ChatGPT

Инструкция по модернизации солнечного трекера на Arduino UNO (с одним кабелем CAT5)

1. Краткое описание проекта и цели модернизации

Данный проект представляет собой двухосевой **солнечный трекер** на базе Arduino UNO. Устройство автоматически поворачивает небольшую солнечную панель за солнцем, используя два сервопривода (горизонтальное вращение и наклон). В исходной реализации датчики освещенности (фоторезисторы) и приводы были подключены к Arduino множеством отдельных проводов. Цель модернизации – **обновить конструкцию**, сделав ее более надежной и удобной за счет следующих улучшений:

- **Минимизация проводов:** заменить пучок кабелей между уличным модулем (панель и датчики) и комнатным модулем (контроллер) **одним кабелем CAT5 с разъемами RJ45**. Это упростит прокладку и повысит надежность соединений.
- **Переход на цифровые интерфейсы:** вместо передачи аналоговых сигналов по длинным проводам использовать шину I^2C для датчиков и управления – так данные будут передаваться в цифровом виде, менее подверженном шумам на расстоянии.
- **Повышение надежности и снижение помех:** использование витой пары CAT5 позволит правильно спарить сигнальные линии с землей/питанием для уменьшения перекрестных наводок [1]. Также сокращение числа соединений снизит риск обрывов и окисления контактов на улице.
- **Снижение энергопотребления:** добавить в прошивку функции энергосбережения – отключение питания сервоприводов, перевод Arduino в спящий режим при простое, отключение подсветки дисплея ночью и т.п. Это увеличит время работы от аккумулятора.
- **Улучшение схемы питания:** перенести контроллер заряда аккумулятора внутрь помещения, чтобы чувствительная электроника находилась в стабильных условиях, а на улице остались только необходимые элементы. Одним кабелем будет передаваться и питание 5 В для датчиков/приводов, и напряжение от солнечной панели (~6 В) на зарядный модуль.

В результате модернизации солнечный трекер станет более **компактным в проводке**, устойчивым к внешним воздействиям и оптимизированным по питанию, сохранив при этом функциональность слежения за солнцем.

2. Структура системы: модуль внутри дома и модуль на улице

Для повышения надежности система разделена на две части (зоны), соединённые между собой одним кабелем:

- **Домашний модуль (внутри помещения):** Здесь расположены основные управляющие и силовые компоненты, защищенные от погоды. Внутри дома устанавливаются плата Arduino UNO (микроконтроллер), **модуль зарядки аккумулятора** (контроллер заряда от солнечной

панели с повышающим преобразователем для 5 В), собственно **аккумулятор** (например, Li-Ion 18650), а также **LCD-дисплей** для индикации и элементы управления (кнопка). Arduino получает питание от аккумулятора (через повышающий преобразователь до 5 В) и управляет всем устройством.

- **Уличный модуль (на наружной части):** Снаружи находятся элементы, непосредственно взаимодействующие с окружающей средой. К ним относятся: **солнечная панель** (фотоэлектрический модуль ~6 В), установленная на подвижной раме; два **сервопривода** (панорамирование и наклон панели); датчики освещённости – например, четыре фоторезистора (LDR) для определения положения солнца; цифровой **датчик освещённости BH1750** для измерения уровня света (люкс); **датчик температуры/влажности DHT11** для мониторинга окружающей среды. В уличном модуле также находятся вспомогательные электронные модули: **АЦП ADS1115** (для считывания нескольких аналоговых датчиков света) и **PCA9685** (широтно-импульсный контроллер для управления сервоприводами). Эти микросхемы облегчают передачу данных по цифровой шине и сокращают количество проводов.

Обе части соединяются единственным кабелем витая пара CAT5, по которому передаются сигналы и питание. На рисунке ниже показан пример конструкции солнечного трекера (для наглядности все компоненты расположены на общей раме):

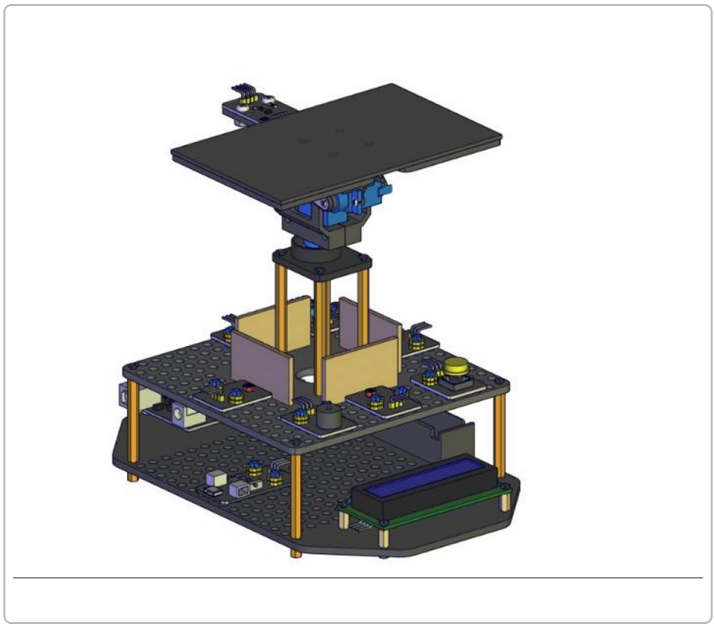

Рис. 1: Пример конструкции солнечного трекера. Верхняя платформа с солнечной панелью и сервоприводами соответствует уличному модулю, нижняя плата с электроникой и дисплеем – домашнему модулю.

В реальной установке уличный модуль будет размещен вне помещения (на крыше, стене и т.д.), а домашний модуль – внутри (например, в комнате или техническом шкафу). Один кабель RJ45 проходит от дома на улицу, что значительно упрощает монтаж по сравнению с пучком отдельных

проводов. Далее подробно рассмотрены схема такого подключения и пошаговая модернизация устройства.

3. Архитектура соединений через кабель CAT5

Основная идея: использовать один восьмижильный кабель CAT5 (витая пара) для связи между Arduino и всеми внешними устройствами. По этому кабелю будут передаваться: питание 5 В на уличный модуль, цифровые сигналы шины I²C (SDA и SCL) и дополнительный сигнальный провод для DHT11, а также напряжение от солнечной панели обратно в дом на модуль зарядки.

Передача данных по I²C: Шина I²C выбрана для связи с удаленными датчиками и контроллерами (ADS1115, PCA9685, BH1750, а также LCD-дисплей внутри). I²C – двухпроводной интерфейс (линия данных SDA и линия тактирования SCL), позволяющий подключить несколько устройств параллельно. Все внешние модули будут подключены к тем же SDA и SCL, что и Arduino UNO (на UNO это пины A4 = SDA и A5 = SCL). Таким образом, вместо нескольких аналоговых входов и PWM-выходов, идущих по длинным проводам, мы проводим **только две линии I²C**. Это упрощает кабель и снижает влияния помех, так как сигнал цифровой и приемопередача помехоустойчивая (любой шум, не превышающий логических порогов, не исказит данные).

Передача питания 5 В: Внутренний модуль (дом) содержит аккумулятор и повышающий преобразователь, формирующий стабильные ~5 В для питания Arduino и всей периферии. Эти 5 В по одной из жил кабеля отправляются на уличный модуль, где питают датчики и сервоприводы. Общий провод (земля, GND) соединяет все компоненты – несколько жил кабеля будут задействованы под землю для надежности. Таким образом, не нужно отдельное питание на улице – вся электроника питается от одного источника внутри дома.

Обратная передача от солнечной панели: Солнечная панель (на ~6 В холостого хода) подключается к другой жиле кабеля, чтобы ее напряжение поступало обратно в дом на модуль заряда. В зарядном модуле это напряжение используется для зарядки аккумулятора. Благодаря этому контроллер заряда и сам аккумулятор можно держать внутри, не подвергая их воздействию температур и влаги, а на улице – только панель. Кабель витая пара способен передавать ток зарядки (порядка сотен мА) на небольшое расстояние без существенных потерь.

Использование витой пары: Очень важно правильно распределить сигналы по витым парам кабеля для снижения помех. Нельзя пускать SDA и SCL по двум проводникам одной скрученной пары – это вызовет сильные перекрестные помехи (витая пара будет работать как трансформатор и одна сигнальная линия наводит шум на другую) [1]. Вместо этого следует комбинировать в паре **один сигнальный провод с проводом постоянного потенциала** (землей или питанием). В нашей схеме SDA будет скручен с землей, а SCL – с проводом питания +5 В. Такое сочетание уменьшит влияние импульсов SCL на SDA и наоборот, поскольку земля и питание имеют относительно стабильный потенциал [2]. Дополнительный сигнальный провод (для DHT11) также по возможности скручивается с землей. Таким образом, мы задействуем витые пары кабеля оптимальным образом:

- Одна витая пара: **SCL + 5В** (тактирование I²C вместе с проводом питания).
- Вторая витая пара: **SDA + GND** (линия данных I²C вместе с землей).
- Третья пара: **DHT11 + GND** (линия цифрового датчика с землей).
- Четвертая пара: **Панель + GND** (линия от солнечной панели с землей).

В результате сигнальные линии отделены друг от друга и защищены от взаимных наводок. Имея несколько проводов GND, мы снижаем сопротивление общего провода (параллельное соединение) и обеспечиваем качественный возврат тока.

> **Примечание:** Длина кабеля I^2C в этой реализации ограничена некоторыми метрами (порядка 5–10 м) без специальных мер. Если требуется большая дистанция (до 20 м и более), применяют специальные расширители шины (например, P82B715 или дифференциальные драйверы) [3]. В рамках данного проекта предполагается, что расстояние сравнительно невелико, и на скорости I^2C 100 кГц связь работает устойчиво. При очень длинном кабеле можно снизить частоту шины (например, до 50 кГц) для повышения надежности.

Токовая нагрузка кабеля: Кабель CAT5 сделан из проводников сечением ~24 AWG, каждый может безопасно нести до ~0.5 А тока [4]. Это достаточно для питания пары небольших сервоприводов и датчиков. Тем не менее, чтобы снизить падение напряжения, мы распределяем ток возврата по нескольким жилам (несколько проводников GND) и используем конденсатор на конце (подробно об этом – в разделе 6). В сумме, при токах в сотни миллиампер, падение напряжения на длине 5–10 м будет небольшим (доли вольта).

Итак, архитектура соединений такова: один конец кабеля подключен к Arduino и модулю питания внутри, другой – к плате с датчиками/приводами снаружи. Ниже описано, как именно распаять разъем RJ45 (категория 5, обжим по стандарту T568B) под наши нужды.

4. Распиновка кабеля и разъема RJ45 (стандарт T568B)

Стандарт **T568B** определяет порядок цветов проводников в витой паре при обжиме коннектора RJ45. Очень важно, чтобы оба конца кабеля были обжаты одинаково (прямой кабель), тогда цвета проводов совпадут по номерам контактов. Ниже приведена схема использования пинов RJ45 в данном проекте:

- **Pin 1 – Бело-оранжевый провод:** SDA (линия данных I^2C).
- **Pin 2 – Оранжевый:** GND (земля, общий минус).
- **Pin 3 – Бело-зелёный:** DHT11 Data (цифровой выход датчика температуры/влажности).
- **Pin 4 – Синий:** SCL (линия тактового сигнала I^2C).
- **Pin 5 – Бело-синий:** +5V (питание 5 В, подаваемое на уличный модуль).
- **Pin 6 – Зелёный:** GND (земля).
- **Pin 7 – Бело-коричневый:** Solar Panel + (плюсовой вывод солнечной панели, ~6 В).
- **Pin 8 – Коричневый:** GND (земля).

Обратите внимание, что **контакты 2, 6 и 8** все используются как земля и соединены между собой (на обеих сторонах кабеля). Это означает, что в кабеле три провода параллельно служат общим проводом GND – такой прием уменьшает сопротивление земли и полезен для токов от сервоприводов. Аналогично, для питания 5 В мы используем контакт 5 (одного провода достаточно для заявленной нагрузки, при необходимости можно был бы задействовать два).

При обжиме кабеля убедитесь, что придерживаетесь схемы T568B на обоих концах (порядок цветов от Pin1 до Pin8: бело-оранжевый, оранжевый, бело-зелёный, синий, бело-синий, зелёный, бело-

коричневый, коричневый). Распайка должна соответствовать вышеописанной функции каждого провода. На уличной стороне эти провода идут к соответствующим модулям, на домашней – к Arduino, аккумулятору и модулю заряда.

> **Совет:** После обжима или пайки соединителей **проверьте мультиметром** правильность распиновки – прозвоните каждый контакт RJ45 с его выводом на плате. Особенно важно убедиться, что линии питания и панели не перепутаны с данными. Также убедитесь в отсутствии коротких замыканий между соседними пинами.

5. Подключение всех компонентов в обновленной системе

На данном этапе рассмотрим, как каждый компонент проекта интегрируется в новую архитектуру (через I^2C и единый кабель). Ниже перечислены основные внешние модули и их подключение к Arduino UNO:

ADS1115 – АЦП для аналоговых датчиков света (фоторезисторов)

Модуль **ADS1115** – это 16-битный 4-канальный аналог-цифровой преобразователь с интерфейсом I^2C. Он используется для опроса нескольких **фоторезисторов** (LDR), которые служат датчиками освещенности для определения положения солнца. Ранее эти LDR могли быть подключены напрямую к аналоговым входам Arduino (A0–A3). После модернизации они подключаются к ADS1115, который оцифровывает их показания и передает в Arduino по I^2C.

- **Соединение LDR с ADS1115:** Каждый из 4-х фотоэлементов подключается к своему входному каналу ADS1115 (A0, A1, A2, A3). Как правило, фоторезистор используется в составе делителя напряжения (в модуле Photoresistor Module уже есть резистор – выход с него и идет как аналоговый сигнал). Выходы модулей фоторезисторов, которые раньше шли на A0–A3 Arduino, теперь подключаются к входам A0–A3 ADS1115. Питание фотодатчиков (обычно 5 В) подается от линии +5V кабеля, земля общая GND.
- **Подключение ADS1115 к шине:** Модуль ADS1115 имеет выводы SDA, SCL, VCC, GND. Соедините SDA модуля с линией SDA кабеля (Pin1 RJ45), SCL – с линией SCL (Pin4 RJ45). VCC ADS1115 подключается к +5V (Pin5 RJ45), GND – к любому общему проводу GND (Pin2/6/8 RJ45). На стороне Arduino SDA и SCL приходят на соответствующие пины A4 и A5 UNO.
- **Адрес и питание:** Убедитесь, что адрес ADS1115 не конфликтует с другими I^2C-устройствами (по умолчанию 0x48, это обычно свободно). Питание 5 В подходит для модуля (он поддерживает 2-5В).
- **Проверка:** После подключения стоит запустить скетч-сканер I^2C, Arduino должен обнаружить устройство по адресу 0x48. Теперь можно считывать значения с каналов ADS вместо analogRead. Благодаря 16-битному разрешению (против 10-бит у Arduino) повышается точность измерения освещенности.

PCA9685 – контроллер PWM для сервоприводов

PCA9685 – это 16-канальный ШИМ-контроллер с I^2C-интерфейсом, используемый в проекте для управления сервоприводами. Он позволяет разгрузить Arduino: ШИМ-сигналы для серв формируются аппаратно на самом PCA9685, а Arduino лишь задает целевую ширину импульса по I^2C.

В исходной схеме сервоприводы, вероятно, подключались к PWM-выходам Arduino (D9, D10 или др.) и управлялись библиотекой Servo. Теперь они будут подключены к выходам PCA9685.

- **Подключение PCA9685 к шине I²C:** Соедините вывод SDA контроллера с линией SDA кабеля (Pin1 RJ45), SCL – с линией SCL (Pin4 RJ45). Питание модуля PCA9685: вывод VCC (логика) подключите к +5 В (Pin5 RJ45), GND – к общему проводу (Pin2/6/8 RJ45). Этот модуль обычно допускает работу логики от 3.3–5 В, и 5 В ему подходит, но **уточните на вашем модуле** (как правило, на платках PCA9685 от Adafruit есть пин VCC для логики и отдельный разъем V+ для питания серв).
- **Питание сервоприводов (V+):** Обратите внимание, на плате PCA9685 обычно есть отдельный вывод или клемма для питания нагрузки (обозначается V+), который питает непосредственно сервоприводы. Его тоже нужно подключить к линии +5 В, чтобы сервоприводы получали питание. И V+ и VCC можно запитать от одного источника 5 В. На плате может быть перемычка (джампер) для соединения VCC и V+ вместе – проверьте документацию. Если перемычки нет, подайте 5 В на клемму V+ отдельно. **Общий провод GND должен быть общим** между PCA9685, сервами и Arduino.
- **Подключение сервоприводов:** Освободите сигнальные провода сервоприводов от пинов Arduino и подключите их к выходным каналам PCA9685. Например, горизонтальный сервопривод на канал 0, вертикальный – на канал 1 (точный номер канала не критичен, но нужно учесть в коде). На PCA9685 3-pin коннекторы: обычно крайний пин – GND (черный/коричневый провод серво), середина – V+ (красный провод серво, 5 В), верхний – сигнал (желтый/оранжевый провод серво). Вставьте разъемы серв в соответствующие каналы. Если клеммника нет, придётся паять провода.
- **Адрес PCA9685:** По умолчанию 0x40 – убедитесь, что он уникален на шине. Если совпадает с другим (маловероятно), измените адрес пайкой A0–A5.
- **Проверка сервоприводов:** После подключения подайте питание и с помощью скетча установите с помощью PCA9685 сервопривод в разные положения (например, с использованием библиотеки Adafruit PWMServoDriver). Сервы должны реагировать. Убедитесь в правильной ориентации: при изменении управляющего значения сервопривод двигается без рывков. Теперь Arduino может позиционировать панель, отправляя команды на PCA9685 по I²C, а не напрямую ШИМ-сигналами.

Цифровой датчик освещенности BH1750

Для измерения абсолютного уровня освещенности (люксы) используется модуль **BH1750**. Это цифровой датчик с интерфейсом I²C, измеряющий освещенность и выдающий значение в люксах. Он дополняет фоторезисторы: по BH1750 можно, например, определить наступление ночи или облачность (общий уровень света), тогда как LDR используются больше для нахождения направления на солнце по разности освещенностей.

- **Подключение BH1750:** Модуль BH1750 имеет 4 вывода: VCC, GND, SDA, SCL (и адресной вход ADDR, обычно не трогаем). Подключите SDA к линии SDA кабеля (Pin1 RJ45), SCL – к SCL (Pin4 RJ45). VCC подключите к +5 В (Pin5 RJ45) – большинство модулей BH1750 имеют встроенный регулятор под 5 В и логический уровень 3.3 В/5 В совместим (проверьте: если модуль без регулятора, давать не более 3.3 В, но обычно китайские модули BH1750 работают от 5 В). GND – к общему проводу кабеля.
- **Адрес BH1750:** Обычно 0x23 (при ADDR = 0) или 0x5C (при ADDR = VCC). Удостоверьтесь, что адрес не конфликтует – 0x23 отлично уживается с ADS1115 и PCA9685.

- **Расположение:** Разместите BH1750 так, чтобы его фотодатчик открыт небосводу (не закрыт тенью от панели). Часто его ставят рядом с панелью, направленным в ту же сторону, чтобы измерять освещенность того же участка неба, что и панель.
- **Использование:** В коде достаточно инициализировать датчик (например, библиотекой BH1750) и опрашивать метод .readLightLevel() для получения люкс. Эти данные можно показывать на дисплее и использовать для решений: например, если освещенность ниже порога, переводить систему в спящий режим (ночь).

Датчик температуры и влажности DHT11

В системе используется простой климатический датчик **DHT11**, измеряющий температуру и влажность. Он нужен в первую очередь для мониторинга условий (влажности, температуры воздуха), возможно, для отображения на дисплее или принятия решений (например, не двигать панель при экстремальном морозе или жаре – хотя DHT11 не суперточен, это скорее информационный параметр).

- **Подключение DHT11:** У DHT11 три вывода: VCC, Data, GND (иногда 4 с NC). Подключите VCC DHT11 к линии +5 В (Pin5 RJ45), GND – к общему GND (Pin2/6/8 RJ45). Вывод Data DHT11 соедините с выделенной линией DHT-data в кабеле – это Pin3 RJ45 (бело-зеленый). На стороне Arduino этот провод должен подключиться к цифровому входу. Выберите свободный цифровой пин, например **D7 на Arduino UNO** (можно любой пригодный GPIO). То есть, на домашнем модуле бело-зеленый провод от RJ45 приходит, и вы его соединяете с пином D7 Arduino.
- **Pull-up резистор:** Для DHT11 необходим подтягивающий резистор (около 10 kΩ) с линии Data к питанию, если вы используете «голый» датчик. Однако большинство модулей DHT11 уже имеют встроенный резистор. Проверьте ваш модуль: если на маленькой платке DHT11 есть резистор между Data и VCC, значит все ок. Если нет, подключите внешний резистор ~10k между Data и +5 В на стороне датчика.
- **Кабель и помехи:** DHT11 – относительно медленный протокол (единицы кГц), он обычно хорошо работает на длине несколько метров. Скрученная с землей пара (Pin3–6) должна обеспечить устойчивую передачу. Если кабель очень длинный (более 5 м), а DHT11 дает сбои, можно снизить скорость опроса (в библиотеке DHT по умолчанию и так задержки).
- **Использование:** В коде Arduino с помощью библиотеки (например, <DHT.h>) опрашивайте DHT11 на выбранном пине (D7). Период опроса DHT11 – не чаще 1 раза в ~2 секунды (это ограничение датчика). Результаты – температура (°C) и влажность (%) – можно выводить на дисплей. DHT11 не очень точный, но даст общее представление (погрешность ~2°C и ~5%RH).

LCD-дисплей 16x2 с интерфейсом I²C

Для отображения информации (например, текущего состояния, показаний датчиков, напряжения батареи) используется **символьный LCD-дисплей 16x2** с I²C адаптером (на базе контроллера PCF8574). Такой дисплей значительно упрощает вывод данных. Он устанавливается внутри дома, рядом с Arduino, чтобы владелец мог наблюдать показания.

- **Подключение LCD:** Плата I²C модуля дисплея имеет 4 пина: VCC, GND, SDA, SCL. Их нужно подключить к Arduino UNO (внутри домашнего модуля). Поскольку Arduino и дисплей оба внутри, можно напрямую соединить: VCC к 5В Arduino, GND к GND Arduino, SDA к A4, SCL к A5. Альтернативно, можно подключить его к ответной колодке RJ45 на домашней стороне: той же

шине SDA (Pin1) и SCL (Pin4), что и уличные устройства, плюс питанию 5В (Pin5) и GND. Так или иначе, дисплей висит на той же I²C-шине, только физически находится рядом с Arduino.

- **Адрес дисплея:** Популярные адреса адаптеров 0x27 или 0x3F. Уточните адрес (например, сканером I²C). Если конфликтует с чем-то (маловероятно), можно поменять адрес пайкой (3 джампера A0-A2 на платке PCF8574).
- **Использование:** В скетче используйте библиотеку LiquidCrystal_I2C (или аналогичную) для управления дисплеем. Инициализируйте, указав адрес и размеры, далее можно выводить строки текста. Дисплей будет показывать, например, напряжение батареи, температуру/влажность, уровень освещенности, режим работы и пр. Формат и содержание – на ваше усмотрение.
- **Подсветка:** Модуль позволяет программно выключать/включать подсветку (методы `backlight()` / `noBacklight()`), так как светодиод дисплея запитан через транзистор с PCF8574. Мы будем использовать это для энергосбережения (например, выключать подсветку ночью или когда батарея разряжена).

Кнопка управления (User Button)

В комплекте имеется **модуль кнопки** – его можно задействовать для пользовательского ввода. Например, по нажатию кнопки можно переключать режимы (ручной/авто), просыпать устройство из спящего режима, либо листать страницы отображения на LCD.

- **Подключение кнопки:** Обычно кнопочный модуль имеет 3 пина (VCC, GND, OUT) с внутренним подтягивающим резистором, или 2 пина (если просто кнопка). Самый простой способ – использовать внутренний **pull-up** Arduino. Подключите один вывод кнопки к выбранному цифровому входу Arduino, другой – к GND. Например, подключите кнопку между пином **D3** Arduino UNO и землей. В скетче этот пин настроим как `INPUT_PULLUP`, так что лог.0 будет при нажатии.
- **Использование:** В коде обработчик нажатия может выполнять определенные действия. К примеру, если устройство спит, можно настроить пробуждение Arduino по прерыванию от кнопки (D2 или D3 позволяют внешнее прерывание). Либо в активном режиме – переключение информации на дисплее (показывать разные параметры по очереди).
- **Проверка:** Убедитесь, что при нажатии на кнопку на выбранном пине регистрируется LOW. Если используется внешний модуль с подтяжкой к VCC, возможно, нужно подключать наоборот (OUT модуля к входу, GND к GND, VCC к 5В).

(Подключение звукового индикатора (пассивного буззера) и LED-модуля, если они есть, не рассматривается подробно, так как в требованиях не упомянуты. При желании можно подключить их к свободным пинам для сигнализации – например, мигать LED или пищать буззером при низком заряде.)

Мониторинг напряжения батареи (через делитель напряжения)

Очень важная часть – **контроль уровня заряда аккумулятора**. Мы реализуем это, измеряя напряжение батареи с помощью встроенного АЦП Arduino. Поскольку аккумулятор Li-Ion (номинал ~3.7 В, максимум ~4.2 В) подключен внутри к зарядному модулю, Arduino может считывать его

напряжение. Напрямую 4.2 В можно подать на аналоговый вход (это ниже 5 В), но для безопасности и для масштабирования под разные ситуации лучше использовать **резистивный делитель**.

- **Делитель напряжения:** Соберите простой делитель из двух резисторов, например по **100 кΩ** каждый (высокое сопротивление, чтобы не разряжать батарею лишним током). Подключите резисторы последовательно между плюсом батареи и GND: верхний конец первого резистора к плюсу батареи (точка BAT+ на модуле заряда), соединение между резисторами – к аналоговому входу Arduino (например **A0**), нижний конец второго резистора – к GND. Таким образом, на A0 будет половина напряжения батареи (при равных резисторах). Если батарея 4.2 В, на A0 получится ~2.1 В.
- **Калибровка:** В коде нужно будет умножать считываемое напряжение на коэффициент делителя (например, ×2, если оба резистора равны). Можно подобрать резисторы для удобного коэффициента или просто математически скорректировать. Учтите, что АЦП Arduino по умолчанию использует опорное 5 В (Vcc), точность ~10 бит. Это даст приемлемую оценку напряжения (с шагом ~0.005 В).
- **Альтернатива ADS1115:** Если свободен канал ADS1115 и хочется большей точности, можно завести батарею на ADS (через делитель, чтобы не превысить Vref ADS). Но в нашем случае и встроенный АЦП справится.
- **Безопасность:** Обязательно убедитесь, что делитель подключен правильно – случайная подача полного напряжения батареи на A0 без делителя может вывести пин из строя. Также, если аккумулятор может быть >5 В (не в нашем случае с одной Li-Ion), делитель обязателен!
- **Использование:** В скетче измерение батареи можно делать не слишком часто (например, раз в 10 секунд или минуту), т.к. напряжение меняется медленно. Значение можно отображать на дисплее (в вольтах или проценте заряда – можно примерно приравнять 4.2 В = 100%, 3.7 В ~ 20%, 3.5 В ~ 0% для Li-Ion).

Подключив все перечисленные модули согласно описанию, вы получите полностью объединенную по I^2C схему. Осталось правильно всё собрать и проверить поэтапно, о чем речь пойдет далее.

6. Питание сервоприводов через CAT5 и установка конденсатора на стороне улицы

Питание сервоприводов по витой паре требует отдельного внимания, поскольку сервы – самые "прожорливые" и шумные устройства в системе. Когда сервопривод резко стартует или меняет направление, он может потреблять импульс тока сотни миллиампер, что приводит к просадке напряжения на проводах. Чтобы минимизировать проблемы (падение напряжения, дрожание сигнала, перезагрузка контроллера), необходимо предпринять следующие меры:

- **Использование нескольких проводников для земли:** В нашем кабеле сразу три жилы выделены под GND. Возвратный ток от сервоприводов распределяется между ними, уменьшая падение на каждом. Это повышает стабильность напряжения питания на удаленном конце.
- **Толщина провода и ток:** Как упоминалось, один проводник CAT5 выдерживает ~0.5 А [4]. Два микро-сервопривода (например, SG90) обычно потребляют ~150–300 мА каждый при движении без нагрузки, и кратковременно до 700–800 мА при рывке или заклинивании. Таким образом, ток в ~0.5–1 А по проводу 5В может иметь место на доли секунды. Это ближний к пределу ток для 24 AWG, но допустимый, тем более что средний ток меньше. Благодаря конденсатору (см. ниже) пиковый ток из кабеля сглаживается. В крайнем случае, можно было

бы пустить +5 В параллельно по двум жилам (например, Pin5 и Pin4, но у нас Pin4 занят SCL). Практика показывает, что одного провода 5В достаточно на короткой линии, а вот **конденсатор обязателен**.

- **Электролитический конденсатор на стороне нагрузок:** На плате уличного модуля, в точке где к питанию +5 В подключены сервы и другие устройства, следует установить крупный электролитический конденсатор емкостью **1000–2200 мкФ (не менее 16 В)**. Подключите его между +5 В и GND (следя за полярностью!). Этот конденсатор будет как локальный резервуар энергии, компенсируя просадку напряжения в моменты старта сервоприводов. Практика робототехники подтверждает, что дополнительный большой конденсатор (1000 µF и более) на линиях питания серв сглаживает провалы напряжения [5] .
- **Дополнительные конденсаторы:** Помимо большого электролита, полезно иметь керамические конденсаторы малой емкости (0.1 мкФ) возле потребителей (на плате PCA9685, на модулях датчиков) – они фильтруют высокочастотный шум. Как правило, эти конденсаторы уже есть на модулях (например, на PCA9685 обычно припаян 10 мкФ + 0.1 мкФ).
- **Развязка питания серво и логики:** Убедитесь, что питание сервоприводов (V+ на PCA9685) действительно параллельно питанию логики 5В и аккумулятора, без дополнительных регуляторов по пути. Если в системе есть отдельный DC-DC для сервоприводов – отлично, но в нашем случае все питается от одного 5 В источника. Поэтому все компоненты «сидят» на одной шине 5 В, и конденсатор улучшает ситуацию для всех.
- **Проверка под нагрузкой:** После сборки, когда все подключено, **измерьте напряжение** на удаленном конце (на клемме PCA9685 V+ и GND) при работе сервоприводов. В идеале, провалы не должны опускать напряжение ниже ~4.5 В, иначе возможно нестабильная работа логики. Если просадка значительна, рассмотрите укорочение кабеля, использование провода большего сечения или понижение скорости/частоты перемещения панели.

Таким образом, питание сервоприводов по CAT5 реализовано и стабилизировано: параллельные провода GND снижают сопротивление, большой конденсатор сглаживает пики потребления, а разумное ограничение скорости движения панели (плавные повороты) предотвратит резкие броски тока. Все это защитит Arduino от перезагрузок и удержит напряжение для датчиков.

(Для справки: в промышленных решениях питания по витой паре (POE) для больших токов объединяют две пары на + и две на –. В нашем случае потребности не столь высоки, но мы фактически сделали похожее – объединили несколько проводов под минус.)

7. Этапы сборки и проверки работоспособности

Теперь, когда мы разобрали все соединения, опишем **пошаговый план модернизации**. Рекомендуется выполнять изменения поэтапно, **проверяя работу на каждом шаге**, чтобы локализовать возможные проблемы.

Ниже перечислены основные этапы:

1. **Подключение датчиков через модуль ADS1115 (проверка показаний):** Сначала реализуйте переход от прямого подключения фоторезисторов к Arduino на схему с ADS1115. Для этого: отключите провода LDR от аналоговых пинов A0–A3 UNO и подключите их к входам A0–A3 модуля ADS1115. Подайте питание на ADS1115 (5В и GND) от Arduino. Подключите SDA и SCL ADS1115 к A4/A5 Arduino (пока можно макетными проводами, без длинного кабеля). Загрузите

в Arduino тестовый скетч: инициализация Wire, опрос ADS1115 (например, с использованием библиотеки Adafruit_ADS1X15) – чтение всех 4 каналов и вывод значений в Serial Monitor. Убедитесь, что значения меняются при засветке каждого датчика (можно поочередно закрывать фотоэлементы рукой и смотреть изменение). При успешном результате – датчики работают через I^2C. Это значит, что переход на цифровой интерфейс удался, и можно двигаться дальше.

2. **Перевод управления сервоприводами на контроллер PCA9685:** Отключите сигнальные провода серв от пинов PWM Arduino (если они были подключены, например D9/D10). Подключите сервоприводы к выходам PCA9685 (как описано в разделе 5). Сначала можно собрать схему PCA9685 рядом с Arduino для испытания (опять же, без длинного кабеля). Подключите модуль PCA9685 к 5 В, GND, SDA, SCL Arduino. Установите библиотеку Adafruit PWM Servo Driver. Напишите небольшой скетч для проверки: инициализируйте PCA9685 (`pwm.begin(); pwm.setPWMFreq(50)` для 50 Гц), затем в loop установите разные ШИМ на каналах 0 и 1 (например, крайние положения 0° и 180°). Проверьте, что сервоприводы движутся корректно в весь диапазон. Если движение происходит рывками или не наблюдается – отладьте питание (возможна просадка: провода питания к PCA9685 должны быть надежны) и соединения SDA/SCL (убедитесь, что адрес 0x40 виден сканером I^2C). При успешном тесте – Arduino научился управлять сервами через PCA9685. Скорректируйте основной скетч трекера: замените вызовы `servo.write(angle)` на соответствующие команды `pwm.setPWM(channel, ...)` для PCA9685. На этом этапе у вас все еще может быть Arduino, ADS1115, PCA9685, BH1750, DHT11 *без* длинного кабеля – все на столе.

3. **Сканирование и проверка шины I^2C с подключенными модулями:** Перед финальным монтажом убедитесь, что **все I^2C-устройства видны на шине**. Подключите **одновременно** ADS1115, PCA9685, BH1750, а также LCD-дисплей к Arduino (пока короткими проводами). Запустите скетч–сканер I^2C (вы легко найдете код сканера, который перебирает адреса 0–127 и выводит найденные). Ожидается обнаружение следующих адресов: ADS1115 (0x48), PCA9685 (0x40), BH1750 (0x23) и LCD (например, 0x27). Если какой-то не определяется – проблема с подключением данного устройства (проверьте питание и линии). Убедившись, что шина в порядке и адреса конфликтов не имеют, можно переходить к подключению через кабель.

4. **Переход на соединение через CAT5 + разъемы RJ45:** Теперь наступает ответственный момент – задействовать заранее подготовленный кабель CAT5 для связи между блоками. Возьмите кабель нужной длины, обожмите разъемы RJ45 (или используйте готовый патч-корд). **Важно:** На стороне Arduino (домашней) лучше использовать RJ45-гнездо (панельный разъем или ответный разъем), к которому припаяны провода к Arduino, модулю питания, etc., а на стороне уличного модуля – аналогичный разъем, соединенный с датчиками/приводами. Либо можно напрямую обжать кабель на нужные провода внутри боксов. В любом случае, реализуйте соединения согласно распиновке, описанной в разделе 4. Например: бело-оранжевый провод кабеля, приходящий на домашний модуль, соедините с A4 Arduino (SDA), оранжевый – с GND Arduino, бело-зеленый – к пину D7 (вход DHT11), синий – к A5 Arduino (SCL), бело-синий – к выходу +5 В преобразователя (и к 5В Arduino), зеленый – к общему GND, бело-коричневый – к плюсу солнечной панели на зарядном модуле, коричневый – к GND (минус панели и общая земля). На уличной стороне выполните аналогично: те же цвета к соответствующим устройствам (удобно заранее пометить или прозвонить). После соединения **еще раз прозвоните** кабель – соответствие пинов RJ45, отсутствие перепутываний.

5. Когда всё готово, подключите системы через кабель. Выполните снова **I²C-сканирование** из пункта 3, но уже сигналами, проходящими по кабелю. Все устройства должны по-прежнему обнаруживаться. Затем проверьте датчик DHT11 – запросите у него данные (возможно, добавив временно вывод на Serial для проверки). Если DHT не отвечает, возможно сигнал искажается – убедитесь, что витая пара для DHT используется правильно (должна быть с GND). Также проверьте, что питание 5 В доходит до внешних модулей – измерьте мультиметром напряжение между +5 и GND на уличной стороне.
6. **Проверка сервоприводов через кабель:** Испытайте поворот панели командой Arduino – отправьте сигнал на сервоприводы как в шаге 2, наблюдайте движение. Следите, не проседает ли питание (Arduino не перезагружается ли в момент движения серв, LCD не мерцает ли). Если заметны проблемы, двигаемся к разделу 6 (установка конденсатора, хотя вы могли его уже припаять).

7. Если все датчики и приводы корректно функционируют через кабель, можно считать, что аппаратная часть модернизации успешно реализована.

8. **Подключение и калибровка измерения заряда батареи:** На завершающем этапе подключите резистивный делитель к батарее и Arduino (как описано в конце раздела 5). Если у вас модуль зарядки с выводом напряжения батареи (например, метка B+), можно припаяться к нему. Другой конец делителя к A0 Arduino и GND. Напишите небольшой код для теста: читайте `analogRead(A0)` и переводите в напряжение. Проверьте показания, сравнив с реальным напряжением аккумулятора (измерьте мультиметром). Отрегулируйте коэффициент расчета. Далее добавьте этот функционал в основной код: теперь контроллер сможет мониторить батарею.

9. **Защита аккумулятора:** Хотя в задаче это не оговорено, убедитесь, что зарядный модуль имеет отсечку по низкому заряду или хотя бы вы предупредите разряд батареи программно. Можно например при напряжении <3.5 В выключать нагрузки или подавать сигнал (например, мигать LED, гасить дисплей). В нашем случае мы реализуем программные меры (см. раздел 10).

Выполнив последовательно эти шаги, вы получите модернизированный трекер, в котором все изменения протестированы. Теперь остается написать/обновить финальный скетч Arduino, объединяющий в себе работу со всеми новыми модулями и реализующий улучшенные алгоритмы.

8. Схема соединений системы (обзор)

Ниже приведено обобщенное описание получившейся схемы соединений после всех изменений. Это поможет убедиться, что все элементы связаны правильно, и понять «маршрут» сигналов и энергопотоков в системе.

- **Солнечная панель (наружная):** Панель (6 В, 1.5 Вт) подключена двумя проводами к уличному модулю. Плюсовой провод панели идет на бело-коричневый провод кабеля (Pin7 RJ45), минус панели – на общий GND (соединен с коричневым проводом Pin8 RJ45). По этому каналу ток от панели поступает во внутренний модуль для зарядки аккумулятора.
- **Модуль зарядки и питания (внутри):** Внутри дома к паре проводов от панели подключен **контроллер заряда** (например, плата на базе CN3065 или аналогичной схемы, как на рисунке

схемы ниже). Он управляет зарядом аккумулятора 18650. Параллельно аккумулятору подключен **повышающий преобразователь** (например, на микросхеме HX3608), который выдает стабилизированные 5 В (обозначим линию как VDD5). Эти 5 В идут на питание Arduino и всех периферийных устройств. Таким образом, когда солнце есть – панель заряжает аккумулятор и одновременно питает нагрузку, когда солнца нет – питание идет от аккумулятора через преобразователь. На схеме модуль зарядки может называться "Solar USB Charging Module". В нашем проекте важно, что **точка 5 В** (выход преобразователя) подключается к зелено-белому проводу кабеля (Pin5 RJ45, +5В) и раздается по системе.

- **Arduino UNO (внутри):** Питается от линии +5В (через пин 5V или Vin, в зависимости от наличия стабилизатора – предпочтительно напрямую в 5V, минуя встроенный линейный стабилизатор, т.к. у нас уже стабильно 5В). GND Arduino подключен к общей земле (к нескольким проводам RJ45 Pin2,6,8, а также к минусу батареи). Arduino соединен с периферией:
- I^2C-шина: пин A4 (SDA) Arduino идет к бело-оранжевому проводу (Pin1 RJ45), A5 (SCL) – к синему проводу (Pin4 RJ45). Таким образом, шина уходит на улицу и также идет на LCD-дисплей внутри.
- Пин D7 Arduino – подключен к бело-зеленому проводу (Pin3 RJ45) – это вход от DHT11.
- Пин D3 Arduino – подключен к кнопке (внутри, кнопка между D3 и GND, с активированным INPUT_PULLUP).
- Аналоговый вход A0 Arduino – подключен к середине делителя напряжения на батарее (для контроля заряда).
- **LCD-дисплей 16x2 (внутри):** Подключен к Arduino: VCC к 5В, GND к GND, SDA к A4, SCL к A5 (т.е. фактически параллельно Arduino на линии SDA/SCL, внутри домашнего модуля). Он показывает параметры системы.
- **Уличный модуль – общие точки питания:** На уличной плате распределения +5 В (поступающие по кабелю) идут на питание всех устройств: ADS1115, BH1750, PCA9685, DHT11, а также на сервоприводы (через PCA9685 V+). Земля уличного модуля объединена (все GND модулей, минусы сервоприводов и панель – вместе и через три провода возвращаются на землю Arduino/аккумулятора).
- На этой плате между линией +5В и GND установлен электролит **1000 мкФ** (или более) для стабильности, как упоминалось.
- **ADS1115 (улица):** Подключен к линии +5В и GND (питается от домовой 5В через кабель). Его SDA соединен с бело-оранжевым (кабель Pin1), SCL – с синим (Pin4). К входам ADS1115 подключены 4 фотодатчика:
- Фоторезисторы (4 шт) установлены, например, по углам панели для отслеживания солнца. Каждый смонтирован в небольшом модуле с резистором, имеющим выходной аналоговый сигнал. Эти выходы подключены к A0–A3 ADS1115 проводами. Питание модулей LDR (если нужно) также 5В/GND с кабеля.
- **BH1750 (улица):** Подключен к тем же линиям SDA (Pin1) и SCL (Pin4) кабеля, питание 5В (Pin5) и GND. Расположен так, чтобы измерять общий свет.
- **PCA9685 и сервоприводы (улица):** Модуль PCA9685 питается: VCC (логика) от +5В, V+ (силовое для серв) тоже от +5В (кабель Pin5), GND – к общему. Его SDA, SCL подключены к тем же линиям кабеля, что и другие устройства. Сигнальные выходы PCA идут к двум сервоприводам:
- Серво 1 (панель вращение по азимуту) подключен к каналу PCA0 (примерно), серво 2 (наклон) – к каналу PCA1. Питание обоих серв (+5 и GND) берут с разъема PCA9685 (который, в свою очередь, связан с 5В/GND кабеля). После сборки обязательно убедитесь, что **полярность серво** верна: GND серво к GND, +5 к +5, сигнальный к каналу – перепутывание может повредить сервопривод.

- **DHT11 (улица):** Питается от +5В/GND кабеля. Его цифровой выход подключен к зеленому-белому проводу (Pin3 RJ45). Этот сигнал идет по кабелю и поступает на D7 Arduino, как описано выше.

Иными словами, шина I²C теперь **общая для всех устройств** внутри и снаружи, питание 5 В раздается по кабелю, а солнечная панель заряжает батарею по тому же кабелю. Такой интегрированный подход упростил систему: вместо десятка отдельных проводов всего один кабель. Ниже представлен упрощенный схематичный обзор системы:

```
Солнечная панель (6В) --(витая пара)---> [Модуль заряда + Аккум + Boost 5В] --
(5В по кабелю)--> Датчики и сервы
                          |                                          |
                    Аккумулятор 18650                        Arduino UNO + LCD +
кнопка
                          |                                          |
Датчики: [4 x LDR -> ADS1115] -- I2C --\                        //--- I2C (A4/
A5)
         [BH1750 (освещенность)] -- I2C ---- Кабель CAT5 ----//---- I2C шина
         [PCA9685 -> 2 x серво] -- I2C --/                        \\--- I2C
(A4/A5)
         [DHT11] -- один провод --/                               \\--- DHT
(D7)
```

(Примечание: на схеме выше символом // показано соединение одной витой парой, а \ другой – для наглядности разделены I2C и DHT, хотя физически они в одном кабеле.)

Убедившись, что все соединено как описано, можно переходить к программной части – финальной прошивке, реализующей логику трекера и энергосбережение.

9. Финальный скетч Arduino (с учетом всех изменений)

Ниже приведен пример кода для Arduino UNO, который соответствует обновленной аппаратуре. Этот скетч включает: чтение датчиков через ADS1115 и DHT11, измерение освещенности BH1750, управление сервами через PCA9685, отображение информации на LCD, обработку нажатия кнопки, а также некоторые функции энергосбережения (отпускание серв, отключение подсветки и т.п.). Код написан с использованием распространенных библиотек – их нужно установить в среде Arduino IDE (через Library Manager): **Adafruit ADS1X15**, **Adafruit PWM Servo Driver**, **Adafruit Sensor и DHT**, **BH1750** (by clawdevice или similar), **LiquidCrystal_I2C**.

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>       // для ADS1115 (Adafruit_ADS1115 класс)
#include <Adafruit_PWMServoDriver.h> // для PCA9685
#include <BH1750.h>                 // для BH1750
#include <DHT.h>                    // для DHT11
#include <LiquidCrystal_I2C.h>      // для LCD 16x2 (на PCF8574)
```

```
// === Настройки и константы ===
#define DHTPIN 7            // пин, куда подключен DHT11 (D7)
#define DHTTYPE DHT11       // тип датчика
DHT dht(DHTPIN, DHTTYPE);

Adafruit_ADS1115 ads;       // АЦП ADS1115
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(0x40); // PCA9685 (адрес
0x40 по умолчению)
BH1750 lightMeter;
LiquidCrystal_I2C lcd(0x27, 16, 2); // адрес дисплея 0x27 (указать ваш при
необходимости)

// Серво настройки (каналы PCA9685 и ограничение углов)
const int servoPanChannel = 0;   // канал для серво поворота панели по
горизонтали
const int servoTiltChannel = 1;  // канал для серво наклона панели
// Минимальные и максимальные значения ШИМ для сервоприводов (зависят от
конкретных серв)
const int SERVOMIN = 120;  // значение для ~0° (примерно 0.5 мс / 20 мс * 4096)
const int SERVOMAX = 490;  // значение для ~180° (примерно 2.0-2.4 мс импульс)
int panAngle = 90;   // текущий угол поворота (азимут) [0-180]
int tiltAngle = 90;  // текущий угол наклона [0-180]

// Глобальные переменные
bool backlightOn = true;
unsigned long lastLcdUpdate = 0;
unsigned long lastDhtRead = 0;
unsigned long lastServoMove = 0;
float batteryVoltage = 0.0;
bool lowPowerMode = false;
bool sleeping = false;

// === Настройки порогов и режимов ===
const float LOW_BATTERY_THRESH = 3.6;   // порог низкого заряда (вольт)
const float CRITICAL_BATTERY_THRESH = 3.4; // критический уровень батареи
const uint16_t LIGHT_THRESHOLD = 20;    // порог освещенности (люкс) для
определения "ночи"
const unsigned long LCD_UPDATE_INTERVAL = 2000; // обновление дисплея раз в 2
секунды
const unsigned long DHT_READ_INTERVAL = 5000;   // опрос DHT раз в 5 секунд
const unsigned long SERVO_MOVE_INTERVAL =
2000; // интервал обновления положения панели 2 сек

// Функция установки угла сервопривода через PCA9685
void setServoAngle(uint8_t channel, int angle) {
  if (angle < 0) angle = 0;
  if (angle > 180) angle = 180;
```

```
  // Интерполируем угол в значение ШИМ:
  int pwmVal = map(angle, 0, 180, SERVOMIN, SERVOMAX);
  pwm.setPWM(channel, 0, pwmVal);
}

// Инициализация систем
void setup() {
  Wire.begin();
  // Инициализация модулей:
  ads.begin();            // ADS1115
  pwm.begin();
  pwm.setPWMFreq(50);     // 50 Hz для сервоприводов
  lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE);
  dht.begin();
  lcd.init();
  lcd.backlight();
  pinMode(DHTPIN, INPUT_PULLUP); // DHT будет подтянут внутренне (если внешний
не стоит)
  pinMode(3, INPUT_PULLUP);      // кнопка на пине D3, используем внутренний
pull-up
  // Начальное приветствие на LCD
  lcd.print("Sun Tracker");
  lcd.setCursor(0, 1);
  lcd.print("Upgrade OK!");
  delay(1500);
  lcd.clear();
  lastLcdUpdate = millis();
}

// Функция чтения всех сенсоров и обновления глобальных переменных
void readSensors() {
  // Чтение 4-х каналов фоторезисторов с ADS1115 (возвращает 0-32767 при
FSR=6.144V)
  int16_t adc0 = ads.readADC_SingleEnded(0);
  int16_t adc1 = ads.readADC_SingleEnded(1);
  int16_t adc2 = ads.readADC_SingleEnded(2);
  int16_t adc3 = ads.readADC_SingleEnded(3);
  // Перевод в условные единицы (можно сразу использовать как относительные
значения)
  // Чем больше значение, тем ярче свет на соответствующем датчике.

  // Чтение освещенности BH1750
  uint16_t lux = lightMeter.readLightLevel(); // люксы

  // Чтение DHT11 (если прошел интервал)
  static float dhtTemp = 0, dhtHum = 0;
  if (millis() - lastDhtRead >= DHT_READ_INTERVAL) {
    lastDhtRead = millis();
```

```
    float t = dht.readTemperature();
    float h = dht.readHumidity();
    // Проверка валидности (DHT11 может возвращать NaN при сбое)
    if (!isnan(t) && !isnan(h)) {
      dhtTemp = t;
      dhtHum = h;
    }
  }

  // Чтение напряжения батареи (с A0 через делитель 1:2)
  int raw = analogRead(A0);
  float v = raw * (5.0 / 1023.0) * 2.0; // умножаем на 2 (делитель 1/2)
  batteryVoltage = v;

  // Обработка значений для управления:
  // Вычислим усредненное "лево-право" и "верх-низ" освещение:
  int16_t leftLight  = (adc0 + adc2) / 2;   // левый столб (например, adc0 верх-лев, adc2 низ-лев)
  int16_t rightLight = (adc1 + adc3) / 2;   // правый столб фоторезисторов
  int16_t topLight   = (adc0 + adc1) / 2;   // верхние два
  int16_t bottomLight= (adc2 + adc3) / 2;   // нижние два

  // Вычислим ошибки по осям:
  int16_t errorAz = leftLight - rightLight;   // положит. если слева ярче (надо повернуть влево)
  int16_t errorEl = topLight - bottomLight;   // положит. если сверху ярче (надо поднять)

  // Зададим чувствительность:
  const int threshold = 50; // порог нечувствительности, чтобы не дергаться от мелких разниц
  int deltaAz = 0;
  int deltaEl = 0;
  if (errorAz > threshold)        deltaAz = +1;
  else if (errorAz < -threshold)  deltaAz = -1;
  if (errorEl > threshold)        deltaEl = +1;
  else if (errorEl < -threshold)  deltaEl = -1;

  // Режим экономии: если очень низкий общий свет или батарея разряжена
  if (lux < LIGHT_THRESHOLD || batteryVoltage < LOW_BATTERY_THRESH) {
    lowPowerMode = true;
  } else {
    lowPowerMode = false;
  }

  // Если не ночной/энергосберегающий режим – обновляем положение панели
  if (!lowPowerMode) {
    // Обновляем целевые углы, делая небольшой шаг
```

```
    panAngle += deltaAz;
    tiltAngle += deltaEl;
    // Ограничим диапазон 0-180:
    if (panAngle < 0) panAngle = 0;
    if (panAngle > 180) panAngle = 180;
    if (tiltAngle < 0) tiltAngle = 0;
    if (tiltAngle > 180) tiltAngle = 180;
    // Двигаем сервы к новым углам
    setServoAngle(servoPanChannel, panAngle);
    setServoAngle(servoTiltChannel, tiltAngle);
    lastServoMove = millis();
  } else {
    // Если мало света или батарея слаба – переводим сервоприводы в "отпущенное"
состояние
    // Вариант: установить ШИМ такой, чтобы не было импульсов (например, 0 или
отключить PCA9685)
    // Здесь мы просто не обновляем, а можно еще и отключить выходы PCA9685, но
библиотека не дает прямого метода.
    // Можно использовать вывод OE на PCA9685, если он выведен, для отключения
всех каналов.
  }

  // Управление подсветкой LCD и спящий режим:
  if (lowPowerMode && backlightOn) {
    // Если условия низкого питания/света – выключим подсветку
    lcd.noBacklight();
    backlightOn = false;
  } else if (!lowPowerMode && !backlightOn) {
    // Если нормальный режим возобновился – включим подсветку
    lcd.backlight();
    backlightOn = true;
  }

  if (batteryVoltage < CRITICAL_BATTERY_THRESH) {
    // Очень низкий заряд – можно предпринять критичные меры, например:
    // - прекратить любые движения, погасить дисплей
    // - войти в режим сна (будить только по таймеру или кнопке)
    // Здесь лишь отметим переменную (глубокий сон можно реализовать через
watchdog).
    sleeping = true;
  } else {
    sleeping = false;
  }

  // Обновление дисплея с заданным интервалом
  if (millis() - lastLcdUpdate >= LCD_UPDATE_INTERVAL) {
    lastLcdUpdate = millis();
    lcd.setCursor(0, 0);
```

```
    if (lowPowerMode) {
      if (lux < LIGHT_THRESHOLD) lcd.print("Night mode  ");
      else if (batteryVoltage < LOW_BATTERY_THRESH) lcd.print("LowBatt mode ");
      else lcd.print("PowerSave   ");
    } else {
      lcd.print("Tracking    "); // режим слежения
    }
    // Первая строка: режим или статус
    // Вторая строка: можно чередовать инфо по кнопке, но для примера выведем
основные параметры
    lcd.setCursor(0, 1);
    // Например: "Bat:4.01V T:25C"
    char buf[17];
    snprintf(buf, sizeof(buf), "Bat:%.2fV T:%.0fC", batteryVoltage,
dht.readTemperature());
    lcd.print(buf);
    // Здесь одновременно показано напряжение батареи и температура.
    // По кнопке можно было бы переключать на "Hum:..% Lux:...", но для простоты
зафиксировано.
  }
}

// Обработка кнопки (например, для переключения отображения или ручного режима)
void handleButton() {
  static bool lastState = HIGH;
  bool currentState = digitalRead(3);
  if (currentState == LOW && lastState == HIGH) {
    // Кнопка нажата (переход из HIGH в LOW)
    // Для примера: будем по нажатию менять, включена ли подсветка (как
демонстрация)
    if (backlightOn) {
      lcd.noBacklight();
      backlightOn = false;
    } else {
      lcd.backlight();
      backlightOn = true;
    }
    // (Вместо этого можно, например, листать страницы LCD или активировать
ручной режим позиционирования.)
  }
  lastState = currentState;
}

void loop() {
  // Если включен критический спящий режим:
  if (sleeping) {
    // Погасим все, остановим движения
    lcd.noBacklight();
```

```
    backlightOn = false;
    // Отпустим сервы: можно установить 0 на выход PCA9685 через OE:
    // pwm.setPWM(servoPanChannel, 0, 0);
    // pwm.setPWM(servoTiltChannel, 0, 0);
    // Войти в спящий режим контроллера:
    // (не будем реализовывать полностью здесь, но можно использовать AVR
sleep_mode())
    // Будем просто долго спать в цикле, просыпаясь изредка:
    delay(10000);
    // Выходим из функции loop, фактически "дремлем"
    return;
  }

  // Обычный режим:
  readSensors();
  handleButton();

  // Добавим небольшую задержку, чтобы loop не циклился слишком часто
  delay(50);
}
```

(Приведенный код носит иллюстративный характер – возможно, его потребуется адаптировать под конкретные библиотеки и условия. Однако в целом он демонстрирует, как объединить работу всех модулей.)

Пояснения к скетчу:

- В начале подключаются библиотеки для всех используемых модулей. Обратите внимание, что адреса I^2C некоторых устройств (LCD, PCA9685) указаны явно – если у вас они отличаются, скорректируйте.
- В разделе **настроек** определены ключевые константы: пин подключения DHT, каналы сервоприводов PCA9685, калибровка сервоимпульсов (минимум/максимум). Значения `SERVOMIN` и `SERVOMAX` могут потребовать подстройки под ваши сервоприводы: их можно определить экспериментально, подав значения и глядя, что серво покрыл весь механический диапазон панели, но не упирается бесконтрольно.
- В `setup()` проводится инициализация всех модулей: начинается Wire, ADS1115, PCA9685 (с заданием частоты 50 Гц), BH1750, DHT11, LCD. Кнопка D3 настроена на вход с подтяжкой. Пара секунд дается на отображение заставки "Sun Tracker Upgrade OK!".
- Основная логика в `loop()`: сначала проверяется `sleeping` (критически низкий заряд) – в таком случае система фактически останавливается (выключает подсветку, "отпускает" сервы, переходит в простой). Здесь демонстрируется простой вариант – долгий `delay(10000)` вместо полноценного AVR sleep, но в реальной жизни можно использовать `avr/sleep.h` для усыпления и просыпаться по прерыванию от кнопки или таймера.
- Если не `sleeping`, вызывается `readSensors()`, которая:
- Читает все 4 канала ADS1115 (фоторезисторы) – получает их значения (пропорциональны освещенности).
- Читает BH1750 (люксы).

- Читает DHT11 (не чаще, чем раз в 5 секунд, чтобы не нагружать).
- Читает напряжение батареи (A0, пересчитанное умножением на 2).
- Затем на основе значений LDR вычисляются усредненные сигналы слева/справа и сверху/снизу и ошибки `errorAz`, `errorEl`. Если, например, слева светят датчики сильнее, `errorAz` будет положительным – нужно повернуть панель влево (увеличить панорамный угол).
- Если ошибка по оси больше порога `threshold`, код задает `deltaAz` или `deltaEl` = +1 или -1, иначе 0. Это означает, что панель будет перемещаться **пошагово** на 1° в ту или иную сторону, пока разница в освещенностях не снизится до порога. Такой метод предотвращает дрожание при почти уравновешенных датчиках.
- Определяется флаг `lowPowerMode`: если освещенность слишком низкая (`lux < LIGHT_THRESHOLD`, ночь или сумерки) **ИЛИ** напряжение батареи ниже порога `LOW_BATTERY_THRESH`, то включается режим энергосбережения. В этом режиме панель не будет активно следить за солнцем (либо солнца нет, либо надо экономить батарею).
- Если `lowPowerMode == false`, то вычисленные `deltaAz` / `deltaEl` используются для обновления текущих углов `panAngle` и `tiltAngle`, и вызывается `setServoAngle()` для обоих каналов, чтобы плавно двигать сервоприводы к новому положению. Если `lowPowerMode == true`, то сервы не двигаются, а код (при желании) мог бы еще и отключить выходы PCA9685 (в коде помечено, что можно было бы поставить OE=LOW или задать 0% ШИМ).
- Также внутри `readSensors()` управляется подсветка LCD: если вошли в `lowPowerMode` и подсветка была включена, она гасится (`lcd.noBacklight()`), и наоборот – если вернулись в нормальный режим, подсветка включается. Это позволяет ночью или при слабом батарее не тратить энергию на подсветку экрана.
- Если батарея опустилась ниже критического уровня `CRITICAL_BATTERY_THRESH`, ставится флаг `sleeping = true` – система перейдет в глубокий энергосберегающий режим (по сути, остановит работу).
- И наконец, обновление текста на LCD: раз в 2 секунды (настраивается `LCD_UPDATE_INTERVAL`) обновляются данные на экране. В данном случае в первой строке выводится режим (например "Night mode", "LowBatt mode" или "Tracking"), а во второй – напряжение батареи и температура. **При желании, по нажатию кнопки, можно менять отображаемую информацию** – например, между [Темп/Батарея] и [Влажность/Освещенность]. В коде для простоты это не реализовано, но по флагу или счетчику нажатий можно легко добавить.
- Функция `handleButton()` отрабатывает нажатие кнопки (D3). В примере она просто переключает подсветку вручную, но можно вместо этого менять режим отображения или, например, переключать в ручное управление (для ручного позиционирования панели кнопками – такой функционал вне рамок текущей задачи, но упомянем как идею).
- Заметим, что после всех вычислений в loop в конце стоит `delay(50)` – небольшая задержка для смягчения нагрузки на процессор и придания стабильности (50 мс – 20 итераций в секунду, более чем достаточно частоты обновления для такой системы).

Этот скетч дает представление, как объединить все модули. Вы можете его изменять под свои требования – например, настроить иной алгоритм слежения (помудренее ПИД-регулятора, если нужно), добавить индикацию уровня заряда (мигание светодиода при низком заряде) или, скажем, парковку панели на ночь (можно запрограммировать, что если ночь – вернуть панель в исходное положение, например на восток, чтобы утром быть готовой).

10. Энергосбережение и особенности работы в разных режимах

Последний раздел – о том, как эффективно использовать обновленное устройство, чтобы продлить время работы от аккумулятора и обеспечить работоспособность в различных условиях.

Отпускание сервоприводов: Сервоприводы потребляют значительный ток, особенно если постоянно удерживают нагрузку (панель) в определенном положении. Когда солнце не двигается быстро (например, в течение минут) или когда панель достигла нужного положения, **имеет смысл снять удерживающее усилие**. В коде это можно реализовать, отключая ШИМ-сигнал на выходе. В нашем случае, поскольку сигнал идет с PCA9685, можно либо установить для данного канала постоянно низкий уровень (записать 0 или 4096 в регистры сравнения, что даст 0% или 100% заполнение – 0% предпочтительнее, т.е. вообще без импульса), либо аппаратно отключить все выходы PCA через пин OE (Output Enable), если он выведен на модуле. После этого серво переходит в пассивный режим – вал можно повернуть рукой, и он не потребляет ток. Конечно, панель может под действием ветра или дисбаланса слегка отклониться, но обычно трекеры строят так, чтобы в сбалансированном состоянии панель не падала сама (например, центр тяжести на ось). В нашей программе мы отключаем сервоприводы, когда наступает `lowPowerMode` (то есть ночью или при очень слабой батарее). Можно пойти дальше: например, даже днем, когда разница датчиков минимальна, временно снимать сигнал до следующего цикла опроса – но это может привести к дрожанию панели, поэтому лучше отпускать на продолжительных паузах.

Режим сна Arduino: Arduino UNO на ATMega328P поддерживает несколько спящих режимов, вплоть до **Power-Down Mode**, где выключается почти все, кроме прерываний. При этом потребление микроконтроллера падает до микроампер. В нашем проекте основной потребитель – это не сам Arduino (который кушает ~10–20 мА в активном режиме), а периферия: преобразователь 5В, дисплей, модули. Тем не менее, в длительные периоды неактивности (ночью) **уместно усыплять контроллер**, чтобы он не гонял пустой цикл. В коде выше мы обозначили `sleeping` при критическом разряде – можно сделать то же при ночи: например, если `lux` очень мал несколько минут – погрузить систему в сон на, скажем, 5 минут, периодически просыпаясь для проверки освещения. Это реализуется через watchdog таймер или внешние таймеры. Поскольку наша система должна реагировать утром, самым простым решением будет не спать всю ночь, а, например, **раз в минуту** просыпаться, проверять люкс – если по-прежнему ночь, снова спать. Или настроить прерывание от RTC (если был) на рассвет – но у нас его нет.

В целом, если сильно экономить энергию не требуется, можно ограничиться **«софт-сном»** – уменьшить частоту опросов ночью, отключить подсветку, сервоприводы и просто ждать утра в цикле с большими задержками. Но при желании продлить работу от батареи, стоит освоить библиотеку Sleep_n0m1 или LowPower, позволяющие усыплять MCU на заданный период. Не забудьте при этом отключить ненужные модули: например, **выключить подсветку LCD** (она потребляет 20мА), **отключить питание датчиков** – хотя это сложнее сделать без доп. транзисторов, но можно, например, подать им питание через MOSFET и отключать. Во всяком случае, с выключенной подсветкой и спящим МК основной потребитель – преобразователь 5В, который сам может потреблять 5–10 мА. Это тоже можно решить, если использовать "умный" модуль – однако углубляться не будем.

Пороги по батарее: В коде мы использовали два порога – `LOW_BATTERY_THRESH` (например 3.6 В) для активации режима энергосбережения и `CRITICAL_BATTERY_THRESH` (~3.4 В) для почти полного отключения. Эти значения можно подправить под тип батареи. Для Li-Ion 18650 разряженной считается 3.0 В, но глубоко разряжать нежелательно – 3.3–3.4 В вполне критично. Можно также ввести **верхний порог**: например, не активировать движение серв, пока напряжение не поднимется выше 3.7 В (если солнце снова появилось и подзарядило). Наш алгоритм и так отключает движение при lowPowerMode и включает обратно, когда условие исчезает (заряд >3.6).

Обновление дисплея и индикация: Для экономии энергии мы уже сделали так, что дисплей обновляется не постоянно, а раз в пару секунд – этого достаточно, чтобы видеть плавно изменяющиеся параметры. Можно увеличить интервал до 5 секунд или больше, особенно для DHT (он меняется медленно). Кроме того, по нажатию кнопки можно **отключать дисплей целиком** – например, сделать режим "спящего экрана", когда горит только светодиод при событии. Но это опционально. Наоборот, можно добавить сигнализацию: мигать подсветкой или каким-то LED, если батарея почти разряжена, чтобы пользователь знал, что скоро устройство отключится.

Особенности работы при разной освещенности: Ваш трекер теперь умеет определять, когда **солнца нет (ночь)** – по показанию BH1750 или суммарному низкому сигналу LDR. Логично, что ночью сервоприводы не нужны – панель можно либо **припарковать** (например, повернуть в горизонтальное положение, чтобы избежать парусности ветром, или на восток, чтобы утром встретить солнце), либо оставить как есть. Если хотите парковать, можно в коде при переходе в night mode выставлять `panAngle = 90, tiltAngle = 0` например, и выполнить `setServoAngle` перед отключением серв. Также ночью можно **полностью отключать питание некоторых модулей**: например, BH1750, ADS1115 и PCA9685 тратят по несколько миллиампер. Если они питаются от той же 5В, единственный способ – отключить 5В конвертер, но тогда и Arduino вырубится. Поэтому, если бы была возможность, хорошо бы переводить ADS1115 и BH1750 в power-down режим командой – BH1750 имеет режим Power Down (команда 0x00), ADS1115 – тоже можно поставить на один измерение и уснуть. Это не критично, но дополнительно снизит потребление в спячке.

Использование кнопки: Мы задействовали одну кнопку. Она может служить **универсальным управлением**: например, короткое нажатие – переключение отображаемых данных на LCD (темп/влажн -> напряжение/люкс -> угол панели и т.п.), длительное нажатие – **переключение режима авто/ручной**. Ручной режим мог бы позволить кнопками (если добавить вторую, или многократными нажатиями одной) вручную двигать панель – это полезно для калибровки или обслуживания. Также кнопку можно использовать для **принудительного пробуждения** из глубокого сна утром, если, скажем, солнце уже встало, а устройство спит (хотя можно настроить так, что оно само проснется по таймеру).

Итоги по энергосбережению: При условии, что устройство большую часть ночи и пасмурного времени будет находиться в пониженном потреблении (минимум движения, отключенная подсветка, спящий контроллер), оно сможет дольше проработать от аккумулятора. Днем, когда солнечно, энергия поступает от панели, поэтому можно позволить активность (вращение панели). Если же день пасмурный и батарея не заряжается, то трекер сам уменьшит свою активность, чтобы дождаться солнца не разрядившись в ноль.

На этом инструкция завершается. Мы рассмотрели и реализовали все поставленные задачи: **переход на один кабель**, разделение на модули, подключение конкретных компонентов (ADS1115, PCA9685, BH1750, DHT11, LCD, пр.), распиновку RJ45, меры по питанию (конденсатор, проверка токов) и поэтапную интеграцию. Финальный код предоставлен как отправная точка для прошивки контроллера. Следуя этой инструкции, вы сможете модернизировать свой солнечный трекер, сделав его более аккуратным в плане проводки и более «умным» в плане контроля питания и состояний. Желаем успехов в сборке и эксплуатаци вашего устройства!

---

[1] [2] [3] Шина I2C: принципы функционирования или зачем ещё тут нужны какие-то резисторы? – kotyara12.ru
https://kotyara12.ru/iot/i2c/

[4] I2C/1-W wiring over long distances with Cat5e - General Electronics - Arduino Forum
https://forum.arduino.cc/t/i2c-1-w-wiring-over-long-distances-with-cat5e/281799

[5] Practical question about servos and capacitors - General Discussions - RobotShop Community
https://community.robotshop.com/forum/t/practical-question-about-servos-and-capacitors/7566