Министерство науки и высшего образования РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____О.В. Непомнящий
«___»_____2023 г.

БАКАЛАВРСКАЯ РАБОТА

090301 Информатика и вычислительная техника

Учебно-образовательный макет «Умная деревня»

Руководитель	_____	_____	старший преподаватель	Л.В. Макуха
	подпись	*дата*	*должность, ученая степень*	
Выпускник	_____	_____		А.П. Халаманов
	подпись	*дата*		
Нормоконтролёр	_____	_____	старший преподаватель	Л.В. Макуха
	подпись	*дата*	*должность, ученая степень*	

Красноярск 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Поступление в институт на направления, связанные с программированием, представляет собой захватывающую возможность для многих абитуриентов. Однако, на пути к освоению этой сферы, они сталкиваются с несколькими значимыми проблемами.

Программирование — это обширная область с разнообразием специализаций и направлений. Студенты, желающие поступить на программные курсы, имеют возможность выбора между различными областями, такими как веб-разработка, мобильная разработка, анализ данных, искусственный интеллект, кибербезопасность и другие. Они должны проявлять гибкость и адаптивность, чтобы выбрать направление, соответствующее их интересам и целям в карьере.

В настоящее время интерес к изучению низкоуровневого программирования растет с каждым днем. Эта область представляет собой важный компонент для тех, кто стремится понять основы функционирования электроники на более глубоком уровне. Изучение низкоуровневого программирования позволяет студентам узнать, как работает аппаратное обеспечение, основные принципы программируемой составляющей, а также как это взаимодействует между собой.

Учебно-образовательный макет умной деревни, разработанный с использованием микроконтроллера Arduino, предоставляет возможность абитуриентам ознакомиться с новыми технологиями, позволяя им погрузиться в мир низкоуровневого программирования и электроники. Данный проект способствует не только заинтересованности будущего студента данной областью программирования, но также помогает понять, что именно он хочет изучать в будущем, а также способствует в выборе направления обучения.

1 Анализ предметной области

Учебные стенды и макеты широко используются образовательными учреждениями, такими как школы, университеты и профессиональные образовательные программы, чтобы облегчить процесс обучения и понимание учебного материала.

Они позволяют визуализировать сложные или абстрактные концепции. Например, трехмерные модели молекул в химии или модели географических формаций в геологии могут сделать абстрактные концепции более доступными и понятными для обучающихся.

Использование стендов и макетов способствует созданию интерактивной обучающей среды. Ученики могут активно участвовать в изучении материала, визуально осматривая и взаимодействуя с объектами на занятии. Это способствует более глубокому усвоению информации.

Представление учебного материала в форме визуальных и тактильных объектов может привлечь внимание учащихся и усилить их интерес к предмету. Макеты и стенды могут быть использованы для обучения практическим навыкам. Например, студенты медицинских школ могут использовать модели органов для изучения анатомии и процедур, а студенты архитектурных программ могут создавать макеты зданий, развивая свои проектные навыки.

В целом, учебные стенды и макеты эффективно дополняют традиционные методы обучения, делая процесс более интересным, доступным и понятным. Использование таких возможностей в обучении, способствуют активной и визуальной образовательной деятельности, что в итоге может привести к более успешному усвоению учебного материала.

1.1 Анализ существующих продуктов

Для создания учебно-образовательного макета "Умная деревня" важно изучить разнообразие существующих образовательных моделей, цель которых состоит в демонстрации принципов работы умных технологий, а также стимулировании взаимодействия студентов с проектом.

На данный момент существует множество учебных макетов, обеспечивающих интерактивное обучение в области умных систем и IoT-приложений. Некоторые из них могут быть направлены на изучение работы сенсоров и исполнительных устройств, включая различные кнопки, датчики движения, а также устройства управления светом и звуком. Другие макеты могут включать в себя программное обеспечение для взаимодействия через голосовые команды или создания интерактивных сценариев.

Проведение анализа существующих учебно-образовательных макетов позволит определить наилучшие методы и подходы к созданию "Умной деревни". Это также позволит выявить преимущества, недостатки и уникальные особенности различных макетов, что способствует выбору наиболее подходящих концепций и методов для обеспечения эффективного и увлекательного образовательного опыта студентов.

1.1.1 LEGO Education

Одним из ведущих производителей образовательных конструкторов является LEGO Education [1]. Эти наборы позволяют ученикам не только собирать модели, но и программировать и управлять роботами, что делает их полезным инструментом для обучения в области STEM.

Применение наборов LEGO Education в рамках учебно-образовательного проекта позволяет создавать интерактивные модели, демонстрирующие принципы автоматизации. Например, школьники могут использовать компоненты LEGO, такие как различные датчики и моторы, чтобы разработать модель,

реагирующую на изменения освещенности или движения. На рисунке 1 продемонстрирован наиболее популярный набор из серии LEGO Education.

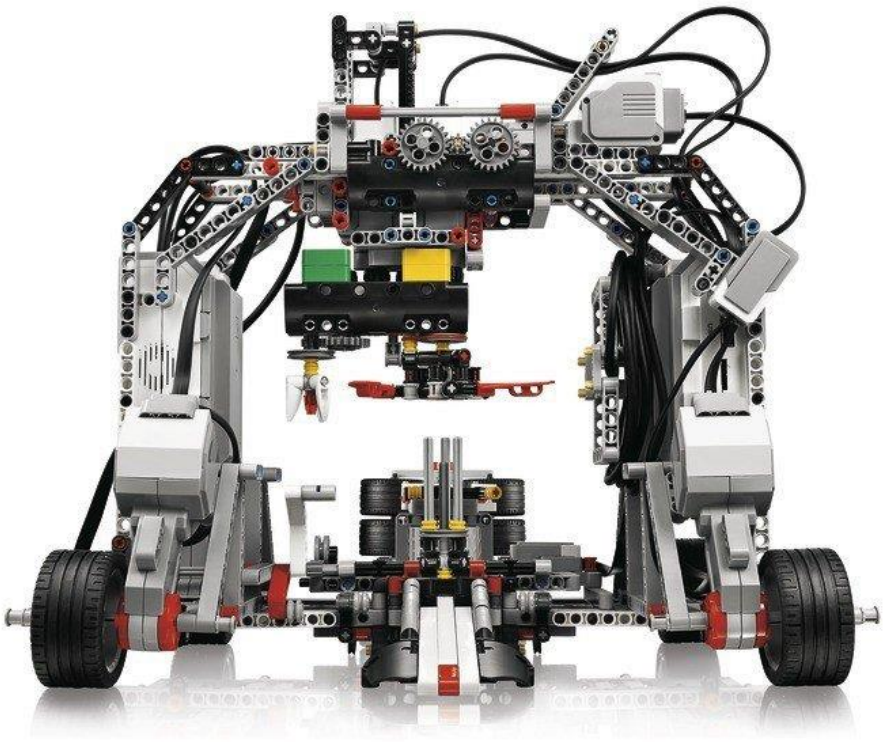

Рисунок 1 – LEGO Education Mindstorms EV3

Однако основным недостатком данного продукта является ограничение гибкости в программировании и управлении моделями. Несмотря на мощные инструменты программирования, предоставленные компанией LEGO Education, возможности для создания сложных сценариев управления могут быть ограничены в пределах предоставленных программных возможностей.

Так же к недостаткам можно отнести цену данного продукта. Наименее дорогие варианты данных наборов с функционалом программирования стартуют от порядка тридцати тысяч рублей. Это может создать значительное ограничение доступности продукта для определенных учебных заведений или отдельных учащихся, ограничивая их возможности в изучении и использовании этих образовательных ресурсов.

1.1.2 Типовой комплект учебного оборудования

Когда необходимо глубже изучить микроконтроллеры, студентам приходит на помощь использование учебных лабораторных стендов, представленный в примере «Микроконтроллер Миландр» [2] (Рисунок 2).

Данный стенд разработан для проведения лабораторно-практических работ в высших, средних и профессионально-технических учебных заведениях с целью освоения знаний, приобретения опыта и получения практических навыков работы с микроконтроллерами и различными периферийными устройствами. В процессе выполнения лабораторно-практических работ студенты знакомятся с функциональностью и возможностями микропроцессорных систем.

Используя встроенные в модуль устройства ввода-вывода, студенты изучают взаимодействие микроконтроллера с данной периферией, экспериментируя с различными интерфейсами и протоколами взаимодействия. Лабораторные работы, выполняемые с использованием этих стендов, могут охватывать такие дисциплины, как «Микропроцессорные системы», «Встраиваемые системы» и «Архитектура ЭВМ».

В процессе выполнения лабораторных работ студенты пишут программы для микроконтроллеров, что способствует их углубленному пониманию программирования на языках C, C++ и Assembler. Это позволяет студентам расширить и усовершенствовать свои навыки в программировании и ознакомиться с различными аспектами работы с микроконтроллерами.

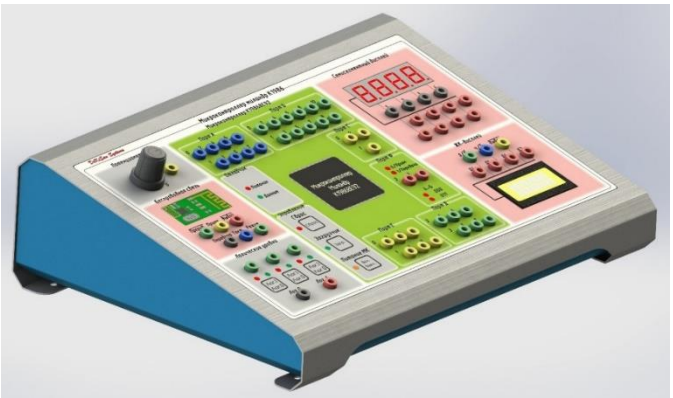

Рисунок 2 – Модуль «Микроконтроллер миландр К1986ВЕ92»

Учебные лабораторные стенды, несмотря на широкий спектр возможностей, обладают ограниченным функционалом, что делает их использование менее гибким и эффективным для образовательных целей.

Основными недостатками таких стендов являются их высокая стоимость и ограничение на возможность замены или смены микроконтроллера, который управляет всей системой. Высокая цена стендов может создать препятствие для их массового использования в образовательных учреждениях, ограничивая доступ студентов к практическим занятиям с использованием этих учебных средств.

Кроме того, невозможность замены микроконтроллера, который управляет функционированием системы, ограничивает возможности адаптации стендов под различные учебные цели или новые технологические требования. Это может снижать гибкость и актуальность учебного процесса, поскольку стенды могут быстро устареть или стать несовместимыми с новыми образовательными задачами из-за ограничений в возможностях замены компонентов.

1.2 Формулировка требований к продукту

Учебные макеты представляют особую категорию моделей, применяемых в образовательном процессе и за его пределами. Суть учебного макета заключается в том, чтобы пользователь, взаимодействуя с ним, мог отточить определенные навыки или осознать принципы его функционирования. Кроме того, некоторые учебные макеты позволяют симулировать реакции в различных сценариях. Данные модели помогают рассмотреть все детали работы объекта, уделяя особое внимание как его внешней, так и внутренней структуре.

Учебные макеты могут быть разделены на 3 вида:

– стендовые модели используются для пояснения работы определенного оборудования и визуального представления информации на выставках, в музеях и других подобных местах. Они служат для наглядной демонстрации принципов функционирования различных устройств;

– модели-тренажеры представляют собой макеты, с которыми обучающийся напрямую взаимодействует. Это могут быть модели стрелкового оружия, которые можно разобрать и собрать заново, или, например, парикмахерские манекены, на которых можно оттачивать навыки стрижки. Данные тренажеры являются важным инструментом для отработки навыков в различных областях профессиональной деятельности;

– натурные образцы представляют собой реалистичные модели объектов, представленные в разрезе. Они размещаются на специальных подставках и снабжаются пояснительными надписями, что обеспечивает возможность изучения основных характеристик предмета и его устройства.

На рисунке 3 приведен пример макета натурного образца.

Рисунок 3 – Макет натурного образца

1.3 Функциональные возможности

Проанализировав имеющуюся продукцию, можно заключить, что проект будет моделью-тренажёром. Наглядный стенд для абитуриентов, с набором исполнительных механизмов с различными способами управления, включая голосовое управление.

Для создания удобной и заинтересовывающей системы управления макетом было решено оснастить макет следующими функциональными возможностями:

– включение/выключение света в домах с помощью управляющих кнопок;

– включение/выключение света в домах с помощью голосовых команд;

– настройка цветовой гаммы света в домах;

– включение/выключение мелодии с помощью голосовых команд;

– включение/выключение света на улице с помощью голосовых команд;

– включение определённых компонентов при приближении к макету.

Так же стоит добавить автоматическое включение/выключение освещения макета в соответствии с временем включения/выключения уличных

фонарей. Освещение и аудиосистема должны работать в доме независимо друг от друга. Система должна работать автономно.

1.4 Цель создания системы

Основной целью данного проекта является демонстрация возможностей низкоуровневого программирования перед абитуриентами с целью пробуждения интереса к этой отрасли. Также проект имеет развлекательный характер и предназначен для широкой аудитории.

1.5 Вывод по первой главе

В данной главе были сформулированы цель и задачи, определяющие основные направления выпускной квалификационной работы. Проанализированные аналоги позволили получить понимание о концепции и требуемом функционале для создаваемого макета умной деревни. Было выявлено, что рассмотренные аналоги не идеально подходят для данной задачи. Они не гибки в использовании и нелегко доступны в виду своей цены.

2 Проектирование системы

2.1 Разработка структурной схемы

Структурная схема микроконтроллерной системы состоит из трёх блоков данных: блок датчиков, блок ввода/вывода данных, обработки и управления, а так же блок исполнительных устройств. На рисунке 4 приведена данная схема и взаимосвязь блоков.

Рисунок 4 – Структурная схема системы

Блок датчиков выполняет важную функцию, считывая и накапливая необходимые данные для точной работы системы. Этот блок использует данные собранные с устройства распознавания голоса, с датчика препятствий, управляющей кнопки и потенциометров в качестве входных данных для подачи нужного сигнала следующему блоку.

Блок ввода/вывода, обработки и управления играет ключевую роль в реализации алгоритмов, опираясь на информацию, поступающую из блока датчиков и мировое время. Он содержит микроконтроллерную плату, который обрабатывает данные и генерирует сигнал для следующего блока.

Блок исполнительных устройств получает команды от блока ввода/вывода, обработки и управления, с помощью которого будет выполнена поставленная пользователем задача. Блок исполнительных устройств будет содержать в себе микропривод, светодиоды и устройство вывода звука.

2.2 Разработка функциональной схемы

Согласно поставленным целям в первой главе, а так же составленной структурной схеме, была основана функциональная схема микроконтроллерной системы «Умная деревня». Данная схема изображена на рисунке 5.

Рисунок 5 – Функциональная схема системы

На основе построенной функциональной схемы, было выявлено, что использоваться будет большое количество модулей одновременно. Поэтому был составлен список компонентов, позволяющий достичь поставленной цели в условиях подключения к одному микроконтроллеру.

2.3 Элементная база

Выбор компонентов будет непосредственно влиять на работоспособность конечного макета и повлияет на восприятие студентами возможностей программирования. Выбор компонентов производился по нескольким критериям:

– доступность на рынке;

– ценовой сегмент;

– возможность использования в системе с минимизацией кода и количества используемых портов;

– наличие в институте нужных компонентов.

По данным критериям были выбраны следующие модули [7].

2.3.1 Управляющая кнопка

В данной работе, нужна всего одна управляющие кнопки. Она отвечает за маленькие домики, требуются, если ученик захочет включить свет в домике не голосовой командой. Кнопка займёт один цифровой вход на микроконтроллере. Для этого нам понадобится самый простой тактовый выводной переключатель. Пример такой кнопки изображён на рисунке 6.

Рисунок 6 – Тактовая кнопка

2.3.2 ИК датчик препятствия

Данный модуль нам нужен для того, чтобы производилось включение электропривода при приближении ученика к стенду. В этом случае нам подходит два похожих между собой модуля. Ультразвуковой дальномер и инфракрасный датчик препятствия. В силу экономии количества используемых портов и упрощения программной реализации, выбор сделан в сторону ИК датчиков препятствия, а конкретно модель YL-63 изображённая на рисунке 7, данные модули в сумме займут 3 цифровых входа на микроконтроллере.

Рисунок 7 – YL – 63

Характеристики данного модуля приведены в таблице 1.

Таблица 1 – Характеристики YL – 63

Наименование показателя	**Значения**
Модель	YL-63(или FC-51)
Напряжение питания, В	3.3 – 5
Тип датчика	диффузионный
Расстояние обнаружения препятствий, см	2 – 30
Эффективный угол обнаружения препятствий, °	35
Габариты, мм	43×16×7

2.3.3 Потенциометр

Для разнообразия функционала макета, было решено использовать потенциометры, с помощью которых будут меняться цвета подключенных RGB светодиодов. Выбор пал на модель GSMIN WH148 B1K изображённую на рисунке 8. Оба этих модуля потребуют 2 аналоговых входа на микроконтроллере.

Рисунок 8 – GSMIN WH148 B1K

Характеристики данного модуля приведены в таблице 2.

Таблица 2 – Характеристики GSMIN WH148 B1K

Наименование показателя	**Значения**
Модель	GSMIN WH148 B1K
Вес, г	5
Мощность, Вт	0.5
Сопротивление, кОм	1
Тип °	Линейный
Особенности	Вращающийся вал
Габариты, мм	16×14×14

2.3.4 Модуль распознавания голоса

Самой важной частью данного проекта является реализация голосового управления светом и мелодией. Написание собственного кода для осуществления задуманного, трудоёмкий процесс, требующий огромного количества различных модулей, библиотек и знаний. Однако есть решение, которое позволит избежать неудачи в реализации, сэкономит время и благодаря своему обширному функционалу, не только подходит для поставленной задачи, но позволит добавить различный функционал в будущем. Более того, этот модуль занимает всего 2 цифровых входа на микроконтроллере. Данным решением является модуль распознавания голоса Elechouse Voice Recognition Module v3.1 изображённый на рисунке 9.

Рисунок 9 – Elechouse v3.1

Характеристики данного модуля приведены в таблице 3.

Таблица 3 – Характеристики Elechouse v3.1

Наименование показателя	**Значения**
Модель	elechouse v3.1
Напряжение питания, В	4.5 – 5.5
Потребляемый ток, мА	менее 40
Флеш-память, МБайт	2
Банк хранения записей	до 80 команд продолжительностью 1.5 сек (одно или два слова)
Банк активных команд	макс. 7 в памяти распознавателя

Окончание таблицы 3

Наименование показателя	Значения
Цифровой интерфейс	UART / GPIO
Аналоговый интерфейс	3.5мм моно – канальный микрофонный разъём
Размеры, мм	47×30×7
Вес, г	18 (5 без микрофона)

2.3.5 Светодиоды

Освещение в макете будет осуществлено за счёт 5 различных светодиодов. Два RGB светодиода потребуются для двух домиков, поскольку сами светодиоды не будет видно глазом, проще всего взять модуль RGB светодиода для этой задачи, которые вместе займут 6 контактов сдвигового регистра. Пример такого изображён на рисунке 10.

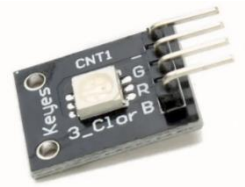

Рисунок 10 – Модуль RGB светодиода

Оставшиеся 3 светодиода будут одинаковые и белого цвета. Два из них нужны для оставшихся домиков, они будут включаться при помощи управляющих кнопок, а оставшийся будет имитировать уличный фонарь. Все вместе они займут 1 цифровой выход на плате и 2 контакта на сдвиговом регистре Пример таких светодиодов представлен на рисунке 11.

Рисунок 11 – Белый светодиод

2.3.6 Сдвиговый регистр

В проекте используется большое количество различных модулей, а особенно светодиодов. При выборе комплектующих была поставлена цель, использовать наименьшее количество портов микроконтроллера, во избежание выбора более мощного микроконтроллера или докупки модуля расширения. Было решено использовать один сдвиговый регистр, в который будут подключены два белых светодиода отвечающих за освещение по управляющим кнопкам и два RGB светодиода, что позволит нам в разы сэкономить количество используемых портов микроконтроллера. Для данной цели достаточно будет использовать сдвиговый регистр 74hc595 представленный на рисунке 12.

Рисунок 12 – 74hc595

Характеристики данного модуля представлены в таблице 4.

Таблица 4 – Характеристики 74hc595

Наименование показателя	**Значения**
Модель	74hc595
Интерфейс	SPI
Напряжение питания, В	2 – 6
Макс. выходной ток выводов, мА	7.8
Диапазон рабочих температур, °С	-40...+85

2.3.7 Пьезоэлектрический динамик

Для воспроизведения простой мелодии достаточно использовать модуль зуммера, который занимает 1 цифровой вход на микроконтроллере. Зуммер изображён на рисунке 13.

Рисунок 13 – Модуль пьезоэлектрического динамика

Характеристики данного модуля представлены в таблице 5.

Таблица 5 – Характеристики зуммера

Наименование показателя	**Значения**
Напряжение питания, В	5
Потребляемый ток, мА	до 30
Интенсивность звука, дБ	>=85
Резонансная частота, Гц	2048
Сопротивление обмотки, Ом	40
Рабочая температура, °С	-20 ... 70
Габариты, мм	30×30×9

2.3.8 Микропривод

В данном макете микропривод потребуется для демонстрации голосовых команд управления деревней. К нему будет прикреплена складная таблица с расписанными на ней командами, которая будет открываться из-за сработавших датчиков препятствия. Для данной цели, нам потребуется привод с осью вращения более 90 градусов. Под все требования подходит Микропривод постоянного вращения Tower Pro SG90, представленный на рисунке 14, занимающий всего 1 цифровой вход на плате.

Рисунок 14 – Tower Pro SG90

Характеристики данного модуля представлены в таблице 6.

Таблица 6 – Характеристики Tower Pro SG90

Наименование показателя	**Значения**
Модель	tower Pro SG90
Тип привода	постоянное вращения
Диапазон поворота вала, °	360
Макс. скорость вращения вала, об/мин	120
Крутящий момент, кг×см	1.3
Напряжение питания, В	4.8 – 6
Потребляемый ток, мА	100
Материал шестерней	нейлон
Материал корпуса	пластик
Длина кабеля, см	25

2.3.9 RTC модуль

Для реализации отключения освещения согласно уличным фонарям, нам потребуется модуль часов реального времени. Однако из-за обширного количества компонентов, уже задействовано 13 цифровых порта на микроконтроллере, поэтому решено было брать такой модуль, который будет задействовать аналоговые порты на плате. Более того, требуется модуль, который будет способен отсчитывать время, даже в условиях, когда система находится без питания. В данном случае нам понадобится RTC модуль GSMIN DS3231 представленный на рисунке 15, он будет занимать 2 аналоговых порта микроконтроллера.

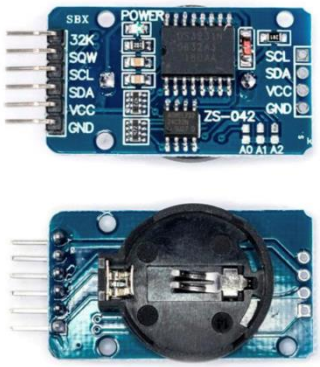

Рисунок 15 – GSMIN DS3231

Характеристики данного модуля представлены в таблице 7.

Таблица 7 – Характеристики RTC модуля

Наименование показателя	**Значения**
Календарь	До 2100 года
Погрешность, ppm	+/– 4
Габариты, мм	40×22

2.3.10 Микроконтроллерная плата

Прежде всего, требуется, чтобы микроконтроллерная плата имела компактные размеры, поскольку предполагается, что она будет установлена в макете таким образом, чтобы оставаться незаметной для студентов. Так же, требуется чтобы количество цифровых и аналоговых портов платы, было достаточно для подключения всех модулей. Подсчитав количество используемых модулями портов, сделан вывод что нам потребуется микроконтроллер, который будет иметь не меньше 13 цифровых входов и не меньше 4 аналоговых.

В связи с тем, что создание данного макета не включает использование беспроводных интерфейсов, выбор пал на плату Arduino Uno. Она имеет 14 цифровых входов и 6 аналоговых, что соответствует нашему требованию

Этот выбор обоснован доступностью и ценой данной платформы. Arduino Uno широко распространена и имеет умеренную стоимость, обладая при этом широким набором функциональных возможностей благодаря разнообразным интерфейсам и встроенным датчикам. Кроме того, данная плата поддерживает множество языков программирования и программных сред разработки, что делает её чрезвычайно гибкой в процессе программирования. На рисунке 16 продемонстрирована микроконтроллерная плата Arduino Uno.

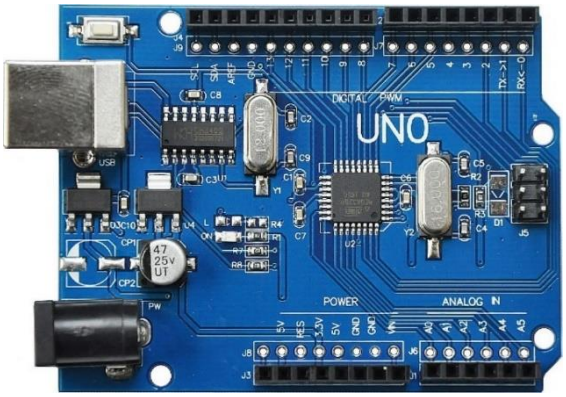

Рисунок 16 – Arduino Uno

В таблице 8 представлены основные характеристики микроконтроллера Arduino Uno[8].

Таблица 8 – Основные характеристики Arduino Uno

Наименование показателя	Значения
Микроконтроллер	ATmega 328
Разрядность, бит	8
Архитектура	AVR
Тактовая частота, МГц	16
ПЗУ, кб	32
ОЗУ, кб	2
EEPROM, кб	1
Габариты, мм	68.6×53.4
Рабочее напряжение, в	5
Напряжение питания, в	5
Цена, руб.	890

2.3.11 Модуль питания и макетная плата

Для достижения автономности работы и избежания перегрузки Arduino UNO, потребуется специальный модуль питания, а для удобства сборки системы нужна макетная плата. Модуль питания для макетной платы MB102, изображённый на рисунке 17, представляет собой компактное устройство, которое обеспечивает стабильное питание для электронных проектов. Он оснащен высококачественными компонентами и может работать с напряжением 3,3 В и 5 В, что делает его идеальным для широкого спектра проектов.

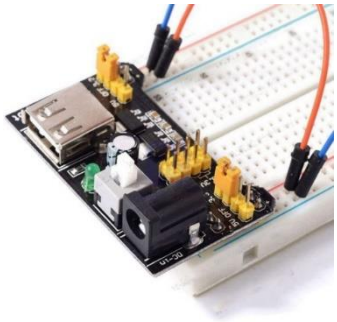

Рисунок 17 – MB102 с макетной платой

2.4 Принципиальная схема

Согласно выбранным компонентам требуемых для учебно-образовательного макета «Умная деревня» и структурной схеме, была составлена принципиальная схема системы. Электрическая схема изображена на рисунке 18.

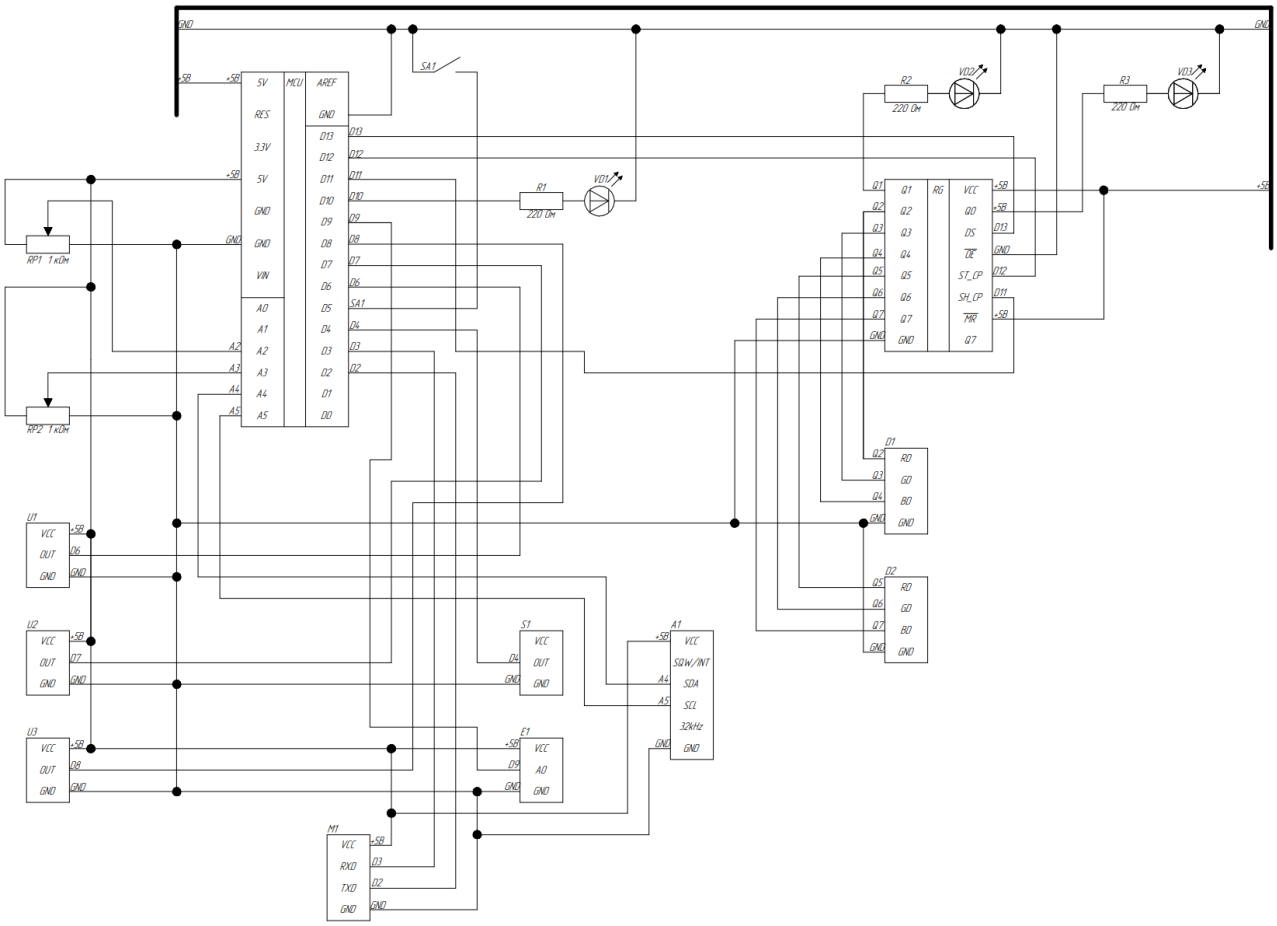

Рисунок 18 – Электрическая схема системы

Основываясь на разработанную принципиальную схему, которая демонстрирует всю взаимосвязь компонентов, можно приступать к написанию программного кода с последующей сборкой макета.

2.5 Выводы по второй главе

В ходе проектирования учебно-образовательного макета «Умная деревня», были разработаны функциональная и структурная схемы системы. Произведена выборка необходимых компонентов, способных обеспечить поставленный задач и разработана электрическая схема. Данная схема позволяет осуществить сборку и тестирование проекта с использованием реальных компонентов.

Список используемых компонентов представлен в приложении В.

3 Разработка системы

3.1 Словесное представление алгоритма функционирования

Инициализация: С запуском программы выполняется инициализация всех подключенных устройств – RGB модулей, светодиодов, электропривода, управляющих кнопок, модуля распознавания голоса, зуммера и ИК датчиков.

Основной цикл: Основной цикл программы, который выполняется бесконечно, пока система подключена к питанию. В данной части программного кода вызываются все функции, которые позволяют взаимодействовать с макетом.

Опрос датчиков: Каждая функция из основного цикла, опрашивает свой датчик. ИК датчик возвращает логический ноль, если кто-то подошёл к макету на близкое расстояние. В этом случае будет вызвано срабатывание электропривода. В случае если модуль распознавания голоса засёк команду, которая записана в реестр активных команд, будет вызвана соответствующая команде функция. Если была нажата управляющая кнопка, будет вызвана функция для включения светодиодов. Потенциометры обрабатываются только в тех случаях, когда RGB модули включены, в противном случае они игнорируются. Зуммер воспроизводит мелодию по соответствующей голосовой команде.

Экономия ресурса: в соответствии с указанным временем, диод, отвечающий за имитацию уличного освещения, будет включаться и выключаться.

3.2 Разработка алгоритма функционирования

Основываясь на словесное описание системы, составлена блок – схема алгоритма функционирования, представленная на рисунке 19.

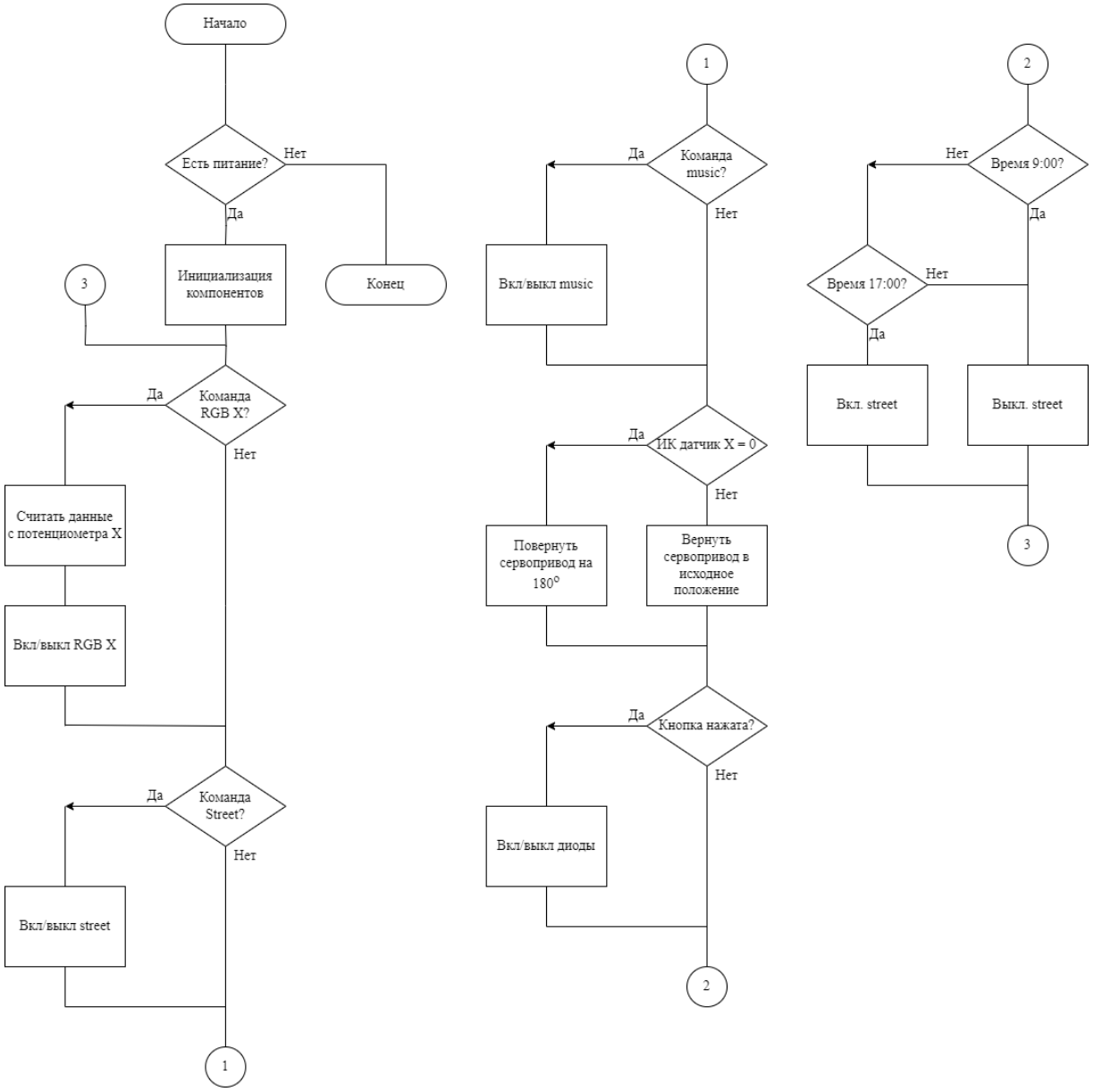

Рисунок 19 – Блок – схема алгоритма разрабатываемой системы

3.3 Моделирование программирования и сборки системы

Для избежания вывода из строя компонентов необходимых для макета, был создан прототип системы и её программный код в среде, позволяющей эмулировать сборку и программирование на микроконтроллерах Arduino. Были реализованы все возможные функции, заданные в первой главе, которые можно было осуществить в эмуляторе. На рисунке 20 представлена модель, которая выполняет такие функции как: включение светодиодов с помощью управляющих кнопок, вращение сервопривода, включение и регулировка цвета RGB модулей.

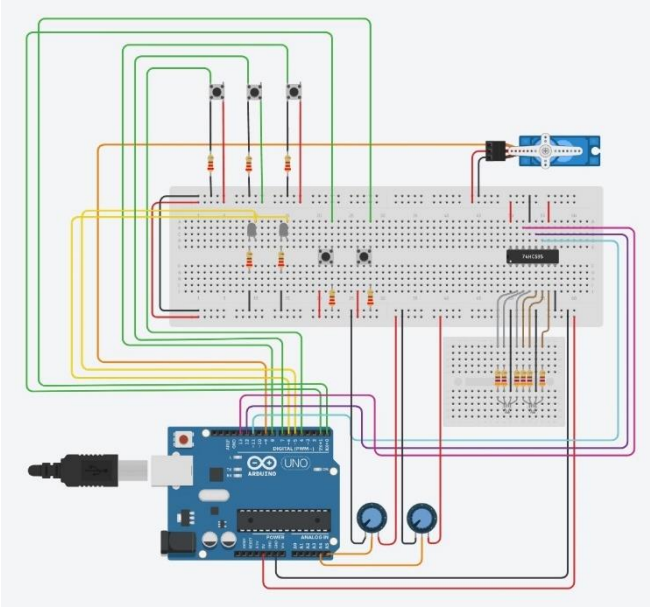

Рисунок 20 – Симуляция системы

3.4 Программный код

После успешной эмуляции требуемой системы, а так же приобретения комплектующих, следующими шагами были поэтапная сборка и программирование макета.

В главном цикле программы реализован опрос голосового модуля и опрос датчиков, не требующих голосовых команд. На рисунке 21 представлена данная часть кода.

```
// MAIN

void loop() {
  int ret;
  ret = myVR.recognize(buf, 50);
  if(ret > 0)
  {
    switch(buf[1])
    {
      case onLeftRecord:                          // Голосовая команда для включения света в 1 доме
        rgbLeft = true;
      break;

      case offLeftRecord:                         // Голосовая команда для выключения света в 1 доме
        rgbLeft = false;
      break;

      case onRightRecord:                         // Голосовая команда для включения света во 2 доме
        rgbRight = true;
      break;

      case offRightRecord:                        // Голосовая команда для выключения света в 2 доме
        rgbRight = false;
      break;

      case musicRecord:                         // Голосовая команда для включения мелодии
        stateMusic = !stateMusic;    // Устанавилваем флаг stateMusic в true для последующего включения мелодии
      break;

      case hideCommandsRecord:                        // Голосовая команда для выключения мелодии
        servoState = false;   // Устанавилваем флаг stateMusic в false для последующего выключения мелодии
      break;

      case streetRecord:                          // Голосовая команда для включения/выключения уличного освещения
        stateLed10 = !stateLed10;
        digitalWrite(ledPin10, stateLed10);       // Включаем/выключаем свет улицы
      break;

    }
  }
  checkCommand();
  checkIr();
  checkButtons();
  checkTime();
  playMusic();
}
```

Рисунок 21 – Главный цикл программы

Для включения RGB модулей, будут задействованы case onLeftRecord и onRightRecord, которые вызываются если пользователь скажет в микрофон голосового модуля “Один” или “Три”. В данных случаях с помощью функции checkPotValue() считываются данные с

потенциометров для выбора цвета. Далее с помощью функций rgbValue1() и rgbValue2() данные с потенциометров преобразуются в соответствующие биты, и полученная битовая последовательность передаётся в сдвиговый регистр. Данная передача реализуется в функции DataSend(). Код для генерации цвета для RGB модулей представлен на рисунках 22 и 23.

```
// функция, которая считывает значения с потенциометров для RGB модулей
void checkPotValue(){
  potValue1 = analogRead(A2);
  potValue2 = analogRead(A3);
}
```

Рисунок 22 – Считывание данных с потенциометров

```
void rgbValue1(){
  if (potValue1 < 200)
  {
    bitWrite(dataToSend, 4, HIGH);
    bitWrite(dataToSend, 5, HIGH);
  }
  if (potValue1 >= 200 && potValue1 <= 300)
  {
    bitWrite(dataToSend, 3, HIGH);
  }
  if (potValue1 > 300 && potValue1 < 500)
  {
    bitWrite(dataToSend, 3, HIGH);
    bitWrite(dataToSend, 5, HIGH);
  }
  if (potValue1 >= 500 && potValue1 <= 600)
  {
    bitWrite(dataToSend, 4, HIGH);
  }
  if (potValue1 > 600 && potValue1 < 800)
  {
    bitWrite(dataToSend, 3, HIGH);
    bitWrite(dataToSend, 4, HIGH);
  }
  if (potValue1 >= 800 && potValue1 <= 900)
  {
    bitWrite(dataToSend, 5, HIGH);
  }
  if (potValue1 > 900)
  {
    bitWrite(dataToSend, 3, HIGH);
    bitWrite(dataToSend, 4, HIGH);
    bitWrite(dataToSend, 5, HIGH);
  }
}
```

```
void rgbValue2(){
  if (potValue2 < 200)
  {
    bitWrite(dataToSend, 1, HIGH);
    bitWrite(dataToSend, 2, HIGH);
  }
  if (potValue2 >= 200 && potValue2 <= 300)
  {
    bitWrite(dataToSend, 0, HIGH);
  }
  if (potValue2 > 300 && potValue2 < 500)
  {
    bitWrite(dataToSend, 0, HIGH);
    bitWrite(dataToSend, 2, HIGH);
  }
  if (potValue2 >= 500 && potValue2 <= 600)
  {
    bitWrite(dataToSend, 1, HIGH);
  }
  if (potValue2 > 600 && potValue2 < 800)
  {
    bitWrite(dataToSend, 0, HIGH);
    bitWrite(dataToSend, 1, HIGH);
  }
  if (potValue2 >= 800 && potValue2 <= 900)
  {
    bitWrite(dataToSend, 2, HIGH);
  }
  if (potValue2 > 900)
  {
    bitWrite(dataToSend, 0, HIGH);
    bitWrite(dataToSend, 1, HIGH);
    bitWrite(dataToSend, 2, HIGH);
  }
}
```

Рисунок 23 – Генерация битовой последовательности

Код для передачи битовой последовательности в сдвиговый регистр представлен на рисунке 24.

```
// Функция отправки данных на РГБ модули и светодиоды маленьких домиков
void DataSend(){
  digitalWrite(latchPin, LOW);                    // Начинаем передачу данных
  shiftOut(dataPin, clockPin, LSBFIRST, dataToSend);
  digitalWrite(latchPin, HIGH);                   // Прекращаем передачу данных
  delay(10);
}
```

Рисунок 24 – Передача данных в сдвиговый регистр

Для выключения RGB модулей будут задействованы case offLeftRecord и offRightRecord, которые вызываются если пользователь скажет в микрофон голосового модуля “Два” или “Четыре”. В этих случаях обнуляются соответствующие биты и передаются в сдвиговый регистр.

Для включения и выключения мелодии будет задействованы case musicRecord, который вызывается если пользователь скажет в микрофон голосового модуля “Пять”. В соответствии с командой устанавливается флаг stateMusic в 1 или 0, указывающий исполняется мелодия в данный момент или нет. Реализация проигрывания мелодии расположена в функции playMusic() после опроса голосового модуля. Это сделано для избежания использования цикла, чтобы мелодию можно было выключить до её завершения, если это потребуется. Код воспроизведения мелодии представлен на рисунке 25.

```
void playMusic(){
  // Воспроизводим музыку если была поданна соответствующая голосовая команда (stateMusic будет true)
  if (stateMusic == true)
  {
    size = sizeof(durations) / sizeof(int);
    if (note < size)
    {
      int duration = 1000 / durations[note];
      tone(buzzer, melody[note], duration);

      // Чтобы различать ноты, установливаем минимальное время между ними
      int pauseBetweenNotes = duration * 1.30;
      delay(pauseBetweenNotes);

      // Останавливаем музыку
      noTone(buzzer);
      note = note + 1;
    }
    else
    {
      note = 0;
    }
  }
  else              // Иначе мелодия играть не будет
  {
    note = 0;
  }
}
```

Рисунок 25 – Воспроизведение мелодии

Для включения и выключения одноцветного светодиода, отвечающего за уличное освещение, будет задействован case streetRecord, который вызывается если пользователь скажет в микрофон голосового модуля “Семь”. В данном случае устанавливается флаг stateLed10 в 1 или 0 и передаётся на 10 цифровой порт Arduino. Данный код представлен на рисунке 26.

```
case streetRecord:                          // Голосовая команда для включения/выключения уличного освещения
  delay(10);            // Пауза для стабилизации
  stateLed10 = !stateLed10;
  digitalWrite(ledPin10, stateLed10);       // Включаем/выключаем свет улицы
  delay(10);            // Пауза для стабилизации
break;
```

Рисунок 26 – Включение и выключение уличного освещения

Если ни одна голосовая команда не была подана, происходит опрос ИК датчиков препятствия, управляющей кнопки и проверяется текущее время.

Опрос ИК датчиков препятствия происходит с помощью функции checkIr(). Код данной функции представлен на рисунке 27.

```
// Функция, которая принимает значение с ИК датчиков
void checkIr(){
  if (barrier1 == 0 || barrier2 == 0 || barrier3 == 0)
  {
    servoState = true;
  }
  if (servoState == true)
  {
    turnServo();
  }
  if (servoState == false)
  {
    returnServo();
  }
}
```

Рисунок 27 – Опрос ИК датчиков препятствия

Если хотя бы один из ИК датчиков препятствия срабатывает, это символизирует о том, что ученик подошёл к макету или провёл рукой. В данном случае произойдёт вызов функции turnServo(), которая отвечает за поворот

сервопривода. Сервопривод своим действием откроет таблицу с голосовыми командами для управления макета, она будет поднята пока ученик не скажет в микрофон голосового модуля команду “Шесть”, которая заставит сервопривод вернуться в исходное положение. Код данных команд представлен на рисунках 28 и 29.

```
// Функция для поворота сервопривода на 180 градусов
void turnServo() {
  servo.write(180);
}
```

Рисунок 28 – Поворот сервопривода

```
// Функция для возврата сервопривода в исходное положение
void returnServo(){
  servo.write(0);
}
```

Рисунок 29 – Исходное положение сервопривода

Опрос управляющей кнопки происходит с помощью функции checkButtons(). В случае нажатия на кнопку, будут отправлены соответствующие биты в сдвиговый регистр и произойдёт включение одноцветных светодиодов, изображающих свет в маленьких домиках. Код данной функции представлен на рисунке 30.

```
// Функция проверки нажатия кнопок для включения диодов в домиках
void checkButtons(){
  if (digitalRead(buttonPin5) == HIGH)
  {
    // Пока кнопку не отпустят, свет не включится, для избежания постоянного включения/выключения света в доме
    while (digitalRead(buttonPin5) == HIGH);
    stateLed5 = !stateLed5;
    bitWrite(dataToSend, 7, stateLed5);
    stateLed4 = !stateLed4;
    bitWrite(dataToSend, 6, stateLed4);
  }
  DataSend();
}
```

Рисунок 30 – Включение света в маленьких домиках

Опрос текущего времени происходит в функции checkTime() с помощью RTC модуля, который способен отслеживать время, даже если питание на систему не подано. Если текущее время будет соответствовать 17:00 (UTC+7), светодиод, отвечающий за уличное освещение, будет включен. Если время будет 9:00 (UTC+7), светодиод будет выключен. Код данной функции представлен на рисунке 31.

```
// Функция, которая смотрит, нужно ли включить/выключить свет улицы, согласно реальному времени
void checkTime(){
  if (rtc.getHours() == 17 && rtc.getMinutes() == 0)        // Если реальное время 5 вечера, включаем свет улицы
  {
    stateLed10 = true;
    digitalWrite(ledPin10, stateLed10);
  }
  if (rtc.getHours() == 9 && rtc.getMinutes() == 0)         // Если реальное время 9 утра, выключаем свет улицы
  {
    stateLed10 = false;
    digitalWrite(ledPin10, stateLed10);
  }
}
```

Рисунок 31 – Уличное освещение согласно реальному времени

Полный код программы для учебно-образовательного макета «Умная деревня» представлен в приложении А.

3.5 Сборка системы

Во время разработки системы, на макетной плате была собрана тестовая модель «умной деревни», для достоверности в правильности подключений компонентов и работоспособности написанного кода. Так же был проведён тест голосовых команд и их распознавания голосовым модулем. Было выявлено, что наилучшими командами, которые распознаются модулем от людей с разным тембром голоса, являются команды “Один”, “Два”, “Три”, “Четыре”, “Пять”, “Шесть”, “Семь”. Однако может быть записана любая команда длинной до двух секунд. Полученный макет представлен на рисунке 32.

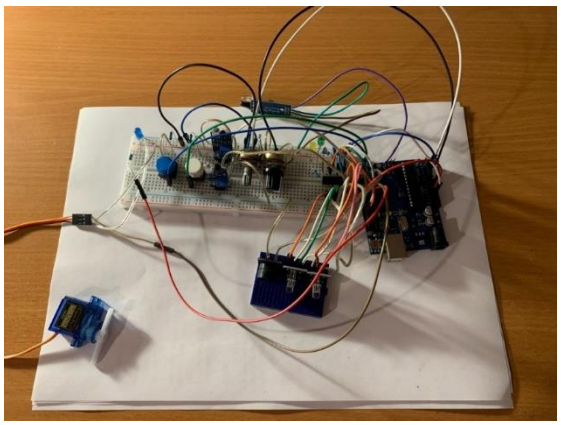

Рисунок 32 – Тестовая модель

После успешного тестирования, все компоненты были размещены в заготовленный стенд, который является результатом изначально задуманного учебно-образовательного макета «Умная деревня». Данный стенд представлен на рисунке 33.

Рисунок 33 – Учебно-образовательный макет «Умная деревня»

3.6 Вывод по третьей главе

В третьей главе был реализован стенд, который соответствует поставленным требованиям. В ходе работы было реализовано подключение всех необходимых компонентов согласно заданному функционалу и принципиальной схеме в единую автономную систему. Был написан программный код и загружен на микроконтроллер. Инструкция по настройке и управлению полученной системы представлена в приложении Б.

ЗАКЛЮЧЕНИЕ

В результате работы был создан учебно-образовательный стенд «умная деревня».

В данной работе был проведён анализ предметной области, что в результате помогло сформировать требования к конечному продукту. Они указывают на преимущества каждого из рассмотренных аналогов, подчёркивая ключевые характеристики, представленные в описаниях. На основе требований был выбран функционал, комплектующие и технологии необходимые для реализации задуманного стенда.

Для реализации проекта были использованы программная среда Arduino IDE, язык программирования C и набор комплектующих для реализации функционала.

В результате разработан макет для вовлечения абитуриентов к изучению программирования, который на простом примере позволяет показать, что может из себя представлять низкоуровневое программирование.

Учебно-образовательный макет предусматривает возможность расширения функционала, а именно подключение большего числа компонентов для увеличения вариантов взаимодействия ученика с макетом, изменение подхода к голосовому управлению или написание собственного алгоритма распознавания голоса.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2001. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления // Консорциум кодекс: электронный фонд правовых и нормативно-технических документов : официальный сайт. – URL: https://docs.cntd.ru/document/1200026224 (дата обращения: 21.12.2023).

2. ГОСТ 7.9-95 (ИСО 214-76). Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования // Научная периодика: проблемы и решения. – URL: https://nppir.ru/wp-content/uploads/22-gost-7.9-95.pdf (дата обращения: 21.12.2023).

3. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и из-дательскому делу. Библиографическая запись. Библиографическое описание. Об-щие требования и правила составления.

4. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления // Российская государственная библиотека: официальный сайт. – URL: https://diss.rsl.ru/datadocs/doc_291wu.pdf (дата обращения: 21.12.2023).

5. История lego, серия lego // LEGO.com : официальный сайт компании: сайт. – URL: https://www.lego.com/ru-ru/history/articles/g-lego-education (дата обращения: 20.12.2023).

6. Типовой комплект учебного оборудования «Микроконтроллер Миландр» // ООО НПП «Учтех-Профи» : сайт. – URL: https://labstand.ru/catalog/mikrokontrollery-i-mikroproczessornaya-tehnika/uchebno-laboratornyj-stend-mikrokontroller-milandr-milandr (дата обращения: 20.12.2023)

7. Интернет-магазин комплектующих для 3D принтеров, ЧПУ станков и робототехники // 3DiY (Тридиай) : сайт. – URL: https://3d-diy.ru/wiki/arduino-moduli/modul-zapisi-golosa-SD1820/?ysclid=lqpj3r8mio759687001 (дата обращения: 20.12.2023).

8. Документация Arduino / программные библиотеки // ARDUINO.CC официальный сайт компании: сайт. – URL: https://docs.arduino.cc/hardware/uno-rev3/ (дата обращения: 21.12.2023).

ПРИЛОЖЕНИЕ А

Программный код системы

Полный код программы представлен в листинге 1.

Листинг 1 – main.ino

```
1  // "Умная деревня" - выпускная работа
2  // Автор: Халаманов Алексей, ст. гр. КИ20-08Б
3  // Данный скетч требует в сборке:
4  // Один голосовой модуль Elechouse V3.1 подключённый к 2 и 3 цифровым
   портам Arduino UNO
5  // Один пьезодинамик подключённый к 4 цифровому порту Arduino UNO
6  // Одна тактовая кнопка подключенная к 5 цифровому порту Arduino UNO
7  // Три ИК датчика препятствия подключенные к 6, 7 и 8 цифровым портам
   Arduino UNO
8  // Один сервопривод подключённый к 9 цифровому порту Arduino UNO
9  // Один одноцветный светодиод подключенный к 10 цифровому контакту Ar-
10 duino UNO
11 // Один сдвиговый регистр 74HC595 подключённый к 11, 12 и 13 цифровым
   портам Arduino UNO
12 // Два одноцветных светодиода подключенные к 0 и 1 контактам сдвигового
13 регистра 74HC595
14 // Два RGB модуля подключенные к 2, 3, 4 и 5, 6, 7 контактам сдвигового
   регистра 74HC595
15 // Два потенциометра подключенные к 2 и 3 аналоговым портам Arduino UNO
16 // Один RTC модуль подключенный к 4 и 5 аналоговым портам Arduino UNO
17 #include <VoiceRecognitionV3.h>
18 #include <pitches.h>
19 #include <Servo.h>
20 #include <microDS3231.h>
21 // Объект для управления голосовым модулем
22 VR myVR(2,3);
23 // Объект для управления RTC
24 MicroDS3231 rtc;
25 // Объект для управления сервоприводом
26 Servo servo;
27 // Пин для зуммера
28 #define buzzer 4
29 // Пины для голосового модуля
30 #define voicePin2 2
31
```

```
#define voicePin3 3
// Пины для ИК датчиков
#define irPin6 6
#define irPin7 7
#define irPin8 8
// Пин для тактовой кнопки (СВЕТ В МАЛЕНЬКИХ ДОМИКАХ)
#define buttonPin5 5
// Пин для сервопривода
#define servoPin 9
// Пин для светодиода уличного освещения
#define ledPin10 10
// Пины для сдвигового регистра
#define latchPin 12 // ST_CP пин (12 в HC595)
#define clockPin 11 // SH_CP пин (11 в HC595)
#define dataPin 13  // DS пин (14 в HC595)
// Глобальные переменные
uint8_t records[7]; // Записанные голосовые команды
uint8_t buf[64];
// Голосовые команды
#define onLeftRecord (0)            // Голосовая команда для включения
света в большом доме 1
#define offLeftRecord (1)           // Голосовая команда для выключения
света в большом доме 1
#define onRightRecord (2)           // Голосовая команда для включения
света в большом доме 2
#define offRightRecord (3)          // Голосовая команда для выключения
света в большом доме 2
#define musicRecord (4)           // Голосовая команда для включения ме-
лодии
#define hideCommandsRecord (5)          // Голосовая команда для выклю-
чения мелодии
#define streetRecord (6)            // Голосовая команда для включе-
ния/выключения освещения на улице
// Сигнал с ИК датчиков, если нет препятствия, то сигнал 1, иначе 0
#define barrier1 digitalRead(irPin6)
#define barrier2 digitalRead(irPin7)
#define barrier3 digitalRead(irPin8)
// Мелодия для зуммера MARIO
int melody[] = {
  NOTE_E5, NOTE_E5, REST, NOTE_E5, REST, NOTE_C5, NOTE_E5,
```

```
 69    NOTE_G5, REST, NOTE_G4, REST,
 70    NOTE_C5, NOTE_G4, REST, NOTE_E4,
 71    NOTE_A4, NOTE_B4, NOTE_AS4, NOTE_A4,
 72    NOTE_G4, NOTE_E5, NOTE_G5, NOTE_A5, NOTE_F5, NOTE_G5,
 73    REST, NOTE_E5,NOTE_C5, NOTE_D5, NOTE_B4,
 74    NOTE_C5, NOTE_G4, REST, NOTE_E4,
 75    NOTE_A4, NOTE_B4, NOTE_AS4, NOTE_A4,
 76    NOTE_G4, NOTE_E5, NOTE_G5, NOTE_A5, NOTE_F5, NOTE_G5,
 77    REST, NOTE_E5,NOTE_C5, NOTE_D5, NOTE_B4,
 78    REST, NOTE_G5, NOTE_FS5, NOTE_F5, NOTE_DS5, NOTE_E5,
 79    REST, NOTE_GS4, NOTE_A4, NOTE_C4, REST, NOTE_A4, NOTE_C5, NOTE_D5,
 80    REST, NOTE_DS5, REST, NOTE_D5,
 81    NOTE_C5, REST,
 82    REST, NOTE_G5, NOTE_FS5, NOTE_F5, NOTE_DS5, NOTE_E5,
 83    REST, NOTE_GS4, NOTE_A4, NOTE_C4, REST, NOTE_A4, NOTE_C5, NOTE_D5,
 84    REST, NOTE_DS5, REST, NOTE_D5,
 85    NOTE_C5, REST,
 86    NOTE_C5, NOTE_C5, NOTE_C5, REST, NOTE_C5, NOTE_D5,
 87    NOTE_E5, NOTE_C5, NOTE_A4, NOTE_G4,
 88    NOTE_C5, NOTE_C5, NOTE_C5, REST, NOTE_C5, NOTE_D5, NOTE_E5,
 89    REST,
 90    NOTE_C5, NOTE_C5, NOTE_C5, REST, NOTE_C5, NOTE_D5,
 91    NOTE_E5, NOTE_C5, NOTE_A4, NOTE_G4,
 92    NOTE_E5, NOTE_E5, REST, NOTE_E5, REST, NOTE_C5, NOTE_E5,
 93    NOTE_G5, REST, NOTE_G4, REST,
 94    NOTE_C5, NOTE_G4, REST, NOTE_E4,
 95    NOTE_A4, NOTE_B4, NOTE_AS4, NOTE_A4,
 96    NOTE_G4, NOTE_E5, NOTE_G5, NOTE_A5, NOTE_F5, NOTE_G5,
 97    REST, NOTE_E5, NOTE_C5, NOTE_D5, NOTE_B4,
 98    NOTE_C5, NOTE_G4, REST, NOTE_E4,
 99    NOTE_A4, NOTE_B4, NOTE_AS4, NOTE_A4,
100    NOTE_G4, NOTE_E5, NOTE_G5, NOTE_A5, NOTE_F5, NOTE_G5,
101    REST, NOTE_E5, NOTE_C5, NOTE_D5, NOTE_B4,
102    NOTE_E5, NOTE_C5, NOTE_G4, REST, NOTE_GS4,
103    NOTE_A4, NOTE_F5, NOTE_F5, NOTE_A4,
104    NOTE_D5, NOTE_A5, NOTE_A5, NOTE_A5, NOTE_G5, NOTE_F5,
105    NOTE_E5, NOTE_C5, NOTE_A4, NOTE_G4,
```

```
106   NOTE_E5, NOTE_C5, NOTE_G4, REST, NOTE_GS4,
107   NOTE_A4, NOTE_F5, NOTE_F5, NOTE_A4,
108   NOTE_B4, NOTE_F5, NOTE_F5, NOTE_F5, NOTE_E5, NOTE_D5,
109   NOTE_C5, NOTE_E4, NOTE_E4, NOTE_C4,
110   NOTE_E5, NOTE_C5, NOTE_G4, REST, NOTE_GS4,
111   NOTE_A4, NOTE_F5, NOTE_F5, NOTE_A4,
112   NOTE_D5, NOTE_A5, NOTE_A5, NOTE_A5, NOTE_G5, NOTE_F5,
113   NOTE_E5, NOTE_C5, NOTE_A4, NOTE_G4,
114   NOTE_E5, NOTE_C5, NOTE_G4, REST, NOTE_GS4,
115   NOTE_A4, NOTE_F5, NOTE_F5, NOTE_A4,
116   NOTE_B4, NOTE_F5, NOTE_F5, NOTE_F5, NOTE_E5, NOTE_D5,
117   NOTE_C5, NOTE_E4, NOTE_E4, NOTE_C4,
118   NOTE_C5, NOTE_C5, NOTE_C5, REST, NOTE_C5, NOTE_D5, NOTE_E5,
119   REST,
120   NOTE_C5, NOTE_C5, NOTE_C5, REST, NOTE_C5, NOTE_D5,
121   NOTE_E5, NOTE_C5, NOTE_A4, NOTE_G4,
122   NOTE_E5, NOTE_E5, REST, NOTE_E5, REST, NOTE_C5, NOTE_E5,
123   NOTE_G5, REST, NOTE_G4, REST,
124   NOTE_E5, NOTE_C5, NOTE_G4, REST, NOTE_GS4,
125   NOTE_A4, NOTE_F5, NOTE_F5, NOTE_A4,
126   NOTE_D5, NOTE_A5, NOTE_A5, NOTE_A5, NOTE_G5, NOTE_F5,
127   NOTE_E5, NOTE_C5, NOTE_A4, NOTE_G4,
128   NOTE_E5, NOTE_C5, NOTE_G4, REST, NOTE_GS4,
129   NOTE_A4, NOTE_F5, NOTE_F5, NOTE_A4,
130   NOTE_B4, NOTE_F5, NOTE_F5, NOTE_F5, NOTE_E5, NOTE_D5,
131   NOTE_C5, NOTE_E4, NOTE_E4, NOTE_C4,
132   // Game over sound
133   NOTE_C5, NOTE_G4, NOTE_E4,
134   NOTE_A4, NOTE_B4, NOTE_A4, NOTE_GS4, NOTE_AS4, NOTE_GS4,
135   NOTE_G4, NOTE_D4, NOTE_E4
136 };
137 int durations[] = {
138   8, 8, 8, 8, 8, 8, 8,
139   4, 4, 8, 4,
140   4, 8, 4, 4,
141   4, 4, 8, 4,
142   8, 8, 8, 4, 8, 8,
```

```
143   8, 4,8, 8, 4,
144   4, 8, 4, 4,
145   4, 4, 8, 4,
146   8, 8, 8, 4, 8, 8,
147   8, 4,8, 8, 4,
148   4, 8, 8, 8, 4, 8,
149   8, 8, 8, 8, 8, 8, 8, 8,
150   4, 4, 8, 4,
151   2, 2,
152   4, 8, 8, 8, 4, 8,
153   8, 8, 8, 8, 8, 8, 8, 8,
154   4, 4, 8, 4,
155   2, 2,
156   8, 4, 8, 8, 8, 4,
157   8, 4, 8, 2,
158   8, 4, 8, 8, 8, 8, 8,
159   1,
160   8, 4, 8, 8, 8, 4,
161   8, 4, 8, 2,
162   8, 8, 8, 8, 8, 8, 4,
163   4, 4, 4, 4,
164   4, 8, 4, 4,
165   4, 4, 8, 4,
166   8, 8, 8, 4, 8, 8,
167   8, 4, 8, 8, 4,
168   4, 8, 4, 4,
169   4, 4, 8, 4,
170   8, 8, 8, 4, 8, 8,
171   8, 4, 8, 8, 4,
172   8, 4, 8, 4, 4,
173   8, 4, 8, 2,
174   8, 8, 8, 8, 8, 8,
175   8, 4, 8, 2,
176   8, 4, 8, 4, 4,
177   8, 4, 8, 2,
178   8, 4, 8, 8, 8, 8,
179   8, 4, 8, 2,
```

```
  8, 4, 8, 4, 4,
  8, 4, 8, 2,
  8, 8, 8, 8, 8, 8,
  8, 4, 8, 2,
  8, 4, 8, 4, 4,
  8, 4, 8, 2,
  8, 4, 8, 8, 8, 8,
  8, 4, 8, 2,
  8, 4, 8, 8, 8, 8, 8,
  1,
  8, 4, 8, 8, 8, 4,
  8, 4, 8, 2,
  8, 8, 8, 8, 8, 8, 4,
  4, 4, 4, 4,
  8, 4, 8, 4, 4,
  8, 4, 8, 2,
  8, 8, 8, 8, 8, 8,
  8, 4, 8, 2,
  8, 4, 8, 4, 4,
  8, 4, 8, 2,
  8, 4, 8, 8, 8, 8,
  8, 4, 8, 2,
  //game over sound
  4, 4, 4,
  8, 8, 8, 8, 8, 8,
  8, 8, 2
};
// Колличество нот
int size = 0;
// Нота
int note = 0;
// Число для отправки на сдвиговый регистр
byte dataToSend = 0;
// Состояние диодов в маленьких домах (включены или выключены)
bool stateLed4 = false;
bool stateLed5 = false;
// Состояние диода на улице (включены или выключены)
```

```
bool stateLed10 = false;
// Состояние мелодии (включены или выключены)
bool stateMusic = false;
// Состояние RGB модулей
bool rgbRight = false;
bool rgbLeft = false;
// Состояние сервопривода
bool servoState = false;
// Переменные для хранения значений с потенциометров для RGB модулей
int potValue1 = 0;          // Значение на 1 потенциометре
int potValue2 = 0;          // Значение на 2 потенциометре
// Функция настройки и инициализации всех компонетов подключенных к ардуино
void setup() {
  delay(300);               // Пауза для стабилизации
  myVR.begin(9600);
  // Загрузка ранее записанных голосовых команд в модуль распознования голоса
  myVR.load((uint8_t)onLeftRecord);
  myVR.load((uint8_t)offLeftRecord);
  myVR.load((uint8_t)onRightRecord);
  myVR.load((uint8_t)offRightRecord);
  myVR.load((uint8_t)musicRecord);
  myVR.load((uint8_t)hideCommandsRecord);
  myVR.load((uint8_t)streetRecord);
  Serial.begin(115200);
  // Настройка пинов для кнопок, включающие свет в маленьких домах
  pinMode(buttonPin5, INPUT_PULLUP);
  // Настройка пинов для ИК датчиков
  pinMode(irPin6, INPUT);
  pinMode(irPin7, INPUT);
  pinMode(irPin8, INPUT);
  // Настройка пина для зуммера
  pinMode(buzzer, OUTPUT);
  // Настройка пина для диода уличного освещения
  pinMode(ledPin10, OUTPUT);
  // Настройка пина для сервопривода
  servo.attach(servoPin);
```

```
  // Настройка пинов для сдвигового регистра
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}
// MAIN
void loop() {
  int ret;
  ret = myVR.recognize(buf, 50);
  if(ret > 0)
  {
    switch(buf[1])
    {
      case onLeftRecord:                              // Голосовая команда для включения света в 1 доме
        rgbLeft = true;
      break;
      case offLeftRecord:                             // Голосовая команда для выключения света в 1 доме
        rgbLeft = false;
      break;
      case onRightRecord:                             // Голосовая команда для включения света во 2 доме
        rgbRight = true;
      break;
      case offRightRecord:                            // Голосовая команда для выключения света в 2 доме
        rgbRight = false;
      break;
      case musicRecord:                              // Голосовая команда для включения мелодии
        stateMusic = !stateMusic;    // Устанавилваем флаг stateMusic в true для последующего включения мелодии
      break;
      case hideCommandsRecord:                              // Голосовая команда для выключения мелодии
        servoState = false;   // Устанавилваем флаг stateMusic в false для последующего выключения мелодии
      break;
      case streetRecord:                              // Голосовая команда для включения/выключения уличного освещения
```

```
291         stateLed10 = !stateLed10;
292         digitalWrite(ledPin10, stateLed10);        // Включаем/выключаем
свет улицы
293       break;
294     }
295   }
296   checkCommand();
297   checkIr();
298   checkButtons();
299   checkTime();
300   playMusic();
301 }
302 // Функция проверки голосовой команды
303 void checkCommand(){
304   if (rgbLeft == true)    // Если была подана команда на включение РГБ
модуля 1 дома
305   {
306     bitWrite(dataToSend, 3, LOW);              // Обнуляем цвета РГБ мо-
дуля 1 дома
307     bitWrite(dataToSend, 4, LOW);              // Обнуляем цвета РГБ мо-
дуля 1 дома
308     bitWrite(dataToSend, 5, LOW);              // Обнуляем цвета РГБ мо-
дуля 1 дома
309     checkPotValue();
310     rgbValue1();
311     DataSend();
312   }
313   if (rgbLeft == false)    // Если была подана команда на выключение РГБ
модуля 1 дома
314   {
315     bitWrite(dataToSend, 3, LOW);              // Обнуляем цвета РГБ мо-
дуля 1 дома
316     bitWrite(dataToSend, 4, LOW);              // Обнуляем цвета РГБ мо-
дуля 1 дома
317     bitWrite(dataToSend, 5, LOW);              // Обнуляем цвета РГБ мо-
дуля 1 дома
318     DataSend();
319   }
320   if (rgbRight == true)    // Если была подана команда на включение РГБ
модуля 2 дома
321   {
```

```
    bitWrite(dataToSend, 0, LOW);               // Обнуляем цвета РГБ модуля 2 дома
    bitWrite(dataToSend, 1, LOW);               // Обнуляем цвета РГБ модуля 2 дома
    bitWrite(dataToSend, 2, LOW);               // Обнуляем цвета РГБ модуля 2 дома
    checkPotValue();
    rgbValue2();
    DataSend();
  }
  if (rgbRight == false)    // Если была подана команда на выключение РГБ модуля 2 дома
  {
    bitWrite(dataToSend, 0, LOW);               // Обнуляем цвета РГБ модуля 1 дома
    bitWrite(dataToSend, 1, LOW);               // Обнуляем цвета РГБ модуля 1 дома
    bitWrite(dataToSend, 2, LOW);               // Обнуляем цвета РГБ модуля 1 дома
    DataSend();
  }
}
// Функция, которая принимает значение с ИК датчиков
void checkIr(){
  if (barrier1 == 0 || barrier2 == 0 || barrier3 == 0)
  {
    servoState = true;
  }
  if (servoState == true)
  {
    turnServo();
  }
  if (servoState == false)
  {
    returnServo();
  }
}
// Функция проверки нажатия кнопок для включения диодов в домиках
void checkButtons(){
  if (digitalRead(buttonPin5) == HIGH)
```

```
365   {
366     // Пока кнопку не отпустят, свет не включится, для избежания посто-
      янного включения/выключения света в доме
367     while (digitalRead(buttonPin5) == HIGH);
368     stateLed5 = !stateLed5;
369     bitWrite(dataToSend, 7, stateLed5);
370     stateLed4 = !stateLed4;
371     bitWrite(dataToSend, 6, stateLed4);
372   }
373   DataSend();
374 }
375 // Функция, которая смотрит, нужно ли включить/выключить свет улицы, со-
    гласно реальному времени
376 void checkTime(){
377   if (rtc.getHours() == 17 && rtc.getMinutes() == 0)        // Если ре-
    альное время 5 вечера, включаем свет улицы
378   {
379     stateLed10 = true;
380     digitalWrite(ledPin10, stateLed10);
381   }
382   if (rtc.getHours() == 9 && rtc.getMinutes() == 0)         // Если ре-
    альное время 9 утра, выключаем свет улицы
383   {
384     stateLed10 = false;
385     digitalWrite(ledPin10, stateLed10);
386   }
387 }
388 // Функция воспроизведения музыки
389 void playMusic(){
390   // Воспроизводим музыку если была поданна соответствующая голосовая
    команда (stateMusic будет true)
391   if (stateMusic == true)
392   {
393     size = sizeof(durations) / sizeof(int);
394     if (note < size)
395     {
396       int duration = 1000 / durations[note];
397       tone(buzzer, melody[note], duration);
398       // Чтобы различать ноты, установливаем минимальное время между
      ними
399
```

```
      int pauseBetweenNotes = duration * 1.30;
      delay(pauseBetweenNotes);
      // Останавливаем музыку
      noTone(buzzer);
      note = note + 1;
    }
    else
    {
      note = 0;
    }
  }
  else                // Иначе мелодия играть не будет
  {
    note = 0;
  }
}
 // Функция, которая считывает значения с потенциометров для RGB модулей
void checkPotValue(){
  potValue1 = analogRead(A2);
  potValue2 = analogRead(A3);
}
// Функция преобразования значения с 1 потенциометра в цвета для РГБ мо-
дуля 1
void rgbValue1(){
  if (potValue1 < 200)
  {
    bitWrite(dataToSend, 4, HIGH);
    bitWrite(dataToSend, 5, HIGH);
  }
  if (potValue1 >= 200 && potValue1 <= 300)
  {
    bitWrite(dataToSend, 3, HIGH);
  }
  if (potValue1 > 300 && potValue1 < 500)
  {
    bitWrite(dataToSend, 3, HIGH);
    bitWrite(dataToSend, 5, HIGH);
  }
```

```
439   if (potValue1 >= 500 && potValue1 <= 600)
440   {
441     bitWrite(dataToSend, 4, HIGH);
442   }
443   if (potValue1 > 600 && potValue1 < 800)
444   {
445     bitWrite(dataToSend, 3, HIGH);
446     bitWrite(dataToSend, 4, HIGH);
447   }
448   if (potValue1 >= 800 && potValue1 <= 900)
449   {
450     bitWrite(dataToSend, 5, HIGH);
451   }
452   if (potValue1 > 900)
453   {
454     bitWrite(dataToSend, 3, HIGH);
455     bitWrite(dataToSend, 4, HIGH);
456     bitWrite(dataToSend, 5, HIGH);
457   }
458 }
459 // Функция преобразования значения с 2 потенциометра в цвета для РГБ мо-
    дуля 2
460
461 void rgbValue2(){
462   if (potValue2 < 200)
463   {
464     bitWrite(dataToSend, 1, HIGH);
465     bitWrite(dataToSend, 2, HIGH);
466   }
467   if (potValue2 >= 200 && potValue2 <= 300)
468   {
469     bitWrite(dataToSend, 0, HIGH);
470   }
471   if (potValue2 > 300 && potValue2 < 500)
472   {
473     bitWrite(dataToSend, 0, HIGH);
474     bitWrite(dataToSend, 2, HIGH);
475   }
      if (potValue2 >= 500 && potValue2 <= 600)
```

```
476   {
477     bitWrite(dataToSend, 1, HIGH);
478   }
479   if (potValue2 > 600 && potValue2 < 800)
480   {
481     bitWrite(dataToSend, 0, HIGH);
482     bitWrite(dataToSend, 1, HIGH);
483   }
484   if (potValue2 >= 800 && potValue2 <= 900)
485   {
486     bitWrite(dataToSend, 2, HIGH);
487   }
488   if (potValue2 > 900)
489   {
490     bitWrite(dataToSend, 0, HIGH);
491     bitWrite(dataToSend, 1, HIGH);
492     bitWrite(dataToSend, 2, HIGH);
493   }
494 }
495 // Функция отправки данных на РГБ модули и светодиоды маленьких домиков
496 void DataSend(){
497   digitalWrite(latchPin, LOW);                          // Начинаем пере-
    дачу данных
498
499   shiftOut(dataPin, clockPin, LSBFIRST, dataToSend);
500   digitalWrite(latchPin, HIGH);                         // Прекращаем
    передачу данных
501   delay(10);
502 }
503 // Функция для поворота сервопривода на 180 градусов
504 void turnServo() {
505   servo.write(180);
506 }
507 // Функция для возврата сервопривода в исходное положение
508 void returnServo(){
509   servo.write(0);
510 }
```

ПРИЛОЖЕНИЕ Б

Инструкция по голосовому управлению

В получившийся макет загружено 7 голосовых команд:

1. «Один» – включение первого RGB модуля;
2. «Два» – выключение первого RGB модуля;
3. «Три» – включение второго RGB модуля;
4. «Четыре» – выключение второго RGB модуля;
5. «Пять» – включение и выключение мелодии;
6. «Шесть» – опустить табличку с голосовыми командами;
7. «Семь» – включить и выключить уличное освещение.

Чтобы запустить макет, нужно подключить его к блоку питания на 5 вольт или через порт USB в компьютер, как только появится питание, макет будет работать.

При записи команд, обязательно нужно учитывать порядок их записи. Чтобы изменить голосовые команды нужно выполнить следующие шаги:

1. Подключить макет к компьютеру по USB порту;
2. В Arduino IDE открыть скетч vr_sample_train из библиотеки VoiceRecognitionV3;
3. Прошить его на ардуино и открыть монитор порта;
4. В мониторе порта указать меню NL и скорость порта 115200;
5. Прописать команду clear;
6. После с помощью команды train или sigtrain записать нужные нам голосовые команды. После того как они будут записаны, прописать команду load с номерами записанных нами команд;
7. В Arduino IDE открыть программный код учебно-образовательного макета «Умная деревня» и прошить его на ардуино.

ПРИЛОЖЕНИЕ В

Перечень элементов

Поз. обозначение	Наименование	Кол.	Примечание
	Схемы интегральные		
MCU	Arduino UNO	1	
	Датчики		
U1–U3	YL–63	3	
A1	GSMIN DS3231	1	
	Резисторы и потенциометры		
RP1–RP2	WH148 R16K4 — 5 кОм	2	
R1–R3	ОМЛТ–2 — 5 — 220 Ом ОЖ0.467.107 ТУ	3	
	Модули записи и воспроизведения звука		
S1	KY–006	1	
M1	Elechouse v3.1	1	
	Кнопки		
SA1	RUICHI GQ12B–10/J/B/W/N	1	
	Сервопривод		
E1	Tower sg90 pro	1	
	Светодиоды		
VD1–VD3	Одноцветный светодиод	3	
D1–D2	RGB KY–016	2	
	Сдвиговый регистр		
RG	74HC595	1	

Изм.	Лист	№ документа	Подпись	Дата	ВКР — 09.03.01.ПЭ		
Разработал		Халаманов А. П.			Разработка учебно-образовательного макета «Умная деревня»	Литера	Лист 1 / Листов 1
Проверил		Макуха Л. В.					
Н. контр.					Перечень элементов	Кафедра вычислительной техники	
Утвердил							

Министерство науки и высшего образования РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____О.В. Непомнящий
«17» 06 2024 г.

БАКАЛАВРСКАЯ РАБОТА

090301 Информатика и вычислительная техника

Учебно-образовательный макет «Умная деревня»

Руководитель	подпись	17.06.24 дата	старший преподаватель должность, ученая степень	Л.В. Макуха
Выпускник	подпись	17.06.24 дата		А.П. Халаманов
Нормоконтролёр	подпись	17.06.24 дата	старший преподаватель должность, ученая степень	Л.В. Макуха

Красноярск 2024