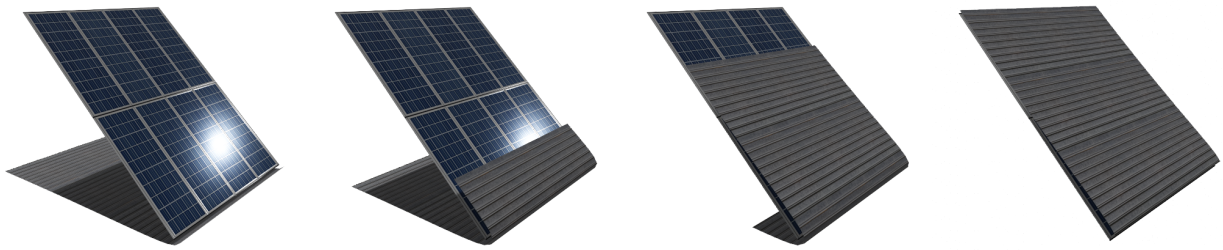
# Automatic solar panel protection canopy

Group 16


| | | | |
|---|---|---|---|
| Manuel Aumeier<br>k12141864 | Simon Danninger<br>k12113945 | Elias Foramitti<br>k11917423 | Liliane Loibl<br>k11907879 |
| Luis Nachtigall<br>k12021202 | Nathalie Palecek<br>k12125213 | Kevin Zauner<br>k1355926 | |


## Motivation

Due to the ever-growing energy crisis and the underlying shortage of fossil energies, it is becoming more and more prominent to shift towards renewable energies like wind and solar power. Solar panels, however, are often at the mercy of all kinds of environmental influences. For instance, when it's cloudy or rainy outside, they can produce close to no energy. But what's even more dangerous, the panels can be damaged by falling branches on a windy day or hail. Therefore, we propose a smart canopy that extends and protects the solar panels from severe weather conditions. The mechanism should function similarly to a garage door and expand automatically in the case of high precipitation or wind force to prevent the solar panels from potential damage. With such a system, we envision ensuring that the solar panels can withstand damage in the long term, thus making solar power systems even more durable and a future-oriented energy source.



## Approach

In the scope of this project, we will develop a first prototype of the idea. We will therefore only demonstrate the concept on one small solar panel, using a model canopy (made from cheap materials like wood, cardboard and string) and will not spend too much time on making all the hardware weatherproof for long-term deployment for now.

However, we intend to design our code and interfaces such that as much of it as possible can be reused in a final product.

The central unit of the system is a Raspberry Pi connected to an anemometer (i.e. wind speed meter), photoresistor, precipitation sensor, a stepper motor driving the canopy, and 2 momentary switches on each end of the canopy rail path (to determine the current state of the canopy). The Raspberry Pi gathers data from all 3 sensors and decides when to open or close the canopy from that.

Since we are only interested in wind speed, not wind direction, a cup anemometer is the best option for our purposes. There are other forms of wind sensors (vane, hot-wire, ultrasonic, laser Doppler, etc.) all of which are, however, more expensive, less reliable or unidirectional.

For precipitation, the decision is less clear. Here, too, many viable solutions exist (contact moisture, infrared-reflection, flow-meter). However, as we are mainly interested in hail, infrared reflections will most likely not work. Because of the higher cost of flow-meter-based precipitation sensors, we opted for a contact moisture sensor. Since we were worried about the reliability of this solution, however, we opted to also include a photoresistor in our design, which will detect changes in light during weather storms. This might, of course, lead to a high number of false positives (especially at night). As most types of solar panels are quite sensitive to light conditions, though, and only produce significant amounts of energy in direct sun, those false positives are probably not too problematic. It might even be beneficial to close the canopy whenever the solar panel is not in use (such as in low-light situations), for extra protection against risks not apparent from the sensor data.

For determining if the canopy is currently open or closed and how far the motor has to turn to alter between the two states, theoretically, the stepper motor alone would be enough. However, we decided to add 2 momentary switches to each end of the canopy rail (activated whenever the canopy reaches one of the ends), to deal with potential slack in the system.

Additionally, the Raspberry Pi serves as an HTTP endpoint, presenting the current sensor data and canopy state and offering simple controls (manual canopy control, and setting thresholds for the closing algorithm).

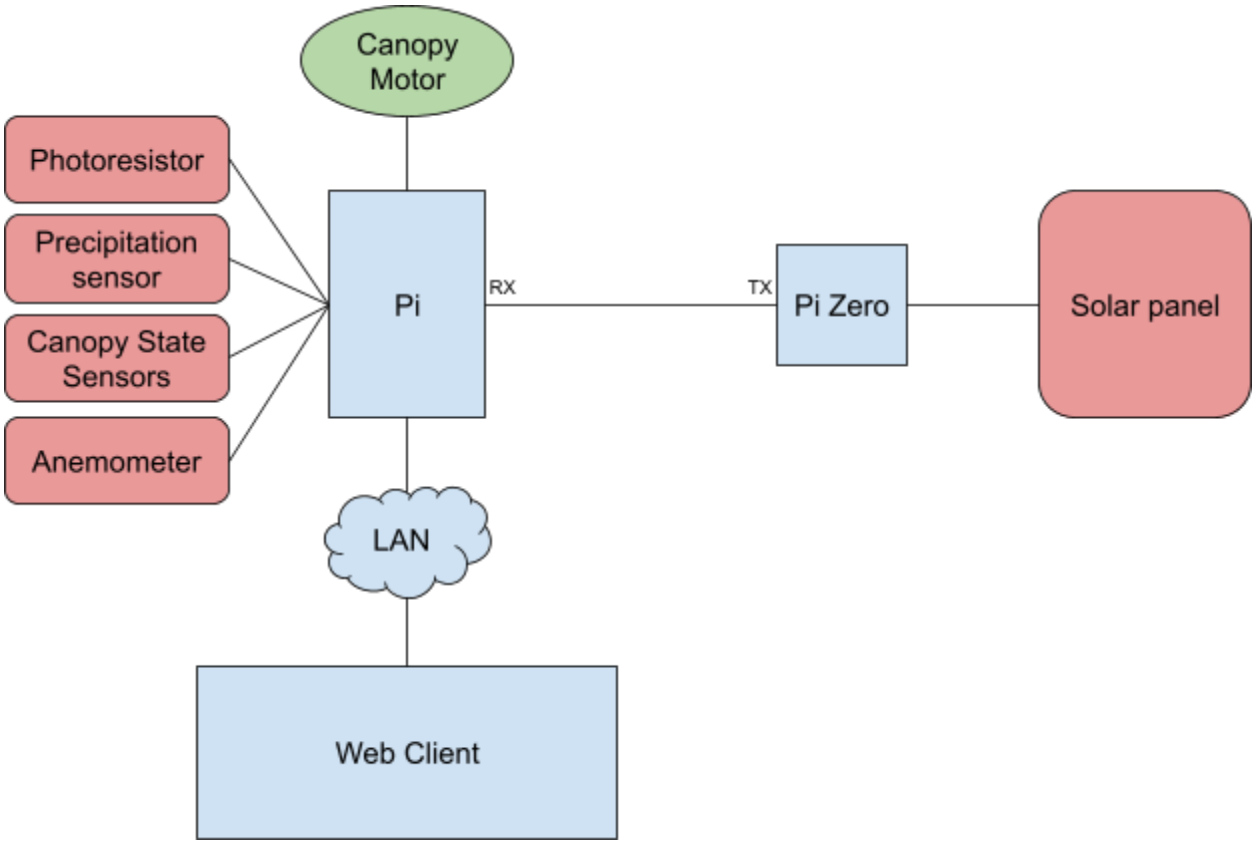
Apart from that, we will connect a Raspberry Pi Zero to the solar panel, measuring the voltage produced at any moment. This information is then shared with the central Raspberry Pi over a serial connection similar to communication schemes of standard charge controllers for full-sized solar systems (such that in a final product this can be

replaced by an actual charge controller or power inverter unit). The voltage data is in turn served on the above-mentioned HTTP page.

The information from this stand-in charge controller will also allow us to investigate the assumption about the insignificance of closing the canopy during low-light situations on the energy output.

Hardware List:

- Raspberry Pi 3
- Raspberry Pi Zero
- Small Reduction Stepper Motor
- Photocell sensor from Sensor pack 900
- Small solar panel (supplied by ourselves)
- Raindrop Rain Sensor Module (supplied by ourselves)
- Cup Anemometer (supplied by ourselves)
- 2 momentary switches (supplied by ourselves)
- Canopy framework (built from wood, cardboard and string)



## Implementation Details

All the used sensors (anemometer, photoresistor, precipitation sensor, and momentary switches) can simply be read using the GPIO interface directly without the need for any additional libraries.

For the web server running on the Raspberry Pi main unit, we will use Flask, a popular Python web server library.

For the Raspberry Pi Zero as a stand-in charge controller, we wanted to roughly imitate the communication schemes of real solar charge controllers. These often send information (usually much more than only the current power output) over RS-485, a more complex serial interface. Simulating the behavior without dedicated RS-485 hardware, we will use the serial connections provided by the RX and TX pins of the Raspberry Pis using the serial standard library in Python. The Raspberry Pi Zero will send the current power output of the solar panel in volts to the Raspberry Pi main unit once per second.

## Verification

For verification, we will test 4 scenarios:

- Creating artificial shade on the photoresistor. If the light level becomes low enough, the canopy will close over the solar panel. Once the shade is removed the canopy opens again.
- Creating artificial rain on the precipitation sensor using a watering can. The canopy closes. Once the sensor dries the canopy opens again.
- Creating artificial hail on the precipitation sensor using crushed ice. The canopy closes. Once the sensor dries the canopy opens again.
- Creating artificial wind using a hair dryer. If the measured wind speed becomes high enough, the canopy will close over the solar panel. Once the wind becomes slower the canopy opens again.
- Closing the canopy manually over the web interface, and opening it again.

During all the tests, the changes to the environment should be reflected in the sensor data published on the web page served by the Raspberry Pi main unit.