DU-도전학기 결과보고서

과제명	도로교통시설 사이버 공격을 탐지하기위한 파이썬 devs 모델 개발		
참여자	성명	소속	학번
	박OO	컴퓨터공학과	
	이OO	컴퓨터공학과	
	김OO	컴퓨터공학과	
	김OO	컴퓨터공학과	
지도교수 의견	상기 학생들은 도전학기 목표를 달성하고자 주 3-4회씩 모여서 실험하였고, 그 결과 의미있는 결과가 도출되어 학술대회 논문으로도 발표하였습니다. 학부생들이 이해하기 어려운 이산사건시스템 명세이론 DEVS 형식론도 공부하여 자율주행차 보안을 위한 모델링을 했을뿐 아니라, DEVS Suite을 이용하여 시뮬레이션 모델을 구현하는 등 학술적으로 매우 우수한 결과를 도출하였습니다. 참가 학생들의 최고의 팀워크와 열정으로 이루어낸 성과라고 생각하며 실험환경 구축 비용, 학술대회 참가비 및 교통비 등을 지원해주시고, 좋은 성과 낼 수 있도록 기회를 제공해주신 대학혁신지원사업단에도 감사드립니다. (소속) 컴퓨터공학전공 (성명) (서명 또는 날인)		

1. 도전 과제 내용

주요 기반 시설에 연결되는 사물인터넷 기기가 증가하면서 IoT 취약점을 악용한 사이버공격이 증가하는 추세이다. 주요 기반 시설은 교통 시설, 에너지 시설, 농업 시설 등 국가 운영에 필수적인 시설들을 의미하며, 사이버 공격에 노출될 경우 사회적인 혼란이나 재산 피해를 유발할 수 있다. 따라서, 사전에 주요 기반 시설에 대한 사이버 공격 시나리오를 개발하여 이를 방어하기 위한 대응 기술을 마련해야 한다. 실제로 2016년~2020년도까지 핵심 정보통신기반시설에 대한 사이버 공격 시도 건수가 총 1만 건을 넘겼으며, 매년 사이버 공격 건수가 증가하고 있는 추세이다. 다양한 IoT 기기 중, 최근 증가하고 있는 자율주행차에 사이버 공격이 발생할 경우, 다수의 인명 피해 및 대규모 재산 피해가 발생할 수 있기 때문에 사전에 다양한 사이버 공격을 예측하고 대응하기 위한 보안 기술을 개발해야 한다. 본 도전과제에서는 물리 공간의 자율주행차와 가상공간의 자율주행차 시뮬레이션 모델이 상호작용하며 사이버 공격을 탐지 및 예측하기 위한 디지털 트윈 모델 구현을 도전해보려 한다. 해당 모델은 실시간 데이터를 활용한 디지털 트윈 환경이기 때문에사이버 공격 시뮬레이션을 통해 현실에서의 여러 사이버 공격에 대한 대응책이나 피해 범위도 예상해 볼 수 있다.

2. 도전 과제 수행 결과 및 성과

이번 도전 학기의 주제는 물리 공간의 자율주행차와 가상공간의 자율주행차 시뮬레이션 모델이 상호작용하며 사이버 공격을 탐지 및 예측하기 위한 디지털 트윈 모델의 구현이다. 해당 주제를 진행하기 위해서 <표 1>과 같이 6가지의 분야로 나누어 진행하였다.

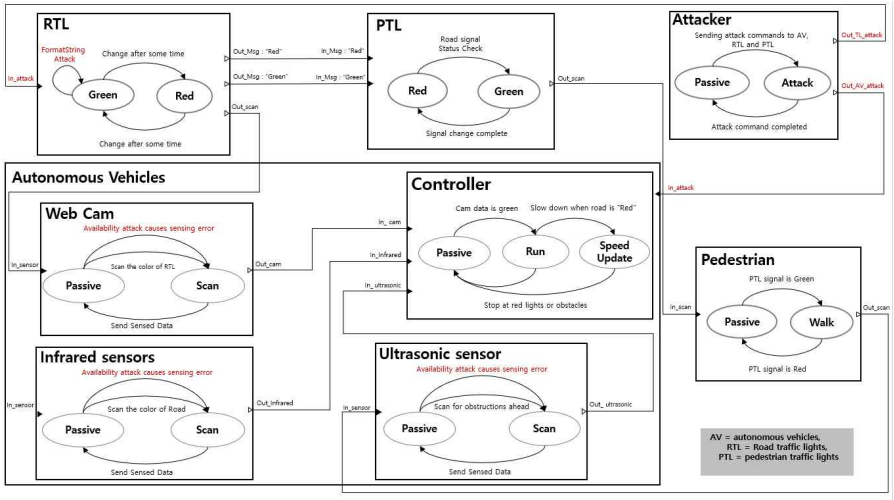
순서	진행 내용
1	DEVS 모델링 그림 및 명세표 제작
2	자율주행 자동차 제작 및 동작을 위한 코딩
3	opecv와 tflite함수를 사용한 객체인식 코드 구현
4	DEVS-Suite Simulator에서 작동하는 Devs 모델 코딩
5	AWS 서비스들를 사용해 센싱 데이터 전달하도록 코딩
6	센싱 데이터와 DEVS 모델이 연동되도록 코딩

<표 1> 순서별 진행 내용

1) DEVS 모델링 그림 및 명세표 제작

원자모델	명세
도로신호	X = {!지정시간 동안 신호 Green 변경} Y = {!지정시간 후 신호 Red 변경} S = {phase|phase∈Red, Green} δ_{ext} : {((Red), !지정시간 동안 신호 Green 변경) = Green} ta : {Red} = 2minutes, : {Green} = 2minutes
보행자신호	X = {!도로 신호가 Red 일 때 Green 변경} Y = {!도로 신호가 Green 일시 Red 변경} S = {phase|phase∈Red, Green} δ_{ext} : {((Red), !도로 신호가 Red 일 때 Green 변경) = Green}
초음파	X = {!전방의 장애물 스캔} Y = {!센싱된 데이터 송신} S = {phase|phase∈Passive, Scan} δ_{ext} : {((Passive), !전방의 장애물 스캔) = Scan}
웹캠	X = {!도로 신호등 스캔} Y = {!센싱된 데이터 송신} S = {phase|phase∈Passive, Scan} δ_{ext} : {((Passive), !도로 신호등 스캔) = Scan}
적외선	X = {!도로 색상 스캔} Y = {!센싱된 데이터 송신} S = {phase|phase∈Passive, Scan} δ_{ext} : {((Passive), !도로 색상 스캔) = Scan}
제어	X = {!도로 신호 Green 및 정상상태 일시 주행, !도로 신호 Red 또는 장애물 감지 시 감속} Y = {!도로 신호 Red, 장애물 근접 접근 또는 비정상 상태 시 정지} S = {phase|phase∈Stop, Run, Speed Update} δ_{ext} : : {((Stop), !도로 신호 Green 및 정상상태 일시 주행) = Run} δ_{ext} : : {((Run), !도로 신호 Red 또는 장애물 감지 시 감속) = Speed Update}
보행자	X = {!보행자 신호가 Green 일 때 진행} Y = {!보행자 신호가 Red 일시 정지} S = {phase|phase∈Stop, Walk} δ_{ext} : {((Stop), !보행자 신호가 Green 일 때 진행) = Walk}
도로	S = {phase|phase∈Passive}

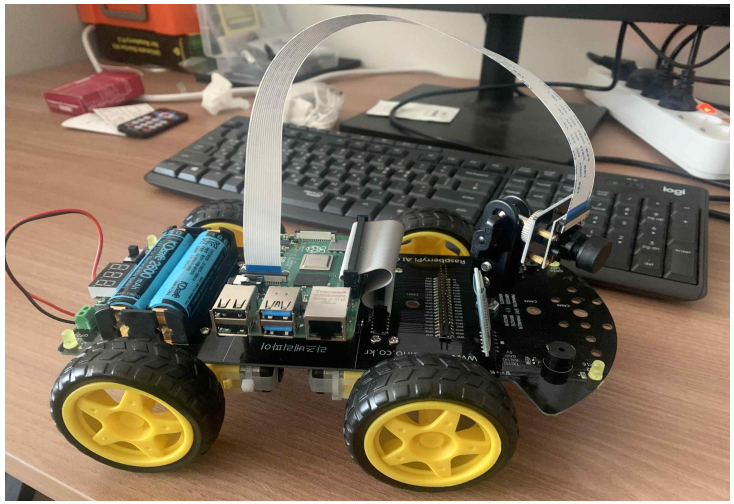
<그림 1> Devs 모델링에 대한 명세표



<그림 2> DEVS 기반 모델링

- 먼저 <그림 1>과 <그림 2>와 같이 DEVS 모델링과 모델링에 대한 상태변화를 기술한 명세표를 제작하였다.

2) 자율주행 자동차 제작 및 동작을 위한 코딩



<그림 3> 제작한 자율 주행 자동차

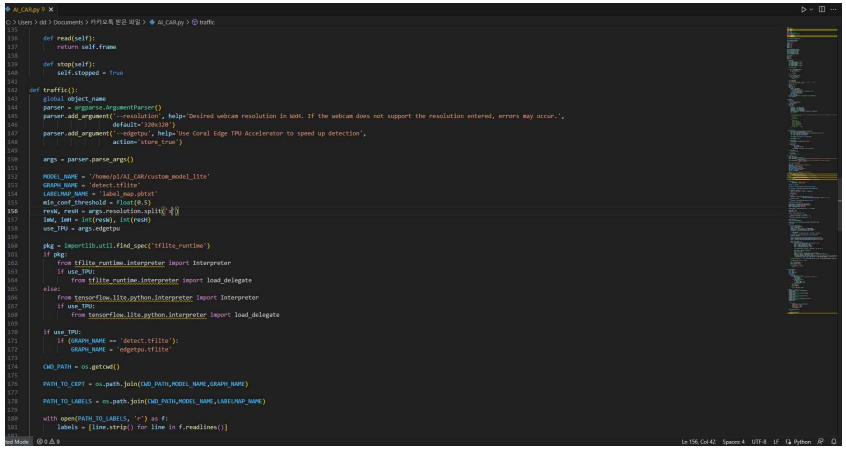

```
Right1 = 11
Right2 = 7

GPIO.setup(EchoPin,GPIO.IN)
GPIO.setup(TrigPin,GPIO.OUT)
GPIO.setup(Left1,GPIO.IN)
GPIO.setup(Left2,GPIO.IN)
GPIO.setup(Right1,GPIO.IN)
GPIO.setup(Right2,GPIO.IN)

object_name = ""
dist = 20
track = 4
speed = 50

def Distance():
    global dist
    GPIO.output(TrigPin,GPIO.LOW)
    time.sleep(0.000002)
    GPIO.output(TrigPin,GPIO.HIGH)
    time.sleep(0.000015)
    GPIO.output(TrigPin,GPIO.LOW)

    t3 = time.time()

    while not GPIO.input(EchoPin):
        t4 = time.time()
        if (t4 - t3) > 0.03 :
            return -1

    t1 = time.time()

    while GPIO.input(EchoPin):
        t5 = time.time()
        if(t5 - t1) > 0.03 :
            return -1

    t2 = time.time()
    time.sleep(0.02)
    #print ("distance_1 is %d " % (((t2 - t1)* 340 / 2) * 100))
    return ((t2 - t1)* 340 / 2) * 100

def Distance_test():
    num = 0
    ultrasonic = []
    while num < 5:
        distance = Distance()
```

<그림 4> 자율 주행차의 기본 동작을 위한 코드

- <그림 3>과 같이 도전과제에서 사용될 자율 주행 자동차를 제작하였고 <그림 4>와 같이 자동차의 기본 동작을 위한 코딩을 진행함

3) opecv와 tflite함수를 사용한 객체인식 코드 구현

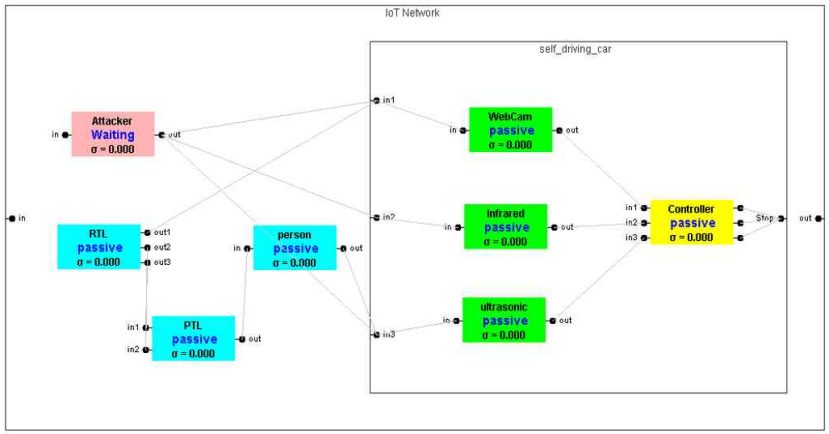
- 학습한 인공지능 모델과 opencv DNN 모듈의 호환성 문제로 인한 opecv DNN모듈을 사용하지 않고 <그림 5>와 같이 tflite함수를 사용하여 객체인식 코드 구현



<그림 5> 객체인식을 위한 코드

- 학습한 인공지능 모델과 opencv DNN 모듈의 호환성 문제로 인한 opecv DNN모듈을 사용하지 않고 tflite함수를 사용한 객체 인식 코드 구현

4) DEVS-Suite Simulator에서 작동하는 Devs 모델 코딩



<그림 6> Devs 모델의 기본 틀

- <그림 6>와 같이 기본적인 원자모델의 틀을 코딩하였고 원자모델간의 Coupling 및 기본적인 데이터 이동을 확인함

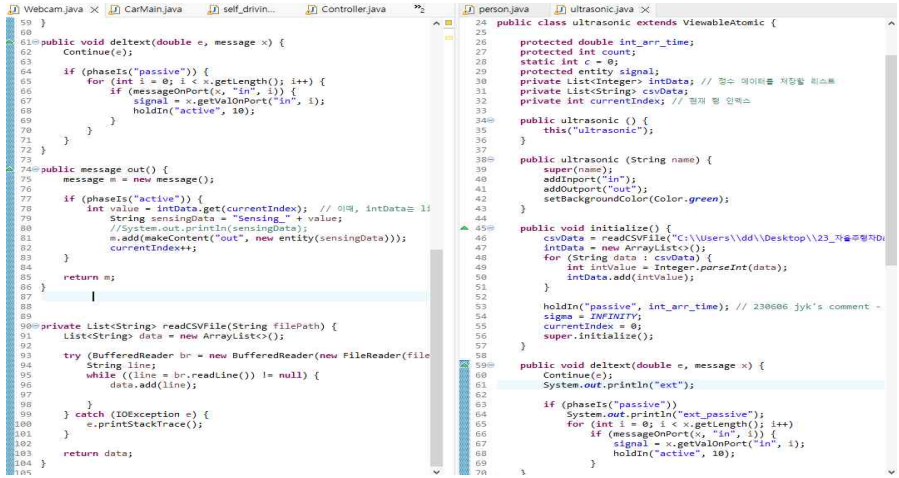
5) AWS 서비스들를 사용해 센싱 데이터 전달하도록 코딩


```
ENDPOINT = "a1lhva7nvsyxuf-ats.iot.us-east-1.amazonaws.com"
CLIENT_ID = "AI_Car"
PATH_TO_CERT = "/home/pi/certs/certificate.pem.crt"
PATH_TO_KEY = "/home/pi/certs/private.pem.key"
PATH_TO_ROOT = "/home/pi/certs/Amazon-root-CA-1.pem"
MESSAGE = "Hello World"
TOPIC = "AI_Car"
RANGE = True

event_loop_group = io.EventLoopGroup(1)
host_resolver = io.DefaultHostResolver(event_loop_group)
client_bootstrap = io.ClientBootstrap(event_loop_group, host_resolver)
mqtt_connection = mqtt_connection_builder.mtls_from_path(
            endpoint=ENDPOINT,
            cert_filepath=PATH_TO_CERT,
            pri_key_filepath=PATH_TO_KEY,
            client_bootstrap=client_bootstrap,
            ca_filepath=PATH_TO_ROOT,
            client_id=CLIENT_ID,
            clean_session=False,
            keep_alive_secs=6
            )
print("Connecting to {} with client ID '{}'...".format(
        ENDPOINT, CLIENT_ID))
connect_future = mqtt_connection.connect()
connect_future.result()
print("Connected!")
print('Begin Publish')
```

<그림 7> AWS 서비스를 통해 센싱 데이터 전달

- MQTT 브로커로써 AWS IoT Core를 사용하고 센싱 데이터를 전달할 때 이용할 저장소로써 AWS S3를 이용함

6) 센싱 데이터와 DEVS 모델이 연동되도록 코딩



<그림 8> 센싱 데이터와 DEVS모델 연동

- 자율 주행 자동차의 센싱 데이터들이 CSV파일 형식으로 저장되면 <그림 8>과 같이 해당 CSV파일의 인덱스 값을 읽어와 자율 주행 자동차와 DEVS모델이 연동 될 수 있도록 하여 디지털 트윈 모델을 구현함

3. 자기 평가

박00 : 지난 도전 학기는 주요 기반 시설의 IoT 취약점 분석 및 보안 솔루션 개발을 주제로 잡고 진행하였다. 하지만 자율 주행 자동차의 객체 인식 기능을, 인공지능을 사용하여 구현하지 못하고 YOLOv5와 파이토치를 이용하여 제작했다. 이와 다르게 이번 학기에서는 mobilenet ssd와 flite 함수를 사용해서 이전에 하지 못했던 인공지능을 이용한 객체 인식을 진행하였고 이번 도전의 핵심이었던 자율 주행 자동창의 센싱 데이터를 실시간으로 연동하는 DEVS모델을 제작하였다. 이번에 처음으로 Devs 모델 코딩을 접해서 DEVS 개념과 DEVS Suite Simulator의 사용법, 해당 시뮬레이터에서 작동할 수 있도록 하는 코딩 등이 을 처음 접하게 되었는데 Devs에서 사용되는 개념을 이해하고 응용하여 모델의 틀을 만들고 다소 어려움이 존재했지만, 팀원들과 문제에 대해 논의하거나 지도교수님에게 조언을 구하여 해결할 수 있었다. 또한, 팀원들과의 협력과 교수님의 조언을 통해서 DEVS Suite Simulator를 사용하여 모델을 확인할 때 동작(Step) 부분을 누르면 외부 상태, 내부상태 변화가 동시에 수행되어 짝수 인덱스만을 읽어 들이는 문제를 해결할 수 있었다. 도전 학기를 진행하면서 팀원과의 협업과 소통의 중요성을 느낄 수 있었다.

이00 : 지난 도전 학기의 주제를 이어 두 번째 도전 학기를 진행하면서 구축하지 못하였던 인공지능을 이용한 객체 인식 및 차선인식을 구현하는 데 중점을 두었으며 또한 Devs의 모델링 및 자율주행 코드의 최적화를 진행하였다. 지난 도전 학기에서는 인공지능 학습에 YOLOv5와 파이토치를 이용하여 호환성 문제로 실패하였지만 지난 실패를 발판삼아 이번 도전 학기를 위해 인공지능에 관한 공부를 진행하였고 opencv dnn 모듈에서 작동 가능하며 라즈베리파이 같은 IoT기기에 특화된 mobilenet ssd를 이용한 인공지능 학습을 목표로 하였었다. 이를 위해 사진 촬영 및 데이터셋 라벨링을 진행하였고 그 후 코랩에서 tensorflow와 mobilenet ssd를 이용한 데이터셋 학습을 진행하고 싶었으나 현재 코랩에서 tensorflow 1.x버전을 지원하지 않고 opencv dnn은 tensorflow 2.x 버전에서 학습한 고정 그래프를 인식하지 못하는 문제가 생겼었다. 이를 해결하고자 많은 시도를 하였지만, 실패하였고 해결방안으로 opencv dnn 모듈을 사용하지 않고 tensowflow 2.x에서 모델을 학습시킨 후 tflite형식의 모델로 변환 후 코드를 재구현하였다. 이후 자율주행 자동차의 기능들을 충돌 없이 유연하게 작동시키기 위해 코드 최적화에 노력을 들였다. 도전 학기를 진행하면서 인공지능에 대한 많은 공부를 하였고 버전 호환성의 중요함을 느꼈으며 저번 도전 학기의 실패를 이번 도전 학기에서 만회할 수 있었던 점에서 뜻깊은 활동이었다.

김00 : 도전 학기를 진행하면서 도로 교통시설에 어떤 사이버 공격이 시도될 수 있는지에 대해 조사하였고, 이를 통해 공격 탐지를 위한 Devs 모델 제작에 도움을 주었다. 이번 도전 학기를 수행하기 위해 Devs 모델링을 위한 개념과 사용법을 처음 접하게 되었는데 Devs에서 사용되는 개념을 이해하고 응용하여 모델의 틀을 만드는데 다소 어려움이 존재했지만 팀원들과 문제에 대해 논의하며 해결할 수 있었다. 또 다른 문제로는 센싱 데이터를 Devs 모델과 연동하는 과정에서 데이터 출력 문제가 발생했지만 팀원들과 교수님께 질문을 하며 해결할 수 있었다. 이번 과제를 진행하면서 어려웠던 부분이나 막혔던 부분들에 대해 팀원들과 함께 해결해나가며 혼자가 아닌 팀원과의 협업

이 얼마나 중요한지를 느낄 수 있었고, 스스로 맡은 부분을 수행해 나가며 성취감을 느낄 수 있었다.

김00 : 도전 학기 초반에는 자율 주행 자동차 공격 조사, 객체 인식에 사용 될 인공지능 조사 및 방법 탐색, 실시간 통신을 위한 브로커 조사를 진행을 완료하여 개인 목표를 완료하였다. 추가로 팀원과 함께 AWS를 이용한 실시간 데이터 연동을 진행하였고 중간 보고서 이후에는 적외선 센서가 차선 인식을 할 수 있도록 자율 주행 자동차 코드를 작성하였다. 도전 학기를 진행하면서 많은 문제와 어려움이 있었는데 크게 고르자면 인공지능 모델을 학습하기 위한 데이터 라벨링 및 코드를 구현하는 것이었다. 특히, openCV 를 활용한 작업은 처음이었기 때문에 새로운 도전과제였다. 어려움을 극복하기 위해 먼저 필요한 데이터를 수집하고, 이를 정확하게 라벨링하여 학습 데이터셋을 구축했습니다. 코드를 구현하기 위해 openCV 의 문서와 예제 코드를 참고하며 학습 모델을 구성했다. 결과적으로, 지속적인 노력과 열정을 통해 인공지능 모델을 학습하기 위한 데이터 라벨링과 코드 구현을 성공적으로 완료할 수 있었다. 이를 통해 openCV 를 활용하여 객체인식에 사용될 모델을 학습 시킬 수 있었다.

4. 최종 결과물

도전 학기 활동을 수행 후에 제출할 수 있는 결과물로는 팀 공통 결과물이 있다. 팀 공통 결과물은 도로교통시설 사이버 공격을 탐지하기위한 JAVA Devs 모델, 학술대회 논문이 있다.

※팀의 경우, 팀 공통 결과물과 개인(팀원별) 결과물을 구분하여 기술
※참고문헌이 있을 시 정확히 명시