Capstone Design 설계 계획서

<table>
<tr><td rowspan="4">팀장</td><td>성 명</td><td colspan="3">권 형 준</td></tr>
<tr><td>소 속</td><td colspan="3">동아대학교 공과대학 전자공학과</td></tr>
<tr><td>연 락 처</td><td colspan="3">주소: 부산광역시 해운대구 중동2로34번길 29, 103동1002호
E-mail: cccc7991@gmail.com
H/P: 01045617991</td></tr>
<tr><td>학 년</td><td>4</td><td>학 번</td><td>2023132</td></tr>
<tr><td>작품명칭</td><td colspan="4">STGC(태양광 추적 발전 충전기)</td></tr>
<tr><td>개발기간</td><td colspan="4">2025년 9월 19일 ~ 2025년 11월 29일</td></tr>
<tr><td rowspan="8">참여학생</td><td>이름</td><td>학번</td><td>휴대폰</td><td>전자메일</td></tr>
<tr><td>송승훈</td><td>2023015</td><td>010-7563-1770</td><td>thdskfen@naver.com</td></tr>
<tr><td>김덕현</td><td>1922984</td><td>010-9956-0362</td><td>rlaejrgus926@naver.com</td></tr>
<tr><td>권형준</td><td>2023132</td><td>010-4561-7991</td><td>cccc7991@gmail.com</td></tr>
<tr><td>한태민</td><td>2022523</td><td>010-4196-9859</td><td>teamin9859@naver.com</td></tr>
<tr><td>백상철</td><td>2023408</td><td>010-3693-0545</td><td>a34105451@gmail.com</td></tr>
<tr><td>김지훈</td><td>1923438</td><td>010-6508-3220</td><td>kjhkkakdj@naver.com</td></tr>
</table>

본인은 Capstone Design 프로그램 운영에 따른 설계 계획서를 첨부와 같이 제출합니다.

2025년 9월 19일

동아대학교 전자공학과 학과장 귀하

Capstone Design 설계 계획서

1. 연구배경 및 목적

석유, 석탄, 천연가스 등 화석연료의 매장량이 얼마 남지 않음과 온실가스로 인한 지구 온난화와 기후변화 문제의 심각성이 증가하고 있다. 또 원자력 발전의 경우, 안정성과 폐기물 처리에서 한계점이 명확하여 신재생에너지 자원의 필요성이 더욱 중요해졌다.

STGC는 많은 에너지를 발전하기보다는 일반 가정용에서부터 재생에너지를 사용해가며 석유, 석탄, 등의 화석연료 사용량을 감소시킨다는 취지에서 개발하게 되었다. 태양광 패널이 태양의 고도에 따라 빛을 추적하여 에너지를 수집하며 이를 원격으로도 현재 충전량과 발전량 등을 확인할 수 있게 하여 편리성도 추가할 예정이다.

본 프로젝트는 소형·저비용 태양광 추적 발전 시스템을 개발하여, 일반 가정이나 소규모 환경에서도 효율적으로 태양광을 활용할 수 있도록 하는 것을 목표로 한다. 그 결과 가정에서부터 시작한 재생에너지 사용은 한 개인이 했을 때는 그 효과가 미약할 수도 있지만 많은 사람들이 점차 사용하며 화석연료를 줄여간다면 세계적으로 아주 큰 효과가 창출될 것이다.

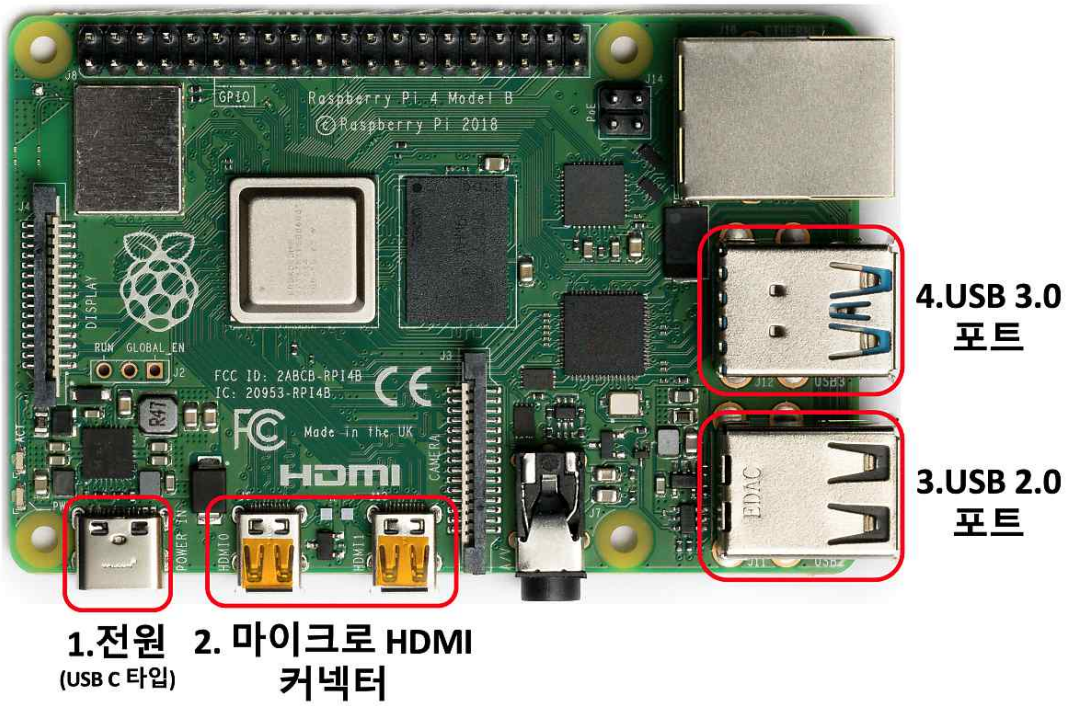
2. 설계 주요내용

*주요 부품 구성

2-1.【라즈베리파이】

■정의: 라즈베리 파이(Raspberry Pi)는 영국의 라즈베리 파이 재단이라는 곳에서 출시된 ARM 기반의 초소형 보드 컴퓨터를 말하며 초보자들을 위한 교육용 프로젝트의 일환으로 개발된 저가형 싱글보드 컴퓨터이다.

-본 프로젝트는 Raspberry Pi 4를 중앙 제어기로 사용한다. Raspberry Pi는 센서 제어, 데이터 로깅 및 모터 제어를 통합 수행할 수 있는 싱글보드 컴퓨터(SBC)이다.



[출처] https://wikidocs.net/blog/@yulian/952/

분류	라즈베리파이4(4GB)	라즈레비파이4+ESP32
구조	RPi4 하나가 센서 읽기 + 태양 위치 계산 + 모터 제어 + 웹서버 모두 담당	RPi4 = 연산·웹서버 ESP32 = 모터·센서 실시간 제어
장점	-하드웨어 단순 (보드 1개) - 개발환경 풍부 (Python, C++, 라이브러리 다양) - 코드 관리 용이 (한 시스템 안에서 해결)	- 모터 제어 안정적 (ESP32가 마이크로초 단위 타이밍 보장) - 모터 노이즈/실패 → RPi4 영향 적음 - 전력 효율↑ (ESP32 슬립/모터 전류 컷 가능) - 확장성 좋음 (센서 추가, IoT 통신 분리)
단점	- 리눅스라 실시간 제어 불안정 (펄스 지터) - 모터/센서 노이즈가 시스템 다운 유발 가능 - 전력소모 큼 (RPi4 3~6W) - 강풍/긴급상황 대응 속도 낮음	- 하드웨어 1개 더 필요 (비용·배선↑) - 프로토콜 설계 필요 (UART/MQTT 등) - 코드가 2개(라즈베리+ESP32)라 관리 번거로움
전력	24h 구동 시 약 60~70 Wh/일	RPi4(간헐 기동) + ESP32 슬립 가능 → 실사용 약 30~50 Wh/일

*두 대를 사용하는 것이 캡스톤 최종 구현에는 더 안정적이고 효율적이라고 생각합니다.
왜냐하면, 라즈베리파이4는 연산,UI · SP32는 실시간제어로 역할 분담이 확실해지고 전력도 아낄 수 있습니다.
(전기를 발전하는 작품이기 때문에 전력을 조금이라도 아끼는게 중요)

■ 라즈베리파이4+GPS모듈+포토다이오드(보조)+INA219 등 센서 제어

1) 센서 데이터 처리

-GPS 모듈을 통해 수집된 위도·경도 및 시간을 바탕으로 태양의 고도(altitude)와 방위각(azimuth)을 계산한다.
-온습도 센서, 포토다이오드 등 센서로부터 데이터를 수집하여 날씨 상황(맑음/흐림)에 따른 동작 모드 전환을 수행한다.
-INA219와 같은 전력 측정 센서로부터 패널 전압, 전류, 전력, 누적 발전량을 받아 저장·가공한다.

2) 모터 제어

-라즈베리파이는 PWM 신호를 생성하여 2개의 서보모터(MG995, MG996R)를 제어한다. 이를 통해 태양의 위치에 맞춰 태양광 패널을 좌우(X축), 상하(Y축)으로 회전시킬 수 있다.
- pigpio, PCA9685와 같은 라이브러리/보드를 활용하여 안정적인 제어 주기를 보장한다.

3) 시스템 통합 관리

-라즈베리파이는 단순히 모터를 구동하는 역할을 넘어, 데이터 수집-처리-제어-전송을 모두 아우르는 중앙 제어 허브로 작동한다.
-소프트웨어 업데이트, 데이터 로깅, 장애 진단(예: 과전류, 저전압 알람) 등의 기능 확장이 가능하다..

▶라즈베리파이에 GPS 모듈 부착 → 위도·경도·UTC 시간 수집

▶수집된 좌표 및 시간 정보를 AI 알고리즘(태양 위치 공식/모델 기반)으로 분석

▶태양의 고도(altitude)와 방위각(azimuth) 계산

▶계산된 태양 방향에 맞추어 서보 모터(X, Y축) 구동 → 태양광 패널 각도 제어

-날씨에 관계없이 태양의 이론적 위치를 추적이 가능하고, 추적 정밀도가 높다. 장시간 운용이 가능하고, 수집된 데이터로 인해서 AI 학습을 접목하여 미래 예측형 제어 확장이 가능하다.

2-2.【태양광 판넬 구동】

-프로젝트에서 제작하는 태양광 추적 장치는 소형 태양광 페널과 2축 서보모터를 이용해서 XY축 회전을 구현한다.

1) 태양광 패널

태양광 패널: 145 × 145 mm (정격 3 W) 소형 패널을 사용합니다. 패널은 3D 프린팅한 U자형 브라켓에 장착되며, 서보모터 2축(MG996R — X축, MG995 — Y축)으로 회전한다.

2) 서보 모터(2개)

-X축(좌우 회전축): 하부 서보모터(MG996R)
지지대 하단에 결합되어 패널 전체를 좌우로 회전시키는 역할.
상대적으로 큰 토크가 요구되므로 고출력 서보 사용한다.

-Y축(상하 회전축): 측면 서보모터(MG995)
지지대 옆면에 연결되어 패널 각도를 상하로 꺾는다.

3) 지지대

-지지대는 U,L,C자형 브레킷구조가 있지만 두 개의 곡선이 맞물려 U자 형태를 이루는 U자 브레킷 구조로 선택하여 3D프린터로 설계하였다.
(U자형 브레킷구조는 두 반원형 곡선이 맞물려 안정적으로 힘을 분산 가능)

2-3.【센서】

1) GPS 모듈
-위도·경도·UTC 획득 → 태양의 고도/방위각 계산

2) 포토다이오드(조도센서)
-보조 센서, 미세보정용(중앙 막대 그림자 방식)

3) 온습도센서
-흐림/우천 판별 → 보호 모드 전환

4) 전류, 전압 측정 센서(INA219)
-PV 전압·전류 측정 → 순간전력 및 누적 발전량 산정

5) 배터리 연료 게이지 센서
-단순 전압 측정이 아닌 잔량(State of Charge, %)을 직접 제공하여 배터리 방전 정도를 보다 정확하게 파악할 수 있다. 이를 통해 저전압이나 과방전으로 인한 시스템 정지나 손상을 예방할 수 있다.

▶보조적으로, IMU 센서(MPU6050 등)를 사용하면 태양광 패널의 기울기(roll, pitch)를 측정할 수 있다. 이를 GPS 기반 계산값과 비교하여 추적 오차를 보정하거나 기구적인 제어 정확도를 향상시키는데 활용 가능하다.
또한 조도 센서(BH1750)를 추가하면 주변 광량(lux)을 수치화할 수 있어, 발전량 대비 환경 광량을 비교하여 태양광 패널의 효율성을 분석하는 자료로 사용할 수 있다.

2-4.【안전대책】

1) 퓨즈 (1차 보호)
-전류가 비정상적으로 오래 흘렀을 때 물리적으로 끊어준다.
-일회성이고 반응속도가 느려 순간적인 스파이크는 못 잡는다.

2) 릴레이 또는 MOSFET (2차 보호)
-센서(INA219 등)로 전압/전류 감지 → 라즈베리파이 판단 임계값 초과 시 릴레이 코일 차단 또는 MOSFET 게이트 OFF → 즉시 전류를 끊어주는 '능동형 차단기'
-소프트웨어로 제어 가능, 반복 사용 가능

▶퓨즈는 구조가 단순하고 확실히 회로를 보호할 수 있으나, 열에 의한 차단 방식이므로 (금속 필라멘트가 과전류 → 발열 → 녹음 과정을 거쳐야 끊어짐) 반응 속도는 전자식 보호 소자에 비해 느리다. 따라서 본 시스템에서는 퓨즈를 1차 보호 장치로 사용하고, 과전류 감지 후 라즈베리파이 제어를 통한 릴레이 또는 MOSFET 차단을 2차 보호로 병행하는 이중 보호 구조를 채택하였다.

*세부 동작 원리

2-5.【태양광 추적 시스템】

■ 태양광 추적 시스템 구성과 동작 구조

①GPS 모듈

②포토다이오드(보조 센서)

③서보모터(X축, Y축)

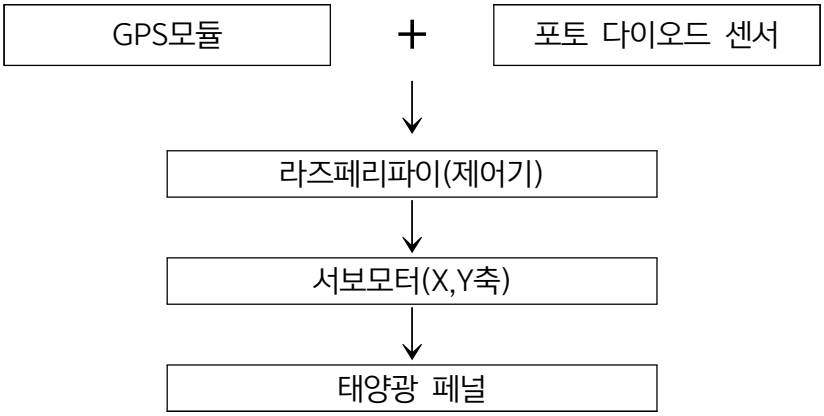
④제어부

① GPS 모듈에서 수집한 위도, 경도, UTC 시간을 바탕으로 태양의 고도(altitude)와 방위각(azimuth)을 계산하여 패널이 태양을 향하도록 합니다. 날씨와 상관없이 이론적인 태양 위치를 추적할 수 있다는 장점이 있다.

② 포토다이오드는 보조 센서로 사용되어 실제 빛의 세기를 측정한다. GPS 기반 계산으로 추적 중에 오차가 발생하거나 구름·환경 변화가 생길 경우, 포토다이오드 신호를 이용해 미세 보정을 수행한다.

③ 서보모터는 2축 구조(X축: MG996R, Y축: MG995)로, 라즈베리파이 제어 신호에 따라 태양광 패널을 좌우·상하 방향으로 회전시킨다.

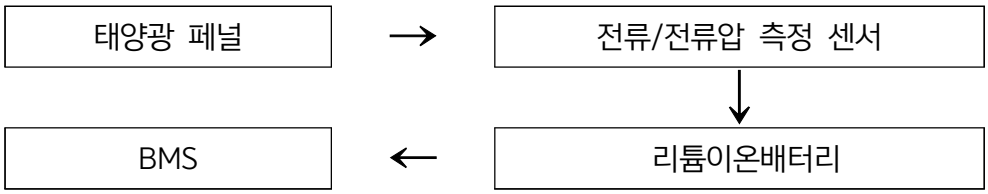
④ 제어부(라즈베리파이)는 GPS·포토다이오드 데이터를 처리하고, 온습도·전류/전압 센서와 함께 전체 시스템을 관리한다. 필요 시 날씨 상태에 따라 동작 모드를 전환하거나 시스템을 Off 상태로 전환한다.



위 그림은 라즈베리파이 기반 태양광 추적 시스템 제어 블록도이다.

1. GPS 모듈: 위도·경도·시간 데이터를 통해 태양 위치 계산
2. 포토다이오드: 실제 빛의 세기를 감지하여 GPS 기반 추적값 보정
3. 라즈베리파이(제어기): GPS+센서 데이터를 처리 → 서보모터 제어 신호 출력
4. 서보모터(X, Y축): 제어 신호에 따라 패널 회전
5. 태양광 패널: 태양을 정면으로 향해 발전 효율 극대화

2-6.【발전 및 충전 시스템】



1. 태양광 패널(145×145mm, 3W)

-외부 빛을 전기로 변환하는 1차 발전 장치.

-패널은 소형이지만 서보모터 기반 추적을 통해 발전 효율을 높일 수 있다.

2. 전류/전압 측정 센서(INA219)

-태양광 패널의 전류(A)와 전압(V)을 측정하여 순간 전력(W)을 계산한다.

-전력 데이터를 시간에 따라 누적해 전체 발전량(Wh)을 산출할 수 있다.

-측정값을 기반으로 과전압, 과전류 상황을 감지할 수 있다.

3. 리튬이온 배터리

-태양광 패널에서 생산된 전력을 저장하는 장치

-가격이 저렴하고 구하기 쉬우며, 소형화·모듈화에 유리

-일반적인 18650 셀(3.6V, 3000mAh)을 사용

4. BMS(Battery Management System)

-셀 밸런싱, 과충전/과방전 보호, 과전류 차단 기능

-배터리 수명 연장과 안전 확보

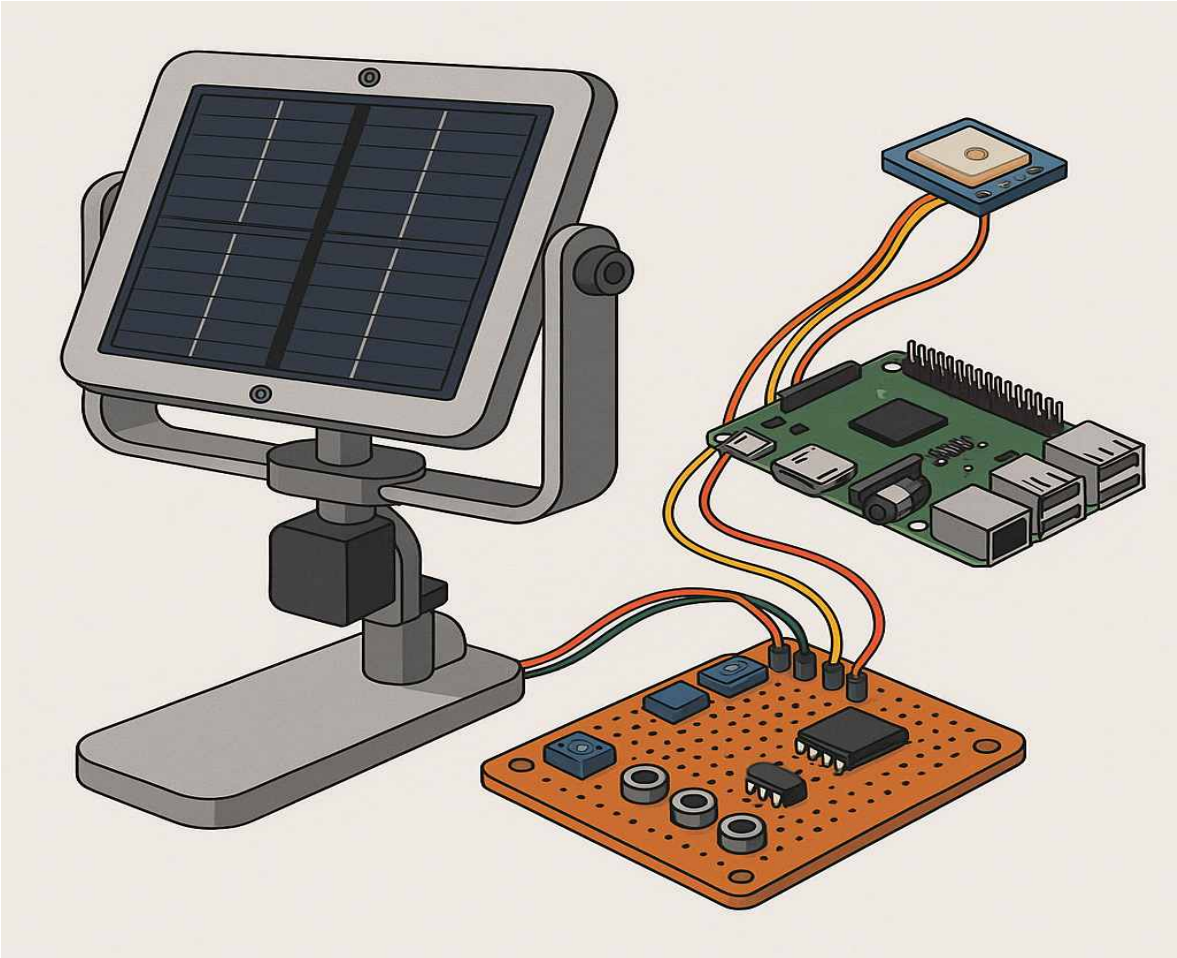
2-7.【데이터 로깅 및 모니터링】

태양광 발전 시스템의 데이터 로깅 및 모니터링 흐름을 나타낸 블록도이다.

INA219 센서를 통해 측정한 전류·전압 값을 라즈베리파이에서 처리한다. 발전 전력(W), 누적 발전량(Wh)을 계산하여 단순 기록이 가능하다. 필요 시 파일(CSV) 형태로 저장해 발전 효율 분석에 활용할 수 있다. 저장한 데이터는 Flask Djang 기반의 웹 서버를 통해 스마트폰 및 PC에서 실시간으로 확인할 수 있다.

3. 설계 방법

3-0.【전체 모습 초안】



3-1.【라즈베리파이 기반 제어】

1. 제어 시스템 개요

본 시스템의 핵심 제어부는 라즈베리파이(Raspberry Pi)로 구성된다. 라즈베리파이는 리눅스 기반의 싱글 보드 컴퓨터(SBC)로, GPIO(범용 입출력) 포트를 직접 제어할 수 있는 강력한 기능을 제공한다. 주요 개발 언어는 Python을 사용하였으며, 이를 통해 센서 데이터 수집과 서보모터의 정밀 제어를 구현하였다.

2. 주요 제어 라이브러리

시스템 구동을 위해 사용된 핵심 Python 라이브러리는 다음과 같다.

RPi.GPIO 또는 pigpio: 라즈베리파이의 GPIO 핀을 제어하는 라이브러리. 서보 모터 구동에 필요한 PWM(Pulse Width Modulation, 펄스 폭 변조) 신호를 생성하는 데 사용된다.

adafruit-circuitpython-ina219: INA219 전류·전압 센서 모듈과의 I2C 통신 및 데이터 수집을 담당한다.

time, csv: 측정된 데이터를 타임스탬프와 함께 CSV 파일 형태로 기록하고, 이를 바탕으로 누적 발전량을 계산하는 데 활용된다.

3. 개발 환경 구성

시스템 제어 프로그램 실행에 필요한 개발 환경은 아래 절차에 따라 구성한다.

1) Python 설치 확인 및 설치

대부분의 라즈베리파이 OS에는 Python 3가 기본 설치되어 있으나, 미설치 환경을 대비하여 아래 터미널 명령어로 설치할 수 있다.

Bash

```
sudo apt-get update
sudo apt-get install python3 python3-pip
```

2) 필수 라이브러리 설치

Python 패키지 관리자인 pip를 사용하여 제어에 필요한 라이브러리들을 일괄 설치한다.

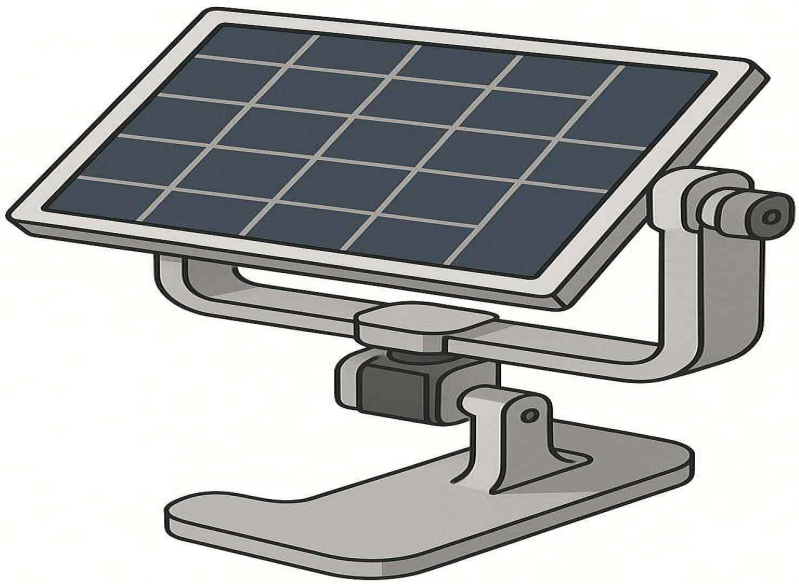
Bash

```
sudo pip3 install RPi.GPIO adafruit-circuitpython-ina219 spidev
```

spidev는 SPI 통신이 필요할 경우를 대비해 함께 설치한다.

아두이노 등 외부 MCU와 연동하는 Firmata 프로토콜을 사용할 경우, pyfirmata 패키지를 추가로 설치할 수 있다.

3-2.【기계 설계】



1. 시스템 구성

본 시스템은 3W급 소형 태양광 패널(145×145mm)의 효율을 극대화하기 위해 2축 추적 방식으로 설계되었다. 주요 구성 요소는 다음과 같다.

태양광 패널: 소형(145×145mm, 3W)

지지대: 3D 프린터로 제작된 U자형 브라켓 구조를 채택하여 패널의 무게를 효과적으로 분산하고 안정적인 지지를 확보함.

구동부:

X축 (좌우, Pan): MG996R 고토크 서보모터

Y축 (상하, Tilt): MG995 표준 서보모터

2. 구동 모터 선정을 위한 토크 계산

시스템의 안정적인 구동을 위해 패널 무게를 견딜 최소 토크를 계산하고, 외부 요인(바람, 마찰 등)을 고려하여 3배의 안전계수를 적용하였다.

토크 계산 공식: $T = (m \cdot g) \cdot L$

(m: 질량, g: 중력가속도, L: 회전축-무게중심 거리)

계산 과정 (패널 질량 0.2kg 가정 시):

필요 힘(F) = $0.2kg \times 9.81m/s^2 \approx 1.96\ N$

레버 암(L) = $145mm / 2 = 0.0725\ m$

최소 필요 토크(T) = $1.96N \times 0.0725m \approx 0.14\ N \cdot m$

최종 필요 토크 (안전계수 x3 적용) = $0.14\ N \cdot m \times 3 = 0.42\ N \cdot m$

3. 결론

계산 결과, 시스템 구동에는 최소 0.42 N·m 이상의 토크가 요구된다. 이에 따라 다음과 같이 모터를 최종 선정하였다.

X축은 Y축 구동부 전체 하중을 감당해야 하므로, 더 높은 힘이 필요한 점을 고려하여 고토크 사양의 **MG996R (약 1 N·m)**을 채택하였다.

Y축은 패널 자체의 하중만 감당하므로 표준 사양의 MG995로도 충분한 구동이 가능하다.

선정된 두 모터 모두 요구 토크 값을 크게 상회하므로, 외부 환경 변화에도 안정적인 패널 추적 성능을 보장할 것으로 판단된다.

3-3.【하드웨어 설계】

-라즈베리파이 GPIO PWM → 서보모터 2축(X, Y)

-GPS 모듈 → 라즈베리파이 (UART 통신)

-포토다이오드 4개 → 라즈베리파이 (ADC 모듈 통해 입력)

-온습도 센서(DHT22) → 라즈베리파이 (GPIO)

-INA219 (I^2C) → 태양광 패널 전류/전압 측정

-보호회로: 퓨즈(1차), MOSFET/릴레이(2차)

3-4.【소프트웨어 설계】

본 시스템의 소프트웨어 아키텍처는 ①핵심 제어 모듈, ②데이터 관리 모듈, ③MCP 서버 모듈, ④AI 클라이언트 모듈로 구성된다. 특히, 시스템의 외부 통신은 MCP(Model Context Protocol) 표준을 준수하여 설계함으로써 AI 애플리케이션과의 높은 호환성과 확장성을 확보한다.

1. 핵심 제어 모듈 & 데이터 관리 및 보호 모듈

1) 센서 데이터 수집:
GPS → 위도/경도/UTC → 태양 고도·방위각 계산

```
import serial
gps = serial.Serial('/dev/ttyAMA0', 9600)
line = gps.readline().decode('utf-8')
```

포토다이오드 → GPS 보조 추적(미세 보정)

```
import spidev
spi = spidev.SpiDev()
spi.open(0,0)
```

INA219 → 전류/전압 측정, 발전량 계산

```
from adafruit_ina219 import INA219
v = ina.bus_voltage
i = ina.current
```

온습도 센서 → 날씨 상태 판별

```
import adafruit_dht
temp = dht.temperature
hum = dht.humidity
```

2) 추적 제어 알고리즘:
-GPS 위치값으로 태양의 이론적 위치 계산
-포토다이오드 값 비교 → 오차 보정

```
error = left_sensor - right_sensor
if abs(error) > threshold:
    adjust_motor(error)
```

-계산된 좌표에 따라 서보모터(PWM) 제어 → 패널 회전

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
pwm = GPIO.PWM(18, 50)    # 50Hz 서보 제어
pwm.start(0)
pwm.ChangeDutyCycle(angle)
```

-온습도 값에 따라 흐림/우천 시 보호 모드(Off) 전환

3) 데이터 기록:
-전류, 전압, 전력, 누적 발전량을 주기적으로 측정
-CSV 파일로 저장하여 분석 가능

```
import csv
with open("log.csv", "a", newline="") as f:
    writer = csv.writer(f)
    writer.writerow([time, v, i, power])
```

-별도의 웹 대시보드/SQLite/Flask 서버는 구현하지 않음

4) 보호 로직:
-INA219 측정값이 설정한 한계치를 초과하면 → 라즈베리파이에서 MOSFET/릴레이 제어 → 회로 차단

```
if v > V_MAX or i > I_MAX:
    cut_off_relay()
```

3. MCP 서버 모듈 (Flask 기반)

이 모듈은 태양광 트래커 시스템 전체를 하나의 'AI가 사용할 수 있는 도구(Tool)'로 추상화하고, MCP 표준에 따라 외부와 통신하는 역할을 수행한다.
역할: 물리적인 하드웨어 제어와 데이터 처리 로직을 표준화된 디지털 인터페이스로 변환한다. AI 클라이언트는 이 인터페이스를 통해 시스템의 내부 동작을 알 필요 없이 필요한 기능(Action)을 호출할 수 있다.
구현: Flask 웹 프레임워크를 사용하여 MCP 사양을 따르는 RESTful API 서버를 구축한다. 각 API 엔드포인트는 MCP의 'Action'에 해당한다.
Python

```
# MCP 서버의 Action 구현 예시
from flask import Flask, jsonify, request

app = Flask(__name__)

# 'getStatus' Action: 시스템의 현재 상태를 반환
@app.route('/mcp/actions/getStatus', methods=['POST'])
def get_status_action():
    status_data = core_logic.get_current_status()
    return jsonify({'result': status_data})

# 'controlPanel' Action: 패널을 제어
@app.route('/mcp/actions/controlPanel', methods=['POST'])
def control_panel_action():
    params = request.get_json().get('params', {})
    axis = params.get('axis')
```

```
    angle = params.get('angle')
    result = core_logic.set_panel_angle(axis, angle)
    return jsonify({'result': f"Panel {axis}-axis moved to {angle} degree
s."})
```

4. AI 클라이언트 및 외부 통신

역할: 사용자 인터페이스를 제공하고, 사용자의 요청을 해석하여 MCP 서버의 Action을 호출하는 지능형 에이전트 역할을 수행한다.

- 네트워크 아키텍처:

로컬 통신 (MCP): 클라이언트(웹 대시보드)는 로컬 네트워크를 통해 라즈베리파이의 MCP 서버와 표준화된 프로토콜로 통신한다.
외부 통신 (Internet): AI 리포트 생성 시, 클라이언트는 인터넷을 통해 외부 LLM API 서버와 통신한다. 이 때, MCP 서버로부터 받은 데이터를 LLM에 전달하는 역할을 한다.

- AI 기능 구현 흐름:

1. 입력: 사용자가 "오늘 발전량 요약해줘" 라고 입력.
2. 해석: 클라이언트는 이 입력을 getReportData(period='today') 라는 Action 호출로 해석.
3. 실행 (MCP): MCP 서버의 getReportData Action을 호출하고 로그 데이터를 응답으로 받음.
4. 생성 (LLM): 받은 로그 데이터를 프롬프트와 함께 외부 LLM API에 전송.
5. 출력: LLM으로부터 받은 분석 리포트를 대시보드에 표시.

3-5.【필요부품 사양】

부품리스트

NO	부품명	사양/모델명	수량	단가
1	라즈베리파이4	4GB	1	86000원
2	18650리튬이온배터리	3500mAh	1	7100원
3	태양광 페널	3W, 145X145mm	1	38900원
4	GPS모듈	NEO-6M	1	5500원
5	온습도센서모듈	DHT11	3	880원
6	포토 다이오드 센서	GL5528	10	500원
7	서보모터1	MG996R서보모터	1	6800원
8	서보모터2	MG995서보모터	1	5000원
9	전류전압측정다이오드	INA219	3	1290원
10	충전보조회로	TP4056	4	1000원
11	18650 배터리 홀더	점퍼핀타입	3	390원
12	실시간 시계 모듈	ds3231	1	8500원
13	25T 서보 암	MAC-027	1	2600원

4. 팀원별 역할

1) 송승훈

1. 조사한 자료를 취합하여 계획서 작성 및 문서 작업
2. 매 주차마다 팀원이 해야 할 과제 지정

2) 권형준

1. 하드웨어 제작 및 배선
2. 발전된 에너지를 어디에 활용할 것인가 구상
3. 주별 회의록 작성

3) 백상철

1. 라즈베리 파이 전체적인 작동 흐름에 대한 조사
2. 파이썬 문법과 활용법 공부

4) 김덕현

1. 라즈베리파이 블루투스 연결에 대해 알아보기
2. Wi-Fi 연결 방법 알아보기

5) 한태민

1. 태양광 추적 메커니즘과 전반적이 동작원리 알아보기
2. 필요부품 정리하기

6) 김지훈

1. 주별 회의록 작성
2. 모터에 대한 라즈베리파이 코드 조사
3. 앱 연결에 대하여 공부

5. 기대효과

본 프로젝트의 태양광 추적 시스템은 태양의 위치를 실시간으로 따라가도록 설계되어, 고정형 패널 대비 발전 효율을 향상시킬 수 있습니다. 또한 전류·전압 센서를 이용한 발전량 측정을 통해 시스템 상태를 쉽게 진단하고, 발전 효율 분석에도 활용할 수 있습니다.

소형·저비용 구조로 제작되어 교육용·실험용으로도 적합하며, 신재생에너지 활용성을 높여 에너지 절약 및 탄소 배출 저감에 기여할 수 있습니다.

6. 향후 확장

1) IoT 기반 데이터 전송, 대용량 배터리 충전 시스템, AI 학습을 통한 예측 제어 등으로 확장 가능하여, 임베디드 시스템·재생에너지·전력전자 분야가 융합된 연구 주제로 발전 가능하다.

2) 본 프로젝트에서는 소형 태양광 패널을 이용해 발전량 측정 및 추적 성능 검증에 초점을 두었으며, 배터리 충전 및 MPPT 기능은 구현 범위에 포함하지 않았다. 그러나 실제 응용 단계에서는 발전 효율 극대화와 안정적인 전력 공급을 위해 MPPT(Maximum Power Point Tracking) 알고리즘과 배터리 관리 회로가 필수적이다.

-MPPT 제어

태양광 패널은 일사량, 온도에 따라 출력 전압·전류가 변한다.

MPPT 알고리즘(예: Perturb & Observe, Incremental Conductance)을 적용하면 최대 전력점에서 동작하도록 제어하여 발전 효율을 높일 수 있다.

향후 시스템 확장 시 DC-DC 컨버터와 연계하여 구현 가능하다.

-배터리 충전 및 관리(BMS)

발전 전력을 효율적으로 저장하기 위해 리튬인산철($LiFePO_4$) 배터리와 같은 고안전성 배터리를 사용할 수 있다.

BMS(Battery Management System)를 통해 과충전/과방전 보호, 셀 밸런싱, 과전류 차단을 수행하여 안정성과 수명을 보장한다.

-확장 가치

MPPT와 배터리 관리 기능을 통합하면, 단순 발전 실험용을 넘어 실제 에너지 저장 및 활용이 가능한 독립형 전력 시스템으로 확장할 수 있다.

IoT 센서 노드, 독립 전원 공급 장치, 소형 친환경 발전 설비 등 다양한 응용이 가능하다.

7. 설계 일정

주차	SW개발내용
1주차	아이디어 회의, 주제 확정, 전체 구조 구상
2주차	부품 선정 및 재료 구매 (패널, 모터, 센서, 배터리, 3D프린팅 재료)
3주차	제어 환경 세팅 (Python GPIO, PWM 기본 코드 작성)
4주차	센서 입력 코드 구현 (포토다이오드, 온습도, GPS 테스트)
5주차	태양 위치 계산 알고리즘(SPA/NOAA) 구현
6주차	센서 + 모터 연동 테스트, 기본 추적 동작 확인
7주차	보정 알고리즘 적용 (포토다이오드 미세보정)
8주차	데이터 로깅/그래프 기능 구현, 로그 저장
9주차	통합 시연 프로그램 완성, GUI/간단 대시보드 구성
10주차	최종 코드 정리, 보고서 제출용 다이어그램 완성

주차	HW개발내용
1주차	아이디어 회의, 주제 확정, 전체 구조 구상
2주차	부품 선정 및 재료 구매 (패널, 모터, 센서, 배터리, 3D프린팅 재료)
3주차	3D 모델링 시작
4주차	3D 프린터 출력 및 1차 조립
5주차	HW 조립 보완(사용하는 센서들의 결선위치 파악)
6주차	전체 HW 통합조립 완료
7주차	전체 HW안정화(케이블 정리, 고정, 하우징, 보완)
8주차	중간 표 준비 (HW사진, 회로도, 3D 구조도)
9주차	최종점검 (내구성, 전원효율 테스트)
10주차	최종 발표 자료 준비(HW 완성품 전시)