

2021

# Portfolio :

# Index :

## 선정 및 개발 목적

## 기술 및 개발 동향

## 시스템 순서도

## 개발 일정

## 자율주행 프로젝트

## 솔리드웍스

- 모티브 선정 및 설계

## 회로도

- Schematic
- PCB
- Gerber
- Custom board

## 통신 및 모터제어

- UART
- PWM

## 피드백 제어

## 이미지 알고리즘

- 선형 회귀 이론 및 적용
- 이미지 프로세싱 과정 및 결과
- 표지판 인식

## Part list

- Atmega328p, Raspberry Pi 4
- Shift register (74HC595)
- Motor driver (L298N)
- Ultrasonic sensor (HC-SR04)

## Web

- Server (Flask)
- Database
- Web page
- Email

## 핵심 코드

- 차선 인식
- UART
- Email

선정 및 개발 목적 :

# 자율 주행 자동차

---

2021.02 ~ 06

허의철, 박준표, 김상길



자율주행차는 첨단 센서와 알고리즘을 사용하여 안전한 주행을 약속하는 기술입니다. 따라서 자율주행차의 도입으로 교통사고로 인한 사상자 수는 급격하게 줄어들 것으로 기대할 수 있습니다. 또한 전기자동차 기술을 차용하여 환경에 유해한 매연을 발생시키지 않고, 클린에너지를 효율적으로 사용하는 환경보호 기술의 대표 사례가 될 것으로 전망됩니다. 매년 천문학적인 재정이 소모되는 교통 인프라 유지보수 비용도 극적으로 줄어들 것으로 예상되고, 주차에 필요한 도심의 공간 부족 문제도 해결할 수 있을 것입니다.

# 기술 및 개발 동향



라이더 센서	센서가 360°로 회전하면 레이저를 발생시켜서 반사된 빛을 통해 주변 사물과의 거리, 속도, 방향을 확인합니다.
레이더 센서	전자기파를 발생시켜서 되돌아오는 신호를 분석하여 물체와의 거리를 판단합니다.
GPS	GPS를 통해서 현재위치를 파악합니다.
카메라	카메라를 통해 받아들이는 이미지를 분석하여 주행상황을 확인합니다.

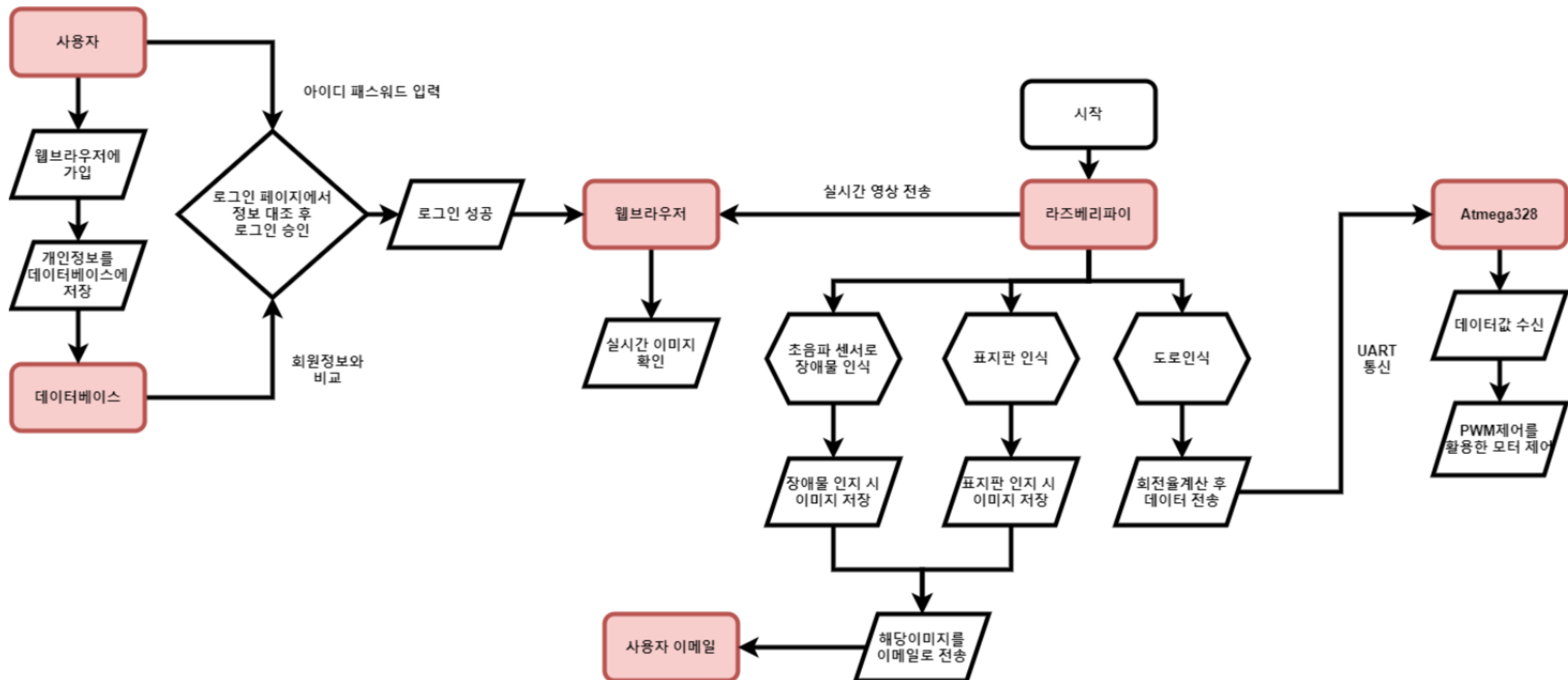
## <자율주행 레벨 5단계 구분>

레벨 0	자동화영역이 없으며 모든 작동을 운전자가 다루어야 합니다.
레벨 1	초보적인 운전이 가능하며 적응식 정속주행시스템, 자동 긴급정지를 자동화합니다.
레벨 2	운전자의 감시 하에 자동화 주행을 실행합니다. 유사시 운전자가 개입하여 주행합니다.
레벨 3	주변상황을 모두 파악하며 운전자의 개입없이도 부분적으로 자율주행이 가능하다. 횡단보도, 행인, 교차로, 신호등 인식 등이 자동제어 됩니다.
레벨 4	운전자의 개입없이 차량 스스로 출발지에서 목적지까지 운행을 관리 제어합니다.

■현재 많은 기업들이 레벨 2~3 수준을 갖고 있으며 2025년 이후에 레벨4에 도달하는 것을 목표로 두고 있습니다.

기업들은 자율주행 분야에서 경쟁 중이며 특허 등을 출허하며 각자의 기술을 보유하기 위해 노력하고 있습니다.

# 순서도



순서	내 용
1	라즈베리파이와 Atmega328사이에서 UART통신으로 정보를 주고 받습니다.
2	라즈베리파이의 카메라는 표지판과 도로를 인식합니다.
3	라즈베리파이에서 도로를 인식하여 회전방향을 모터의PWM값으로 연산하여 Atmega328로 전달합니다.
4	Atmega328은 전달받은 데이터로 모터를 제어합니다.
5	표지판을 인지할 경우 표지판에 맞는 제어 값을 계산하여 데이터를 전달하며
6	초음파센서를 통해 근접한 물체를 인지시에는 주행을 정지하고 사진을 촬영하여
7	이용자의 이메일로 사진을 보냅니다



## 개발 일정

기간 강조 표시: 1

**% 완료**

■ % 완료(계획 초과)

■ 중간 점검

[illegible]

# 자율 주행 RC카 :



자율 주행 RC카 사진.



■ 카메라를의 영상 처리를 이용한 자율주행 알고리즘 시스템을 구성하여 운전자 없이도 운행이 가능하도록 하는 것을 목적으로 합니다.



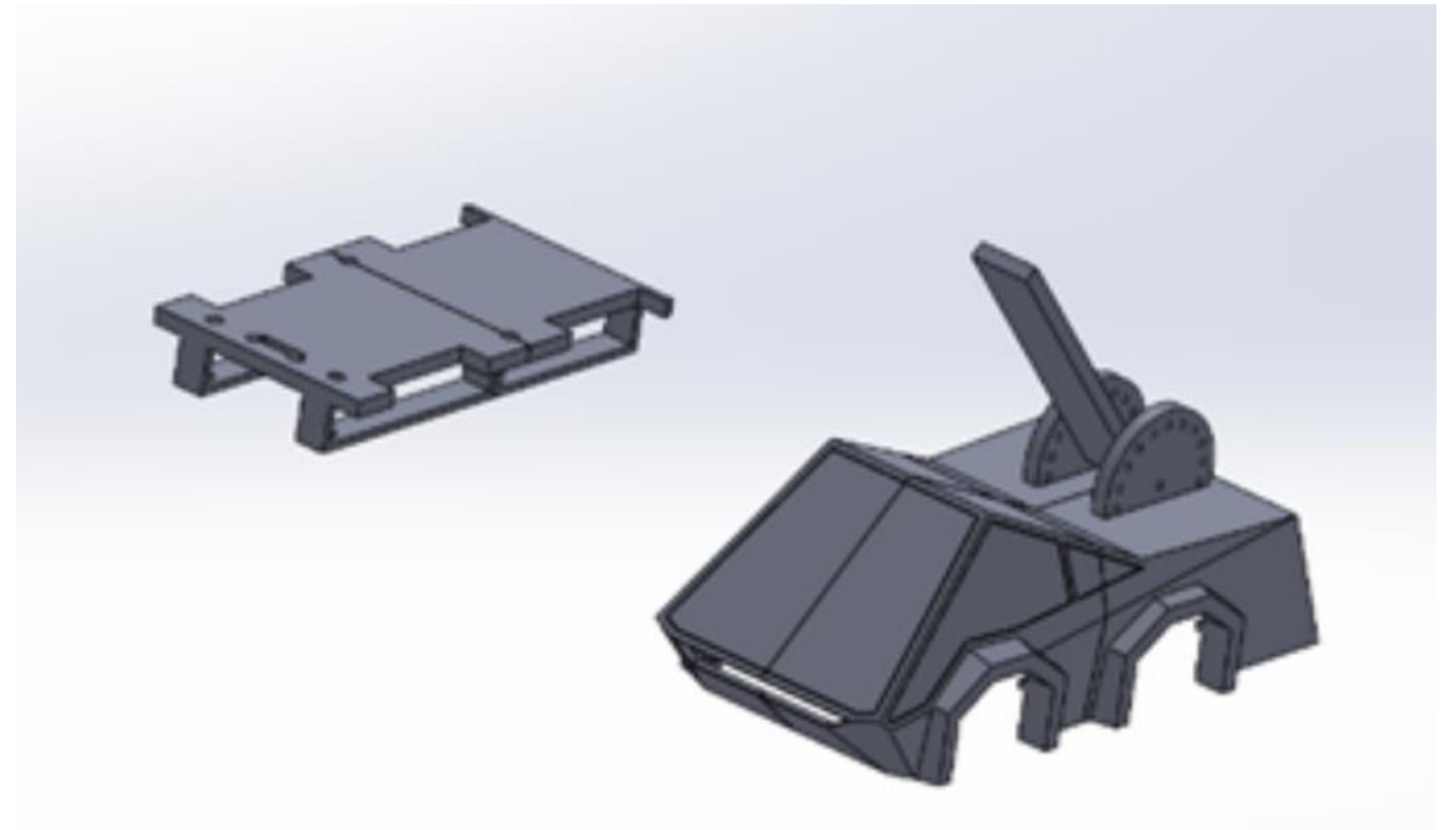
# 솔리드 워크스 :

## 1. 설계의 모티브



"사이버 트럭"에서 각진 디자인과 미래지향적인 디자인을 참고하였습니다.  
영화 마션에 나오는 "로버"라는 자동차의 유틸리티성과 넓은 윈도우에서 설계의 영향을 받았습니다.

## 2. 솔리드 워크스를 이용한 설계

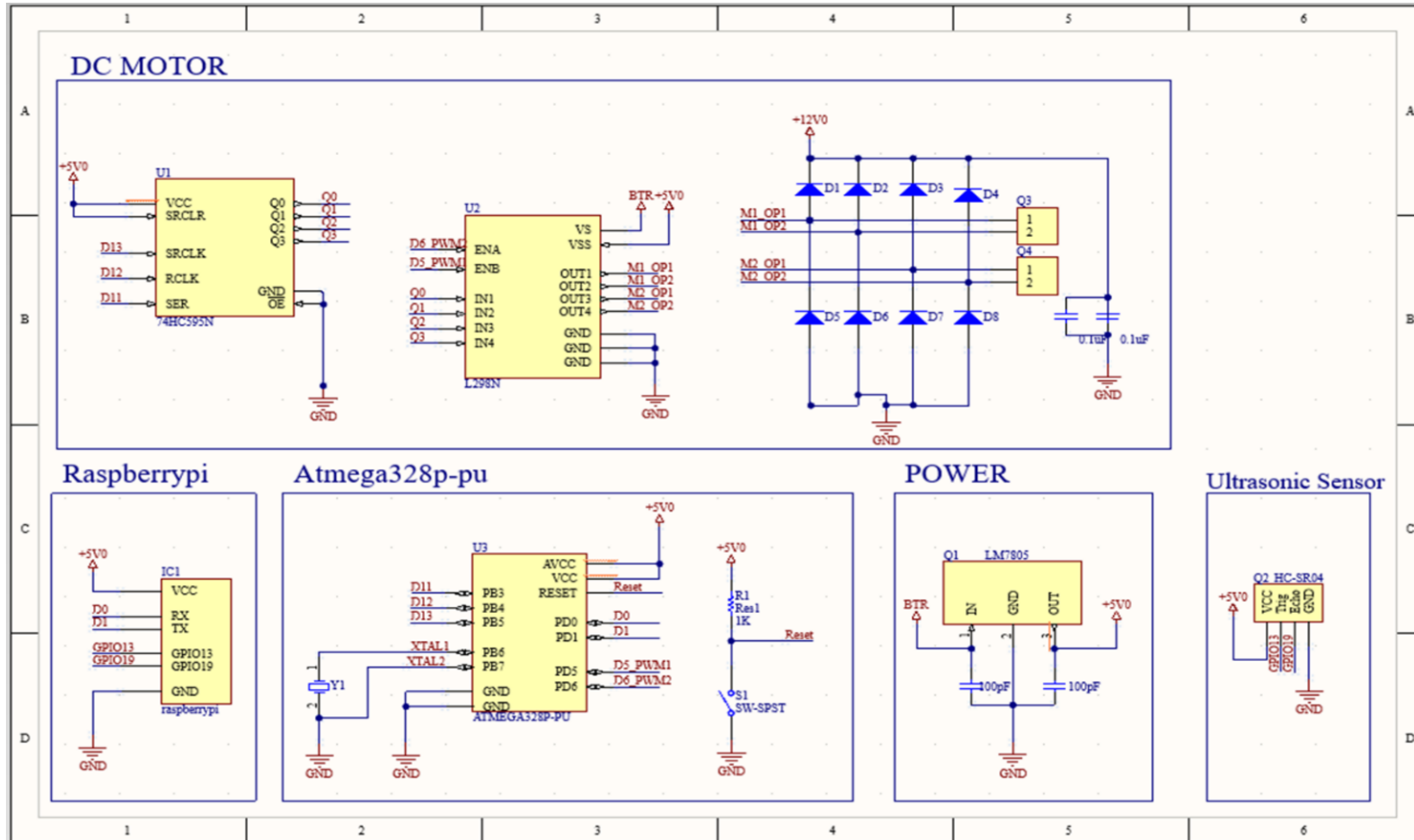


자동차 케이스 안에 넉넉한 공간을 두어 라즈베리파이, 커스텀보드, 건전지등이 들어갈 공간을 충분히 확보하였고 자동차의 뒷부분에 카메라 지지대를 두어서 카메라 설치를 용이하게 하였습니다.

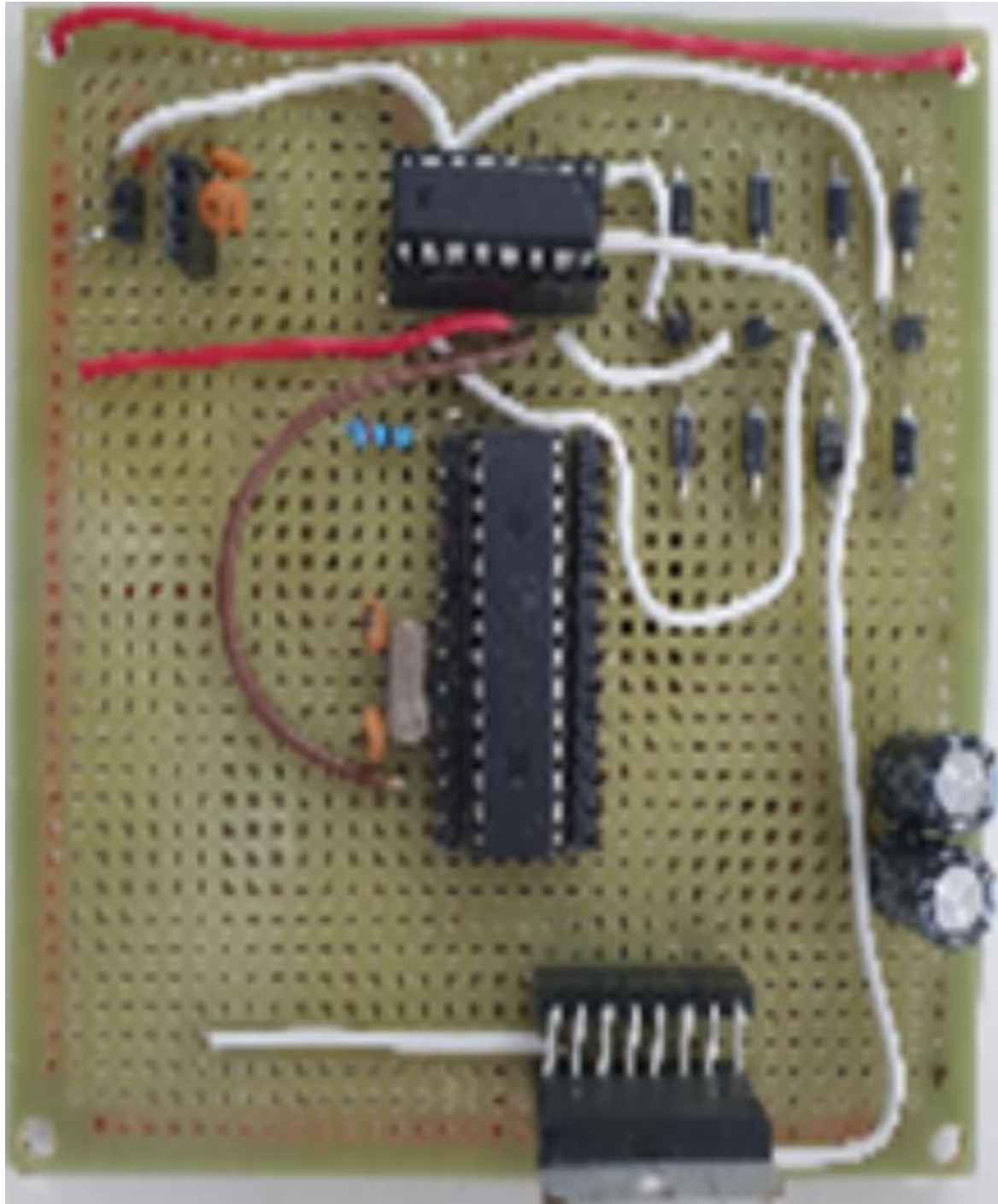


# 회로도

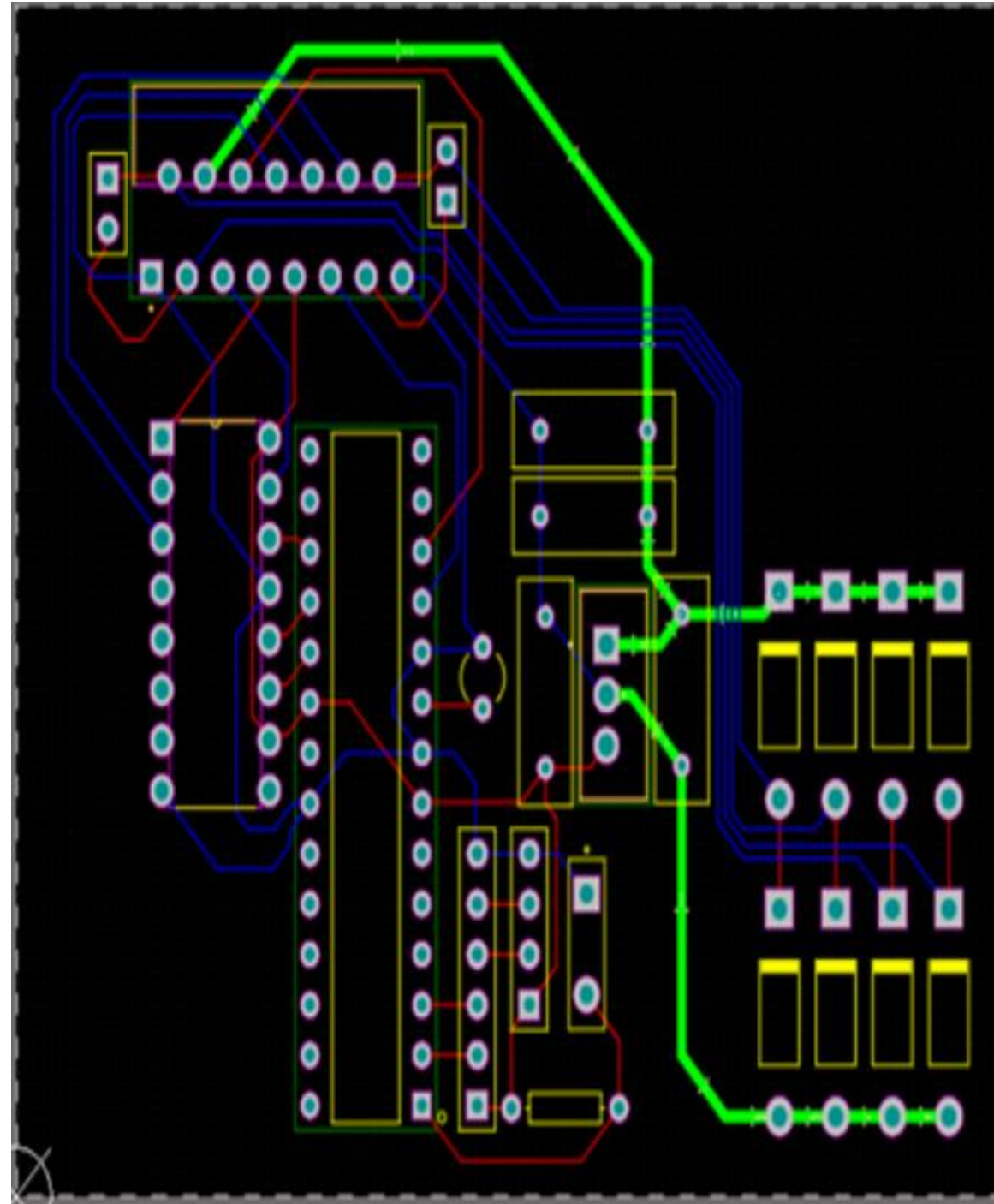
<Schematic>



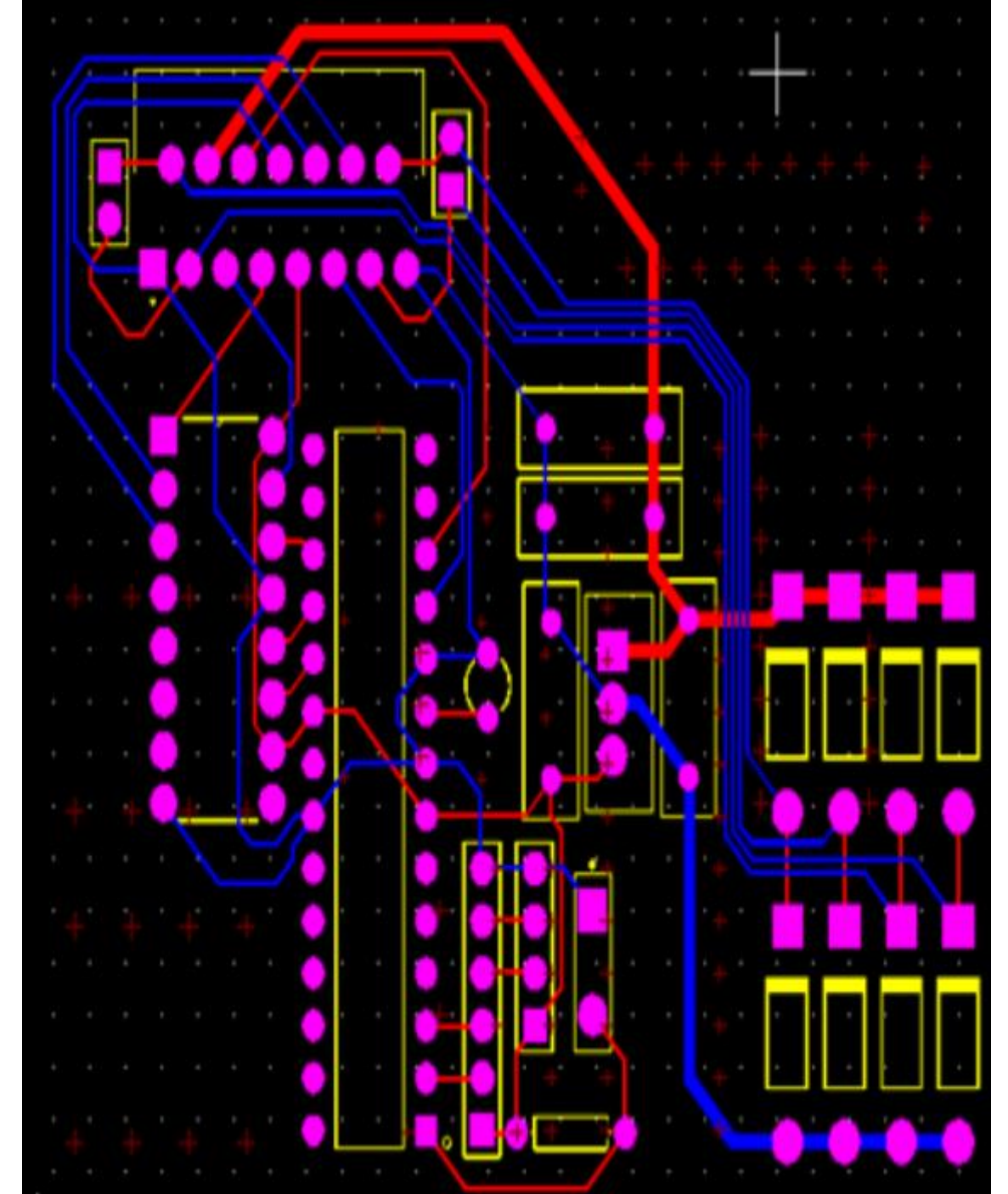
atmega328 커스텀 보드



PCB

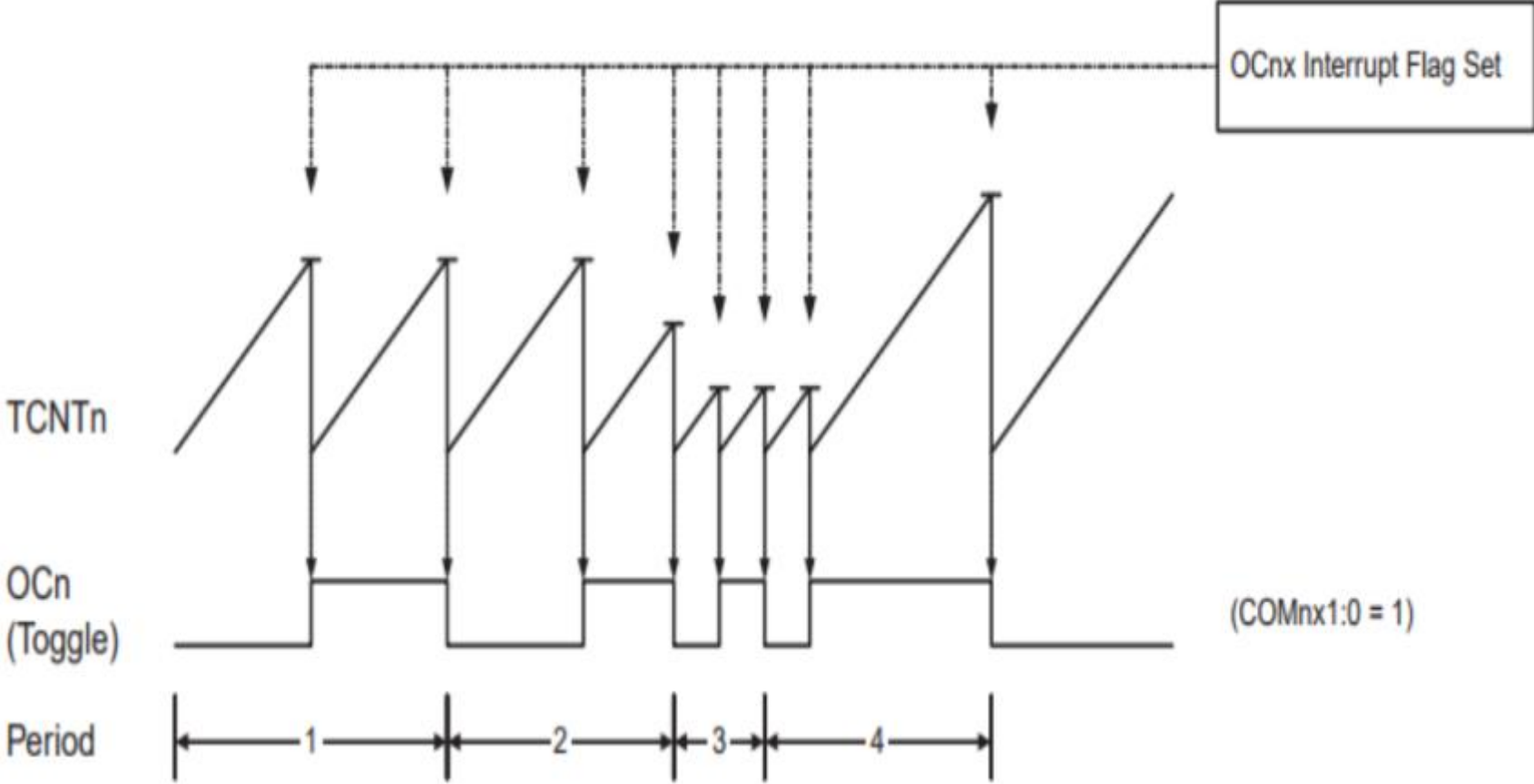


Gerber





# 통신 및 모터제어



## <UART 통신>

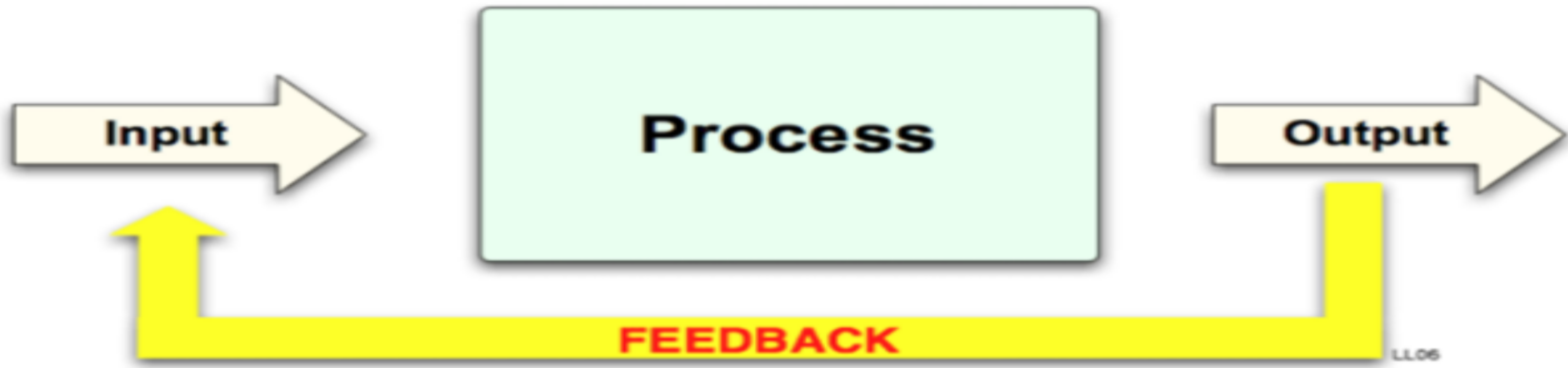
순서	내 용
1	라즈베리파이와 Atmega328 커스텀보드와 baud rate를 9600으로 맞춥니다.
2	Atmega328의 RX인터럽트를 켜주고 UART 통신 관련 레지스터에서 UART통신을 설정합니다.
3	데이터 전송(TX) 시, 데이터의 시작과 끝을 알려주기 위해 시작 문자 '_' 와 끝 문자 '/'를 데이터와 함께 보냅니다. ex) "_(data)/"
4	데이터 수신(RX) 시, 데이터의 시작 문자 '_' 부터 끝 문자 '/' 까지 데이터를 받아 읽습니다.

## <PWM 제어>

순서	내 용
1	타이머 인터럽트를 이용하여 PWM을 제어합니다.
2	타이머 인터럽트가 발생하면 PWM의 해당 핀은 출력을 합니다.
3	카운트가 255에 도달하면 다시 0 으로 초기화하고 해당 핀의 출력은 목표 값이 될 때까지 0이 유지되고, 목표 값 이후로 출력 값은 1이 됩니다.



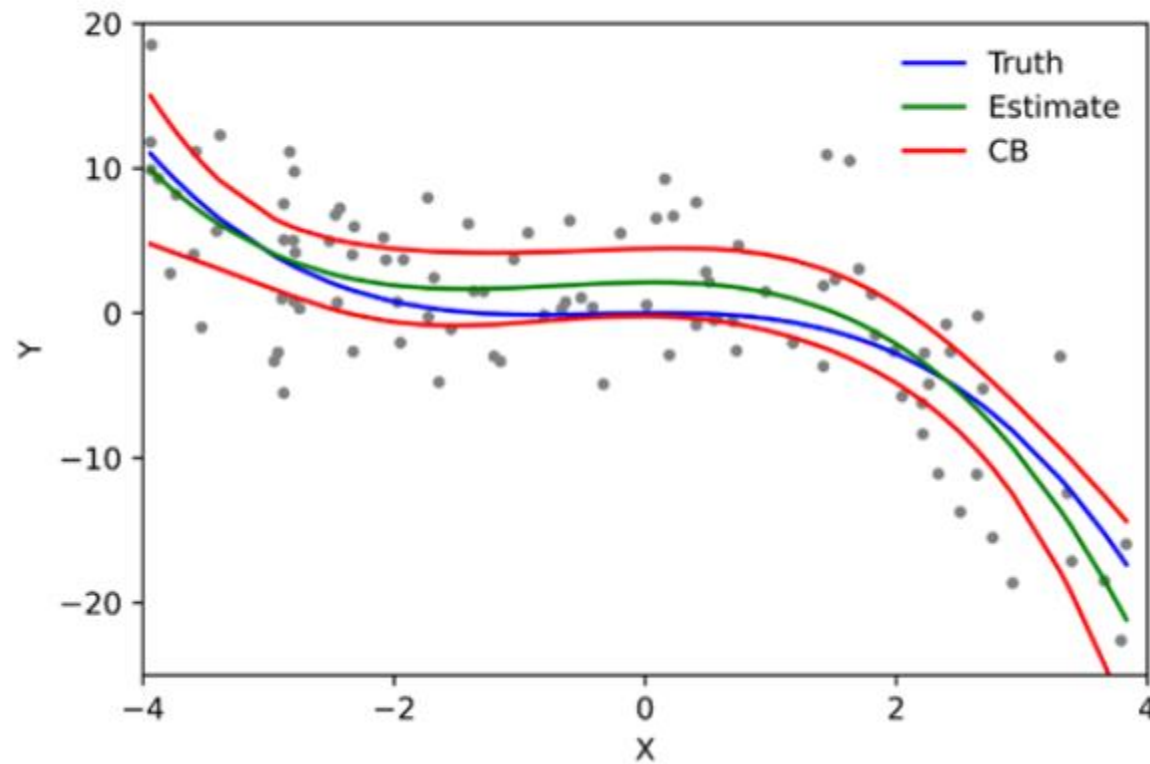
# 피드백 제어 :



순서	내 용
1	PWM 제어를 통해 바퀴의 회전을 제어합니다.
2	주행 시 직진 차선을 따라 차량이 운동을 시작합니다.
3	회전 구간 시 도로 중앙과 카메라의 중심 초점 간의 거리의 차를 제어기에 피드백해 줍니다.
4	피드백 받은 값으로 모터의 PWM 값을 보정하여, 도로와 카메라의 중심 간의 오차를 줄입니다.

# 이미지알고리즘

## 1. 선형회귀 이론



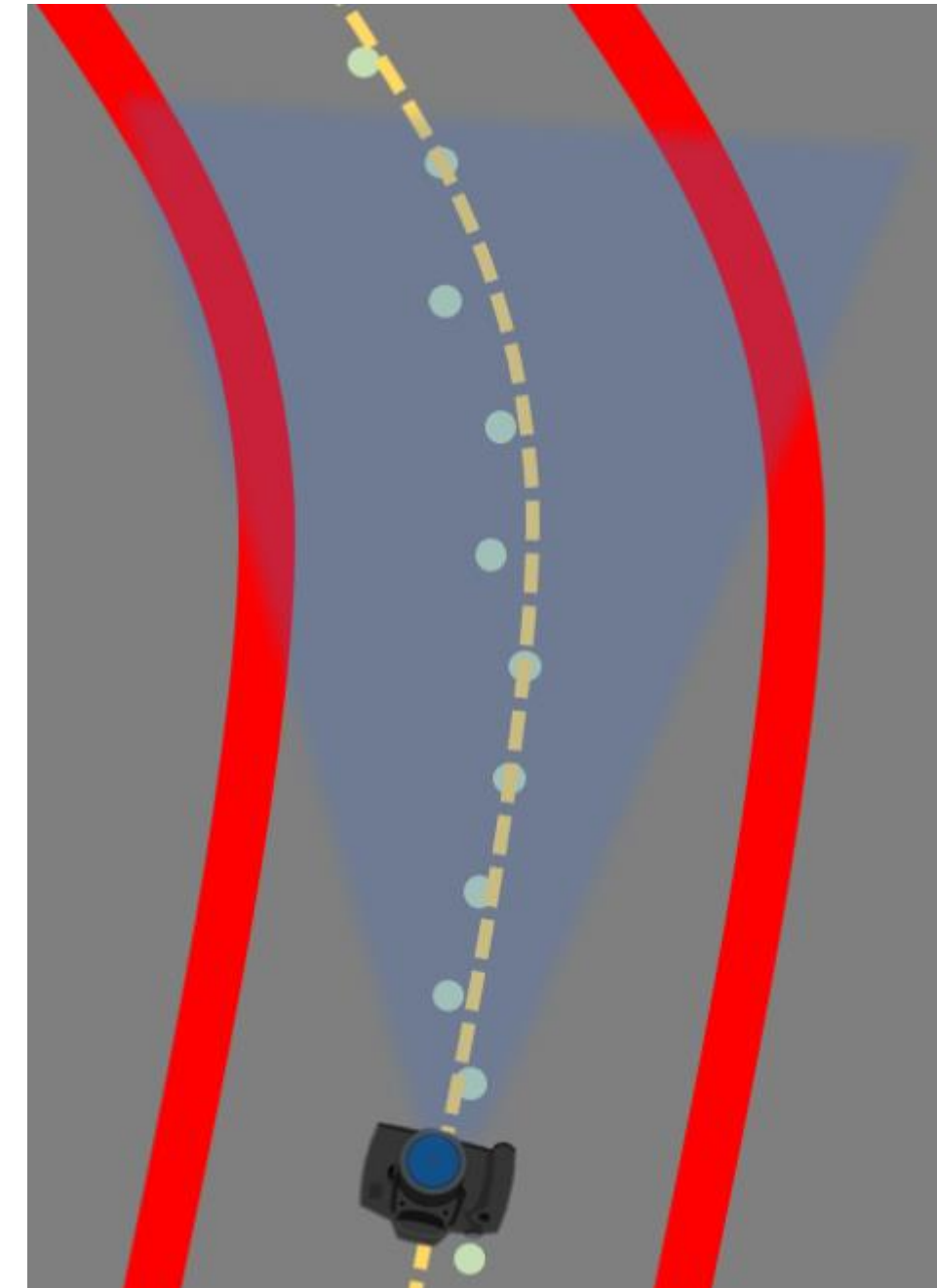
<선형 회귀의 예>

이미지의 차선의 중심을 이미지 픽셀의 행기준으로 규칙적인 간격을 두고 좌표를 획득합니다.

좌표는 선형회귀를 이용해서 차선을 2차 함수 그래프로 나타냈습니다.

선형 회귀는 종속 변수  $y$ 와 한 개 이상의 독립 변수 (또는 설명 변수)  $X$ 와의 선형 상관 관계를 모델링하는 회귀분석 기법입니다.

## 2. 선형회귀 적용

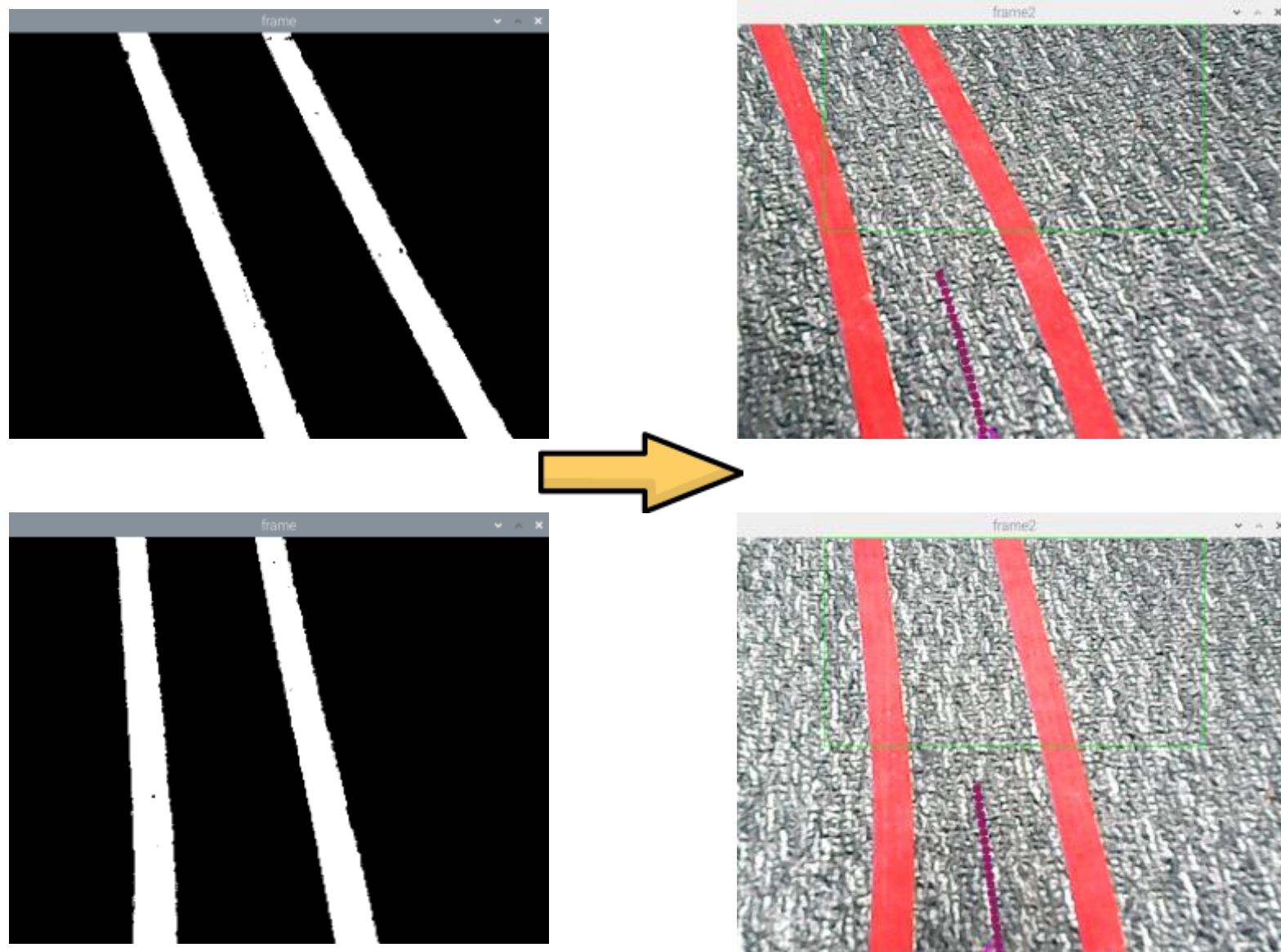


왼쪽 사진과 같이 일정 간격으로 도로의 중심 좌표를 구하고 그 좌표들을 이용하여 2차함수로 선형회귀를 하여 자동차가 차선을 인식할 수 있도록 했습니다.



# 이미지알고리즘:

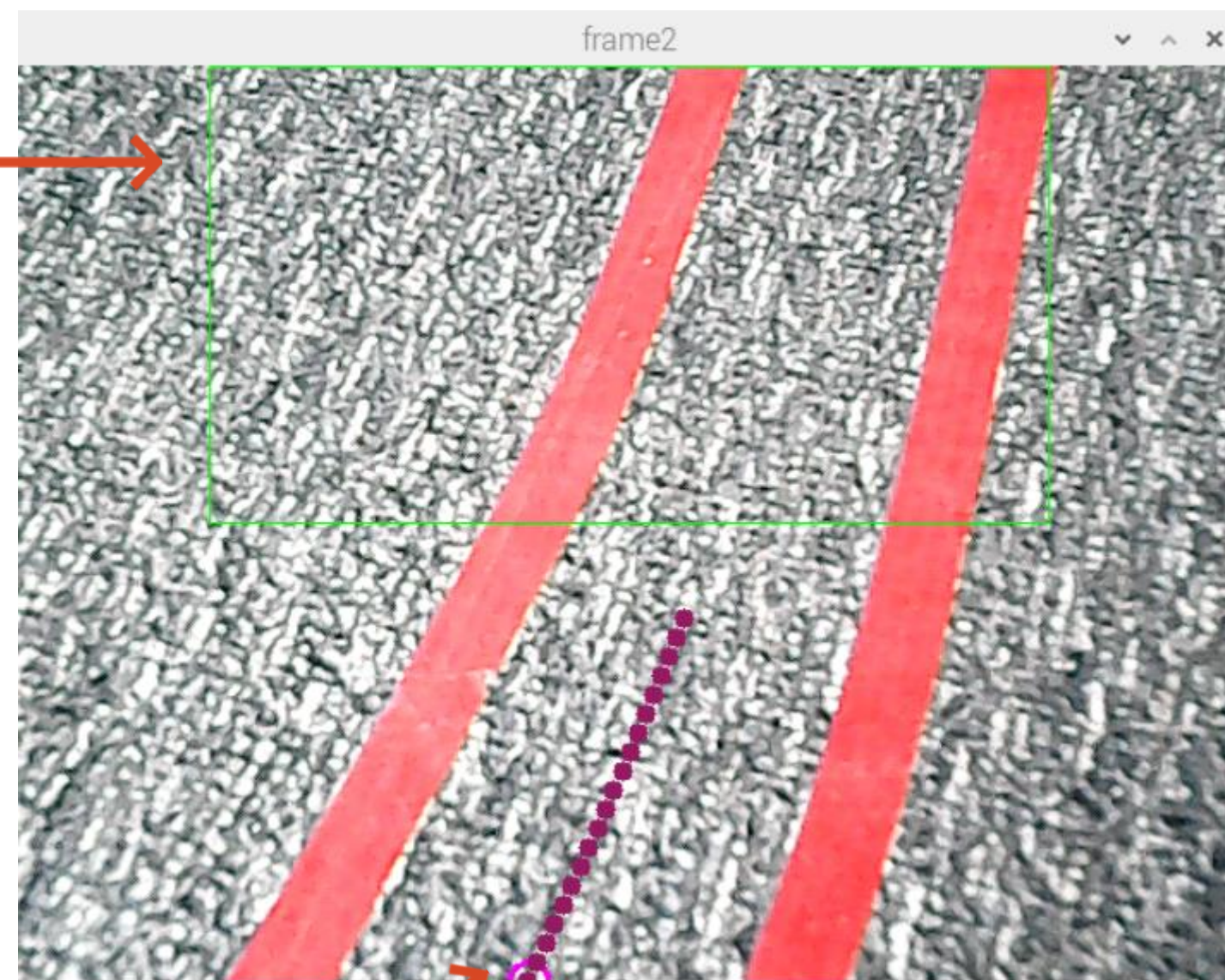
## 3.이미지 프로세싱 과정



이진이미지로 도로만 검출하여 선형회귀를 통해서 프레임에 점을 찍어 도로를 나타냅니다.

## 4.이미지 프로세싱 결과

ROI

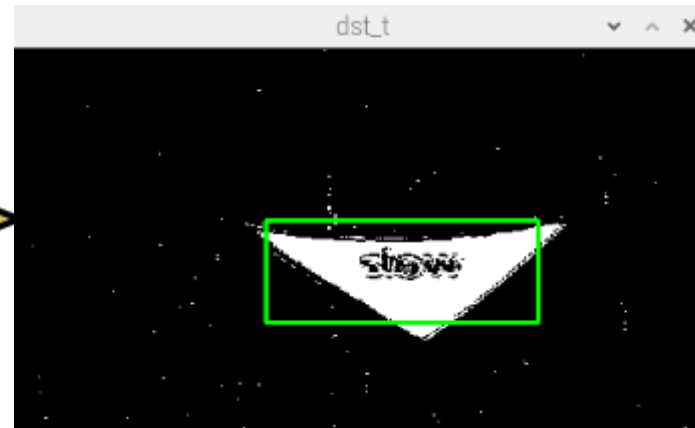
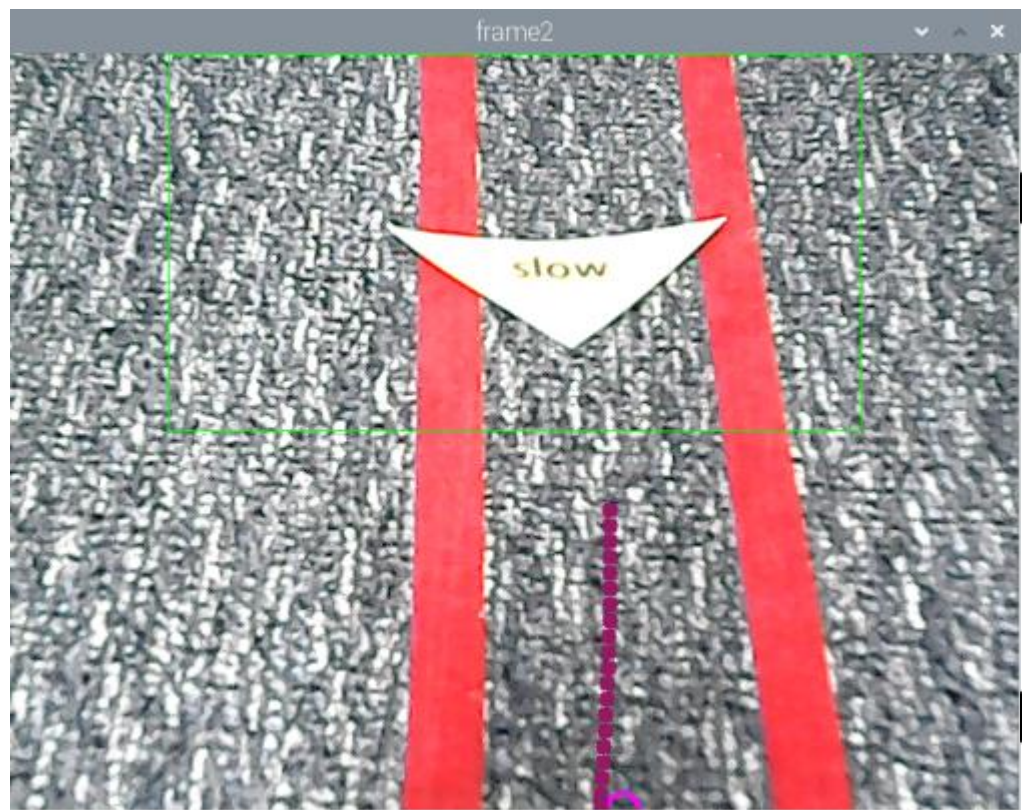


도로인지



# 이미지알고리즘:

## 5. 표지판 인지



표지판 인지 시에는 이 와 같이 박스로 표시해서  
이용자에게 알려줍니다.

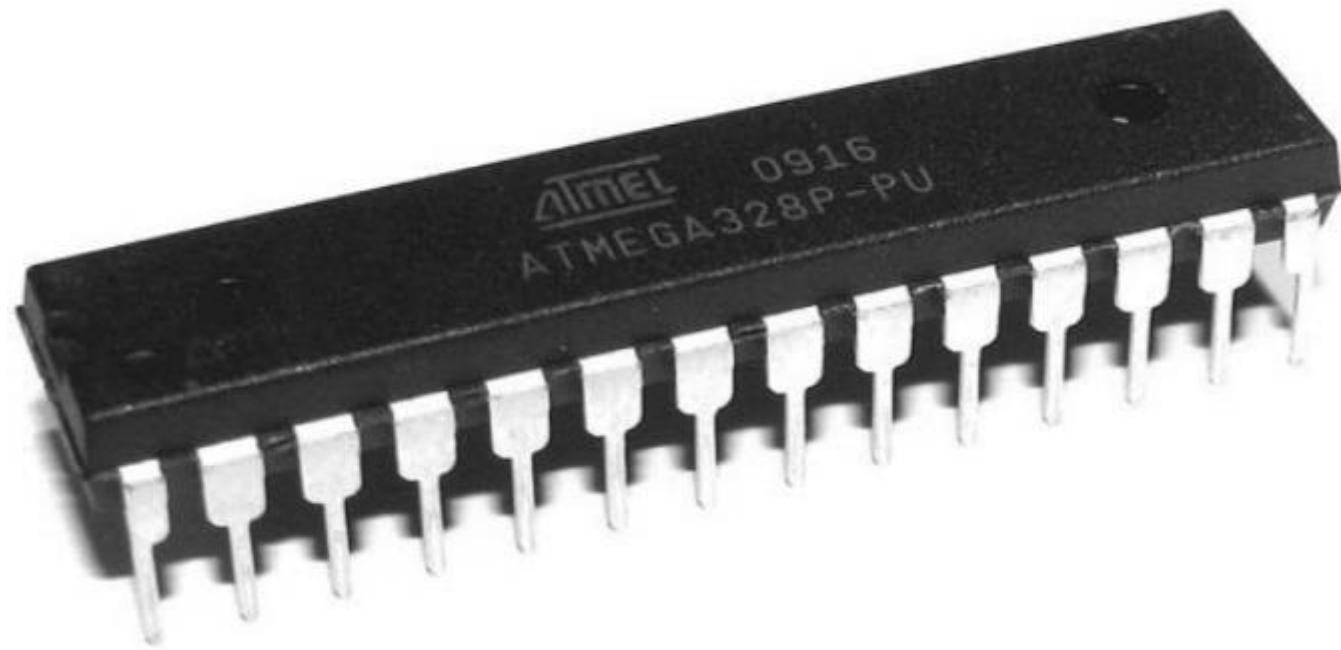
이미지 프로세싱 시간을 줄이기 위해 ROI(관심지역)를 지정했습니다.  
ROI지역안에서만 표지판 인지 작업을 실행합니다.



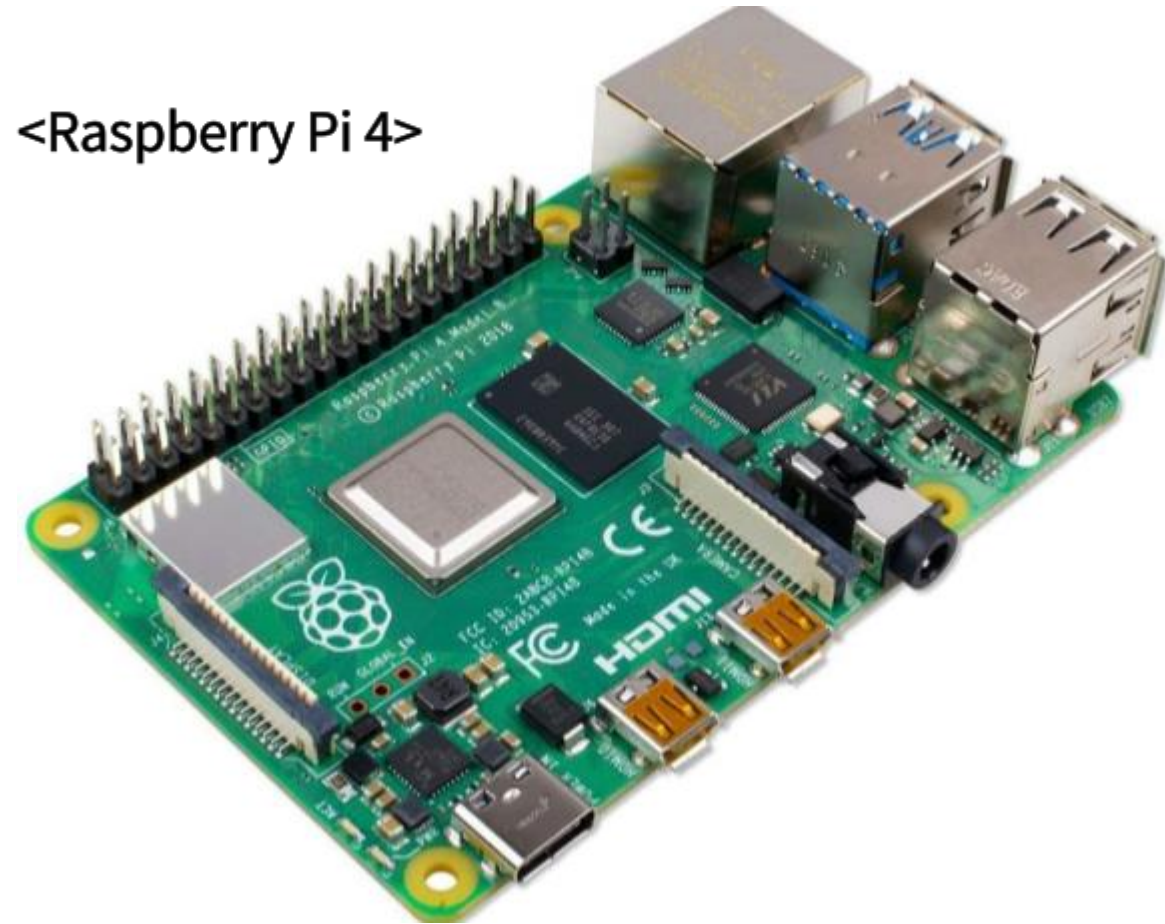
해당 프레임은 이용자의 Email로 전송합니다.

# Part List

<Atmega328p>



<Raspberry Pi 4>



- Raspberry Pi 4 에서 Open CV 를 이용하여 도로를 인식한 데이터를 UART 통신을 통해서 Atmega328p 에 데이터를 전달하여 작동부(모터)를 제어하도록 설정하였습니다.

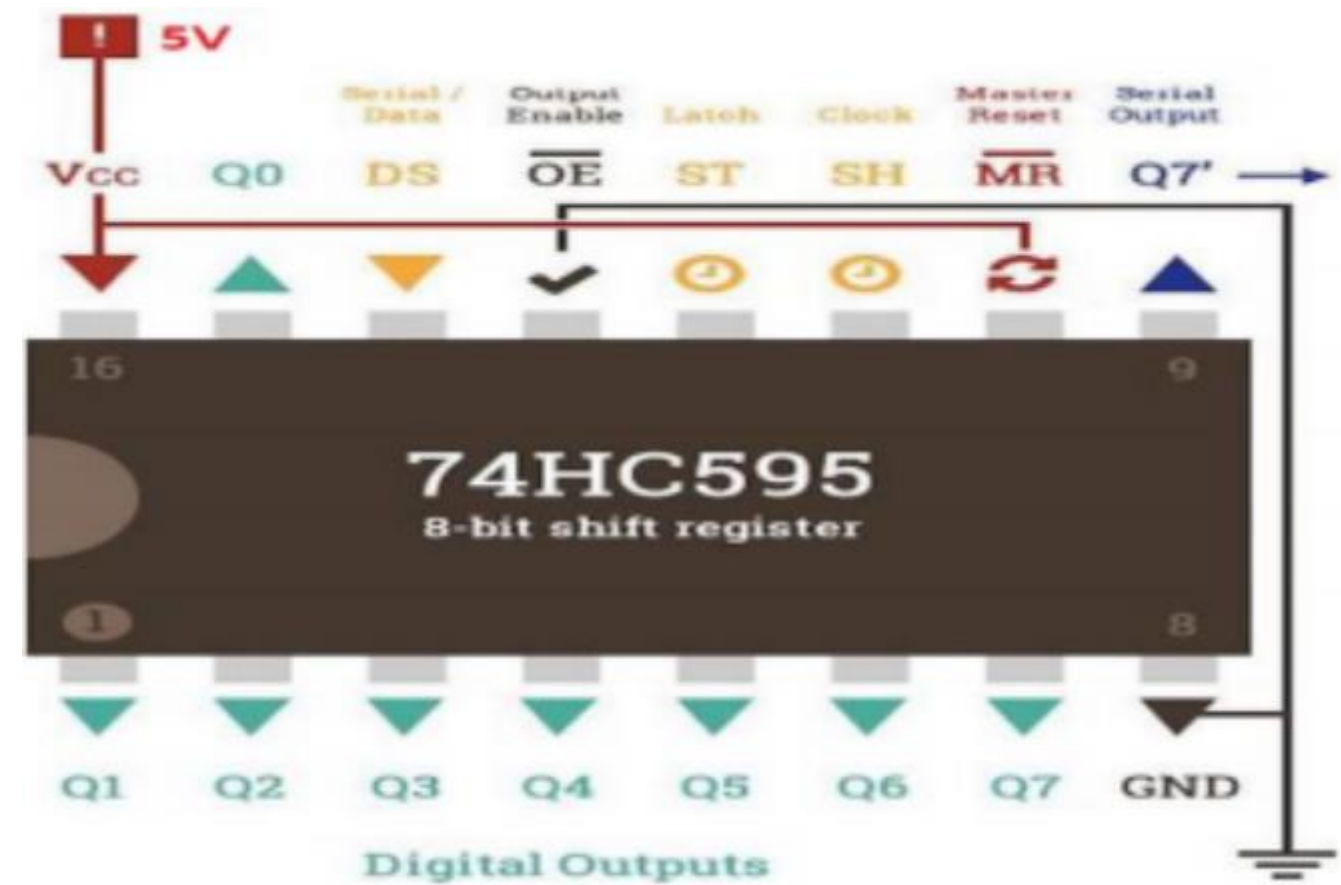


# Part List

<Shift Register 8bit (74HC595)>



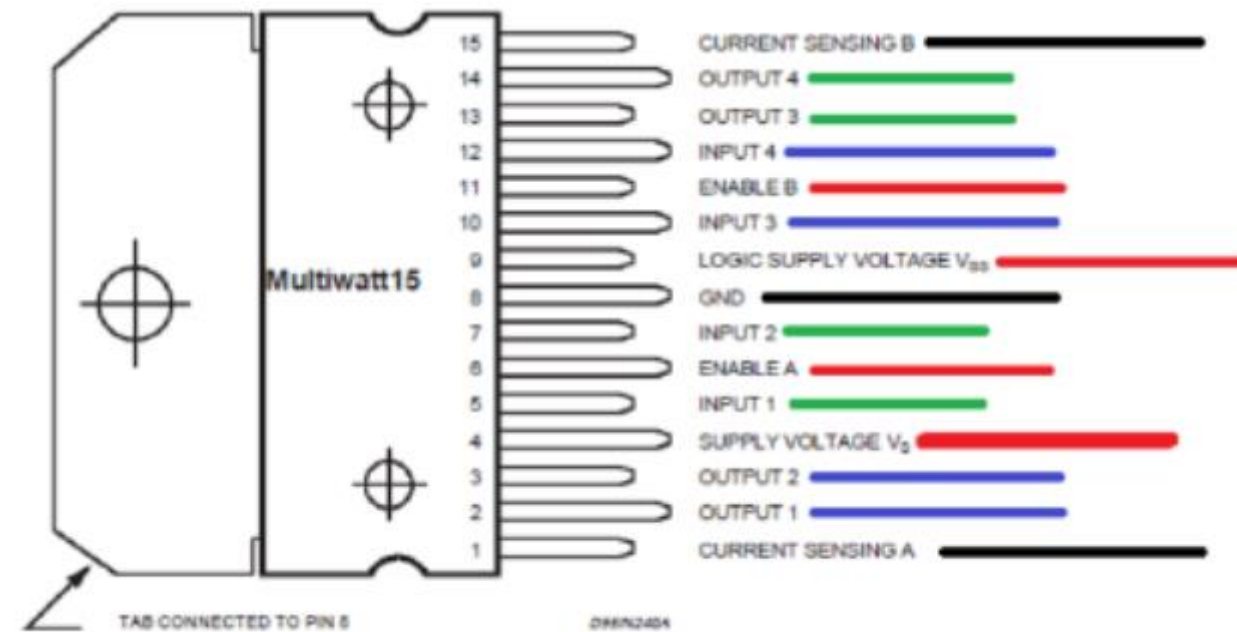
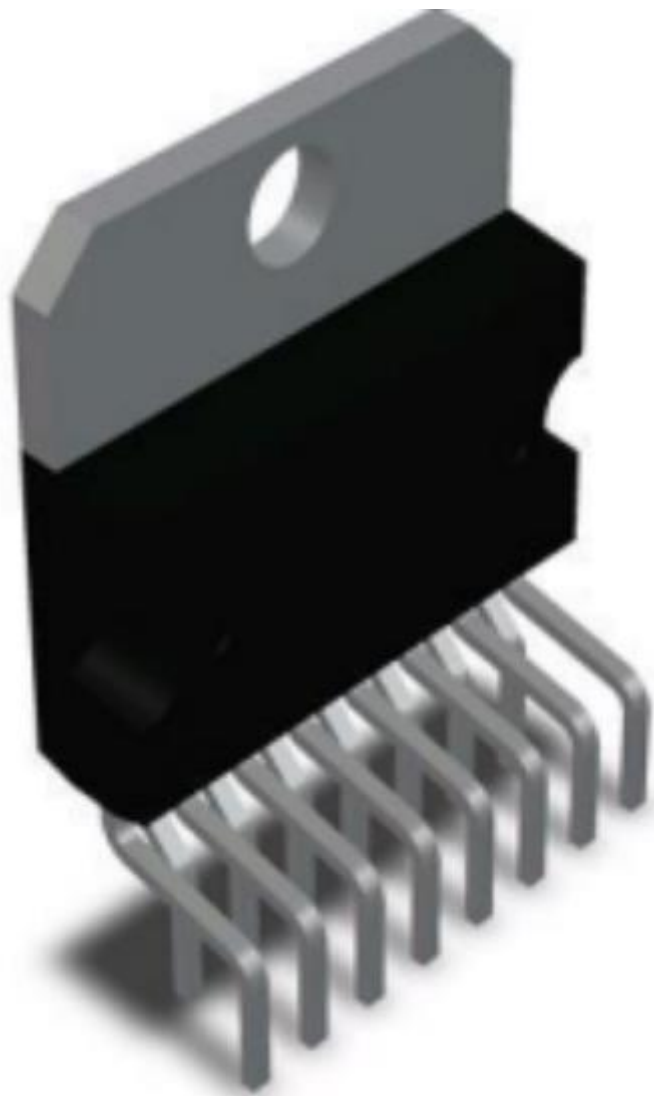
\*\*8bit SIPO (Serial-In, Parallel-Out)  
최대 8개의 디지털신호를 입력 신호 3개의 핀을 통해 'Shift register'의  
최대 8개의 디지털 신호를 제어함으로써 핀의 개수를 더 확보하도록 설계하였습니다.



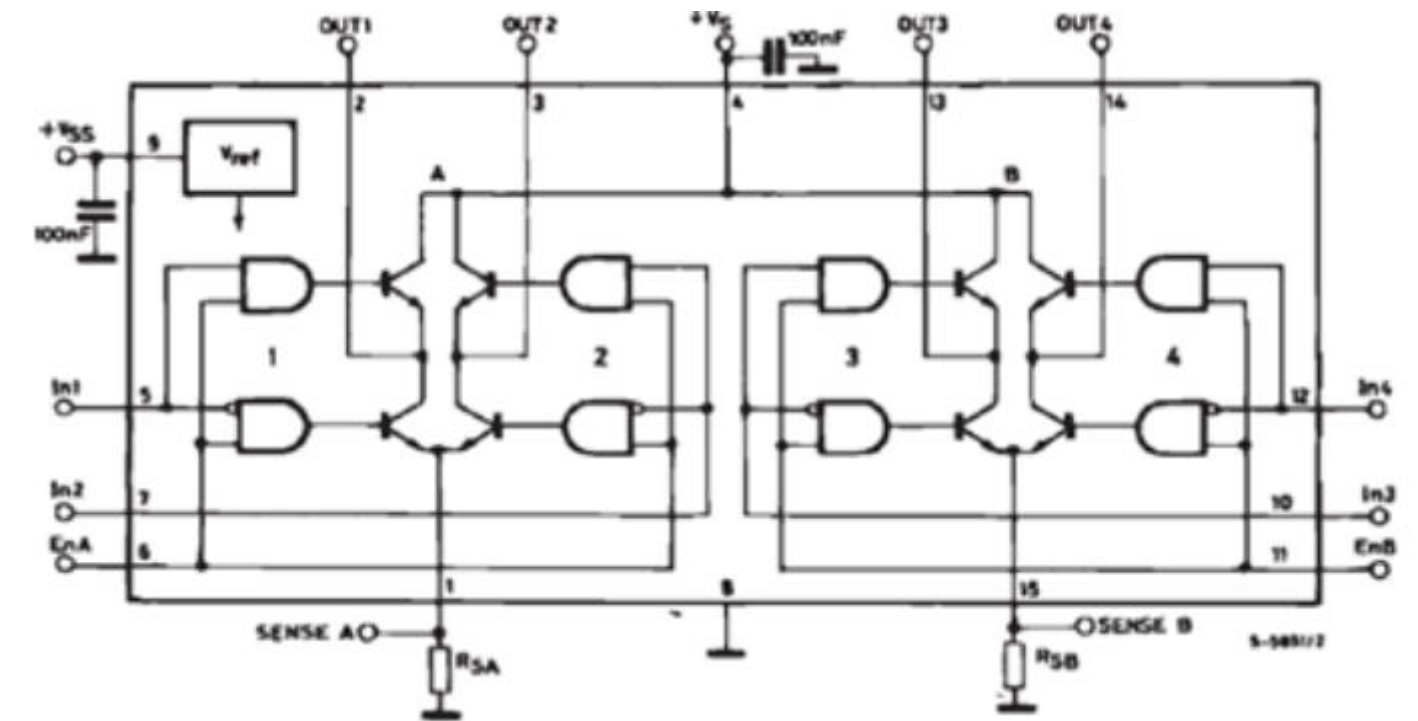
모터드라이버 INPUT1-4 핀을 연결하여 모터를 제어하기 위해 Q0~Q3에 연결하였습니다.  
SH(CLOCK), ST(LATCH), DS(DATA)핀을 통해 데이터를 전달하며  
모터의 회전 방향(정회전, 역회전)을 결정하도록 구현했습니다.



# Part List



- External Voltage(9V)
- Logic Supply Voltage(5V)
- Ground(GND)
- Digital Signal(INPUT)
- DC Motor(OUTPUT)



■ L298N을 통해 디지털 신호에 따라 DC 모터의 정/역 제어를 하였고 그에 따라 자동차의 방향을 구현했습니다.

내부에 2개의 H-BRIDGE와 ANDGATE를 통해 디지털 신호 값을 읽어 스위치형식으로 전류를 공급하여 DC 모터의 정역제어를 할 수 있습니다.  
전원 부분에 커패시터를 통해 불안정한 전류가 흐르는 것을 방지해주어 모터에 일어나는 잡음을 최소화 할 수 있습니다.

CURRENT SENSING(Analog to Digital Converter)핀은 사용하지 않아 접지로 연결하였습니다.

OUTPUT1, 2와 OUTPUT3, 4를 2개의 DC모터(Left, Right) 각각의 +단자와, -단자에 연결하였습니다.

<Full-Bridge-Motor-driver (L298N)>

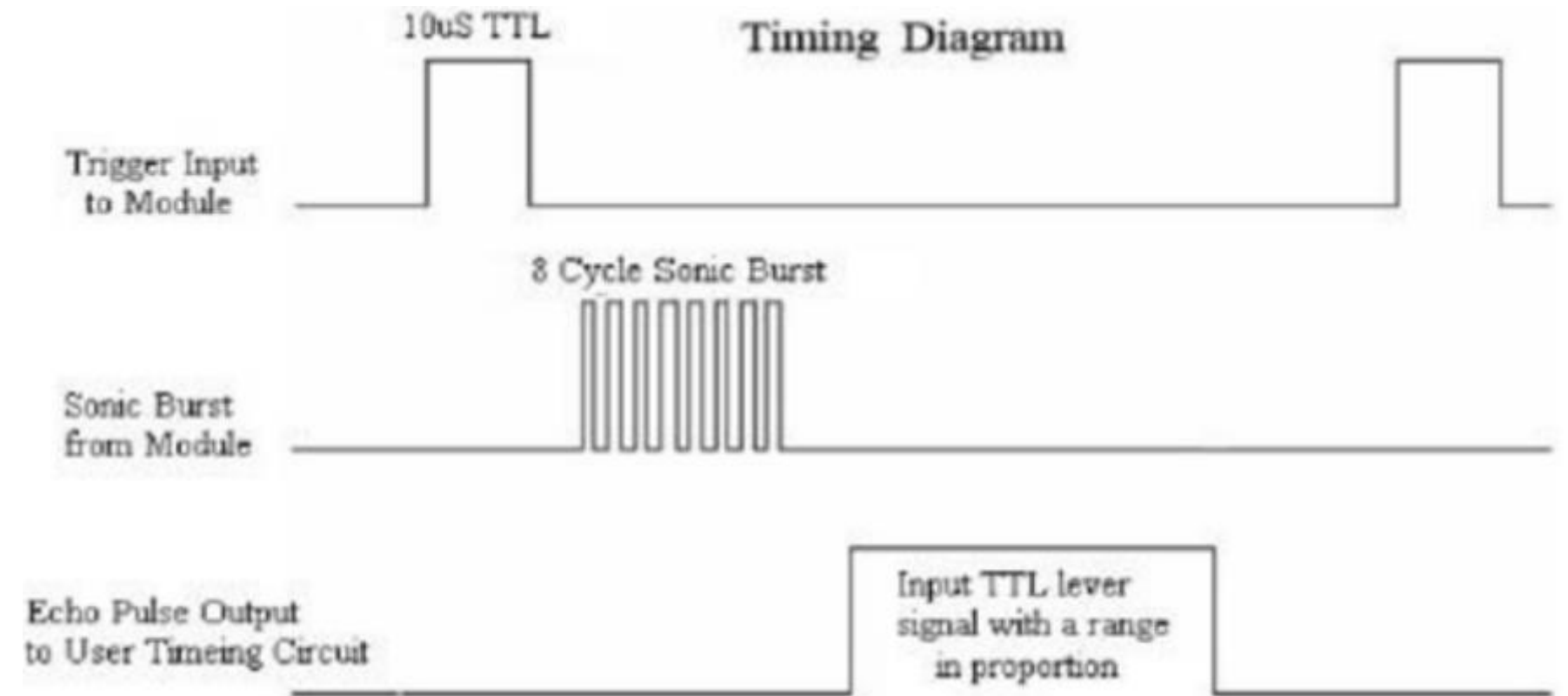
# Part List



- Power Supply: +5V DC
- Quiescent Current: <2mA
- Working current: 15mA
- Effectual Angle: <15°
- Ranging Distance: 2-400 cm
- Resolution: 0.3 cm
- Measuring Angle: 30°
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm
- Weight: approx. 10 g

## <Ultrasonic Sensor (HC-SR04)>

■ 초음파 센서를 통해 주행 중, 전방의 거리를 감지해 물체와의 충돌을 감지 합니다.



## <Ultrasonic Sensor Timing Diagram>

# Web

## 1. Flask 서버

```
pi@raspberrypi:~/final $ python fla.py
* Serving Flask app "fla" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8888/ (Press CTRL+C to quit)
```

라즈베리 파이에서 구축한 서버를 외부에서 접속할 수 있도록 포트포워딩 설정을 했고 PC와 모바일에서 접속이 가능합니다.

## 2. 데이터베이스 테이블<유저 아이디와 비밀번호>

```
MariaDB [seon]> select*from accounts;
+-----+-----+-----+-----+
| id | username | password | regdate |
| email | fromip | | |
+-----+-----+-----+-----+
| 1 | test | $2b$12$lBvHtAPEQ5m0RsBxX15fL0oLcF4oSP0S2zsdcbVq2rS2ImWQvsv/a |
| 1234@naver.com | '118.235.6.180' | 2021-05-21 01:28:25 |
+-----+-----+-----+-----+
```

데이터 베이스에 이용자의 아이디와 비밀번호를 저장합니다.

## 3. 데이터베이스 테이블<초음파 센서 거리값>

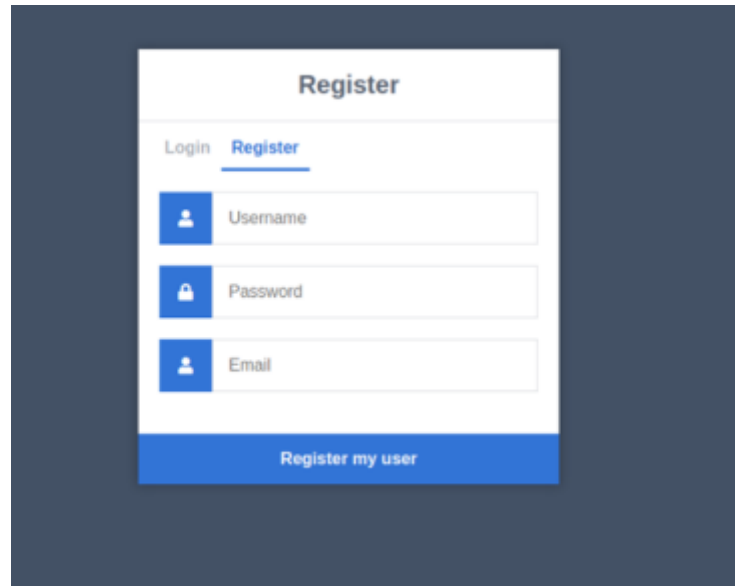
```
2021-05-27 03:12:12 | 11.49 |
2021-05-27 03:12:13 | 10.24 |
2021-05-27 03:12:13 | 7.73 |
2021-05-27 03:12:18 | 11.9 |
2021-05-27 03:12:18 | 11.07 |
2021-05-27 03:12:19 | 11.06 |
2021-05-27 03:12:19 | 12.76 |
2021-05-27 03:12:20 | 11.1 |
2021-05-27 03:21:44 | 109.19 |
2021-05-27 03:21:45 | 138.26 |
2021-05-27 03:21:45 | 10.1 |
2021-05-27 03:21:46 | 6.58 |
2021-05-27 03:21:51 | 15.82 |
2021-05-27 03:21:51 | 9.56 |
2021-05-27 03:21:56 | 11.56 |
2021-05-27 03:21:56 | 76.43 |
2021-05-27 03:21:57 | 10.34 |
2021-05-27 03:21:57 | 153.81 |
2021-05-27 03:21:58 | 997.33 |
2021-05-27 03:21:58 | 11.32 |
+-----+-----+-----+
1978 rows in set (0.024 sec)
```

라즈베리파이에 연결된 초음파센서로 측정한 거리를 데이터베이스에 저장하고 실시간으로 웹에서 보여줍니다.

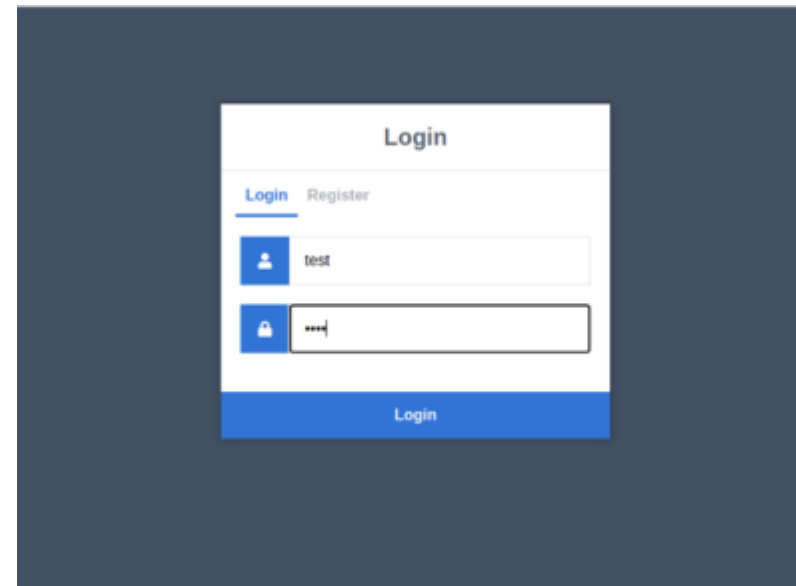


# Web

## 1. Web 회원가입

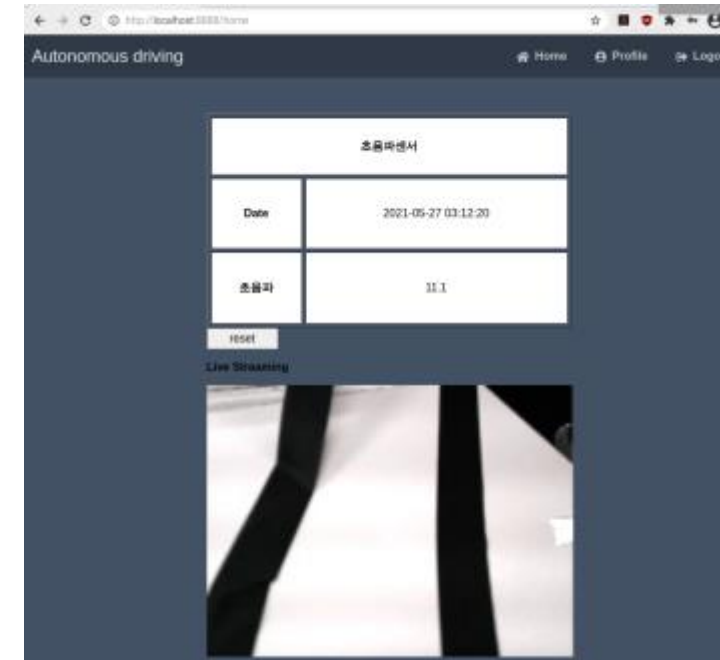
A web registration form titled "Register" with tabs for "Login" and "Register". It contains three input fields: "Username", "Password", and "Email", each with a corresponding icon (person, lock, and envelope). A blue button at the bottom says "Register my user".

## 2. Web 로그인

A web login form titled "Login" with tabs for "Login" and "Register". It contains two input fields: "Username" (with "test" entered) and "Password" (with masked characters). A blue button at the bottom says "Login".

회원가입 후 웹페이지 이용이 가능합니다. (회원가입 중복처리)  
회원가입 정보는 데이터베이스에 전송되어 테이블에 저장되고  
비밀번호는 bcrypt를 이용하여 암호화 하였습니다.  
로그인 시도 시에는 데이터베이스와 정보를 대조하여 로그인을 승인합니다.

## 3. Web 페이지



<PC환경>



<모바일 환경>

로그인 승인 시 웹페이지에서 데이터베이스에 저장된  
초음파센서 값과 웹캠을 실시간으로 확인 할 수 있습니다.

# Web

## 1. 표지판 인지 시 이메일 전송

☆ 라즈베리파이에서 보낸 메일

보낸사람 VIP <pjp9157@gmail.com>

받는사람 <tkdrif765@naver.com>

일반 첨부파일 1개 (216KB) 모두 저장

homepifinalcap41.jpg 216KB

라즈베리파이에서 사진파일을 발송하였습니다.

이미지 미리보기 (1개의 이미지가 있습니다.)



homepifinalcap41.jpg

## 2. 장애물 인지 시 이메일 전송

☆ 라즈베리파이에서 보낸 메일

보낸사람 VIP <pjp9157@gmail.com>

받는사람 <tkdrif765@naver.com>

일반 첨부파일 1개 (52KB) 모두 저장

homepifinalcap0.jpg 52KB

라즈베리파이에서 사진파일을 발송하였습니다.

이미지 미리보기 (1개의 이미지가 있습니다.)



homepifinalcap0.jpg

자율 주행 시 표지판 인지 또는 초음파 센서로 장애물을 일정 거리에서 인지할 때 웹캠은 해당 표지판이나 장애물을 캡처하여 해당 프레임을 JPG파일로 라즈베리파이에 저장하고 그 저장된 파일을 사용자의 Email로 전송합니다. 이용자는 Email에서 왼쪽 사진과 같이 사진을 확인할 수 있습니다.

# 핵심 코드 :

## Python <차선인식>

```
def find_Dir_and_rec_load(frame):
    total = 0 # 도로인자(픽셀)의 번호의 합
    count = 1 # 도로인자(픽셀) 갯수
    center_x = [] # 도로 중심의 X좌표의 리스트
    center_y = [] # 도로 중심의 y좌표의 리스트
    HSV_frame = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV) # RGB -> HSV 이미지 변환
    lower_hsv_1 = cv2.inRange(HSV_frame, (0, 100, 100), (5, 255, 255)) # 최소 색검출
    upper_hsv_2 = cv2.inRange(HSV_frame, (169, 100, 100), (180, 255, 255)) # 최대 색검출
    dst = cv2.addWeighted(lower_hsv_1, 1.0, upper_hsv_2, 1.0, 0.0) #lower_hsv_1 + upper_hsv_2
    for i in range(640): # 카메라 이미지 넓이(640) x 높이(480)
        if dst.item(479,i) != 0: # 이진 이미지 픽셀(y좌표, x좌표)이 0(black)이 아니면
            total += i # i의 총합.
            count += 1 # 나누는 값.
    x_value1 = round(total/count) # 평균값
    Dir = x_value1 - 320 # 조향할 방향 설정 (차선 중앙의 평균값이 카메라의 정중앙과 가까울수록 Dir=0)
    cv2.circle(frame,(x_value1,479),10,(255,0,255),2,None,0) # 차선의 중심(평균값)에 원을 그림
    total = 0 # 초기화
    count = 1 # 초기화
    for w in range(10): # 선형회귀에서 사용할 도로의 10개의 중점을 찾는다.
        for i in range(640):
            if dst.item(w*45,i) != 0 # 각 높이(45씩 증가)에서 차선의 평균값을 구함
                total += i # i의 총합.
                count += 1 # 나누는 값.
```

```
        center_x.append(round(total/count)) # center_x [list]에 x축 평균값을 추가
        center_y.append(w*45) # center_y [list]에 y축 위치를 추가
    total = 0 # 초기화
    count = 1 # 초기화
    fit = np.polyfit(center_y,center_x, 2) # 2차 함수로 선형회귀를 구함
    ploty = np.linspace(0, dst.shape[0] - 1, dst.shape[0])
    fitx = fit[0] * ploty ** 2 + fit[1] * ploty + fit[2]
    tx = np.trunc(fitx)
    pts = np.array([np.transpose(np.vstack([tx, ploty]))])
    for i in range(20): #인정한 도로를 나타내는 20개의 점 찍기
        po_x,po_y = pts[0,479-i*10,0],pts[0,479-i*10,1]
        po_xx = int(po_x)
        po_yy = int(po_y)
        cv2.line(frame,(po_xx,po_yy),(po_xx,po_yy),(100,25,150),10,None,0)
    cv2.imshow('frame', dst)
    center_x = []
    center_y = []
    return Dir
```



# 핵심 코드 :

## C언어 AVR <UART통신>

```
void UART_Init(void) { // UART 초기화
    UCSR0A = 2; // 2배속 모드 설정.
    UCSR0B = (1<<RXCIE0) | (1<<RXEN0) | (1<<TXEN0); // RX INT enable, RX & TX Enable
    UCSR0C = (1<<UCSZ01) | (1<<UCSZ00); // 8bit (no parity)
    UBRR0H = 0; UBRR0L = 207; // 9600 보드레이트 설정
}

void UART_TX_CH(char c) {
    while(!(UCSR0A & (1<<UDRE0))); // 정보를 받을 수 있는 상태면 while문 탈출
    UDR0 = c; // 버퍼(데이터를 전송할 정보를 담는 버퍼)에 문자 송신.
}

void UART_TX_STR(char *s) {
    for(int i=0; i<strlen(s); i++) { // 문자열 길이 만큼 반복
        UART_TX_CH(*(s+i)); // 주소 값 이동하며, 문자 입력.
    }
    memset(s, 0, sizeof(*s) * strlen(s)); // 문자열 NULL 초기화
}
```

```
// UART Receive Interrupt
ISR(USART_RX_vect) {
    int i;
    char rxdata; // 문자 받는 변수
    rxdata = UDR0; // 문자 수신.
    if(!rx_flag) { // 'rx_flag' 비활성화 시 수행
        if(rxdata == STX) {} // 변수가 '_' 일 때
        else if(rxdata == ETX) { // 변수가 '/' 일 때
            for(i=0; (i<rx_cnt) && (i<15); i++) {
                tx_buf[i] = rx_buf[i]; // 버퍼(rx)의 값을 버퍼(tx)에 저장
                rx_buf[i] = NULL; // '버퍼(rx)'를 'NULL'로 초기화
            }
            rx_cnt = 0; // 카운트 변수(rx) 초기화
            rx_flag = 1; // 'rx_flag' 활성화
        }
    }
    else {
        rx_buf[rx_cnt] = rxdata; // 버퍼(rx)에 'rxdata' 입력: (l,□,□,□, r,□,□,□)
        rx_cnt++; // 카운트(rx)
    }
}
```

# 핵심 코드



## Python<Email전송>

```
def Email():
    global pic_num
    imgRows="/home/pi/final/cap{0}.jpg".format(pic_num)
    if imgRows is not None:
        fromEmail='pjp9157@gmail.com' //보내는 이메일
        toEmail='tkdrif765@naver.com' //받는 이메일
        smtp=smtplib.SMTP('smtp.gmail.com',587)
        smtp.starttls()
        smtp.login('이메일','비밀번호')
        msg=MIMEMultipart()
        message=MIMEText('라즈베리파이에서 사진파일을 발송하였습니다.',_charset='utf-8')
        msg['Subject']= '라즈베리파이에서 보낸 메일'
        msg['from']= fromEmail
        msg['To']=toEmail
        msg.attach(message)
        with open(imgRows,'rb') as test:
            etcPart= MIMEApplication(test.read())
            etcPart.add_header('Content-Disposition','attachment', filename=imgRows)
            msg.attach(etcPart)
            smtp.sendmail(fromEmail,toEmail,msg.as_string())
            smtp.quit()
            pic_num += 1
```

## C언어 AVR <PWM 제어>

```
void PWM_init() { // PWM 초기화
    DDRD |= (1 << PD5); // PD5 pin output활성화
    DDRD |= (1 << PD6); // PD6 pin 활성화
    TCCR0A |= (1 << WGM01) | (1 << WGM00); // 파형 설정
    TCCR0A |= (1 << COM0A1); // 비반전 모드
    TCCR0A |= (1 << COM0B1); // 비반전 모드
    TCCR0B |= (1 << CS02); // 64 분주비 설정
}

void PWM_PD5(int duty) { // PD5 듀티비 설정.
    OCR0B = duty; // OCR0B: 듀티비 저장 레지스터
}

void PWM_PD6(int duty) { // PD6 듀티비 설정.
    OCR0A = duty; // OCR0A: 듀티비 저장 레지스터
}
```