

DU-도전학기 결과보고서

과제명	디지털 트윈 기반 자율주행차 보안 기술 개발		
참여자	성명	소속	학번
	김OO	컴퓨터공학과	
	김OO	컴퓨터공학과	
지도교수 의견	<p>상기 학생들은 처음 도전학기를 수행하는 2학년생이지만, 뛰어난 팀워크를 통해 목표한 계획을 모두 달성하는 훌륭한 성과를 이루었습니다. 라즈베리파이를 활용한 디지털트윈 물리세계 구현, Unity AirSim을 활용한 디지털트윈 가상세계 구현, 네트워크 및 암호화 기술 구현을 통한 디지털트윈 통신 기술 개발 등 2학년 학생들에게는 어려운 기술임에도 불구하고, 함께 탐구하고 시행착오를 겪으며 모두 성공적으로 구현해냈습니다. 학생들이 개발한 내용은 지난 5월 개최된 대한임베디드공학회 춘계학술대회에서 논문으로 발표되었으며 국내/외 참가자들에게 많은 관심을 받았습니다. 학생들이 도전학기를 통해 한층 더 성장하는 계기가 되었길 바랍니다.</p> <p>(소속) 컴퓨터공학과 (성명) 김지연 (서명 또는 날인)</p>		

1. 도전 과제 내용

가. Raspberry pi를 활용한 자율주행차 모델 개발

라즈베리파이란 영국의 Raspberry Pi Foundation에서 만든 초소형 컴퓨터이다. 이러한 라즈베리파이는 리눅스 기반 개발 보드에서 공통 소프트웨어 API 및 하드웨어 인터페이스를 제공한다. 본팀은 이러한 라즈베리파이를 활용한 자동차 키트인 라즈봇을 이용하여 연구를 진행하였다. 먼저, VNC viewer를 활용하여 라즈베리파이에 훈련데이터를 수집하는 코드를 작성하였다. 해당 코드는 사용자가 주행을 하면 라즈봇이 0.1초 간격으로 촬영을 진행하고 해당 이미지는 자동차의 움직임 방향을 이름으로 하여 저장된다. 이렇게 획득한 150개 가량의 이미지는 선별하여 고품질의 딥러닝 모델이 생성되도록 하였다. 생성된 딥러닝 모델을 활용하여 해당 모델의 예측값에 따라 자동차 바퀴의 방향이 바뀌게 되는 코드를 작성해 자율주행차를 구현하였다.

나. Unity Airsim 기능을 활용한 디지털 트윈 가상세계 구현

디지털 트윈이란 라이프사이클 전반에 걸쳐 실시간 데이터를 사용하여 업데이트되

고, 시뮬레이션, 머신 러닝, 추론을 통해 의사 결정을 돕는 객체 또는 시스템의 가상 표현을 뜻한다. 쉽게 말해, 물리 세계의 상황을 Unity와 같은 가상세계에 복제하고, 여기서 시뮬레이션이나 머신러닝, 추론 등을 통해 의사 결정을 돕는 기술이다. 본 팀은 이러한 디지털 트윈의 가상세계로 Unity를 채택하였다. Unity의 기능 중의 하나인 Airsim 기능은 자율주행차의 시뮬레이션 테스트를 위해 만들어진 만큼 본 팀의 연구에 적합할 것이라 판단하였다. 해당 Unity Airsim 내에 디지털 트윈의 물리 세계인 자율주행차로부터 자율주행차 현재의 데이터를 받으면 Unity 안의 자동차 Asset이 이에 상응하게 움직이는 코드를 작성하였다. 또한, Unity 내에서 물체를 감지하였을 때 자율주행차로 피드백 데이터를 보내 충돌사고를 방지하는 프로그램을 작성하였다.

다. TCP/IP 통신

TCP는 응용 프로그램이 데이터를 교환할 수 있는 네트워크 대화를 설정하고 유지하는 방법을 정의하는 표준어이다. 이러한 TCP는 IP 네트워크를 통해서 통신하는 호스트에서 실행되는 애플리케이션 간에 신뢰할 수 있고, 순서가 정해져 있어 오류를 확인하고 전송할 수 있다. 본 팀은 이러한 TCP 프로토콜을 활용하여 디지털 트윈의 물리세계인 자율주행차와 가상세계인 Unity의 통신을 구현하였다. 물리 세계와 가상 세계의 통신은 각각 TCP/IP 코드의 Client 코드와 Server 코드로 구현하였다.

라. 암호화

본 팀은 앞서 자율주행차와 Unity의 데이터 통신까지 구현하였다. 이러한 데이터의 송수신 과정에서 공격자로 인한 송수신 되는 데이터를 도청하는 스니핑 공격은 사용자로 하여금 안전사고를 유발할 수 있다. 이에 본 팀은 이를 예방하고자 대칭형 암호화로 알려진 AES 암호화를 해당 연구에 적용하였다.

2. 도전 과제 수행 결과 및 성과

- 팀 공동 과제 수행 결과

본 팀은 도전학기 계획서에 작성한 대로 자율주행차를 구현하였고, 이에 대한 가상세계를 Unity 내에 복제함으로써 성공적으로 구현하였다. 여기서 이들 자율주행차와 Unity 내의 피드백 송수신을 위한 TCP/IP 프로토콜을 자율주행차와 Unity 내에 작성하여 원활하게 통신 되도록 구현하였다. 추가적으로 이들 사이에 송수신되는 패킷은 AES 암호화를 통해 암호화되어 송수신되도록 하여 공격자로부터 스니핑 공격을 당하여 충돌사고를 유발할 위험을 예방하였다. 해당 연구를 통해 자율주행차는 암호화가 된 안전한 데이터를 송수신하면서 성공적으로 디지털 트윈의 가상세계로부터 충돌 상황을 예방하기 위한 피드백 데이터를 성공적으로 수신할 수 있게 하였다.

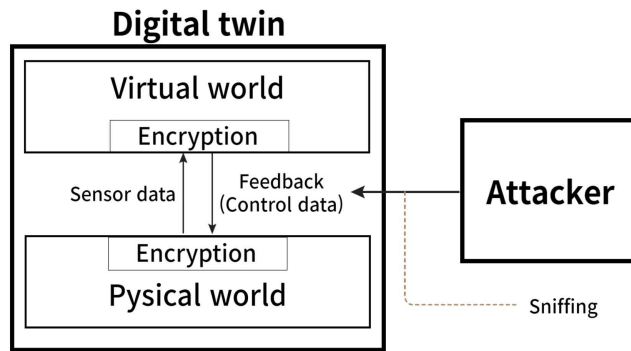


그림 2 연구 구상도

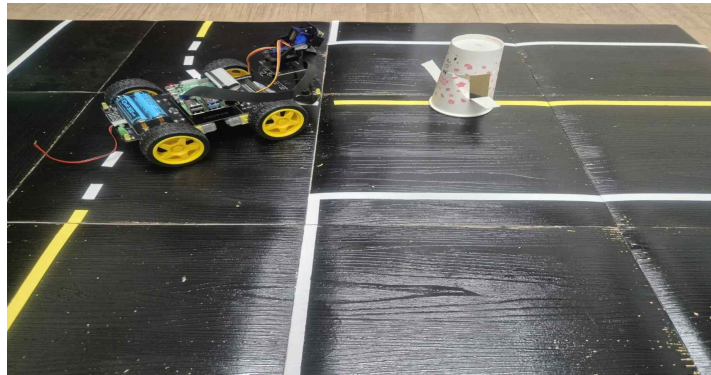


그림 3 디지털 트윈 물리 세계(Raspberry pi)



그림 4 디지털 트윈의 가상세계(Unity Airsim)

3. 자기 평가

- 팀원 개별 과제 수행 결과

● 김OO

처음 도전학기 때 계획했던 것과는 달리 자율주행차 구현을 맡았다. 성공적으로 훈련 데이터 수집 코드를 기반으로 수집된 데이터를 통해 딥러닝을 진행하였고, 해당 모델을 활용하여 자율주행차를 구현하였다. 이후 원활한 통신을 위해 TCP 통신 및 암호화 통신을 구현하는데 노력하였다. 이후 학술대회에서 포스터 발표를 진행하며 맡은 역할을 성공적으로 마무리하였다.

● 김OO

도전학기 때 계획했던 것과 달리 Unity로 디지털트윈 하는 것을 맡았다. Unity Airsim을 활용해 Unity 내 도로를 구현하였고, 이를 기반으로 TCP 통신을 통해 라즈베리파이와 Unity가 서로 통신이 가능하도록 하였다. 이후 TCP 통신을 통해 Unity 내의 자율주행차가 움직일 수 있도록 하였으며 통신 시 위험을 대비해 암호화 통신을 구현하였다. 학술대회 포스터 및 논문 작성에 도움을 주었으며 맡은 역할을 성공적으로 마무리하였다.


4. 최종 결과물

- 개인(팀원) 결과물

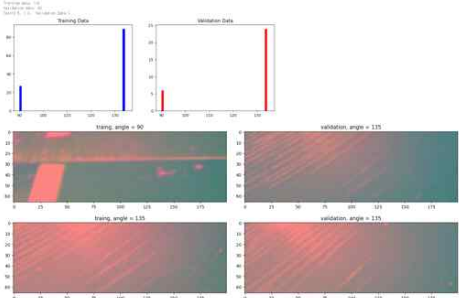
● 김OO

: 라즈베리파이를 활용하여 딥러닝을 통한 자율주행차 구현

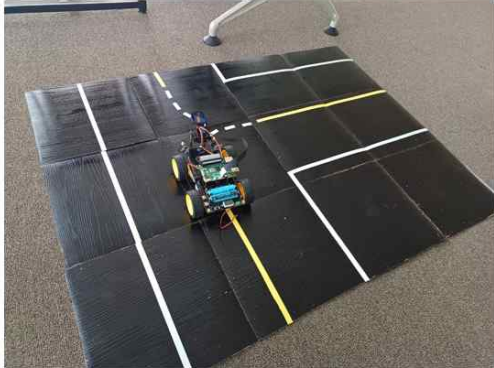
- 자율주행차 구현



1. 수집된 훈련데이터



2. 수집된 훈련데이터를 바탕으로 딥러닝 진행



3. 자율주행 중인 자율주행차

● 김OO

: Unity를 활용하여 디지털 트윈 가상세계 구현 및 충돌사고 발생 시 피드백 송신 코드 구현

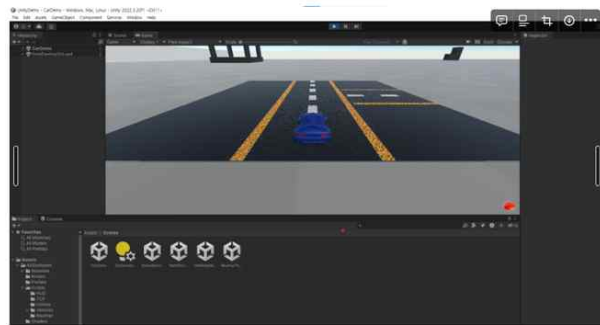
- Unity내 디지털 트윈 가상세계 구현

```
using (connectedTcpClient = tcpListener.AcceptTcpClient())
{
    using (NetworkStream stream = connectedTcpClient.GetStream())
    {
        byte[] buffer = new byte[1024];
        int bytesRead = stream.Read(buffer, 0, buffer.Length);

        string data = Encoding.UTF8.GetString(buffer, 0, bytesRead);

        CarMovement carComponent = GetComponent<CarMovement>();
        if (carComponent != null)
        {
            receivedData = carComponent.carmovedata;
            Debug.Log(receivedData);
        }
        else
        {
            Debug.Log("please");
        }
        Debug.Log("받은 데이터: "+ data);
    }
}
```

1. 자율주행차로부터 받은 데이터 Unity 내에 실시간 업데이트 하는 코드 작성



2. Unity 디지털 트윈 가상세계 예시

- 팀 공통 결과물: TCP/IP 통신 코드, 암호화 코드, 학술대회 논문

```

{
    static void Main(string[] args)
    {
        int port = 37709;
        IPAddress localAddr = IPAddress.Parse("192.168.179.1");

        TcpListener server = new TcpListener(localAddr, port);

        server.Start();

        Console.WriteLine($"Server started on {localAddr}: {port}");

        Console.WriteLine("Waiting for a connection...");

        TcpClient client = server.AcceptTcpClient();

        Console.WriteLine("Connected!");

        NetworkStream stream = client.GetStream();

        while (true)
        {
            //여기서부터 유니티 반환값 코드 작성

            byte[] buffer = new byte[1024];
            int bytesRead = stream.Read(buffer, 0, buffer.Length);

            string data = Encoding.UTF8.GetString(buffer, 0, bytesRead);

            Console.WriteLine($"받은 데이터: {data}");
        }
    }
}

```

그림 7 Unity 내 TCP server 코드

```

def Connect():
    host = "192.168.137.1"
    port = 37709

    global client_socket

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    client_socket.connect((host,port))
    |
    print("Connecting!")

```

그림 8 라즈베리파이 내 TCP client코드

```

def id_class_name(class_id, classes):
    for key, value in classes.items():
        if class_id == key:
            return value

class SimpleEnDecrypt:
    def __init__(self, key=None):
        if key is None:
            key = Fernet.generate_key()

        self.key = key
        self.f = Fernet(self.key)

    def encrypt(self, data, is_out_string=True):
        if isinstance(data, bytes):
            ou = self.f.encrypt(data)
        else:
            ou = self.f.encrypt(data.encode('utf-8'))
        if is_out_string is True:
            return ou.decode('utf-8')
        else:
            return ou

    def decrypt(self, data, is_out_string=True):
        if isinstance(data, bytes):
            ou = self.f.decrypt(data)
        else:
            ou = self.f.decrypt(data.encode('utf-8'))
        if is_out_string is True:
            return ou.decode('utf-8')
        else:
            return ou

```

그림 9 AES 암호화 통신 코드

