**3조 기말 프로젝트 레포트**

**모바일 시스템 및 응용**

김민종 20092518

이지훈 20112584

신현욱 20111470

성진하 20132528

지도 교수 : 이강희

**- 목차 -**

1. 실행 방법

1.1 Arduino 설치

1.2 Proessing 및 라이브러리 설치

1.3 로봇 및 비전 프로그램 세팅

1. 로봇 및 소스코드 설명

2.1 로봇 구성

2.2 소스코드 설명

1. 실행 시나리오

3.1 Velocity 함수 동작 시나리오

3.2 Angle 함수 동작 시나리오

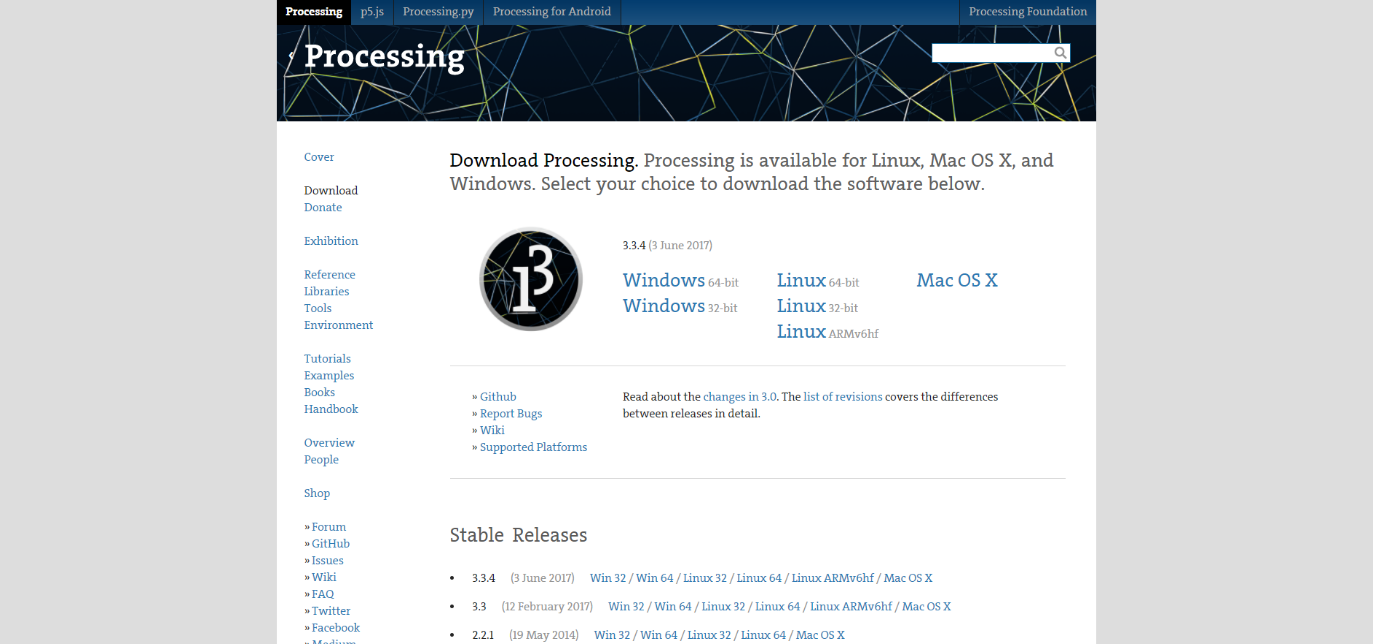
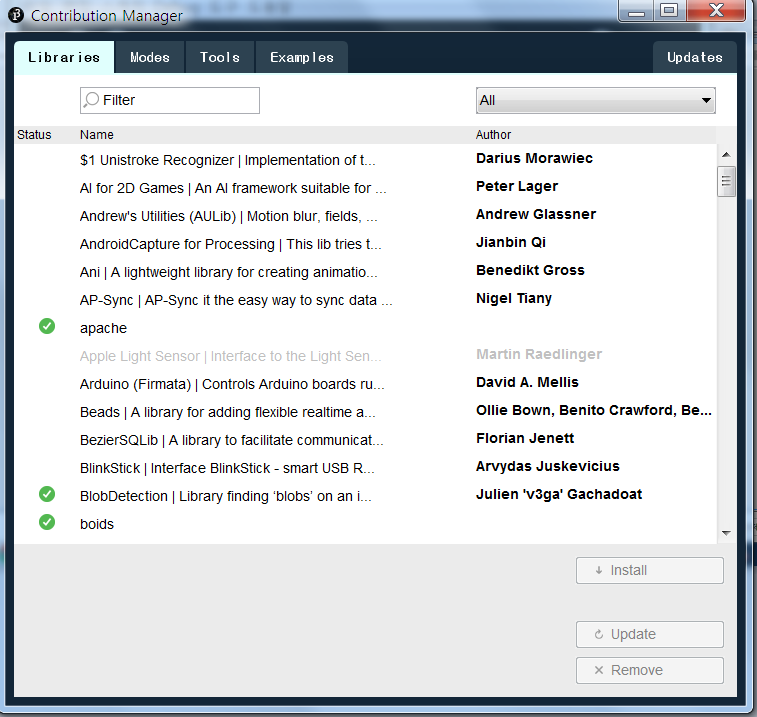
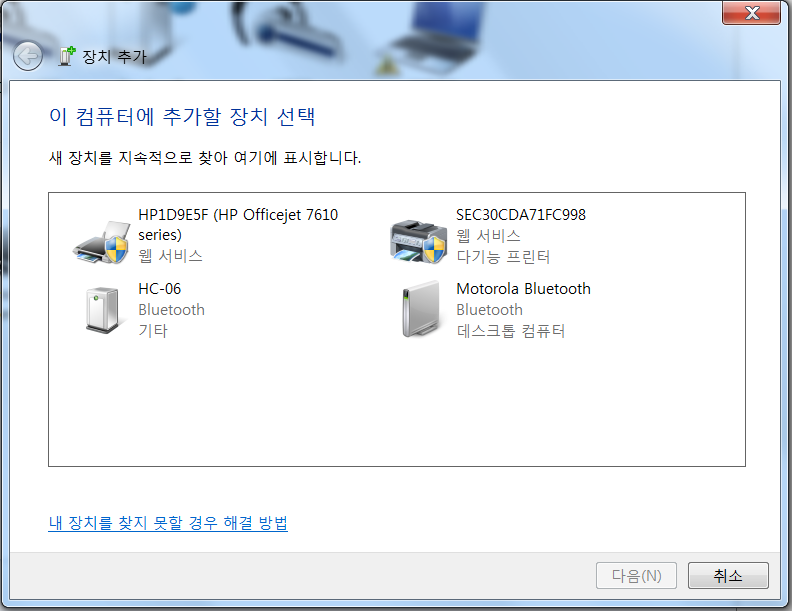
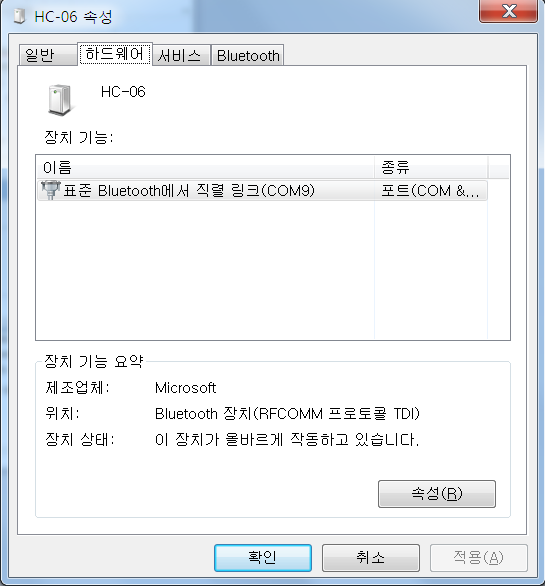
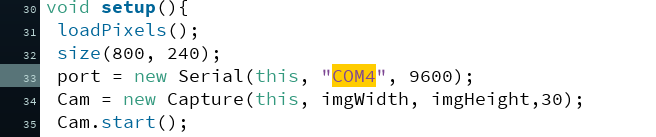
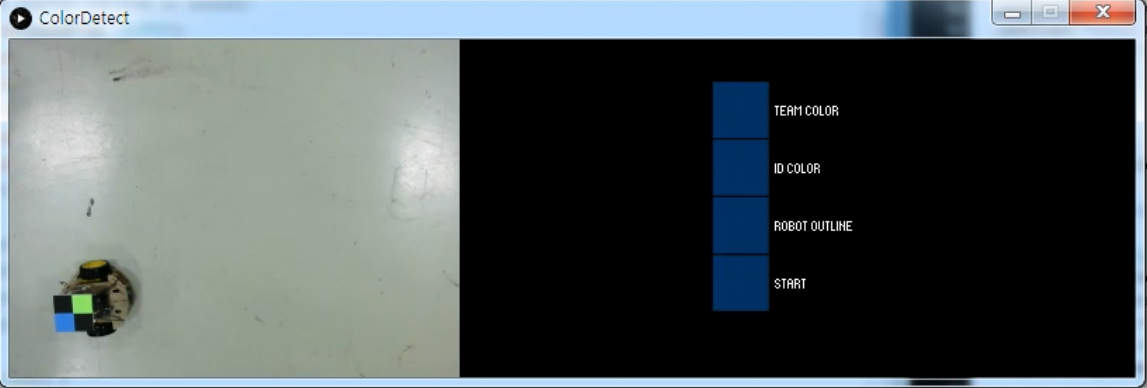
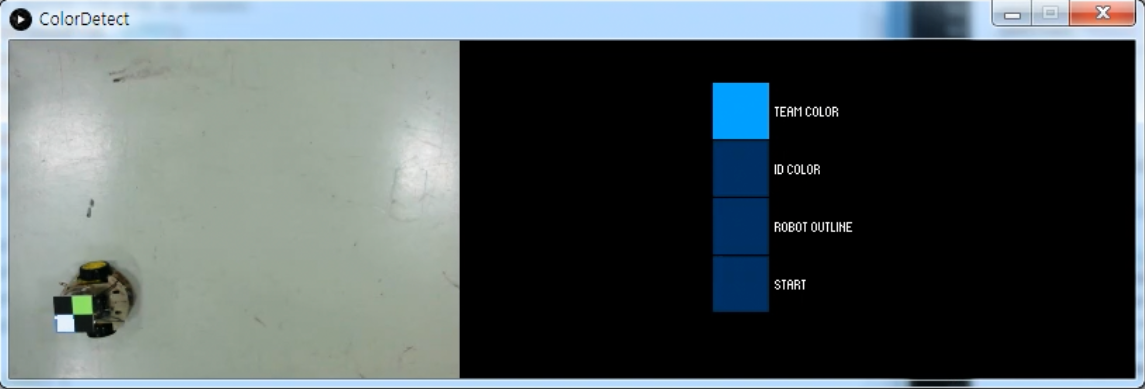
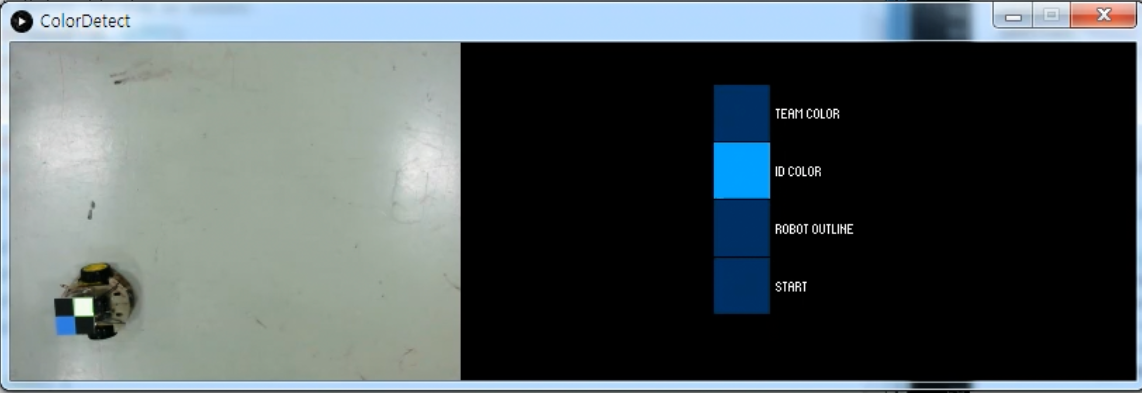
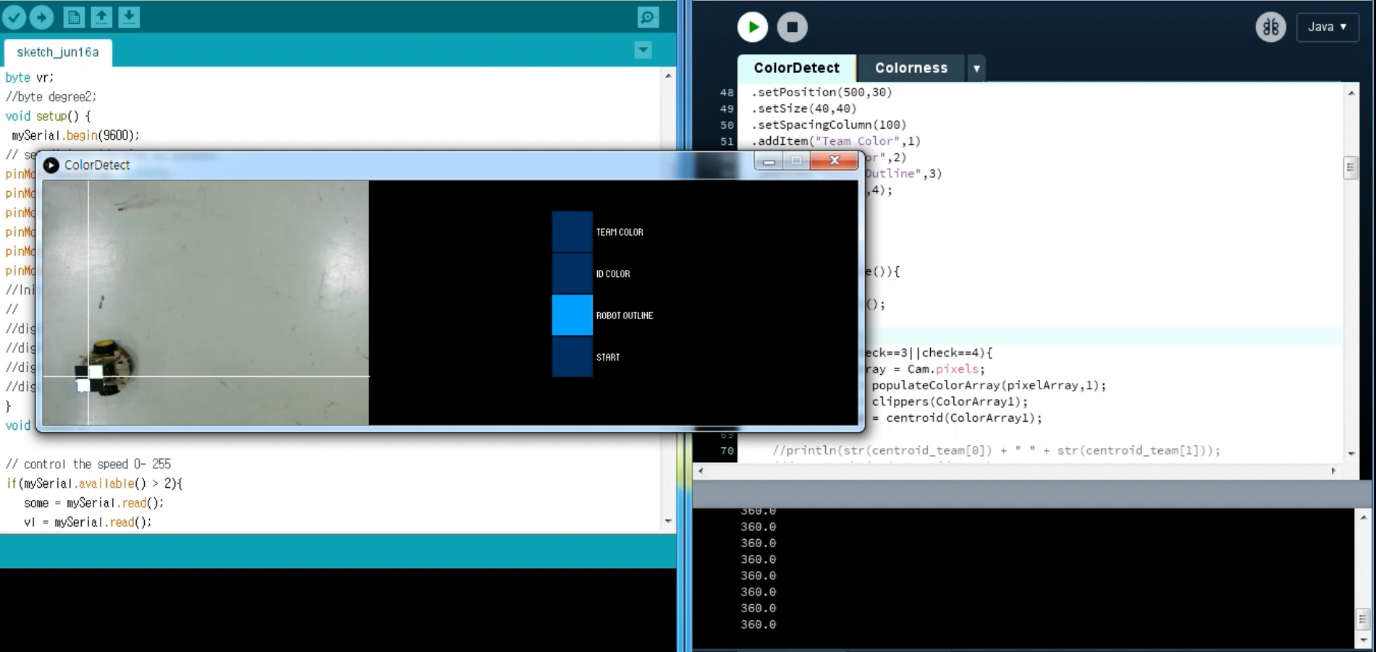
3.3 Position 함수 동작 시나리오

1. 프로젝트 분석
2. **실행 방법**
   1. Arduino 설치
   2. 아두이노 공식 사이트의 다운로드 페이지로 접속한다.

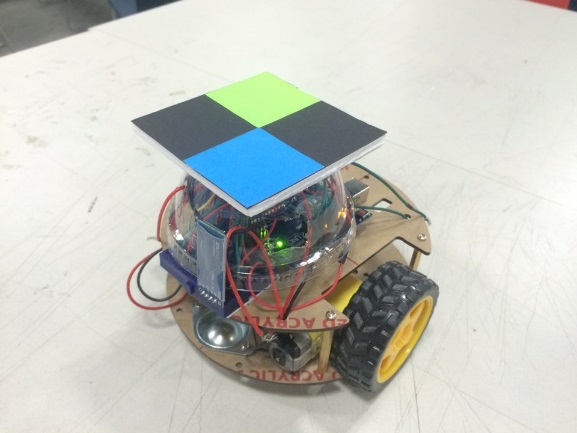
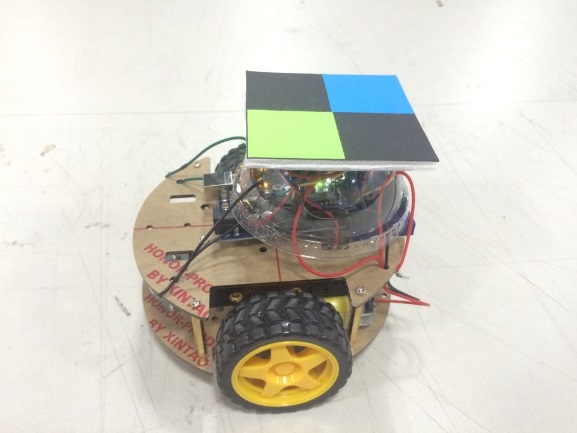
(<https://www.arduino.cc/en/Main/Software>)

* 1. 자신의 컴퓨터 환경과 맞는 파일을 다운한 후 exe 파일을 실행시켜 설치를 마무리한다.
  2. Processing 및 라이브러리 설치
     1. 프로세싱 공식 사이트의 다운로드 페이지로 접속한다.

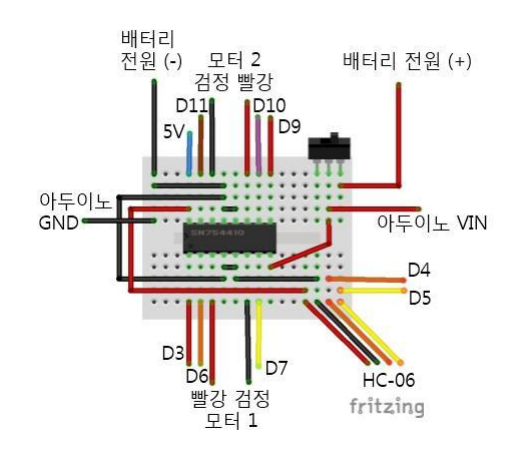
(<https://processing.org/download/>)

1. 자신의 컴퓨터 환경과 맞는 파일을 다운한 후 압축을 해제하면 설치가 마무리 된다.
2. 라이브러리 설치를 위해 Processing.exe를 실행하여 빈 스케치를 연다.
3. 상단 메뉴에서 ‘스케치->내부 라이브러리->라이브러리 추가하기’ 순으로 클릭하여 아래의 그림 같은 창을 연다.
4. 왼쪽 상단의 검색창에 ‘video’를 입력하고 아래 그림과 같은 라이브러리를 선택한 후 ‘install’ 버튼을 클릭하여 라이브러리를 설치한다.
5. 검색창에 ‘ControlP5’를 입력하여 검색된 라이브러리를 선택한 후 ‘install’ 버튼을 클릭하여 라이브러리 설치를 마무리한다.
   1. 로봇 및 비전 프로그램 세팅
6. 함께 첨부된 ‘RobotSoccer.ino’ 아두이노 파일을 실행한 후 로봇의 전원을 끄고 로봇과 컴퓨터를 USB Serial Port로 연결한다.
7. 상단 메뉴에서 ‘툴->포트’ 순으로 클릭한 후 자신의 로봇이 연결된 포트(Arduino/Genuino Uno)를 클릭하여 포트를 변경해준다.
8. 포트 설정까지 끝낸 후 좌측 상단의 확인 버튼(컴파일)을 눌러 컴파일하고 오른쪽의 업로드 버튼을 눌러 로봇으로 소스를 업로드 해준다.
9. 연결된 USB Serial Port를 분리하고 로봇의 전원을 켜고 컴퓨터의 ‘제어판->하드웨어 및 소리->장치추가’로 들어가면 화면에 ‘HC-06’이 검색된 것을 확인 할 수 있다.
10. 검색된 ‘HC-06’을 더블클릭 하고 아래 화면에서 ‘장치의 연결 코드 입력’으로 들어가 ‘1234’를 입력하고 확인 버튼을 눌러 드라이버를 설치한다.
11. 제어판의 ‘장치 및 프린터’로 들어가 방금 설치된 ‘HC-06’의 속성의 하드웨어 탭에서 어떤 포트로 연결되었는지 확인한다.
12. 위의 경우 ‘COM9’포트로 연결된 것을 확인 할 수 있다. Arduino로 다시 들어가 방금 연결 확인한 포트로 포트를 재설정한다.(위의 경우 ‘COM9)
13. 함께 첨부 된 ‘RobotSoccer.pde’ processing 파일을 열어 소스 중 33번쨰 줄의 포트 이름을 연결된 포트 이름으로 변경한다.
14. ‘RobotSoccer.pde’를 실행하면 아래와 같은 실행화면이 열리게 된다.
15. Team Color를 설정해주기 위해 우측 Radio 버튼 중 ‘Team Color’를 클릭하고 좌측 화면에서 해당 색을 클릭한다.
16. ID Color를 설정해주기 위해 우측 Radio 버튼 중 ‘ID Color’를 클릭하고 좌측 화면에서 해당 색을 클릭한다.
17. 우측 Radio 버튼 중 ‘Robot Outline’을 클릭하면 현재 로봇의 중심 위치, Team Color, ID Color등이 화면에 실시간으로 인식되어 표시되고 Processing 콘솔 창에 현재 로봇의 각도가 표시된다.
18.  우측 Radio 버튼 중 ‘START’를 클릭하면 로봇이 자동으로 position1, position2, position3, position4을 거쳐 이동하게 된다.
19. **로봇 및 소스코드 설명**
    1. 로봇 구성
20. 로봇 외관

로봇 외관은 아래 사진과 같이 구성 하였다.

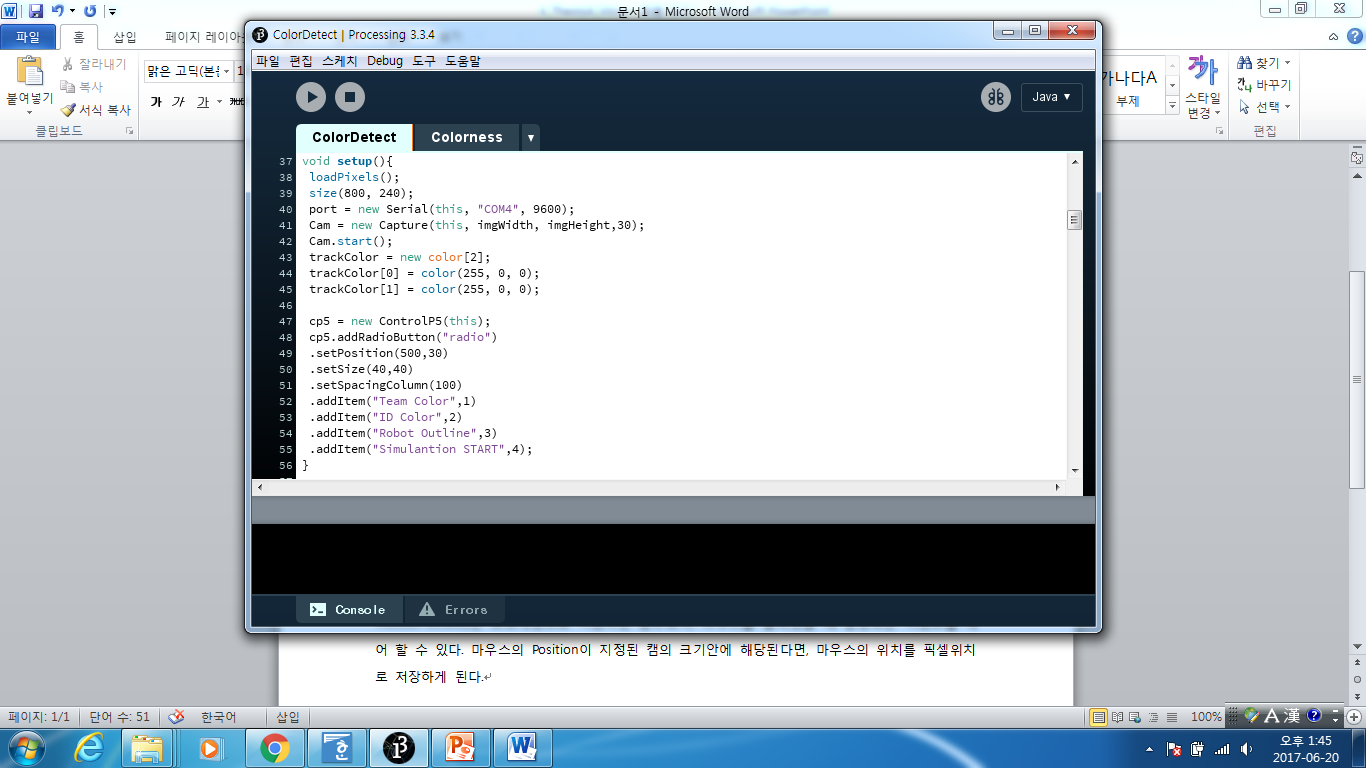


1. 로봇 회로도

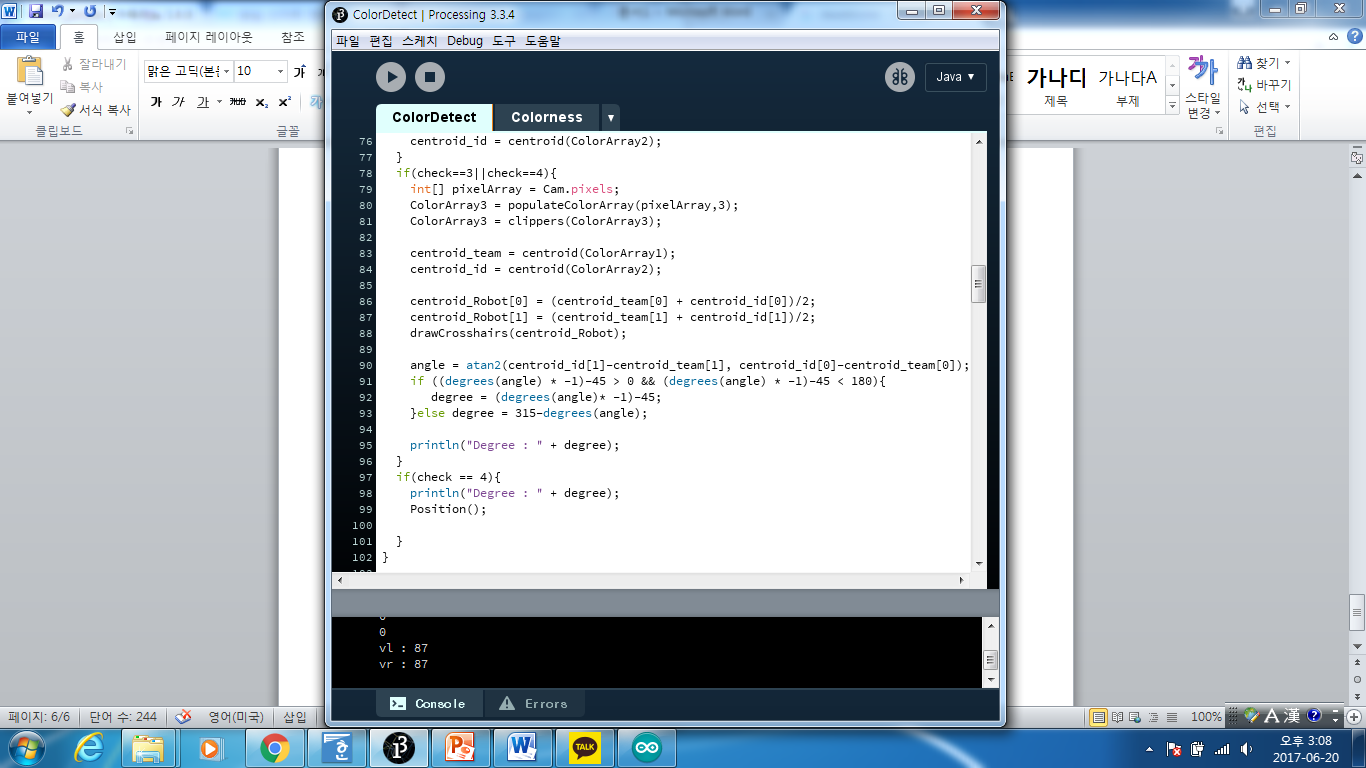
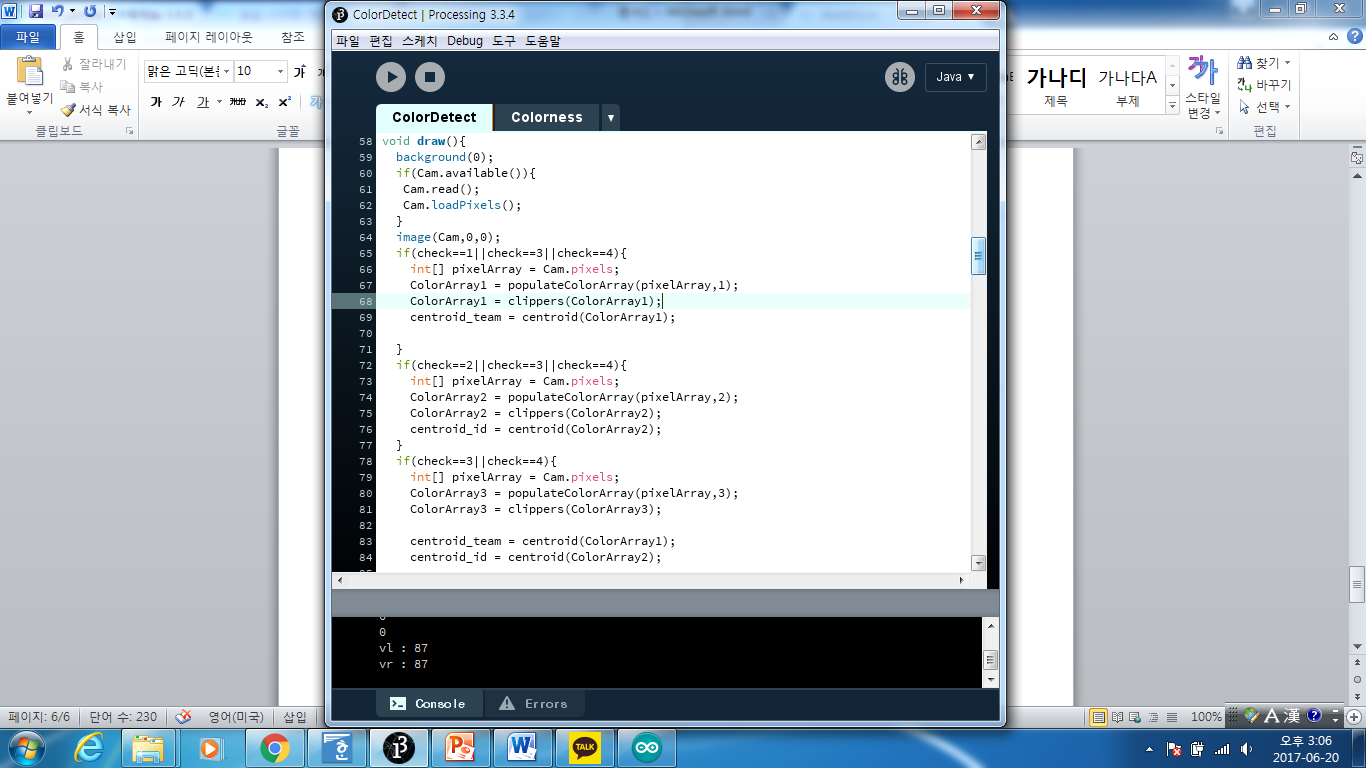
로봇은 위 사진과 같은 회로도로 핀들을 연결 하였다.

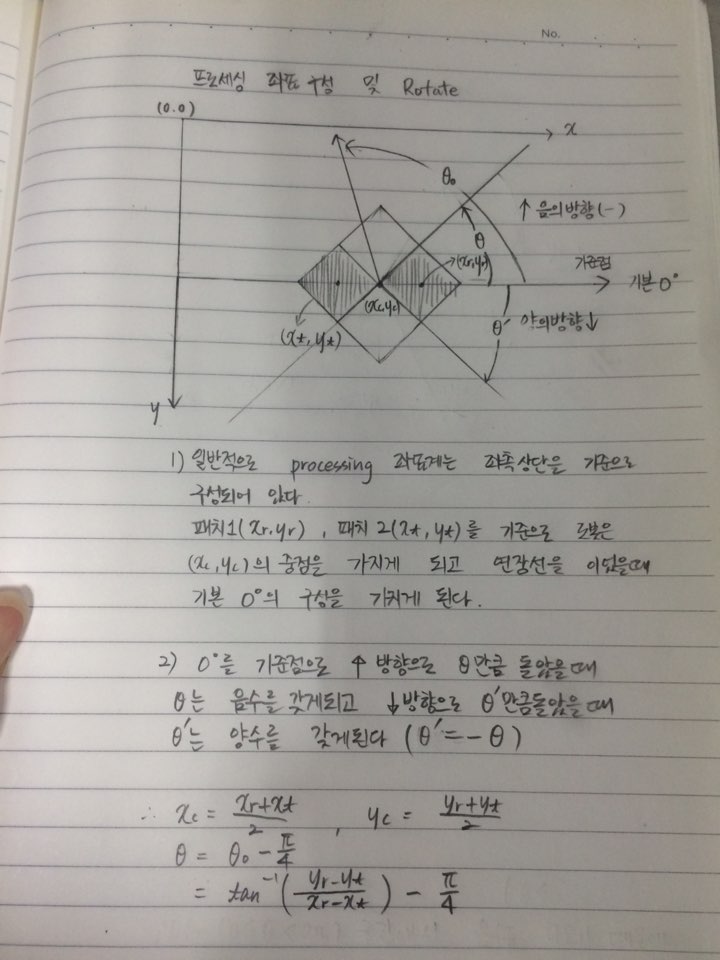
* 1. 소스코드 설명

1. Setup()

Setup()함수 에서는 기본적으로 캠의 크기와 위치 프레임을 세팅하고, Arduino와 Processing간의 블루투스 통신을 위한 포트를 설정하고 저장한다. 또한 프로그램 내부의 Radio 버튼을 위해 ControlP5를 사용하여 초기 버튼을 세팅한다.

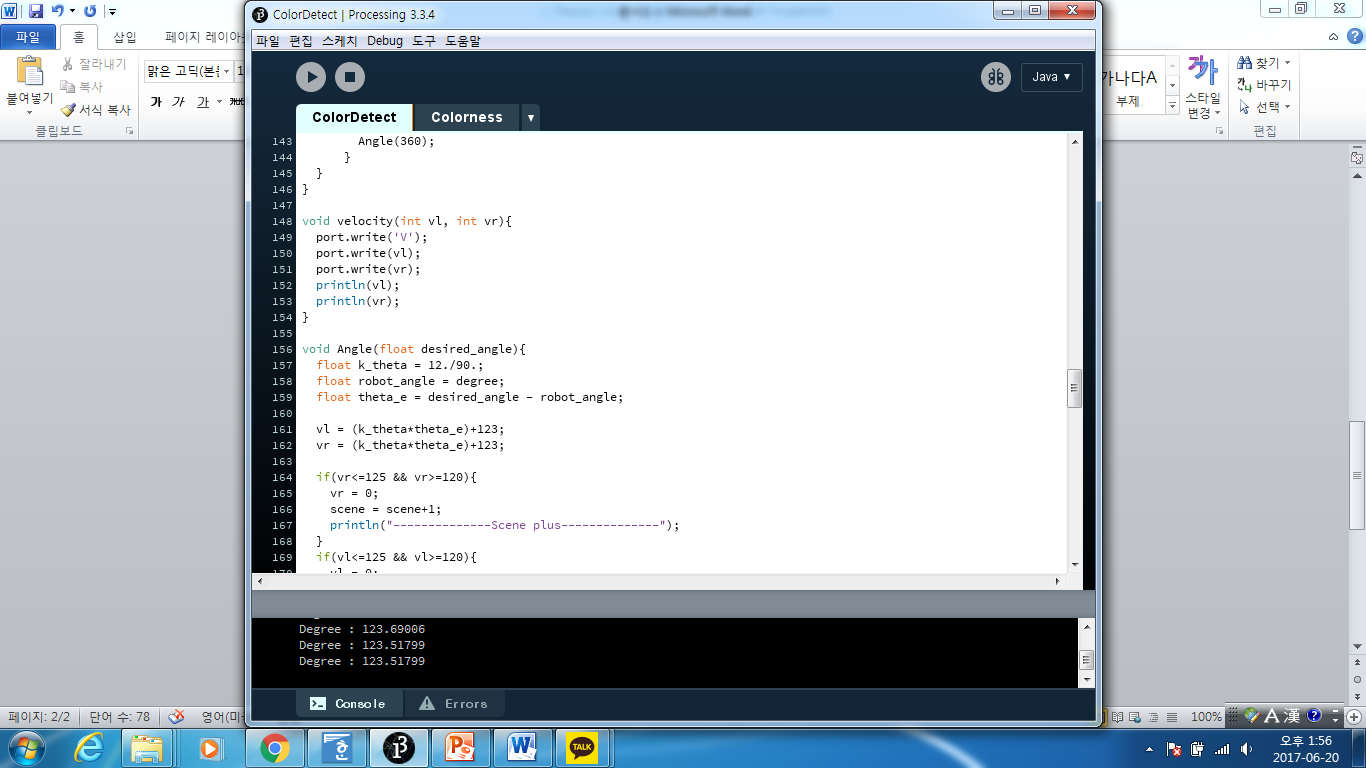
1. Draw()

Draw()함수 에서는 Cam을 프레임속도만큼 읽고, 캠을 픽셀영역으로 나눈다. 캠의 위치를 설정하고, 조건에 해당되면, 프로그램의 모든 구성요소들을 컨트롤한다. 영상처리부터 시작하여, 로봇의 패치를 인식하고, 인식된 패치의 중점을 통하여 로봇의 중심좌표 및 로봇의 각도를 뽑아내게 된다. 조건1과 조건2로 두 개의 패치를 인식하고, 조건3은 조건1,2에서 인식된 영역을 동시에 만족하며, 중심점을 찾고 로봇의 영역을 지정하며, 각도를 산출할 수 있도록 한다. 로봇은 Processing 좌표계를 사용하며, 좌표계를 통하여, 좌측 Rotation을 기준으로 360도 방향으로 인지하게 된다.

Processing 좌표계는 좌측상단(0,0)을 기준으로 시작된다. 로봇은 실제로 그림과 같은 방향일 경우 기준 0도가 되게 되는데 atan2()를 통하여 로봇의 각도를 산출하게 된다. 기준점을 기준으로 위쪽 방향으로 돌렸을 경우에 음의 각도로 인식하고, 아래쪽 방향으로 돌렸을 경우에 양의 각도로 인식하게되어 θ’ = - θ가 된다. 0≤θ<180 와 0≤θ<360 범위를 나누고 다음과 같이 정의하면 0≤θ≤360의 각도를 산출 할 수 있게 된다.

(1) 0≤θ<180 : 기준점으로 위의 영역에 해당되므로 θ를 음수로 변환 시켜 주면 된다.

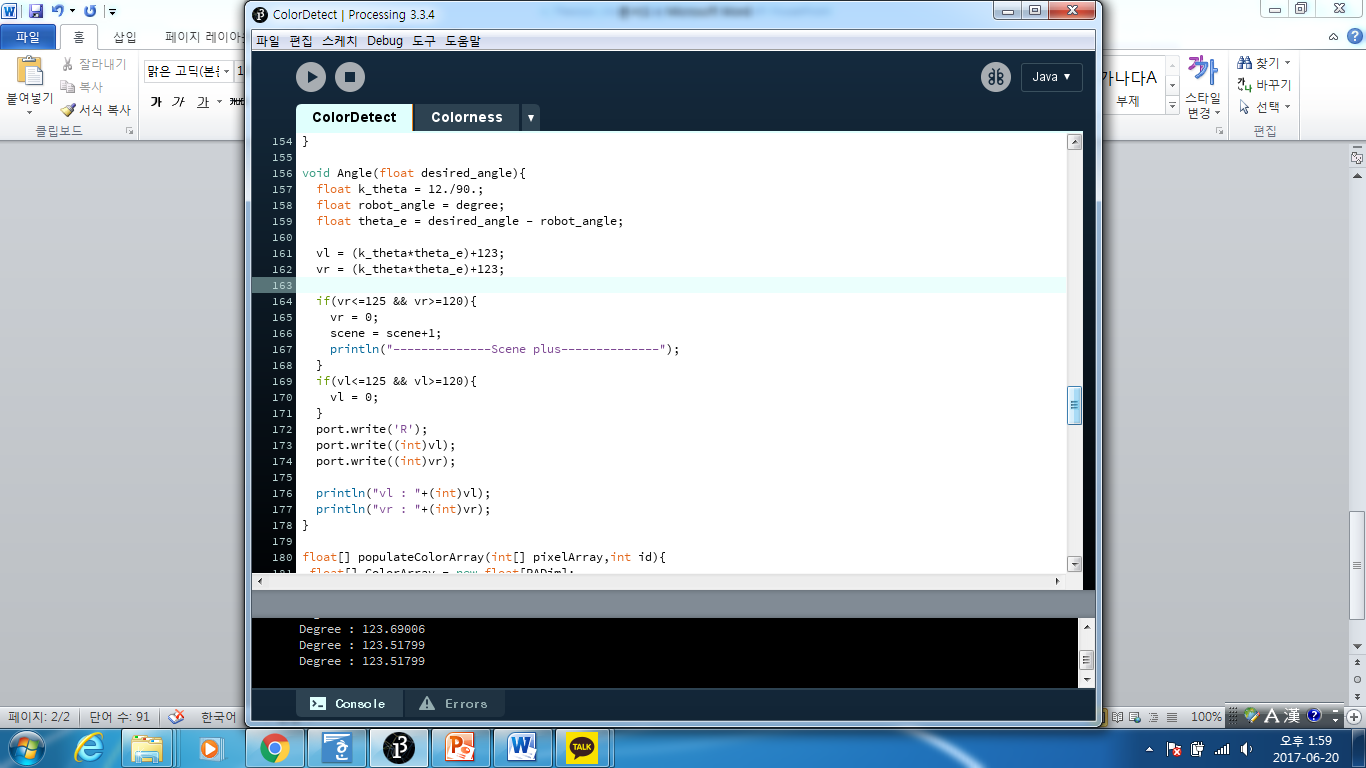
(2) 0≤θ<360 : 기준점으로 아래의 영역에 해당되므로 θ를 360- θ로 변환 시켜 주면 된다.

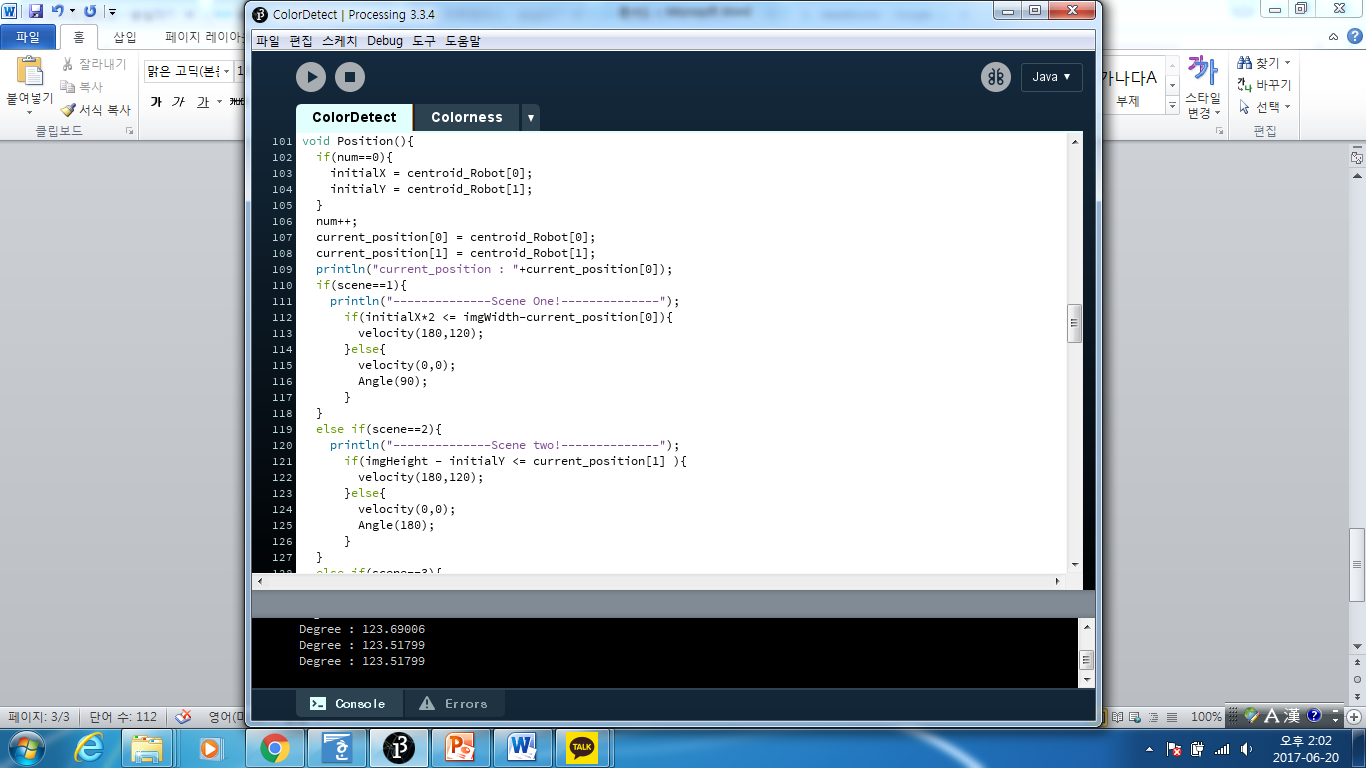
1. Velocity()

Velocity() 함수는 vl, vr 변수를 받으면 블루투스를 통해 port.write()를 사용하여 값을 Arduino로 넘겨주게 된다. 로봇의 DC모터 측정결과 왼쪽 모터와 오른쪽 모터의 DeadZone이 다른 것을 확인 하였다.

|  |  |
| --- | --- |
| 바닥에 놓지 않았을 경우 | 바닥에 놓았을 경우(일직선으로) |
| Left DC Motor : 98 | Left DC Motor : 150 |
| Right DC Motor : 50 | Right DC Motor : 90 |

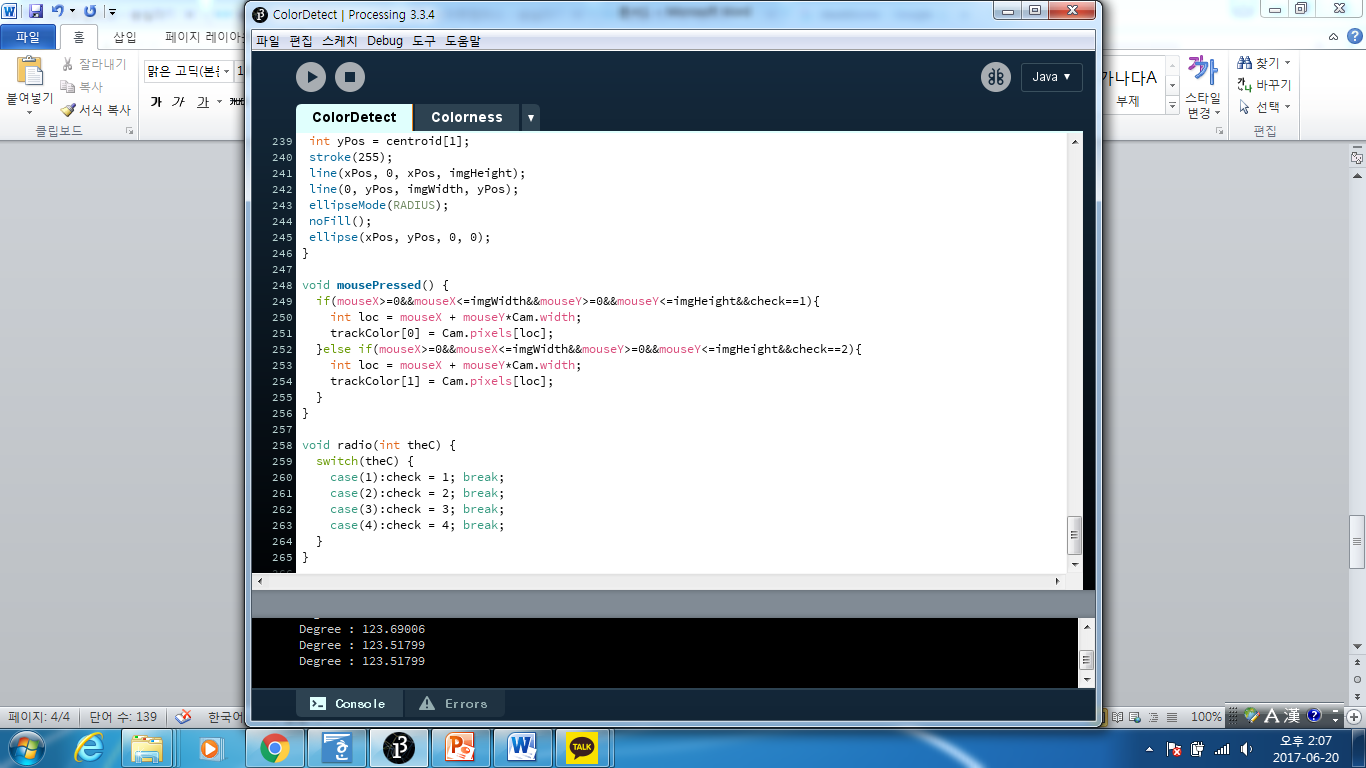
1. Angle()

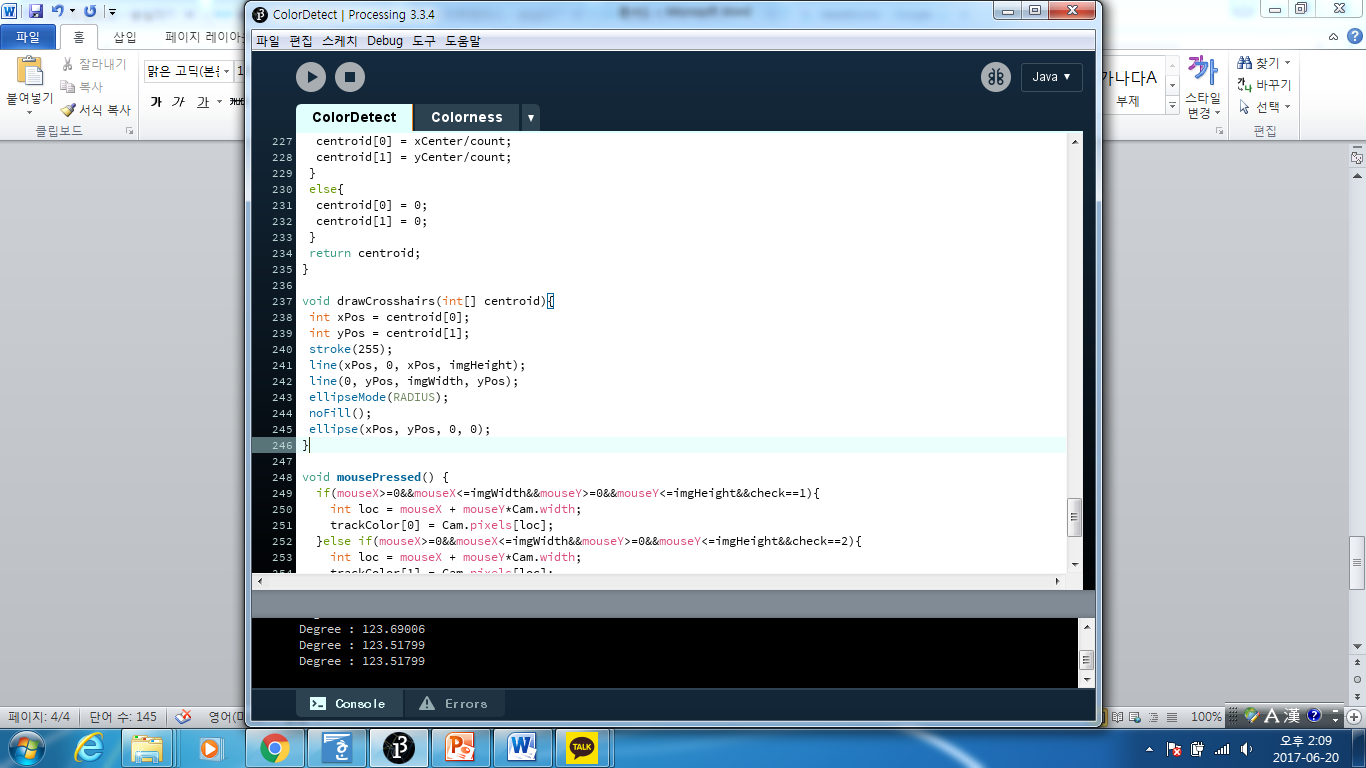
함수는 원하는 각도로 돌게 하는 함수이다. 기본적으로 각도의 범위를 0~360도로 지정해 놓았고, 모터마다 DeadZone 값이 다르므로, 오차범위를 지정하고 속도를 넘겨주게 된다.

1. Position()

Position()함수는 지정된 좌표범위에 해당되면, 지정된 동작을 하게끔 만드는 함수이다. 처음 고정된 로봇의 위치를 세팅하고, 캠 크기의 속성을 가지고 있는 imgWidth, imgHeight을 사용하여 Velocity()함수와 Angle()함수를 조건에 맞게 실행시킨다.

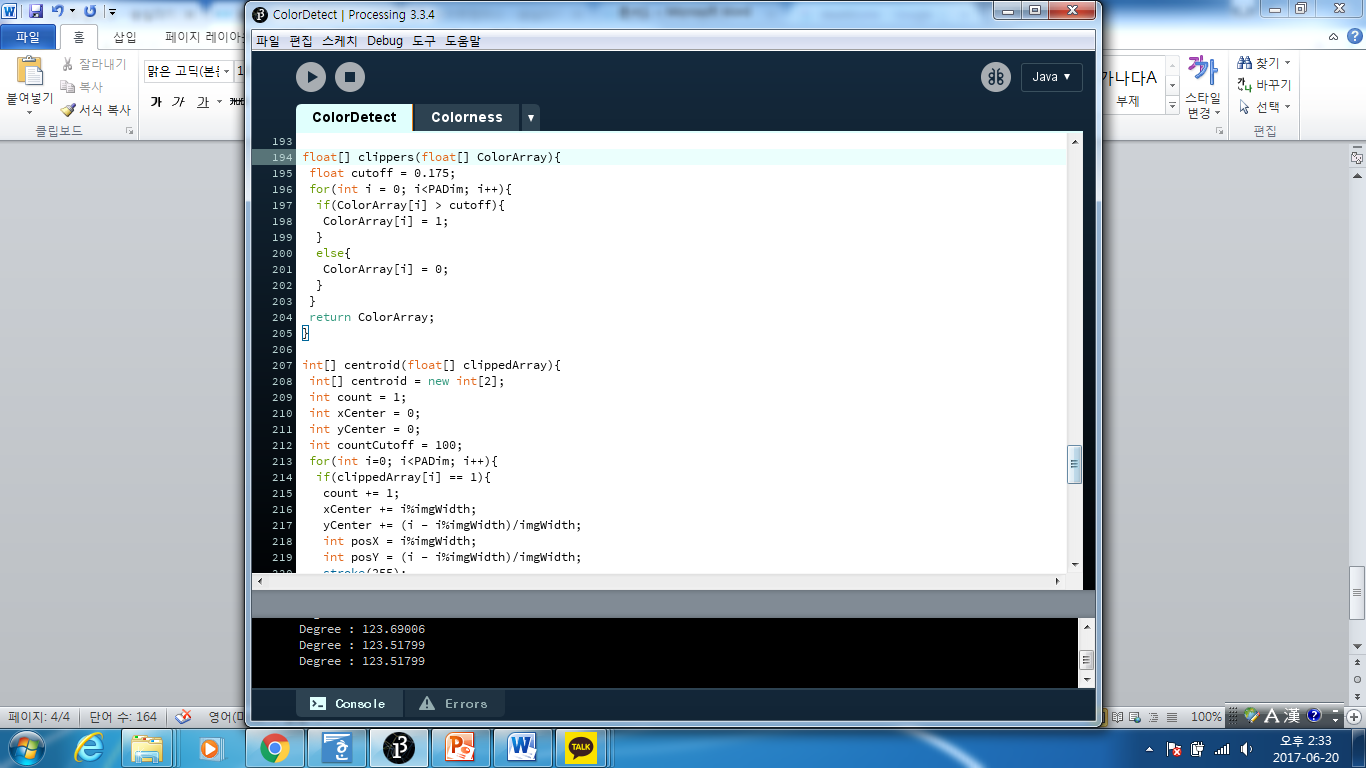
1. Radio()

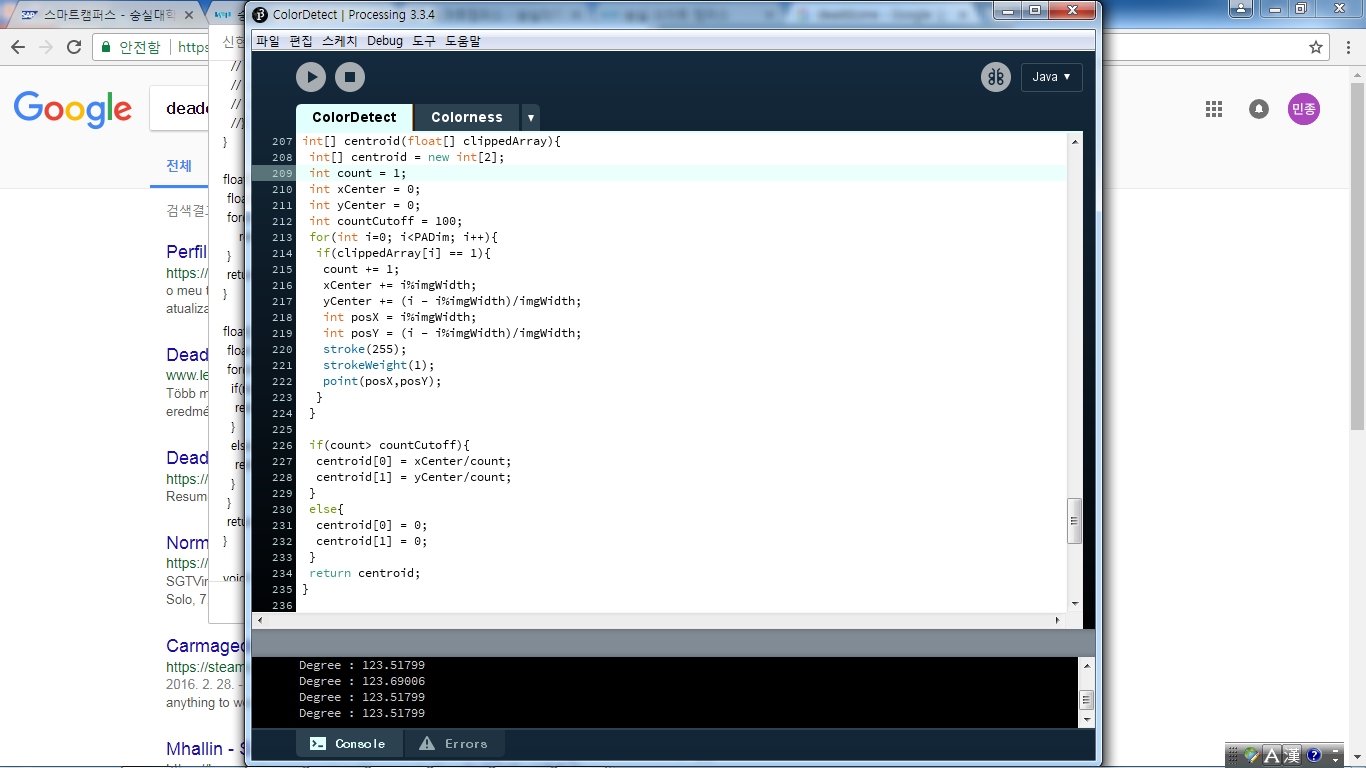
Radio() 함수는 ControlP5를 통해 버튼 형식으로 제어하는 함수이다.

1. drawCrosshairs()

drawCrosshairs()함수는 검출된 centroid[]를 받고, 해당 픽셀좌표에 십자가모양의 라인을 그려주는 함수이다. 프로그램에서는 중점을 잡게 되면, 로봇의 중점을 파악하기 위해 사용하였다.

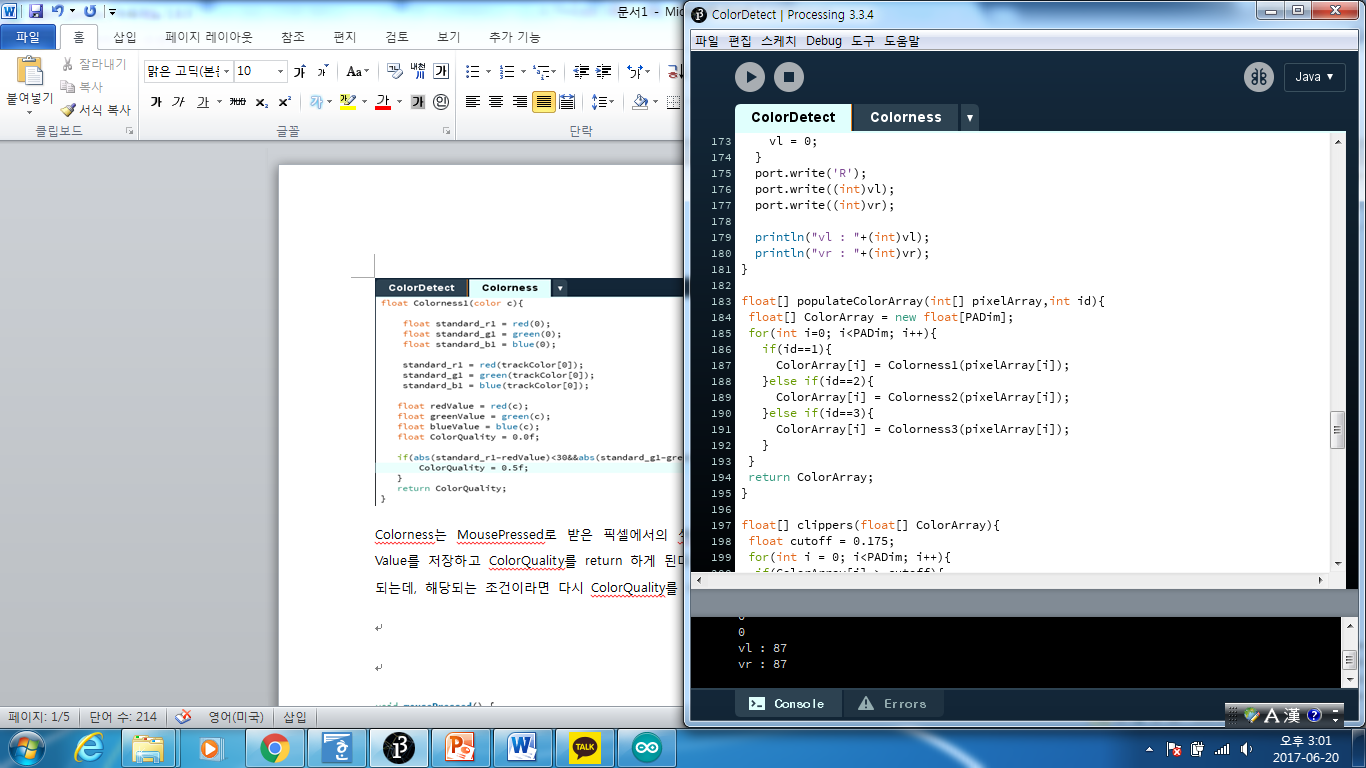
1. Clippers()

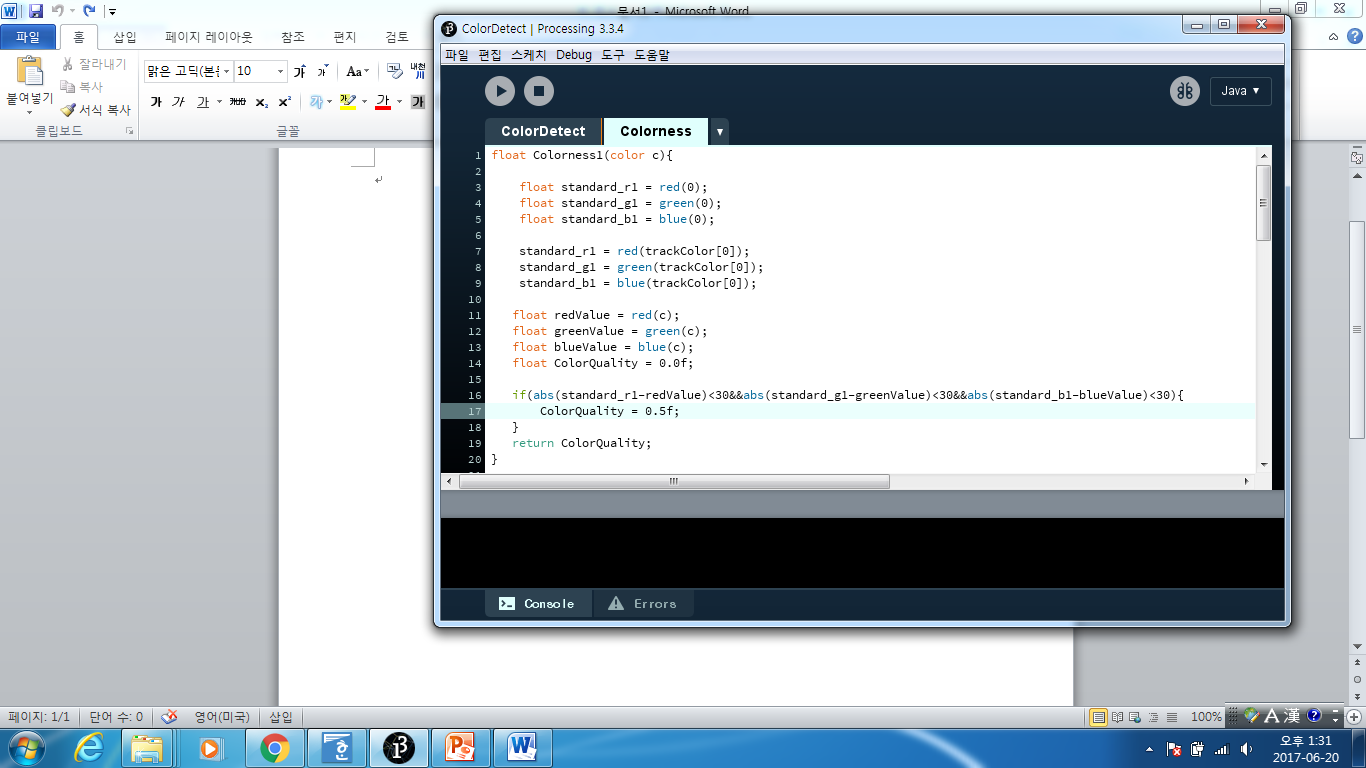
Clippers는 마우스를 클릭했을 때 인지하는 색상을 모두 1로 바꾸고 아닌 것들은 0으로 바꾸어 해당된 색상의 영역을 인지할 수 있도록 만들게 하고 마지막으로 ColorArray값을 return하게 된다.

1. Centroid()

Centroid는 clippedArray[] 픽셀좌표에 해당되는 값에서 1인 값들을 찾고, 그 값에서 중점을 찾고, 영역자체를 흰색으로 표시하게 해준다. 프로그램에서는 자신이 찾은 색상 영역을 자동으로 지정하도록 하기 위해 적용하였다.

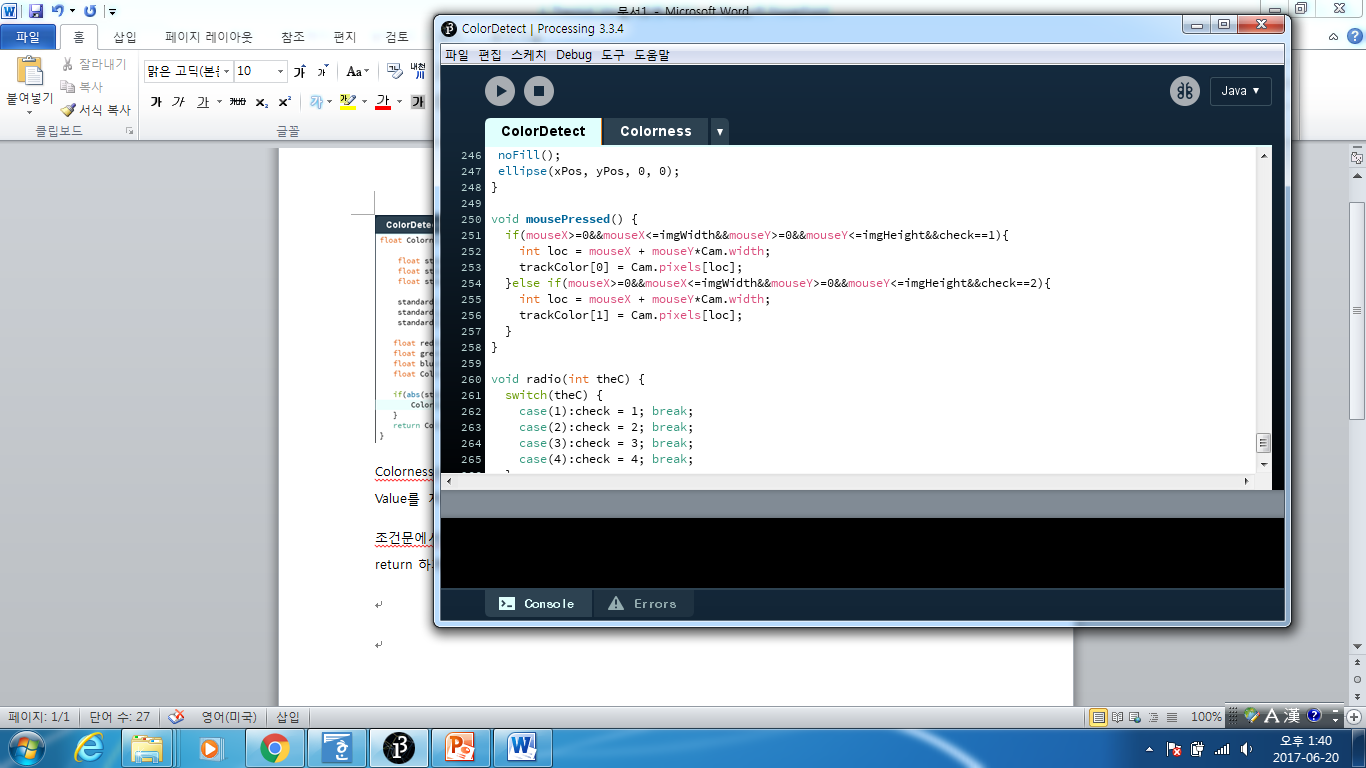
1. PopulateColorArray()

populateColorArray는 id(radio button)에 해당되는 조건을 만족할 경우, 화면의 픽셀영역들을 모두 검사하여 Colorness로 받은 값을 저장하는 부분이다.

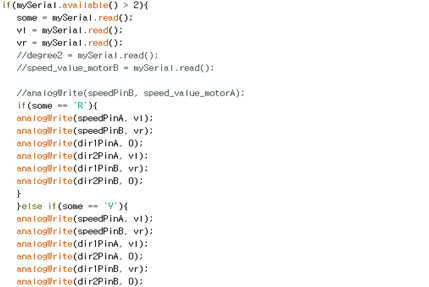
1. Colorness()

Colorness는 MousePressed로 받은 픽셀에서의 색상값을 R,G,B의 속성값과 비교하여, 해당되는 Value를 저장하고 ColorQuality를 return 하게 된다. 조건문에서는 색상의 오차범위를 지정해주게 되는데, 해당되는 조건이라면 다시 ColorQuality를 return 하게 된다.

1. mousePressed()

mousePressed()는 Processing으로 지원하는 함수로서, 마우스를 클릭했을 때 발생되는 이벤트를 제어 할 수 있다. 마우스의 Position이 지정된 캠의 크기안에 해당된다면, 마우스의 위치를 픽셀위치로 저장하게 된다.

1. Arduino



Arduino 블루투스 HC-06의 RX,TX를 설정하고 DC모터를 모터드라이버에 연결하여 전원부와 모터드라이버의 핀의 위치를 세팅한다. 블루투스 통신이 실행되면 R(Rotation), V(Velocity), vl, vr의 값을 Processing으로부터 받아오고 조건에 해당되는 값을 넣어 로봇이 작동할 수 있도록 한다.

1. **실행 시나리오**
   1. Velocity 함수 동작 시나리오(Velocity.mp4)

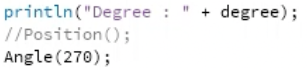
Velocity 함수에 매개변수로 바퀴의 속도값을 넘기면 영상에서와 같이 로봇이 현재 향하고 있는 방향으로 움직이는 것을 확인 할 수 있다.

* 1. Angle 함수 동작 시나리오

1. Angle 함수에 매개변수로 90입력(Angle(90).mp4)

로봇을 90도 회전 시키기 위해 Angle 함수에 매개변수로 ‘90’을 넘겨 주었을 경우, 로봇이 양의방향(반시계 방향)으로 90도 회전하게 된다.

1. Angle 함수에 매개변수로 270입력(Angle(270).mp4)

로봇을 270도 회전 시키기 위해 Angle 함수에 매개변수로 ‘270’을 넘겨 주었을 경우, 로봇이 양의 방향(반시계 방향)으로 270도 회전하게 된다.

* 1. Position 함수 동작 시나리오(Position.mp4)

Position 함수의 전체 내용중 로봇이 첫번째 구간(시작위치->두번째 위치)만 이동하도록 하였다. Position함수를 호출하면 로봇은 시작위치부터 일정거리를 직선으로 이동한 후 정지하게된다.

* 1. 최종 시나리오(Final\_Robot.mp4, Final\_vision.mp4)

비전 프로그램 실행 후 Team Color와 ID Color를 설정하고 Start 버튼을 누르면 로봇이 이동하게 된다.

* + - * 첫번째 구간 : Position1에서부터 Position2로 이동 후 정지, 반시계 방향 90도 회전
      * 두번째 구간 : Position2에서부터 Position3으로 이동 후 정지, 반시계 방향으로 90도 회전(시작 각도로부터 180도 회전한 상태)
      * 세번째 구간 : Position3에서부터 Position4로 이동 후 정지, 반시계 방향으로 90도 회전(시작 각도로부터 270도 회전한 상태)
      * 네번째 구간 : Position4에서부터 Position1(시작위치)으로 이동 후 정지, 반시계 방향으로 90도 회전(시작 각도로부터 360도 회전한 상태)

1. **프로젝트 분석**

- 건전지 문제

- Arduino에서 음수값을 못 받는 문제

- 255가 넘어가면 1부터 다시 시작되는 문제

- 양쪽 모터의 DeadZone이 다른 문제

- 카메라 프레임에 따른 통신 속도 문제