Краевое государственное бюджетное
профессиональное образовательное учреждение
«Хабаровский колледж отраслевых технологий и сферы обслуживания»

Методическое пособие

От молчания к диалогу: интеграция деловой ролевой игры в преподавание микропроцессорных систем

по МДК2.1 «Микропроцессорные системы»
для специальности
09.02.01 Компьютерные системы и комплексы

Разработано преподавателем
информационных дисциплин
Мазур Т.В.

Хабаровск, 2026

СОДЕРЖАНИЕ

Описание проекта 3
Сценарий урока 1. Контроль доступа 8
Сценарий урока 2. Шаговый двигатель 16
Сценарий урока 3. Датчик температуры и влажности 24
Сценарий урока 4. RGB-светодиод 30
Сценарий урока 5. Часы реального времени 37
Сценарий урока 6. «Умная оранжерея» 46
Сложности применения метода и пути их преодоления 60
Список использованных источников 63

Описание проекта

Цель: формирование профессиональных компетенций (ПК 2.1–2.5) у студентов специальности 09.02.01 «Компьютерные системы и комплексы» и развитие «гибких навыков» (Soft Skills: ОК 4, ОК 5) через моделирование полного жизненного цикла создания IoT-устройства – от общения с заказчиком до тестирования прототипа.

Методология: геймификация, ролевая игра, перевёрнутый класс, элементы Agile (спринты, планирование, рефлексия).

Продукт проекта: методическая разработка с готовыми сценариями и указаниями к выполнению работ.

Традиционно моя деятельность как преподавателя МДК 2.1 «Микропроцессорные системы» была сфокусирована на **hard skills**. Моя главная цель – дать студентам актуальные знания, помочь им не отставать от технического прогресса и развить мышление инженера и программиста. Мы создаем прототипы автоматизированных систем «Умного дома»: сигнализации, климат-контроля, автоматического полива, умных розеток и кормушек для питомцев.

Но в последнее время я заметила тревожную тенденцию. Студенты разучились говорить в профессиональном контексте. Нет, они прекрасно и оживленно болтают на переменах, но в рамках академических часов обсуждения и генерация идей практически сошли на нет. Обычно при фронтальном опросе отвечают два-три самых активных человека, создавая иллюзию работы всей группы. Если присмотреться, часто молчат даже сильные студенты. Они не отвечают не потому, что не знают материал, а потому что стесняются, боятся ошибиться, не успевают за более быстрыми или просто не привыкли рассуждать вслух.

Эту проблему ярко подтвердил недавний случай. У нас проводила мастер-класс представитель IT-компании. Она была открыта, доброжелательна, с отличным чувством юмора. Но когда она начала задавать

вопросы аудитории, стало очевидно: она физически не может «расшевелить» зал. Ребята были настолько зажаты, обратную связь давали со скрипом.

Почему так происходит? Отчасти в этом есть и моя вина. Я, как и многие коллеги, уже давно не практикую классические устные экзамены, где студент сидит перед комиссией и рассказывает материал. Из-за возможностей списывания и желания объективно оценить именно навыки кодинга, мы перешли на тесты, решение задач и написание программ. На защите лабораторных работ я чаще прошу что-то изменить или дополнить в коде. Мы отлично научили их писать код, но лишили практики вербализации своих мыслей. В учебном процессе у них почти нет возможности коммуницировать с «заказчиками» и «коллегами», от которых в реальной жизни напрямую зависит результат.

Не все могут работать группой. Часто лидер группы делает за всех большую часть работы и не слышно «живых» обсуждений, споров, в которых рождается истина.

Поэтому у меня возникла идея: вернуть студентов в диалог через элементы деловой ролевой игры.

Можно ли проводить технические занятия без игровых методов? Конечно. Но дидактическая игра, предшествующая лабораторной работе, позволяет показать значимость создаваемой системы, делает процесс эмоционально вовлекающим и, главное, заставляет студентов готовиться и говорить.

Сценарий игры «Студия автоматизации»

Стандартный сюжет игры: **Заказчик** обращается в инжиниринговую фирму, специализирующуюся на «Умных домах», с задачей автоматизировать определенную подсистему (тема должна соответствовать теме текущего занятия). Заранее между студентами распределяются роли.

Заказчик. Клиент, который хочет превратить свой дом в «Умный». Он формулирует требования. Например, если тема «Контроль доступа», он может захотеть открывать дверь по карте, коду, отпечатку пальца или сигналу с

телефона, требовать звукового сигнала и автозакрытия через 10 секунд. Если тема «Датчик звука» – включать свет по хлопку, делать светомузыку или открывать дверь двойным хлопком. Заказчик может выдвигать любые, порой безумные пожелания. Это провоцирует живой разговор на актуальные темы автоматизации.

Заказчику не запрещается выступать с «капризным» требованием (например, «Хочу, чтобы система работала, даже если отключили свет» или «Нужно, чтобы пин-код менялся каждый день»). Его задача – максимально усложнить ТЗ в рамках разумного.

Преподаватель может подстраховать Заказчика. Например, сделать карточки с вариантами заданий.

Менеджер. Принимает заказ, ведет переговоры с клиентом, переводит его пожелания на технический язык и делает уточнения. Фиксирует итоговое задание на доске или бумаге.

В идеале Менеджер являются модератором обсуждения, передаёт слово другим участникам в соответствие с их ролями, уточняет детали.

Кладовщик. Ограничивает фантазию заказчика, сталкивая её с суровой реальностью. Реализовать всё мы не можем – нас ограничивает состав стартового набора Arduino. Кладовщик оценивает, какие устройства нужны, и честно говорит: «Сканера отпечатка пальца у нас нет, поэтому эту функцию исключаем». Он же подбирает доступные аналоги.

Схемотехник. Отвечает за аппаратную часть. Его задача – рассказать об особенностях подключения выбранных устройств. Эта роль требует предварительной домашней подготовки (поиск схем, datasheet). Схемотехник знает тему, но не знает пожеланий заказчика, поэтому ему приходится оперативно искать информацию и тесно взаимодействовать с Кладовщиком. Рисует схему подключения. Отвечает за то, чтобы не сжечь плату (расчет токов, необходимость транзисторов/реле).

Программист. Самая сложная роль. Он рассказывает о библиотеках, логике построения кода и алгоритмах. Здесь часто требуется помощь

преподавателя: в ходе дискуссии между Программистом и преподавателем рождается оптимальное техническое решение, и мне, как наставнику, иногда приходится принимать альтернативную точку зрения студента.

Тестировщик. Его задача – не протестировать (это будет позже), а предложить стратегию тестирования. Например, для системы доступа он должен придумать сценарии: что будет, если приложить чужую карту? Что будет, если дернуть провод во время работы?

Бухгалтер. (Не обязательная, но очень интересная роль). Подсчитывает смету и стоимость комплектующих в реальных магазинах.

Методология: «Перевёрнутый класс» и переход к практике

Игра строится на методе **«Перевёрнутого класса»**. Студенты выбирают заранее роли и дома просматривают теоретические материалы и примеры проектов на Arduino.

Следующий этап урока – уже не игровой, а классическая лабораторная работа. Студентам выдаются методические рекомендации. Разумеется, текст методички готов заранее и не может предугадать все решения, озвученные в игре. Но теперь методичка служит лишь **базой**. Студенты берут её за основу и «нанизывают» на неё дополнения: всё то, что они сами придумали, что требовал Заказчик и что согласовал Кладовщик.

Подведение итогов (Рефлексия)

Финал игры – важнейшая часть. Рефлексия выполняется в формате фронтальной беседы, где я задаю наводящие вопросы:

– *Что в итоге удалось реализовать?*

– *Какие требования заказчика оказались завышены и почему?*

– *Как в перспективе (при наличии бюджета) эту систему можно усовершенствовать?*

– *Во сколько примерно обойдется оборудование для серийного образца?*

Такой подход позволяет не только научить студентов работать с микроконтроллерами, но и вернуть им уверенность в себе, научить

аргументировать свои технические решения и общаться – именно то, чего так не хватает современным выпускникам.

Состав продукта включает:

- описание технологии, игры, ролей;
- примерные сценарии занятий с применением технологии;
- тексты базовых практических работ;
- критерии оценки;
- советы для преподавателя при проведении занятий;
- описание трудностей, которые могут возникнуть и пути их преодоления.

Ниже приведены сценарии занятий, которые можно провести поданной методике. Сценарий изучается преподавателем, чтобы в соответствие с ним можно было проинструктировать Заказчика, Программиста, Схемотехника (возможно и других), чтобы направлять разговор в нужное русло, подкидывать идеи. В сценарии выделены моменты, на которые нужно обратить внимание. А также даны тексты практических работ. В отличие от практических работ без применения методики, в них нет заданий для самостоятельной работы. Так как эти задания составляются в соответствие с пожеланиями Заказчика и могут варьироваться.

СЦЕНАРИЙ УРОКА 1. Контроль доступа

Тема: Разработка интеллектуальной системы контроля доступа (СКУД) и сигнализации

Продолжительность: 90 минут (пара).

ЭТАП 0. Подготовка (Перевёрнутый класс): за неделю до урока студенты получают тему и выбирают роли. Выбравшие роли Схемотехника и Программиста заранее изучают материал, каждый в своей области. Заказчик обдумывает заказ. Остальные «роли» готовятся на своё усмотрение, допускается импровизация.

ЭТАП 1. Брифинг или «Летучка» (5 минут)

Преподаватель (директор компании): «Коллеги, к нам обратился крупный клиент. Ему нужно автоматизировать вход в серверную комнату. У нас 80 минут на создание MVP (минимально жизнеспособного продукта)».

Студенты садятся по группам (по 2-3 человек). Получившие на этот урок студенты выходят к доске.

ЭТАП 2. Игровое проектирование (15 минут)

Преподаватель обращается к заказчику: «Что бы вы хотели видеть в проектируемой СКД?»

Заказчик: «Я хочу, чтобы дверь открывалась по карте. А также по ключу. Но! Если кто-то приложит чужую карту 3 раза подряд, должна включиться сирена и отправляться СМС мне на телефон. И еще, чтобы после открытия дверь сама захлопывалась через 5 секунд».

Менеджер записывает на доску/флипчарт/лист А3: *1. RFID доступ. 2. Защита от подбора (3 ошибки). 3. Сирена. 4. Автозакрытие (Серво).*

Кладовщик: «СМС отправлять нечем, GSM-модуля SIM800L в наборе нет. Можем предложить громкую сирену (зуммер) и мигание красным светодиодом. Заказчик, устроит?» -> *Торг, согласование.*

Схемотехник: «Сервопривод SG90 слабоват для реальной двери, но для макета пойдет. Зуммер подключаем через транзистор, пин Ардуино не потянет».

Заказчик: «Я часто забываю карты и ключи в кармане другой куртки, поэтому мне ещё нужен дополнительный способ по коду, введённому с клавиатуры»

Бухгалтер: «Это удорожит конструкцию»

Программист: «Используем библиотеку MFRC522. Для счетчика ошибок заведем переменную int errorCount. Для автозакрытия используем millis(), delay() нельзя, иначе мы пропустим считывание карты».

Тестировщик: «Я буду тестировать так: 1. Приложу свою карту. 2. Приложу чужую 1 раз. 2. Приложу чужую 3 раза подряд. 3. Дерну провод от RFID во время работы».

Бухгалтер: «Считаем: Arduino Nano (300р) + RFID (250р) + Серво (150р) + Зуммер (50р). Итого прототип: 750 рублей. В серийном производстве будет дешевле». При добавлении клавиатуры плюс 200 рублей.

Итог этапа: на руках у команды лист с утвержденным ТЗ и схемой.

ЭТАП 3. Лабораторная работа (50 минут)

Студенты получают базовую методичку (где описано простое считывание RFID и поворот серво).

Задача: взять код из методички за основу и «нарастить» на него функции, которые они сами себе придумали в игре (счетчик ошибок, сирена, таймер на millis()).

В методичке отсутствуют задания для самостоятельной работы, как это было раньше, так как задание придумывает «заказчик», фиксирует менеджер.

Роль преподавателя: не давать готовых решений. Отвечать на вопросы программиста, направлять схемотехника, если тот путает полярность.

ЭТАП 4. Демонстрация и тестирование (10 минут)

Преподаватель вместе с Заказчиком, Менеджером, Схемотехником, Программистом совместно проверяют выполнение задания, задают вопросы,

можно акцентировать внимание на сильных сторонах разработанного прототипа, сделать замечания. Не запрещается остальным командам разработчиков посмотреть на прототипы других.

ЭТАП 5. Рефлексия (10 минут)

Фронтальная беседа.

Дополнительные моменты, которые можно применить

1. **«Внезапное изменение ТЗ»:** в середине практической работы может возникнуть дополнительная идея. вы вбрасываете в игру событие: *«Заказчик передумал! Он говорит, что красный светодиод ему не нравится, хочет синий. А еще он хочет, чтобы при успешном входе проигрывалась мелодия»*. *Цель:* научить студентов гибкости и показать, как изменения в середине проекта ломают код и схему.

2. **Особенности оценивание:** оценку получает команда, выполняющая задание.

- **Оценка «5»:** прототип работает + ТЗ выполнено + устройство протестировано.
- **Оценка «4»:** прототип работает, но не все реализуемые пожелания заказчика выполнены. (например, нет сирены при 3 ошибках).
- **Оценка «3»:** работает только базовая функция из методички (считывание карты).

Практическая работа

Контроль доступа: RFID-rc522 + Servo + Arduino

Цель: сделать простую систему, которая будет отпирать замок по электронному ключу (Метке).

Необходимое оборудование

- Arduino
- RFID-модуль RC522
- Сервопривод
- Провода Папа-Мама

- Провода Папа-Папа или набор проводов для макетирования
- Зуммер пьезоэлектрический
- Breadboard

Для реализации проекта необходимо установить библиотеку: RFID Library for MFRC522.

Краткие теоретические сведения

Технические характеристики RFID-модуля RC522

- Напряжение питания: 3.3V;
- Потребляемый ток :13-26mA;
- Рабочая частота: 13.56MHz;
- Дальность считывания: 0 - 60 мм;
- Интерфейс: SPI;
- Скорость передачи: максимальная 10МБит/с;
- Размер: 40мм х 60мм.

Интерфейсы и назначение выводов

Микросхема MFRC522 поддерживает интерфейсы SPI, UART и I2C Выбор интерфейса осуществляется установкой логических уровней на определенных выводах микросхемы. На данном модуле выбран интерфейс SPI.



Назначение выводов интерфейса SPI:

- SDA – выбор ведомого;
- SCK –сигнал синхронизации;
- MOSI – передача от master к slave;

- MISO – передача от slave к master;
- RST – вывод для сброса;
- IRQ – вывод прерывания;
- GND – земля;
- Vcc –питание 3.3 В.

Сигнал сброса RST – это сигнал, поступающий от цифрового выхода контроллера. При поступлении сигнала LOW происходит перезагрузка считывателя. Также ридер установкой на RST низкого уровня сообщает, что находится в режиме сна, для вывода модуля из режима сна необходимо подать на данный вывод сигнал HIGH.

Ход работы

Сборка схемы

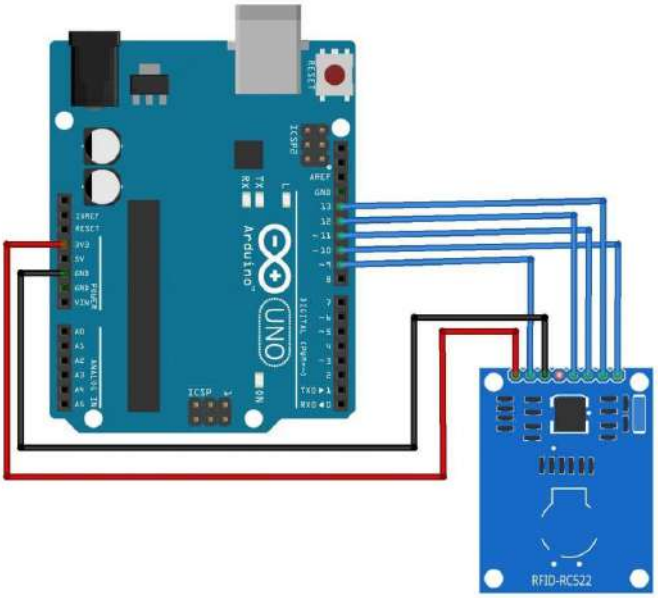
Зуммер подключаем в следующей последовательности:

Arduino	**Зуммер**
GND	GND
pin 5	IO

В роли отпирающего механизма будет использоваться сервопривод. Сервопривод может быть выбран любой, в зависимости от требуемых вам размеров и усилий, который создает сервопривод. У сервопривода имеется 3 контакта:

Arduino	**Сервопривод**
5V *	Красный (Центральный)
GND	Черный или Коричневый (Левый)
pin 6	Белый или Оранжевый (Правый)

Подключение модуля RFID-RC522 к плате Arduino по будем производить по схеме соединений на рисунке.

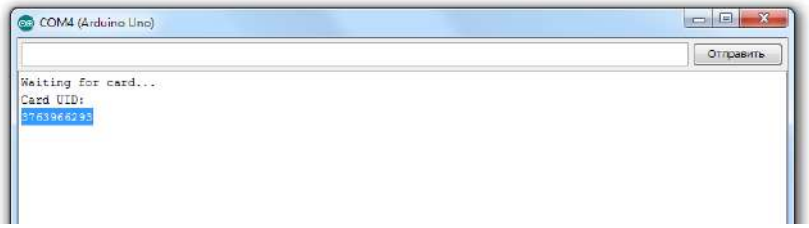


Программирование

```
#include <Servo.h>
#include <SPI.h>
#include <MFRC522.h> // библиотека "RFID".
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);
unsigned long uidDec, uidDecTemp;  // для хранения номера метки в десятичном формате
Servo servo;
void setup() {
  Serial.begin(9600);
  Serial.println("Waiting for card...");
  SPI.begin();  //  инициализация SPI / Init SPI bus.
  mfrc522.PCD_Init();     // инициализация MFRC522
  servo.attach(6);
  servo.write(0);  // устанавливаем серво в закрытое состояние
}
```

```
void loop() {
  // Поиск новой метки
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
  // Выбор метки
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  uidDec = 0;
  // Выдача серийного номера метки.
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
    uidDecTemp = mfrc522.uid.uidByte[i];
    uidDec = uidDec * 256 + uidDecTemp;
  }
  Serial.println("Card UID: ");
  Serial.println(uidDec); // Выводим UID метки в консоль.
  if (uidDec == 3763966293) // Сравниваем Uid метки, если он равен заданому то серва открывает.
  {
    tone(5, 200, 500); // Делаем звуковой сигнал, Открытие
    servo.write(90); // Поворациваем серву на угол 90 градусов(Отпираем какой либо механизм: задвижку, поворациваем ключ и т.д.)
    delay(3000); // пауза 3 сек и механизм запирается.
    tone(5, 500, 500); // Делаем звуковой сигнал,  Закрытие
  }
  servo.write(0);  // устанавливаем серву в закрытое сосотояние
}
```

Для того, что бы узнать UID карточки (Метки), необходимо записать данный скетч в arduino, собрать схему, изложенную выше, и открыть Консоль (Мониторинг последовательного порта). Когда вы поднесете метку к RFID, в консоли выведется номер



Полученный UID необходимо ввести в следующую строчку:

if (uidDec == 3763966293) // Сравниваем Uid метки, если он равен заданному, то сервопривод открывает задвижку.

У каждой карточки данный идентификатор уникальный и не повторяется. Таком образом, когда вы поднесете карточку, идентификатор которой вы задали в программе, система откроет доступ с помощью сервопривода.

СЦЕНАРИЙ УРОКА 2. Шаговый двигатель

Тема: Разработка умного привода на базе шагового двигателя

Формат: ролевая игра, элементы Agile, перевёрнутый класс.

Время: 90 минут (2 академических часа).

ЭТАП 0. Подготовка (Перевёрнутый класс / Домашнее задание)

За неделю до урока преподаватель распределяет роли. **Задача студентов до урока:**

1. Изучить предоставленный базовый материал (теорию по шаговым двигателям, распиновку драйвера ULN2003, работу библиотеки Stepper.h).

2. Подготовить краткий доклад по своей роли. Схемотехник ищет datasheet на 28BYJ-48, особое внимание на структуру и принцип работы. Можно предложить сделать презентацию с анимацией, которая объясняет, как работает шаговый двигатель https://dzen.ru/a/XgoiR8BccQCtZqlZ. Программист читает документацию по Stepper.h и функции map().

3. Особое внимание на то, как правильно подключить питание. Лучше от внешнего элемента 5-12 В, но у нас в наборе такого нет (о чём должен предупредить Схомотехник). Если питание от платы, нужно следить за перегревом.

ЭТАП 1. Брифинг и «Заказ» (5 минут)

Преподаватель (директор инжинирингового бюро): «Коллеги, к нам обратился заказчик. Ему нужна система автоматизации для "Умных жалюзи" в серверной комнате. У нас есть 80 минут на создание MVP (минимально жизнеспособного продукта) на базе шагового двигателя 28BYJ-48 и драйвера ULN2003. Команды, начинаем спринт-планирование!»

ЭТАП 2. Спринт-планирование (10 минут)

Заказчик озвучивает требования.

Требование 1 (Калибровка). «При включении питания жалюзи должны провести самодиагностику: сделать три полных оборота по часовой стрелке, а потом один против, чтобы сбросить натяжение ткани».

Требование 2 (Пульт управления). «Мне нужны две физические кнопки: одна открывает жалюзи (вращение по часовой), другая закрывает (против часовой)».

Требование 3 (Шумоподавление): «Двигатель шумит! Добавьте ручку, чтобы я мог плавно регулировать скорость вращения от минимальной до максимальной».

Кладовщик проверяет наличие компонентов в стартовом наборе. Отсекает нереализуемое (например, если заказчик попросит энкодер, а его нет в наборе – предлагает потенциометр).

Схемотехник рисует схему на листе А4. Определяет, какие пины Arduino (10, 11, 12, 13) пойдут на драйвер (IN1-IN4), а какие на кнопки и потенциометр. *Важно: напоминает команде, что шаговый двигатель нежелательно питать от 5V Arduino напрямую, нужен внешний источник.*

Программист составляет псевдокод. *Ключевой момент:* программист должен заметить, что если использовать delay() из базового примера, кнопки не будут реагировать на нажатия во время вращения. Он предлагает использовать библиотеку Stepper.h и функцию map() для потенциометра.

1. **Тестировщик** пишет чек-лист для приёмки:

Тест 1: подать питание -> двигатель делает 3 оборота в одну сторону, 1 в другую.

Тест 2: нажать кнопку "Открыть" во время калибровки (проверка на блокировку).

Тест 3: выкрутить потенциометр в минимум -> двигатель должен остановиться, но не гудеть.

ЭТАП 3. Реализация / Лабораторная работа (60 минут)

Студенты собирают схему на макетной плате и пишут код. **Роль преподавателя:** не давать готовых решений, а задавать наводящие вопросы.

Если команда застряла на Задании 1: «Посмотрите в базовом примере, как задаётся количество шагов на один оборот. Сколько шагов вам нужно для трёх оборотов?»

Если команда застряла на Задании 3: «Как преобразовать диапазон аналогового сигнала (0-1023) в диапазон оборотов в минуту (1-15)? Вспомните функцию map()».

Если двигатель гудит, но не крутится: «Проверьте распиновку драйвера ULN2003. Правильно ли вы подключили IN1, IN2, IN3, IN4? Не перепутали ли вы последовательность?»

Студенты берут за основу код из методички, но ***обязаны*** *модифицировать его под требования Заказчика (добавить кнопки, потенциометр и логику калибровки).*

ЭТАП 4. Демонстрация и тестирование (10 минут)

Преподаватель вместе с Заказчиком, Менеджером, Схемотехником, Программистом совместно проверяют выполнение задания, задают вопросы, можно акцентировать внимание на сильных сторонах разработанного прототипа, сделать замечания. Не запрещается остальным командам разработчиков посмотреть на прототипы других.

ЭТАП 5. Рефлексия (10 минут)

Фронтальная беседа.

Дополнительные моменты, которые можно применить

1. **«Внезапное изменение ТЗ»:** в середине практической работы может возникнуть дополнительная идея. вы вбрасываете в игру событие: *«Заказчик передумал! Он говорит, что красный светодиод ему не нравится, хочет синий. А еще он хочет, чтобы при успешном входе проигрывалась мелодия»*. *Цель:* научить студентов гибкости и показать, как изменения в середине проекта ломают код и схему.

2. **Особенности оценивание:** оценку получает команда, выполняющая задание.

Оценка «5»: прототип работает + ТЗ выполнено + устройство протестировано.

Оценка «4»: прототип работает, но не все реализуемые пожелания заказчика выполнены. (например, нет сирены при 3 ошибках).

Оценка «3»: работает только базовая функция из методички (считывание карты).

Вопросы для обсуждения

1. «Команда 3, у вас двигатель при калибровке (3 оборота) работал очень медленно. Почему?» *(Ожидаемый ответ: В библиотеке Stepper.h мы задали setSpeed(15), а для калибровки можно было задать скорость выше, но мы использовали одну переменную для всего цикла).*

2. «Схемотехники, почему драйвер ULN2003 нагрелся во время тестов?» *(Обсуждение токов, КПД шаговых двигателей и необходимости радиаторов в реальных устройствах).*

3. «Программисты, кто использовал delay() для опроса кнопок? Что произошло, когда вы нажали кнопку во время задержки?» *(Вывод о важности неблокирующего кода и прерываний, которые будут изучаться позже).*

4. «Как бы вы улучшили этот проект, если бы у вас было еще 2 недели?» *(Добавить дисплей, концевые выключатели, чтобы жалюзи не рвали ткань, подключить Wi-Fi).*

На что акцентировать внимание

1. **Проблема «Гудящего двигателя»:** двигатель 28BYJ-48 в паре с драйвером ULN2003 часто начинает гудеть и не вращаться, если задать слишком высокую скорость (например, myStepper.setSpeed(30)).

Если команда столкнется с этим, не подсказывайте сразу. Пусть они сами методом проб и ошибок найдут предел (обычно это 12-15 об/мин для этого мотора в полном шаге). Это формирует **ПК 2.4 (Тестирование и верификация)**.

2. **Полушаговый режим:** в методичке упомянута таблица полушагового управления. Если сильная команда закончит базовые задания быстро, дайте им челлендж: *«Заказчик хочет, чтобы жалюзи двигались максимально плавно и тихо. Реализуйте полушаговый режим вручную, без библиотеки, используя массивы»*.

3. **Питание:** шаговый двигатель 28BYJ-48 потребляет около 200-300 мА. Если питать его от 5V пина Arduino, плата может перегреться или уйти в перезагрузку.

Практическая работа
Подключение и управление Шаговым двигателем

Цель: научиться подключать шаговый двигатель к контроллеру, управлять им с использованием библиотеки и без. Получить опыт совместного использования шагового двигателя с другими устройствами.

Основные теоретические сведения

Шаговый двигатель - один из основных компонентов роботехники, ЧПУ-станков, 3D-принетеров и других автоматических систем.



Шаговый электродвигатель – синхронный бесщёточный электродвигатель с несколькими обмотками, в котором ток, подаваемый в одну из обмоток статора, вызывает фиксацию ротора. Последовательная активация обмоток двигателя вызывает дискретные угловые перемещения (шаги) ротора.

Питание такого двигателя невозможно без системы управления, или как его еще называют, драйвера – он подаёт импульсы в нужные обмотки, чтобы повернуть ротор на нужный угол.

Подключение

13 – IN1; 12 – IN2; 11 – IN3; 10 – IN4.

Управление без использования библиотек

Номер шага	Синий провод (A)	Розовый провод (B)	Желтый провод (A”)	Оранжевый провод (B”)
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

Дальше напишем в Arudino IDE код, который будет подавать на выходы сигналы в соответствии с таблицей выше.

```
// назначим переменные с номерами портов
int in1 = 13;
int in2 = 12;
int in3 = 11;
int in4 = 10;
const int dl = 2; // переменная для задержки
// назначим указанные пины как выходы
void setup() {
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
}
void loop() {
//сформируем сигналы для первого шага
digitalWrite( in1, HIGH );
digitalWrite( in2, HIGH );
digitalWrite( in3, LOW );
digitalWrite( in4, LOW );
delay(dl); //Задержка между шагами, чем она меньше - тем быстрее вращение вала.
//сформируем сигналы для второго шага
digitalWrite( in1, LOW );
```

```
digitalWrite( in2, HIGH );
digitalWrite( in3, HIGH );
digitalWrite( in4, LOW );
delay(dl);
//сформируем сигналы для третьего шага
digitalWrite( in1, LOW );
digitalWrite( in2, LOW );
digitalWrite( in3, HIGH );
digitalWrite( in4, HIGH );
delay(dl);
//сформируем сигналы для четвертого шага
digitalWrite( in1, HIGH );
digitalWrite( in2, LOW );
digitalWrite( in3, LOW );
digitalWrite( in4, HIGH );
delay(dl);
}
```

Двигатель начнет вращаться, скорость вращения задаётся переменной dl.

Можно поэкспериментировать с полушаговым управлением. В таблице ниже приведен порядок подачи сигналов на обмотки рассматриваемого двигателя для его реализации.

Номер шага	Синий провод (A)	Розовый провод (B)	Желтый провод (A")	Оранжевый провод (B")
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

Но на практике такой подход к управлению шаговым двигателем не используется. Для этого есть готовые библиотеки, в Arduino IDE есть встроенная «Stepper».

```
#include <Stepper.h>
const int stepsPerRevolution = 2048; // изменить в соответствии с количеством
//шагов вашего двигателя. Вообще здесь задаём количество шагов, на
которые повернется двигатель
// мы указали для 1 полного оборота
// назначаем пины, к которым подключен драйвер ШД
Stepper myStepper(stepsPerRevolution, 13, 12, 11, 10);
void setup() {
// указываем номинальную скорость 15 об.мин:
myStepper.setSpeed(15);
// включаем последовательный порт:
Serial.begin(9600);
}
void loop() {
// делаем оборот по часовой стрелке:
Serial.println("clockwise"); // сообщение в монитор порта
myStepper.step(stepsPerRevolution);
delay(500);
// делаем оборот против часовой стрелке:
Serial.println("counterclockwise"); // сообщение в монитор порта
myStepper.step(-stepsPerRevolution);
delay(500);
}
```

СЦЕНАРИЙ УРОКА 3. Датчик температуры и влажности

Тема: Разработка системы климат-контроля на базе датчика DHT11

Продолжительность: 90 минут (2 академических часа)

Формат: ролевая игра (Agile/Scrum), перевёрнутый класс, командная работа.

ЭТАП 0. Подготовка (Перевёрнутый класс / Домашнее задание)

За неделю до занятия объявляется тема. Задача **Заказчика** и **Программиста** (все остальные роли по своему усмотрению) до урока:

1. Изучить принцип работы датчика DHT11 (почему он медленный, зачем ему задержка между опросами).
2. Повторить основы работы с библиотекой DHT.h, LCD-дисплеем (I2C) и сервоприводом.
3. Подготовить краткий доклад по своей роли (Схемотехник ищет распиновку DHT11 и требования по питанию, Программист читает про функцию isnan()).

ЭТАП 1. Брифинг и «Заказ» (15 минут)

Преподаватель (руководитель инжинирингового бюро): «Коллеги, к нам обратился владелец загородного дома. Ему нужна система мониторинга микроклимата в спальне с функцией автоматического проветривания. У нас 80 минут на создание MVP. Команды, начинаем спринт-планирование!»

Заказчик озвучивает требования.

Требование 1 (Мониторинг): «Я хочу видеть текущую температуру и влажность на экране. А еще мне нужно, чтобы система запоминала **максимальные и минимальные** показатели за день и тоже их показывала!».

Требование 2 (Световая индикация): «Если влажность упадет ниже 30% – зажгите желтый светодиод. Если ниже 20% – красный и включите противный звуковой сигнал (зуммер)!».

Требование 3 (Автоматика): «Если в комнате жарко (больше 26°C) **И** при этом на улице светло (день), автоматически открывайте форточку (сервопривод). Ночью форточку не открывать!».

Секретное требование (Каприз): «А если датчик сломается или отвалится провод, система не должна зависать. Пусть просто пищит!»

Менеджер переводит пожелания на технический язык, фиксирует ТЗ на листе или доске. Организует коммуникацию с другими сотрудниками. Уточняет у Заказчика: «Под "светло" вы имеете в виду конкретный порог освещенности?». Ночь – это считается отсутствие освещённости?

Кладовщик оценивает стартовый набор. «DHT11, LCD 1602 с I2C, Servo SG90, фоторезистор, зуммер, светодиоды есть. Но у DHT11 есть особенность – он опрашивается не чаще 1 раза в секунду. Заказчик, учтите, что реакция на резкое изменение влажности будет с задержкой».

Схемотехник рисует (предоставляет) схему. «DHT11 подключаем к цифровому пину. LCD вешаем на I2C (SCL, SDA), чтобы не тратить пины. Фоторезистор – через делитель напряжения на аналоговый пин. **Важно:** сервопривод лучше питать от внешнего источника, иначе при его движении будет просадка напряжения и DHT11 начнет выдавать ошибки».

Программист составляет псевдокод. «Используем библиотеки DHT.h, LiquidCrystal_I2C.h, Servo.h. Для максимума/минимума заведем переменные maxTemp, minTemp. Особое внимание лаконичному алгоритмы. Обязательно добавим проверку isnan(t) – если датчик отвалился, выводим на экран "ERROR" и включаем зуммер».

Тестировщик пишет чек-лист приемки *до начала сборки*:

Тест 1: подышать на датчик (влажность должна вырасти).

Тест 2: придвинуть к датчику работающий фен (температура должна вырасти, серво должно повернуться, если светло).

Тест 3: закрыть фоторезистор пальцем (имитация ночи) – серво должно вернуться в 0, даже если жарко.

Тест 4: выдернуть провод из DHT11 – на экране "ERROR", зуммер пищит.

Бухгалтер считает смету прототипа для отчета Заказчику.

ЭТАП 2. Реализация / Лабораторная работа (60 минут)

Студенты собирают схему на макетной плате и пишут код. **Роль преподавателя:** ходит между столами, не дает готовых решений, а задает наводящие вопросы.

Если команда застряла на максимумах/минимумах: «Как вы инициализируете переменную минимума при включении? Если вы зададите minTemp = 0, система никогда не обновит минимум. Подумайте, как сделать это правильно».

Если команда застряла на ночном режиме: «Вы используете delay(2000) для опроса DHT11. Пока программа "спит" в задержке, она не читает фоторезистор. Как сделать так, чтобы опрос света происходил чаще?» *(Подсказка: использовать millis() для неблокирующего опроса DHT11).*

Если команда игнорирует isnan(): «Заказчик просил, чтобы система не зависала при обрыве провода. Что вернет функция readTemperature(), если провод отвалится?»

ЭТАП 3. Приёмка и тестирование (15 минут)

Тестировщик или представители команд выполняют чек-лист тестирования на устройстве (дышат на датчик, закрывают фоторезистор, выдергивают провод). **Заказчик** с преподавателем проверяют, выполнены ли пожелания (особенно секретное требование про защиту от обрыва провода). Если устройство не проходит тест, Тестировщик записывает «Баг-репорт» (например: «При дыхании на датчик влажность растет, но максимум не обновляется»).

ЭТАП 4. Рефлексия (5 минут)

Фронтальная беседа. Преподаватель подводит итоги, связывая опыт студентов с реальной инженерией и заставляя их говорить.

Вопросы для обсуждения (направленные на развитие коммуникации и технического мышления):

1. «Команда 3, у вас при тестировании зуммер начал пищать сам по себе, без обрыва провода. В чем была причина?» *(Ожидаемый ответ: DHT11 очень медленный датчик. Если опрашивать его чаще, чем раз в 1-2 секунды, он возвращает NaN, и срабатывала защита от обрыва. Мы добавили задержку 2000 мс).*

2. «Что в ТЗ было самым сложным? Удалось ли Программистам дать информацию для реализации логики "День + Жара = Форточка"? Как вы обошли проблему блокирующего delay()?»

3. «Схемотехники, почему мы использовали LCD с модулем I2C, а не обычный? Сколько пинов мы сэкономили?»

4. «Бухгалтеры, во сколько обошелся наш прототип? А теперь представьте, что мы делаем это для реального "Умного дома" от Яндекса или Aqara. Чем их решение будет отличаться от нашего?»

На что акцентировать внимание

Проблема «Зависания» и NaN: датчик DHT11 имеет встроенный АЦП и очень медленный. Он не может отдавать данные чаще, чем раз в 1-2 секунды. Если студент напишет опрос в цикле loop() без задержки, датчик «захлебнется» и начнет возвращать NaN (Not a Number).

Это идеальный повод обсудить с Программистом и Тестировщиком важность таймингов в embedded-системах и необходимость обработки исключений (isnan()).

Инициализация Min/Max: в задании 2 (максимумы и минимумы) студенты часто делают ошибку: float minTemp = 0;. В итоге минимум всегда остается 0.

На этапе планирования направьте Программиста к мысли: «При первом включении и минимум, и максимум должны быть равны первому считанному значению с датчика» или минимум должен быть достаточно большим числом, а максимум – нулём.

Питание сервопривода: если студент подключит сервопривод к пину 5V Arduino, при повороте вала (открытие форточки) произойдет просадка напряжения, и DHT11 может выдавать ошибки.

Правило урока: если команда столкнется с этим, пусть они сами методом проб и ошибок поймут, что нужно разделить питание или добавить конденсатор. Это формирует **ПК 2.4 (Тестирование и верификация)**.

Практическая работа
Датчик температуры и влажности DHT11

Цель: научиться подключать и определять показания температуры и влажности с датчика DHT11

Краткие теоретические сведения

Существует семейство датчиков влажности и температуры DHT (DHT11, DHT22 и др.), которые используются для построения домашней метеостанции. Одновременно измерять температуру и относительную влажность выглядит оправданным, поскольку второе напрямую зависит от первого. Так, повышение температуры батарей центрального отопления приводит к уменьшению относительной влажности воздуха.

Считается, что оптимальное значение влажности около 50% – именно при такой влажности растения, люди и животные чувствуют себя комфортно.



Датчик температуры и влажности DHT11 является дешёвым датчиком, который удобно использовать в домашних условиях в учебных целях. Он не

обеспечивает точные показания и имеет ограниченный диапазон измерений. Но тем не менее, он очень популярен из-за своей простоты. Существуют более дорогие аналоги, в частности его старший брат DHT22.

Датчики DHT состоят из двух основных частей: ёмкостный датчик влажности и термистор. Также в корпусе установлен простенький чип для преобразования аналогового сигнала в цифровой. Считывать цифровой сигнал на выходе достаточно просто, можно использовать любой контроллер, не только Arduino.

Библиотека для DHT

В нашем наборе модульное исполнение датчика с тремя ножками и готовым установленным резистором.

```
#include "DHT.h"
#define DHTPIN 2     // к какому цифровому пину подключаем
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");
  dht.begin(); }
void loop() {  // Wait a few seconds between measurements.
  delay(2000);
  float h = dht.readHumidity(); // Read temperature as Celsius (the default)
  float t = dht.readTemperature(); // Check if any reads failed and exit early
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return; }
  Serial.print("Humidity: "); Serial.print(h); Serial.print(" %\t");
  Serial.print("Temperature: "); Serial.print(t); Serial.println(" *C ");}
```

СЦЕНАРИЙ УРОКА 4. RGB-светодиод

Тема: Разработка «умного» RGB-светильника

Продолжительность: 90 минут.

Формат: ролевая деловая игра, перевёрнутый класс.

ЭТАП 0. Подготовка (Перевёрнутый класс)

За неделю до урока студенты выбирают роли или им назначает преподаватель.

Домашнее задание: изучить физическую природу ШИМ (широтно-импульсной модуляции), разобраться в понятиях «общий анод» и «общий катод», вспомнить работу функции map() и analogWrite().

ЭТАП 1. Брифинг и «Заказ» (3 минуты)

Преподаватель (директор инжинирингового бюро): «Коллеги, к нам обратился владелец сети современных коворкингов. Ему нужно разработать прототип "Умного светильника" для зон отдыха. Клиент устал от скучного белого света и хочет управлять атмосферой в помещении. У нас 80 минут на создание MVP (минимального жизнеспособного продукта). Начинаем спринт-планирование!»

ЭТАП 2. Спринт-планирование / Ролевая игра (17 минут)

Заказчик озвучивает требования.

Требование 1 (Базовое): «Мне нужны готовые пресеты (сохраненный набор настроек для быстрого изменения параметров): Красный, Зеленый, Синий, Желтый, Фиолетовый, Голубой и Белый. Переключение по кнопке или автоматически».

Требование 2 (Плавность): «Никаких резких миганий! Переходы между цветами должны быть плавными, как в кино».

Требование 2 (Ручное управление): «Я хочу сам смешивать цвета! Дайте мне три ручки, чтобы я мог настроить идеальный оттенок для вечеринки или работы».

Менеджер переводит пожелания Заказчика в ТЗ. «Итак, нам нужно 3 аналоговых входа для ручек, 3 ШИМ-выхода для RGB-светодиода и логика пресетов».

Кладовщик оценивает набор. «У нас есть RGB светодиоды, 3 потенциометра по 10 кОм. У нас в наборе RGB светодиоды с общим минусом. Это критически важно для Программиста, иначе цвета будут инвертированы!»

Схемотехник рисует схему. «К каждому цветовому кристаллу (R, G, B) обязательно нужен токоограничивающий резистор на 220 Ом, иначе мы сожжем диод за секунду. НО, если вы используете микросхему из набора, то обратите внимание, так резисторы уже встроенные. Потенциометры вешаем на A0, A1, A2. RGB подключаем к пинам 9, 10, 11 (они поддерживают ШИМ, помечены тильдой ~)».

Программист даёт советы по формированию кода. Основные вопросы: как сформировать базовые цвета. Как сделать плавное загорание, затухание (напомнить циклы). «Для ручного режима считываем analogRead() (получаем 0-1023) и обязательно используем map(), чтобы перевести это в диапазон analogWrite() (0-255). Для пресетов используем switch-case или массивы».

Тестировщик пишет чек-лист:

Тест 1: выкрутить все ручки в минимум и максимум. Проверить, получаются ли чистые базовые цвета.

Тест 2: смешать красный и зеленый. Должен получиться желтый.

Тест 3: проверить, не перегревается ли светодиод при работе на максимальной яркости (белый цвет).

ЭТАП 3. Реализация / Лабораторная работа (50 минут)

Студенты собирают схему и пишут код.

Если команда застряла на инверсии цветов: «Почему у вас при максимуме на ручке цвет гаснет, а при минимуме – загорается на полную? Вспомните, что сказал Кладовщик про общий катод/анод!»

Если команда поставила резисторы несмотря на то, что они есть встроенные обратить внимание на тусклость цветов.

Если команда использует delay() для плавности: «Клиент жаловался, что пока светильник "перетекает" из красного в синий, ручки потенциометров не реагируют. Как сделать плавность без блокировки?» *(Подсказка: использовать маленькие шаги в цикле for или millis())*.

На 20-й минуте этапа преподаватель вбрасывает событие: *«Коллеги, срочная новость от Заказчика! Он хочет добавить функцию "Плавное засыпание". При нажатии дополнительной кнопки светильник должен не просто погаснуть, а медленно, в течение 3 секунд, уменьшить яркость всех каналов до нуля, используя цикл for и analogWrite!»*.

ЭТАП 4. Приёмка и Тестирование (10 минут)

Преподаватель с Заказчиком и Тестировщиком из проверяют выполнение чек-листа: крутит ручки, проверяет смешение цветов (R+G=Желтый, R+B=Фиолетовый), нажимает кнопку "Плавное засыпание". Если цвет не тот (например, вместо желтого получается тусклый оранжевый), Тестировщик записывает «Баг-репорт»: *«Неверная калибровка каналов или отсутствует функция map()»*.

ЭТАП 5. Рефлексия (10 минут)

Фронтальная беседа. Преподаватель подводит итоги, связывая опыт студентов с реальной электроникой.

Вопросы для обсуждения:

1. «Команда 2, почему при включении "Белого цвета" (все три канала на 255) ваш светодиод начал светить тускло или плата Arduino нагрелась?» *(Обсуждение токопотребления RGB диода и необходимости внешнего питания для мощных LED-лент).*

2. «Программисты, что бы произошло, если бы мы подали на пин RGB светодиода просто HIGH или LOW без ШИМ?» *(Вывод: мы бы получили только 8 базовых цветов, а не 16 миллионов оттенков).* Студенты часто подключают RGB к обычным цифровым пинам (например, 2, 3, 4) и удивляются, почему analogWrite() не работает (светодиод просто мигает или горит наполовину).

3. «Как функция map() спасла нас от написания сложных математических формул при переводе показаний потенциометра в яркость?»

4. **Связь с курсовым проектированием:** этот урок является идеальной базой для тем КП, связанных с «Умным домом», системами освещения и индикации статусов. Напомните студентам, что код смешивания цветов через потенциометры они могут использовать для создания физического интерфейса управления в своих курсовых проектах.

Оценивание на уроке (Формирующее)

«5» (Отлично): светильник собирается, ручки плавно меняют цвета (16 млн оттенков), пресеты работают, функция плавное затухание реализована. Тестировщик подтверждает отсутствие мерцания.

«4» (Хорошо): ручное управление работает, но пресеты «прыгают» или инвертированы (путаница с катодом/анодом), либо функция плавного затухания работает рывками (из-за большого delay() в цикле for).

«3» (Удовлетворительно): студенты смогли только зажечь базовые цвета (Пример 1). Ручное управление (Пример 3) не работает из-за отсутствия map() или подключения не к ШИМ-пинам.

«2» (Неудовлетворительно): светодиод не горит, сгорел резистор, схема собрана неверно.

Практическая работа

Тема: Управление RGB светодиодом на Arduino

Цель работы:

– изучить принципы работы RGB светодиодов;

– научиться подключать RGB светодиод к микроконтроллеру Arduino;

– освоить управление цветом и яркостью свечения с помощью ШИМ (широтно-импульсной модуляции);

– получить практические навыки программирования плавных цветовых переходов;

– изучить работу с аналоговыми входами для управления устройством.

Необходимое оборудование

Компонент	Количество	Назначение
Arduino Uno / Nano	1 шт.	Плата микроконтроллера
RGB светодиод (общий катод или анод)	1 шт.	Основной индикатор
Потенциометр 10 кОм	3 шт.	Управление яркостью цветов
Макетная плата	1 шт.	Для сборки схемы
Соединительные провода	10-12 шт.	Для подключения компонентов

Теоретические сведения

RGB светодиоды объединяют три кристалла разных цветов в одном корпусе. RGB LED имеет 4 вывода – один общий (анод или катод имеет самый длинный вывод) и три цветовых вывода. К каждому цветовому выходу следует подключать резистор. Кроме того, модуль RGB LED Arduino может сразу монтироваться на плате и иметь встроенные резисторы.



Пример 1. Базовые цвета

Подключаем к 11, 10, 9 входам!

Программирование

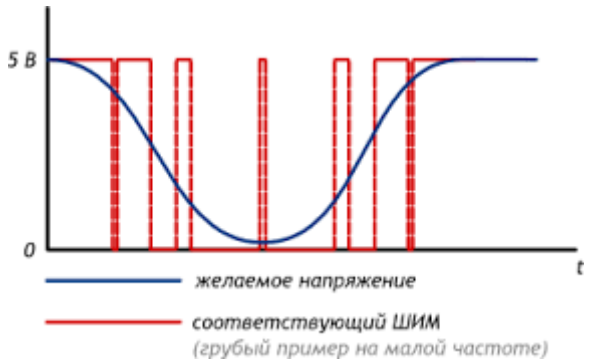
```
int R = 11;
int G = 10;
int B = 9;
void setup() {
  pinMode(R, OUTPUT);  // используем Pin 11 для вывода
  pinMode(G, OUTPUT);  // используем Pin 10 для вывода
  pinMode(B, OUTPUT);  // используем Pin 9 для вывода
}
```

```
void loop() {
  digitalWrite(R, HIGH); // включаем красный свет
  digitalWrite(G, LOW);
  digitalWrite(B, LOW);
  delay(1000); // устанавливаем паузу для эффекта
}
```

Пример 2. Плавное зажигание светодиодов с использованием ШИМ

Управление rgb светодиодом на Arduino можно сделать плавным, используя аналоговые выходы с «ШИМ». Для этого ножки светодиода необходимо подключить к аналоговым выходам, например, к пинам 11, 10 и 9. И подавать на аналоговые выходы микроконтроллера различные значения ШИМ (PWM), для этого воспользуемся циклом for, с помощью которого можно повторять нужные команды в программе.

Что такое ШИМ? Процесс управления мощностью путём изменения длительности импульса при постоянной частоте.



Один выход ШИМ платы Arduino позволяет установить 256 (0 - 255) уровней яркости.

```
void loop() {
   // плавное включение/выключение красного цвета
  for (int i = 0; i <= 255; i++) {
    analogWrite(R, i);
    delay(2);
  }
  for (int i = 255; i >= 0; i--) {
```

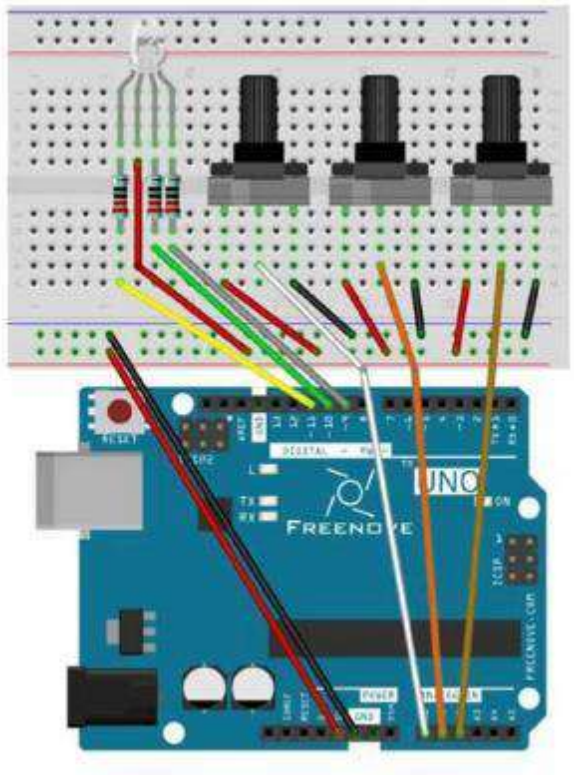
```
    analogWrite(R, i);
    delay(2);
  }
}
```

Пример 3. Управление RGB светодиодом с помощью потенциометров

Напишем скетч, который будет считывать состояние 3-х потенциометров и преобразовывать их в сигнал на цифровых выводах.

Один выход ШИМ платы Arduino позволяет установить 256 (0 - 255) уровней яркости. Таким образом, благодаря сочетанию трех различных цветов, RGB светодиод может иметь $256^3 = 16\ 777\ 216$ цветов свечения.

Подключим 3 потенциометра ко входам A0, A1 и A2 и контакты RGB светодиода к выводам D9, D10, D11 платы Arduino Uno.




```
int R = 11;
int G = 10;
int B = 9;
void setup() {
  pinMode(R, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(B, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  int adcValue;
  adcValue = analogRead(A0);
  analogWrite(R, map(adcValue, 0, 1023, 0, 255));
  adcValue = analogRead(A1);
  analogWrite(G, map(adcValue, 0, 1023, 0, 255));
  adcValue = analogRead(A2);
  analogWrite(B, map(adcValue, 0, 1023, 0, 255)); delay(100);}
```

СЦЕНАРИЙ УРОКА 5. Часы реального времени

Тема: Разработка системы автополива по расписанию с использованием модуля часов реального времени DS1302

Продолжительность: 90 минут (2 академических часа)

Формат: ролевая игра (Agile/Scrum), перевёрнутый класс, командная работа.

ЭТАП 0. Подготовка (Перевёрнутый класс / Домашнее задание)

За неделю до занятия преподаватель пре распределяет роли.

Задача студентов до урока:

1. Изучить принцип работы модуля DS1302 (Real Time Clock), назначение контактов (VCC, GND, CLK, DAT, RST).
2. Разобраться с библиотекой RTC by Makuna (или аналогичной DS1302RTC), понять, как устанавливать и считывать время.
3. Повторить работу с LCD-дисплеем 1602 по интерфейсу I2C (библиотека LiquidCrystal_I2C).
4. Подготовить краткий доклад по своей роли (Схемотехник ищет распиновку DS1302 и примеры подключения, Программист изучает функции библиотеки RTC).

ЭТАП 1. Брифинг и «Заказ» (5 минут)

Преподаватель (директор инжинирингового бюро): «Коллеги, к нам обратился владелец небольшой теплицы. Он уезжает в командировки и не может каждый день поливать растения. Ему нужна система автополива, которая будет работать по расписанию, даже когда его нет дома. У нас 80 минут на создание минимального жизнеспособного продукта. Команды, начинаем спринт-планирование!»

ЭТАП 2. Спринт-планирование / Ролевая игра (15 минут)

Заказчик озвучивает требования.

Требование 1 (Базовое): «Я хочу, чтобы система показывала текущее время и дату на дисплее. Чтобы я всегда знал, который час!»

Требование 2 (Автополив): «Полив должен включаться автоматически в 8:00 утра и в 20:00 вечера. Реле насоса должно работать ровно 30 секунд, потом выключаться!»

Требование 3 (Ручной режим): «Иногда мне нужно полить вручную. Добавьте кнопку – нажал, и насос работает, пока держу кнопку!»

Секретное требование (Каприз): «А если батарейка в часах сядет, система не должна поливать в случайное время! Пусть мигает индикатор ошибки!»

Менеджер переводит пожелания на технический язык, фиксирует ТЗ на листе А3. Уточняет у Заказчика: «Под "30 секунд" вы имеете в виду точное время? Что если насос засорится?»

Кладовщик оценивает стартовый набор. «DS1302 есть, LCD 1602 с I2C есть, реле есть, кнопка есть. **Важный нюанс:** у DS1302 должна быть батарейка CR2032, иначе при отключении питания время сбросится на 01.01.2000. Заказчик, учтите это в эксплуатации!»

Схемотехник рисует схему. «DS1302 подключаем по протоколу ThreeWire: CLK к пину 6, DAT к пину 7, RST к пину 8. LCD вешаем на I2C (A4, A5), чтобы не тратить пины. Реле насоса – через транзистор на пин 10. Кнопка ручного полива – на пин 2 с подтяжкой к питанию. **Важно:** DS1302 нельзя подключать к 3.3V, только к 5V!»

Программист составляет алгоритм программы. «Используем библиотеку RTC by Makuna и LiquidCrystal_I2C. При первом запуске устанавливаем время через __DATE__ и __TIME__ (компилятор подставит текущее время ПК). В цикле loop() каждую секунду считываем время, сравниваем с 8:00 и 20:00, если совпадает – включаем реле на 30 секунд через millis(). В режиме проверки лучше сделать 5 секунд, чтобы долго не ждать. Для ручного режима – опрос кнопки с устранением дребезга».

Тестировщик пишет чек-лист приемки.

Тест 1: отключить питание на 10 секунд, включить снова – время должно сохраниться (если есть батарейка).

Тест 2: установить время 7:59, подождать 1 минуту – реле должно включиться ровно в 8:00 на 30 секунд.

Тест 3: нажать кнопку ручного полива – реле должно работать, пока кнопка нажата.

Тест 4: вытащить батарейку из DS1302, отключить питание, включить снова – на дисплее должно появиться сообщение "RTC ERROR".

Бухгалтер Считает смету прототипа для отчета Заказчику. «DS1302 с батарейкой – 150 руб., LCD 1602 I2C – 200 руб., реле – 80 руб., кнопка – 20 руб., Arduino Nano – 300 руб. Итого: 750 рублей. В серийном производстве будет дешевле».

ЭТАП 3. Реализация / Лабораторная работа (60 минут)

Студенты собирают схему на макетной плате и пишут код.

Если команда застряла на установке времени: «Как вы узнаете, что время установилось правильно? Выведите его в Serial Monitor при первом запуске. Что произойдет, если загрузить скетч дважды?» *(Ожидаемый ответ: время установится заново, нужно закомментировать строку установки времени после первого запуска).*

Если команда застряла на автополиве: «Вы используете delay(30000) для 30-секундного полива? А что будет, если в это время нажать кнопку ручного режима? Как сделать неблокирующий таймер?» *(Подсказка: использовать millis() для отсчета 30 секунд).*

Если команда игнорирует батарейку: «Заказчик просил защиту от сброса времени. Как вы проверите, что RTC работает? Что вернет библиотека, если модуль не отвечает?» *(Подсказка: использовать функцию Rtc.IsDateTimeValid()).*

Если команда путается в протоколе ThreeWire: «Почему DS1302 использует три провода (CLK, DAT, RST), а не I2C (два провода)? В чем разница?» *(Обсуждение специфики протокола).*

«Внезапное изменение задания»

На 20-й минуте этапа преподаватель вбрасывает событие:

«Коллеги, срочная новость от Заказчика! Он сказал, что в выходные дни (суббота и воскресенье) он бывает дома и поливает сам. Добавьте функцию: если день недели – суббота или воскресенье, автополив должен отключаться!»

Это заставляет студентов изучить, как DS1302 хранит день недели (функция now.DayOfWeek()). Добавить условие if (dayOfWeek != 6 && dayOfWeek != 7) перед включением реле. Понять, что день недели в библиотеке может нумероваться с 1 (понедельник) или с 0 (воскресенье) – нужно проверить документацию.

ЭТАП 4. Приёмка и тестирование (10 минут)

Преподаватель с Заказчиком и Тестировщиком проверяют выполнение чек-листа:

– отключает питание на 10 секунд, включает снова – проверяет, сохранилось ли время.

– устанавливает время 7:59 (через код или вручную), ждет 1 минуту – проверяет срабатывание в 8:00.

– нажимает кнопку ручного полива – проверяет реакцию.

– вытаскивает батарейку из DS1302, отключает питание, включает снова проверяет появление ошибки.

ЭТАП 5. Рефлексия (10 минут)

Фронтальная беседа. Преподаватель подводит итоги, связывая опыт студентов с реальной инженерией.

Вопросы для обсуждения

1. «Команда 2, у вас после отключения питания время сбросилось на 01.01.2000. В чем была причина?» *(Ожидаемый ответ: забыли установить батарейку CR2032 в модуль DS1302, или батарейка была разряжена).*

2. «Программисты, кто использовал delay(30000) для полива? Что произошло, когда вы пытались нажать кнопку ручного режима во время полива?» *(Вывод о важности неблокирующего кода на millis()).*

3. «Схемотехники, почему DS1302 использует три провода (CLK, DAT, RST), а не два как I2C? В чем преимущества и недостатки этого протокола?» *(Обсуждение: ThreeWire проще в реализации, но медленнее и требует больше пинов).*

4. «Заказчики, чье ТЗ было самым сложным? Удалось ли Программистам реализовать функцию "отключить полив в выходные"? Как вы определили день недели?»

5. «Как бы вы улучшили этот проект, если бы у вас было еще 2 недели?» *(Добавить настройку времени полива через кнопки, добавить датчик влажности почвы, чтобы полив включался только если почва сухая, подключить Wi-Fi для удаленного мониторинга).*

6. «Бухгалтеры, во сколько обошелся ваш прототип? А теперь представьте, что мы делаем это для реального "Умного дома" от Яндекса или Aqara. Чем их решение будет отличаться?» *(Вывод: у них свои RTC-чипы с более высокой точностью, облачный синхронизация времени через NTP, мобильное приложение).*

На что акцентировать внимание

Проблема установки времени при каждом запуске: самая частая ошибка студентов – они оставляют в коде строку Rtc.SetDateTime(compiled) в функции setup(). При каждой загрузке скетча время сбрасывается на момент компиляции.

На этапе Спринт-планирования направьте Программиста к мысли: «Нужно установить время только один раз, а потом закомментировать эту строку». Или использовать флаг в EEPROM: «Если первый запуск – установить время, иначе – считать из RTC».

Проблема с батарейкой CR2032: может попасться разряженная батарейка. Студенты должны распознать эту проблему, суметь воспользоваться мультиметром для определения уровня заряда батареи. При отключении питания время сбрасывается.

Перед началом работы Кладовщик обязан проверить наличие батарейки (и заряд) во всех модулях.

Формат времени и день недели: в библиотеке RTC by Makuna день недели может нумероваться по-разному (1 = понедельник или 1 = воскресенье). Это важно для реализации функции "отключить полив в выходные".

Пусть студенты сами экспериментально определят, какой день недели соответствует какому числу. Это формирует навык работы с документацией.

Неблокирующий таймер для полива: студенты часто используют delay(30000) для 30-секундного полива. Это блокирует весь цикл loop(), и кнопка ручного режима не реагирует.

На этапе Рефлексии обсудите использование millis() для отсчета времени полива без блокировки.

Шкала оценок:

«5» (Отлично): 10-12 баллов. Команда собрала устройство, реализовала все требования Заказчика (включая защиту от сброса времени и функцию "отключить в выходные"), прошла кросс-тестирование без критических багов. Использован неблокирующий код (millis()).

«4» (Хорошо): 7-9 баллов. Устройство работает, но есть недочеты (например, используется delay() для полива, или не реализована защита от сброса времени, или Change Request выполнен частично).

«3» (Удовлетворительно): 5-6 баллов. Команда смогла только вывести время на LCD (базовое задание из методички), но не смогла реализовать автополив или ручной режим.

«2» (Неудовлетворительно): менее 5 баллов. Модуль DS1302 не считывает данные, время не отображается, схема собрана неверно (перепутаны пины CLK, DAT, RST).

На что акцентировать внимание

Типичная ошибка «Время сбрасывается при каждом запуске»: если студенты оставили в коде Rtc.SetDateTime(compiled) в setup(), время будет

сбрасываться при каждой загрузке скетча. Это отличный повод обсудить с ними концепцию «первого запуска» и использование EEPROM для хранения флага инициализации.

Проблема «Реле не выключается»: если студенты используют delay(30000) для полива, реле будет включено 30 секунд, но в это время весь код заблокирован. Кнопка ручного режима не будет реагировать. Это наглядная демонстрация важности неблокирующего кода.

Связь с реальными проектами: на этапе Ретроспективы покажите студентам реальные системы автополива (например, от компаний Gardena или Xiaomi). Обсудите, чем их решения отличаются от студенческого прототипа (точность RTC, облачная синхронизация, мобильное приложение, датчики влажности почвы).

Варианты усложнения для сильных команд:

Добавить настройку времени полива через кнопки (без перепрошивки).

Добавить датчик влажности почвы – полив включается только если почва сухая И наступило время полива.

Добавить логирование всех поливов в Serial Monitor с датой и временем.

Реализовать функцию «отпуск» – отключить автополив на заданный период через комбинацию кнопок.

Практическая работа

Подключение DS1302 к Arduino

Цель: научиться подключать и использовать элемент «Часы реального времени» в проектах.

Теоретические сведения

Основная и единственная микросхема на модуле, это DS1302, как еще называют «Real Time Clock» (Часы Реального Времени), сокращенно RTC. Данная микросхема поддерживает секунды, минуты, часы, день недели, дата, месяц и год информацию, а так же следит за количеством дней в месяце и делает поправку на високосный год. Микросхема работает в 24-часовом или 12-часовом формате с индикатором AM/PM.



Подключение

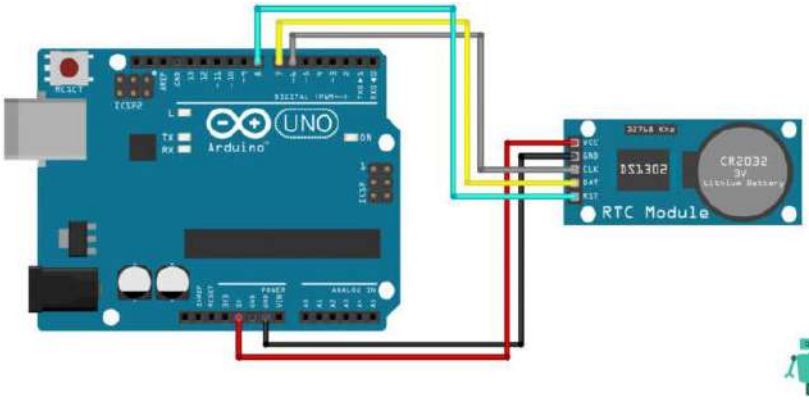
Назначение контактов:

► **Vcc и GND** – Питание модуля, 2 – 5.5 В;

► **CLK –** Вход тактовой частоты последовательных данных.

► **DAT –** Ввод/вывод последовательных данных

► **RST –** Включение логике RTC



Установка библиотеки

Для работы с модулем необходимо установить библиотеку «**RTC by Makuna**», для этого, перейдите в **Скетч** —> **Подключить библиотеку** —> **Управление библиотеками**.

В строке поиска введите «**1302**», найдите библиотеку « **RTC by Makuna** » и установите данную библиотеку.

Программирование

```
#include <ThreeWire.h>     // Подключаем библиотеку ThreeWire
#include <RtcDS1302.h>     // Подключаем библиотеку RtcDS1302
ThreeWire myWire(7,6,8);  // Указываем вывода IO, SCLK, CE
RtcDS1302<ThreeWire> Rtc(myWire);
void setup () // Установка последовательной связи на скорости 9600
    Serial.print("Data: ");  // Отправка данных на последовательный порт
    Serial.println(__DATE__);     // Получение даты и времени с ПК
    Serial.print("Time: ");  // Отправка данных на последовательный порт
    Serial.println(__TIME__);    // Получение даты и времени с ПК
    Rtc.Begin();         // Инициализация RTC
    RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
         // Копирование даты и времени в compiled
    Rtc.SetDateTime(compiled);            // Установка времени
    Serial.println();   }
void loop () {
  Rtc.GetDateTime();
  RtcDateTime now = Rtc.GetDateTime();
  Serial.print("DATA RTC ");   // Отправка данных на последовательный порт
  Serial.print(now.Month());      // Отправка месяца
  Serial.print("."); // Отправка данных на последовательный порт
  Serial.print(now.Day());         // Отправка дня
  Serial.print(".");  // Отправка данных на последовательный порт
  Serial.print(now.Year());        // Отправка года
  Serial.print(" TIME RTC ");   // Отправка данных на последовательный порт
  Serial.print(now.Hour());                // Отправка часа
  Serial.print(":");  // Отправка данных на последовательный порт
  Serial.print(now.Minute());     // Отправка минут
  Serial.print(":");   // Отправка данных на последовательный порт
  Serial.println(now.Second());   // Отправка секунд
  delay(2000);      }
```

СЦЕНАРИЙ УРОКА 6. «Умная оранжерея»

Тема: Разработка IoT-системы «Умная оранжерея» с мобильным интерфейсом

Формат: ролевая деловая игра (Agile/Scrum), перевёрнутый класс, командная работа.

Продолжительность: 180 минут (две пары).

Повышаем реалистичность: в центре класса стоит реальный горшок с растением. Студенты не просто пишут код, они «спасают» редкий цветок.



Максимальная интеграция – затрагивает сразу несколько тем: датчики (DHT11, LDR), актуаторы (сервопривод, реле), межсистемные интерфейсы (Bluetooth) и мобильные приложения.

В игру встроен механизм внезапное изменение ТЗ, что проверяет стрессоустойчивость и гибкость команды.

Связь с курсовым проектированием: урок является репетицией к темам «Умная теплица» и другим, затрагивающим мобильное приложение.

ЭТАП 0. Подготовка (Перевёрнутый класс / Домашнее задание)

За неделю до занятия между студентами распределяются роли. Схемотехнику нужно повторить распиновку DHT11, LDR, сервопривода, RGB и реле. Подготовить черновики схем.

К этому занятию самая большая подготовка у Программиста. Он должен разобраться с понятием гистерезиса. И объяснить группе, что это такое, для чего нужно и как применить.

На этой практической работе предполагается, что со всеми устройствами по отдельности студенты уже знакомились, есть методички по подключению и управлению, можно открыть свои старые работы, которыми можно воспользоваться. Опыт создания мобильного приложения тоже имеется. То есть цель работы – обобщение. интеграция имеющихся знаний и ресурсов в один проект.

ЭТАП 1. Брифинг и «Заказ» (5 минут)

Преподаватель: «Коллеги, к нам обратился владелец элитного ресторана. Он купил за баснословные деньги редкое тропическое растение, но оно постоянно гибнет. Ему нужна полностью автоматизированная IoT-система «Умная оранжерея». У нас 80 минут на создание MVP. Команды, начинаем спринт-планирование!»

ЭТАП 2. Спринт-планирование / Ролевая игра (15 минут)



Идет активное обсуждение. Преподаватель выступает в роли технического директора, направляет дискуссию, но не дает готовых решений (в идеале процессом руководит Менеджер, но, так как часто он не

справляется, то преподаватель может взять на себя функции модератора беседы).

Заказчик озвучивает требования.

Требование 1 (Климат): «Растению нужна влажность строго 60-70%. Если падает – включайте увлажнитель (реле). Если выше 80% – открывайте форточку (сервопривод)!» *(Здесь студенты должны вспомнить про гистерезис из методички по КП, иначе реле будет бесконечно щелкать).*

Требование 2 (Свет): «Ночью должен гореть тусклый красный свет (RGB или обычный светодиод), днем – яркий белый. Определите это сами по освещенности!»

Требование 3 (Мобильное приложение): «Я хочу видеть температуру на телефоне и иметь кнопку «Аварийный полив»!»

Менеджер переводит «хотелки» на технический язык. «Так, увлажнитель – это реле, лучше светодиод, форточка – сервопривод. Заказчик, мы используем Bluetooth или Wi-Fi?». Фиксирует ТЗ на доске или листе.

Кладовщик оценивает стартовый набор. «У нас есть реле (свеодиоды), но нет водяного насоса. Мы будем имитировать его включение зажиганием светодиода или зуммера. Заказчик, устроит?». Определяет, какой датчик для измерения температуры и влажности есть у нас в наборе. Есть RGB-светодиод для фиолетового цвета.

Схемотехник представляет схемы (лучше сделать заранее распечатки). На словах должен акцентировать внимание на спорных моментах. Например, датчики влажности различных производителей могут по-разному подключаться. У нас так… Фоторезистор подключается через сопротивление 1Ком, если достаточно светло и 10Ком, если темно. Обязательно напомнить про подключение BlueTooth модуля «крест-накрест» RX-TX, TX-RX. И про то, что когда код загружаем в плату Arduino, то RX и TX от неё надо отключить.

Важный момент: Схемотехник должен напомнить команде, что реле и сервопривод нельзя питать напрямую от 5V Arduino, нужна внешняя нагрузка или хотя бы раздельное питание.

Программист: напоминает о наличии библиотеки DHT11. «Для влажности используем dht.readHumidity(). Обязательно добавим гистерезис 5%. Для мобильного приложения используем Bluetooth модуль HC-05». Программист может посоветовать, что, если затруднительно создать приложение, то можно использовать программу Arduino Bluetooth Controller для приёма и отображение сигналов.

Тестировщик пишет чек-лист приемки *до начала сборки*:

Тест 1: подышать на датчик DHT11 (влажность должна вырасти, должно сработать реле).

Тест 2: закрыть датчик света пальцем (имитация ночи) – должен загореться красный светодиод.

Тест 3: нажать кнопку в мобильном приложении – должно сработать «аварийное» действие и т.п.

ЭТАП 3. Реализация / Лабораторная работа (120 минут)



Студенты собирают схему на макетной плате и пишут код. Преподаватель направляет, консультирует.

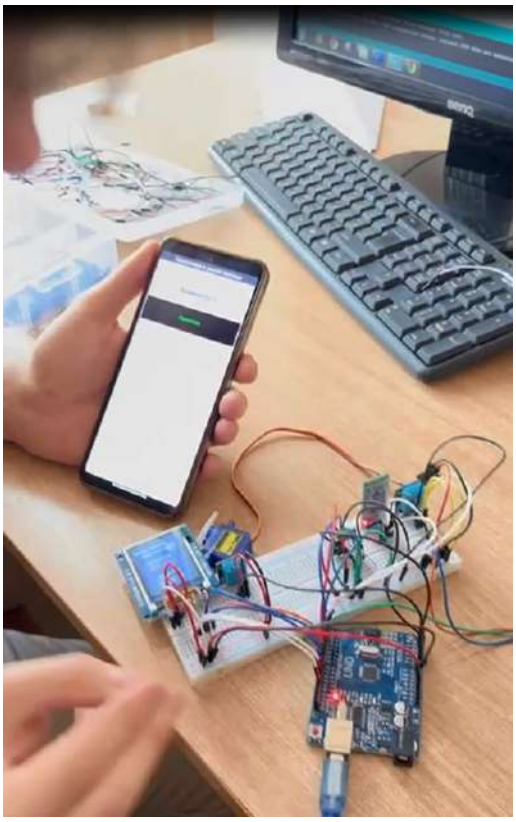
Если команда застряла на гистерезисе: «Коллеги, ваше реле щелкает как пулемет. Что такое гистерезис?»

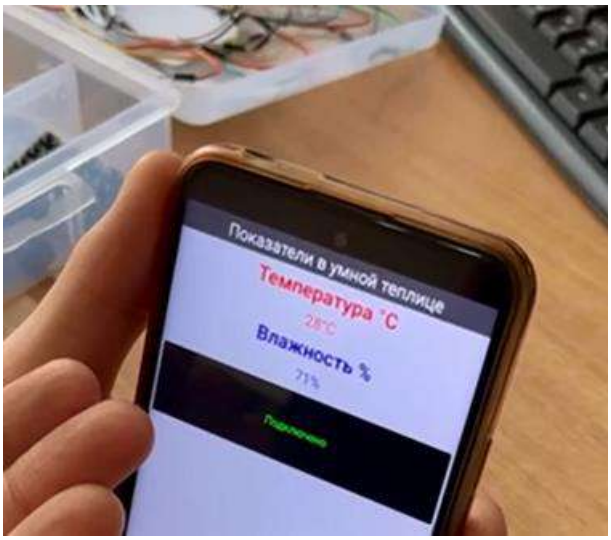
Если команда игнорирует мобильное приложение: «Заказчик нервничает, он не видит данных на телефоне. Программист, как вы передаете строку с температурой в Serial/Bluetooth?»

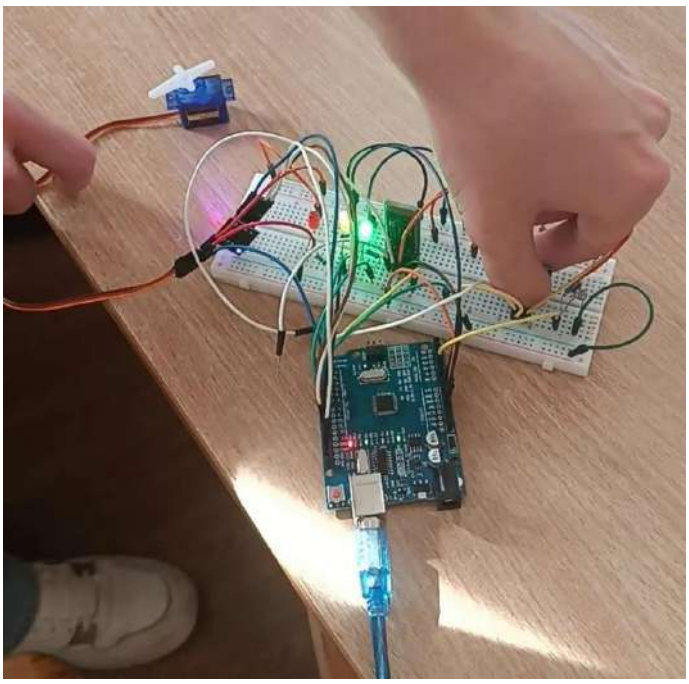
На 20-й минуте этапа преподаватель вбрасывает в игру событие. Подходит к командам и говорит: *«Коллеги, срочная новость от Заказчика! Он прочитал статью, что его растение любит "тропический ливень". Добавьте в код функцию: каждые 30 секунд реле должно включаться на 2 секунды, имитируя дождь, но только если сейчас "день" (светло)!»* Это заставляет студентов экстренно править код, используя millis() вместо delay(), что является высшим пилотажем в Arduino.

ЭТАП 4. Приёмка и тестирование (15 минут)

Преподаватель с Заказчиком и Тестировщиком проверяют выполнение чек-листа







ЭТАП 5. Рефлексия (15 минут)

Фронтальная беседа. Преподаватель подводит итоги, связывая опыт студентов с реальной инженерией.

Вопросы для обсуждения (направленные на развитие технического мышления)

1. «Команда 3, у вас реле включалось и выключалось 10 раз за минуту при влажности 65%. В чем была ошибка?» *(Ожидаемый ответ: Не учли гистерезис. Мы добавили порог включения 60% и выключения 70%).*

2. «Программисты, кто использовал delay(30000) для имитации "ливня"? Что произошло с опросом датчиков и кнопкой из мобильного приложения в эти 30 секунд?» *(Вывод о важности неблокирующего кода на millis()).*

3. «Схемотехники, почему при включении реле (имитации насоса) у вас перезагружалась плата Arduino?» *(Обсуждение электромагнитных помех и необходимости раздельного питания/конденсаторов).*

4. «Бухгалтеры, во сколько обошелся ваш прототип? А теперь представьте, что мы делаем это для реального агрохолдинга. Чем их решение будет отличаться?» *(Вывод: промышленные контроллеры, IP68 корпуса, LoRaWAN вместо Bluetooth).*

ПРАКТИЧЕСКАЯ РАБОТА

Тема: Система мониторинга микроклимата с мобильным интерфейсом на базе Arduino и Bluetooth

Цель работы:

– научиться организовывать беспроводную связь между Arduino и мобильным устройством с помощью Bluetooth-модуля HC-05/HC-06.

– освоить работу с библиотекой SoftwareSerial для создания дополнительного последовательного порта.

– интегрировать датчик DHT11 и Bluetooth-модуль в единую систему.

– разработать мобильное приложение в MIT App Inventor для отображения показаний датчиков в реальном времени.

– получить опыт создания полноценного IoT-прототипа «Умный дом».

Теоретические сведения

1.1. Bluetooth-модуль HC-05/HC-06

Модули HC-05 и HC-06 – это популярные Bluetooth-адаптеры для связи микроконтроллеров с мобильными устройствами по протоколу UART (последовательный порт).

Основные характеристики:

– рабочее напряжение: 3.3В – 6В (логические уровни 3.3В)

– скорость передачи по умолчанию: 9600 бод

– дальность связи: до 10 метров

Важное отличие:

HC-05 – может работать как Master (ведущий) и Slave (ведомый), поддерживает AT-команды для настройки.

HC-06 – работает только как Slave, проще в использовании для базовых задач.

1.2. Библиотека SoftwareSerial

Стандартный последовательный порт Arduino (пины 0 и 1) используется для связи с компьютером через USB и отладки через Serial Monitor. Чтобы

одновременно работать с Bluetooth и монитором порта, используется библиотека SoftwareSerial, которая позволяет создать **дополнительный программный последовательный порт** на любых цифровых пинах.

```
#include <SoftwareSerial.h>
SoftwareSerial bt(8, 9); // RX, TX – пины Arduino для Bluetooth
```

1.3. Формат передачи данных

Для удобного парсинга данных в мобильном приложении используется простой текстовый протокол с разделителями:

```
25;60;
```

где первое число – температура, второе – влажность, а точка с запятой (;) служит разделителем полей и маркером конца пакета.

2. Необходимое оборудование

Компонент	Количество	Назначение
Arduino Uno / Nano	1 шт.	Плата микроконтроллера
Датчик DHT11 (модуль с 3 пинами)	1 шт.	Измерение температуры и влажности
Bluetooth-модуль HC-05 или HC-06	1 шт.	Беспроводная связь
Резистор 1 кОм	1 шт.	Делитель напряжения для RX Bluetooth
Резистор 2 кОм	1 шт.	Делитель напряжения для RX Bluetooth
Макетная плата	1 шт.	Для сборки схемы
Соединительные провода	10-12 шт.	Для подключения компонентов
Смартфон на Android	1 шт.	Для запуска мобильного приложения

3. Схема подключения

3.1. Подключение DHT11

Пин DHT11	Пин Arduino
VCC	5V
GND	GND
DATA	2

3.2. Подключение Bluetooth-модуля HC-05/HC-06

Пин Bluetooth	Пин Arduino	Примечание
VCC	5V	Питание
GND	GND	Общий провод
TXD	8 (RX SoftwareSerial)	Прямое подключение
RXD	9 (TX SoftwareSerial)	Через делитель напряжения!

Подключите датчик температуры и Bluetooth-модуль к контроллеру Arduino. Обратите внимание на подключение 8, 9 на пины RX, TX.

```
#include <SoftwareSerial.h>
SoftwareSerial bt(8, 9); // RX, TX
#include <DHT.h>
#define DHTPIN 2
#define DHTTYPE DHT11
#define dataPin 2
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  dht.begin(); // запускаем датчик
 Serial.begin(9600);
 bt.begin(9600);
 Serial.println("Ready");}
void loop(){
delay(2000); // задержка 2 сек между считываниями
  int hum = dht.readHumidity(); // считываем влажность
  int temp = dht.readTemperature(); // считываем температуру
bt.print(temp); //отправляем показания в MIT App
 bt.print(";");
bt.print(hum); // отправляем показания в MIT App
 bt.println(";");
  delay(1000);}
```

Пояснения к коду:

1. Библиотека SoftwareSerial создаёт виртуальный последовательный порт на пинах 8 и 9, освобождая стандартные пины 0 и 1 для отладки.
2. Задержка delay(2000) обязательна – DHT11 не может отдавать данные чаще, чем раз в 1-2 секунды.
3. Функция isnan() проверяет, не вернул ли датчик ошибку (NaN – Not a Number).
4. Формат temp;hum; выбран для удобного парсинга в мобильном приложении: символ ; служит разделителем полей.

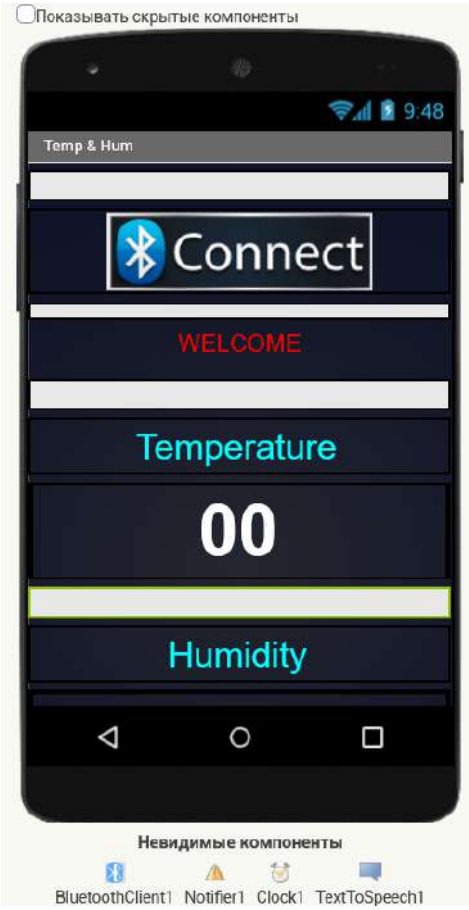
5. Разработка мобильного приложения в MIT App Inventor

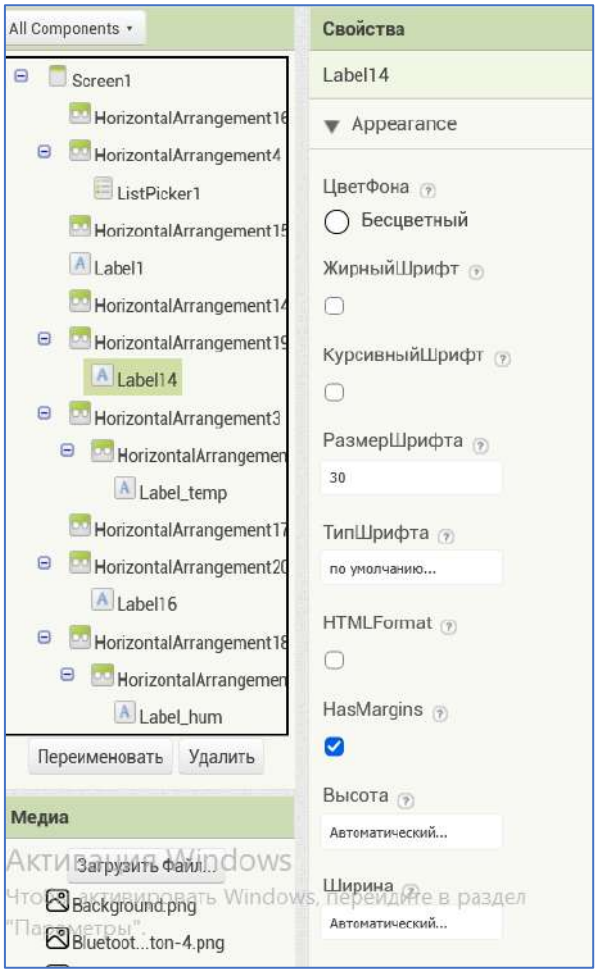
5.1. Дизайн интерфейса (Designer)

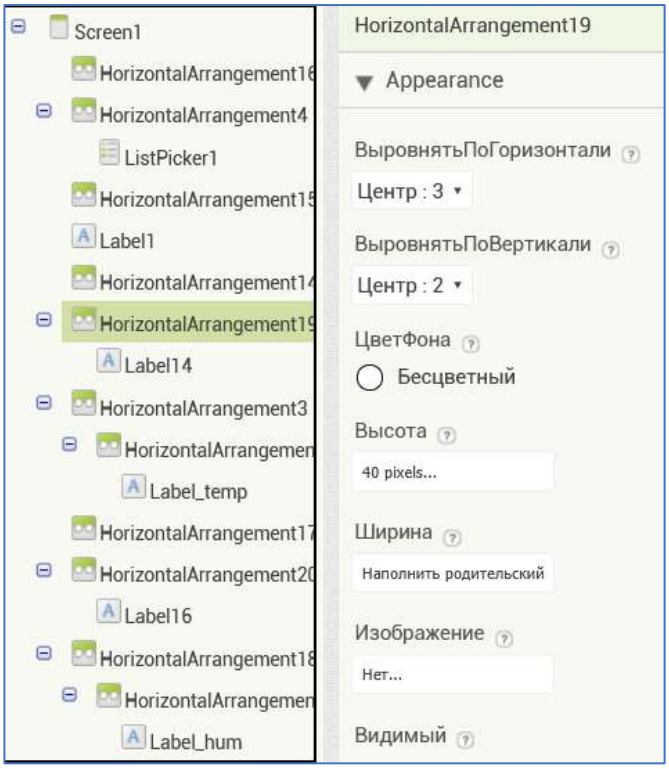
Перейдите на ai2.appinventor.mit.edu и создайте новый проект «DHT Monitor».

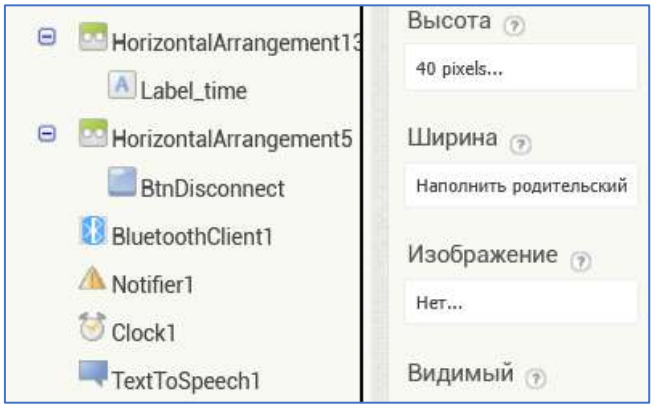
5. **Добавьте компоненты и выполните настройку их свойств, можно ориентироваться на предложенный интерфейс. можно создать свой.**

Компонент	**Имя**	**Назначение**
Label	LblStatus	Индикатор статуса подключения
Label	LblTemp	Отображение температуры
Label	LblHum	Отображение влажности
Button	BtnConnect	Кнопка подключения
ListPicker	LstDevices	Список Bluetooth-устройств
BluetoothClient	BTClient	Компонент связи
Clock	ClkTimer	Таймер опроса (Interval: 2000 мс)









5.2. Блочное программирование (Blocks)

когда BtnConnect.Щелчок

делать: вызвать LstDevices.Open

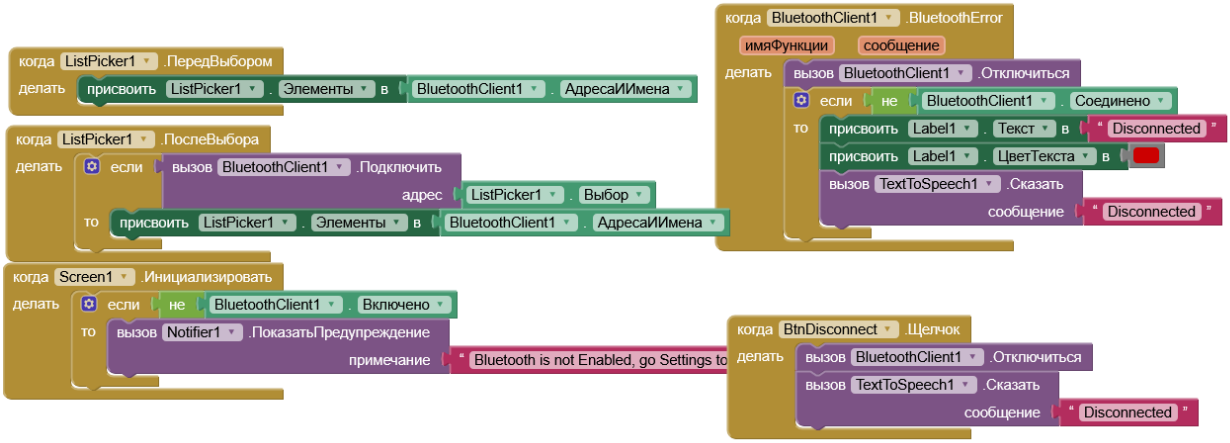
когда LstDevices.ПослеВыбора

делать:

если BTClient.Подключить(LstDevices.Выбор)

то: присвоить LblStatus.Текст ← "Connected"

присвоить ClkTimer.TimerEnabled ← true



Блок 2: Приём и парсинг данных (в ClkTimer.Timer)

когда ClkTimer.Таймер

делать:

если BTClient.Соединено И BTClient.БайтовДоступноДляПринятия > 0

то: инициализировать локальную data ← BTClient.ПолучитьТекст

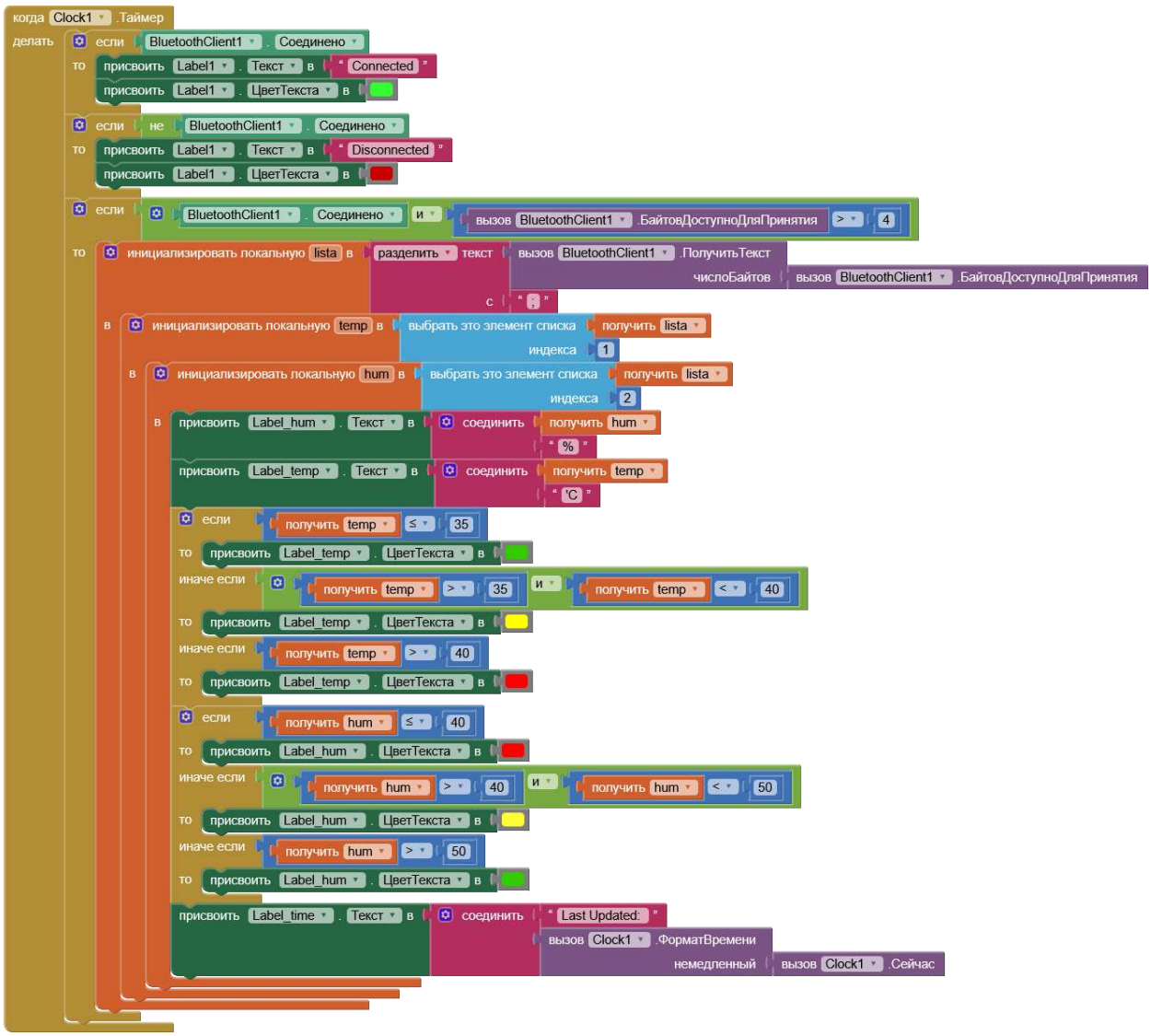
инициализировать локальную parts ← разделить data по ";"

если длина(parts) >= 2

то: присвоить LblTemp.Текст ← соединить(element 1 из parts, " °C")

присвоить LblHum.Текст ← соединить(element 2 из parts, " %")

Пояснение: Функция разделить текст разбивает строку "25;60;" по символу ; и создаёт список ["25", "60", ""]. Мы берём первый и второй элементы.



6. Задания для самостоятельной работы

Задание 1. Базовый уровень (Модификация передачи данных)

1.1. Модифицируйте код Arduino так, чтобы данные отправлялись только при изменении температуры или влажности более чем на 1 единицу (защита от «спама» в Bluetooth-канал). 1.2. Добавьте в передачу данных третий параметр **индекс комфорта** (рассчитывается по формуле: если температура 20-25°C и влажность 40-60%, то «Комфорт», иначе «Дискомфорт»).

Задание 2. Средний уровень (Доработка мобильного приложения)

2.1. Добавьте в приложение **цветовую индикацию**:

– температура выше 30°C – красный текст

– температура 20-30°C – зелёный текст

– температура ниже 20°C – синий текст 2.2. Добавьте кнопку **«Экспорт данных»**, которая сохраняет последние 10 показаний в файл на телефоне (используйте компонент TinyDB).

Задание 3. Продвинутый уровень (Двусторонняя связь)

3.1. Реализуйте **управление реле** с телефона. Добавьте в приложение кнопку «Включить вентилятор», которая отправляет в Arduino символ F. Arduino при получении F должна включать реле на пине 13. 3.2. Добавьте **автоматическое переподключение**: если связь оборвалась (например, Arduino перезагрузилась), приложение должно автоматически пытаться reconnect каждые 5 секунд.

Подсказка для Arduino:

```
void loop() {
  // ... отправка данных ...
  if (bt.available()) {
    char cmd = bt.read();
    if (cmd == 'F') {
      digitalWrite(13, HIGH);
      bt.println("Fan ON");
    } else if (cmd == 'f') {
      digitalWrite(13, LOW);
      bt.println("Fan OFF");
    }
  }
}
```

Сложности применения метода и пути их преодоления

Внедрение ролевой игры и элементов Agile в технические дисциплины – для меня это сложный вариант проведение занятия. Можно сказать, что это вызов педагогическим способностям. Провести обычное традиционное занятие, где преподаватель объясняет, а студенты молча собирают схему по методичке и пишут программу, намного легче. Инновационный формат требует от преподавателя колоссальной энергии, гибкости и готовности отказаться от привычного контроля.

Ниже я подробно описываю основные «подводные камни», с которыми сталкиваюсь на практике, и то, как мы учимся с ними работать.

1. Сопротивление модели «Перевернутого класса» и отложенная мотивация

Не люблю модель «перевернутого класса». Я понимаю, что сама могу объяснить материал лучше. Многолетний опыт помогает акцентировать внимание на главном и существенном, спрогнозировать типичные ошибки и сразу их предупредить. Но правильно ли это, когда обучающие получают материал «на блюдечке», развивается ли у них при этом навык анализа? Может быть дать им совершать ошибки, на которых, как не банально звучит, они будут учиться.

Студенты изначально не проявляют энтузиазма к домашней подготовке. Им непривычно и неинтересно заранее искать распиновку датчиков или читать про библиотеки, если они не видят в этом прямой и немедленной выгоды. На старте урока часто чувствуется, что роли выбраны формально, а теория не изучена. **Как это преодолевается?** Здесь работает эффект **отложенной мотивации**. Как только начинается этап «Спринт-планирования» и Заказчик начинает выдвигать свои (порой безумные) требования, студенты мгновенно осознают: их незнание материала приведет к провалу переговоров. Воодушевление и азарт приходят не *до* урока, а *в процессе*. После занятия студенты обычно уходят с горящими глазами и хорошим впечатлением, потому что они сами «добыли» знания, чтобы решить поставленную задачу.

2. Психологический барьер «отпускания контроля» (Трудно не вмешиваться)

Это, пожалуй, самая большая личная сложность для преподавателя. Когда видишь, что Схемотехник неправильно подключает провода, а Программист пишет заведомо нерабочий цикл, рука так и тянется подойти и сказать: «Так не пойдет, делайте вот так». В традиционном уроке мы привыкли сразу давать «правильный» ответ, экономя время.

Всегда очень хочется подкорректировать даже требования Заказчика. «А можно ведь ещё …». Человек знакомый с темой видит гораздо больше вариантов решения задачи. И мне, как преподавателю хочется, чтобы студенты их тоже увидели, а это уже вмешательство в процесс, который делегировали студентам.

Необходимо осознанно переключать себя из позиции «Транслятора знаний» в позицию **Тьютора**. Я разрешаю студентам совершать ошибки, если они не ведут к порче оборудования. «Продуктивное затруднение» – это часть обучения. Вместо того чтобы давать готовое решение, я задаю наводящие вопросы: «А что скажет на это Кладовщик?», «Хватит ли тока на этом пине?». Студент сам должен дойти до истины, даже если это займет на 10 минут больше.

3. Импровизация вместо жесткого сценария

Разработанные мной подробные сценарии и карточки Заказчика **никогда не выдаются студентам**. Они существуют только для преподавателя как «аптечка первой помощи» на случай, если разговор зайдет в тупик или Заказчик потребует нереализуемых функций (например, «хочу, чтобы Ардуино распознавало лица»). Из-за этого беседа почти никогда не идет строго по плану, преподавателю приходится постоянно импровизировать на ходу.

Как можно это преодолеть? Сценарий выступает не как жесткий регламент, а как **карта возможных ветвлений**. Импровизация становится главным инструментом. Если Заказчик просит невозможного, я не говорю

«нет». Я передаю мяч Кладовщику: «Кладовщик, у нас есть в наборе сканер сетчатки? Нет? Тогда что мы можем предложить Заказчику в качестве компромисса?». Это учит студентов главному инженерному навыку – искать компромисс между «хочу» и «могу».

4. Высокая ресурсоемкость и риски тайминга

Обычная лабораторная работа укладывается в тайминг с точностью до минуты. В ролевой игре этап обсуждения и споров может затянуться, и на физическую сборку схемы не останется времени. **Как это преодолевается.** Введение правила **«Тихого часа»** и жесткого тайминга для каждого этапа. Я планировала использовать таймер на экране. Если на этапе планирования начинаются бесконечные споры, я выступаю в роли директора компании и говорю: «Коллеги, время спринт-планирования вышло. Утверждаем ТЗ в текущем виде, все недочеты будем править на этапе тестирования». Это учит студентов работать в условиях дедлайнов.

5. Дисбаланс вклада участников команды

В любой группе найдутся студенты, которые будут пытаться «отсидеться», особенно если роль кажется им скучной (например, Бухгалтер или Тестировщик).

Ротация ролей и кросс-проверка. На следующем спринте (следующем занятии) студенты меняются ролями. Кроме того, на этапе защиты я задаю вопросы не только Программисту, но и Тестировщику: «Какие именно баги вы нашли?», или Бухгалтеру: «Почему вы выбрали именно этот датчик, он же дороже?». Это гарантирует, что каждый участник должен понимать общий контекст проекта.

Вывод. *Несмотря на описанные сложности и высокую трудоемкость подготовки, переход к интерактивным ролевым форматам полностью оправдывает затраченные усилия. Мы перестаем готовить студентов просто к "нажатию кнопок" по инструкции. Мы учим их общаться, аргументировать, работать в условиях неопределенности и брать на себя ответственность за инженерные решения. Именно эти навыки (Soft Skills в*

связке с Hard Skills) сегодня являются главным запросом от IT- и инжиниринговых компаний».

Список использованных источников

1. Боровский, А. С. Программирование микроконтроллера Arduino в информационно-управляющих системах : учебное пособие / А. С. Боровский, М. Ю. Шрейдер. – Оренбург : Оренбургский государственный университет, 2017. – 113 с. – ISBN 978-5-7410-1853-8. – Текст : электронный // IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/78913.html (дата обращения: 24.05.2024). – Режим доступа: для авториз. пользователей.
2. Задорожный, А. Ф. Основы построения микропроцессорных систем управления : учебное пособие / А. Ф. Задорожный, П. А. Графеев. – Новосибирск : Новосиб. гос. архитектурно-строит. ун-т (Сибстрин), 2018. – 105 с. – ISBN 978-5-7795-0846-9. – Текст : электронный // IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/85875.html (дата обращения: 24.05.2024). – Режим доступа: для авториз. пользователей.
3. Петин, В. А. Практическая энциклопедия Arduino / В. А. Петин, А. А. Биняковский. – 2-е изд. – Москва : ДМК Пресс, 2020. – 166 с. – ISBN 978-5-97060-798-5. – Текст : электронный // IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/124729.html (дата обращения: 24.05.2024). – Режим доступа: для авториз. пользователей.
4. Кульченко, А. Е. Using the Arduino platform in robotic development : учебное пособие / А. Е. Кульченко, М. Ю. Медведев. – Ростов-на-Дону, Таганрог : Изд-во Южного федерального университета, 2022. – 134 с. – ISBN 978-5-9275-4254-3. – Текст : электронный // IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/129088.html (дата обращения: 24.05.2024). – Режим доступа: для авториз. пользователей.