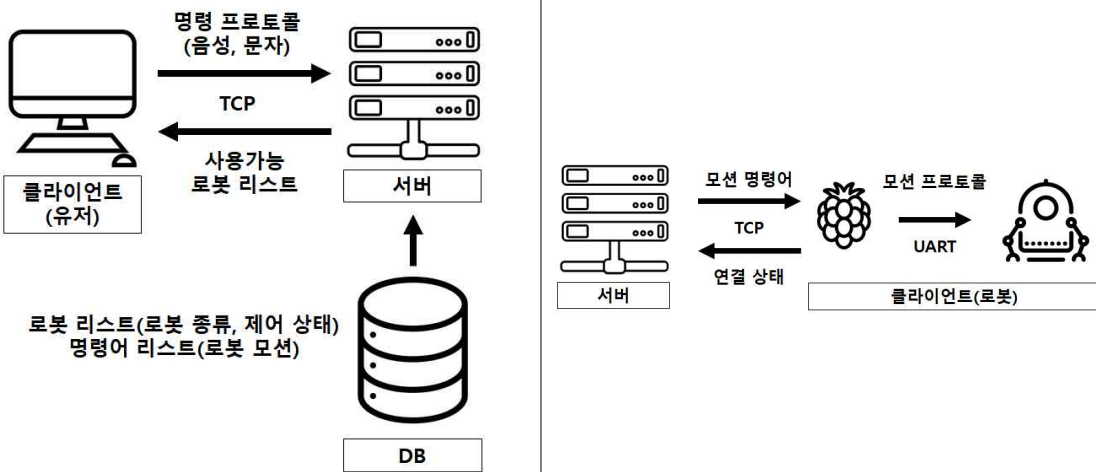
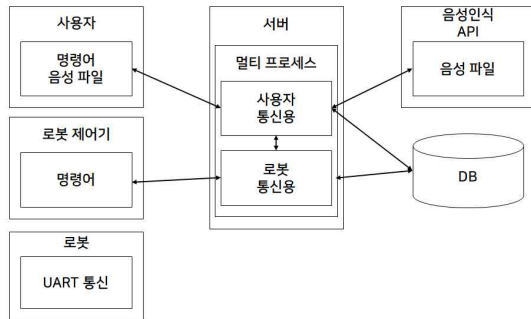


## <개발 계획서 요약>

(조) 팀 명		( 5조) 이거 끝나면 군대 갈 팀		
프로젝트 명		Control+R		
배경 및 당위성		최근 IT기술의 발전과 함께 로봇과 인공지능 분야에 많은 관심이 쏟아지고 있다. 로봇은 인간의 육체 노동력의 한계를 극복하고, 정신적인 업무를 인공지능의 도움을 받으며 더 효율적인 업무 수행에 있어서 큰 도움이 될 수 있다. 이러한 로봇을 좀 더 효율적으로 다루기 위한 'Control+R' 프로젝트를 제안한다.		
제안 내용	최종 목표	'Control+R'은 다양한 방법으로 로봇의 제어를 보조해주는 서비스다. 음성인식과, 채팅시스템 등 사용자가 원하는 방식으로 제어할 수 있도록 로봇과 컨트롤러를 중계해 주는 플랫폼이다.		
	시스템 개요			
		<div> <div>&lt;그림 1&gt; 유저 클라이언트 - 서버 간 구조</div> <div>&lt;그림 2&gt; 로봇 클라이언트 - 서버 간 구조</div> </div>		
		<ul style="list-style-type: none"> <li>- 클라이언트(유저)에서 서버에게 원하는 명령어 전송</li> <li>- 서버에서는 수신한 데이터를 가공(음성파일 머신러닝 등)</li> <li>- 가공된 정보를 이용해 DB에서 해당 로봇 모션 호출</li> <li>- 라즈베리파이와 서버가 통신하여 로봇 모션 정보 통신</li> <li>- 라즈베리파이의 GPIO를 이용해 로봇을 UART통신으로 제어</li> </ul>		
개발 방법	클라이언트	서버		
	<b>유저</b> <ul style="list-style-type: none"> <li>- 유저의 입력을 받을 수 있는 구조</li> <li>- 음성, 텍스트, 키 입력 등 여러 메소드를 유저가 선택 가능</li> </ul> <b>로봇</b> <ul style="list-style-type: none"> <li>- 라즈베리파이와 서버 간 TCP 통신</li> <li>- 로봇을 시리얼 핀을 이용해 제어</li> </ul>	<ul style="list-style-type: none"> <li>- 클라이언트에서 받은 데이터를 처리</li> <li>- 처리된 데이터를 기반으로 DB에서 적절한 모션을 꺼내옴</li> <li>- 해당하는 모션정보를 TCP 소켓을 이용해 라즈베리파이로 전송</li> </ul>		
기대효과 (학습적 교육 효과 및 실용성 포함)		<b>학습적 측면</b> <ul style="list-style-type: none"> <li>- 멀티 프로세스를 이용한 TCP통신을 이해하고 구현함</li> <li>- REST API를 이용한 http통신으로 데이터를 주고받는 과정을 이해할 수 있음</li> </ul> <b>실용적 측면</b> <ul style="list-style-type: none"> <li>- 여러 로봇들을 통일된 프로토콜로 제어하여 효율적으로 다룰 수 있음</li> <li>- 하나의 로봇을 다양한 방법으로 제어하여 확장성을 높임</li> </ul>		
중심어	로봇		TCP소켓	음성인식
	머신러닝		라즈베리파이	UART

## <개발 계획서 세부>

(조) 팀 명	(5조) 이거끝나면군대갈팀																					
제목	Control+R(로봇 통신 중개 플랫폼)																					
시스템 구성도		<div style="text-align: center;">32bit</div> <table border="1" style="width: 100%; border-collapse: collapse;"><tr><td style="width: 50%;">데이터 용도</td><td style="width: 50%;">데이터 정보 개수</td></tr><tr><td>데이터 타입</td><td>데이터 개수</td></tr><tr><td style="text-align: center;">...</td><td style="text-align: center;">...</td></tr><tr><td>데이터 길이</td><td>데이터</td></tr><tr><td style="text-align: center;">...</td><td style="text-align: center;">...</td></tr></table> <div style="text-align: center;">1번 로봇 연결 요청</div> <table border="1" style="width: 100%; border-collapse: collapse;"><tr><td style="width: 50%;">User-&gt;server, 연결 요청(int)</td><td style="width: 50%;">2</td></tr><tr><td>string</td><td>1</td></tr><tr><td>int</td><td>1</td></tr><tr><td>5</td><td>char **</td></tr><tr><td>4</td><td>int *</td></tr></table> <div style="text-align: right;">LINK 1</div>	데이터 용도	데이터 정보 개수	데이터 타입	데이터 개수	...	...	데이터 길이	데이터	...	...	User->server, 연결 요청(int)	2	string	1	int	1	5	char **	4	int *
	데이터 용도	데이터 정보 개수																				
데이터 타입	데이터 개수																					
...	...																					
데이터 길이	데이터																					
...	...																					
User->server, 연결 요청(int)	2																					
string	1																					
int	1																					
5	char **																					
4	int *																					
	<그림 3> 서버-클라이언트 세부 구조	<그림 4> 데이터 통신 프로토콜																				
주요 SW 모듈	<p><b>조종기 클라이언트</b></p> <ol style="list-style-type: none"><li>1. 연결할 로봇을 정하기 위해 서버로부터 로봇 리스트를 가져옴</li><li>2. 사용권한 요청을 위한 명령어와 사용할 로봇의 ID를 전송</li><li>3. 서버에게 다양한 형식의 로봇을 위한 명령어를 전송</li><li>4. 연결을 끝내기 위한 명령어를 서버에 전송하고 소켓을 종료</li></ol> <p><b>서버</b></p> <ol style="list-style-type: none"><li>1. 클라이언트로부터 리스트에 대한 요청이 들어올 경우 DB에서 로봇 리스트를 불러와 클라이언트에게 전송</li><li>2. 클라이언트로부터 로봇 연결 요청을 받은 경우 클라이언트로부터 받은 id를 사용해 DB에서 로봇이 현재 사용하고 있는지 확인 후 연결 성공 여부를 클라이언트에 전송</li><li>3. 클라이언트로부터 명령어를 받아 모션 ID로 변환 후 DB에서 로봇에 맞는 모션 프로토콜을 가져옴</li><li>4. 로봇을 제어하는 파이에게 모션 프로토콜 전송</li></ol> <p><b>라즈베리 파이</b></p> <ol style="list-style-type: none"><li>1. 서버로부터 모션 명령어를 받음(TCP)</li><li>2. 모션 명령어로 로봇을 직접 제어 or 로봇 제어기에 명령어 전송(UART)</li></ol> <p><b>DB</b></p> <ol style="list-style-type: none"><li>1. 로봇이 생성될 때 마다 리스트에 저장 로봇 ID/로봇 종류/클라이언트 ID</li><li>2. 로봇 종류마다 모션 프로토콜 리스트 저장 로봇 종류/모션 ID/모션 프로토콜</li></ol>																					
	위험요소	<ul style="list-style-type: none"><li>- 음성인식 API서비스가 불안정 할 수 있음</li><li>- 이미 사용 중인 로봇에 대해 교착상태가 일어날 수 있다</li><li>- 로봇이 배터리로 동작하므로 클라이언트 서버가 강제 종료되는 일이 잦을 수 있다</li><li>- 한 로봇이 여러 명령을 동시에 처리할 수 없다</li></ul>																				
위험요소의 해결 방법	<ul style="list-style-type: none"><li>- 대체로 사용할 수 있는 음성인식 API를 찾아보기</li><li>- Blocking을 이용해 사용 중인 로봇에 대한 접근을 막는다.</li><li>- 시그널과 timeout을 통해 클라이언트가 종료를 감지하여 처리한다.</li><li>- 명령어 큐를 만들어 순차적으로 처리할 수 있도록 설계한다.</li></ul>																					