**1. 서 론**

**라즈베리파이를 이용한 게임기**

**김태완, 정재석**

**Tae-Wan Kim, Jae-Seok Jeong**

**요약**

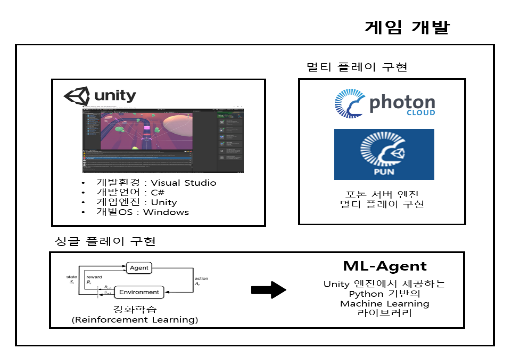
닌텐도 스위치와 같이 여러 가지 센서를 활용하여 게임을 할 수 있도록 아두이노의 여러 가지 센서와 라즈베리파이를 이용한 게임기를 만들게 되었다. 게임 속 컨텐츠로는 ML-Agent를 이용한 강화학습을 통해 만들어진 AI와 겨루는 싱글플레이, 클라우드 형식의 포톤 PUN을 이용한 멀티플레이, 아두이노에서 센서값을 블루투스 모듈을 통해 센서값을 전달하고, 전달된 센서값은 플레이어의 이동 및 추가조작을 하는데 활용된다. 만들어진 게임은 오픈소스로서 Github을 통해, 자유롭게 수정 및 배포가 가능하다.

키보드와 마우스를 이용하는 컴퓨터 게임과 달리 닌텐도 스위치는 자이로센서와 같은 센서들을 이용해서 게임을 하기 때문에, 컴퓨터로 게임을 하는 것보다 활동적이다. 또한 키보드와 마우스 같은 부품을 조이스틱으로 대체하고, 배터리가 따로 구비되어있어, 컴퓨터에 비해 휴대성이라는 장점을 갖고 있다. 그러나 닌텐도 스위치는 게임을 오픈소스 형태로 제공하지 않고 있다.

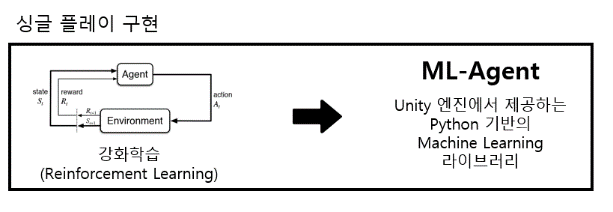
한편, 라즈베리파이는 닌텐도 스위치의 휴대성과 활동성이라는 장점을 가지고, 닌텐도 스위치와 달리 많은 프로그램이 오픈소스 형태로 제공되고 있다.

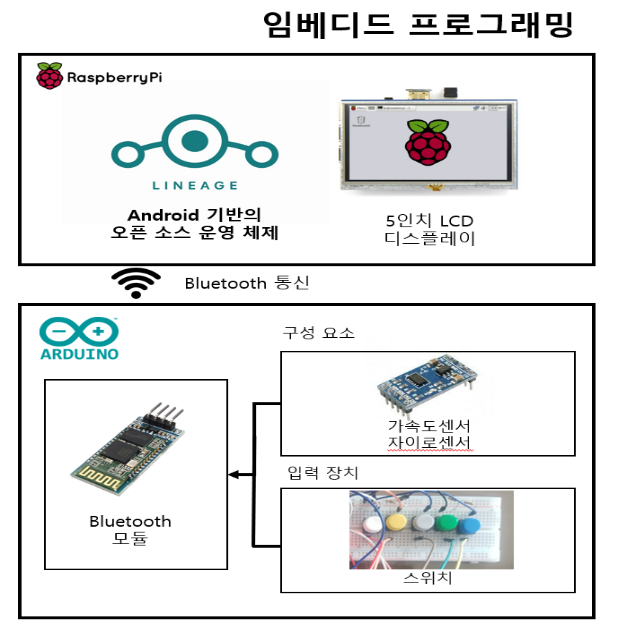
이러한 라즈베리파이의 오픈소스라는 장점을 이용하고, 오픈소스로 제공되는 유니티 엔진을 이용하여, 게임을 제작해 라즈베리파이와 유니티엔진으로 만든 게임을 접목시켜 닌텐도 스위치와 같은 게임기를 만들고자 하였다. 만든 게임은 오픈소스로서, GitHub을 통해 자유롭게 수정 및 배포가 가능하다.

**2. 개발환경 및 시스템 구성**



**<그림 1> 게임 개발 환경**

게임 엔진으로는 유니티엔진을 사용했고, 개발언어는 C#, Visual Studio를 이용했다. 싱글플레이 구현은 유니티에서 제공하는 ML-Agent를 사용했다. ML-Agent는 Python 기반의 Machine Learning 라이브러리로 예를 들어, 벽을 박으면, 보상을 낮게 주어, 그 행동을 기피하는 패턴을 AI에게 학습시켰다. 멀티플레이는 유니티와 호환이 잘 되는 포톤PUN을 이용하여 서버를 구현하여 멀티플레이를 구현하였다.



**<그림 2> 임베디드 프로그래밍 개발 환경**

유니티 엔진으로 만든 게임을 라즈베리파이에서 구동시키기 위해, 라즈베리파이에 Android 기반의 오픈소스 운영체제인 Lineage OS를 탑재하였다. 게임 조작을 위해 아두이노를 사용하였고, 아두이노의 자이로센서, 가속도센서인 MPU-6050과 스위치를 활용하였다. 이렇게 센서를 통해 얻은 센서값을 라즈베리파이로 전달하기 위해 블루투스 모듈인 HC-06을 통해서 라즈베리파이로 센서값을 전달하는 데 활용하였다.

**3. 기능 구현**

**3.1 싱글플레이 구현**

**<그림 3> 싱글플레이를 위한 ML-Agent**

싱글플레이를 위한 AI는 유니티엔진에서 제공하는 Python 기반의 Machine Learning 라이브러리를 사용하였다. ML-Agent는 여러 가지 상황에서 많은 반복을 통해 AI가 행동했을 때, 행동에 대한 보상을 책정하여 가장 큰 보상을 얻을 수 있도록 하는 강화학습이다.

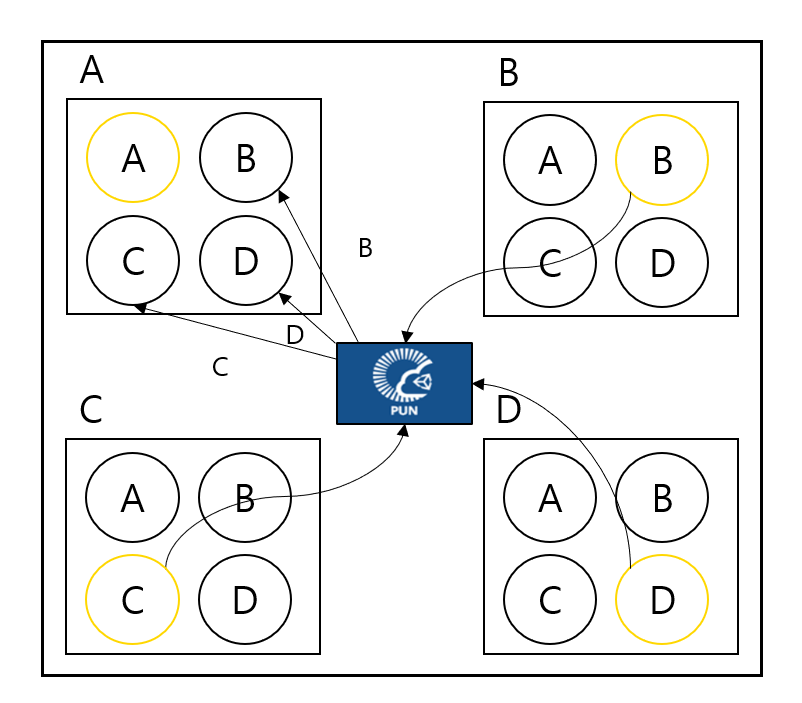
강화학습은 Pytorch를 통해 가상환경에서 강화학습을 한 뒤, 나온 강화학습 모델을 생성한다. 생성된 모델을 AI에게 적용하여 학습된 행동을 하게 한다.



**<그림 4> 강화학습을 위한 체크포인트**

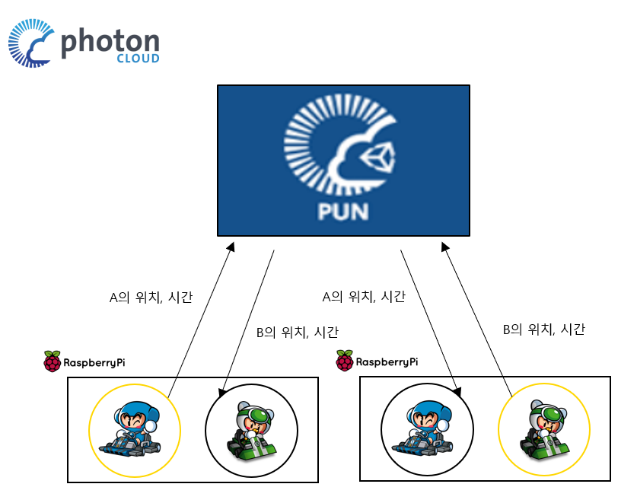
개발된 게임에서는 트랙에 체크포인트를 설정하였다. 체크포인트는 시작 지점에서부터 골인 지점까지 번호를 부여하였고, AI가 번호 순서대로 지날 때마다 reward 함수를 통해서 보상을 얻게 된다. 짧은 시간에 많은 체크포인트를 지날수록, 많은 보상을 얻게 하여 AI가 빠르게 트랙을 돌 수 있도록 Pytorch를 통해 강화학습을 구현하였고, AI에 적용하였다.

**3.2 멀티플레이 구현**



**<그림 5> 포톤서버의 릴레이 방식**

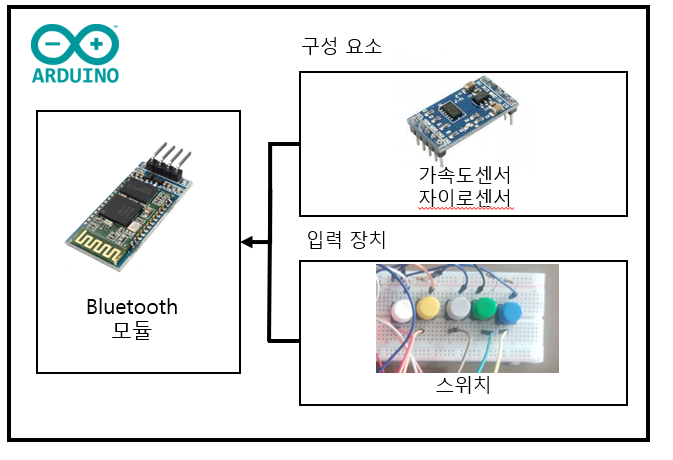
멀티플레이를 위한 서버는 클라우드 형식의 서버인 포톤PUN 서버를 이용하였으며, 포톤PUN은 그림과 같이 각 플레이어의 데이터를 PUN으로 전달을 하게 되면, 이때 PUN이 플레이어의 데이터를 중계하여 서버에 있는 모든 플레이어에게 각 플레이어의 데이터를 전달해서 동기화하는 방식인 릴레이 방식으로 구현하였다. 먼저 멀티플레이를 실행하는 플레이어가 멀티플레이를 위한 방을 만들게 되고, 그 방에 다음 플레이어가 접속을 하게 된다. 접속된 플레이어는 PUN의 릴레이방식으로 데이터를 주고받는다.



<**그림 6> 게임에서의 서버역할**

구현한 게임에서는 플레이어 A, B의 위치와 시간 정보를 PUN으로 전달하고, PUN이 각각의 플레이어에게 전달하게 된다. 이러한 방식으로 서로의 위치와 시간 정보를 공유하여, 멀티플레이를 구현하였다.

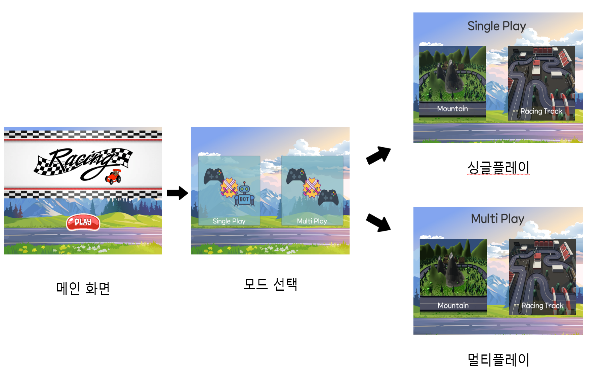
**3.3 조작 구현**



**<그림 7> 블루투스를 이용한 조작 구현**

플레이어 조작에서는 아두이노를 사용하였다. 아두이노에서 플레이어의 움직임에 필요한 센서값을 가속도센서, 자이로센서인 MPU-6050를 통해서 센서값을 가져오고, 플레이어 카트에 필요한 엑셀 기능 및 게임의 추가 조작에 필요한 센서값은 스위치를 통해서 가져오게 된다. 가져온 센서값은 블루투스 모듈인 HC-06을 통해서 라즈베리파이에 전달하게 된다. 전달된 센서값은 유니티에서 지원하는 Unity Android Bluetooth API를 통해서 센서값을 게임에 전달하게 된다. 아두이노에서 BTSerial를 통해 버퍼 형태로 센서값을 전달한다. MPU-6050을 통해서 전달된 센서값은 플레이어의 좌우 움직임에 활용하게 되고, 스위치를 통해서 전달된 센서값은 플레이어의 엑셀, 추가 조작에 활용하게 된다.

**3.4 게임 진행**



**<그림 8> 게임 화면 순서**

게임 화면 순서로는 메인화면, 모드 선택, 그 후 싱글플레이, 멀티플레이로 나뉘게 되며, 맵을 선택할 수 있게 된다. 메인화면에서 플레이 버튼을 누르면, 모드 선택화면으로 넘어가게 되고, 모드 선택화면에서 싱글플레이를 누르면 싱글플레이로, 멀티플레이를 누르면 멀티플레이로 전환된다.

싱글플레이를 시작하게 되면, 플레어어와 AI가 경쟁을 하게 된다. 멀티플레이를 시작하게 되면, 플레이어와 또 다른 플레이어가 경쟁을 하게 된다. 그 후 1등이 마지막 바퀴를 돌고 골인을 하게 되면 게임이 종료되며, 1등의 랩타임과 레이싱 순위가 나타나게 된다. 그 후 메인화면으로 돌아갈 수 있게 된다.



**<그림 9> 레이싱 종료 화면**

**4. 결 론**

기획에 있어서 게임 실행에 있어서, 터치스크친을 통한 터치로 게임 진행을 목표로 하였다, 화면 크기로 인해 라즈베리파이 정품 터치스크린이 아닌 일반 디스플레이로 대체하여 터치가 되지 않았다. 이런 부족한 면도 있지만, 모든 활용한 소스가 오픈소스라는 점에서 만든 프로그램을 Github를 통해서 자유롭게 수정, 배포, 활용이 가능하다는 점이 장점이라고 생각한다.

**[참고문헌]**

[1] Design and Implementation of Reinforcement Learning Environment Using Unity 3D-based ML-Agents Toolkit(2019), 한국정보처리학회 2019년도 춘계학술발표대회, pp.548 – 551

[2] Photon Unity Network: Photon Engine, https://www.photonengine.com/ko-KR/PUN

[3] Unity Manual, https://docs.unity3d.com