많은 각광을 받고 있는 추세이다. 따라서 이번 연구 목적 또한 이 다족 보행 로봇 중 가장 자세 안정성이 높고 비교적 제어가 간단한 '6족 보행로봇(Hexapod)'을 직접 제작 및 메커니즘을 이해 하고자 한다.

본 연구에서 제작하게 될 6족 보행로봇의 주된 특징으로는 아래와 같다.

(1) 1족 제어 방식과 3족 제어 방식 이동
(2) 6족 3관절 방식을 통한 부드럽고, 유연한 자세 구현
(3) 컴퓨터와의 시리얼 통신을 이용하여 다리 각도 제어
(4) 기본적인 전진, 좌우 회전을 포함한 대기 및 앉기 등 다양한 자세 가능
(3) Bluetooth 통신을 이용하여 원거리 제어(스마트폰과 연동)

이 외에도 다양한 센서들을 이용하여 완성도 높은 로봇을 만들 수 있지만, 본 연구의 주된 목적은 로봇 메커니즘 개발이므로 다른 기능은 개발에 포함하지 않았다.

이 연구에서 로봇의 주 핵심 부품인 메인 MCU(Micro Controller Unit)은 ARM Cortex-M3 프로세서를 가지고 있는 ST사의 'Nucleo STM32f103RB'를 사용할 것이며, 로봇의 몸체 프레임은 3D Printer을 이용하여 제작하고, 로봇의 구동을 위한 모터로는 보급형 서보모터인 'SG-90'를 사용하였다. 기타 자세한 부품 정보는 다음 목차에 기술하였다.

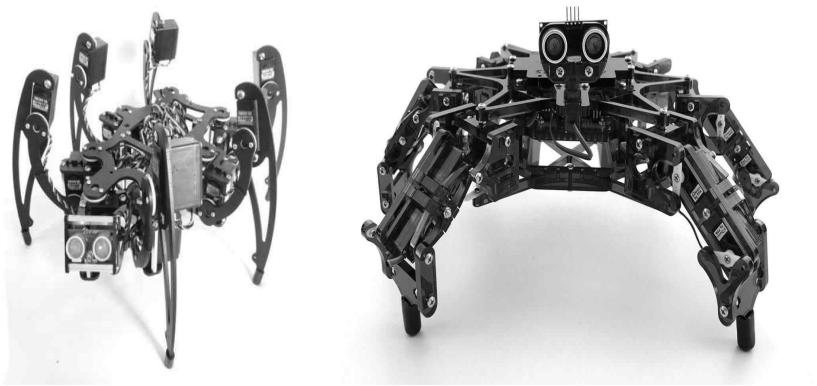


그림 1-3) 다양한 6족 보행로봇 예시
(이미지 출처 : http://arcbotics.com/products/hexy/)

2. 연구의 방향

1) 6족 보행로봇의 특징 확인

본 연구의 방향은 6족 보행 로봇의 메커니즘을 이해하고 개발하여 6족 보행 로봇이 가지고 있는 다양한 특징에 대한 확인하는 것이다. 앞서 기술한바와 같이 6족 보행로봇(이는 다족 보행로봇과 같이 적용된다)이 가지는 주된 특징은 자세의 안정성이 높다는 것이다. 이는 즉 로봇에 하중이 무거워 지거나, 다리가 파손되거나, 몸체가 비정상적으로 기울었을 때 등 다양한 환경조건에도 이동에 있어 자유도가 높다는 것이다. 이에 따라 본 연구는 로봇 메커니즘 제작 속에서 자세 안정성 및 무게중심 변화 등을 관찰하는 것을 주 연구방향으로 삼았다.

아래 사진은 본 연구에서 제작한 로봇의 개략적인 다리 구조도이다.

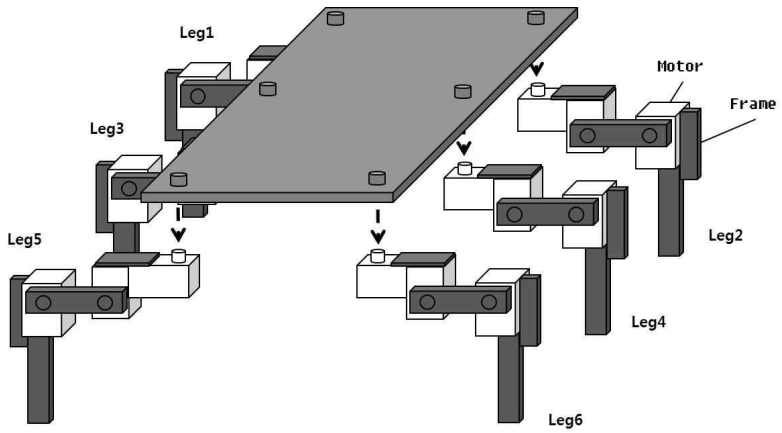


그림 1-4) 제작 로봇의 다리 구조도

각 다리 당 3개의 모터로 구성이 되어있으며, 앞 2개의 다리와 뒤 2개의 다리를 적절하게 펼쳐 로봇 다리가 원형 모양을 이루며 펼쳐서 기본자세를 잡는 방식이다. 이는 6족 보행로봇에 있어 가장 기본적인 구조중 하나이며, 특정 다리의 힘이 강하지 않더라도(혹은 다리를 들어도) 로봇 전체의 균형을 쉽게 유지할 수 있다는 점이 강점이다.

2) 보행 방식 특성 비교

6족 보행 로봇 동작을 관찰함과 함께 1족 제어 이동(다리를 1개씩 움직여 이동)과 3족 제어 이동(다리를 3개씩 움직여 이동) 2가지 이동방식을 만들어 이 두 이동에 대한 특성을 확인해보는 것이다.

먼저 아래 그림은 1족 제어 이동을 표현한 것이다.

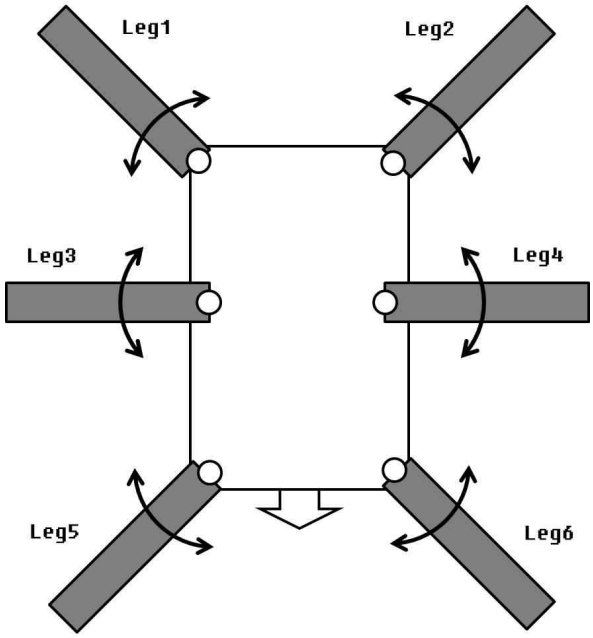


그림 1-5) 1족 제어 이동 동작

1족 제어 이동은 1개 동작 구간마다 1개의 다리가 제어되어 총 이동하는데 걸리는 시간은 길어지는데 반에, 나머지 5개의 다리가 하중을 견기기 때문에 로봇의 무게가 많이 나가거나, 이동시 몸체의 움직임을 최소화 시킬 수 있다. 이에 따라 본 연구에서는 이 동작을 총 작은 여러 동작 단계로 구성하였으며, 이전 다리의 대각선 방향의 다리를 움직이는 순서(leg1 -> leg4 -> leg5 -> ..)로 제작하여, 최대한 몸체의 움직임이 적도록 하는 것을 목표로 하였다.

다음으로 3족 제어 이동 방식은 아래 그림처럼 표현된다.

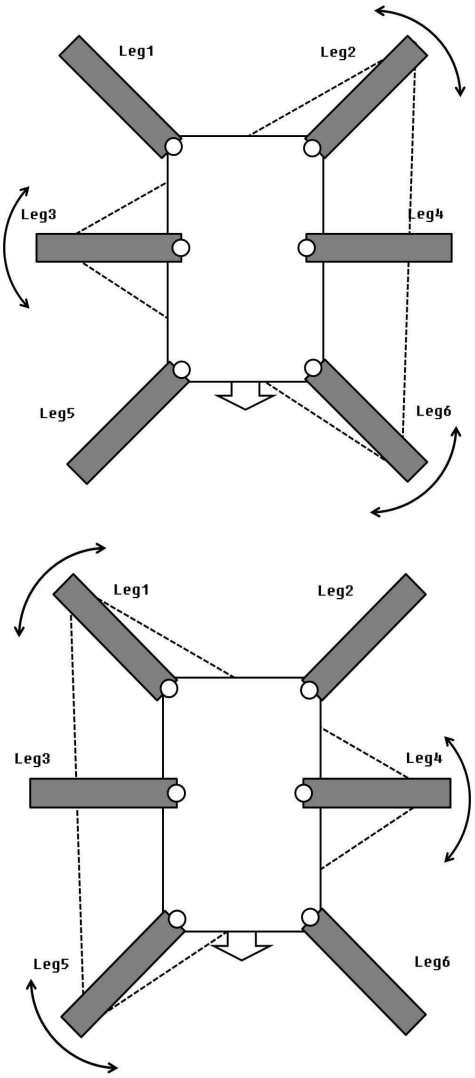


그림 1-6) 3족 제어 이동

3족 제어 이동 방식은 현재 6족 보행로봇에 있어 가장 보편적으로 알려져 있다. 왼쪽 그림처럼 하나의 삼각형처럼 묶여진 다리 3개가 동시에 동작(leg2,3,6 -> leg1,4,5)을 하여 이전 1족 제어 보행보다 총 동작 구간 수가 적어져 가장 빠른 이동 속도를 보여줄 수 있지만, 3개의 다리가 동작하는 동안 나머지 3개의 다리가 몸체의 하중을 견뎌야 하기 때문에 몸체가 무겁거나, 몸체의 균형을 유지해야하는 경우에는 부적절할 수가 있다.
또한 다리가 서로 엇갈려 동작하는 구간이 있으므로 몸체에 흔들림이 발생할 수 있다.

허나, 최근 들어 이 3족 제어 이동 방식을 개선하여, 마치 다족 동물이 움직이는 모습(하나의 다리가 동작하는 도중에 다른 다리가 동시에 동작)과 비슷하게 만들어낸 이동 방식이 있지만, 본 연구에서는 제외하기로 한다.

3. 제작 방법

1) 필요 부품 구매 및 프레임 제작

6족 보행 로봇 제작을 위한 필요 부품 및 해당 부품 가격은 아래 표로 정리하였다.

번호	부품명	필요 개수(단위: 개)	가격(개당)	사용 부분
1	NUCLEO-F103RB	1	16,000 ₩	메인 보드
2	SG-90 서보모터	18+2	3,000 ₩	바디
3	로봇 프레임	-	-	
4	기타 회로 부속(PCB, 저항 등)	-	-	
5	HC-06 블루투스 모듈	1	4,200 ₩	통신
6	폼 보드(포맥스)	1	2,000 ₩	추가 바디
7				
8				
총	-	-	-	-

그림 1-7) 필요 부품 및 가격(참고사이트 : 엘레파츠. http://www.eleparts.co.kr/)

기타 회로부속(PCB판, 저항, LED)등은 실험실 기자재를 활용하였으며, 폼 보드를 이용하여 로봇 제작에 사용될 거치대등을 제작하였다.

로봇 프레임은 3D Printer을 이용하여 제작하고, 프레임 모델링은 오픈소스를 활용하였다. 아래 사진은 제작이 완료된 6족 보행로봇 프레임이다.

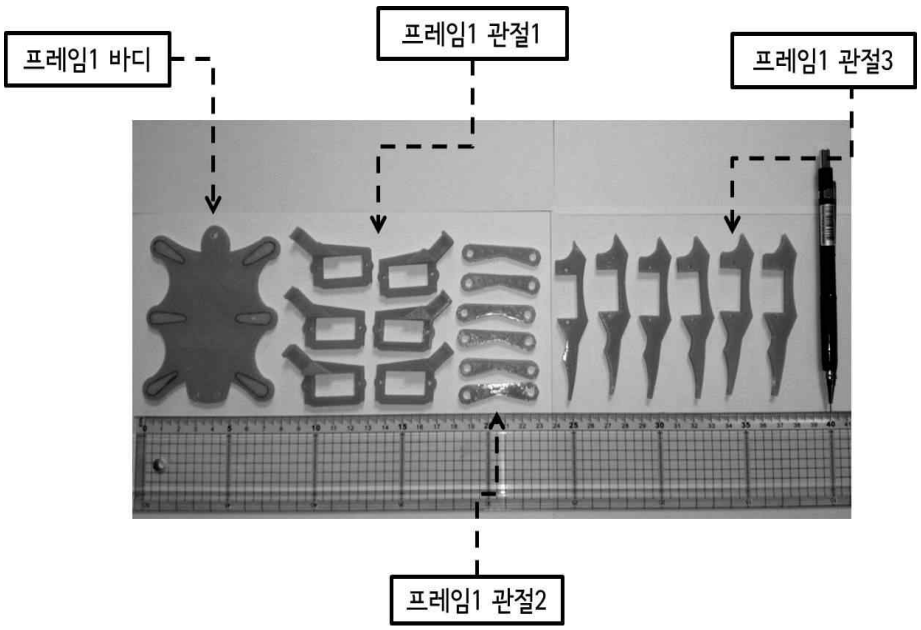


그림 1-8) 3D프린터로 제작한 프레임

2) Hardware 제작 방법

로봇 하드웨어 제작 설계도는 아래와 같다.

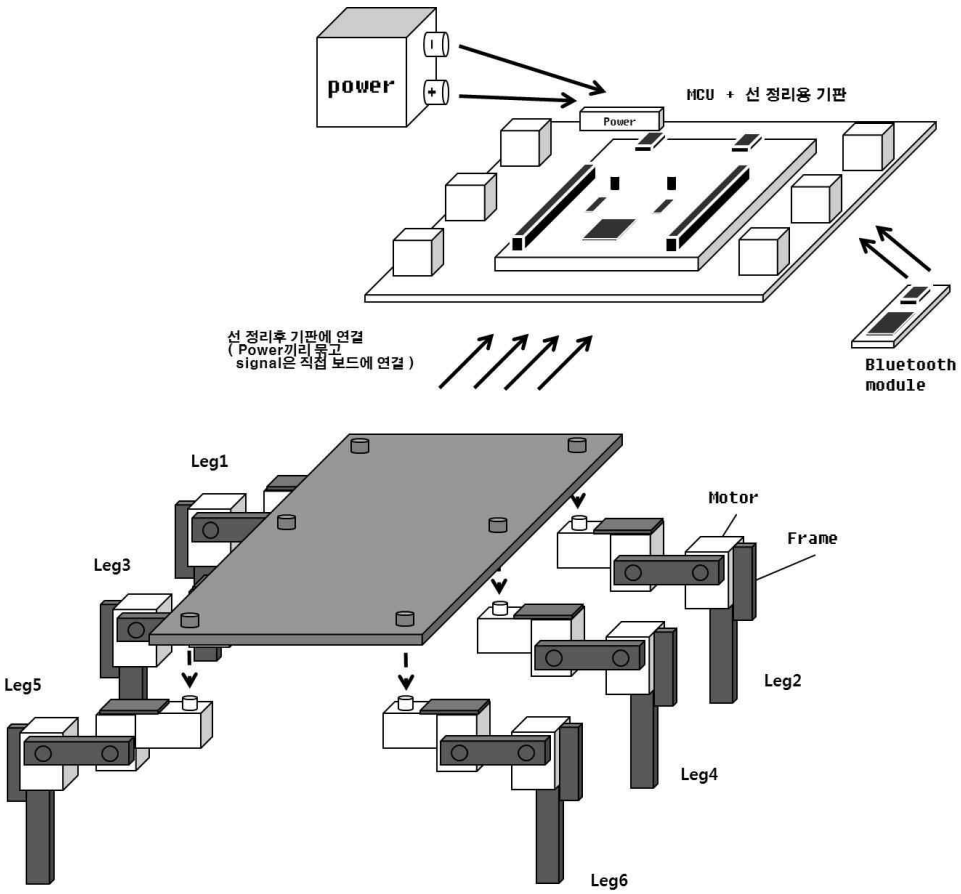


그림 1-9) 로봇 전체 설계도

프레임과 모터를 결합하여 몸체를 제작하며, MCU 및 Bluetooth과 전원을 따로 PCB판으로 묶어서 몸체와 연장된 선으로 연결하여 제작하였다. 이는 몸체에 기판을 직접 연결할 수 있지만, 하중 문제와 더불어 사용 서보모터의 낮은 토크로 인해 로봇 자세가 불안정해질 수 있기에 연장선으로 따로 연결하여, 일단은 6족 보행 메커니즘 구현에 우선을 두었다. 보다 자세한 회로도는 '제 3장 제작 과정' 에 기술하였다.

3) Software 제작 방법

로봇 소프트웨어 제작에 있어 개략적인 알고리즘은 아래 그림과 같다

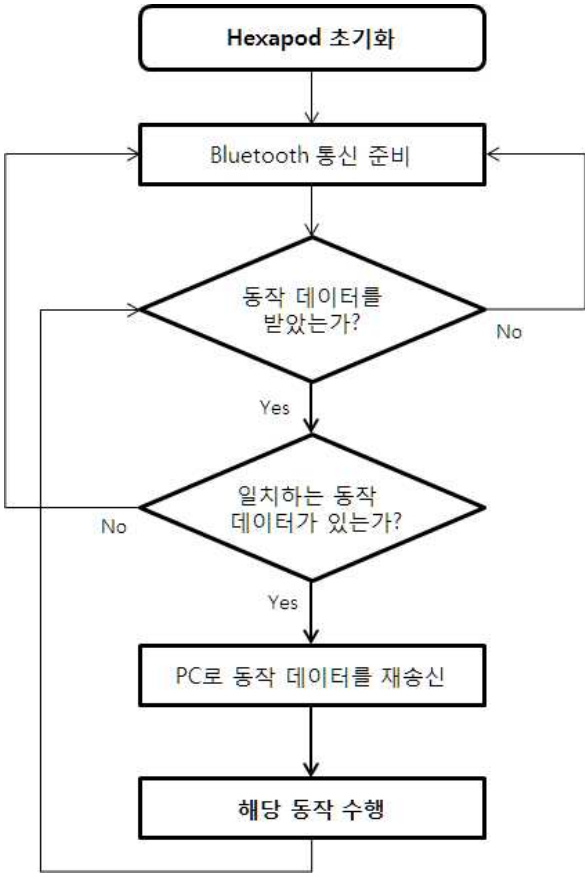


그림 1-10) 개략적인 동작 알고리즘

Bluetooth통신을 이용하여 동작하는 로봇만큼, 스마트폰-Bluetooth-MCU,PC 간의 통신을 하도록 제작하였다. 이는 스마트폰에서 보낸 동작 데이터를 토대로 MCU가 올바른 데이터인지 여부를 결정한 뒤 PC에 이 데이터를 메시지를 보내며, 해당하는 동작을 수행하는 방식으로 구성된다. 보다 자세한 소프트웨어 알고리즘은 '제 3장 제작 과정'에 기술하였다.

제 Ⅱ 장 사용 부품

1. Nucleo STM32f103RB MCU

1) 부품 소개 및 특성

Nucleo-f103RB 보드는 ST사에서 STM32F103RB 마이크로 컨트롤러를 기반으로 만든 보드이다. 보드에 사용된 STM32F103RB MCU는 AVR MCU와 다르게 32bit ARM 프로세서를 사용하기 때문에 클럭 속도 및 동작 속도가 빠른 점이 특징이다. 특히나 Nucleo-f103RB 보드는 다른 Nucleo 보드에 비해 입출력 핀이 많고, 다양한 주변 장치들이 내장되어 있으며 가격 또한 저렴하여 현재 Cortex 기반 MCU 중에서 가장 널리 쓰이고 있는 중이다. 적절한 사양과 ARM 프로세서 제작 경험을 위하여 이번 6족 보행로봇 제작에 이 보드를 사용하게 되었다.

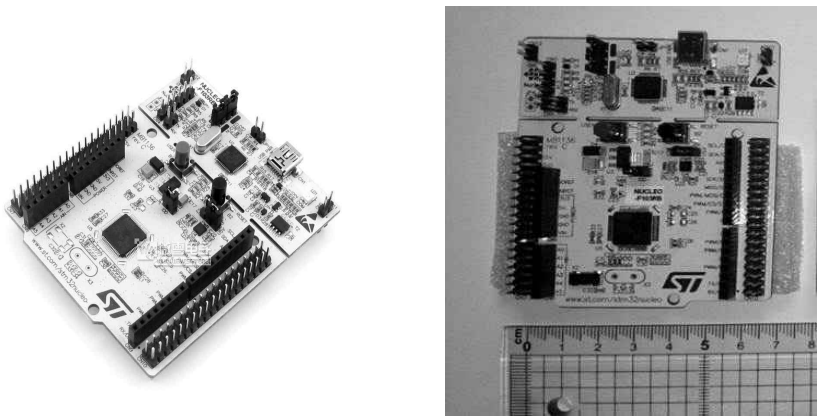


그림 2-1) 6족 보행 로봇에 사용된 Nucleo-f103RB 보드
(이미지 출처 : Stmicroelectronics)

또한, Nucleo-f103RB 보드는 메인 MCU와 함께 ST-LINK 보드가 함께 구성되어 있어 별도의 JTAG 디버거가 필요 없이 PC와 바로 연결이 가능하다는 장점을 가지고 있다.

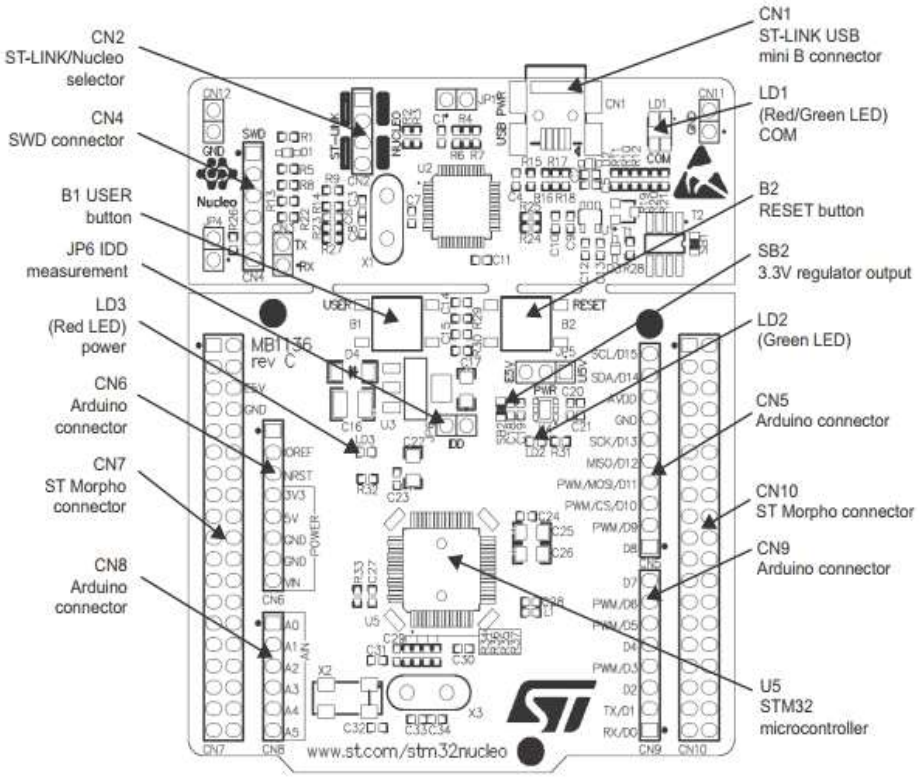


그림 2-2) Nucleo-F103RB 보드의 구조(이미지 출처 : Stmicroelectronics)

Nucleo-f103RB 보드의 상세 사양은 아래 표와 같다.

STM32F103RB 마이크로 컨트롤러 사양	
코어	ARM 32bit Cortex-M3
최대 동작 주파수	72 MHz
동작 속도	1.25 DMIPS/MHZ(Dhrystone 2.1)
메모리	64/128KB 플래시 메모리
	12KB SRAM
클럭	4 ~ 16 MHz 크리스탈 오실레이터
	8MHz 및 40Hz의 RC(Real Time) 내장
동작 전압	2.0 ~ 3.6V
입/출력(GPIO) 핀	51 pin
	16개의 외부 인터럽트 사용 가능
	대부분의 핀은 5V 입력이 허용됨
타이머	범용 타이머 3개, 고성능 타이머 1개
통신용 주변장치	USART 3개, USB 1개
	CAN 1개, SPI 2개, I2C 2개,
A/D 변환기	2개(16채널)
Nucleo-F103RB 보드 사양	
외부 연결 커넥터	아두이노 우노 호환 커넥터
	Morpho extension 핀 헤더
추가 지원	mbed 지원
	ST-Link/V2-1 디버거 내장
전원	USB의 VBUS 핀을 통한 전원공급
	아두이노 커넥터나 ST Morpho 외부 커넥터 이용(7V < VIN < 12V)
	E5V, +3.3V 핀 이용
패키지 추가 사항	3개의 LED 장착
	2개의 푸시버튼 장착
	32.768kHz 크리스탈 오실레이터 내장

표 2-1) Nucleo STM32f103RB 사양

Nucleo-F103RB 보드의 핀 구조는 아두이노 보드 호환을 위하여 순서가 섞여 있으며, 아두이노 보드 뿐만 아니라 아두이노 쉴드 모듈 또한 연결이 가능하여 다양한 외부 보드를 연결해 활용도를 높일 수 있는 장점이 있다.

보드 핀 구조에 대한 그림은 아래와 같다.

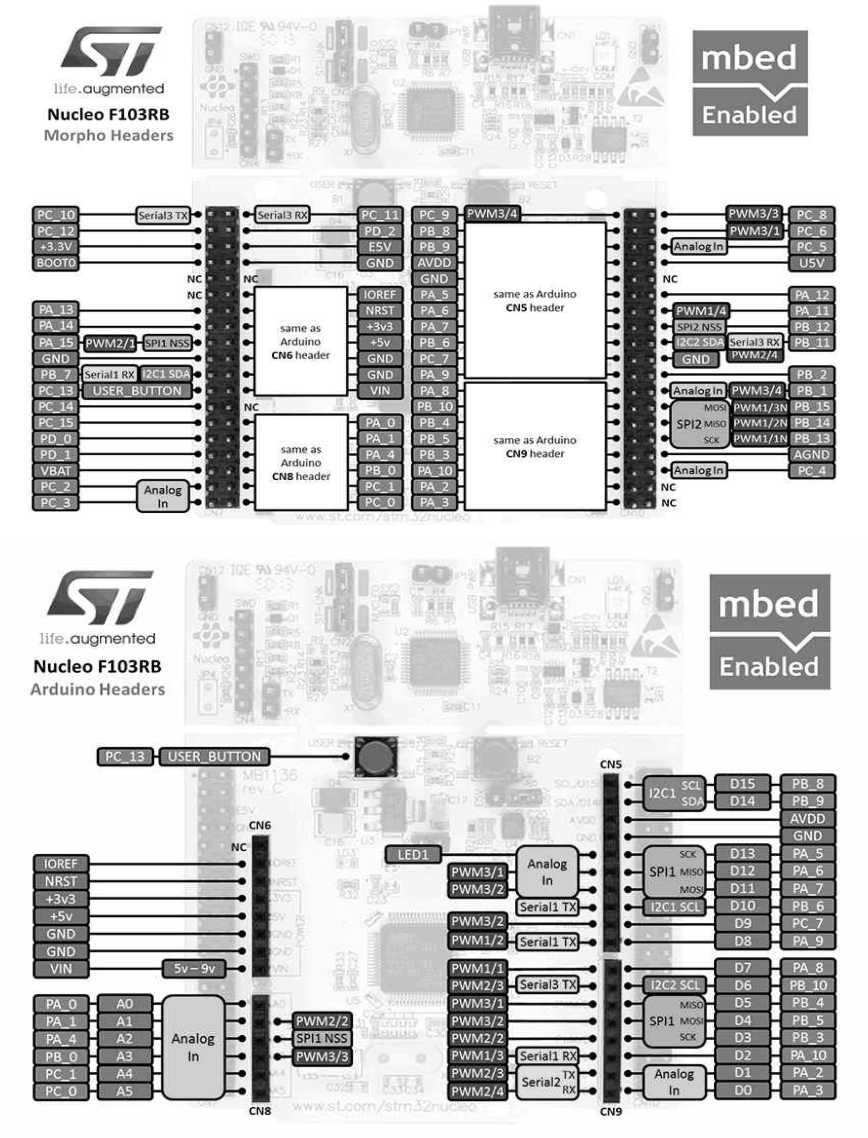


그림 2-3) Nucleo-F103RB 보드의 핀 배치도

그리고 로봇 소프트웨어 제작에 사용할 개발환경(IDE)은 ARM사에서 개발한 공식 통합 개발환경 프로그램인 ' Keil MDK '을 사용하며, 사용하는 언어는 'C/C++ ' 이다. 또한 복잡한 레지스터 구조를 가지고 있는 ARM 프로세서를 보다 쉽게 제작하기 위해 여러 함수와 구조체 등이 포함되어 있는 펌웨어 파일인 'STM32CubeF1' 을 사용하였다. STM32CubeF1 속에는 ARM 프로세서 레지스터를 'HAL Driver' 라는 드라이버를 통해 레지스터 값을 직접 써넣지 않고 구동용 함수로 불러와 사용하게 된다. 이로써 개발자는 프로그램 작성 및 디버깅이 아주 쉬워지게 되는 것이다.

이 HAL Driver 속 구동용 함수는 구조체 형태로 구성 되어 있어, 프로그램 제작 시 구조체의 값을 변경하게 되면 자동으로 해당하는 기능의 레지스터 값이 변경되게 된다.

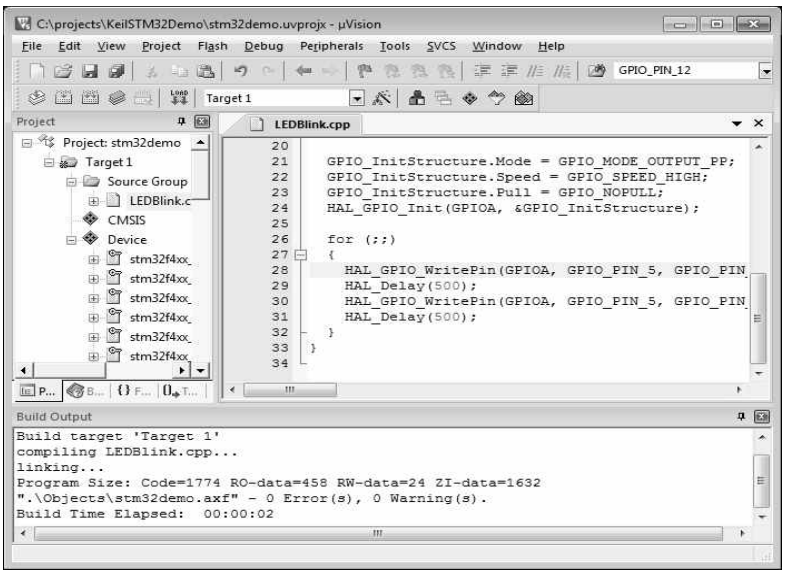


그림 2-4) Keil MDK 및 HAL Driver 사용 예시
(이미지 출처 : keil.com)

2. 서보 모터(SG-90)

1) 부품 소개 및 특성

6족 보행 로봇을 구성하는데 있어 가장 중요한 부품 중 하나는 바로 모터이다. 본 연구에서 제작한 로봇에 사용된 모터는 'SG-90 ' 으로 저렴한 가격에 사용하게 편한 구조로 되어 있어 초기 타입 로봇 제작에 인기 있는 모터이다.

서보모터란 일반 DC모터와 다르게 Pulse 입력 핀이 있어, 이 입력 핀에 PWM 파형형태의 전압을 가하면, 해당하는 각도로 모터가 회전하게 된다. 아래는 본 로봇에 사용된 'SG-90 ' 모터와 작동 방식 특성, 사양을 나타낸다.

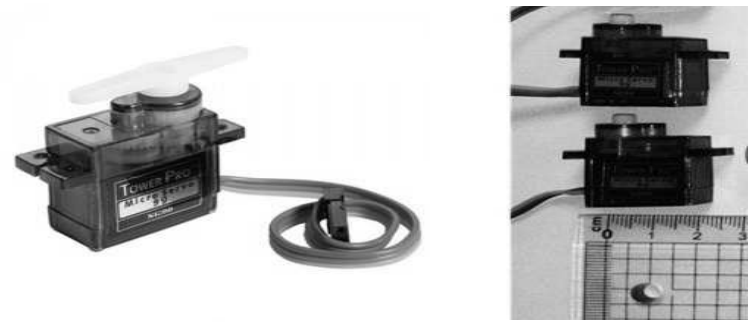


그림 2-5) SG-90 서보 모터
(이미지 출처 : www.eleparts.co.kr/EPXCDPFJ)

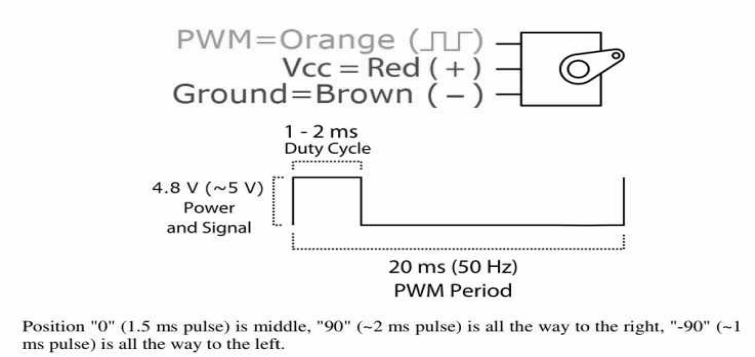


그림 2-6) SG-90 서보모터의 작동 방식

무게	9g
규격	22.2 x 11.8 x 31 mm
정지 토크	1.8 kgf · cm
동작 속도	0.1 s/60 degree
동작 전압	4.8V(~5V)
Dead band width	10 us
작동 온도	0℃ ~ 55℃

표 2-2) SG-90 서보모터 사양

그림에서 보여지듯이 'SG-90 ' 서보모터는 2개의 전원 핀(Vcc,GND) 과 1개의 PWM핀으로 구성되어 있다. 총 20ms(50Hz)의 길이를 가지며, 1~2ms 의 Duty Cycle, 4.8 ~ 5V 크기를 가지는 PWM 핀 입력에 따라 모터가 움직이게 된다. 따라서 서보모터에 대한 기본 라이브러리가 없는 Nucleo-F103RB 보드에서는 모터를 사용하기 전에 신호에 알맞게 타이머 설정을 해준 뒤 원하는 각에 해당하는 PWM 파형을 만들어 줘야 한다. 이에 대한 자세한 내용은 다음 장에 기술하였다.

3. BlueTooth module(HC-06)

1) 부품 소개 및 특성

본 연구에서 제작한 6족 보행로봇은 기본적으로 PC와의 시리얼 통신 및 스마트폰과의 블루투스 통신으로 작동하기 때문에 블루투스 모듈인 'HC-06'를 사용하였다. 'HC-06'은 TX(데이터 송신),RX(데이터 수신),VCC,GND로 총 4개의 핀으로 구성되어 있어 사용하는 Nucleo 보드에 직접 연결하여 사용하였다. 이때 통신 환경 설정은 9600bps, 8bit 통신, stop bit = 1로 하였다. 아래는 'HC-06' 사진 및 사용예시, 부품 상세 사양이다.

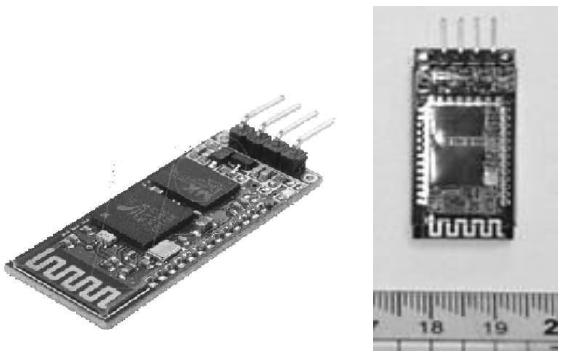


그림 2-7) 블루투스 모듈 'HC-06'

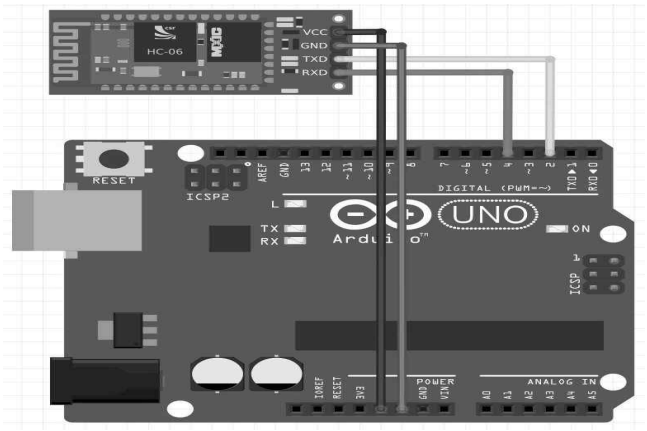


그림 2-8) HC-06 사용 예시

내부 구조	2.4GHz 안테나 내장
	디지털 2.4GHz 무선통신
	외부 8Mbit FlASH
	Bluetooth Class 2
동작 전압	3.3VDC ~ 5.0 VDC
동작 온도	-25℃ ~ 75℃
Default Pairing	name : HC-06
	Serial Port : 9600,N,8,1
	Pairing Code : 1234
동작 거리	최대 10M
핀 구조	RX,TX,Vcc,GND

표 2-3) HC-06 상세 사양

또한 스마트폰에 사용될 앱은 'Bluetooth Serial Controller ‘이라는 무료 어플리케이션을 활용하였다. 이 어플리케이션을 통해 사전에 설정한 데이터를 보내게 되면 Nucleo 보드가 이를 수신하여 로봇이 움직이게 된다. 아래 사진은 동작을 위한 설정이 완료된 어플리케이션 사진이다.

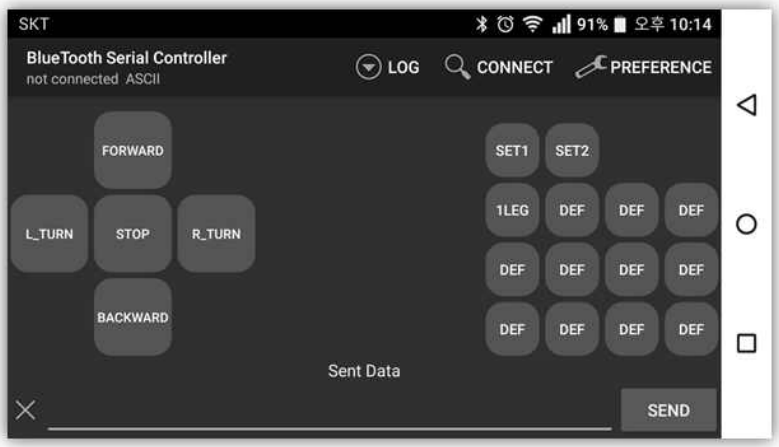


그림 2-9) Bluetooth Serial Controller 앱 화면

제 Ⅲ 장 제작 과정

1. Hardware

1) 제작 설계도

아래 사진은 본 연구에 만든 6족 보행로봇의 전체 설계도 및 구성 회로도이다. (회로도 제작 : Fritzing)

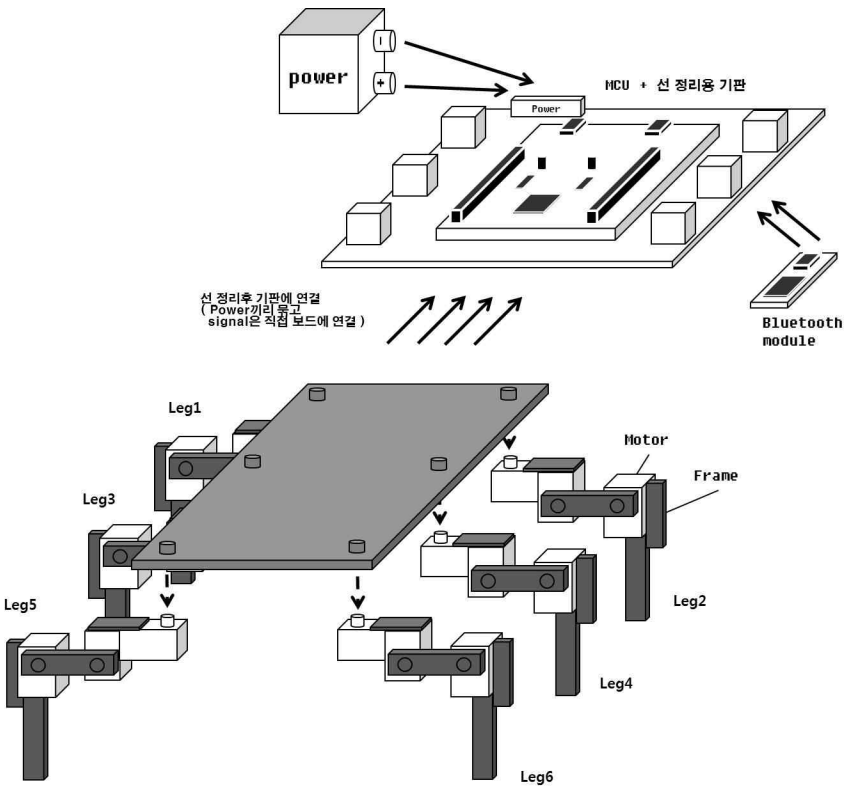


그림 3-1) 6족 보행 로봇 전체 설계도

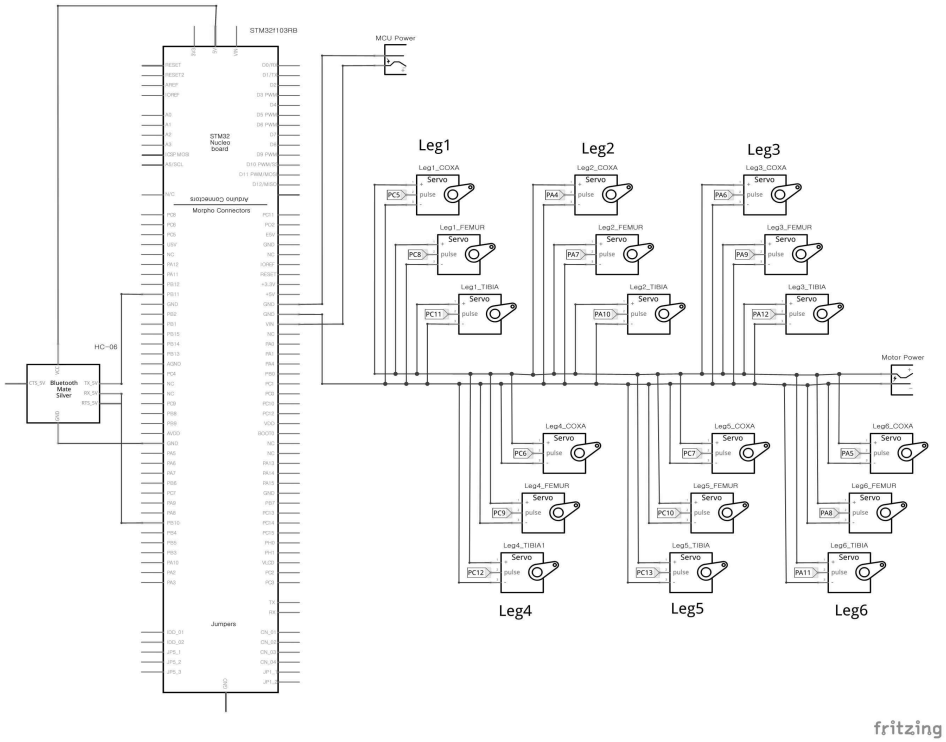


그림 3-2) 구성 회로도

회로도를 보면, Nucleo 보드에 블루투스 묘듈 및 전원부는 아래표 같이 연결 되었다.

항목	이름	핀 번호
블루투스 연결부	RX	PB10
	TX	PB11
	Vcc	5V
	GND	GND
전원부	+ 전원	Vin
	- 전원	GND

표 3-1) 블루투스, 전원부 연결핀

로봇 몸체에 있는 서보모터는 각 신호 핀을 보드에 직접 연결하였는데, 다리 구조를 아래 그림처럼 명시하였을 때, 다리를 구성하는 모터의 신호 핀이 연결된 보드의 핀 번호는 다음 표와 같다.

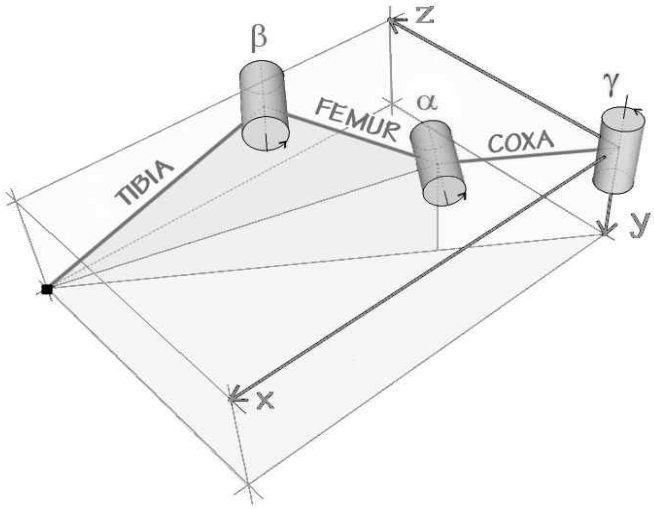


그림 3-3) 다리 구조 명칭
(이미지 출처 : https://oscarliang.com/)

다리 번호	COXA	FEMUR	TIBIA
Leg1	PC5	PC8	PC11
Leg2	PA4	PA7	PA10
Leg3	PA6	PA9	PA12
Leg4	PC6	PC9	PC12
Leg5	PC7	PC10	PC13
Leg6	PA5	PA8	PA11

표 3-2) 로봇 다리 연결핀

로봇 다리에 사용된 총 핀 개수는 PA4 ~ PA12, PC5 ~ PC13 으로 총 18개가 사용 되었다.

2) Hardware 완성

아래 사진은 Hardware가 완성된 6족 보행 로봇이다.
(크기 : 가로 30cm / 세로 30cm / 높이 10com)

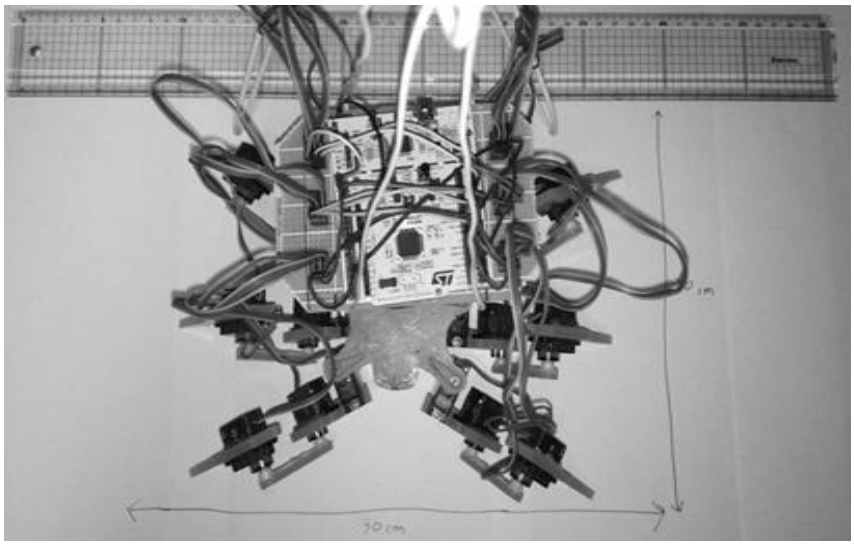


그림 3-4) Hardware 제작이 끝난 로봇(상단)

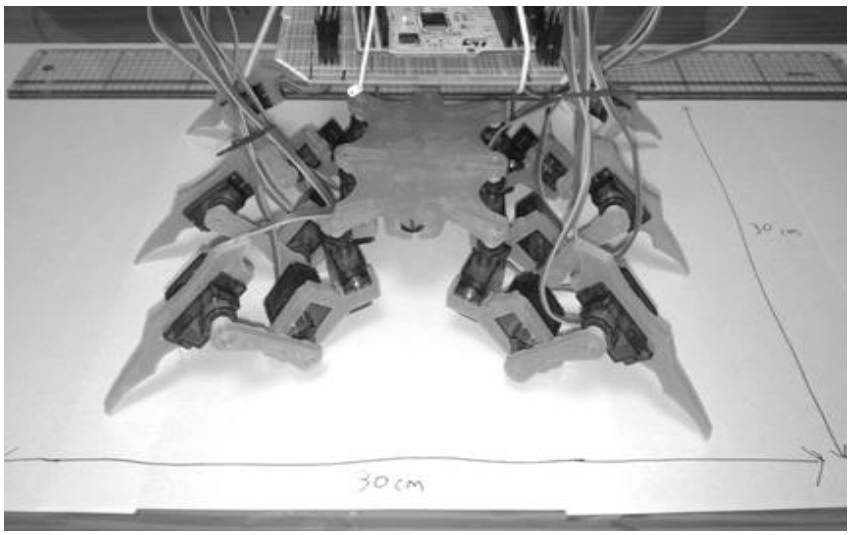


그림 3-5) Hardware 제작이 끝난 로봇(정면)

2. 메커니즘

본 연구에서 제작한 6족 보행 로봇이 가지고 있는 동작 1족 제어 전진, 3족 제어 전진, 좌회전, 우회전으로 총 4가지이다.

1) 1다리 제어 전진

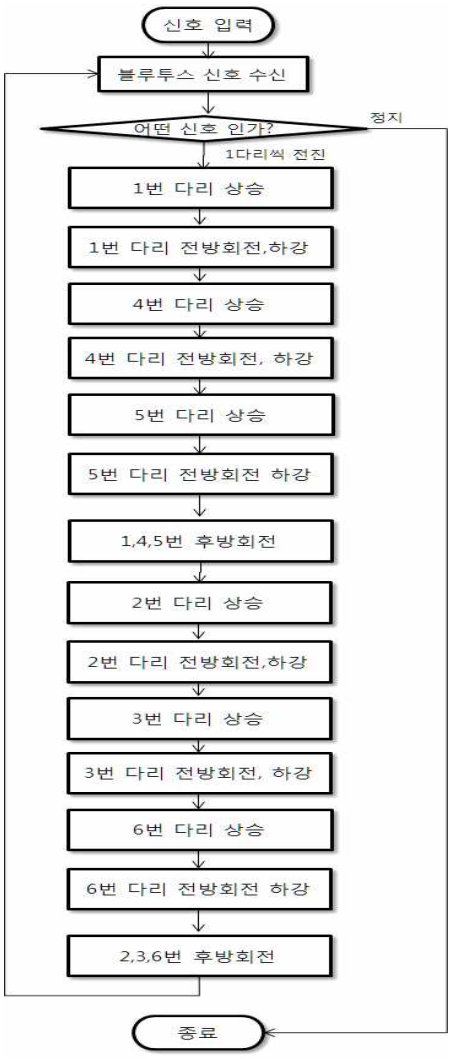


그림 3-6) 1족 제어 전진 알고리즘

로봇이 1족 제어 전진을 하기 위한 단계는 아래와 같다.

1단계) 1,4,5번 다리가 차례대로 올라간 뒤 이동하고자하는 로봇의 앞쪽 방향으로 회전한 후 내려간다. 이때, 이후 몸체를 밀기 위하여 이 다리들이 몸체를 조금 들어 올린다.

2단계) 앞쪽으로 움직인 1,4,5번 다리가 다리 뒤로 움직이며 몸체를 밀어 앞으로 움직인다.

3단계) 2,3,6번 다리도 마찬가지로 차례대로 상승, 앞으로 이동, 하강을 한다. 이때도 몸체를 조금 들어 올린다.

4단계) 앞쪽으로 움직인 2,3,6번 다리도 마찬가지로 뒤로 움직여 몸체를 밀어 앞으로 움직인다.

아래 그림은 1족 제어 전진 과정을 로봇으로 그려낸 것이다.

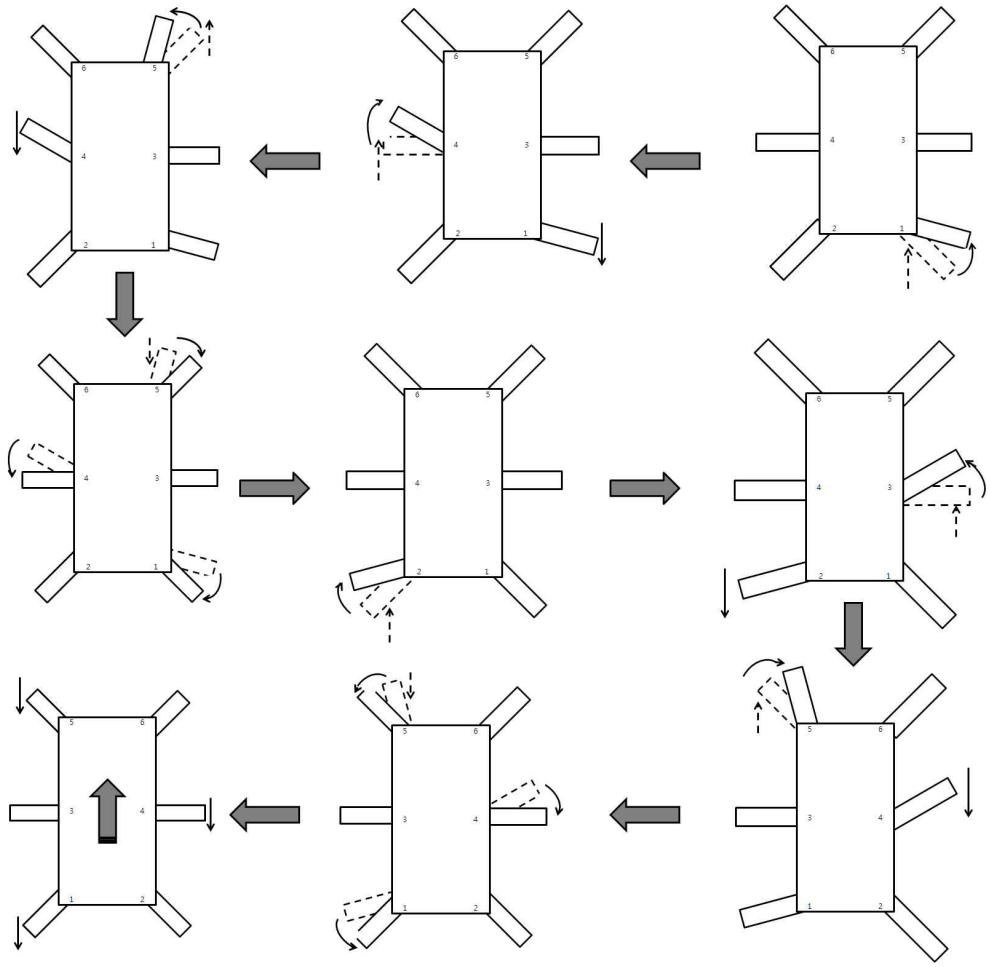


그림 3-7) 1족 제어 전진 과정

2) 3족 제어 전진

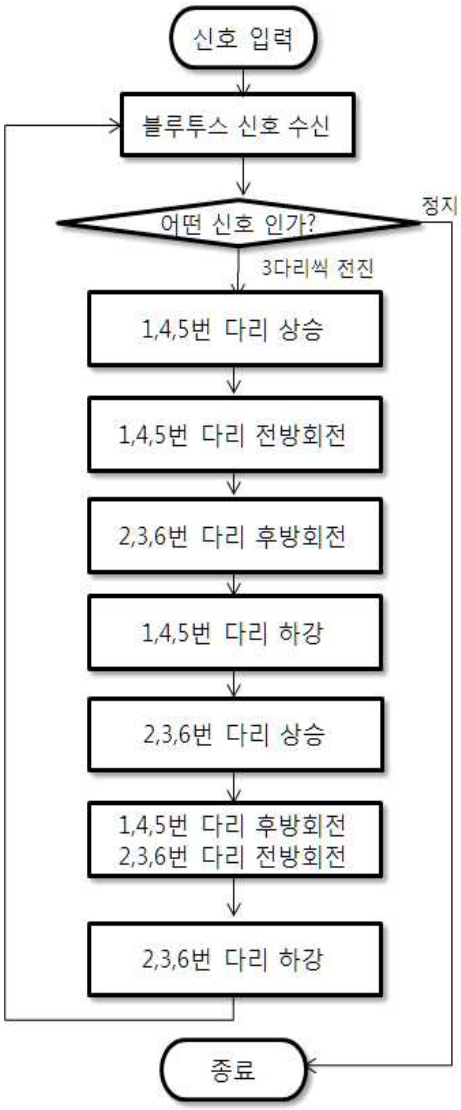


그림 3-8) 3족 제어 전진 알고리즘

로봇이 3족 제어 전진을 하기위한 단계는 아래와 같다.

1단계) 1,4,5번 다리가 올라간 뒤 로봇이 움직이고자 하는 앞쪽 방향으로 회전한다.

2단계) 2,3,6번 다리를 뒤쪽으로 회전시킨다.

3단계) 올라간 1,4,5번 다리를 다시 내린다.

4단계) 2,3,6번 다리를 올린 뒤,앞쪽으로 회전시킴과 함께, 1,4,5번다리를 뒤쪽으로 회전시켜 로봇이 전진을 한다.

5단계) 2,3,6번 다리를 내려 원래상태로 돌아오게 된다.

아래 그림은 3족 제어 전진 과정을 로봇으로 그려낸 것이다.

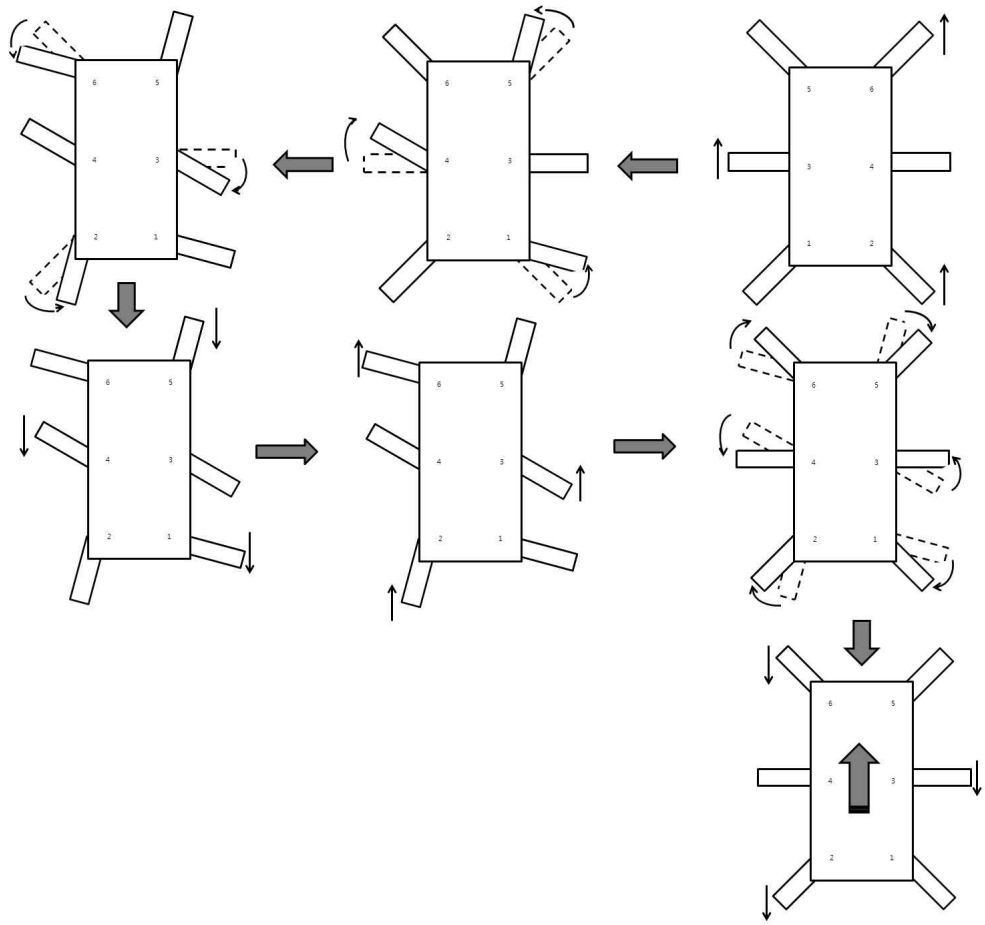


그림 3-9) 3족 제어 전진 과정

3) 좌회전, 우회전

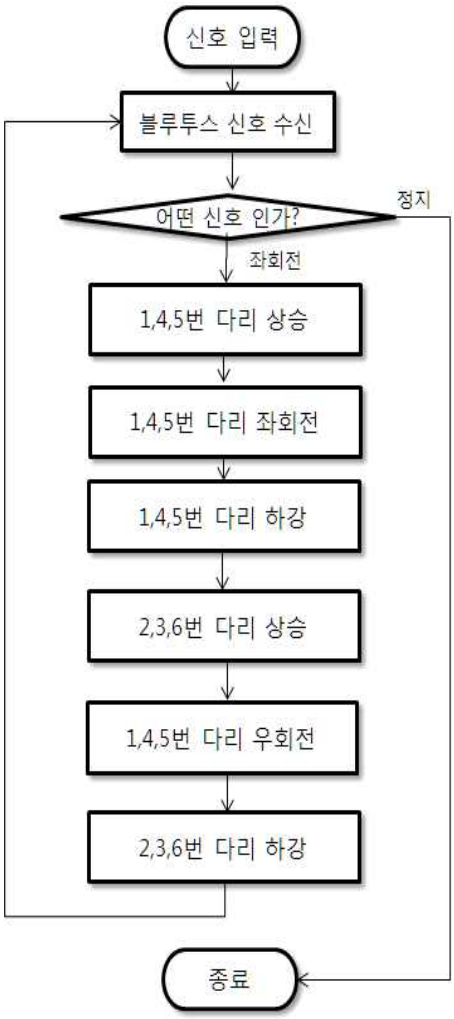


그림 3-10) 좌회전 알고리즘

로봇이 좌회전을 하기위한 단계는 아래와 같다.

1단계) 1,4,5번 다리를 올린 뒤, 회전하고자하는 왼쪽방향으로 회전하고 다시 다리를 내린다.

2단계) 2,3,6번 다리를 올린 뒤, 1,4,5번 다리를 회전하고자 하는 방향 반대인 오른쪽으로 회전하여, 몸체 전체가 왼쪽으로 회전한다.

3단계) 올라갔던 2,3,6번 다리를 다시 내린다.

그림 3-11) 우회전 알고리즘

다음으로 로봇이 우회전을 하기위한 단계는 아래와 같다.

1단계) 2,3,6번 다리를 올린 뒤, 회전하고자하는 오른쪽방향으로 회전하고 다시 다리를 내린다.

2단계) 1,4,5번 다리를 올린 뒤, 2,3,6번 다리를 회전하고자 하는 방향 반대인 왼쪽으로 회전하여, 몸체 전체가 오른쪽으로 회전한다.

3단계) 올라갔던 1,4,5번 다리를 다시 내린다.

아래 그림은 좌회전과 우회전 과정을 로봇으로 그려낸 것이다.

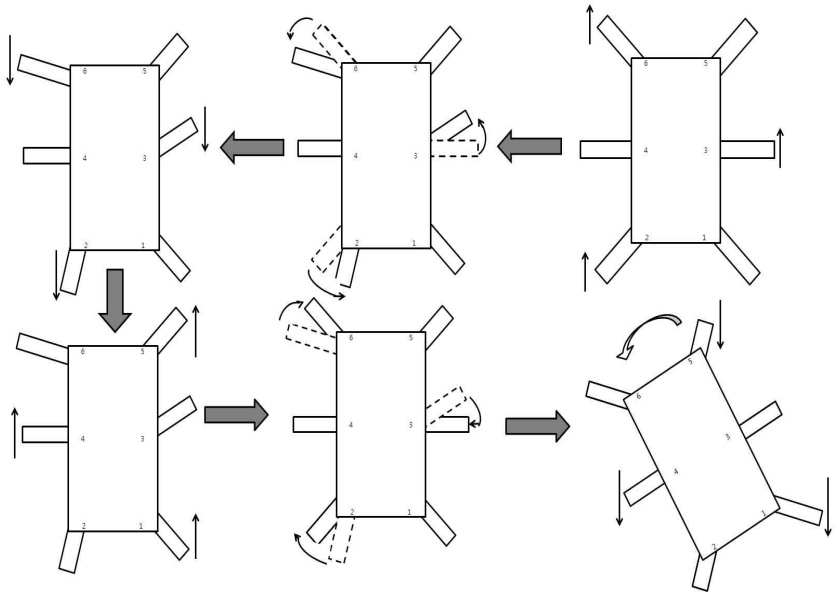


그림 3-12a) 왼쪽 회전 동작 과정

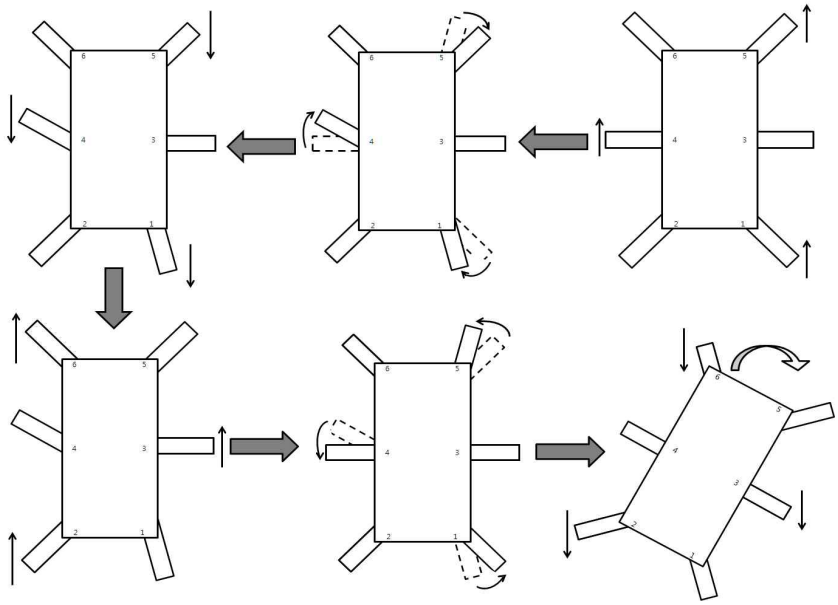


그림 3-12b) 오른쪽 회전 동작 과정

2. Software

1) Code 알고리즘

본 연구를 위해 제작한 6족 보행로봇에 사용한 전체적인 Code 알고리즘을 정리한 그림은 아래와 같다.

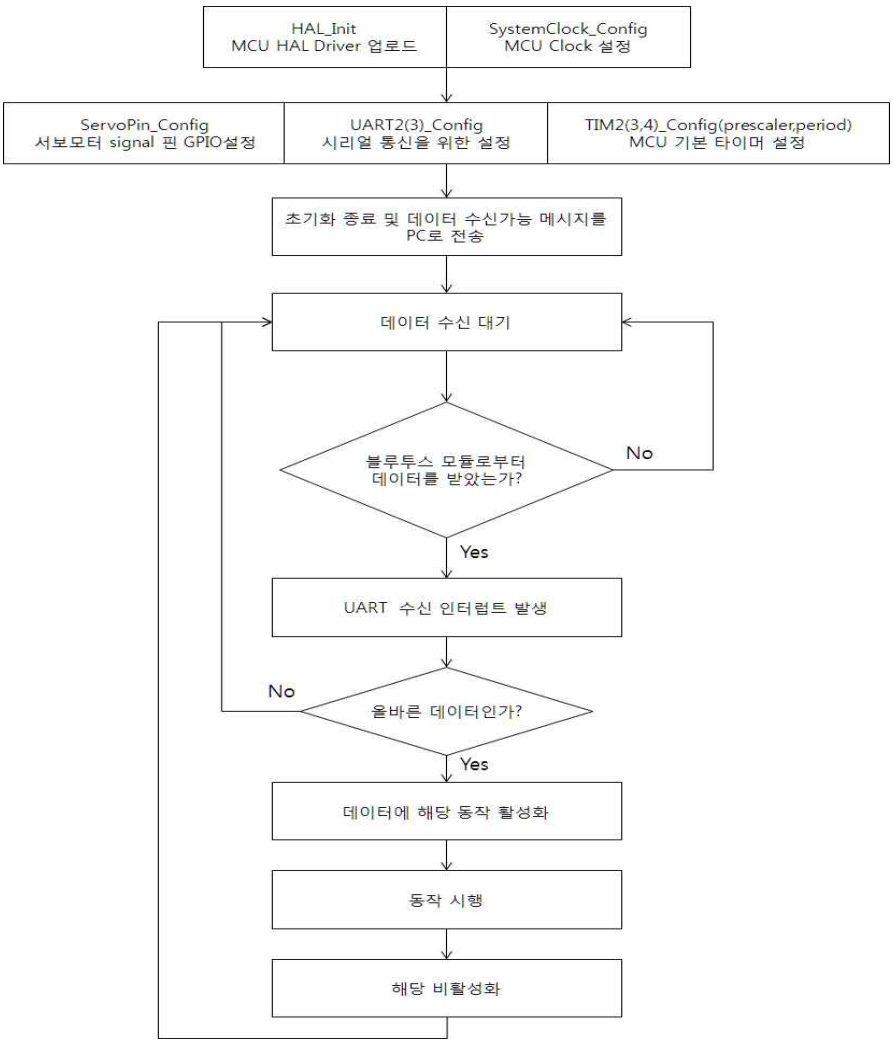


그림 3-13) 전체적인 코드 알고리즘

전체 코드 알고리즘에 대해 단계별로 설명하자면,

1단계) HAL_Init()
- 이는 MCU에 HAL Driver를 업로드하여 HAL 함수를 사용할 수 있게 해주는 초기 설정이다.
SystemClock_Config()
- MCU의 클럭 주파수 및 기능을 초기화 시킨다.

2단계) ServoPin_Config()
- 서보모터에 사용 될 Port A와 Port B를 활성화 시키며, 출력모드로 초기화 한다.
UART2(3)_Config(0
- 블루투스와 PC간의 시리얼 통신을 위하여 UART2(블루투스)와 UART3(PC)를 활성화 및 초기화 시킨다(9600dps,8bit data, 1bit stopbit)
TIM2(3,4)_Config(prescaler,period)
- 서보모터 동작에 사용될 타이머2,3,4를 활성화하며, prescaler = 127 (서보모터 동작을 위한 50Hz를 맞추기 위해), period = 9999(변화없음) 로 초기화한다.

3단계) 초기화가 모두 끝나면, UART3를 통해 PC로 로봇이 동작할 준비가 되었다고 메시지를 전송한다.

4단계) 로봇이 블루투스를 통해 스마트폰으로부터 데이터를 수신받기를 기다리며, 만약 로봇이 데이터를 받았을 시 UART 수신 인터럽트를 통해 올바른 데이터가 수신되었는지 판단한다.

5단계) 올바른 동작 데이터라 확인이 되면, 로봇은 해당 동작을 활성화후 동작을 시행 그리고 다시 동작을 비활성화한다(이는 동작이 중복됨을 방지함이다.)

이렇게 총 5단계의 알고리즘을 거쳐 6족 보행로봇은 초기화부터 동작까지 하게 되는 것이다.

동작 코드 상세설명에 앞서, 이번에 제작한 6족 보행로봇은 6족 3관절이므로 총 18개의 서보모터가 동작을 하여야한다. 즉, 18개의 타이머로 만들어진 PWM 신호를 각 모터에 가해야 로봇이 동작을 하는 것이다. 하지만 STM32F103RB MCU의 경우 총 4개의 타이머와 각각의 타이머는 4개의 채널을 가지고 있으므로 최대 16개의 PWM 신호를 만들어 낼 수 있으므로 결국 2개의 PWM 신호가 부족해진다. 이를 해결하기 위하여 여러 PWM 신호를 만들어 낼 수 있는 'Servo Motor Controller'를 사용하는 것도 방법이지만, 본 연구에서는 컨트롤러를 사용하지 않고 9개(타이머 3개, 채널 3개)의 PWM 신호를 빠르게 2번, 서로 다르게 동작하여, 마치 18개의 PWM 신호처럼 보이게 제작하였다. 아래 그림은 6개의 다리를 삼각형 모양으로 묶어 2개의 Part로 구분하여 각각 9개의 타이머 채널을 어떻게 적용했는지 표현한 것이다.

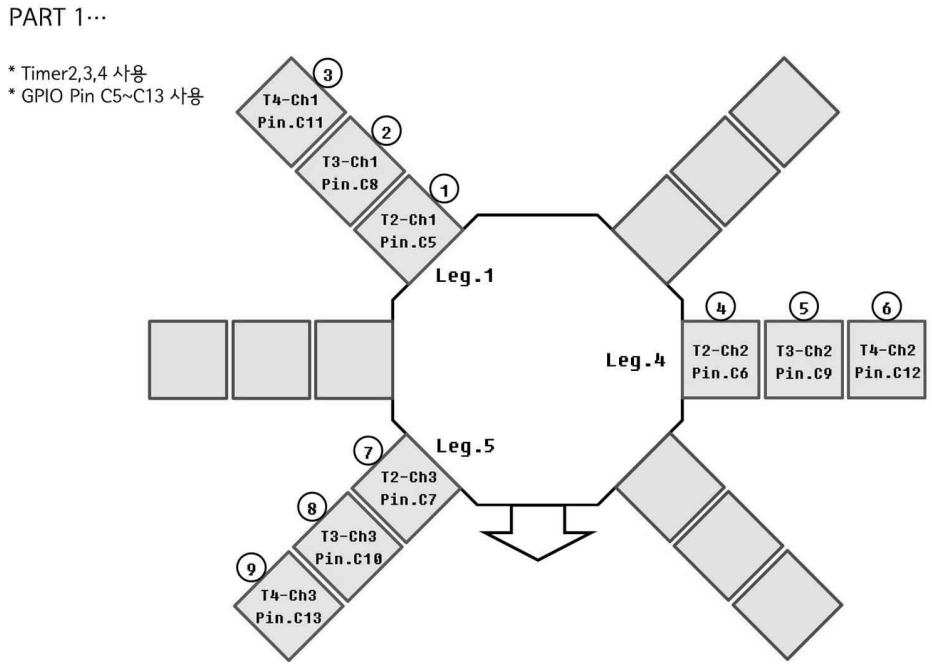


그림 3-14) 6족 보행로봇 다리를 구성하는 Part1

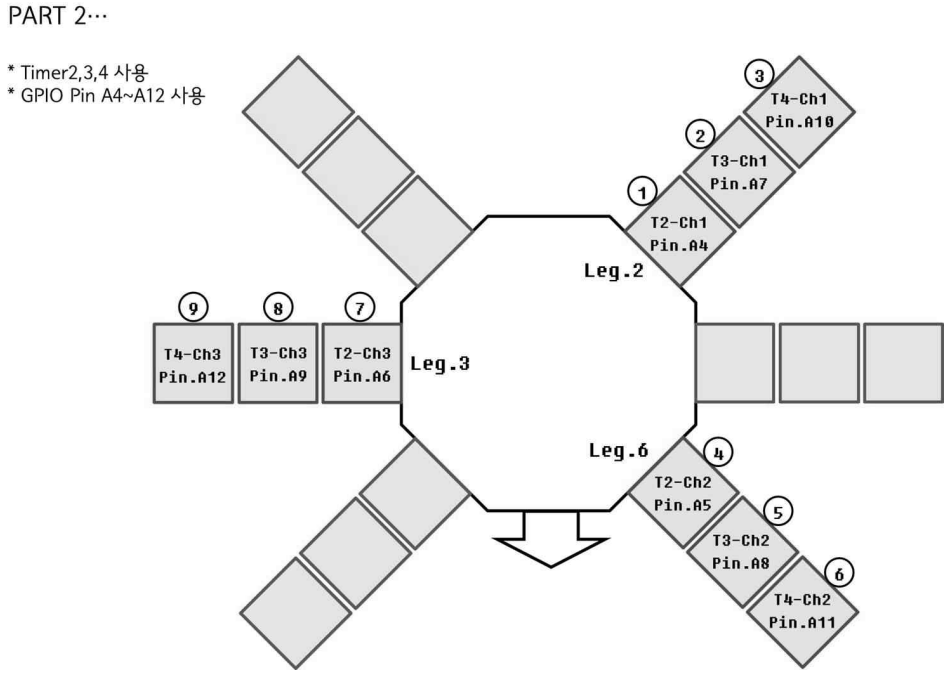


그림 3-15) 6족 보행로봇 다리를 구성하는 Part2

위 그림에서 보이듯이 타이머2 ~ 타이머4와 채널1 ~ 채널3이 Part에 따라 2번씩 사용된 것을 볼 수 있다. 이로써

(타이머 3개 x 채널 3개) X Part 2개 = 18개의 PWM 신호

를 만들어 낼 수 있는 것이다. 이로 인해 왜 앞서 알고리즘에서 총 3개의 타이머(각 3개의 채널)만 사용되었음을 알 수 있다.

2) 주 코드 설명

서보모터를 움직이기 위해서는 PWM 신호가 필요한데, 앞장에서 설명하였듯이, Nucleo-F103RB 보드에는 서보모터에 관한 기본 라이브러리가 없어 본 로봇 제작에서는 직접 파형을 만들어 동작하게 하였다. 아래 그림은 서보모터 동작을 위한 PWM파형 제작 과정이다.

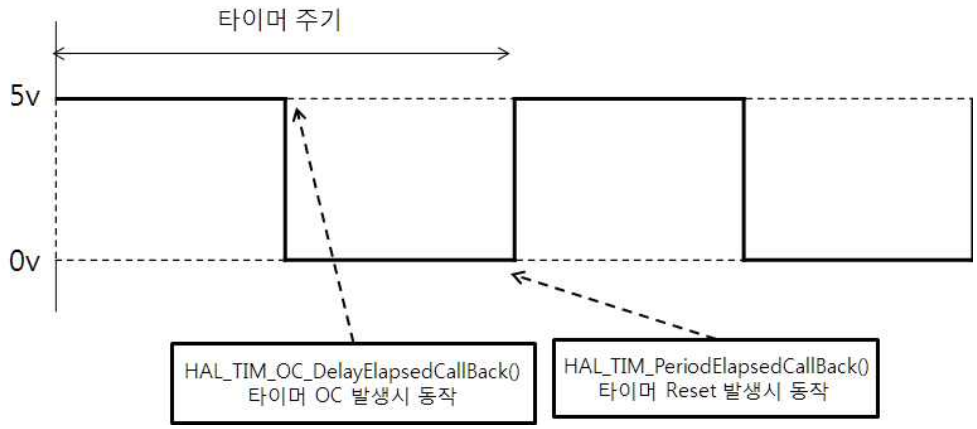


그림 3-16) 서보모터 PWM 파형 제작 과정

타이머 주기는 SG-90 서보모터에 맞춰 50Hz를 이전 타이머 초기화를 통해 설정하였으며, PWM 파형을 만들기 위하여, 타이머가 매회 리셋 시 시작 값을 HAL_TIM_PeriodElapsedCallBack() 함수를 통해 출력을 High (5v)로 설정하고, HAL_TIM_OC_DelayElapsedCallBack() 함수를 이용해 비교 값(여기서는 원하는 서보모터 각도를 위한 값)과 일치하는 경우 출력을 Low(0v)로 낮추어 일정한 Duty비를 갖는 PWM 파형을 만들었다. 이때 입력되는 비교 값을 매 동작마다 변경해야 하므로 이를 함수화 시킨 것이 바로 모터 동작에 주로 사용될
TIM_OC_Config(TIM_HandleTypeDef handle_n, int pulse_value1,
int pulse_value2,int pulse_value3)
함수이다. 이 함수는 순서대로 타이머 번호, 다리의 COXA값, 다리의 FEMUR값, 다리의 TIBIA 값을 입력 받아 해당하는 다리가 동작하도록 구성되어 있다.

또한 이를 한번 더 정리하여, leg_number[]={coxa,femur,titia} 배열로 원하는 다리 각도를 저장한 뒤, Leg_Angle_Set0() 또는 Leg_Angle_Set1() 함수를 통해 다리가 저장한 각도대로 움직이도록 설계하였다.(Part에 따라 달라짐)

이후로는 앞서 설명한 메커니즘을 토대로 다리 각도를 적절히 선택하여, 원하는 동작으로 움직이도록 반복하는 것이 로봇의 동작 방식이다.

```
void Leg_Reset2(){

   leg_1[0] = 850; leg_1[1] = 800; leg_1[2] = 600;
   leg_4[0] = 750; leg_4[1] = 700; leg_4[2] = 550;
   leg_5[0] = 650; leg_5[1] = 800; leg_5[2] = 550;
   Leg_Angle_Set0();

   part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

   leg_2[0] = 850; leg_2[1] = 800; leg_2[2] = 600;
   leg_3[0] = 750; leg_3[1] = 900; leg_3[2] = 650;
   leg_6[0] = 650; leg_6[1] = 800; leg_6[2] = 550;
   Leg_Angle_Set1();

   part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

   |
}
```

그림 3-17) 다리 리셋2 동작 코드

위 그림은 다리 리셋2 동작 코드이며 내용을 살펴보면, 이전 설명과 같이 Leg_number[] 배열로 각도를 저장한 뒤, Leg_Angle_Set0(), Leg_Angle_Set1() 함수로 동작을 시킨다. 이때, Part가 옮겨질 때마다 Time_Oc_It_Stop() 함수로 이전 Part에 사용된 타이머를 완전 중지시켜, 이전에 남아있는 타이머 PWM파형의 영향을 끼지 않도록 하였다.

또한, 각 다리에 저장되는 각도는 300~1200의 값이 저장되며, 이는 0° ~ 180° 와 대칭되는 각도임을 주의해야한다. 그리고 로봇을 기준으로 좌우대칭되는 모터구조를 참고하여 미리 계산 값을 재계산 하므로 이와 관계없이 모든 다리가 공통으로 값이 줄어들수록 다리의 COXA부분은 앞쪽으로, FEMUR부분은 위로 올라가며, TIBIA부분은 로봇 안쪽으로 움직이게 된다.

예를 들어 leg2[0]=850은 leg2[0]=650이 되면 (850-650)/(900/180) = 약 40° 앞으로 움직이며, leg2[1]=800은 leg2[1]=600이 되면 (800-600)/(900/180) = 약 40° 위로, leg2[2]=600은 leg2[2]=500이 되면 (600-500)/(900/180) = 약 20° 안쪽으로 움직이게 되는 것이다. 아래 그림은 위 예시를 보기 쉽게 그린 것이다.

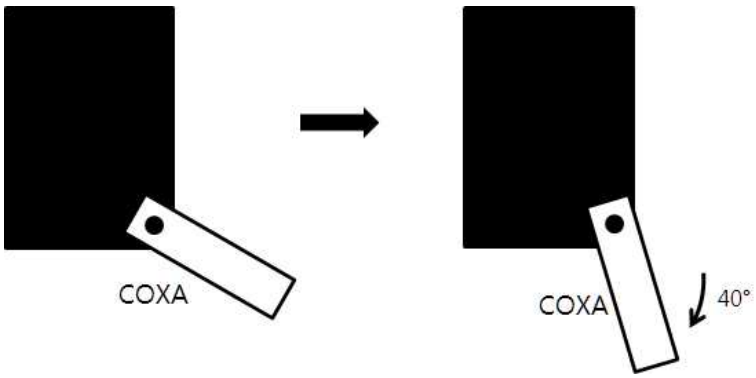


그림 3-18) COXA 부분 변화(850 -> 650)

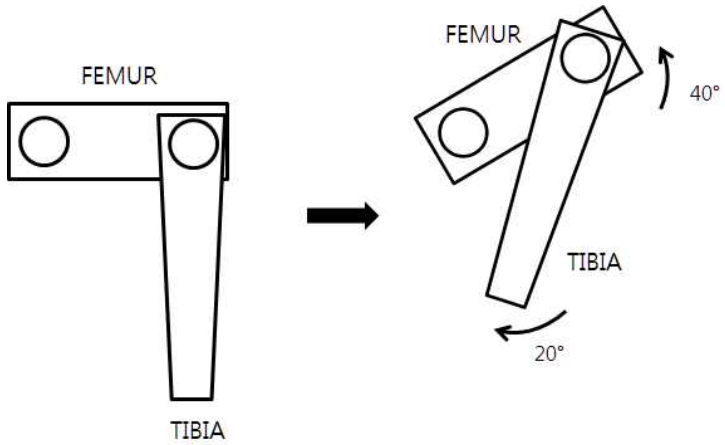


그림 3-19) FEMUR, TIBIA 부분 변화(800 -> 600, 600 -> 500)

이후 나머지 동작들 역시 각도 설정, 다리 동작, (타이머 중지), 각도 설정, 다리 동작.... 을 반복 하는 구조로 이루어져 있다. 아래 그림들은 리셋2 동작 이외 왼쪽, 오른쪽 회전 동작 코드와 3족 제어 전진 코드이다.

```
void Leg_test3_TurnLeft(){

    leg_1[1] = 600; leg_1[2] = 500;
    leg_4[1] = 500; leg_4[2] = 450;
    leg_5[1] = 600; leg_5[2] = 450;
    Leg_Angle_Set0();                          // 1,4,5번 다리 올리기

    leg_1[0] = 600;
    leg_4[0] = 900;
    leg_5[0] = 400;
    Leg_Angle_Set0();                          // 1,4,5번 다리 왼쪽 회전

    leg_1[1] = 800; leg_1[2] = 600;
    leg_4[1] = 700; leg_4[2] = 550;
    leg_5[1] = 800; leg_5[2] = 550;
    Leg_Angle_Set0();                          // 1,4,5번 다리 내리기

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

    leg_2[1] = 600; leg_2[2] = 500;
    leg_3[1] = 700; leg_3[2] = 550;
    leg_6[1] = 600; leg_6[2] = 450;
    Leg_Angle_Set1();                          // 2,3,6번 다리 올리기

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

    leg_1[0] = 850;
    leg_4[0] = 750;
    leg_5[0] = 650;
    Leg_Angle_Set0();                         // 1,4,5번 원래대로

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

    leg_2[1] = 800; leg_2[2] = 600;
    leg_3[1] = 900; leg_3[2] = 650;
    leg_6[1] = 800; leg_6[2] = 550;
    Leg_Angle_Set1();                          // 2,3,6번 다리 내리기

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);
}
//---------------- 왼쪽으로 회전 ----------------------------------------------------------------
void Leg_test4_TurnRight(){

    leg_2[1] = 600; leg_2[2] = 500;
    leg_3[1] = 700; leg_3[2] = 550;
    leg_6[1] = 600; leg_6[2] = 450;
    Leg_Angle_Set1();                                          // 2,3,6번 다리 올리기

    leg_2[0] = 600;
    leg_3[0] = 1000;
    leg_6[0] = 400;
    Leg_Angle_Set1();                                          // 2,3,6번 다리 오른쪽 회전

    leg_2[1] = 800; leg_2[2] = 600;
    leg_3[1] = 900; leg_3[2] = 650;
    leg_6[1] = 800; leg_6[2] = 550;
    Leg_Angle_Set1();                                          // 2,3,6번 다리 내리기

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

    leg_1[1] = 600; leg_1[2] = 500;
    leg_4[1] = 500; leg_4[2] = 450;
    leg_5[1] = 600; leg_5[2] = 450;
    Leg_Angle_Set0();                                          // 1,4,5번 다리 올리기

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

    leg_2[0] = 850;
    leg_3[0] = 750;
    leg_6[0] = 650;
    Leg_Angle_Set1();                                         // 2,3,6번 다리 원래대로

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

    leg_1[1] = 800; leg_1[2] = 600;
    leg_4[1] = 700; leg_4[2] = 550;
    leg_5[1] = 800; leg_5[2] = 550;
    Leg_Angle_Set0();                                          // 1,4,5번 다리 내리기

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);
}
//----------------- 오른쪽으로 회전 ----------------------------------------------------------------
```

그림 3-20) 왼쪽, 오른쪽 회전 동작 코드

```
void Leg_test2_TriLegMove(){

    leg_1[1] = 600; leg_1[2] = 500;
    leg_4[1] = 500; leg_4[2] = 450;
    leg_5[1] = 600; leg_5[2] = 450;
    Leg_Angle_Set0();                                          // 1,4,5번 다리 올리기

    leg_1[0] = 700;
    leg_4[0] = 650;
    leg_5[0] = 550;
    Leg_Angle_Set0();                                          // 1,4,5번 다리 앞으로

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

    leg_2[0] = 1000;
    leg_3[0] = 900;
    leg_6[0] = 850;
    Leg_Angle_Set1();                                           // 2,3,6번 다리 뒤로

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

    leg_1[1] = 850; leg_1[2] = 600;
    leg_4[1] = 700; leg_4[2] = 550;
    leg_5[1] = 800; leg_5[2] = 600;
    Leg_Angle_Set0();                                           // 1,4,5번 다리 내리기

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

    leg_2[1] = 600; leg_2[2] = 500;
    leg_3[1] = 700; leg_3[2] = 550;
    leg_6[1] = 600; leg_6[2] = 450;
    Leg_Angle_Set1();                                           // 2,3,6번 다리 올리기

    leg_2[0] = 850;
    leg_3[0] = 750;
    leg_6[0] = 650;
    Leg_Angle_Set1();                                          // 2,3,6번 다리 앞으로

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

    leg_1[0] = 850;
    leg_4[0] = 750;
    leg_5[0] = 650;
    Leg_Angle_Set0();                                           // 1,4,5번 다리 뒤로

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);

    leg_2[1] = 900; leg_2[2] = 630;
    leg_3[1] = 900; leg_3[2] = 650;
    leg_6[1] = 800; leg_6[2] = 550;
    Leg_Angle_Set1();                                             // 2,3,6번 다리 내리기

    part_check = 2; Tim_Oc_It_Stop(); HAL_Delay(120);
}
//--------------- 3다리씩 앞으로 전진 ----------------------------------------------------
```

그림 3-21) 3족 제어 전진 동작 코드

마지막으로 통신부 코드는 블루투스 모듈 HC-06과의 통신을 위한 bps=9600, data length=8bit, stop_bit=1로 설정을 해주기 위해 UART2(3)_Config() 함수 내 구조체를 재설정하였다. 아래 그림은 UART2(3)_Config() 함수 코드이다.

```
void UART2_Config(void)
{
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_USART2_CLK_ENABLE();

    GPIO_Init_Struct.Pin = GPIO_PIN_2;
    GPIO_Init_Struct.Mode   = GPIO_MODE_AF_PP;
    GPIO_Init_Struct.Pull   = GPIO_PULLUP;
    GPIO_Init_Struct.Speed  = GPIO_SPEED_FREQ_HIGH;
    HAL_GPIO_Init(GPIOA, &GPIO_Init_Struct);

    GPIO_Init_Struct.Pin = GPIO_PIN_3;
    GPIO_Init_Struct.Mode   = GPIO_MODE_INPUT;
    GPIO_Init_Struct.Pull   = GPIO_NOPULL;
    GPIO_Init_Struct.Speed  = GPIO_SPEED_FREQ_HIGH;
    HAL_GPIO_Init(GPIOA, &GPIO_Init_Struct);

    UartHandle.Instance  = USART2;
    UartHandle.Init.BaudRate   = 9600;
    UartHandle.Init.WordLength = UART_WORDLENGTH_8B;
    UartHandle.Init.StopBits   = UART_STOPBITS_1;
    UartHandle.Init.Parity  = UART_PARITY_NONE;
    UartHandle.Init.HwFlowCtl  = UART_HWCONTROL_NONE;
    UartHandle.Init.Mode = UART_MODE_TX_RX;

    HAL_UART_Init(&UartHandle);

    //HAL_NVIC_SetPriority(USART2_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(USART2_IRQn);
}
```

그림 3-22) Uart2(3) 초기화 코드

다음으로 무한 반복문 속에서 수신 받은 문자 1개를 버퍼에 저장한 뒤 HAL Driver 내 HAL_UART_TxCpltCallback() 와HAL_UART_RxCpltCallback() 함수를 이용하여, 수신된 문자를 확인하여 해당하는 문자와 연결된 동작을 활성화 하는 Check 변수를 증가시키게 되면 비로써 무한 반복문 속에 있는 해당 동작 함수가 수행되게 된다. 아래 그림은 데이터 수신 및 버퍼 확인 코드와 무한 루프 내 코드이다.

```
    if(RxBuffer[0] == 'f'){
    HAL_UART_Transmit_IT(&UartHandle,(uint8_t*)RxBuffer,1);
    Leg_test2_TriLegMove_set = 1;
    }
    else if(RxBuffer[0] == 'b'){
    HAL_UART_Transmit_IT(&UartHandle,(uint8_t*)RxBuffer,1);
    }
    else if(RxBuffer[0] == 'l'){
    HAL_UART_Transmit_IT(&UartHandle,(uint8_t*)RxBuffer,1);
    Leg_test3_TurnLeft_set = 1;
    }
    else if(RxBuffer[0] == 'r'){
    HAL_UART_Transmit_IT(&UartHandle,(uint8_t*)RxBuffer,1);
    Leg_test4_TurnRight_set = 1;
    }
    else if(RxBuffer[0] == 's'){
    HAL_UART_Transmit_IT(&UartHandle,(uint8_t*)RxBuffer,1);
    Leg_Reset2_set = 1;
    }
    else if(RxBuffer[0] == '1'){
    HAL_UART_Transmit_IT(&UartHandle,(uint8_t*)RxBuffer,1);
    Leg_Reset1_set = 1;
    }
    else if(RxBuffer[0] == '2'){
    HAL_UART_Transmit_IT(&UartHandle,(uint8_t*)RxBuffer,1);
    Leg_Reset2_set = 1;
    }
    else if(RxBuffer[0] == '3'){
    HAL_UART_Transmit_IT(&UartHandle,(uint8_t*)RxBuffer,1);
    Leg_test1_1LegMove_set = 1;
    }
    else if(RxBuffer[0] == '4'){
    HAL_UART_Transmit_IT(&UartHandle,(uint8_t*)RxBuffer,1);
    Leg_test5_LegCheck_set = 1;
    }
    else if(RxBuffer[0] == '5'){
    HAL_UART_Transmit_IT(&UartHandle,(uint8_t*)RxBuffer,1);
    Leg_test6_Hand_set = 1;
    }
}
```

```
while (1){
   HAL_UART_Receive_IT(&UartHandle_B,(uint8_t*)RxBuffer,1);
   HAL_Delay(10);

   if(Leg_Reset1_set == 1){
      Leg_Reset1();
      Leg_Reset1_set = 0;
   }
   else if(Leg_Reset2_set == 1){
      Leg_Reset2();
      Leg_Reset2_set = 0;
   }
   else if(Leg_test1_1LegMove_set == 1){
      Leg_test1_1LegMove();
      Leg_test1_1LegMove_set = 0;
   }
   else if(Leg_test2_TriLegMove_set == 1){
      Leg_test2_TriLegMove();
      Leg_test2_TriLegMove_set = 0;
   }
   else if(Leg_test3_TurnLeft_set == 1){
      Leg_test3_TurnLeft();
      Leg_test3_TurnLeft_set = 0;
   }
   else if(Leg_test4_TurnRight_set == 1){
       Leg_test4_TurnRight();
       Leg_test4_TurnRight_set = 0;
   }
   else if(Leg_test5_LegCheck_set == 1){
       Leg_test5_LegCheck();
       Leg_test5_LegCheck_set = 0;
   }
   else if(Leg_test6_Hand_set == 1){
       Leg_test6_Hand();
       Leg_test6_Hand_set = 0;
   }
}
```

그림 3-23) 버퍼 확인 코드(왼쪽) 및 무한 반복문 속 코드(오른쪽)

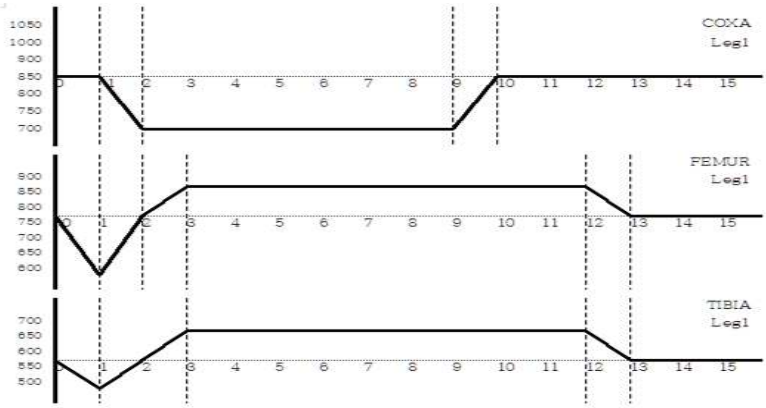
제 Ⅳ 장 결론 및 고찰

1. 결론 분석

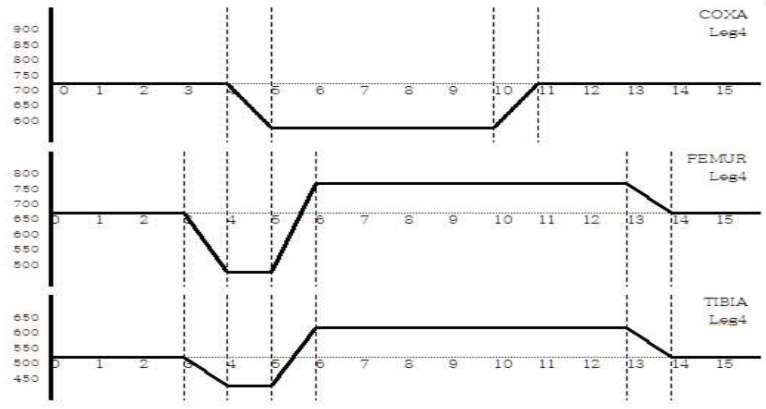
우선 6족 보행 로봇의 1족 제어 보행과 3족 제어 보행, 좌우회전 동작 구현에 있어 앞서 계획한 동작 알고리즘을 토대로 모터에 각 단계마다 걸릴 실제 각도(펄스 값)을 계산하였다. 아래 그림은 해당 각도 값을 그래프로 그린 것이며, 6족 보행로봇이 동작을 할 때 마다 다리의 각도가 어떤 흐름으로 변화하는지 살펴볼 수 있다.

• 1족 전진 동작 모터 각도 그래프

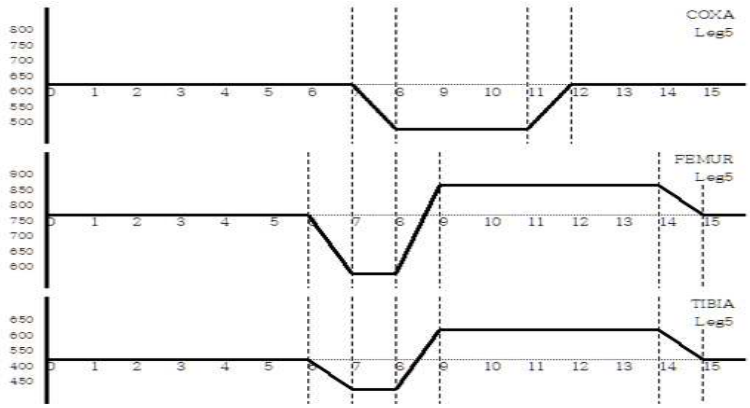
다리1



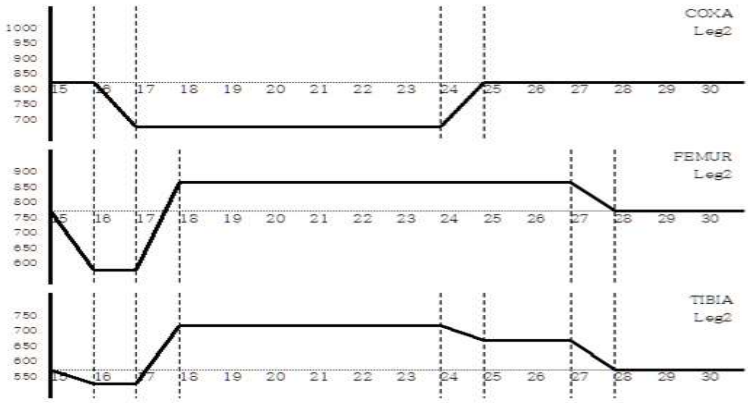
다리4



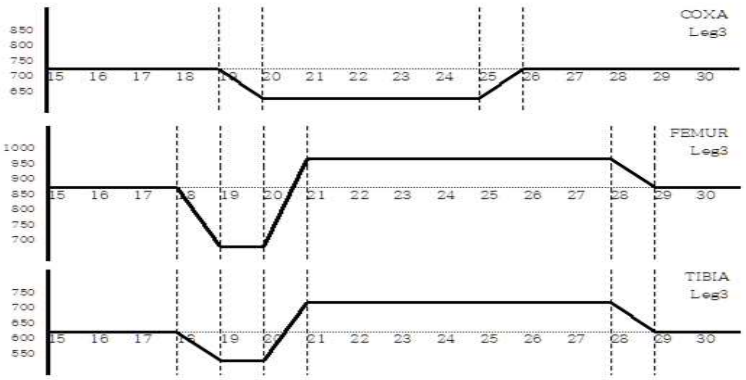
다리5


COXA
Leg5
FEMUR
Leg5
TIBIA
Leg5

다리2


COXA
Leg2
FEMUR
Leg2
TIBIA
Leg2

다리3


COXA
Leg3
FEMUR
Leg3
TIBIA
Leg3

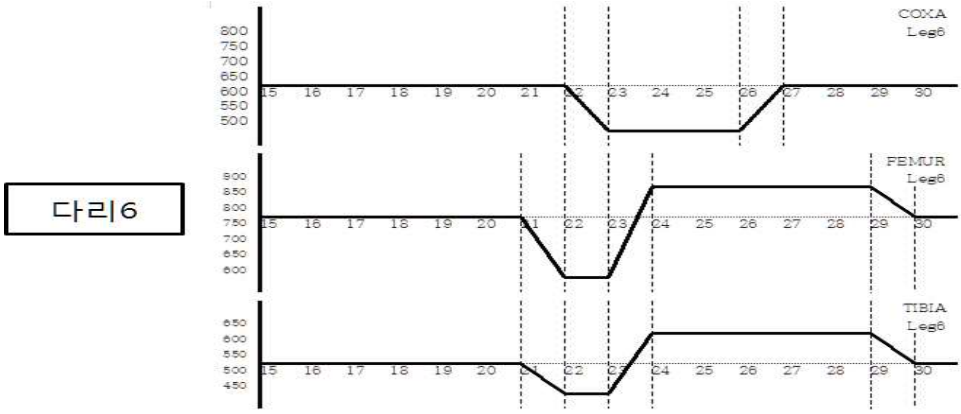


그림 4-1) 1족 제어 전진 모터 각도 그래프

그래프 순서는 다리가 동작하는 순서인 1,4,5,2,3,6번이며, 그래프의 가운데 수치가 로봇의 기본 대기상태인 각도수치이며, 그래프가 상승과 하강함에 따라 다리의 각도가 변한다. (1족 제어 보행은 이전 알고리즘 도에서 봤듯이 1개씩 다리가 제어되어 움직인다는 점을 참고)

그래프에 나타난 실제 각도 값은 각 모터가 전부 똑같은 각도로 움직이지 않는다는 걸 고려하여 약간씩 서로 다르게 주었으며, 최종적으로 모든 그래프가 완벽히 같은 모양은 아니지만, 전체적으로 유사하다는 것을 볼 수 있다.
또한 나머지 3족 제어 전진 및 좌우회전 그래프 또한 삼각형으로 묶인 다리끼리는 유사한 그래프 형태가 나타남을 볼 수 있었다.

• 3족 전진 동작 모터 각도 그래프

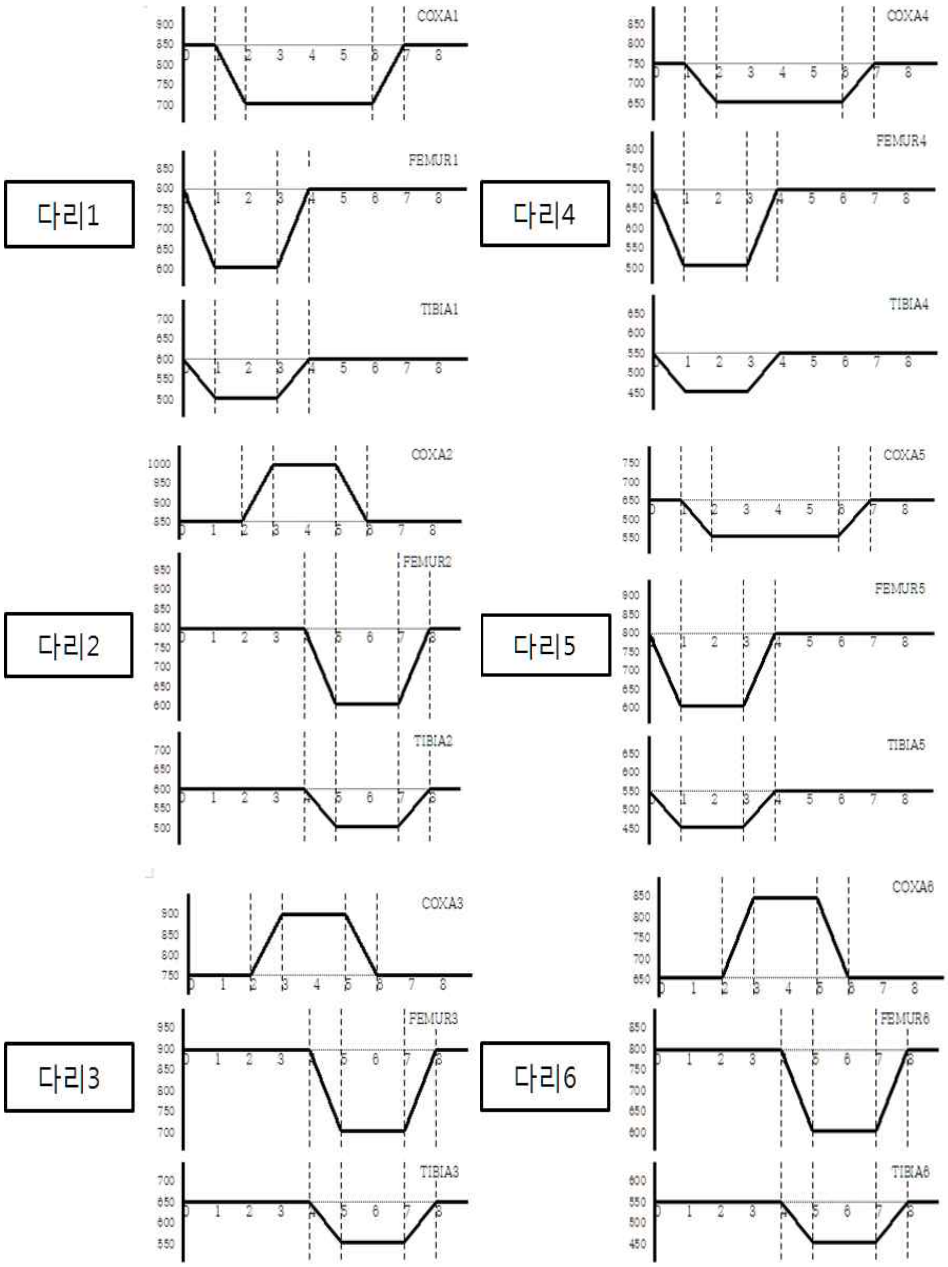


그림 4-2) 3족 제어 전진 모터 각도 그래프

• 좌회전 동작 모터 각도 그래프

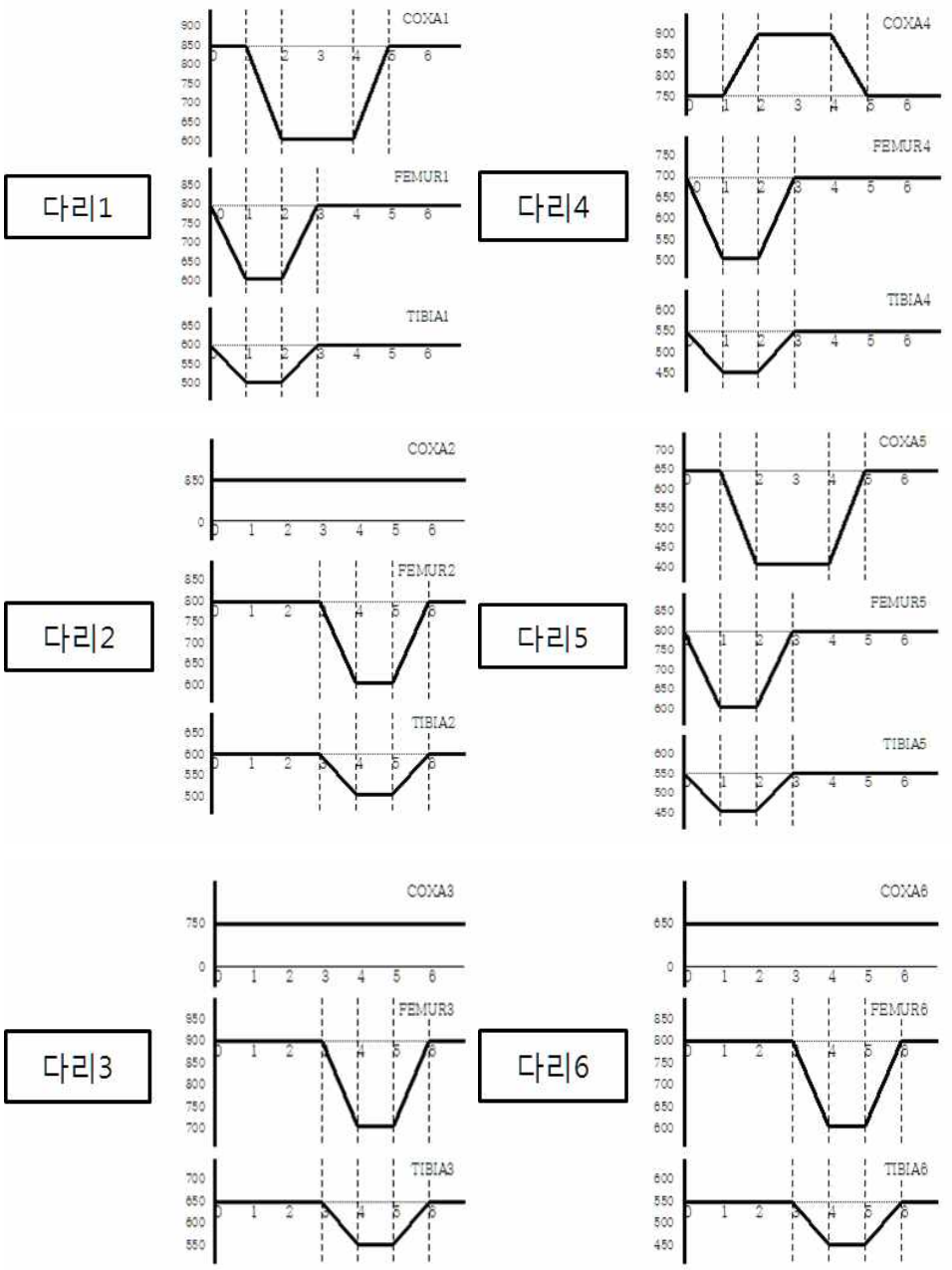


그림 4-3) 왼쪽 회전 모터 각도 그래프

• 우회전 동작 모터 각도 그래프

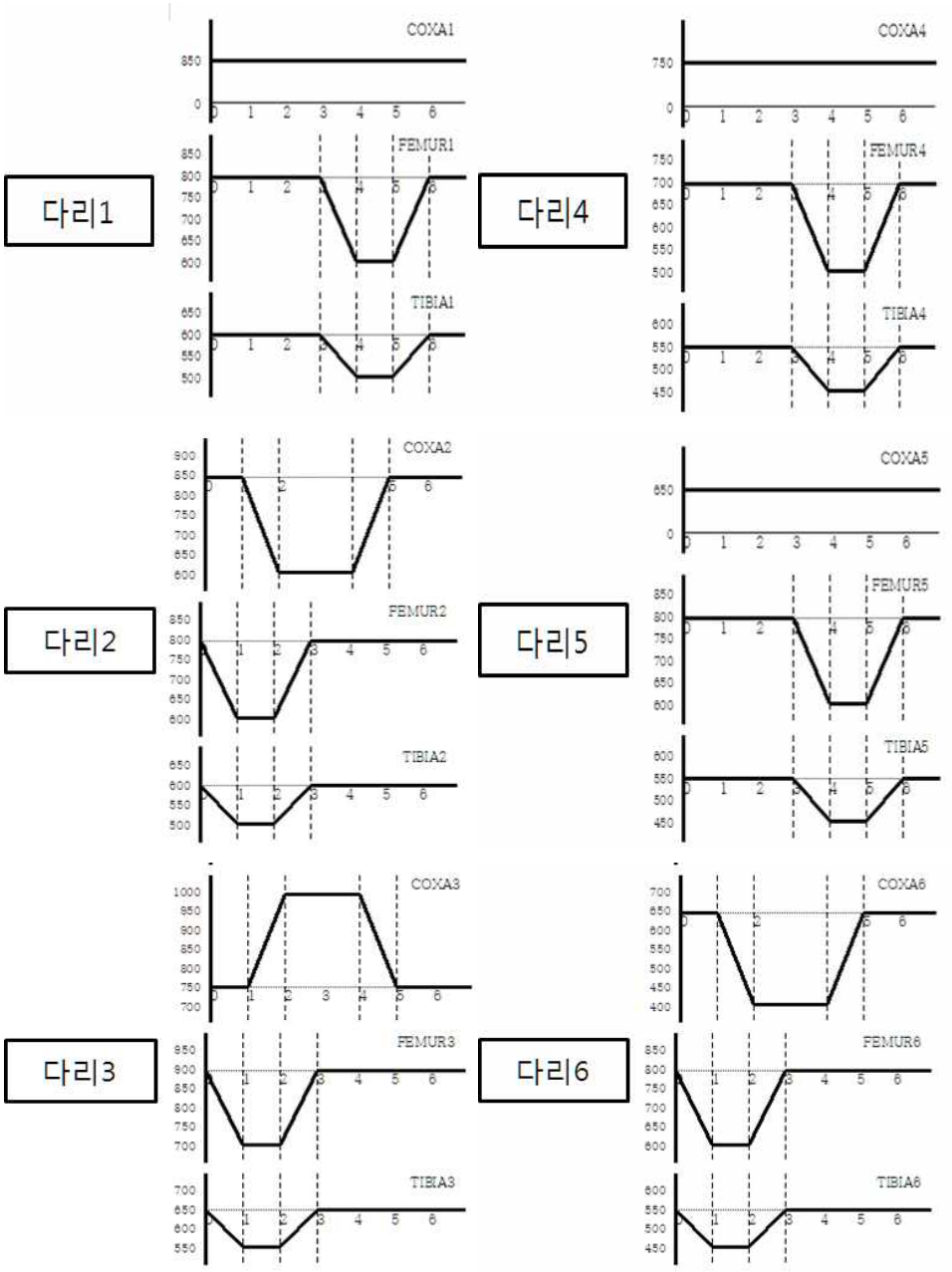
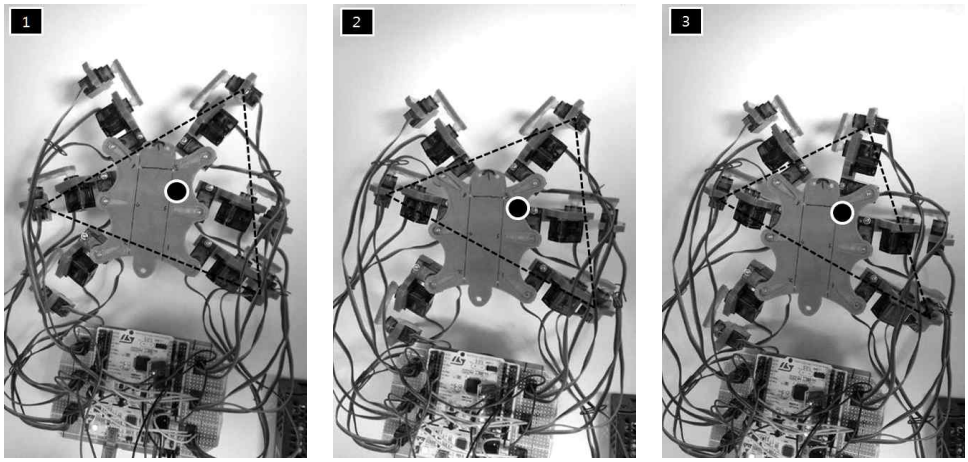
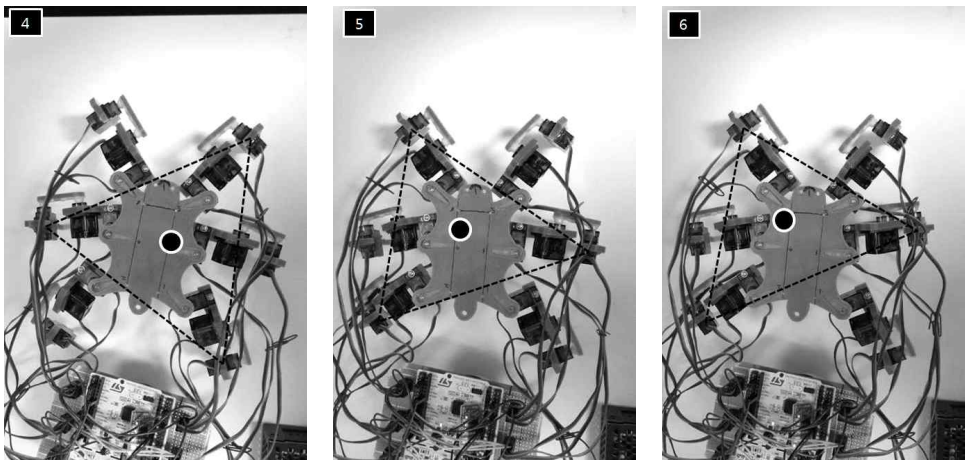


그림 4-4) 오른쪽 회전 모터 각도 그래프

다음으로 완성된 6족 보행 로봇의 움직임을 관찰하며, 자세 안정성을 보기 위해 다리의 COXA 부분이 바뀔 때마다 로봇의 무게중심이 어떻게 변하는지를 살펴본 사진이다.

• 1족 보행 동작 사진





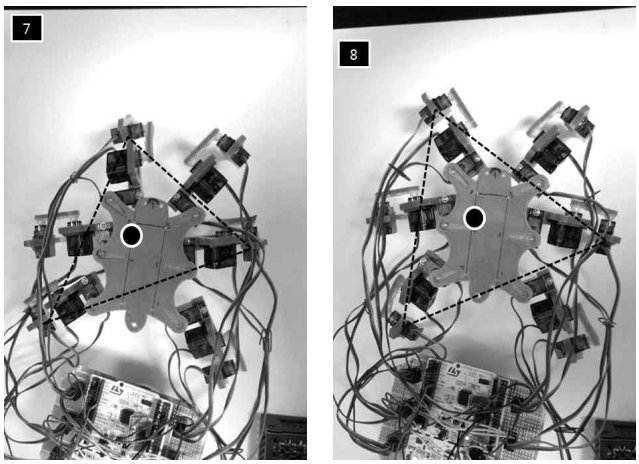
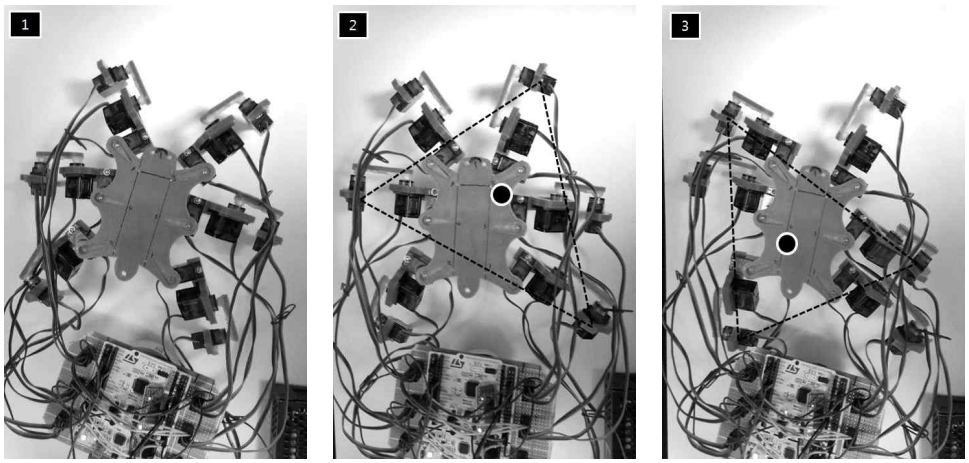


그림 4-5) 1족 제어 보행 동작 사진

그림에서 검은색 점으로 표시된 부분은 매 동작 중 다리의 COXA이 움직일 때마다 변화하는 다리 그룹의 무게중심을 표시한 것이다. 로봇이 아무런 동작을 하지 않은 대기 상태에서는 이 무게중심들이 로봇의 중심 쪽에 속해있다는 점에서 1족 제어 전진 같은 경우는 매 동작 단계마다 무게중심 변화폭이 크지 않다는 것을 볼 수 있다. 즉, 1족 제어 보행 같은 경우는 로봇이 크게 흔들림이 없이 비교적 안정된 자세로 움직인다는 걸 알 수 있다.

다음은 같은 전진동작인 3족 제어 보행의 경우이다.

• 3족 보행 동작 사진



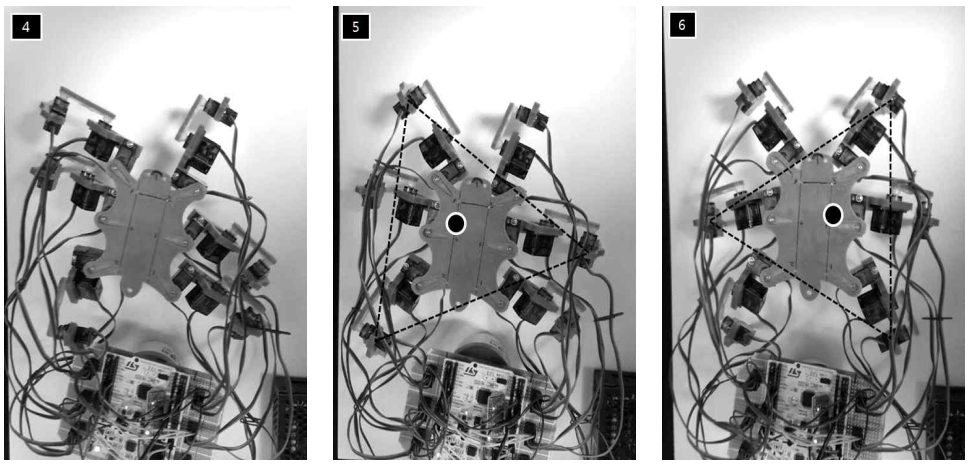
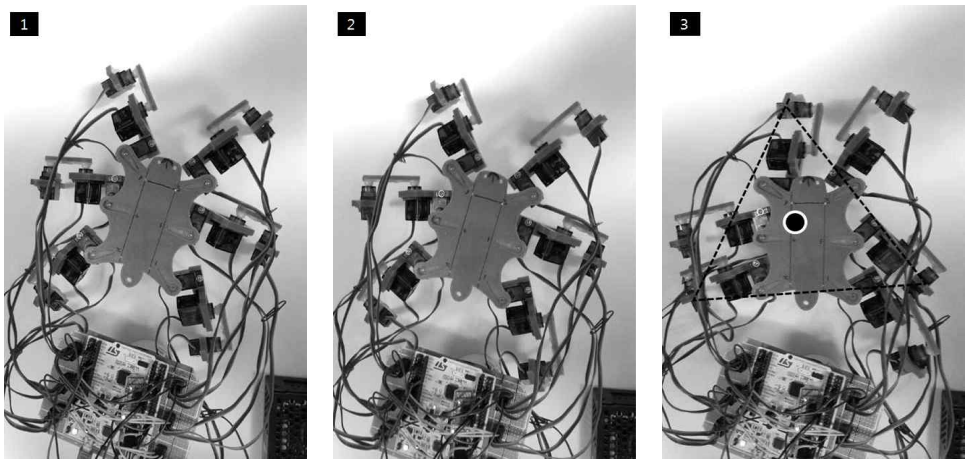


그림 4-6) 3족 보행 동작 사진

3족 보행 전진 동작 같은 경우, 이전 1족 제어보다는 적은 동작 단계를 가지고 있어 동작 속도는 빠르지만, 3개의 다리가 동시에 엇갈려 움직이므로 사진처럼 무게중심의 변화가 비교적 큰 폭으로 생긴다는 걸 알 수 있다. 이로 인해 실제 구동 시 동작마다 몸체에 조금씩 흔들림이 발생한다. 다음은 좌회전과 우회전 동작의 사진이다.

• 왼쪽 회전 동작 사진



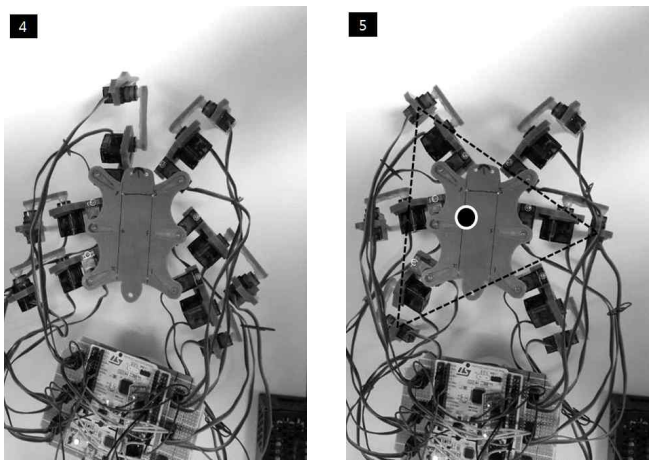
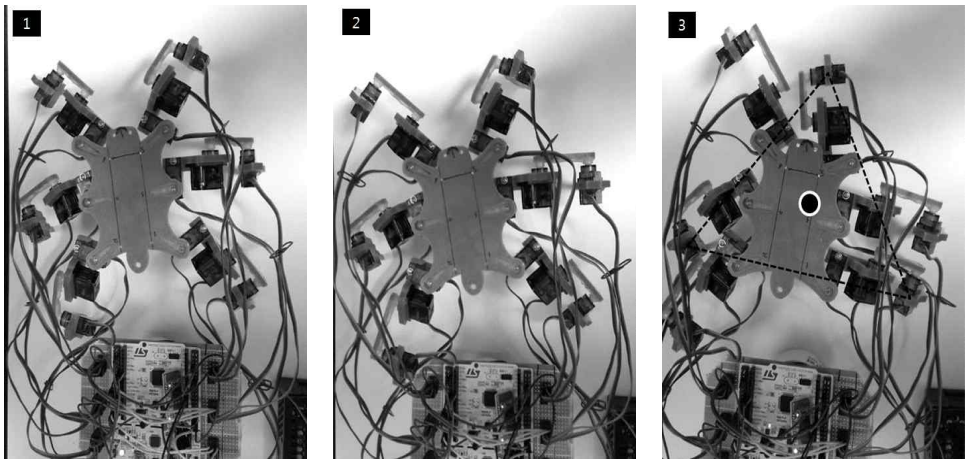


그림 4-7) 왼쪽 회전 동작 사진

• 오른쪽 회전 동작 사진



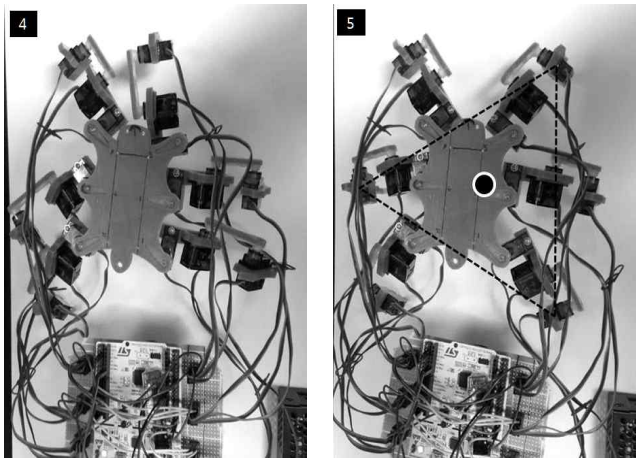


그림 4-8) 오른쪽 회전 동작 사진

회전 동작의 경우, 다리 3개가 삼각형모양을 그대로 유지한 채 움직이기 때문에 무게 중심의 변화가 거의 없다는 걸 볼 수 있다. 따라서 거의 몸체에 흔들림 없이 동작하게 된다.

따라서 동작 사진을 관찰한 결과를 정리하면, 1족 제어 전진은 3족 제어 전진에 비해 동작 단계가 많아 동작 시간이 길어지지만, 그만큼 몸체의 흔들림과 안전성이 높아진다는 특징을 가지고 있고, 3족 제어 전은 반대로 동작 단계가 적어 동작 시간이 빠르지만, 그만큼 몸체의 흔들림이 생긴다는 점과 모터의 하중이 커진다는 점이 특징을 가지고 있다. 하지만 6족 보행의 특징상 1족 제어 전진이나, 3족 제어 전진이나 모두 동작 시 몸체 무게중심이 크게 변화하지는 않아 높은 자세 안정성을 가지고 있다는 것을 볼 수 있다. 이는 연구방향에서 제시했던 6족 보행 로봇의 특징과 일치하는 것을 보여준다.

2. 고찰

현대 사회 속에서 점차 그 영역을 넓혀가는 보행로봇, 본 연구를 통해 보행 로봇을 ARM 프로세서를 사용하여 직접 제작해보고, 동작 알고리즘을 만들어보면서 6족 보행 로봇이 가지는 장점과 보행 메커니즘 방식을 이해하고 동작 제어에 있어서 발생할 수 있는 여러 외부 요인들에서 대해서 알 수 있었던 계기가 되었다. 그리고 같은 보행 로봇일지라도, 보행 방식에 따라 로봇의 이동 속도, 자세 안정성 등의 차이를 보인다는 점 또한 확인 할 수 있었다.

더불어 본 로봇 제작을 마치면서, 아래와 같은 취약점 발견할 수 있었다.

1) 9개의 타이머를 이용하여 18개 타이머를 제어하는 방식으로 인해 타이머가 동작하지 않은 모터의 토크가 줄어들어, 몸체 하중을 견디는데 무리가 생긴다.
2) 보급형 서보모터로 제작함에 있어, 섬세한 각도 조절 및 높은 기동 토크를 낼 수 없다.
3) 수직적인 모터 각도 조절로 인해 유연한 로봇 움직임을 구현되지 못했다.

이러한 취약점을 보완하기 위하여, Servo Motor Controller 및 고급 서보 모터 등을 사용하고, 로봇 각도 수치를 계산식화 하여 보다 유연한 로봇 움직임을 만들어 내는 것을 차후 과제로 남기며 본 6족 보행 로봇 연구를 마치기로 한다.

참고 문헌

- 차동혁 · 김재일. 『STM32F1 시리즈를 이용한 ARM Cortex-M3 구조와 응용』. 홍릉과학출판사, 2016

- 네이버 카페, “CortexWorld", 2008,08,04
  http://cafe.naver.com/CortexWorld

- NetworkOracle, "Basic Walking Motion for an Arduino Based Hexapod",
  http://networkoracle.com/hexapodwalk.html

- 성영휘 · 심한울(금오공과대학교 산학협력단) “6족 보행 로봇 및 그 보행 제어 방법”, Google Patent KR101279090B1, 2011.07.29

- **참고 사이트 및 데이터 시트**

Hexapod frame open source
(https://www.thingiverse.com)

STM32F103RB Data sheet.
(http://www.st.com/en/microcontrollers/stm32f103rb.html)

SG-90 Data sheet
(http://akizukidenshi.com/download/ds/towerpro/SG90.pdf)

HC-06 Bluetooth Module Datasheet
(https:www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf)

부 록

부록1) 전체 소스 코드..