МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВВГУ»)
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ОТЧЕТ
ПО УЧЕБНОЙ ТЕХНОЛОГИЧЕСКОЙ (ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ) ПРАКТИКЕ
ООО «Центр Развития Робототехники»
г. Владивосток

Выполнил:
студент
гр. БИК-23-ИВ2 _____И.А. Кухарчик

Руководитель практики
от университета:
к. ф. - м. н., доцент каф. ИТС_____И.А. Белоус

Владивосток 2026

Содержание

Введение

Моя проектно-технологическая практика проходила в период с 15 июня по 18 июля 2026 года на базе ООО «Центр развития робототехники» в городе Владивостоке. Данная компания специализируется на образовательной и подводной робототехнике, сборке телеуправляемых необитаемых подводных аппаратов (ТНПА) и подготовке команд к профильным турнирам.

В соответствии с индивидуальным заданием и требованиями программы практики для освоения компетенций (ОПК-2, ОПК-4, ПКВ-1), передо мной были поставлены следующие задачи:

1) Изучение специфики работы и основных направлений деятельности предприятия.
2) Сбор и анализ технической информации по инфокоммуникационному оборудованию.
3) Аппаратная подготовка чипов для платформы DreamBot в соответствии с внутренними регламентами компании.
4) Прошивка микроконтроллеров и их первичная техническая диагностика.
5) Автоматизация производственного процесса переноса аудиоданных на подготовленные электронные модули.
6) Разработка программно-аппаратного комплекса для диагностики цифровых портов микроконтроллеров Arduino Nano.
7) Работа с контрольно-измерительной аппаратурой (осциллограф) для верификации сигналов.
8) Участие в технологической подготовке, планировании и развертывании материально-технической базы на новой производственной площадке предприятия с учетом требований эргономики и правил техники безопасности

В ходе практики был выполнен полный цикл индивидуальных работ: от написания автоматизирующего скрипта на Python до осциллографического контроля сигналов и массового тестирования интерфейсов передачи данных.

1 Общая характеристика предприятия ООО «Центр развития робототехники»

1.1 Основная информация о компании

ООО «Центр Развития Робототехники» было основано 24 декабря 2013 года (учредители - С.А. Мун и Д.Ю. Алексеев) и на сегодняшний день является одним из главных центров развития образовательной робототехники на Дальнем Востоке. Компания активно сотрудничает со студентами профильных ИТ-направлений, предоставляя отличную базу для реальных инженерных исследований и проектов.

1.2 Направления деятельности Ключевые векторы работы организации

1) Создание образовательных платформ (включая DreamBot) и наборов для сборки подводных аппаратов линейки MiddleROV.

2) Проведение кружков и образовательных программ для школьников и подростков.

3) Подготовка команд к международным и всероссийским турнирам (Российская робототехническая олимпиада, RoboCup Junior, MATE ROV Competition).

4) НИОКР в сфере инфокоммуникационных технологий и гидроакустики.

1.3 Организационная структура предприятия

Предприятие условно делится на три укрупненных блока в соответствии с рисунком 1:

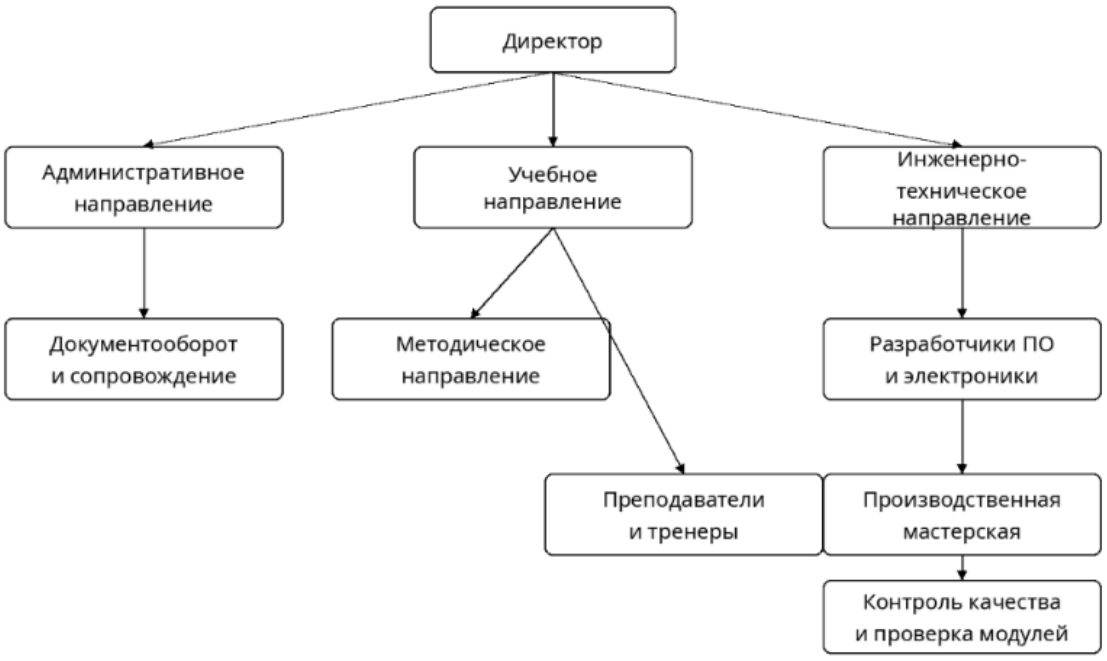


Рисунок 1 - План-схема структуры предприятия

1) Конструкторско-производственный (отвечает за проектирование и физическую сборку).

2) Подразделение программирования и тестирования (прошивка, отладка и диагностика).

3) Методический (сопровождение и интеграция готовых продуктов в образовательный процесс). В ходе практики я был прикреплен к подразделению, отвечающему за подготовку и тестирование аппаратной части выпускаемых изделий

Такое распределение обязанностей между подразделениями обеспечивает последовательное прохождение изделием всех этапов - от проектирования до итоговой проверки готовности к эксплуатации, а также позволяет своевременно выявлять и устранять узкие места производственного процесса за счёт регулярного взаимодействия специалистов разных подразделений.

2 Техническая база и объекты работ

2.1 Оборудование и инструменты, использованные в ходе практики

Для успешного выполнения производственных задач мне потребовались паяльная станция, программатор для первоначальной заливки кода в чипы, увеличительная оптика для контроля качества паяных соединений и ноутбук со средой разработки Arduino IDE.

Таблица 1 - Оборудование и инструменты, использованные в ходе практики

Наименование	Назначение
Программатор USBtinyISP	Прошивка микроконтроллерных чипов перед монтажом на платы
Персональный компьютер (ноутбук)	Работа со специализированным ПО для прошивки и диагностики плат
Увеличительная оптика	Контроль качества паяных соединений
Arduino IDE	Специализированное ПО для прошивки и диагностики плат
Python VSC (Visual Studio Code)	Среда программирования на языке Python для написания кода по автоматизации процессов
Arduino Mega/Nano/Uno	Отладочная плата для связки компьютера с микроконтроллерами
Тепловизор	Для контроля температуры микроконтроллера

2.2 Автоматизация с использованием Python

Для работы с файловой системой и устранения рутины был выбран язык Python в среде программирования Visual Studio Code, показанном на рисунке 2.1. С помощью стандартных встроенных библиотек os и shutil был реализован алгоритм переноса аудиофайлов из одного каталога в другой с автоматическим ведением подробного лог-файла.

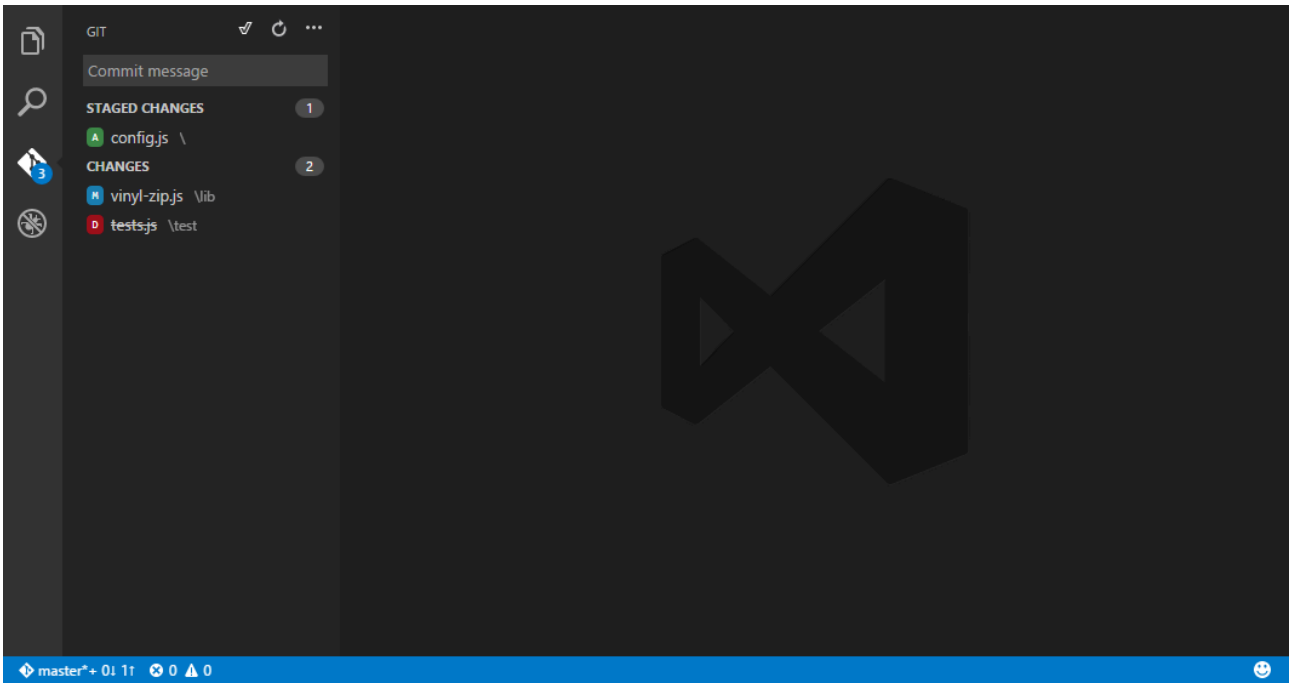


Рисунок 2.1 – Среда программирования Visual Studio Code

2.3 Программно-аппаратные средства диагностики микроконтроллеров

Одной из ключевых задач производственного процесса является контроль целостности и работоспособности цифровых портов ввода-вывода (I/O) на платах Arduino Nano. Для тестирования интерфейсов передачи данных без использования аппаратного UART-модуля (который занят связью с ПК) было принято решение использовать программную эмуляцию.

С помощью библиотеки SoftwareSerial в среде Arduino IDE любой цифровой пин контроллера может быть сконфигурирован как передатчик (TX) или приемник (RX). Для считывания эмулируемого сигнала использовался USB-UART конвертер (выполненный в форм-факторе USB-флешки). Перед началом массовой диагностики плат программный сигнал подлежал обязательной верификации на физическом уровне с применением осциллографа, что позволило подтвердить корректность формы и таймингов передаваемых пакетов данных.

3 Выполненные работы

3.1 Подготовка компонентов для пайки плат

Перед установкой на текстолит абсолютно все микроконтроллеры проходили этап предварительной программной заливки на программаторе показанном на рисунке 3.1.

Рисунок 3.1 – Программатор USBTinyISP

3.2 Программирование микроконтроллеров ATXC1200 с использованием программатора USBtinyISP

Основной и наиболее трудоёмкой частью практики стало массовое программирование микроконтроллеров ATXC1200 в корпусе SOIC-8. Данные микросхемы являются центральными вычислительными элементами для различных электронных модулей. В зависимости от назначения модуля (датчик линии, звуковой датчик, датчик освещённости, драйвер двигателя и т.д.) в контроллер загружалась соответствующая прошивка, содержащая

алгоритмы обработки сенсорных данных, формирования управляющих сигналов и взаимодействия с внешними устройствами по интерфейсам UART, I2C или SPI.

Этапы программирования одного микроконтроллера:

1) Визуальный входной контроль. Каждый микроконтроллер перед установкой в программатор подвергался тщательному осмотру под лупой. Проверялось отсутствие механических повреждений корпуса (сколов, трещин), загрязнений на выводах, следов окисления или коррозии. Также контролировалась правильность маркировки, чтобы исключить попадание компонентов других типов. При обнаружении дефектов микросхема откладывалась в отдельную тару для последующей утилизации или возврата поставщику.

2) Позиционирование в цанговом разъёме. Микросхема аккуратно захватывалась пинцетом за края корпуса и помещалась в переходную плату с цанговым разъёмом, предназначенным для корпусов SOIC-8. Ориентация осуществлялась по ключу - скосу на одном из углов корпуса, который должен совпадать с меткой на разъёме. Неправильная установка могла привести к подаче напряжения на несоответствующие выводы и выходу микросхемы из строя. После фиксации проверялась надёжность контакта: все восемь выводов должны были быть плотно прижаты цангами.

3) Подключение программатора USBtinyISP. Переходная плата соединялась с программатором USBtinyISP через стандартный 10-контактный шлейф (IDC-10). Программатор, в свою очередь, подключался к USB-порту компьютера. При этом на программаторе загорался светодиод, сигнализирующий о наличии питания и готовности к работе. Через монитор устройств Windows проверялось, что драйвер программатора установлен корректно и устройство отображается в системе как дополнительный COM-порт.

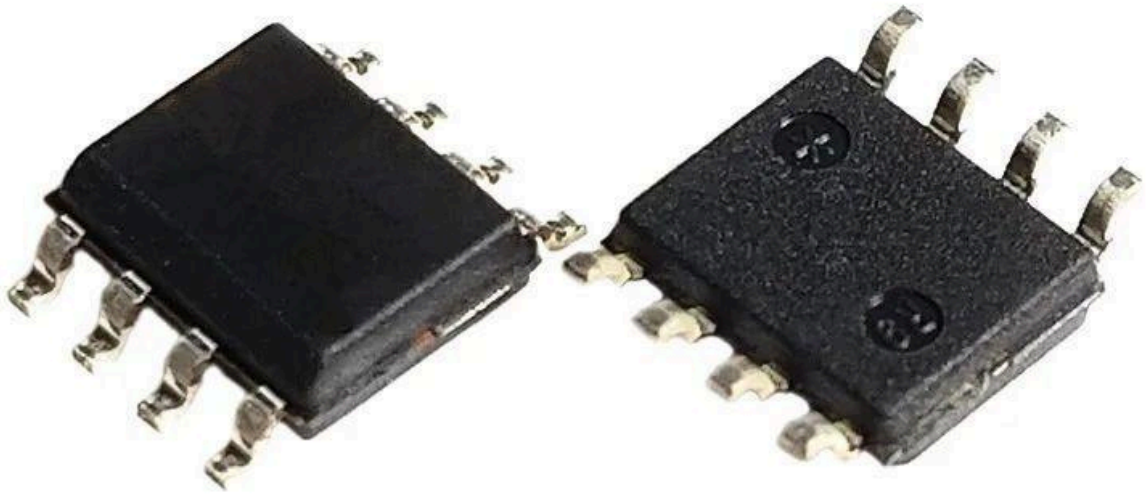


Рисунок 3.2 Микросхема ATXC1200, прошивка которой выполнялась в ходе прохождения практики

1) Запуск среды Arduino IDE и выбор конфигурации. Открывался файл скетча, соответствующий типу модуля, для которого предназначен контроллер.

В меню «Инструменты» выбирались следующие параметры:

1) плата: Arduino Nano (или Arduino Pro Mini, в зависимости от версии модуля);

процессор: ATmega328P (для совместимости с архитектурой AVR);

2) частота тактирования: 16 МГц (внешний кварцевый резонатор) или 8 МГц (внутренний RC-генератор) - уточнялось по документации модуля;

3) порт: соответствующий USBtinyISP;

4) программатор: USBtinyISP (или USBASP).

Кроме того, проверялись настройки оптимизации компилятора (размер кода, использование оперативной памяти) и версия ядра AVR, чтобы избежать конфликтов библиотек.

2) Компиляция и загрузка прошивки. После нажатия кнопки «Загрузить» начинался процесс компиляции исходного кода в машинный код для AVR, после чего среда IDE через программатор отправляла скомпилированный hex-файл в память программ микроконтроллера. Операция занимала от 15 до 30 секунд в зависимости от размера прошивки. На экране отображался лог процесса: «Подключение к программатору», «Стирание памяти», «Запись страниц», «Верификация». Успешное завершение подтверждалось сообщением «Загрузка завершена».

3) Верификация и проверка контрольной суммы. После записи выполнялась автоматическая проверка - программатор считывал записанные данные и сравнивал их с исходным hex-файлом. При обнаружении расхождений (из-за помех, плохого контакта или нестабильного питания) загрузка повторялась. В случае повторяющихся ошибок микросхема откладывалась для дополнительной диагностики.

4) Настройка фьюз-бит. Для корректной работы микроконтроллера в составе модуля дополнительно настраивались фьюз-биты - специальные конфигурационные регистры, определяющие источник тактового сигнала, уровень напряжения программирования, защиту памяти и другие важные параметры. Использовалась предустановленная конфигурация из методических указаний предприятия, которая гарантировала совместимость с остальной схемотехникой модуля.

После завершения программирования микроконтроллер извлекался из цангового разъёма с помощью пинцета и помещался в пластиковую тару с обозначением типа модуля и версии прошивки. Все операции выполнялись в перчатках для предотвращения загрязнения выводов кожным жиром, который мог бы ухудшить электрический контакт при последующей установке на печатную плату.

Всего за период практики мной таким образом было запрограммировано 220 микроконтроллеров ATXC1200. Благодаря отработке движений и настройке среды разработки время на одну микросхему удалось сократить с первоначальных 5-6 минут до 2-3 минут к концу практики. При этом процент брака из-за ошибок программирования не превысил 1,5%, что свидетельствует о высокой стабильности оборудования и правильности соблюдения технологической дисциплины.

Перечень элементов, необходимых для сборки одной платы и всей партии изделий (батареи, датчики расстояния, серводвигатели, кнопки, датчики касания, звуковые излучатели, мотор-редукторы, джойстики, датчики звука и освещения, датчики линии и

цвета, светодиоды, пульты управления), фиксировался в таблице учета в соответствии с рисунком 3.2. Такой учёт позволял планировать и контролировать подготовку компонентов, избегая нехватки или избытка материалов при параллельной работе нескольких практикантов. После прошивки микроконтроллеров, они складывались в емкость и отправлялись на пайку в мастерскую.

№ Компонента	Устройство	Количество на один набор	Количество всего (на все наборы)	Чипов (плат) подготовлено и передано в мастерскую	Чипов (плат) осталось подготовить	Плат успешно проверено и передано в мастерскую	Плат осталось проверить	Очередь прошивки (приоритет)
14	Батарейный блок	1	40	40	0	40	0	1
2	Датчик расстояния	1	40	40	0	40	0	2
11	Сервомотор	1	40	40	0	40	0	3
3	Кнопка	2	80	80	0	80	0	4
5	Датчик касания	2	80	80	0	80	0	5
6	Пищалка	1	40	40	0	40	0	6
12	Мотор-редуктор	2	80	80	0	0	80	7
4	Джойстик	1	40	40	0	0	40	8
8	Датчик звука	1	40	40	0	40	0	9
9	Датчик освещения	1	40	40	0	40	0	10
7	Датчик линии	2	80	80	0	80	0	11
1	Светодиод	3	120	120	0	120	0	12
10	Датчик цвета	1	40	40	0	40	0	13
13	Главный блок	1	40	40	0	0	40	14
15	Пульт	1	40	40	0	0	40	15
–	MP3	1	40	40	0	–	–	16
–	Bluetooth	2	80	80	0	–	–	17

Рисунок 3.2 - Таблица учета компонентов

3.3 Тестирование микроконтроллеров

Сразу после программной прошивки я проводил первичное нагрузочное тестирование: проверял корректность инициализации и стабильность отклика плат на тестовые команды. Плата подключалась к самосборной приставке (Gamepad) через USB-C и с помощью значений на экране приставки проверялась работоспособность и правильная прошивка чипа в соответствии с рисунком 3.3, так же проверка могла происходить на тестовом стенде Breadboard и Arduino Mega как показано на рисунке 3.4

3.4 Тестирование звуковых излучателей (пищалок)

Пищалки (пьезозуммеры) в наборах DreamBot служат для звуковой индикации событий - подачи сигналов об окончании программы, обнаружении препятствий, ошибках и т.п. Их проверка проводилась одновременно с тестированием управляющих микроконтроллеров, но также выполнялась и выборочная проверка самих излучателей как отдельных компонентов.

Методика проверки пищалок:

1) Визуальный осмотр. Проверялось отсутствие механических повреждений корпуса пьезоэлемента, целостность выводов и контактных площадок.

2) Электрический тест. С помощью мультиметра измерялось сопротивление пьезоизлучателя в деактивированном состоянии. Для исправного элемента сопротивление

должно было быть в пределах 10–100 кОм (в зависимости от модели). Короткое замыкание или обрыв свидетельствовали о браке.

3) Акустический тест. Пищалка подключалась к ШИМ-выводу Arduino, на который подавался меандр с частотой 1 кГц и амплитудой 5 В. На слух оценивалась громкость и чистота звука: он должен был быть отчётливым, без искажений и посторонних призвуков. Дополнительно проверялась работа на частотах 500 Гц, 2 кГц и 4 кГц для подтверждения широкополосности (если она требовалась по спецификации).

4) Тест на длительную работу. Пищалка включалась в непрерывном режиме на 60 секунд. Контролировалось, что громкость не падает, а сам излучатель не перегревается. Перегрев мог свидетельствовать о внутреннем коротком замыкании или несоответствии номинальной мощности в соответствии с рисунков 3.3.

Всего было проверено более 80 пищалок. Брак составил менее 2% (один излучатель оказался с обрывом, один - со значительно пониженной громкостью). Все исправные излучатели были допущены к комплектации наборов DreamBot.

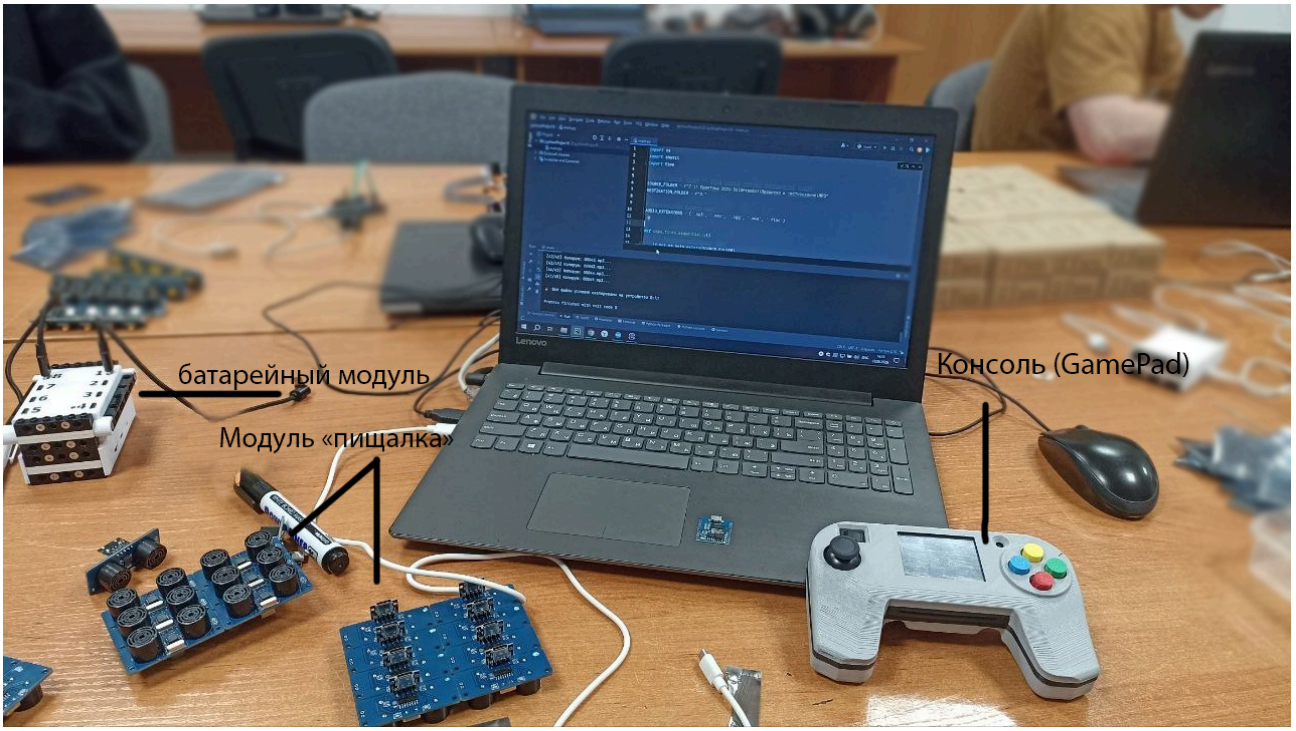


Рисунок 3.3 - Проверка микроконтроллеров

3.5 Тестирование датчиков касания

В ходе практики также выполнялась проверка датчиков касания, входящих в состав робототехнических наборов DreamBot. Датчики касания являются важными элементами интерактивных систем, позволяя роботу реагировать на прикосновения, что необходимо для выполнения соревновательных задач В отличие от механических кнопок, емкостные датчики касания (например, на основе микросхемы TTP223) не имеют подвижных частей и

срабатывают при изменении емкости, когда палец приближается к сенсорной площадке Принцип работы таких датчиков основан на изменении электрической емкости между сенсорным электродом и землей - при прикосновении пальца ёмкость увеличивается, и датчик формирует логический сигнал высокого уровня на выходе.

Методика проверки датчиков касания:

1. Визуальный осмотр и подготовка.Проверялось состояние сенсорной площадки - отсутствие загрязнений, царапин или окислений, которые могли бы повлиять на чувствительность. Контакты питания и вывода сигнала очищались от возможных загрязнений изопропиловым спиртом.

2. Подключение к тестовому стенду. Датчик касания подключался к цифровому входу тестового стенда на базе Arduino Uno: питание 5 В подавалось на VCC, GND - на землю, выходной сигнал (OUT) - на цифровой пин D2 Для обеспечения надежного контакта использовались соединительные провода с разъемами «папа-папа».

3. Загрузка тестовой программы. В микроконтроллер стенда загружался специализированный скетч-анализатор с использованием библиотеки Touch Sensor, которая обеспечивает считывание состояния датчика с подавлением дребезга контактов (debounce) Базовый код тестовой программы имел следующий вид:

cpp
Копировать
Скачать

```
#include <TouchSensor.h>

Touch Sensor touch(2);  // датчик подключен к пину D2

void setup() {
  Serial.begin(9600);
  Serial.println("Тестирование датчика касания");
}

void loop() {
  touch.update();  // обновление состояния датчика

  if (touch.justTouched()) {
    Serial.println("Прикосновение зафиксировано");
  }
```

```
  if (touch.justReleased()) {
    Serial.println("Прикосновение завершено");
  }

  delay(50);
}
```

Такая программа позволяла не только фиксировать сам факт касания, но и отслеживать момент начала и окончания прикосновения, что важно для интерактивных задач:

1) Тестирование реакции на прикосновение. При проведении пальцем по сенсорной площадке в мониторе последовательного порта должно было появляться сообщение «Прикосновение зафиксировано», а при отведении пальца - «Прикосновение завершено». Тест повторялся не менее 10 раз с чередованием коротких касаний (длительностью менее 0,5 с) и длительных (более 2 с) для проверки стабильности срабатывания.

2) Проверка чувствительности. Оценивалась реакция датчика на разную силу нажатия - от лёгкого прикосновения подушечкой пальца до уверенного нажатия. Ёмкостный датчик должен срабатывать даже при лёгком касании, так как его работа не зависит от силы давления, а только от изменения ёмкости. При этом датчик не должен давать ложных срабатываний без касания (например, из-за электромагнитных помех или статического электричества).

3) Тестирование на разных материалах. Для моделирования реальных условий эксплуатации проверялось срабатывание датчика при использовании различных предметов: через тонкую бумагу, через пластиковую прокладку и при касании металлическим предметом (которое, в отличие от пальца, не должно вызывать срабатывания ёмкостного датчика). Такая проверка позволяла выявить датчики с завышенной или заниженной чувствительностью.

4) Проверка длительной работы. Датчик подключался к стенду и оставлялся в режиме ожидания на 5 минут без внешних воздействий как показано на рисунке 3.4 - за это время не должно было появляться ложных срабатываний. Затем выполнялось 50 последовательных касаний с интервалом 0,5 с - все они должны были быть зафиксированы корректно без пропусков.

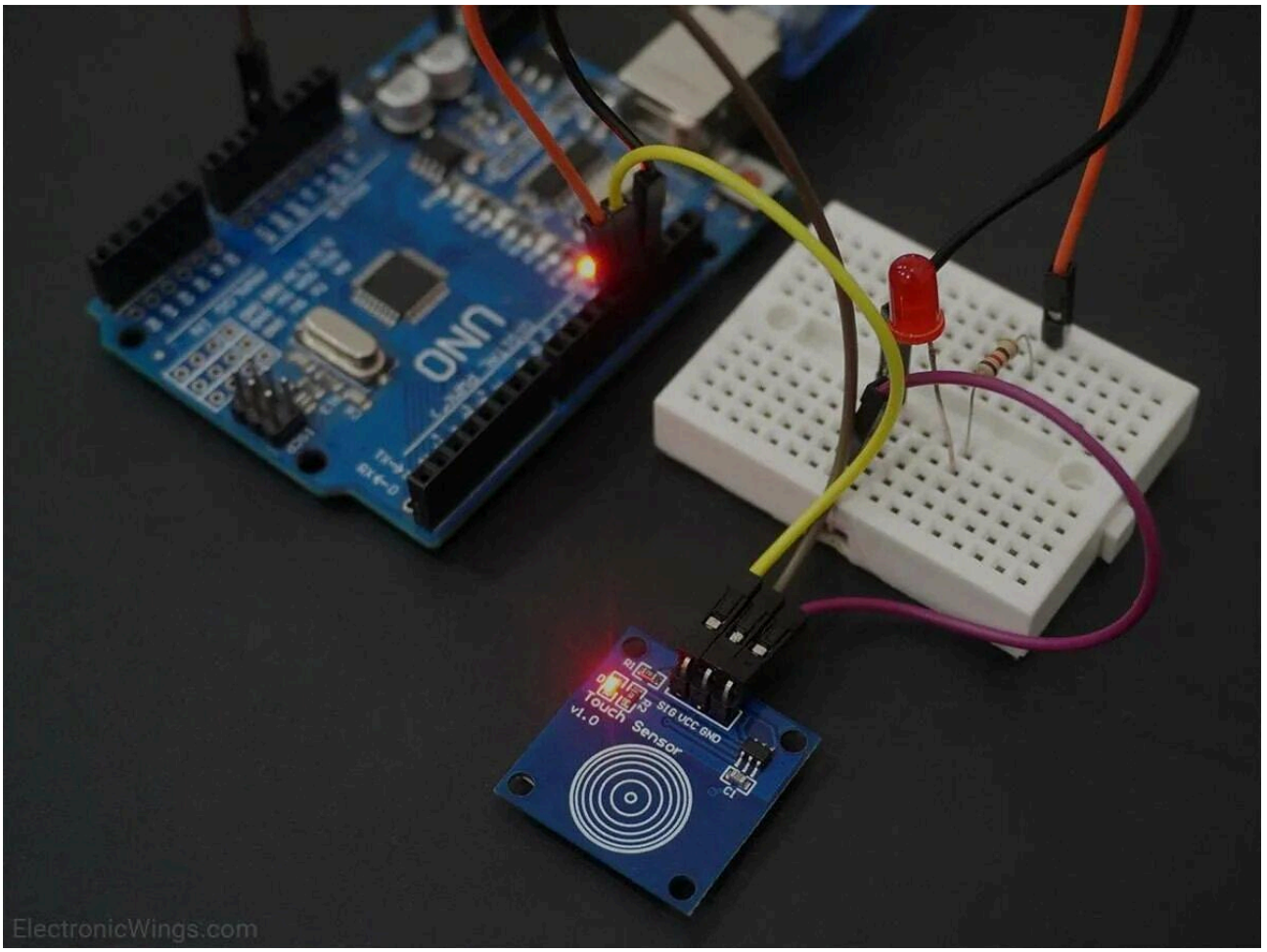


Рисунок 3.4 – проверка датчиков касания

Всего за период практики было проверено 65 датчиков касания. Бракованными признаны 3 датчика (около 4,6%):

1) два датчика имели заниженную чувствительность - срабатывали только при сильном нажатии, что недопустимо для ёмкостных датчиков;

2) один датчик выдавал ложные срабатывания без прикосновения, вероятно, из-за внутреннего дефекта микросхемы TTP223.

Исправные датчики маркировались и передавались для комплектации наборов DreamBot, где они используются для интерактивных сценариев - например, робот может реагировать на касание, останавливаясь или меняя направление движения Результаты тестирования были зафиксированы в протоколах и переданы руководителю практики с рекомендацией ужесточить входной контроль поставщика датчиков касания.

3.6 Результаты тестирования датчиков касания

В дополнение к основным количественным показателям, следует выделить результаты по проверке датчиков касания:

1) Протестировано датчиков касания: 65 шт. Брак: 3 шт. (4,6%). Основные причины брака: два датчика с заниженной чувствительностью (срабатывали только при сильном нажатии) и один датчик с ложными срабатываниями в состоянии покоя.

2) Время тестирования одного датчика составляло в среднем 1 минуту (включая все этапы проверки).

3) Рекомендация по результатам тестирования: ужесточить входной контроль данной категории компонентов у поставщика с проведением выборочной проверки чувствительности на предварительном этапе.

При прошивке использовавшихся типов плат в среде Arduino IDE применялись три различных набора параметров конфигурации, приведенные в таблице 2.

Таблица 2 - Настройки Arduino IDE для прошивки плат DreamBot

Параметр	Arduino Uno	Arduino Nano	Arduino Mega
Board (Плата)	Arduino Uno	Arduino Nano	Arduino Mega or Mega 2560
Processor (Процессор)	ATmega328P	ATmega328P (Old Bootloader)	ATmega2560
Port (Порт)	COM3 (или другой)	COM3 (или другой)	COM3 (или другой)
Upload Speed	115200	115200	115200
Programmer	AVRISP mkII	AVRISP mkII	AVRISP mkII
Микроконтроллер	ATmega328P	ATmega328P	ATmega2560
Цифровые пины	14	14	54
Аналоговые входы	6	8	16
UART порты	1	1	4
Флеш-память	32 Кб	32 Кб	256 Кб
ОЗУ (SRAM)	2 Кб	2 Кб	8 Кб

Размер платы, мм	68,6 × 53,3	45 × 18	101,5 × 53,3
Применение в DreamBot	Основная плата управления	Модули сенсоров	Расширенное управление моторами

3.4 Прошивка микроконтроллеров семейства Arduino

В ходе выполнения производственных задач по подготовке оборудования для робототехнических конструкторов DreamBot мной производилась массовая прошивка плат различных конфигураций семейства Arduino (включая Arduino Uno, Arduino Nano и Arduino Mega 2560).

Технологический процесс заливки управляющих программ выполнялся по следующему алгоритму:

1) Аппаратное подключение: Тестируемая плата подключалась к USB-интерфейсу рабочей станции . При успешной подаче питания на плате активировался светодиод PWR в соответствии с рисунком 3.7.

2) Конфигурирование среды разработки: В среде Arduino IDE производилась настройка параметров целевой платформы. В меню Инструменты -> Плата выбирался тип контроллера (например, Arduino Uno или Arduino Mega). При работе с платами Arduino Nano дополнительно настраивался тип процессора (ATmega328P или ATmega328P (Old Bootloader)) в зависимости от ревизии распаянного чипа и версии его загрузчика.

3) Выбор интерфейса: В диспетчере устройств определялся номер назначенного виртуального COM-порта (драйвер CH340/FTDI), после чего данный порт указывался в настройках подключения среды разработки.

4) Компиляция и загрузка: Инициализировался процесс компиляции исходного кода скетча. После успешной проверки синтаксиса утилита AVRDUDE производила очистку флеш-памяти контроллера и запись новой прошивки. Успешный сеанс связи и передачи данных визуально контролировался по циклическому мерцанию светодиодов RX и TX на плате устройства. Процесс считался завершенным при выводе сообщения «Загрузка завершена» (Done uploading) в консоли отладки среды Arduino IDE.

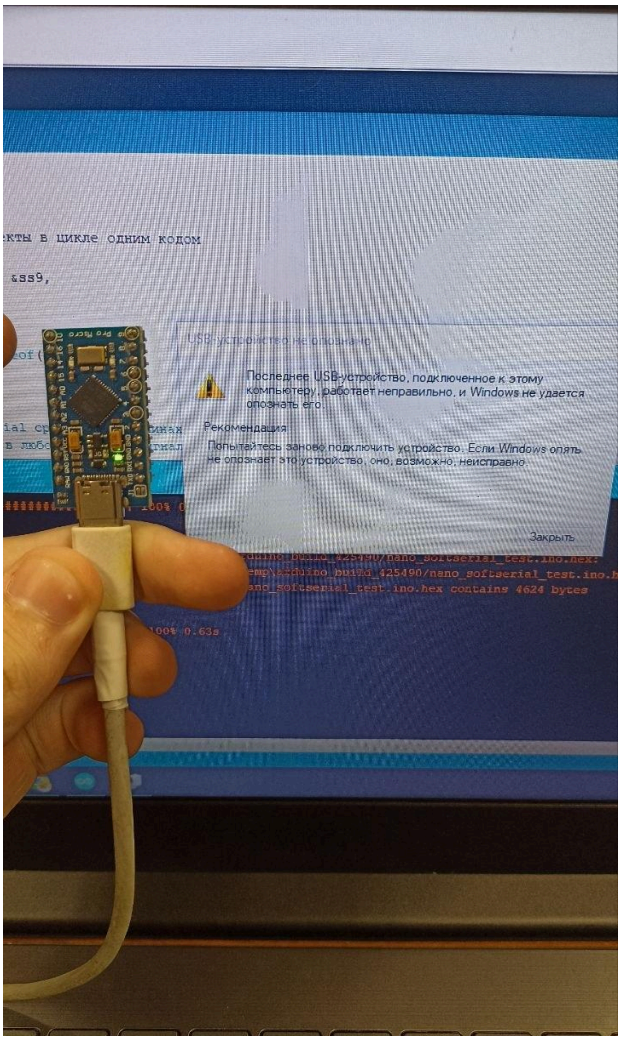


Рисунок 3.7 - Прошивка плат Arduino Nano

3.5 Участие в развертывании новой производственно-технической площадки

В связи с расширением производственных мощностей ООО «Центр Развития Робототехники» и открытием нового сборочно-логистического цеха, в период практики мной были выполнены работы по развертыванию инфраструктуры новой площадки.

В рамках данной задачи были выполнены следующие технологические операции:

1) Анализ и подготовка пространства: Оценка геометрических параметров помещений второго этажа нового корпуса для оптимального размещения монтажных и сборочных столов.

2) Транспортировка и позиционирование оборудования: Организация перемещения, подъема на второй этаж и позиционирования тяжелого технологического оборудования, включая станки с числовым программным управлением (ЧПУ), сверлильные стенды и сопутствующую оснастку.

3) Эргономическая планировка зон: Размещение оборудования производилось в строгом соответствии с принципами бережливого производства (5S) для минимизации временных потерь при перемещении сотрудников между технологическими постами.

4) Контроль техники безопасности: Особое внимание в процессе пусконаладочных и такелажных работ уделялось соблюдению правил охраны труда при манипуляциях с крупногабаритным и тяжелым оборудованием.

Данный этап работы позволил на практике ознакомиться с процессами организации сборочного производства «с нуля», что критически важно для понимания полного жизненного цикла разработки и выпуска электронных систем.

3.6 Автоматизация загрузки аудиофайлов

В рамках индивидуального задания мне было поручено подготовить партию из 40 прошитых и припаянных звуковых модулей (чипов). Задача заключалась в переносе 40 различных аудиофайлов на каждую из 40 плат. Подключение плат к рабочей станции осуществлялось посредством интерфейса USB-C в соответствии с рисунком 3.4.

Выполнение данной операции вручную через стандартный файловый менеджер ОС занимало недопустимо много времени и несло высокий риск пропуска файлов. Для решения этой проблемы я разработал автоматизирующий скрипт на языке Python (с использованием библиотек os и shutil). Скрипт автоматически определял подключение новой платы по USB-C, форматировал каталог и последовательно загружал нужный пакет из 40 аудиофайлов, выводя статус готовности на экран .

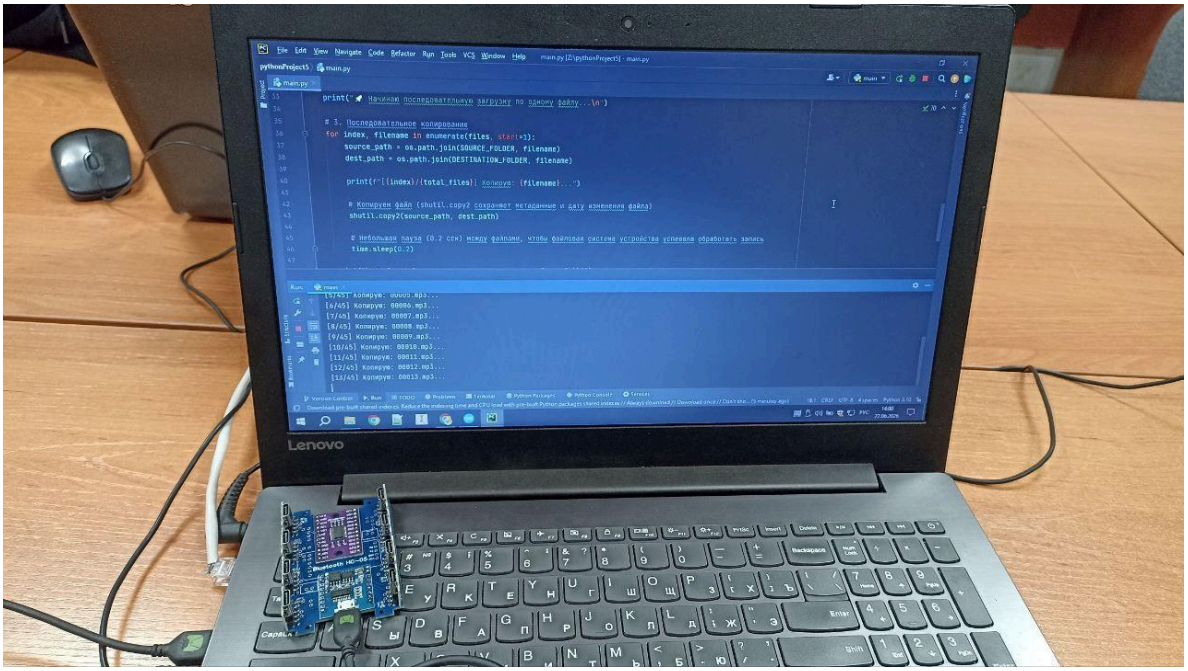


Рисунок 3.4 - Процесс использования Python-скрипта для автоматического копирования аудиофайлов

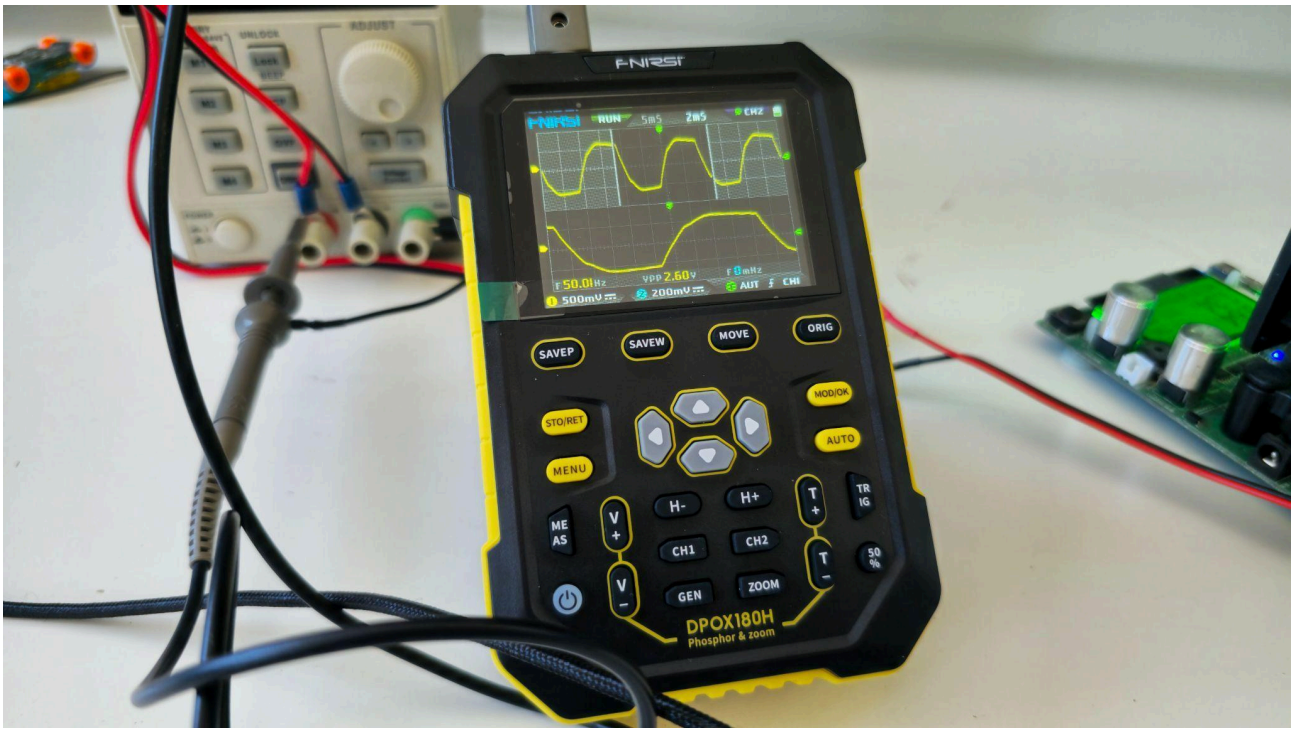
Пример используемого рабочего Python-скрипта для автоматизации копирования аудиофайлов:

```
import os
import shutil
import time

desktop = os.path.join(os.path.expanduser("~"), "Desktop")
src = os.path.join(desktop, "музыкальные файлы")
dst = os.path.join(desktop, "чип ардуино")

os.makedirs(dst, exist_ok=True)

files = [f for f in os.listdir(src) if os.path.isfile(os.path.join(src, f))]

for filename in files:
    src_path = os.path.join(src, filename)
    dst_path = os.path.join(dst, filename)
    shutil.move(src_path, dst_path)
    print(f"Перенесён: {filename}")
    time.sleep(0.5)
```

3.7 Верификация тестового сигнала с помощью осциллографа

Вторым этапом моего индивидуального задания стала диагностика цифровых портов плат Arduino Nano. Был написан код с использованием библиотеки SoftwareSerial, который циклично отправлял осмысленное текстовое сообщение «Test » со всех цифровых пинов контроллера (аналоговые пины не тестировались).

Прежде чем приступать к массовой проверке плат через монитор порта, необходимо было аппаратно убедиться, что написанный код действительно генерирует правильный сигнал на выводах контроллера. Для этого я подключил тестовую плату к осциллографу. На экране прибора была зафиксирована четкая осциллограмма в соответствии с рисунком 3.5, соответствующая логическим уровням передаваемого сообщения «Test ». Это подтвердило корректность работы программной эмуляции UART.



Источник: [7]

Рисунок 3.5 - Осциллограф на котором проводилась проверка

3.8 Массовое тестирование цифровых портов Arduino Nano

После успешной аппаратной верификации осциллографом я приступил к массовому тестированию портов. Для проверки использовался USB-UART конвертер («флешка»), подключенный к ПК .

Аппаратное подключение для диагностики производилось по следующей схеме:

1) Вывод GND конвертера соединялся с выводом GND Arduino.

2) Вывод RX конвертера соединялся проводом типа «папа-мама», «мама-мама» с проверяемым цифровым пином Arduino (выступающим в роли программного TX) в соответствии с рисунком 3.6.

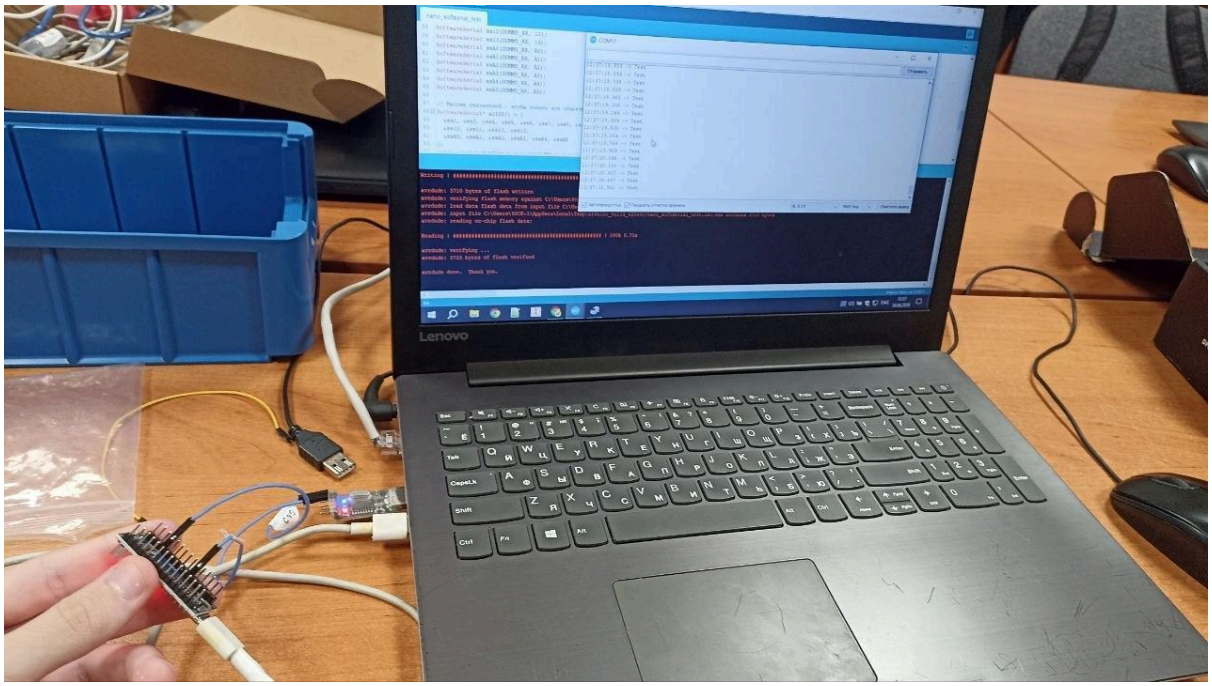


Рисунок 3.6 - Проверка пинов Arduino Nano с монитора порта

Ардуино была подключена к ПК по USB для питания. Я вручную поочередно перевставлял контакт провода (перемещая его по всем цифровым пинам платы). Параллельно на экране ПК была открыта консоль (монитор порта), настроенная на COM-порт UART-конвертера. При подключении к полностью исправному порту Arduino в консоли непрерывно отображалось принимаемое сообщение «Test ». Данная методика позволила быстро и со 100% точностью отбраковать модули с выгоревшими портами.

4 Результаты прохождения практики

4.1 Общие итоги

В ходе прохождения практики мной было подготовлено 220 микроконтроллерных чипов для экосистемы DreamBot (включая звуковые mp3-модули). Для этого объема я выполнил полный цикл операций: от аппаратной прошивки через программатор до финальной верификации. Работоспособность прошитых чипов проверялась лично мной с помощью специализированного диагностического стенда (геймпада с экраном), на консоль которого выводились телеметрические данные тестируемого модуля.

Отдельным блоком моей индивидуальной работы стала загрузка диагностического кода и ручная проверка цифровых портов ввода-вывода на платах Arduino Nano, а также написание автоматизирующего скрипта на языке Python для пакетной обработки звуковых модулей. За месяц работы я значительно повысил навыки аппаратной диагностики телекоммуникационных платформ и автоматизации рутинных задач.

4.2 Сроки выполнения этапов

1) 1 неделя: Вводный инструктаж, глубокое изучение документации предприятия; организация рабочего места.
2) 2 неделя: Прошивка компонентов (220 штук) в среде Arduino IDE и их аппаратный контроль, прошивка плат Arduino.
3) 3 неделя: написание Python-скрипта для mp3, проверка пинов Arduino Nano с помощью Software serial , проверка плат, развертывание производственной инфраструктуры.
4) 4 неделя: Подведение итогов, оформление и систематизация данного отчета.

Заключение

В ходе прохождения учебной технологической (проектно-технологической) практики на предприятии ООО «Центр развития робототехники» были достигнуты цели по закреплению теоретических знаний и получению практического опыта решения профессиональных ИТ-задач. Особое внимание было уделено выполнению индивидуального задания, в рамках которого успешно освоены предусмотренные программой компетенции:

1) В составе рабочей группы я принял участие в подготовке 220 чипов, лично выполнив прошивку и проверку на диагностическом стенде (геймпаде с консолью) включая звуковые модули. Для автоматизации подготовки звуковых модулей был разработан скрипт на языке Python. Внедрение скрипта позволило полностью автоматизировать процесс переноса 40 аудиофайлов на каждую плату через интерфейс USB-C. Это исключило рутинные действия в интерфейсе ОС, многократно сократило время предпродажной подготовки партии и продемонстрировало уверенное владение современными программными технологиями в профессиональной деятельности.

2) В рамках освоения компетенции ПКВ-1 (диагностика и мониторинг оборудования): Был разработан и успешно применен программно-аппаратный алгоритм диагностики цифровых портов микроконтроллеров Arduino Nano. С помощью библиотеки SoftwareSerial была реализована эмуляция UART-передатчика. Получен практический опыт работы с контрольно-измерительным оборудованием: первичная верификация сгенерированного сигнала успешно проведена с использованием осциллографа. Дальнейшее массовое тестирование осуществлялось путем ручной коммутации пинов (GND к GND, программный TX к RX) с использованием USB-UART конвертера и визуальным контролем принимаемых тестовых пакетов («Test ») в мониторе порта.

3) Приобретен практический опыт в области промышленной логистики и развертывания производственной инфраструктуры. В ходе участия в открытии нового технологического цеха ООО «Центр Развития Робототехники» были успешно решены задачи по транспортировке, позиционированию и подготовке к пусконаладке сложного оборудования (включая станки с ЧПУ) с соблюдением стандартов охраны труда и промышленной безопасности.

Все задачи, поставленные руководителями в начале практики, выполнены в полном объеме, а полученные практические компетенции полностью отвечают профилю подготовки по направлению 11.03.02 «Инфокоммуникационные технологии и системы связи».

Список использованных источников

1) ООО «Центр развития робототехники»: реквизиты организации [Электронный ресурс] // Rusprofile. – Режим доступа: https://www.rusprofile.ru (дата обращения: 03.07.2026).

2) Центр развития робототехники: официальный сайт [Электронный ресурс]. – Режим доступа: https://robocenter.org (дата обращения: 03.07.2026).

3) DreamBot: официальный сайт [Электронный ресурс]. – Режим доступа: https://dreambot.org (дата обращения: 02.07.2026).

4) DreamBot User Guide: руководство пользователя [Электронный ресурс]. – Режим доступа: https://dreambot.org/guides/ (дата обращения: 02.07.2026).

5) Указания по подготовке плат для DreamBot : внутренний методический документ / ООО «Центр развития робототехники». – Владивосток : ООО «ЦРР», 2026. – 18 с.

6) Указания по прошивке чипов : внутренний методический документ / ООО «Центр развития робототехники». – Владивосток : ООО «ЦРР», 2026. – 15 с.

7) Обзор портативного осциллографа Fnirsi DPOX180H. Функциональные возможности и результаты тестирования [Электронный ресурс] // https://www.ixbt.com/live/instruments/obzor-portativnogo-oscillografa-fnirsi-dpox180h-funkcionalnye-vozmozhnosti-i-rezultaty-testirovaniya.html

8) DreamBot QuickStart / CLI: автоматизация запуска [Электронный ресурс]. – Режим доступа: https://www.dreambot.org/guides/user-guide/quickstart/ (дата обращения: 02.07.2026).

9) СК-СТО-ТР-04-1.005-2015. Требования к оформлению текстовой части выпускных квалификационных работ, курсовых работ (проектов), рефератов, контрольных работ, отчетов по практикам, лабораторным работам : методический документ. – Владивосток : Изд-во ВГУЭС, 2015. – 46 с.