AI & SUSTAINABILITY IN VET EDUCATION

Name of the Author/s: GR Team

Title of WP: WP3 – AI-SAVES@ProjectHub

Activity Reference: 3.2 – Project template for AI-SAVES@ProjectHub

Logo:

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Table of Contents

Table of Contents **2**

1. Project Title 3

2. Project Category 3

3. Project Description 3

4. Keywords 4

5. Target Audience 4

6. Project Objectives 4

7. Project Timeline 4

8. Resources Provided 5

9. Tools and Technologies Used 5

10. Step-by-Step Implementation Plan 5

11. Evaluation Criteria 9

12. Languages Available 9

13. Contributors 9

14. Feedback and Continuous Improvement 10

Submission Notes **10**

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

AI-SAVES@PROJECTHUB

1. Project Title:

Automatic Plant Rotator With Sunlight Sensor

2. Project Category:

AI / IoT / Smart Agriculture – STEM Maker Project

3. Project Description:

This project presents a smart plant rotator system that automatically rotates a plant pot depending on the direction of sunlight.
The system uses light sensors and a motor controlled by a microcontroller (e.g., Raspberry Pi or Arduino) to detect where the light is strongest and rotate the plant accordingly.
The goal is to ensure that plants receive even sunlight exposure, improving growth and preventing plants from bending toward one side.
This project introduces students to smart agriculture technologies, sensors, motor control, and IoT-based plant monitoring.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

4. Keywords:

smart agriculture
IoT gardening
light sensor
plant rotator
Raspberry Pi
automation
STEM
environmental technology

---

5. Target Audience:

Vocational education and training (VET)
STEM laboratories
Students interested in IoT and environmental technology

---

6. Project Objectives:

Understand how light sensors detect sunlight intensity

Learn to control motors using a microcontroller

Explore automation in agriculture

Develop skills in electronics and programming

Understand how technology can support sustainable plant growth

---

7. Project Timeline:

Estimated duration: 6–8 hours

Phase	Time
Introduction to smart agriculture	1 hour
Hardware assembly	2 hours
Electronics wiring	1 hour
Programming	2 hours
Testing and calibration	1–2 hours

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

8. Resources Provided:

---

9. Tools and Technologies Used:

Raspberry Pi

Stepper motor (e.g., 28BYJ-48)

Motor driver (ULN2003)

Light sensors (LDR)

Rotating platform

Power supply

Jumper wires

---

10. Step-by-Step Implementation Plan:

Step 1: Connect BYJ48 Motor With Raspberry Pi

Co-funded by the European Union

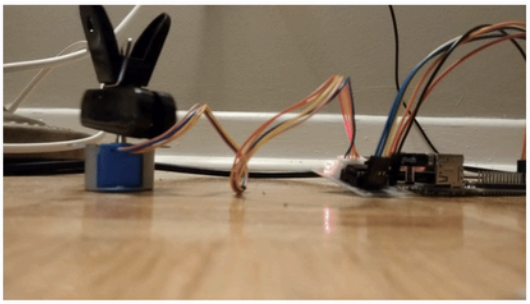
This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Keith Weaver 604 Followers About Follow Sign in Get started

Controlling Stepper Motors using Python with a Raspberry Pi

Keith Weaver May 27, 2018 · 4 min read

In this tutorial, I will show you how to setup a stepper motor with a Raspberry Pi. I'm using a 40 pin Raspberry Pi 3 (same for 28 pin) and Kumantech Motor Stepper kit.



When I got started with Stepper Motors I followed this well explained video below. There is no written portion or at least as of now, the link in the description no longer works. I'm simply providing my code sample and more images of my setup.

I mainly followed this tutorial to get the motor working with my Raspberry Pi.
You can modify the 512 (360 degrees) number in the test script to change how much you want the motor to rotate by.
for i in range(512):
Modify the following line to change the speed of rotation.
time.sleep(0.001)
Step 2: Attach Plate, Motor, and Foundation

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



I miraculously found a smart cart chassis kit that fit well with the BYJ48 stepper motor and also provided stability to the plate so I decided to combine everything.

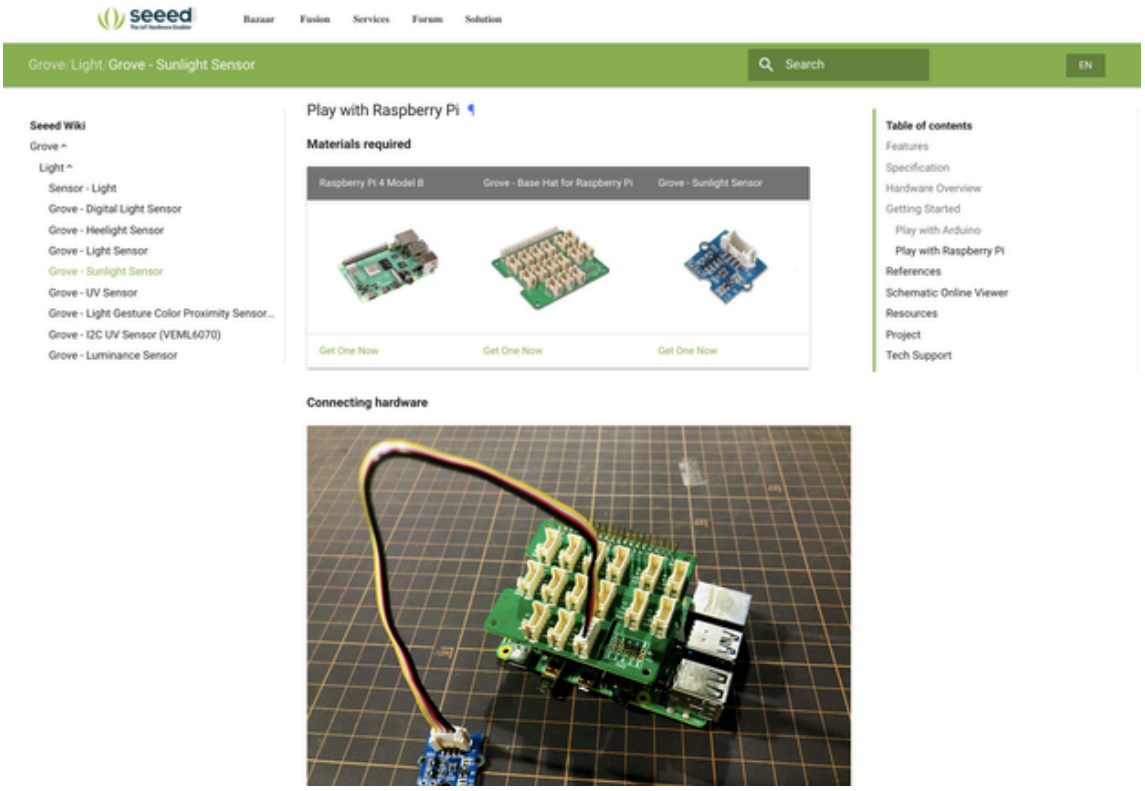
1. Superglued 3D printed plate with a wheel from the smart car kit
2. Screwed in BYJ48 motor to the smart car body (now the foundation of this device)
3. Had to apply a thin layer of scotch tape around the motor shaft so that it fit perfectly with the wheel

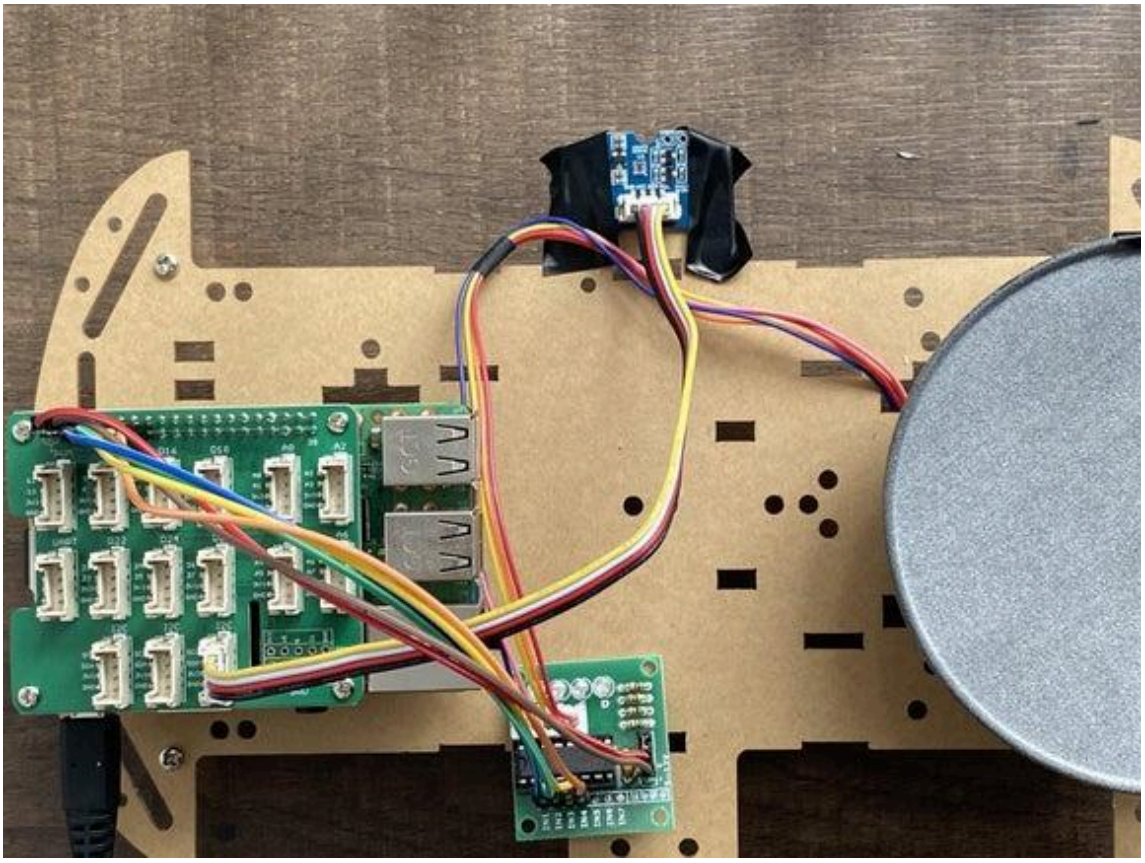
Before attaching everything, I would recommend placing your plant/pot on the plate and making sure that it's stable even when it rotates. My plant weighs roughly 1kg.

Step 3: Attaching Grove Sunlight Sensor

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

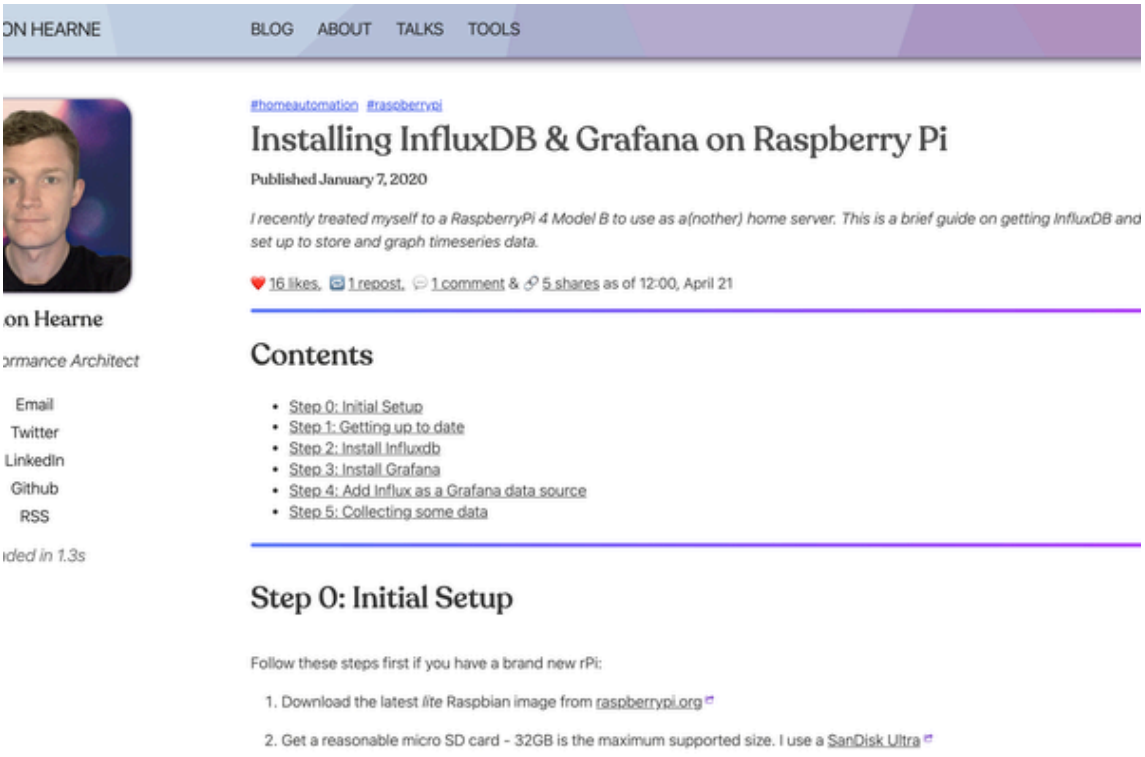


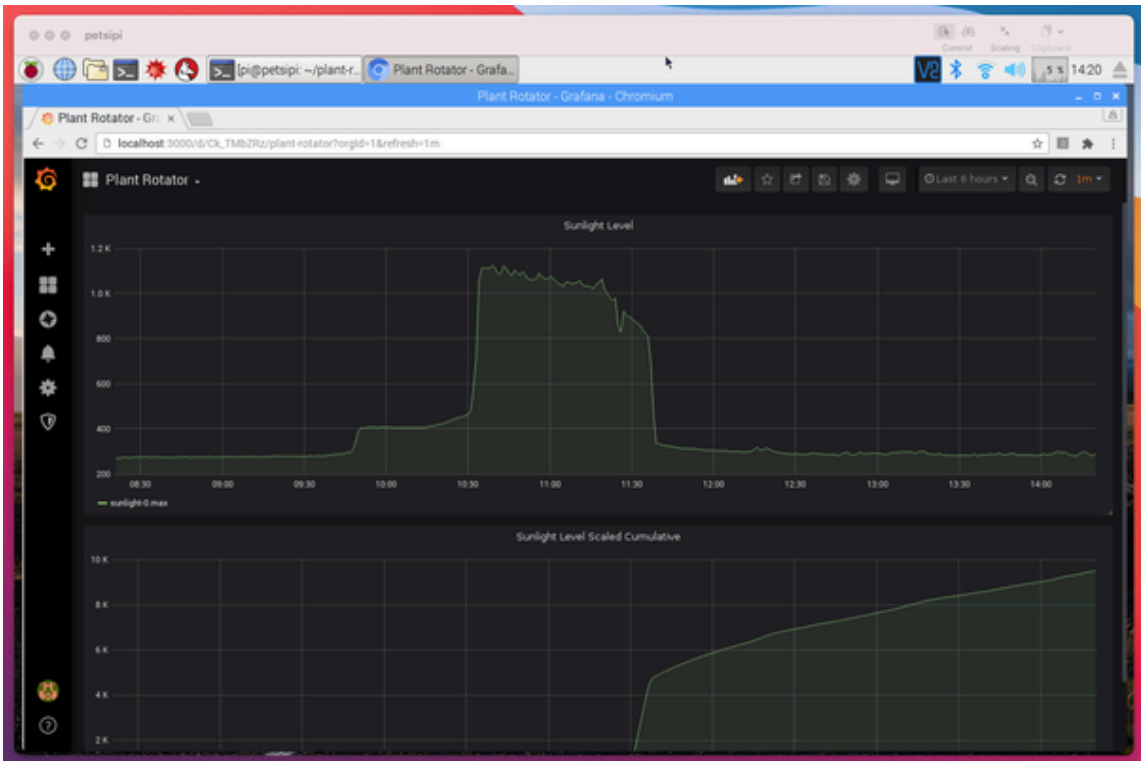


I mainly followed this tutorial to get the sunlight sensor working with my Raspberry Pi.
If you can get the test script to run, you should be set!
python3 examples/BasicRead.py
I then taped the sensor to the foundation so that the sensor is always in the same position.
Step 4: Collecting Sunlight Data

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.





I followed [this] tutorial to set up InfluxDB and Grafana on my raspberry Pi.
InfluxDB is a time series database designed to handle high write and query loads
Grafana is a multi-platform open source analytics and interactive visualization web application
After getting influx running, I created a database called *plant* and set a retention policy so that we don't retain the data forever.
influx
> create database plant
> create retention policy "two_weeks" on "plant" duration 2w replication 1

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

I wrote up a quick script to save the sunlight level to InfluxDB

```
# collect.py

from influxdb import InfluxDBClient

def read_metric():
    import seeed_si114x
    SI1145 = seeed_si114x.grove_si114x()
    return SI1145.ReadVisible

def save_metric(metric, value):
    client = InfluxDBClient('localhost', 8086, 'pi', 'password', 'plant')
    point = {
        'measurement': metric,
        'fields': {
            'value': value
        }
    }
    client.write_points([point])

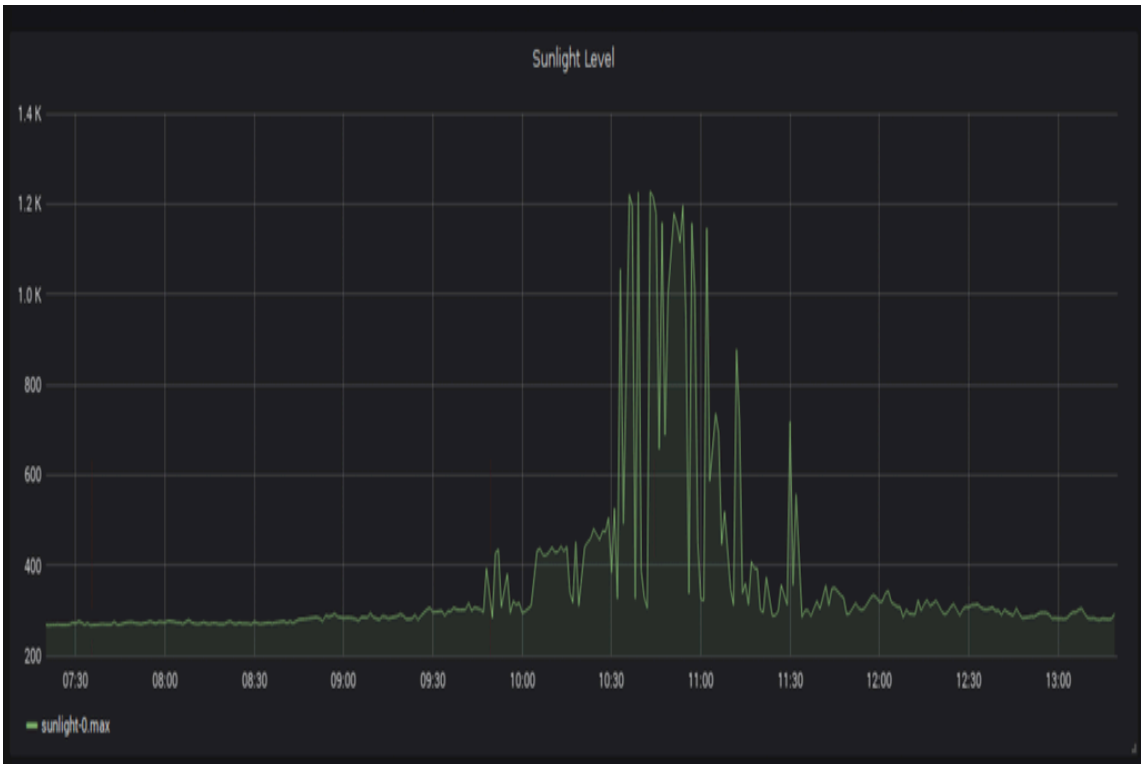
if __name__ == '__main__':
    sun_value = read_metric()
    save_metric('sunlight', sun_value)
```

I added this script to a cron job so that it is executed every minute

```
* * * * * python3 /home/pi/plant-rotator/collect.py
```

I graphed the collected data in Graphana to establish a baseline.

Step 5: Determining Baseline and Rotation Threshold



After a couple days of data collection, I noticed that the sunlight level typically ranges from about 258~1200. I decided to ignore all values below 265 as those numbers were observed

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

during the night and set a threshold of 50,000 accumulative delta.
In other words, I keep track of the sunlight level for one section every minute, and keep a running count of how much sunlight exposure exceeding 265 that section gets. When that aggregated number exceeds 50,000, I rotate the plant 90 degrees, reset the counter, and start tracking again for the new section.
Logic (pseudo code):

```
If ∑max(sunlight_level - 265)  > 50000:
   rotate
   section = (section + 1) % 4
```

If the plant is getting the max sunlight level around 1200, the device should rotate every 50 minutes, but in practice, it rotates about once a day.
Step 6: Combining Everything
Based on the threshold from the previous section, I combined everything and wrote a program that automates the data collection and rotation of the plant. The program consists of a script and two text files that keep track of the the current section and the running total of sunlight exposure.

```
# run.py

import RPi.GPIO as GPIO
import time
from influxdb import InfluxDBClient

THRESH = 265
ROTATE_THRESH = 50000

def rotate():
   GPIO.setmode(GPIO.BOARD)
   control_pins = [7,11,13,15]
   for pin in control_pins:
      GPIO.setup(pin, GPIO.OUT)
      GPIO.output(pin, 0)
   halfstep_seq = [
      [1,0,0,0],
      [1,1,0,0],
      [0,1,0,0],
      [0,1,1,0],
      [0,0,1,0],
      [0,0,1,1],
      [0,0,0,1],
      [1,0,0,1]
   ]
   for i in range(128):
      for halfstep in range(8):
         for pin in range(4):
            GPIO.output(control_pins[pin], halfstep_seq[halfstep][pin])
         time.sleep(0.002)
   GPIO.cleanup()

def get_section():
   with open('/home/pi/plant-rotator/section_sunlight.txt', 'r') as f:
      section_value = int(f.readline())
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
    with open('/home/pi/plant-rotator/current_section.txt', 'r') as f:
        section = int(f.readline())

    if section_value > ROTATE_THRESH:
        section = (section + 1) % 4
        with open('/home/pi/plant-rotator/current_section.txt', 'w') as f:
            f.write('%s\n' % section)
        section_value = 0
        rotate()
    return section, section_value

def read_metric():
    # the rotation has to happen before this import
    import seeed_si114x
    SI1145 = seeed_si114x.grove_si114x()
    return SI1145.ReadVisible

def save_metric(metric, value):
    client = InfluxDBClient('localhost', 8086, 'pi', 'password', 'plant')
    point = {
        'measurement': metric,
        'fields': {
            'value': value
        }
    }
    client.write_points([point])

def update_section_value(new_value):
    with open('/home/pi/plant-rotator/section_sunlight.txt', 'w') as f:
        f.write('%s\n' % new_value)

if __name__ == '__main__':
    section, section_value = get_section()
    sun_value = read_metric()
    delta = max(sun_value - THRESH, 0)

    save_metric('sunlight-%s' % section, sun_value)
    save_metric('sunlight-delta-%s' % section, delta)

    update_section_value(section_value + delta)
```

We replace the data collection cron job with this new script so that it runs every minute.

```
* * * * * python3 /home/pi/plant-rotator/run.py
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

11. Evaluation Criteria:

Criterion	Description
System assembly	Correct mechanical construction
Electronics	Proper wiring and sensor integration
Programming	Code successfully controls rotation
Functionality	Plant rotates toward light
Innovation	Additional improvements or features

---

12. Languages Available:

English

---

13. Contributors:

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

17. Feedback and Continuous Improvement:

Teachers and students can suggest improvements such as:

adding AI plant disease detection

integrating IoT monitoring platforms

developing mobile app control

.

Submission Notes:

Please ensure that all required documents and resources are attached to your submission. The AI-SAVES@ProjectHub review team will evaluate the submission based on the provided information. Once approved, the project will be published on the hub, and partners will be notified.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.