Video + Technical Description

This project is a battery-powered, two-axis solar tracker that continuously repositions a panel to face the brightest available light source. Light-dependent resistors on the panel guide the two motors, while mechanical limit sensors and an angular position sensor keep the panel within its safe range of motion. Every reading passes through a fixed-point Kalman filter before reaching the control logic, reducing sensor noise and providing smooth, stable input.

The "go beyond" algorithm makes the tracker more accurate than a simple light-following mechanism. Because the LDRs and other sensors are powered by the battery, their ADC readings can drift as the battery voltage drops. The system compensates by continuously recalibrating against the current battery level rather than relying on fixed reference values. The potentiometer is also calibrated at startup by sweeping fully left and right to record its ADC limits before normal operation begins. Together, these steps allow the algorithm to combine changing sensor data into reliable movement decisions.

In the demonstration video, the tracker actively and responsively follows the light source while stopping at the tilt limits, confirming that the core algorithm works end to end. [Project Video]

IOT Data

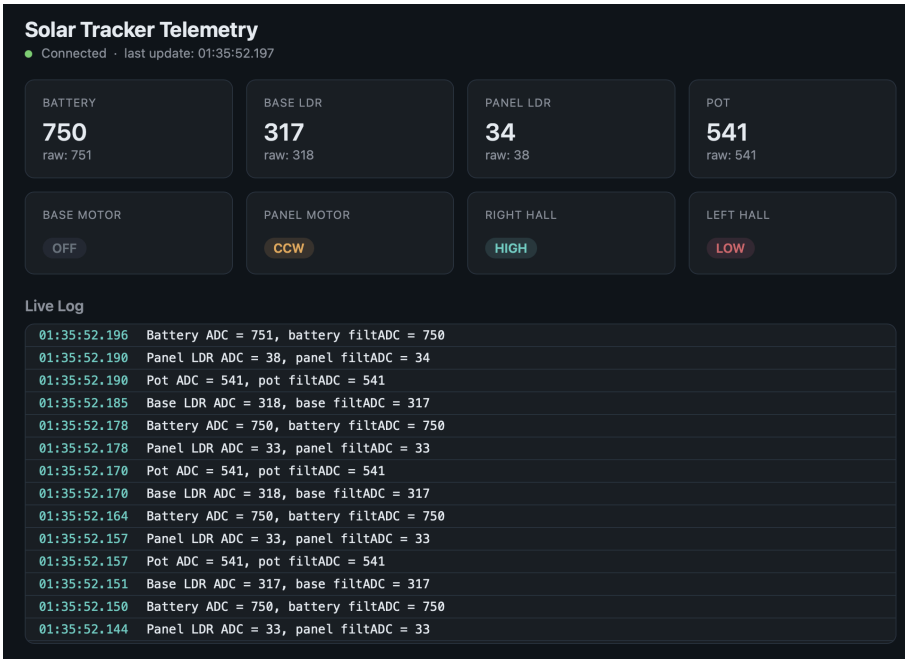


Fig 1. IOT Data

Data is sent to my laptop from the ATmega328P via UART. My laptop is running a Python program which displays the IOT data on a localhost website.

To ensure reliable serial communication without packet loss, the microcontroller is clocked with a 9.216 MHz crystal. This specific frequency achieves a 0% theoretical UBRR baud rate error across standard communication rates (I used 115200 baud), eliminating timing drift during telemetry transmission.

Enclosure + Setup

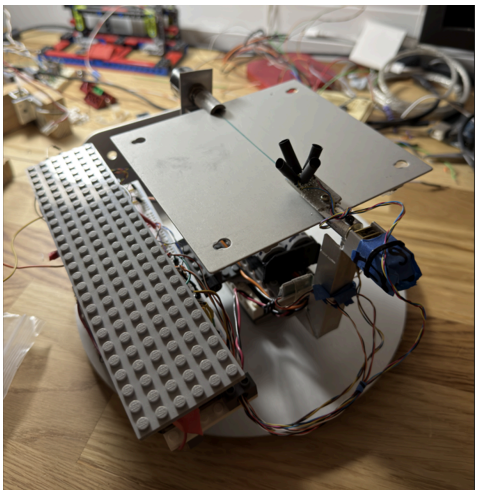


Fig 2. Picture of the Structure

The tracker is built on a rigid metal U-shaped base bolted to a flat metal plate, forming a stiff, self-supporting frame that carries both axes of motion. The entire assembly sits on a LEGO turntable driven by a motor-and-wheel so all moving mass is fully supported by the structure, not by wiring. The batteries mount at the base of the U to keep the center of gravity low and the platform stable while it turns. The electronics ride on a breadboard above the rotation motor, enclosed in a LEGO housing that keeps the wiring contained. Every subsystem is screwed or fastened to the frame using hand-cut brackets and custom couplings. For example, the panel motor and pots are coupled to the rotating metal panel. Nothing structural relies on tape. The result is a single self-contained unit that can be picked up, rotated, and operated as one piece, with each module (motors, panel, battery, and electronics) on its own mount so the build stays organized and debuggable.

Final BOM

Part	**Total**	**Link**	**Notes**
ATmega328P-AU MCU	$2.66	DigiKey	
Discrete electronics	$46.02	DigiKey	OP495 quad op-amp, 4 NMOS + 4 PMOS (two H-bridges), 8× 1N4007 diodes, crystal, resistors, caps, LED
10-turn pot, 10 kΩ	$18.27	RS	Precision calibration
LDR (GL5528-type), 4×	$3.00	Link	~1 kΩ bright, 100s of kΩ dark
LF21115TMR magnetic switch, 2×	$1.68	JLCPCB	One per axis
N20 6 V gear motors (2-pack)	$11.99	Amazon	

Part	Total	Link	Notes
Breadboard + jumper wire kit	$18.94	Adafruit / Digilent	
FTDI USB-serial cable	$19.95	Adafruit	
Li-ion cells 3300 mAh (2×) + holder	$18.48	18650 Store	4.2 V charged, series pack
LEGO structure, turntable, tire	$18.00	BrickLink	
Aluminum: U-bar, plates, screws, couplings, ground-plate base	$77.05	Home Depot	
Tape + zip ties	$13.41	Home Depot	
TOTAL	**$249.55**		

Table 1. Final BOM

The table below lists every part used in the final light tracker, including the enclosure and mounting hardware. Costs reflect current retail prices from the linked vendors; parts I already owned (such as the antenna ground plate used as the base) are still priced at replacement value so the total honestly represents what it would cost to reproduce the build. The most significant expenses are the aluminum structure, which provides a rigid, corrosion-resistant, and non-magnetic frame, and the precision components (the OP495 quad op-amp and 10-turn potentiometer) that preserve the tested analog sensing design. Notably, no "learning expenses" line is included: because I had all of the components and materials at home, iteration and mistakes along the way consumed parts I already owned rather than requiring replacement purchases. The total cost to reproduce the project comes to $249.55.

Reflection

My original CP1 premise, a battery-powered dual-axis tracker that uses an LDR array to find the brightest light, with Hall-effect tilt limits, a potentiometer for angle feedback, a capacitive touch input, and filtered signals on a custom board, held up well in Rev 1. The architecture was sound: the tracker follows a moving light source reliably, holds steady when it is fixed, and stops at its limits without over-rotating. The sensor choices I justified in CP1 all earned their place. A few things did change once the design was built and tested. Solar self-recharging was not realistic, so the meaningful "go beyond" became compensating for battery droop and self-calibrating at startup, which helps tracking accuracy far more. The ADCs are continuously recalibrating to adjust to the battery droop. A limit-overshoot issue was also solved more simply in software. These were evidence-driven refinements, not failures of the concept. Additionally, I made the Kalman filter fixed point improving CPU performance. On data and system quality, I am confident. Each sensor was characterized before integration, so I understand how

every signal behaves and where its limits are rather than trusting a datasheet. Filtering gives clean, stable readings, and UART/IoT logging let me confirm consistent behavior rather than a one-time demo. The system's limitations are ones of scope rather than reliability. Each sensor is used for exactly what it does well: the LDRs resolve direction rather than absolute light level and the Hall sensors and pot enforce the travel endpoints rather than tracking angle continuously. These choices keep each measurement inside the range where I verified it, which is exactly why the system behaves predictably. Rev 1 delivers clean data and repeatable tracking, and it does so on every run, not just the recorded one.

PCB

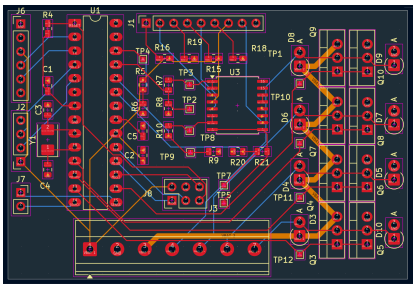


Fig 3. Circuit schematic

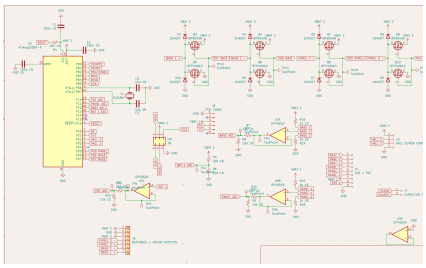


Fig 4. PCB layout / routing

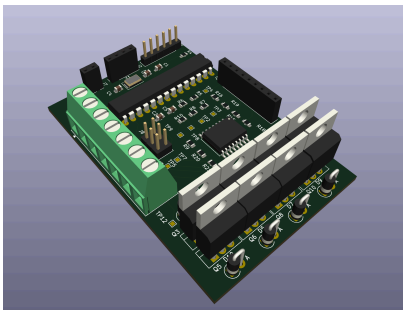


Fig 5. 3D rendered board

Component Selection and Board Stack-Up: For the printed circuit board, I used KiCad to design a four-layer stack-up (F.Cu, GND, B.Cu, and PWR) with a combination of surface-mount and through-hole components. The SMT components include the resistors, capacitors, the OP495 quad op-amp, and the crystal, while the through-hole components include the male and female pin headers, screw terminals, the ATmega328P microcontroller, the NMOS/PMOS MOSFETs, and the diodes. The MOSFETs were selected in a THT package rather than SMT, which allows small heatsinks to be added to the H-bridge devices if thermal performance under load makes that necessary.

Header and Connector Strategy: Interface connections were partitioned across several header types based on signal type and mechanical requirements. The potentiometer and LDR sensors share a single female pin header, while the Hall effect sensor is routed to its own header on the opposite side of the board to ease trace routing. Because the Hall effect output is a purely digital signal, it requires only VCC, GND, and a VOUT line to a GPIO, so the two active pins were placed where a two-pin footprint could be accommodated without competing for space with higher pin-count interfaces. The capacitive sensor, implemented as a 1 MΩ resistor with a soldered wire lead, was assigned its own female header adjacent to the Hall effect header to maintain physical and electrical separation between sensor subsystems. Battery and motor connections were routed through screw terminals rather than pin headers; although screw terminals consume more board area, they provide a more secure, vibration-resistant connection and were prioritized for the two interfaces most susceptible to accidental disconnection during testing. The FTDI interface was placed above the Hall effect header as a male pin header, since the FTDI cable itself terminates in a female connector. Only GND and RXD are required to read messages via UART, but the remaining signal lines, including TXD, were wired accounting for the six-pin footprint. Finally, an ISP header was placed in the middle of the board as a male connector to allow reprogramming the ATmega328 microcontroller. With all other headers and terminals pushed to

the board perimeter to preserve a clean layout, the center region was unused and placing the ISP header there made productive use of that space. As with the FTDI interface, the ISP cable is inherently female, requiring a male connector on the board. The header breaks out the full SPI bus along with reset and VCC, enabling in-circuit programming of the MCU without removal from the board.

Component Placement: Component placement was guided by signal path and noise-isolation considerations. Decoupling capacitors and the crystal were placed very close to the MCU (on its left side) and decoupling capacitance were placed close to both the supply pin and ground to minimize loop inductance. The right side of the MCU houses the analog signal conditioning circuitry containing the OP495 quad op-amp with supporting resistors arranged around it, along with a decoupling capacitor on the ADC voltage reference pin . This section was laid out to remain organized and inspectable, with test-point pads included at nearly every node to support design-for-manufacturing (DFM) verification. Further to the right, the diodes and MOSFETs forming the two H-bridge motor drivers were grouped as an isolated cluster, both to limit noise coupling into the sensitive analog stage and because only four traces connect this cluster back to the MCU, making central placement unnecessary.

Routing and Layer Strategy: Routing was performed sequentially by layer to manage complexity. On the top layer (F.Cu), only SMT components were routed initially, prioritizing short, direct connections and deferring longer traces that would require weaving around other components. The bottom signal layer (B.Cu) was then used to route the remaining signal traces, with vias added wherever an SMT pad required a transition to that layer, since those components have connections only on the top surface. The GND layer was implemented as a solid copper pour rather than discrete traces, functioning as a continuous reference plane; only stitching vias down from SMT components were required to tie into it. The fourth layer, PWR, was routed with two electrically separate battery nets: V Bat 1 supplies the MCU and sensor circuitry, while V Bat 2 supplies the motors. Having separate batteries helps to isolate the higher, noisier current draw of the motors from the sensitive analog signal path. On the SMT side, vias were added to tie components into V Bat 1, while V Bat 2 was routed exclusively to the motor drive circuitry. Trace width on the V Bat 2 net was widened to 1.0 mm to reduce trace resistance and limit IR drop under the motors' higher current demand. V Bat 1 traces were left at a narrower default width, as the MCU and sensor loads draw minimal current. Following routing, the design was verified using the DRC checker and returned no errors or warnings.

What I Learned This Week

This week I learned how to set up interrupts in bare metal. Initially, I had problems with the motors overshooting the hall effect sensors and not stopping in time. I thought I would need hardware interrupts, so I learned how to set them up, but I ultimately fixed the problem by coding some software flags instead.

AI Interactions

I used AI to help make my IoT logger more visually appealing. I was already getting all the data through the FTDI cable, but used AI to make it look better. I also used AI to proofread my code and check for bugs or errors before I compiled and uploaded it.