Simple is the Best

Steve Jobs

저비용 센서기반 신호등 제어 시스템

팀 이름 : 활짝웃자

곽민경 (1-2)

박지효 (1-4)


지각...뛰면 된다!!


운전자 상황
(엄마 출현)

연구 동기

보행자와 운전자 모두
신호가 바뀔 때까지
오래 기다린다.

→

도로, 횡단보도 신호등에
금속 감지 센서를
부착하면 대기 시간을
줄일 수 있다.

↓

현재 여러 나라에서
센서기반 신호등
시스템을 사용 중이지만
가격이 비싸다.

→

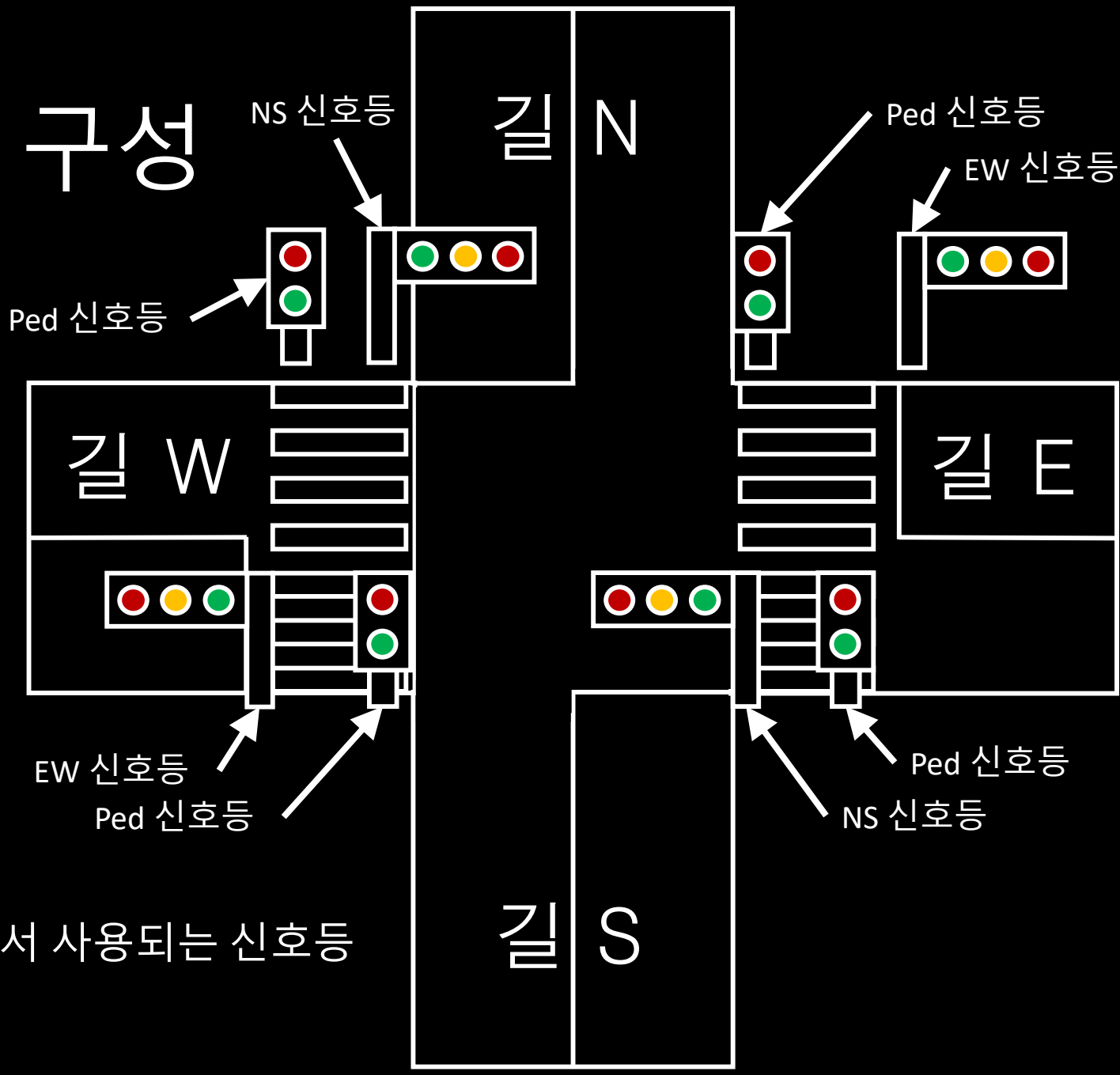
라즈베리 파이와
근접센서를 사용해,
저비용 센서기반 신호등
시스템 구축이 가능하다.

연구 목표

1. 보행자와 운전자의 평균 대기 시간을 현재 신호등 시스템보다 줄일 수 있다.
   - 센서기반 신호등 시스템과 현재 신호등 시스템 시뮬레이터를 프로그래밍하여 평균 대기 시간을 측정한다.

2. 라즈베리파이와 센서를 이용하여 센서기반 신호등 시스템을 만들 수 있다.
   - 라즈베리파이와 센서를 이용하여 모형으로 실제 구현한다.

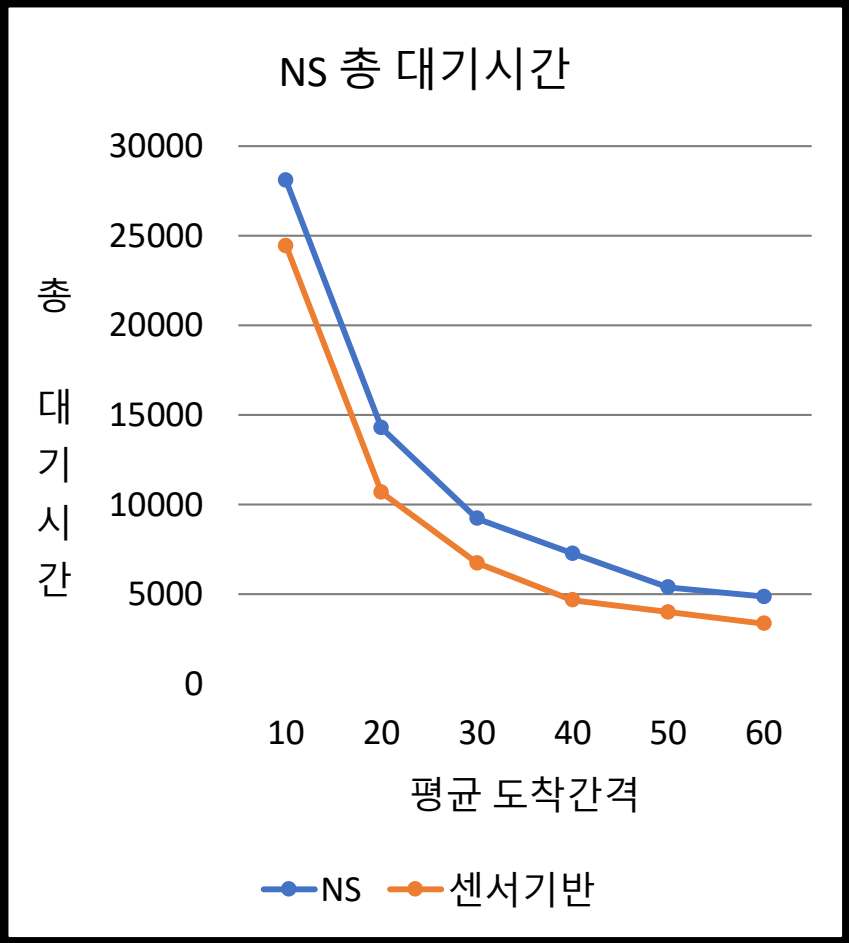
연구용 신호등 구성

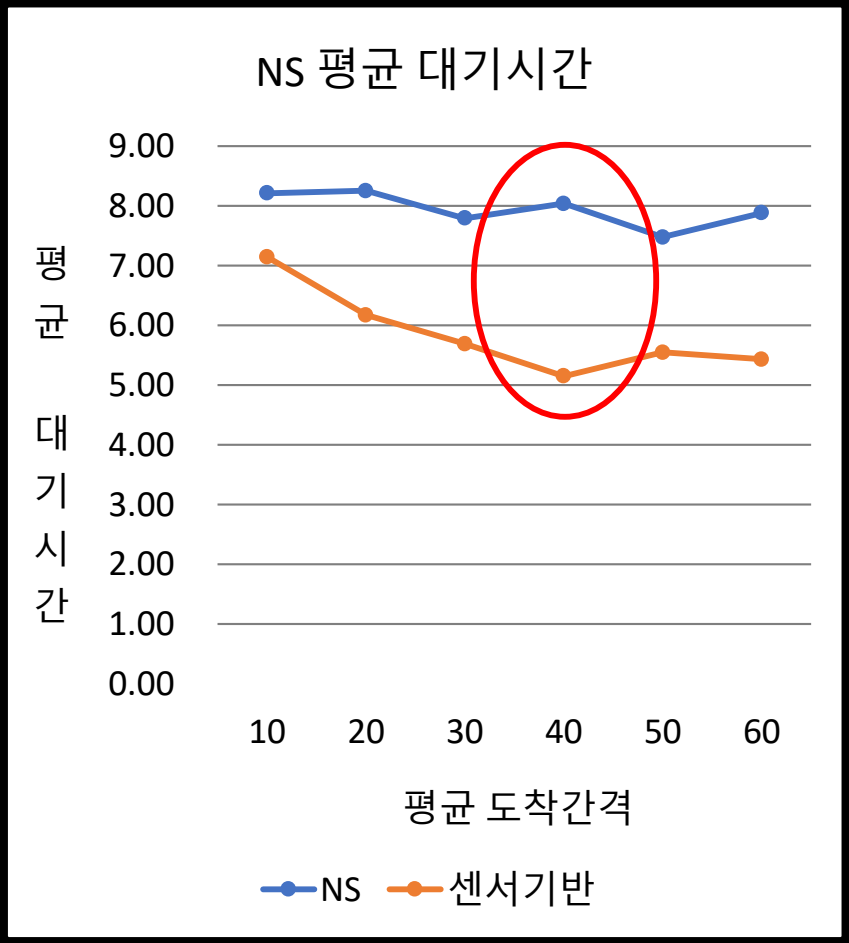


- 시뮬레이션에서 사용되는 신호등 구성이다.

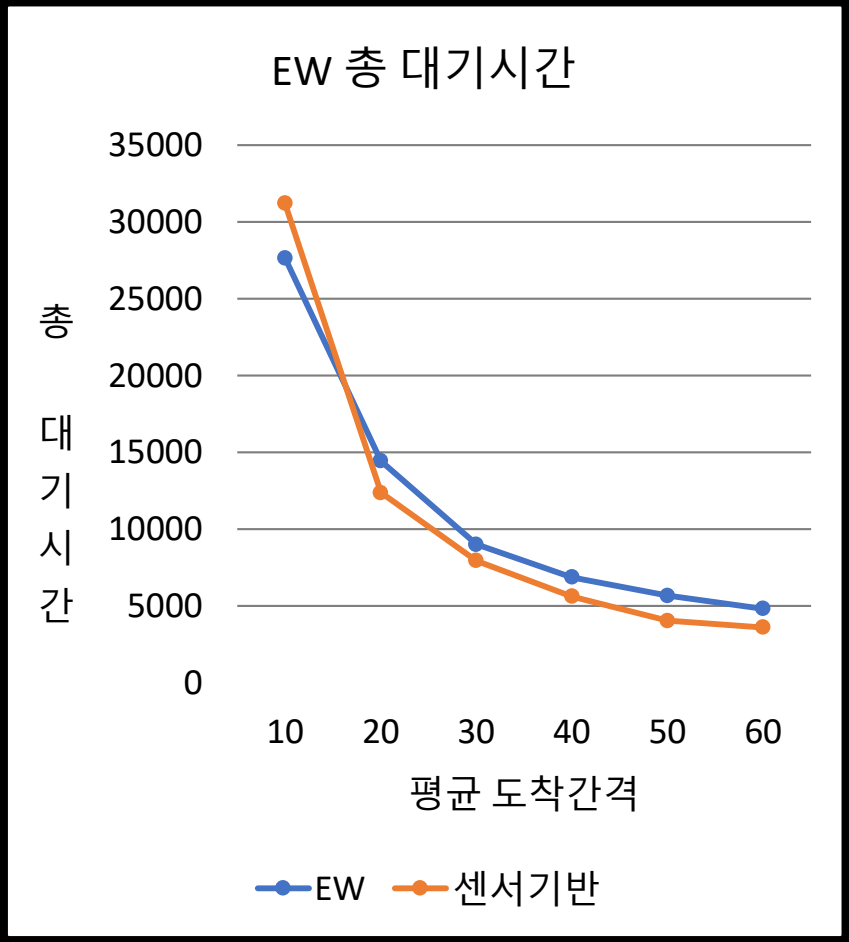
1. 평균 대기시간을 줄일 수 있다

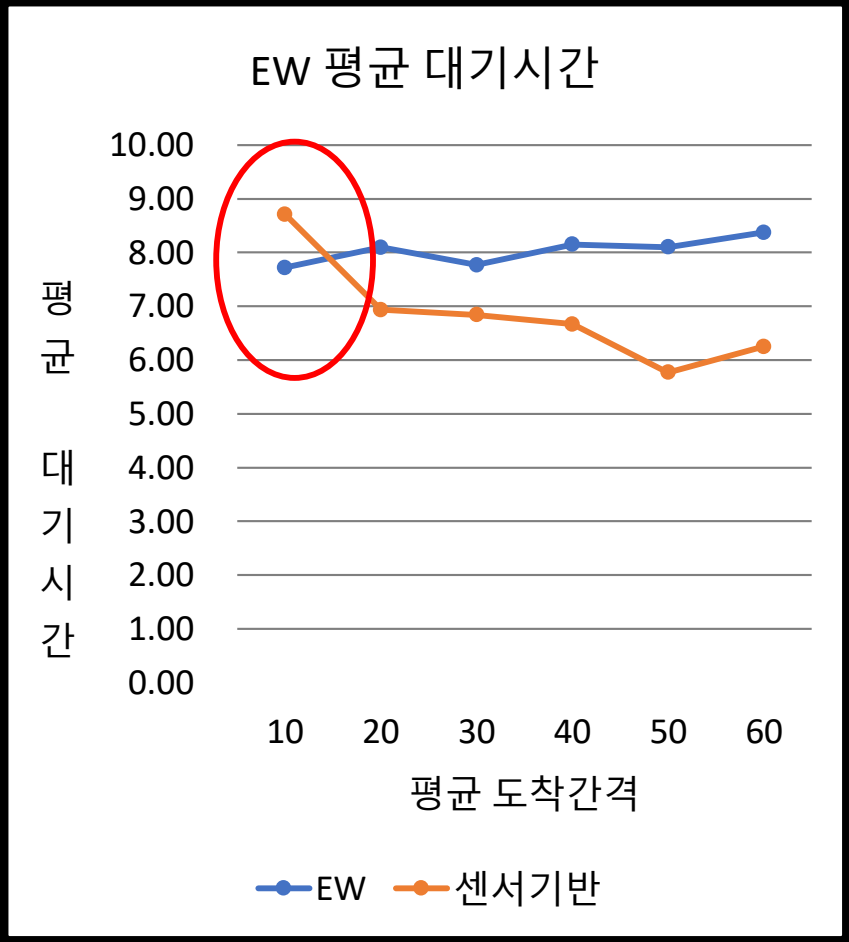
NS 총 대기시간, 평균 대기시간



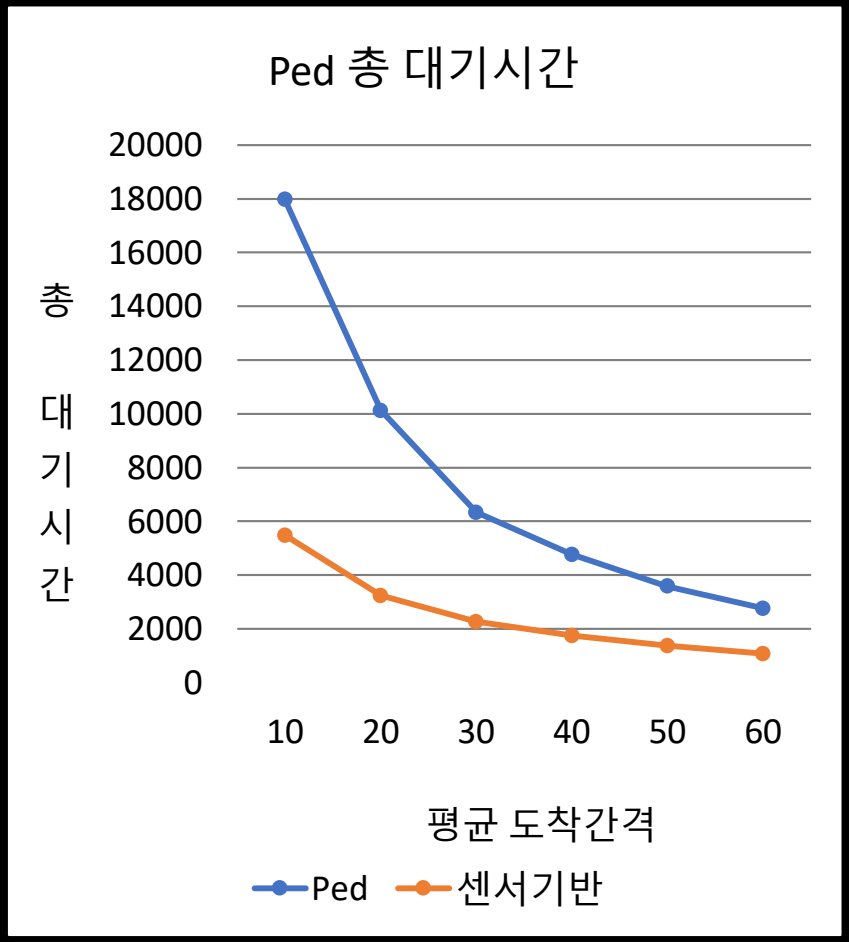


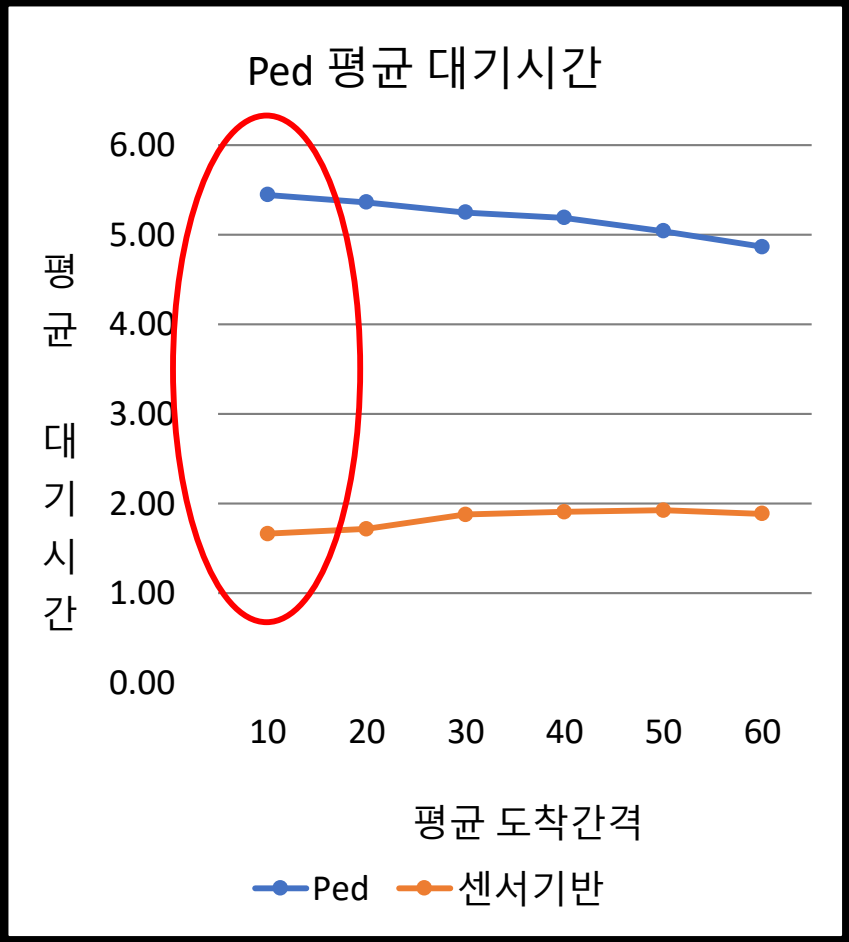
EW 총 대기시간, 평균 대기시간





Ped 총 대기시간, 평균 대기시간





sim.py 코드

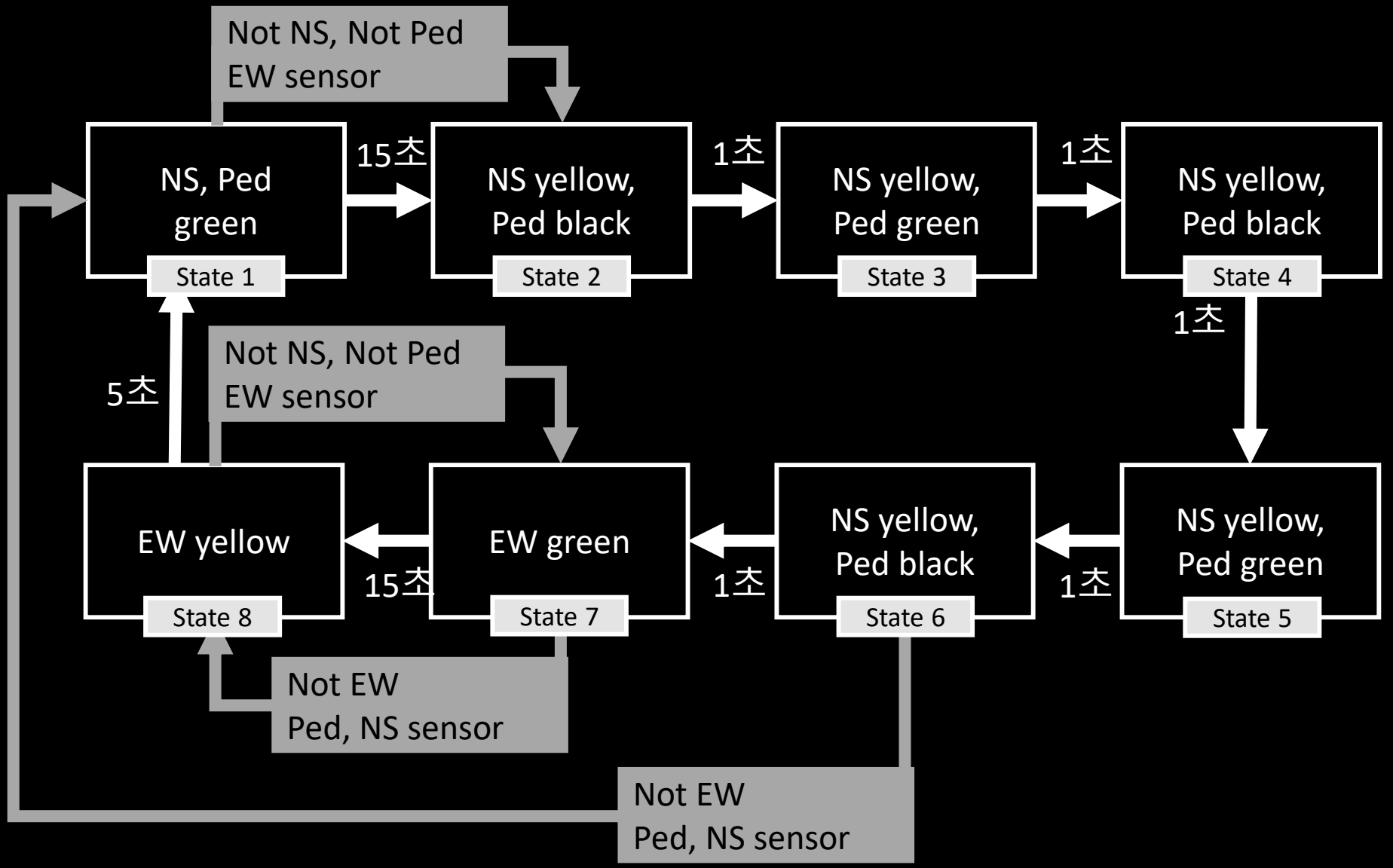
```
import random
import math
def nextTime(rateParameter): # functions
that cause random values
return -math.log(1.0 - random.random()) /
rateParameter # Poisson distribution
random.seed(21) # [7]
q_ns = [] # how many cars are on the ns
road for a certain period of time
q_ew = [] # how many cars are on the ew
road for a certain period of time
q_ped = [] # how many people are on the
crosswalk for a certain period of time
for i in range(36000): # 10 hours
simulation
q_ns.append(0)
q_ew.append(0)
q_ped.append(0)
t1 = 0 # t = time
t2 = 0
t3 = 0
while t1 < 36000:
q_ns[int(t1)] = 1
t1 = t1 + math.ceil(nextTime(1/10.0)) #
increased by 10 seconds
while t2 < 36000:
q_ew[int(t2)] = 1
t2 = t2 + math.ceil(nextTime(1/10.0))
while t3 < 36000:
q_ped[int(t3)] = 1
t3 = t3 + math.ceil(nextTime(1/10.0))
state = 1
t = 0
i = 0
ns_car = 0 # total cars on ns road
ew_car = 0 # total cars on ew road
ped = 0 # total people on crosswalk
w_ns = 0 # total time cars wait on ns road
w_ew = 0 # total time cars wait on ew road
w_ped = 0 # total time people wait on
crosswalk
while t < 36000:
print "time:", t
if i < 15:
state = 1
print state
if i >= 15 and i < 20:
state = 2
print state
if i >= 20 and i < 35:
state = 3
print state
if i >= 35 and i < 40:
state = 4
print state
if i == 40:
state = 1
print state
i = 0
if state == 1: #ns, ped
ns_car = ns_car + q_ns[t]
ew_car = ew_car + q_ew[t]
if q_ew[t] == 1:
w_ew = w_ew + 20 - i
if state == 2: #yellow
ns_car = ns_car + q_ns[t]
ew_car = ew_car + q_ew[t]
if q_ns[t] == 1:
w_ns = w_ns + 40 – I
if q_ew[t] == 1:
w_ew = w_ew + 20 - i
if state == 3: #ew
ns_car = ns_car + q_ns[t]
ew_car = ew_car + q_ew[t]
ped = ped + q_ped[t]
if q_ns[t] == 1:
w_ns = w_ns + 40 - i
if q_ped[t] == 1:
w_ped = w_ped +40 - i
if state == 4: #yellow
ns_car = ns_car + q_ns[t]
ew_car = ew_car + q_ew[t]
ped = ped + q_ped[t]
if q_ns[t] == 1:
w_ns = w_ns + 40 - i
if q_ew[t] == 1:
w_ew = w_ew + 60 - i
if q_ped[t] == 1:
w_ped = w_ped + 40 - i
if state == 1 or state == 2:
ped = ped + q_ped[t]
i = i + 1
t = t + 1
print "ns - cars:", ns_car
print "ew - cars:", ew_car
print "ped - people:", ped
print "ns - cars waiting signal, average
time:", w_ns, float(w_ns)/ns_car # average
waiting time on ns road
print "ew - cars waiting signal, average
time:", w_ew, float(w_ew)/ew_car # average
waiting time on ew road
print "ped - people waiting signal,
average time:", w_ped, float(w_ped)/ped #
average waiting time on crosswalk
```

sim_sensor.py 코드

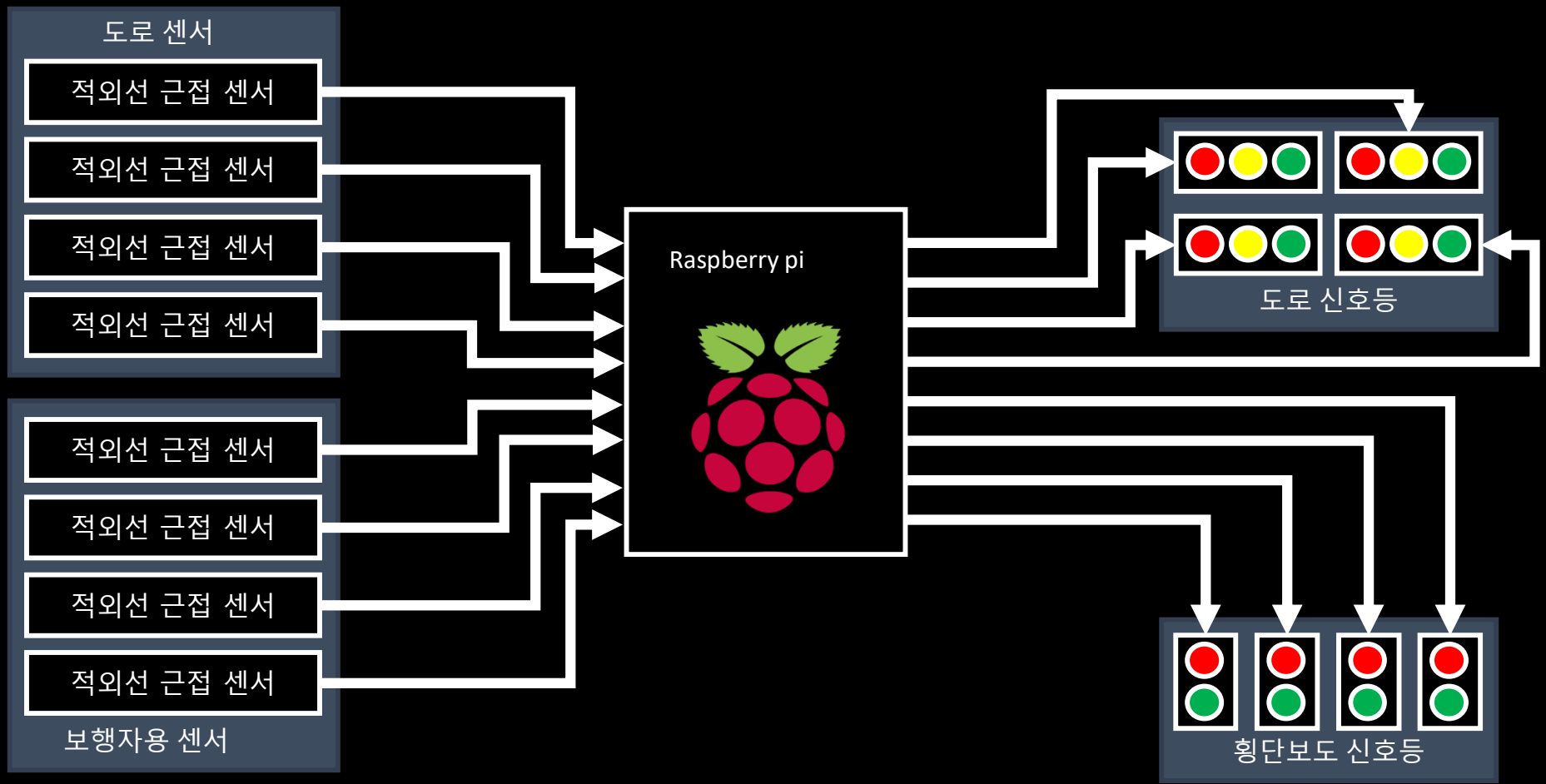
```
import random
import math
def nextTime(rateParameter): #
functions that cause random
values
return -math.log(1.0 -
random.random()) / rateParameter
# Poisson distribution
random.seed(21) # [7]
q_ns = [] # how many cars are on
the ns road for a certain period
of time
q_ew = [] # how many cars are on
the ew road for a certain period
of time
q_ped = [] # how many people are
on the crosswalk for a certain
period of time
for i in range(36000): # 10 hours
simulation
q_ns.append(0)
q_ew.append(0)
q_ped.append(0)
t1 = 0 # t = time
t2 = 0
t3 = 0
while t1 < 36000:
q_ns[int(t1)] = 1
t1 = t1 +
math.ceil(nextTime(1/10.0)) #
increased by 10 seconds
while t2 < 36000:
q_ew[int(t2)] = 1
t2 = t2 +
math.ceil(nextTime(1/10.0))
while t3 < 36000:
q_ped[int(t3)] = 1
t3 = t3 +
math.ceil(nextTime(1/10.0))
state = 1
t = 0
i = 0
ns_car = 0 # total cars on ns
road
ew_car = 0 # total cars on ew
road
ped = 0 # total people on
crosswalk
w_ns = 0 # total time cars wait
on ns road
w_ew = 0 # total time cars wait
on ew road
w_ped = 0 # total time people
wait on crosswalk
ox_ns = 0 # whether there is car
on ns road
ox_ew = 0 # whether there is car
on ew road
ox_ped = 0 # whether there is car
on crosswalk
while t < 36000:
print "time:", t
if q_ns[t] == 1:
ox_ns = 1
if q_ew[t] == 1:
ox_ew = 1
if q_ped[t] == 1:
ox_ped = 1
if i < 15:
state = 1
ox_ns = 0
ox_ped = 0
print state
if q_ns[t] == 0 and ox_ew == 1
and q_ped[t] == 0: # if there is
no car
on ns road, there is a car on ew
road, there is no car on
crosswalk
i = 15
if i >= 15 and i < 20:
state = 2
print state
if i >= 20 and i < 35:
state = 3
ox_ew = 0
print state
if ox_ns == 1 and q_ew[t] == 0
and ox_ped == 0:
# if there is a car on ns road,
there is no car on ew road, there
is no car on crosswalk
i = 35
if ox_ns == 1 and q_ew[t] == 0
and ox_ped == 1: # if there is a
car on ns road, there is no car
on ew road, there is a car on
crosswalk
i = 35
if ox_ns == 0 and q_ew[t] == 0
and ox_ped == 1: # if there is no
car on ns road, there is no car
on ew road, there is a car on
crosswalk
i = 35
if i >= 35 and i < 40:
state = 4
print state
if i == 40:
state = 1
print state
i = 0
if state == 1: # ns, ped
ns_car = ns_car + q_ns[t]
ew_car = ew_car + q_ew[t]
if q_ew[t] == 1:
w_ew = w_ew + 20 - i
if state == 2: # yellow
ns_car = ns_car + q_ns[t]
ew_car = ew_car + q_ew[t]
if q_ns[t] == 1:
w_ns = w_ns + 40 - i
if q_ew[t] == 1:
w_ew = w_ew + 20 - i
if state == 3: # ew
ns_car = ns_car + q_ns[t]
ew_car = ew_car + q_ew[t]
ped = ped + q_ped[t]
if q_ns[t] == 1:
w_ns = w_ns + 40 - i
if q_ped[t] == 1:
w_ped = w_ped +40 - i
if state == 4: # yellow
ns_car = ns_car + q_ns[t]
ew_car = ew_car + q_ew[t]
ped = ped + q_ped[t]
if q_ns[t] == 1:
w_ns = w_ns + 40 - i
if q_ew[t] == 1:
w_ew = w_ew + 60 - i
if q_ped[t] == 1:
w_ped = w_ped + 40 - i
if state == 1 or state == 2:
ped = ped + q_ped[t]
i = i + 1
t = t + 1
print "ns - cars:", ns_car
print "ew - cars:", ew_car
print "ped - people:", ped
print "ns - cars waiting signal,
average time:", w_ns, waiting
average time:", w_ns,
float(w_ns)/ns_car # average
waiting time on ns road
print "ew - cars waiting signal,
average time:", w_ew,
float(w_ew)/ew_car # average
waiting time on ew road
print "ped - people waiting
signal, average time:", w_ped,
float(w_ped)/ped # average
waiting time on crosswalk
```

2. 저비용 센서기반 시스템 구축이 가능하다

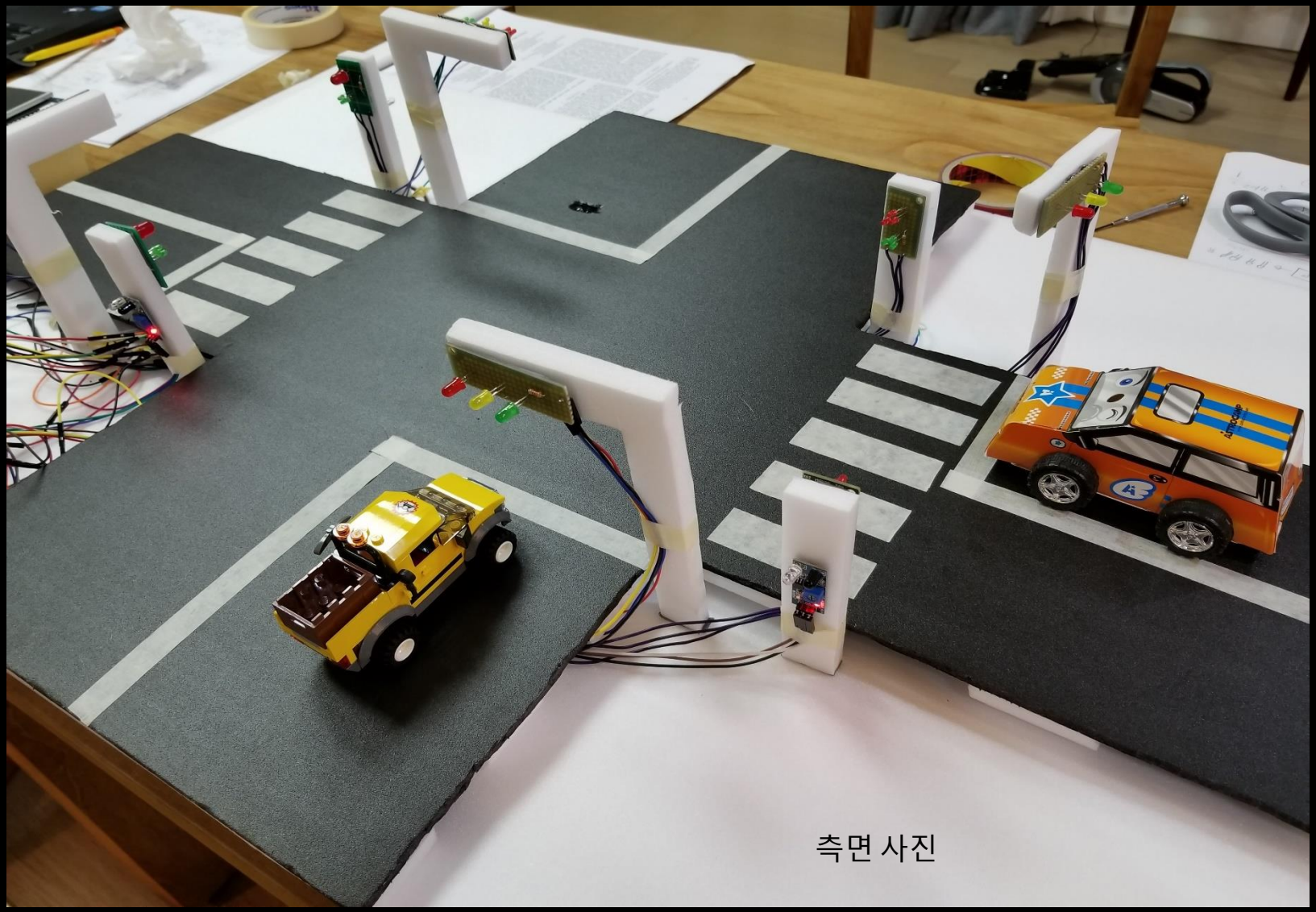
센서기반 신호등 시스템 순서도



센서기반 신호등 시스템 주요 구성



센서기반 신호등 시스템 모형



측면 사진

shinho.py 코드

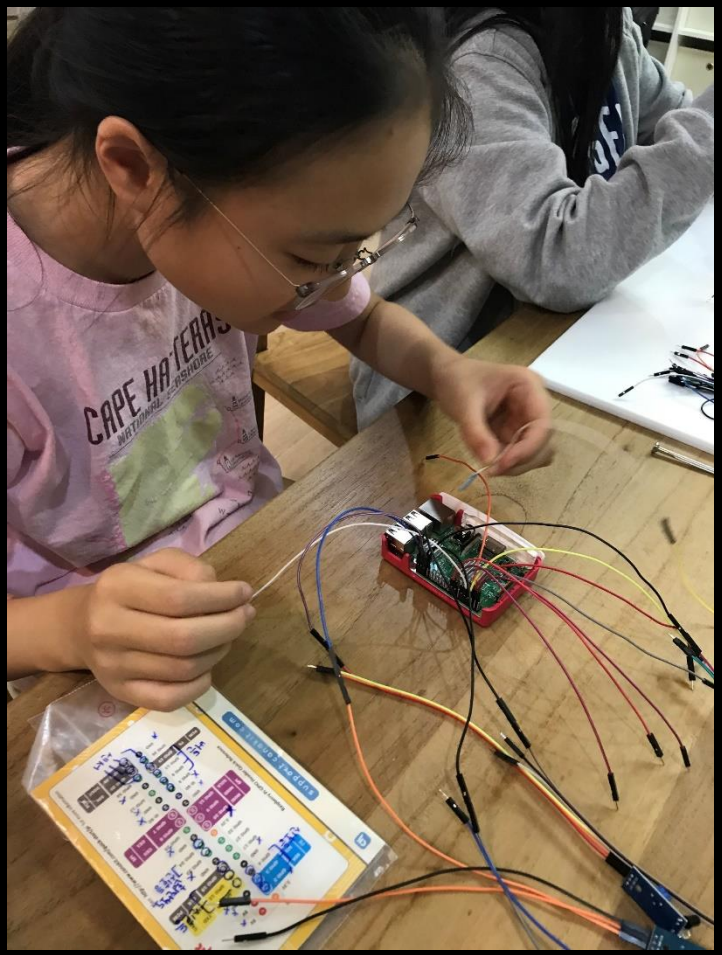
```
import time
import RPi.GPIO as GPIO
import time
import threading
from gpiozero import MotionSensor
def light(b):
#Fuction to turn on the light accordin
g to the array
GPIO.output(23, True)
GPIO.output(24, True)
GPIO.output(2, True)
GPIO.output(3, True)
GPIO.output(4, True)
GPIO.output(14, True)
GPIO.output(15, True)
GPIO.output(18, True)
if shinN[b] == 'G':
GPIO.output(2, False)
if shinN[b] == 'Y':
GPIO.output(3, False)
if shinN[b] == 'R':
GPIO.output(4, False)
if shinS[b] == 'G':
GPIO.output(2, False)
if shinS[b] == 'Y':
GPIO.output(3, False)
if shinS[b] == 'R':
GPIO.output(4, False)
if shinW[b] == 'G':
GPIO.output(14, False)
if shinW[b] == 'Y':
GPIO.output(15, False)
if shinW[b] == 'R':
GPIO.output(18, False)
if shinE[b] == 'G':
GPIO.output(14, False)
if shinE[b] == 'Y':
GPIO.output(15, False)
if shinE[b] == 'R':
GPIO.output(18, False)
if hshinW[b] == 'G':
GPIO.output(23, False)
if hshinW[b] == 'R':
GPIO.output(24, False)
if hshinE[b] == 'G':
GPIO.output(23, False)
if hshinE[b] == 'R':
GPIO.output(24, False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.OUT)#ped light
GPIO.setup(24, GPIO.OUT)
GPIO.setup(2,GPIO.OUT)#traffic   light
SN
GPIO.setup(3, GPIO.OUT)
GPIO.setup(4, GPIO.OUT)
GPIO.setup(14,GPIO.OUT)#traffic   light
EW
GPIO.setup(15, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)
sensorn = MotionSensor(16)
sensors = MotionSensor(20)
sensore = MotionSensor(21)
sensorw = MotionSensor(26)
buttonn = MotionSensor(5)
buttons = MotionSensor(6)
buttone = MotionSensor(13)
buttonw = MotionSensor(19)
GPIO.output(23, True)
GPIO.output(24, True)
GPIO.output(2, True)
GPIO.output(3, True)
GPIO.output(4, True)
GPIO.output(14, True)
GPIO.output(15, True)
GPIO.output(18, True)
i = 1
ped1 = 0
ped2 = 0
ped3 = 0
ped4 = 0
shinN = ['G', 'Y', 'Y', 'Y', 'Y', 'Y',
'R',  'R']#g=green,  y=yellow,  r=red,
b=black
shinS = ['G', 'Y', 'Y', 'Y', 'Y', 'Y',
'R', 'R']
shinE = ['R', 'R', 'R', 'R', 'R', 'R',
'G', 'Y']
shinW = ['R', 'R', 'R', 'R', 'R', 'R',
'G', 'Y']
hshinW = ['G', 'B', 'G', 'B', 'G', 'B',
'R', 'R']
hshinE = ['G', 'B', 'G', 'B', 'G', 'B',
'R', 'R']
state = 1
a = 1
while True:
time.sleep(1)
EW = 0
NS = 0
nsen = sensorn.motion_detected
ssen = sensors.motion_detected
esen = sensore.motion_detected
wsen = sensorw.motion_detected
ped1 = buttonn.motion_detected
ped2 = buttone.motion_detected
ped3 = buttonw.motion_detected
ped4 = buttons.motion_detected
if not nsen or not ssen or not ped1 or
not ped2 or not ped3 or not ped4:
NS = 1
if not esen or not wsen:
EW = 1
b = state - 1
print i, " ", shinN[b], " ", shinS[b],
"  ",  shinE[b],  "  ",  shinW[b],  "  ",
hshinW[b], " ", hshinE[b], " ", state
print
light(b)
i = i + 1
if NS == 1 and EW == 0 and state == 6:
state = 1
a = 0
if NS == 1 and EW == 0 and state == 7:
state = state + 1
a = 0
if EW == 1 and NS == 0 and state == 8:
state = 7
a = 0
if EW == 1 and NS == 0 and state == 1:
state = state + 1
a = 0
if a == 15:
state = state + 1
a = 0
elif a == 1 and state == 2:
state = state + 1
a = 0
elif a == 1 and state == 3:
state = state + 1
a = 0
elif a == 1 and state == 4:
state = state + 1
a = 0
elif a == 1 and state == 5:
state = state + 1
a = 0
elif a == 1 and state == 6:
state = state + 1
a = 0
elif a == 5 and state == 8:
state = 1
a = 0
a = a + 1
```

신호등 시스템 비교

기존 신호등 시스템의 동작 순서	센서기반 신호등 시스템의 동작 순서
• 정해진 순서대로 불이 켜짐 • 길NS 초록불, 횡단보도 초록불(15초) • 길NS노란불, 횡단보도 깜빡임(5초) • 길EW초록불(15초) • 길EW노란불(5초)	• 기존 신호등 시스템 순서대로 가다가 현재 켜진 것과 반대의 센서가 감지되면 그 신호를 켬 • 지나가는 차나 사람이 없으면, 차나 사람이 대기 중인 신호를 켜 줌

모형제작을 위한 연구 진행 과정

납땜 작업



라즈베리파이에 전선 연결

연구 결과 정리

연구 결과

보행자와 운전자의 평균 대기 시간 비교

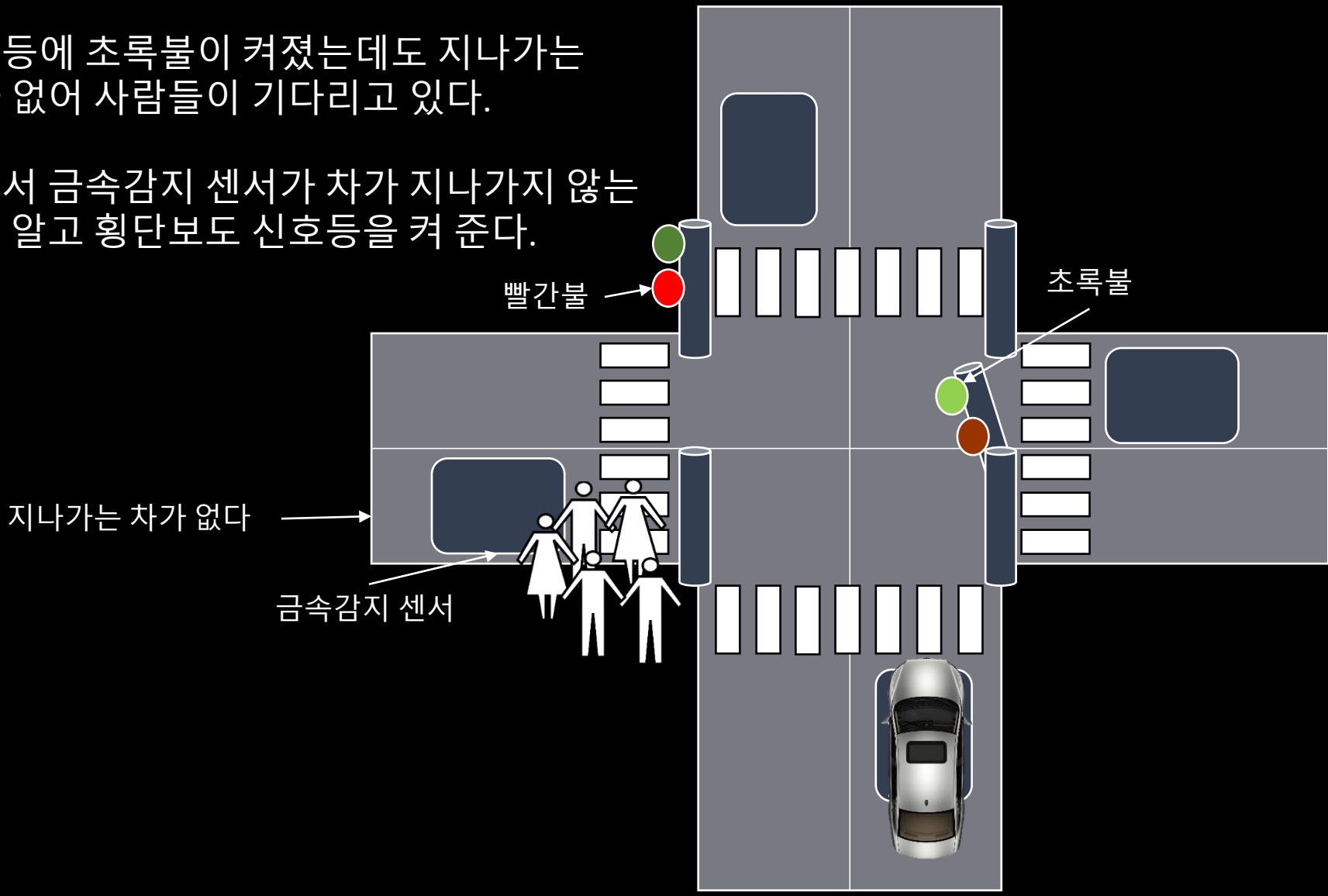
	일반 신호등	센서기반 신호등	개선 폭
운전자	7.98초	6.36초	약 20% 감소
보행자	5.19초	1.83초	약 65% 감소

- 센서기반 신호등 시스템을 저비용의 적외선 근접센서로 실현 가능
- 라즈베리파이를 이용한 모형으로 제작하는 목표 달성

감사합니다

센서기반 신호등의 일반적 구성

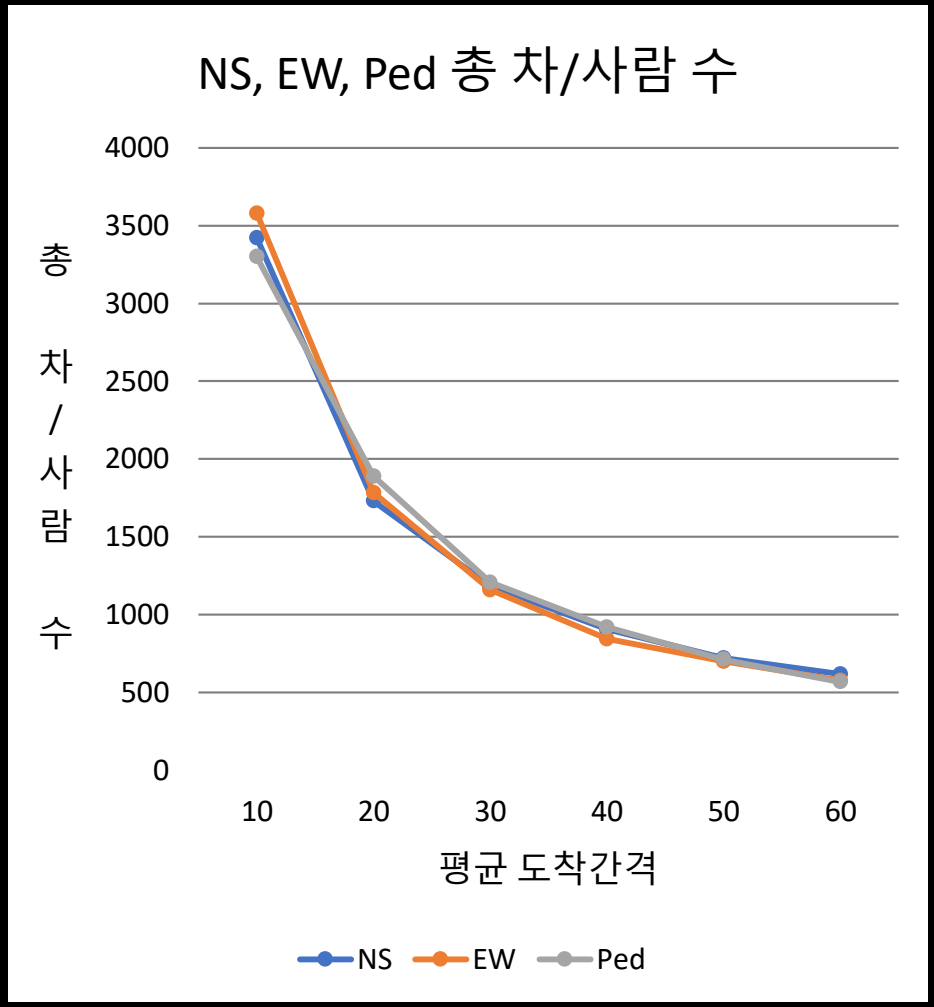
- 신호등에 초록불이 켜졌는데도 지나가는 차가 없어 사람들이 기다리고 있다.
- 그래서 금속감지 센서가 차가 지나가지 않는 것을 알고 횡단보도 신호등을 켜 준다.



실험에 사용된 데이터
(NS, EW, Ped 총 차/사람 수)

- Poisson(포아송) 분포를 이용하여 무작위로 차량이나 보행자를 생성

ex) 10초 - NS, EW, Ped의 총 수 =약 3,600명/대 정도



평균 도착 간격동안 차가 한 대씩 올 때 NS 도로, EW도로, 횡단보도에서 기다리는 차량 / 사람들의 수