

졸업과제 최종보고서

음성 인식 오픈소스를 이용한 블루투스 로봇 펫 '쿠키'

김정구 교수님

분과 C 개냥이조

201612148 전양희

201469118 이규원

201624428 김범미

부산대학교 정보컴퓨터공학부

목차

1.서론.....	3
1.1개요.....	3
1)목표.....	3
2)선정배경.....	3
1.2요구사항.....	3
1)요구조건분석.....	3
2)산업체멘토링결과.....	4
3)제약사항.....	5
1.3개발환경.....	5
1)Arduino.....	5
2)Android Studio.....	5
2.본론.....	5
2.1보드구성.....	6
2.2과제수행과정.....	6
1)Arduino.....	9
2)Android Smart application.....	20
2.3과제수행결과.....	20
1)전체 시스템 구성도.....	20
2)Board 구성도.....	22
3)Application.....	22
4)시스템 동작 및 .결과.....	24
3.결론.....	26
3.1결론.....	26
3.2보완사항.....	27

3.3추진일정.....	27
3.4역할분담.....	28
3.5참고문헌.....	28

1.서론

1.1 개요

1)목표

이번 졸업과제물인 아두이노 로봇 '쿠키'는 기본적으로 아두이노 MEGA보드를 베이스로 하고 있으며, 블루투스4와 음성 인식 오픈소스를 기반으로, 이름을 부르면 반응을 하고, 간단한 명령을 수행할 수 있으며, 시간이 지나면 배고픔을 표시하는 등, 실제 강아지의 행동과 유사하게 만들었다.

2)선정 배경

현대 사회의 바쁜 현대인들에게는 애완동물을 기를 만한 시간과 형편이 부족하다. 그러나 동물을 기르고 싶은 사람들에게 조금이나마 만족감을 주기 위한 대안으로 수 많은 로봇펫이 출시되었다. 그러나 기존의 로봇펫들은 정해진 기능을 잘 수행하기는 하지만 정말 애완동물을 기르는 것과는 차이가 크다는 아쉬움이 있었다. 우리가 실제로 애완동물을 기르게 되면 애완동물을 성장시키고 교감하게 되며 어느 정도의 노력을 기울여야 하는데 현재의 로봇펫은 이런 부분이 제대로 구현되어 있지 않다. 우리는 이런 점을 개선하여 조금 더 실제와 비슷한 로봇펫을 구현하고 싶어 이 프로젝트를 시작하게 되었다.

1.2 요구사항

1) 요구조건 분석

- 어플리케이션 설계

스마트폰을 통해 로봇 펫과 교감할 수 있도록 블루투스 연결 및 음성 인식 정보를 전송하는 간단한 어플리케이션을 제작한다.

- 음성 인식 기능

네이버의 음성인식 api를 사용하여 음성을 인식하고 그것을 텍스트로 변환하여 정해진 명령어를 인식하면 펫이 해당 동작을 수행하도록 개발한다.

- 레벨 기능

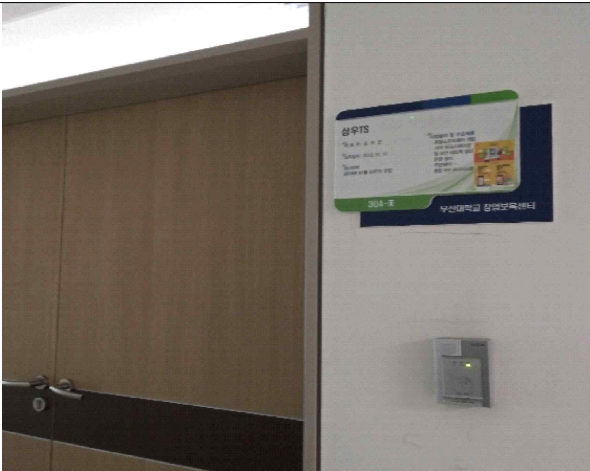

실제 펫은 처음부터 모든 명령어를 수행할 수 있지 않고 점차 명령어를 배워나가므로 레벨 시스템을 도입하여 각 레벨에 맞게 수행할 수 있는 명령어를 지정하며 레벨이 오를수록 수행 가능 명령이 늘어난다..

- 명령 미입력 상태

실제 펫은 아무런 명령이 들어오지 않아도 가만히 있지 않고 움직이므로 실제 펫과 가깝게 하기 위해 명령이 들어오지 않을 때 동작을 수행하도록 한다.

2) 산업체 멘토링 결과

팀 명	개냥이조	
팀 원	이규원, 전양희, 김범미	
과 제 명	음성 인식 오픈소스를 이용한 블루투스 로봇 펫 ‘쿠키’	
산 업 체 멘 토	기 업 명	삼우TS
	성 명	김부곤

세부 활동 내용	
<p>1. 산학협력 멘토링 내용</p> <p>디자인과 관련하여 3D프린팅이나 봉제인형을 개조해서 씌우는 방법 등 더 펫같이 보일 수 있는 방법과 휴대폰으로 음성인식을 하는 방법보다 휴대폰을 하나 더 활용하여 펫에 보이지 않게 숨겨 더 자연스럽게 보일 수 있는 방법을 제시해주셨다. 시중의 로봇청소기처럼 모서리 등에 충돌하기 전 멈추고 뒤로 가도록 개선해보라고 하셨고 펫이 동작 중일 때 명령을 내리면 그 명령이 누락되는 경우가 발생하는 부분에 대해 멘토링을 요청했을 때 동작 중일 때 명령은 저장해뒀다가 동작이 끝나고 그 명령을 실행하는 방향으로 시도해보라고 멘토링 해주셨다.</p> <p>추후에 기존의 오픈소스 기술들을 활용하여 바퀴가 아닌 4족 보행을 하거나 펫이 사람을 따라다니고 로봇청소기와 결합하여 청소를 수행할 수 있는 펫 등의 발전방향을 제시해주셨다.</p> <p>2. 산학협력 멘토링 후기</p> <p>그동안 생각 하지 못했던 새로운 관점으로 결과물을 평가해주셔서 사용자 기준에서 조금 더 만족도를 높일 수 있는 여러 가지 개선방안을 알아갈 수 있는 유익한 시간이었다. 또한 외관 퀄리티 향상을 위한 실질적이며 구체적인 조언도 해주셨으며 졸업과제 이후의 개선 및 활용 방안에 대해서도 조언해주셔서 많은 도움이 되었다. 결과물 피드백부터 혁신적인 제품을 만드는 방법론까지 약 1시간 가량 열정적으로 피드백 해주신 점이 인상 깊었다.</p>	
	
회사 전경	
	멘토링 회의 모습

3) 제약사항

- 로봇 펫 하드웨어 제작의 어려움

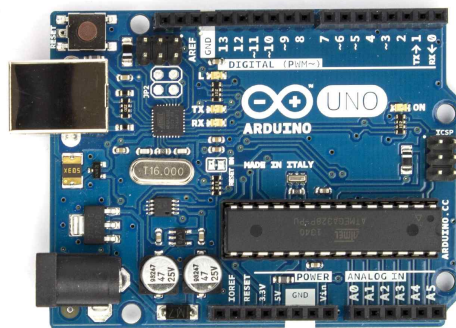
-> 하드웨어에 대한 지식이 상대적으로 부족하기 때문에 로봇 펫의 하드웨어를 독자적으로 제작하는 것은 어렵다. 따라서 다른 센서나 모듈과 함께 사용할 수 있고 개발하기 쉬운 아두이노를 기반으로 제작하였다.

- 명령어의 개수 제한

-> 명령어의 개수를 무한정으로 넣을 수 없기 때문에 자주 쓸 만한 명령어를 10개 정도만 만들어 음성인식과 어플리케이션을 통해서 명령을 내릴 수 있도록 하였다.

1.3 개발환경

1)Arduino



아두이노는 오픈 소스를 기반으로 한 단일 보드 마이크로컨트롤러로 완성된 보드와 관련 개발 도구 및 환경을 말한다. 2005년 이탈리아의 IDII(Interaction Design Institutelvera)에서 하드웨어에 익숙지 않은 학생들이 자신들의 디자인 작품을 손쉽게 제어할 수 있게 하려고 고안된 아두이노는 처음에 AVR을 기반으로 만들어졌으며, 아트멜 AVR 계열의 보드가 현재 가장 많이 판매되고 있다.

아두이노는 다수의 스위치나 센서로부터 값을 받아들여, LED나 모터와 같은 외부 전자 장치들을 통제함으로써 환경과 상호작용이 가능한 물건을 만들어 낼 수 있다. 임베디드 시스템 중의 하나로 쉽게 개발할 수 있는 환경을 이용하여, 장치를 제어할 수 있다.

아두이노의 가장 큰 장점은 마이크로컨트롤러를 쉽게 동작시킬 수 있다는 것이다. 아두이노는 컴파일된 펌웨어를 USB를 통해 쉽게 업로드 할 수 있으며, 다른 모듈에 비해 비교적 저렴하고, 윈도를 비롯해 맥 OS X, 리눅스와 같은 여러 OS를 모두 지원한다. 그리고 아두이노 보드의 회로도가 CCL에 따라 공개되어 있으므로, 누구나 직접 보드를 만들고 수정할 수 있다.

2)Android Studio



안드로이드 스튜디오(영어: Android Studio)는 안드로이드 및 안드로이드 전용 어플(앱)제작을 위한 공식 통합 개발 환경 (IDE)이다. 2013년 5월 16일, 구글 I/O 컨퍼런스에서 구글의 제품 관리자인 Ellie Powers에 의해서 발표되었다.

2.본론

2.1 보드 구성

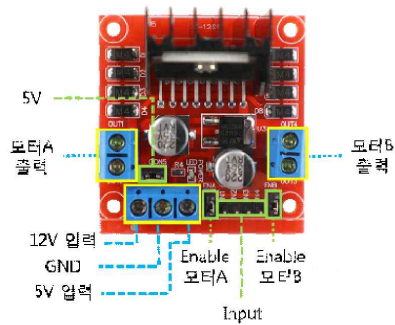
dc모터



DC(Direct Current) 모터는 모터의 가장 일반적인 타입이다. DC모터는 보통 음극 양극이 존재한다. 이 두 개의 극을 배터리에 직접 연결하면 모터가 작동된다. 만약 극을 바꾸면 모터가 반대 방향으로 작동하게 된다.

모터 드라이버 L298N

모터의 속도를 제어 및 회전 방향을 제어할 수 있는 회로 장치로서 아두이노 보드 또는 MCU를 통해 사용할 수 있는 드라이버이다. DC모터 또는 스텝핑 모트럴 연결하여 제어 가능하며, 입력 전압은 12V 또는 5V로 사용할 수 있다.



초음파센서 HC-SR04



초음파의 원리를 이용하여 20~5000mm 거리를 15도 이내에 측정할 수 있다. 5V 전원을 인가한 후 Trig핀을 통해 10us의 펄스를 인가하면 8개의 40kHz 펄스를 발생시키고, 측정된 거리에 따라 150us ~ 25ms의 펄스를 Echo핀을 통해 출력시킨다.

서보모터 SG90



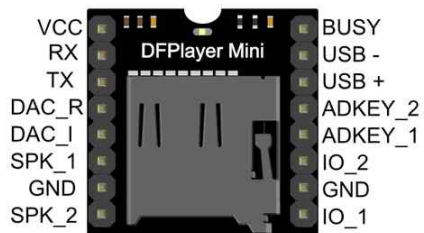
DC모터에 귀환 회로를 추가하여 정확한 위치 제어가 가능하게 구성된 모터로 정밀 제어가 가능하고 스텝 모터보다 빠른 장점이 있다. 귀, 꼬리, 앞발 등이 움직이는 모습을 실제 펫처럼 정밀하게 제어하기 위해 서보모터를 사용하게 되었다.

블루투스 HC-06 모듈



HC-06은 아두이노에서 블루투스 2.0 통신을 바로 사용할 수 있도록 제작된 모듈이다. 다른 블루투스 모듈에 비해 사용하기 쉽다는 장점 때문에 선택하게 되었으며 HC-06과 스마트폰의 통신은 슬레이브 모듈이 필요하다. 총 4개의 핀이 있으며 신호를 받기위한 핀인 RX핀, 신호를 전송하기 위한 TX핀, GND(Ground), VCC핀이 있다.

아두이노 DF Player Mini MP3



소형 MP3 모듈로 스피커에 직접 연결해서 음원을 재생할 수 있다. 마이크로 SD카드에 음원을 저장할 수 있으며, 아두이노와 연결해서 편리하게 제어할 수 있다. 사용방법도 간단하고 단순히 펫의 소리를 내는 도구로 선택하게 되었다.

아두이노 LCD 16x2 4핀(I2C제어)



LCD(Liquid Crystal Display)란 후면에 백라이트를 두고 전면에 액정을 두어 액정이 전기 신호에 따라 빛을 차단하거나 통과시키는 방식으로 빛을 내는 액정 표시장치이다. 1602 (16x2)는 LCD중 가장 보편적이고 쉽게 활용할 수 있어 펫의 상태를 나타내는 도구로 선택하게 되었고 LCD 뒷면에 I2C변환 모듈을 부착하여 16개의 제어 핀을 4개의 핀(GND, VCC, SDA, SCL)으로 제어가 가능하도록 하였다.

2.2 과제 수행 과정

1)Arduino

가. 블루투스 연결

```
void setup() {  
    Serial.begin(9600); //시리얼 모니터  
    bluetooth.begin(9600); //블루투스 시리얼 개방  
  
    if(bluetooth.available()) {  
        value = bluetooth.read();  
        Serial.write(bluetooth.read());  
    }  
  
    #define BT_TXD 3 //블루투스 TX  
    #define BT_RXD 2 //블루투스 RX  
  
    SoftwareSerial bluetooth(BT_RXD, BT_TXD); //시리얼 통신 선언  
    char value; //블루투스에서 받는 값 이름
```

우선 #define으로 블루투스 TX, RX pin을 설정해 주었다. 이 후 Serial1으로(SoftwareSerial에서 추후 변경됨) 시리얼 통신을 선언한 뒤, 블루투스에서 받는 값 이름도 선언하였다.

setup()에서 시리얼 모니터와 블루투스 시리얼을 개방하고, 본문의 loop()에서 if문을 이용하여 bluetooth에서 받은 값이 있을 때, 받은 값 value를 읽고 전송하도록 코드를 구성하였다.

나. 로봇 펫 이동

```
void controlPet(char value);  
void SmartCar_Go();  
void SmartCar_Back();  
void SmartCar_Stop();  
void SmartCar_Left();  
void SmartCar_Right();  
void SmartCar_sLeft();  
void SmartCar_sRight();
```

```

//motor PIN 세팅
int RightMotor_E_pin = 4;      // 오른쪽 모터의 Enable & PWM
int RightMotor_1_pin = 6;      // 오른쪽 모터 제어선 IN1
int RightMotor_2_pin = 7;      // 오른쪽 모터 제어선 IN2
int LeftMotor_3_pin = 8;       // 왼쪽 모터 제어선 IN3
int LeftMotor_4_pin = 9;       // 왼쪽 모터 제어선 IN4
int LeftMotor_E_pin = 5;       // 왼쪽 모터의 Enable & PWM

pinMode(13, OUTPUT);
pinMode(RightMotor_E_pin, OUTPUT);      // 출력모드로 설정
pinMode(RightMotor_1_pin, OUTPUT);
pinMode(RightMotor_2_pin, OUTPUT);
pinMode(LeftMotor_3_pin, OUTPUT);
pinMode(LeftMotor_4_pin, OUTPUT);
pinMode(LeftMotor_E_pin, OUTPUT);
pinMode(6, OUTPUT);

```

이동 함수를 선언하고, 모터 제어선의 pin 설정을 하고, 출력모드로 설정한다.

```

void SmartCar_Stop() {          // 정지
    for(int i=E_carSpeed; i>=0; i=i-5){
        if(prev_command == 'f'){
            analogWrite(RightMotor_E_pin, i);
            analogWrite(LeftMotor_E_pin, i);
            delay(20);
        }
        else if(prev_command == 'b'){
            analogWrite(RightMotor_E_pin, i);
            analogWrite(LeftMotor_E_pin, i);
            delay(20);
        }
    }

    digitalWrite(RightMotor_E_pin, LOW); // 오른쪽 모터 정지
    digitalWrite(LeftMotor_E_pin, LOW);  // 왼쪽 모터 정지
    lcd.print("STOP!");
    delay(200);
    lcd.init();
}

```

```

void SmartCar_Back() {          // 후진
    Serial.println("Backward");
    digitalWrite(RightMotor_1_pin, LOW);
    digitalWrite(RightMotor_2_pin, HIGH);
    digitalWrite(LeftMotor_3_pin, LOW);
    digitalWrite(LeftMotor_4_pin, HIGH);

    for(int i=0; i<=E_carSpeed; i=i+5){
        analogWrite(RightMotor_E_pin, i);
        analogWrite(LeftMotor_E_pin, i);
        delay(20);
    }
    lcd.print("Back");
    delay(200);
    lcd.init();
    prev_command = 'b';
}

```

```

void SmartCar_Left() {          // 좌회전
    Serial.println("Left");
    digitalWrite(RightMotor_1_pin, LOW);
    digitalWrite(RightMotor_2_pin, HIGH);
    digitalWrite(LeftMotor_3_pin, HIGH);
    digitalWrite(LeftMotor_4_pin, LOW);

    for(int i=0; i<=E_carSpeed; i=i+5){
        analogWrite(RightMotor_E_pin, i);
        analogWrite(LeftMotor_E_pin, i);
        delay(20);
    }
    lcd.print("Left!");
    delay(200);
    lcd.init();
    prev_command = 'l';
}

```

```

void SmartCar_Go() {          // 전진
    Serial.println("Forward");

    digitalWrite(RightMotor_1_pin, HIGH);
    digitalWrite(RightMotor_2_pin, LOW);
    digitalWrite(LeftMotor_3_pin, HIGH);
    digitalWrite(LeftMotor_4_pin, LOW);

    for(int i=0; i<=E_carSpeed; i=i+5){
        analogWrite(RightMotor_E_pin, i);    // 오른쪽 모터 전진 PWM제어
        analogWrite(LeftMotor_E_pin, i);
        delay(20);
    }
    lcd.print("Go Go");
    delay(200);
    lcd.init();
    prev_command = 'f';
}

void SmartCar_Turn() { // 돌아
    Serial.println("turn");
    digitalWrite(RightMotor_1_pin, LOW);
    digitalWrite(RightMotor_2_pin, HIGH);
    digitalWrite(LeftMotor_3_pin, HIGH);
    digitalWrite(LeftMotor_4_pin, HIGH);

    for(int i=0; i<=E_carSpeed; i=i+5){
        analogWrite(RightMotor_E_pin, i);
        analogWrite(LeftMotor_E_pin, i);
        delay(20);
    }
    lcd.print("Turning..");
    delay(200);
    lcd.init();
}

```

```

void SmartCar_Right() {          // 우회전
    Serial.println("Right");
    digitalWrite(RightMotor_1_pin, HIGH);
    digitalWrite(RightMotor_2_pin, LOW);
    digitalWrite(LeftMotor_3_pin, LOW);
    digitalWrite(LeftMotor_4_pin, HIGH);

    for(int i=0; i<=E_carSpeed; i=i+5){
        analogWrite(RightMotor_E_pin, i);
        analogWrite(LeftMotor_E_pin, i);
        delay(20);
    }
    lcd.print("Right!");
    delay(200);
    lcd.init();
    prev_command = 'r';
}

```

로봇 펫의 전진, 후진, 좌회전, 우회전, 돌기, 정지를 구현한 코드이다. car speed는 130으로 설정하였고, 각 동작마다 LCD에 해당 동작을 출력하도록 설정하였다. 각 동작마다 4개의 바퀴가 맞게 움직이도록 설정하였다. 약 좌회전과 약 우회전의 코드도 구현되어 있지만, 실제 로봇 펫의 움직임에 넣기에는 적절하지 않다고 판단하여서 실제로 사용은 하지 않았다.

다. 로봇 펫의 배고픔

```

void hungrypet(unsigned long timer)
{
    /*Serial.println(hungry);
    prehungry=hungry;
    delay(1000);*/
    if((hungry == 1) && (hungry_val<=1)){
        Serial.println("hungry");
        SmartCar_Turn();
        delay(2000);
        SmartCar_Stop();
        lcd.print("hungry");
        delay(200);
        lcd.init();
        mp3_play(4);
        delay(4000);
        hungry_val += 1;
    }
}

```

```

--
hungrypet(timer);
if ((timer % 81000 >= 0) && (timer % 81000 <= 1000)) {
    hungry = 1;
    hungry_val = 0;
}

```

timer의 값이 81000이 될 때마다 hungry 상태로 바뀌는 함수이다.

hungry 상태로 들어갈 시, 쿠키는 turn 하면서 LCD창에 배고프다는 메시지를 남기고, mp3_play() 함수로 구슬프게 짚는 소리를 낸다.

```

void feedpet(char value)
{
    if(value == 'j')
    { //밥주기
        /*prehungry=hungry;
        timer0_millis=0;
        prehungry=0;*/
        Serial.println("full");
        lcd.print("Yum Yum");
        delay(200);
        mp3_play(6);
        lcd.init();
        delay(4000);
        hungry = 0;
    }
}

```

배고픔 상태의 쿠키를 원상복구 시키려면, feedpet() 함수를 이용하여 쿠키에게 밥을 주는 명령을 수행하면 된다. 밥을 성공적으로 주었을 때, 쿠키는 LCD에 "Yum Yum" 메시지를 남기며 기쁘게 짚는다.

라. 로봇 펫의 무빙 패턴


```

void movingpet(unsigned long timer, char value) {
  /*Serial.println(moving);
  premoving = moving;
  delay(1000);*/
  if ((value == 'a') && (timer % 31000 >= 0) && (timer % 31000 <= 1000)) {
    for(pos = 0; pos <= 90; pos +=1) {
      pos2 += 1;
      LEar.write(pos);
      REar.write(pos2);
      delay(5);
    }
    /*for(pos2 = 0; pos2 <= 90; pos2 +=1) {
      REar.write(pos2);
      delay(5);
    }*/
    for(pos = 90; pos>=0; pos -=1) {
      pos2 -= 1;
      LEar.write(pos);
      REar.write(pos2);
      delay(5);
    }
    /*for(pos2 = 90; pos2>=0; pos2 -=1) {
      REar.write(pos2);
      delay(5);
    }*/
    Serial.println("Ear");
  }
  if ((value == 'a') && (timer % 51000 >= 0) && (timer % 51000 <= 1000)) {
    for(pos3 = 0; pos3 <= 170; pos3 +=1) {
      Tail.write(pos3);
      delay(3);
    }
    for(pos3 = 170; pos3>=0; pos3 -=1) {
      Tail.write(pos3);
      delay(3);
    }
    Serial.println("Moving");
  }
}

```

movingpet() 함수는 일정 딜레이(31000 ms)마다 귀를 쫑긋거리고, 일정 딜레이(51000ms)마다 꼬리를 흔드는 함수이다. 명령을 내릴 때까지 가만히 있는 것 보다는, 혼자서 조금씩 움직이는 것이 더 실제 강아지에 가까워 보여서 구현하였다.

```

    }
    if ((value == 'a') && (timer % 71000 >= 0) && (timer % 71000 <= 1000)) {
        SmartCar_GoBack();
        Serial.println("Moving");
    }
}

void boringpet(unsigned long timer, char value) {
    if (boring_val % 6 == 5) {
        for(pos = 90; pos>=0; pos -=1) {
            pos2 -= 1;
            LEar.write(pos);
            REar.write(pos2);
        }
        Serial.println("Boring");
        lcd.print("Boring");
        delay(1000);
        lcd.init();
    }
}

    if ((timer % 11000 >= 0) && (timer % 11000 <= 1000)) {
        boring_val += 1;
        Serial.println("boring_val++");
    }
}

```

boringpet()함수는 일정 시간동안 명령을 받지 않으면 심심해하는 함수이다. boring_val 값이 증가하면 쿠키는 귀를 쫑긋거리며 LCD에 지루하다는 메시지를 띄우게 된다.

마. 로봇 펫 컨트롤

```

    else if(value == '4') {                //돌아
        SmartCar_Turn();
    }
}

```



```

else if(value == '6'){           //손
    for(pos4 = 30; pos4 <= 120; pos4 +=1) {
        Hand.write(pos4);
        delay(10);
    }
    for(pos4 = 120; pos4>=30; pos4 -=1) {
        Hand.write(pos4);
        delay(10);
    }
    lcd.print("Hand");
    delay(200);
    lcd.init();
}
else if(value == 'l'){           // 좌회전 명령
    SmartCar_Left();
}
else if(value == 'r'){           // 우회전 명령
    SmartCar_Right();
}
else if(value == 'q'){           // 약좌회전 명령 (실제사용x)
    SmartCar_sLeft();
}
else if(value == 'w'){           // 약우회전 명령 (실제사용x)
    SmartCar_sRight();
}
else if(value == '9'){           //귀 쫓긋
    for(pos = 0; pos <= 90; pos +=1) {
        LEar.write(pos);
    }
    for(pos2 = 0; pos2 <= 90; pos2 +=1) {
        REar.write(pos2);
        delay(5);
    }
    for(pos = 90; pos>=0; pos -=1) {
        LEar.write(pos);
    }
    for(pos2 = 90; pos2>=0; pos2 -=1) {
        REar.write(pos2);
        delay(5);
    }
    lcd.print("EAR");
    delay(200);
    lcd.init();
}

```

```

void controlPet(char value)
{
    if(value == 'f'){ // 전진 명령
        SmartCar_Go();
    }
    else if(value == 'k'){//쿠키~ 하고 부르면 멍멍하고 짹음
        for(pos = 0; pos <= 90; pos +=1) {
            pos2 += 1;
            LEar.write(pos);
            REar.write(pos2);
            delay(5);
        }
        /*for(pos2 = 0; pos2 <= 90; pos2 +=1) {
            REar.write(pos2);
            delay(5);
        }*/
        for(pos = 90; pos>=0; pos -=1) {
            pos2 -= 1;
            LEar.write(pos);
            REar.write(pos2);
            delay(5);
        }
        /*for(pos2 = 90; pos2>=0; pos2 -=1) {
            REar.write(pos2);
            delay(5);
        }*/
        RandomBark();
        delay(4000);
    }
    else if(value == 'b'){ // 후진 명령
        SmartCar_Back();
    }
    else if(value == 's'){ // 정지 명령
        SmartCar_Stop();
    }
}

```

```

else if(value == '7'){           //산책 (귀, 꼬리, 음성)
    mp3_play(6);
    delay(50);
    for(pos = 0; pos <= 90; pos +=1) {
        pos2 += 1;
        LEar.write(pos);
        REar.write(pos2);
        delay(5);
    }
    /*for(pos2 = 0; pos2 <= 90; pos2 +=1) {
        REar.write(pos2);
        delay(5);
    }*/
    for(pos = 90; pos>=0; pos -=1) {
        pos2 -= 1;
        LEar.write(pos);
        REar.write(pos2);
        delay(5);
    }
    /*for(pos2 = 90; pos2>=0; pos2 -=1) {
        REar.write(pos2);
        delay(5);
    }*/
    for(pos3 = 0; pos3 <= 170; pos3 +=1) {
        Tail.write(pos3);
        delay(3);
    }
    for(pos3 = 170; pos3>=0; pos3 -=1) {
        Tail.write(pos3);
        delay(3);
    }
    lcd.print("WOOOOOW!");
    delay(200);
    lcd.init();
}

```

어플에서 음성을 인식한 뒤, 해당 단어에 대응하는 문자열을 보내면 로봇 펫이 작동하는 형식이다. 명령은 쿠키, 손, 전진, 후진, 정진, 돌아, 좌회전, 우회전, 귀 쫓긋, 산책이 있다. 각 명령을 수행할 때마다 해당 명령이 LCD에 표시되도록 구현하였다.

바. 기타 동작

```
void SmartCar_GoBack() {
    SmartCar_Go();
    delay(500);
    SmartCar_Stop();
    delay(500);
    SmartCar_Back();
    delay(500);
    SmartCar_Stop();
}

/*void SmartCar_Sonic() {
    if()
}*/

void RandomBark(){ //랜덤으로 짓는 함수
    int a;

    lcd.print("YES!!");
    delay(200);
    lcd.init();

    a = random(2);

    if(a==0) {
        mp3_play(6);
    }
    else if(a==1) {
        mp3_play(1);
    }
    else if(a==2) {
        mp3_play(2);
    }
}
```

SmartCar_GoBack() 함수는 (라)의 무빙 패턴에 들어있는 앞뒤로 반복하여 움직이는 동작을 구현한 함수이고, RandomBark는 응답할 때 랜덤한 순서로 mp3_play()로 짓는 함수이다.

2) Android Smart Phone Application

```

if(mResult.contains("손")){
    if(level>1){
        if(bt.getServiceState() == BluetoothState.STATE_CONNECTED){
            bt.send( data: "6", CRLF: true);
            ++exp;
        }
        else{
            Toast.makeText( context: MainActivity.this, text: "블루투스 연결이 안되어있어요", Toast.LENGTH_SHORT).show();
        }
    }
    else {
        Toast.makeText( context: MainActivity.this, text: "레벨이 부족해요", Toast.LENGTH_SHORT).show();
    }
    Log.d(TAG, msg: "you call hand");
}

```

음성인식 결과값에 '손' 과 같은 특정 명령 키워드가 포함 되어있으면 블루투스에 그에 해당하는 문자코드를 전송한다. 전송과 동시에 레벨에 필요한 경험치 또한 상승한다. 안정성을 위해 블루투스 연결이 안되어있을 때는 전송이 되지 않고 연결이 안되었다는 토스트 메시지를 출력한다.

```

if(exp>=levelHuddle){
    ++level;
    exp=0;
    levtext.setText(String.valueOf(level));
    dbHelper.update(level);

    Toast.makeText( context: MainActivity.this, text: "레벨이 올랐어요", Toast.LENGTH_SHORT).show();
}

```

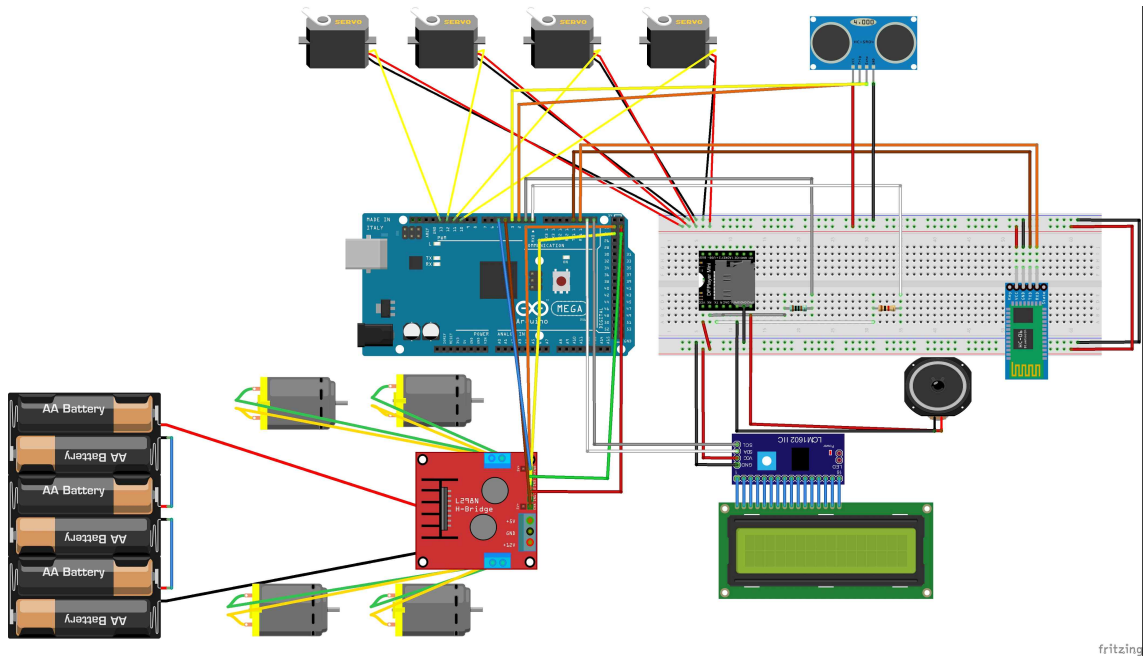
로봇펫을 성장시키는 재미를 주기 위해 레벨 시스템을 만들었다. 명령을 수행할 때 마다 경험치가 쌓이고 경험치에 도달하면 레벨이 오른다. 레벨 정보는 SQLite 로 구현한 내부 DB에 저장된다. 특정 명령은 레벨 제한을 두어 현재 레벨이 제한보다 낮으면 전송하지 않고 레벨이 부족하다는 토스트 메시지를 출력하게 하였다.

2.3 과제 수행 결과

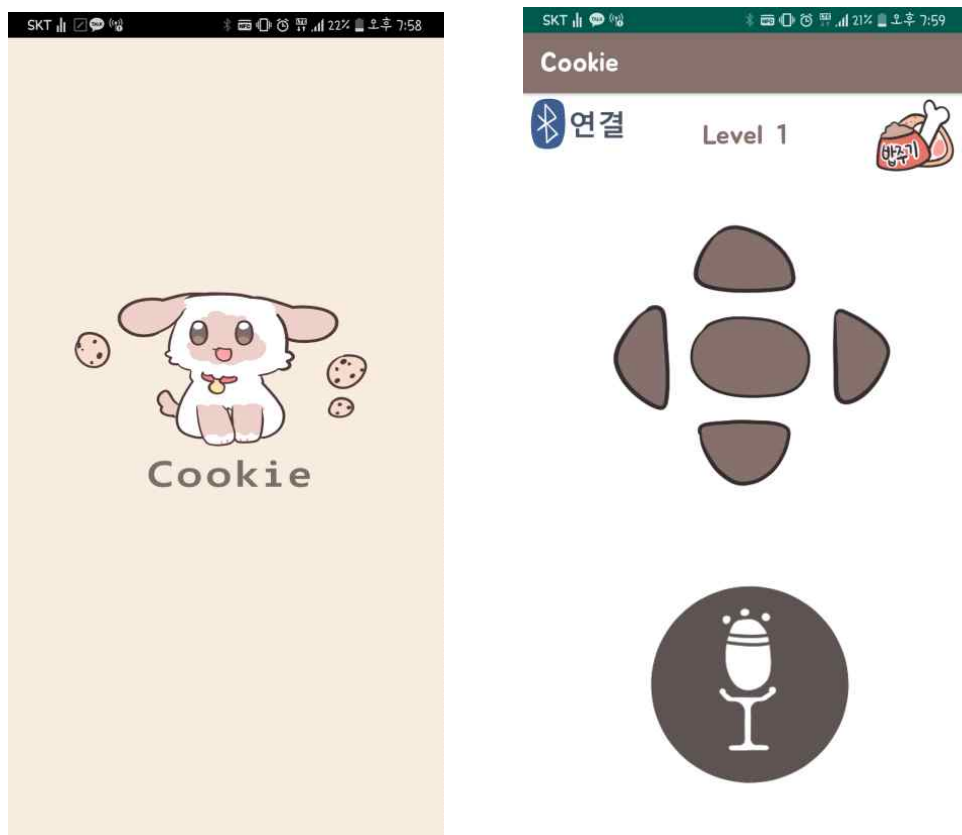
1) 전체 시스템 구성도

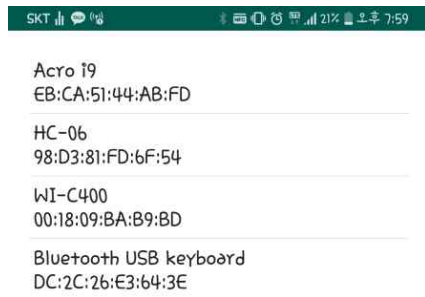


2) Board 구성도



3) Android Smart Phone Application Application



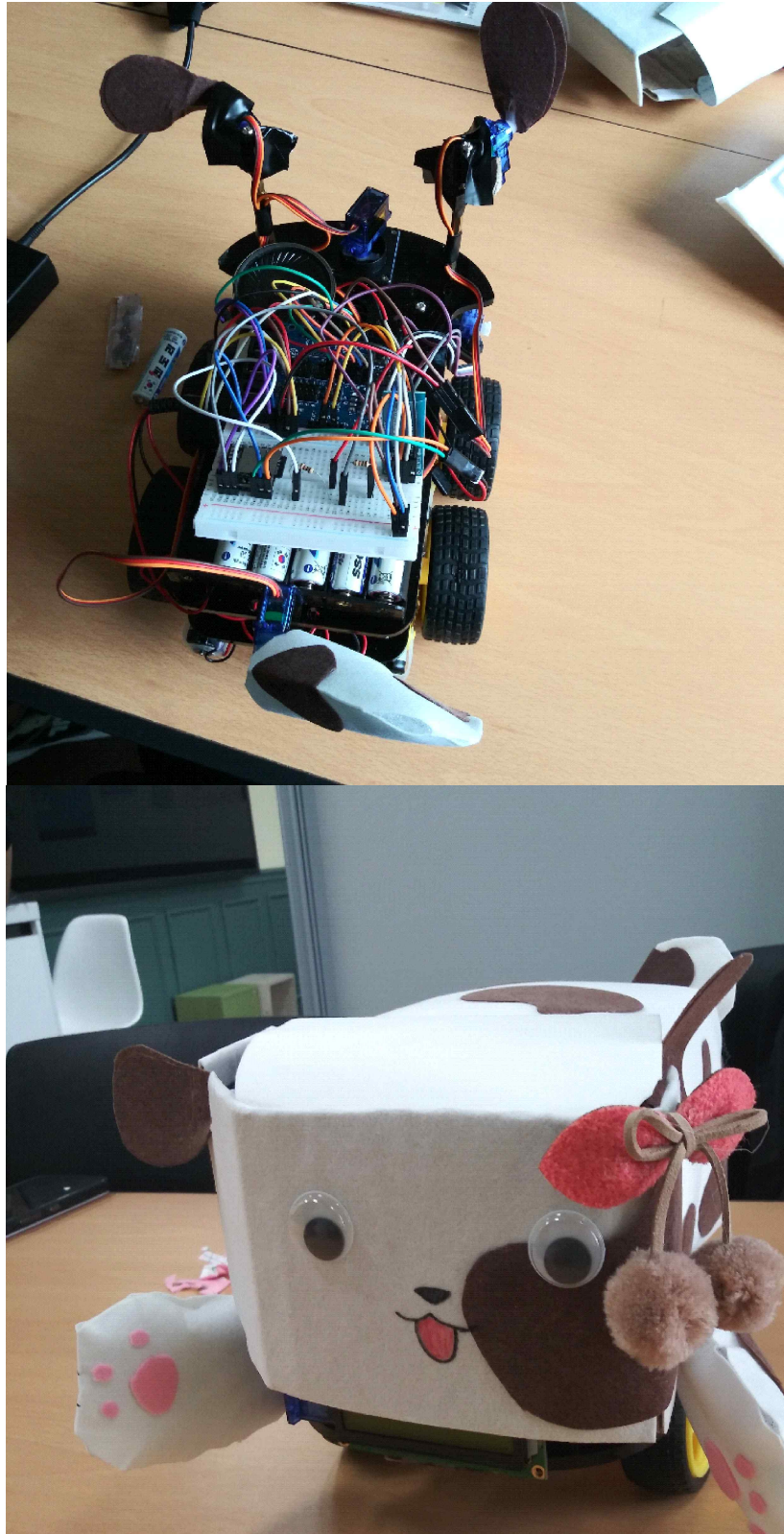


로딩페이지 : 로봇펫 쿠키의 로고 이미지와 제품명이 3초간 노출된다.

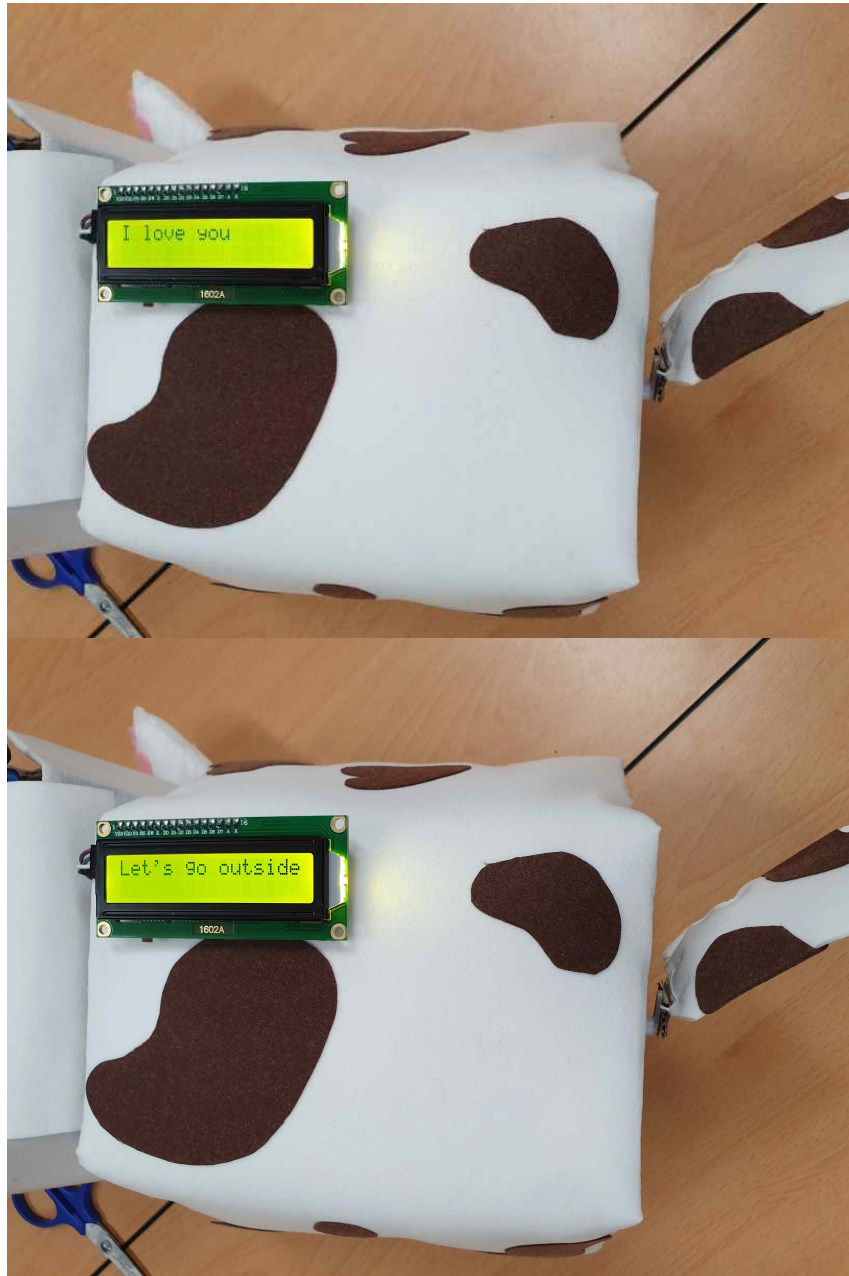
메인페이지 : 블루투스과 기기 연결 버튼, 레벨 표시, 밥주기 버튼, 상하좌우와 정지로 구성된 펫 움직임 컨트롤 버튼, 음성인식 버튼으로 구성되어 있다.

블루투스 연결페이지 : 휴대폰과 로봇펫의 블루투스 모듈(HC-06)을 연결 할 수 있다.

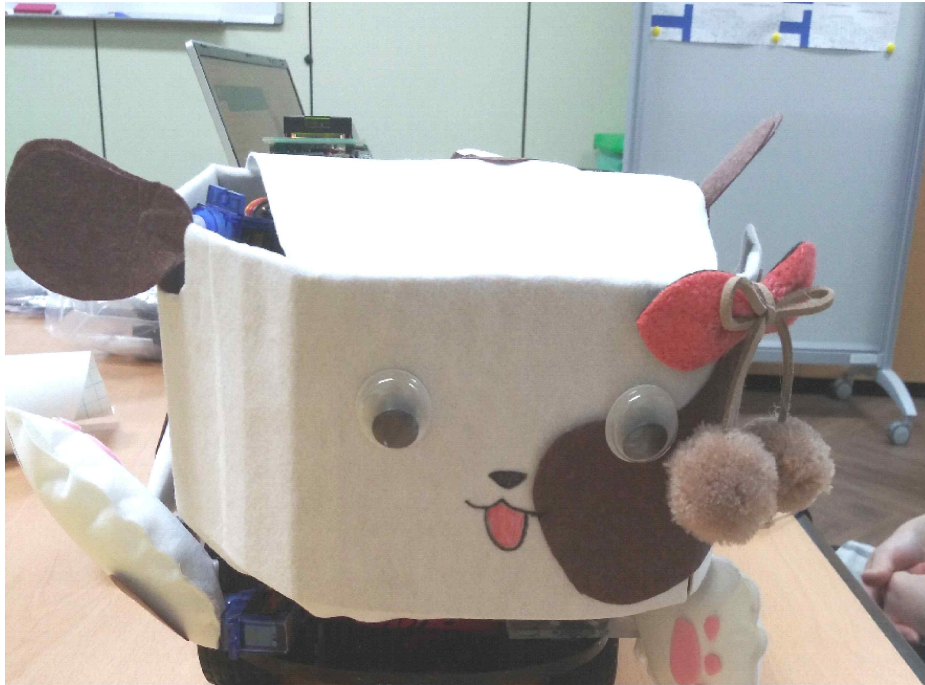
4) 시스템 동작 및 결과'



아두이노 MEGA 보드에 바퀴용 DC모터, 로봇펫의 손과 꼬리와 귀를 제어 할 서보모터, 부딪힘 방지용 초음파 센서, 로봇펫의 말이 표시 되는 LCD 모듈, 짖는 소리 구현을 위한 MP3 모듈, 블루투스 모듈을 연결 하여 로봇펫의 내부 하드웨어를 구현하였다. 그리고 최대한 가볍고 저렴하며 따뜻하고 부드러운 촉감을 살릴 수 있도록 솜, 접착 부직포, 골판지 등을 이용해 외형을 입혔다.



실제 강아지와 최대한 가깝게 구현하기 위하여 명령을 받지 않았을 때도 혼자서 움직이도록 구현하였다. 또한 사용자와의 정서적 교감과 사용자의 심리적 안정감 향상을 위해 LCD모듈을 통해 스스로 주인에게 말을 건네도록 하였다.



스마트폰 Application을 켜고 이름을 부르거나 "손", "산책 가자." "돌아!" 등 명령을 하면 로봇펫이 그에 맞는 동작을 하고 소리로 호응한다. 그러다 강아지가 배고픈 상태가 되면 명령을 받지 않고 짹짹 소리를 내며 빙글 빙글 돌게 하여 실제 강아지와 최대한 비슷하게 배고픈 상태를 구현하였다. Application에서 밥을 주면 밥을 먹는 소리가 나면서 배부른 상태가 된다.

3. 결론

3.1 결론

우리는 기존의 로봇펫들이 실제 애완동물을 기르는 것과 많은 차이가 있다는 점을 개선하기 위하여 본 프로젝트를 실시하였으며 그 결과 여러 부분에서 개선이 이루어졌다. 우선 레벨 시스템을 도입하여 펫이 성장하는 것을 가시적으로 확인할 수 있게 되었으며 레벨이 높아짐에 따라 수행 가능한 명령이 추가 되어 성장시키는 즐거움을 주었다. 또한 밥먹기 시스템과 지루함 시스템을 통해 펫이 배가 고프거나 일정 시간동안 놀아주지 않고 방치하면 해결 될 때까지 소리와 몸짓으로 표현하게 하였다. 그리고 펫에게 명령을 내리지 않았을 때도 스스로 움직이고 사용자와 소통을 시도 하도록 구현하였다. 이런 기능들을 추가함으로써 조금 더 실제 애완동물 육성과 흡사한 형태의 로봇펫이 완성 되었다.

1) 하드웨어적인 부분에서 몸체를 부직포로 감싸는 것 보다 털 달린 천과 솜으로 감싸는 것이 더욱 실제 강아지에 가까웠을 것이라 생각한다. 또한 사족 보행을 구현하였다면 더욱 실감나는 움직임을 수행할 수 있었을 것이다.

3.3 추진 일정

[illegible]

3.4 역할분담

학 번	성 명	구성원별 역할
201469118	이규원	아두이노를 통한 로봇펫 명령 수행 기능 구현 안드로이드 앱 프론트엔드 및 UI 디자인 로봇펫 밥주기 기능 구현 로봇펫 외형 제작 및 디자인
201624428	김범미	안드로이드 앱 백엔드, SQLite 기반 앱 데이터베이스 설계 안드로이드 앱 오픈소스 음성인식 API 조사 및 적용 로봇펫 레벨업 기능 및 짖기 기능 구현 로봇펫 음성 데이터 수집
201612148	전양희	아두이노 메가를 기반으로 각종 센서와 모듈을 적용한 아 두이노 보드 구현 및 설계 로봇펫과 어플리케이션 간 블루투스 통신 기능 구현 로봇펫 충돌 방지 및 대기 상태 시 행동 구현 로봇펫 내부 몸체 조립 및 제작

3.5 참고 문헌

- [1] Arduino, Product page, <https://www.arduino.cc/en/Main/Products>
- [2] Android Studio, developers page, <https://developer.android.com/studio/features/>