한국직업능력교육원 안산

자율주행 자동차

- TEAM 3조
- 강현수, 김우영

K-DIGITAL TRAINING

목 차

프로젝트 주제 및 기획의도

- 라즈베리파이를 이용한 차선인식 자율주행 자동차 제작
- 지금까지 수업을 받으며 배운 것을 토대로 자율주행 자동차라는 주제의 프로젝트를 진행하고자 함

프로젝트 개요

- 4개의 모터로 연결된 RC카 바퀴 제어
- 카메라에서 사진 파일을 받아 OPEN CV로
  파일 전 처리 후 데이터 검출
- 검출된 차선 데이터를 수집 및 학습
- 학습한 데이터로 이동 가능하도록 바퀴 모터 제어

주차별 프로젝트 수행단계

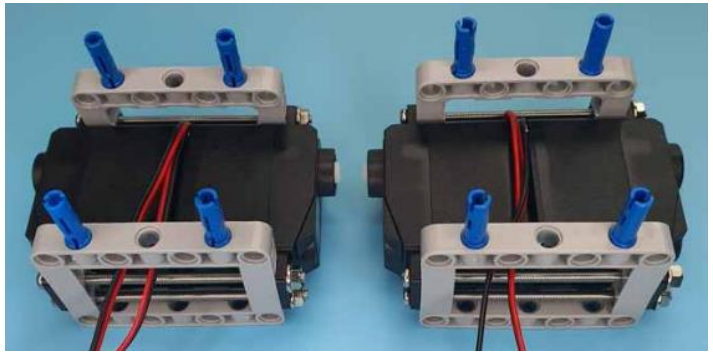
week 01	week 02	week 03	week 04	week 05
• 동키카 • 자동차 조립 • 차선 만들기	• 코드 작성	• 테스트로 데이터 수집 • 수집한 데이터로 코드 수정	• 수정 단계 및 자율주행 안정화 • 자율주행 자동차 완성	• 프레젠테이션 & 결과보고서 작성 • 발표

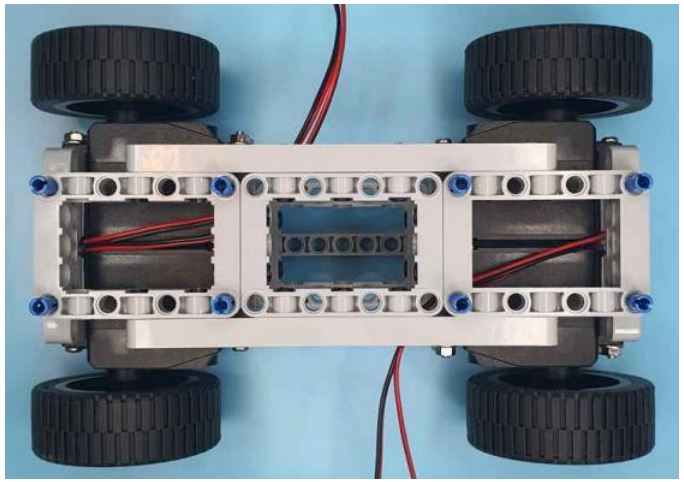
WEEK 01

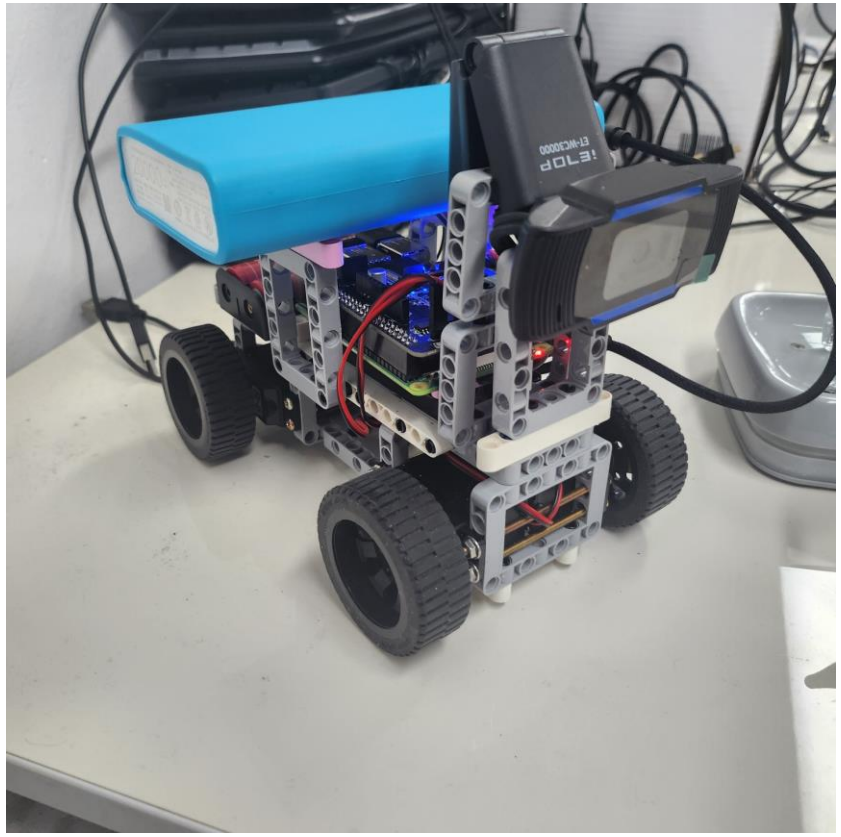
동키카 시뮬레이터를 사용하여 인공지능 학습 전체 과정을 살펴봄

`

WEEK 01







WEEK 01

자동차 조립 및 차선 만들기

WEEK 02

```
CCW_L = [12,20,18,21]
CCW_R = [13,22,19,23]
wheels = []
MOT_PREQ = 100

GPIO.setup(CCW_L, GPIO.OUT)
GPIO.setup(CCW_R, GPIO.OUT)
GPIO.output(CCW_L, False)
GPIO.output(CCW_R, False)

wheel_L = GPIO.PWM(CCW_L[1], MOT_PREQ)
wheel_L.start(0.0)
wheels.append(wheel_L)
wheel_L_B = GPIO.PWM(CCW_L[3], MOT_PREQ)
wheel_L_B.start(0.0)
wheels.append(wheel_L_B)

wheel_R = GPIO.PWM(CCW_R[1], MOT_PREQ)
wheel_R.start(0.0)
wheels.append(wheel_R)
wheel_R_B = GPIO.PWM(CCW_R[3], MOT_PREQ)
wheel_R_B.start(0.0)
wheels.append(wheel_R_B)
```

```
def wheel_go():
    GPIO.output(CCW_L[0], True)
    GPIO.output(CCW_L[2], True)
    wheels[0].ChangeDutyCycle(60)
    wheels[1].ChangeDutyCycle(60)

    GPIO.output(CCW_R[0], True)
    GPIO.output(CCW_R[2], True)
    wheels[2].ChangeDutyCycle(70)
    wheels[3].ChangeDutyCycle(70)

def wheel_back():
    GPIO.output(CCW_L[0], False)
    GPIO.output(CCW_L[2], False)
    wheels[0].ChangeDutyCycle(20)
    wheels[1].ChangeDutyCycle(20)

    GPIO.output(CCW_R[0], False)
    GPIO.output(CCW_R[2], False)
    wheels[2].ChangeDutyCycle(15)
    wheels[3].ChangeDutyCycle(15)
```

```
def wheel_right():
    GPIO.output(CCW_L[0], True)
    GPIO.output(CCW_L[2], True)
    wheels[0].ChangeDutyCycle(15)
    wheels[1].ChangeDutyCycle(15)

    GPIO.output(CCW_R[0], True)
    GPIO.output(CCW_R[2], True)
    wheels[2].ChangeDutyCycle(95)
    wheels[3].ChangeDutyCycle(95)

def wheel_left():
    GPIO.output(CCW_L[0], True)
    GPIO.output(CCW_L[2], True)
    wheels[0].ChangeDutyCycle(95)
    wheels[1].ChangeDutyCycle(95)

    GPIO.output(CCW_R[0], True)
    GPIO.output(CCW_R[2], True)
    wheels[2].ChangeDutyCycle(15)
    wheels[3].ChangeDutyCycle(15)

def wheel_stop():
    GPIO.output(CCW_L[0], False)
    GPIO.output(CCW_L[2], False)
    wheels[0].ChangeDutyCycle(0)
    wheels[1].ChangeDutyCycle(0)

    GPIO.output(CCW_R[0], False)
    GPIO.output(CCW_R[2], False)
    wheels[2].ChangeDutyCycle(0)
    wheels[3].ChangeDutyCycle(0)
```

모터 제어 코드

좌,우 이동은 안쪽과 바깥쪽 출력을 다르게 주어 방향 제어

WEEK 02

```python
def main():
    camera = cv.VideoCapture(0)
    camera.set(cv.CAP_PROP_FRAME_WIDTH, 400)
    camera.set(cv.CAP_PROP_FRAME_HEIGHT, 300)
    path = "/home/pi/Study/project/camera/data4/data4"
    i = 0
    carState = "stop"

    while (camera.isOpened()):
        # delay = int(1000/camera.get(cv.CAP_PROP_FPS))
        keyValue = cv.waitKey(1)

        if keyValue & 0xFF == 27:
            print('finish')
            break
```

```python
    elif keyValue == 119:
        print("go")
        carState = "go"
        wheel_go()

    elif keyValue == 115:
        print("back")
        carState = "back"
        wheel_back()

    elif keyValue == 100:
        print("right")
        carState = "right"
        wheel_right()

    elif keyValue == 97:
        print("left")
        carState = "left"
        wheel_left()

    elif keyValue == 32:
        print("stop")
        carState = "stop"
        wheel_stop()
```

```python
    _, img = camera.read()
    #cv.imshow("Original", img)

    height, _, _ = img.shape
    img_cut = img[int(height/2):, :, :]

    gray_img = cv.cvtColor(img_cut, cv.COLOR_BGR2GRAY)

    blur_img = cv.GaussianBlur(gray_img, (7,7), 0)

    _, binary_img = cv.threshold(blur_img, 0, 255, cv.THRESH_BINARY+cv.THRESH_OTSU)

    canny_img = cv.Canny(blur_img, 100, 150)



    cv.imshow("Original", img)

    if carState == "left":
        cv.imwrite("%s_%05d_%03d.png" %(path, i, 45), canny_img)
        i += 1

    elif carState == "right":
        cv.imwrite("%s_%05d_%03d.png" %(path, i, 135), canny_img)
        i += 1

    elif carState == "go":
        cv.imwrite("%s_%05d_%03d.png" %(path, i, 90), canny_img)
        i += 1
```

이미지 파일 수집

키보드로 방향을 제어하여 앞, 좌, 우를 구분하여 전처리된 이미지 파일을 수집

WEEK 02

```python
# 이미지 전처리 및 추론
height, _, _ = frame.shape
frame_cut = frame[int(height/2):, :, :]

gray_frame = cv.cvtColor(frame_cut, cv.COLOR_BGR2GRAY)

blur_frame = cv.GaussianBlur(gray_frame, (7,7), 0)

_, binary_frame = cv.threshold(blur_frame, 0, 255, cv.THRESH_BINARY+cv.THRESH_OTSU)

canny_frame = cv.Canny(binary_frame, 100, 150)

frame = canny_frame / 255.0
frame = np.expand_dims(frame, axis=0)   # Add a batch dimension
frame = np.expand_dims(frame, axis=-1)  # Add the channel dimension
frame = frame.astype(np.float32)

interpreter.set_tensor(input_tensor_index, frame)
interpreter.invoke()
output_data = interpreter.get_tensor(output_tensor_index)

predicted_angle = output_data[0][0]
print(f'예측 각도: {predicted_angle}')
```

```python
if 68 < predicted_angle < 128:
    print('go')
    wheel_go()

elif 131 < predicted_angle < 145:
    print("right")
    wheel_right()

elif 40 < predicted_angle < 55:
    print('left')
    wheel_left()

else:
    print('stop')
    wheel_stop()


keyValue = cv.waitKey(delay)

if keyValue & 0xFF == 27:
    print('finish')
    break
```

CNN 모델을 불러와

RC카에 적용하여 구동

카메라에 보여지는 이미지의 각도를

예측해서 전진 및 좌,우 이동

WEEK 02

```
[ ]  # 학습 모델 구성
     def train_model():
         img_input = Input(shape=[120, 320, 1])
         H = Conv2D(12, (5,5))(img_input)
         H = BatchNormalization()(H)
         H = Activation('relu')(H)
         H = MaxPool2D()(H)
         H = Dropout(0.2)(H)

         H = Conv2D(32, (5,5))(img_input)
         H = BatchNormalization()(H)
         H = Activation('relu')(H)
         H = MaxPool2D()(H)
         H = Dropout(0.3)(H)

         H = Conv2D(32, (3,3))(H)
         H = BatchNormalization()(H)
         H = Activation('relu')(H)
         H = MaxPool2D()(H)

         H = Conv2D(8, (3,3))(H)
         H = BatchNormalization()(H)
         H = Activation('relu')(H)
         H = MaxPool2D()(H)

         H = Flatten()(H)
         H = Dropout(0.2)(H)

         H = Dense(20)(H)
         H = BatchNormalization()(H)
         H = Activation('relu')(H)

         H = Dense(10)(H)
         H = BatchNormalization()(H)
         H = Activation('relu')(H)

         H = Dense(5)(H)
         H = BatchNormalization()(H)
         H = Activation('relu')(H)

         img_output = Dense(1)(H)

         model = keras.models.Model(img_input, img_output)
         model.compile(loss='mse', optimizer='Adam')

         return model
```

```
Epoch 82/100
211/211 [==============================] - 5s 24ms/step - loss: 94.4728 - val_loss: 282.0556
Epoch 83/100
211/211 [==============================] - 5s 24ms/step - loss: 96.3770 - val_loss: 283.5792
Epoch 84/100
211/211 [==============================] - 5s 24ms/step - loss: 93.7119 - val_loss: 291.8171
Epoch 85/100
211/211 [==============================] - 5s 24ms/step - loss: 90.8613 - val_loss: 293.5360
Epoch 86/100
211/211 [==============================] - 5s 24ms/step - loss: 96.3836 - val_loss: 341.1849
Epoch 87/100
211/211 [==============================] - 5s 24ms/step - loss: 90.0716 - val_loss: 379.7145
Epoch 88/100
211/211 [==============================] - 5s 24ms/step - loss: 89.0694 - val_loss: 393.6039
Epoch 89/100
211/211 [==============================] - 5s 24ms/step - loss: 86.7644 - val_loss: 314.4246
Epoch 90/100
211/211 [==============================] - 5s 24ms/step - loss: 85.3020 - val_loss: 349.4422
Epoch 91/100
211/211 [==============================] - 5s 25ms/step - loss: 82.2061 - val_loss: 306.2502
Epoch 92/100
211/211 [==============================] - 5s 24ms/step - loss: 85.1183 - val_loss: 321.8221
Epoch 93/100
211/211 [==============================] - 5s 25ms/step - loss: 92.2902 - val_loss: 327.2849
Epoch 94/100
211/211 [==============================] - 5s 24ms/step - loss: 93.5145 - val_loss: 353.5376
Epoch 95/100
211/211 [==============================] - 5s 24ms/step - loss: 79.0666 - val_loss: 279.5193
Epoch 96/100
211/211 [==============================] - 5s 24ms/step - loss: 73.1792 - val_loss: 305.9805
Epoch 97/100
211/211 [==============================] - 5s 24ms/step - loss: 82.8537 - val_loss: 357.3875
Epoch 98/100
211/211 [==============================] - 5s 24ms/step - loss: 79.6577 - val_loss: 290.4259
Epoch 99/100
211/211 [==============================] - 5s 24ms/step - loss: 78.2725 - val_loss: 435.5838
Epoch 100/100
211/211 [==============================] - 5s 24ms/step - loss: 76.4661 - val_loss: 292.5804
```

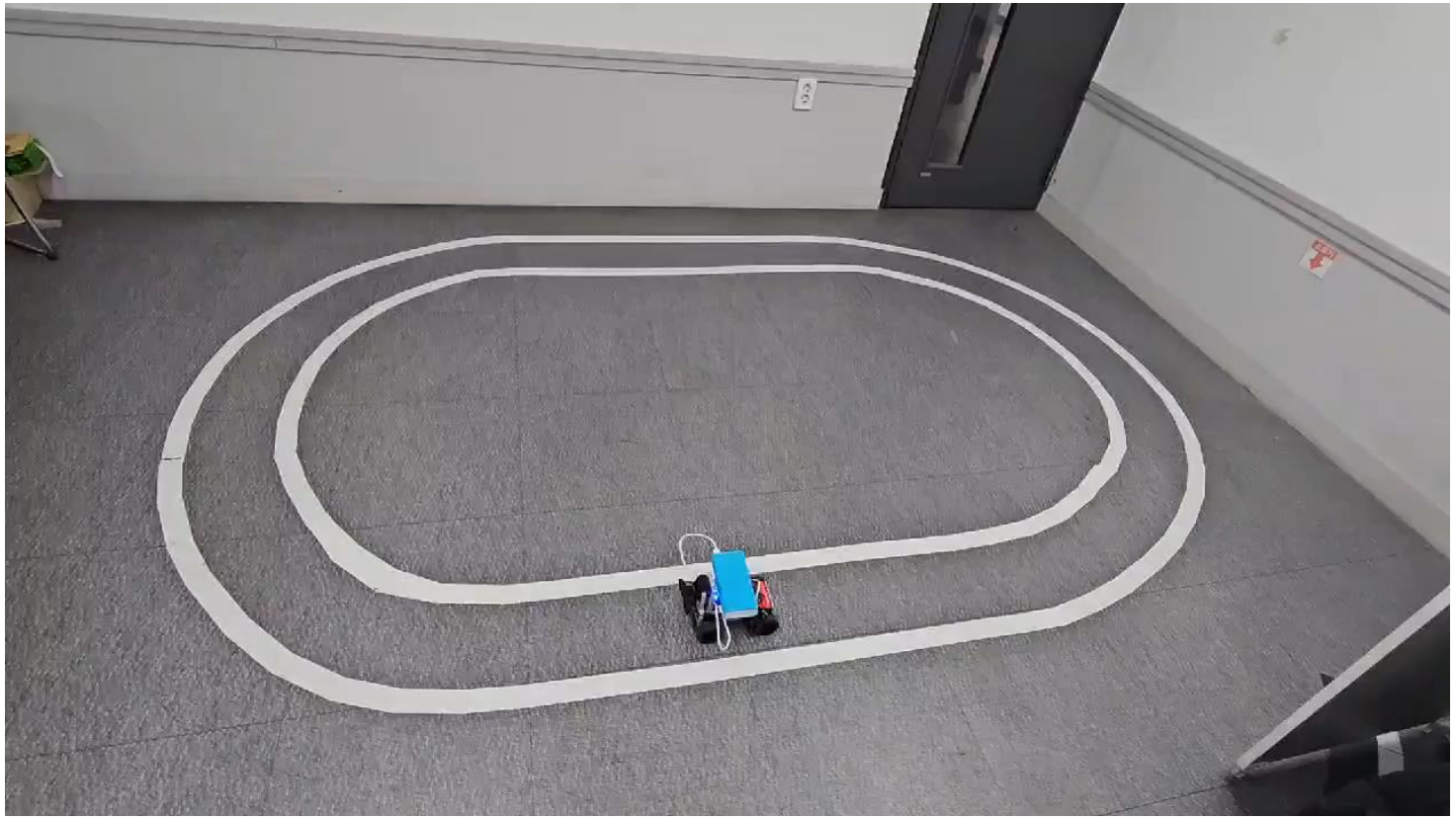
CNN 모델을 이용하여

학습 모델 구현

WEEK 03

학습에 필요한 차선 이미지 수집

WEEK 04



최종 학습된 모델을

이용한 자율주행

자체 평가

1. 바퀴와 바닥의 재질 차이로 자동차가 미끄러지는 현상이 발생하여 모터 제어에 어려움을 겪음.

2. 데이터를 수집하여 학습시키는 과정에서 자동차가 차선을 제대로 인식하지 못하는 오류를 많이 겪음.

3. 텐서플로우를 이용하여 학습할 때 오차율이 눈에 띄게 감소하지 않아 조정하는데 어려움을 겪음.

처음 계획했던 일정이나 방법으로 원활히 되지는 않았지만 위 사항들을 해결해 나가는 과정에서 문제를 바라보는 시선이 달라졌으며, 디버깅 능력이 향상되었습니다.

THANK YOU

감사합니다