**Муниципальное автономное общеобразовательное учреждение**

**г. Владимира «Средняя общеобразовательная школа №25»**

**ИНДИВИДУАЛЬНЫЙ ИТОГОВЫЙ ПРОЕКТ**

**Устройство интеллектуальной системы доступа**

Выполнил:

Игнатов Кирилл Олегович

обучающийся 10 «А» класса

Руководитель проекта:

Варакин Алексей Александрович

Консультант проекта:

Шумарин Сергей Викторович

г. Владимир

2022

**ОГЛАВЛЕНИЕ**

ВВЕДЕНИЕ…………………………………………………………………... с.

I РАЗДЕЛ 1. (теоретический) ……………………………………………….. с.

II РАЗДЕЛ 2. (практический) ……………………………………………….. с.

ЗАКЛЮЧЕНИЕ ………………………………………………………........... с.

СПИСОК ЛИТЕРАТУРЫ …………………………………………………... с.

ПРИЛОЖЕНИЯ

**ВЕДЕНИЕ**

**Тема проекта и её актуальность:**

Устройство интеллектуальной системы доступа. Тема была выбрана из-за возросшей в последние годы потребности в увеличении уровня приватности на различных предприятиях. Использовать данный продукт можно в местах, где требуется контролировать и ограничивать доступ.

**Цель проекта:**

Разработка прототипа системы интеллектуального доступа на базе RFID.

**Задачи:**

1. Изучить теоретический материал.
2. Изучение языка программирования C++ и среды разработки Arduino, изучение остального теоретического материала.
3. Изучить систему автоматизированного проектирования SolidWorks.
4. Установка необходимых модулей на макетную плату Arduino, написание программы.
5. Выращивание вертушки на 3D принтере.
6. Сборка устройства.

**Этапы:**

1. Размещение компонентов на макетной плате.
2. Написание прошивки для устройства.
3. Создание детали-вертушки и установка ее на шаговой мотор.
4. Отладка.

**РАЗДЕЛ I**.(теоретический)

# I.1 Принцип действия.

Различные системы контроля и управления доступом сейчас очень распространены и востребованы. Системы контроля и управления доступом, сокращенно СКУД - комплекс аппаратуры, алгоритмов и программ, предназначенный для ограничения входа/выхода на обслуживаемый объект, внутренних перемещений, фиксации и протоколирования событий и решения других сопутствующих задач. СКУДы предназначены для: идентификации пользователей; ограничению доступа на какой-либо объект; отслеживать другие события, например, открытие дверей и перемещение идентифицированных пользователей во внутреннем пространстве; вести и изменять базу данных пользователей и идентификаторов; предоставлять администраторам системы доступ к базам данных и/или отчетам; протоколировать происходящие в системе события, в том числе с применением алгоритмов шифрования; обмениваться данными и сигналами с другими комплексами, например, охранной сигнализации или бухучета.

Одной из основных задач СКУД является идентификация пользователя для применения к нему ограничений или предоставления прав. Соответственно, в комплект системы в обязательном порядке входят специальные датчики – считыватели. Они делятся на считыватели кодов и считыватели биометрических данных таких, как отпечаток пальца или сетчатка глаза. Если перечислять основные типы считывателей, можно отметить: оптические (считыватели штрих кодов), магнитные (активируются картами с магнитными лентами), считыватели виртуальных данных (мобильные телефоны с приложениями, которые передают данные считывателю), считыватели ПИН-кодов и биометрические сканеры. Также одним из типов считывания является бесконтактный. Основу технологии составляет считывание информации, записанной в области памяти специальных транспондеров (RFID-меток). Этот вариант получил максимальное распространение (например, в картах proximity) на текущий момент.

Бесконтактные метки начали появляться в конце прошлого века, всего насчитывается четыре поколения бесконтактных меток. Годом появления первого поколения принято считать 1990 г., носителем была proximity карта, частота радиосигнала составляла 125КГц, шифрования не подразумевалось. Второе поколение бесконтактных меток появилось в 1998 г. от предыдущего поколения его отличали повышенная частота радиосигнала – 13,56 МГц и шифрование в лице передачи секретных ключей. Следующее поколение, третье, вышло в 2011 г. от предыдущего его отличали невозможность копирования носителя и защита 128-битным шифрованием Secure Identity Object. Четвертое поколение вышло в 2018 г., носителем стал смартфон, частота радиосигнала составляла 2.4 ГГц (Wi-Fi), защита осталось той же, что и в предыдущем поколении.

Основными преимуществами бесконтактных меток являются:

* Простота аппаратуры для считывания сигналов транспондеров и ее доступность на рынке в разных ценовых диапазонах.
* Высокая надежность чипов – срок службы их практически не ограничен.
* Удобство использования – метка может быть размещена практически на любом носителе.

Технология практически не имеет недостатков. Однако с появлением в широкой продаже устройств для работы с RFID-метками была частично скомпрометирована – сегодня изготовить дубликат proximity-карты или другого аналогичного девайса можно в течение нескольких минут.

Решаемые СКУД задачи требуют наличия в системах исполнительных механизмов, получивших название преграждающих. К ним относят:

* Замки и защелки с электрическим приводом.
* Автоматические двери.
* Турникеты.
* Управляемые ворота, шлагбаумы.

Их назначение – заблокировать или предоставить доступ в ответ на определенный набор событий (прежде всего, прохождение пользователем идентификации) и условий. Совместно с ними или независимо могут работать и другие устройства, такие как:

* Приводы камер наблюдения.
* Аппаратура записи.
* Приборы сигнализации и оповещения, как общего, так и персонального с рассылкой сообщений по сети или другим каналам связи.

Вертушка, которая является замком выращена на 3d принтере. 3d принтер – это станок с числовым программным управлением, реализующий только аддитивные операции, то есть только добавляющий порции материала к заготовке. Обычно использует метод послойной печати детали. Принцип 3D-печати по любой существующей технологии — создание объемных объектов из совокупности плоских слоев.

Цифровая модель изделия разделяется на слои специальной программой — слайсером, а принтер печатает эти слои, один на другом, составляя из них трехмерный объект. Так, из множества слоев, получается объемная деталь.

Общий принцип один, но технологии различаются; самая распространенная и доступная среди них — FDM. Моделирование методом послойного наплавления (FDM), также известное как производство способом наплавления нитей (FFF) — самый популярный и массовый тип 3D-печати. Стандартное FDM-устройство работает как термоклеевой пистолет, управляемый роботом, что не удивляет, ведь разработка технологии FDM когда-то начиналась с опытов с термоклеем. Пластиковый пруток проталкивается через горячее сопло, где он плавится, а выходя из него укладывается слоями. Процесс повторяется снова и снова, пока не появится готовый 3D-объект. Единственное отличие в том, что 3D-принтеры используют не стержни термоклея, а пластиковый филамент намотанный на катушки.

# I.2 Технические характеристики компонентов.

Планируется за основу взять плату Arduino Nano, Nano – одна из самых миниатюрных плат Arduino. Она является полным аналогом Arduino Uno – так же работает на чипе ATmega328P. Технические характеристики платы:

* Напряжение питания 5В;
* Входное питание 7-12В (рекомендованное);
* Количество цифровых пинов – 14, из них 6 могут использоваться в качестве выходов ШИМ;
* 8 аналоговых входов;
* Максимальный ток цифрового выхода 40 мА;
* Флэш- память 16 Кб или 32 Кб, в зависимости от чипа;
* ОЗУ 1 Кб или 2 Кб, в зависимости от чипа;
* EEPROM 512 байт или 1 Кб;
* Частота 16 МГц;
* Размеры 19 х 42 мм;
* Вес 7 г.

На плате имеются 4 светодиода, которые показывают состояние сигнала. Они обозначены как TX, RX, PWR и L. На первых двух светодиод загорается, когда уровень сигнала низкий, и показывает, что сигнал TX или RX активен. Светодиод PWR загорается при напряжении в 5 В и показывает, что подключено питание. Последний светодиод – общего назначения, загорается, когда подается высокий сигнал.

При создании проекта будет использован шаговой двигатель. Шаговые двигатели преобразуют электрические импульсы в перемещение вала на определенный угол. Минимально возможный угол перемещения шагового двигателя, называется шагом. Была использована модель 28BYJ-48. Характеристики:

* Напряжение питания – 5В или 12В;
* Число фаз – 4;
* Коэффициент редукции – 1/63.68395;
* Количество шагов ротора – 64;
* Номинальная скорость вращения – 15 оборот/мин;
* Крутящий момент – 450 г\*см;
* Размеры (диаметр, высота) –25x18 мм;
* Вес – 40 грамм.

Считывателем будет взят RFID модуль RC522. Радиочастотная идентификация (RFID) — это технология бесконтактной идентификации объектов при помощи радиочастотного канала связи. Идентификация объектов производится по уникальному идентификатору, который имеет каждая электронная метка. Считыватель излучает электромагнитные волны определенной частоты. Метки отправляют в ответ информацию – идентификационный номер, данные памяти и пр. Преимущества технологии RFID:

* бесконтактная
* возможность скрытой установки меток
* высокая скорость считывания данных
* возможность установки во вредных средах
* невозможность подделки

Технические характеристики RC522:

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

Также при создании проекта использовалась макетная плата Arduino и драйвер для шагового двигателя ULN2003.

**РАЗДЕЛ II**. (практический)

# II.1 Структура устройства.

Разработанное устройство состоит из Arduino Nano, с подключенном к нему шаговым двигателем и считывателем RDIF меток, структурная схема показана на рисунке 1

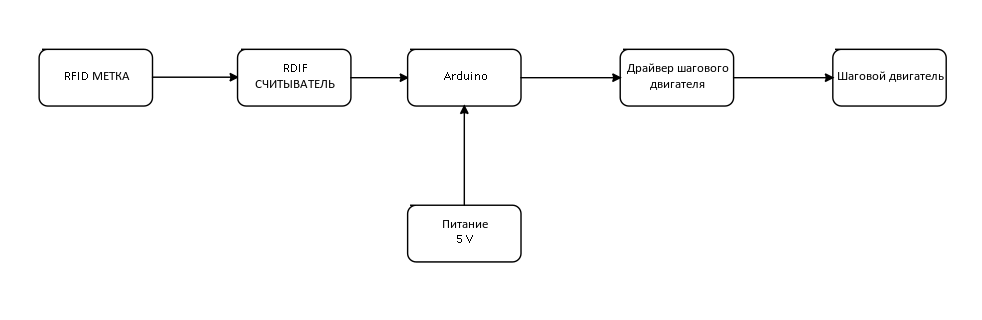


Рисунок 1. Структурная схема электронного замка.

# II.2 Описание программы.

Для открытия замка необходимо приложить валидный RFID ключ к считывателю. После проверки в случае совпадения будет осуществлено открытие замка на 5 секунд. Алгоритм работы показан на рисунке 2. Программа прошивки для Arduino приведена в Приложении.

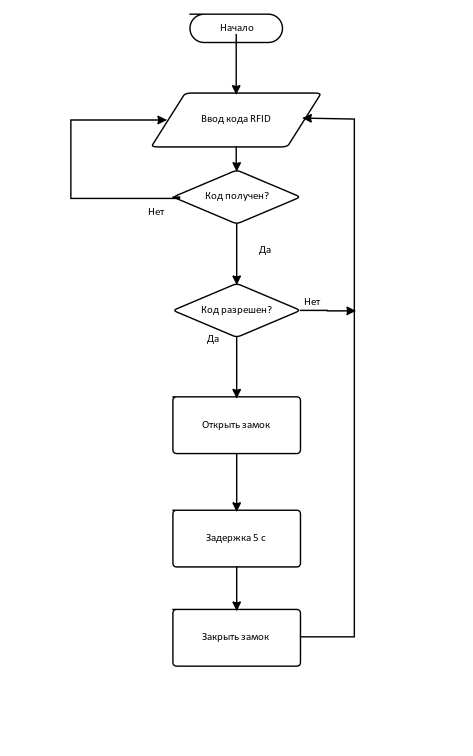


Рисунок 2. Алгоритм работы программы.

Программа написана на языке программирования C++ в среде разработки Arduino. Сначала идет импортирование необходимых библиотек: SPI, она позволяет микроконтроллеру взаимодействовать с устройствами, поддерживающими SPI протокол; MFRC522, эта библиотека нужна для чтения/записи RFID карты или метки. Далее идет определение констант для подключения модулей к плате Arduino Nano, после нее идет описание переменных, необходимых для работы шагового мотора. Далее идет определение функции CheckLastPhase(), она предназначена для проверки того, совершил ли двигатель нужное количество шагов, в данном случае это 1024 шага, что эквивалентно 90 градусам. Если мотор совершил 1024 шага в нужном направлении, то программа это направление меняет, умножая значение переменной на -1 и изменяет значение переменной flip на true, что сообщает о том, что сделан поворот на 90 градусов. Далее мы определяем функцию Flipping(). Данная функция запускает двигатель и после каждого шага проверяет количество пройденных шагов с помощью функции CheckLastPhase(). В функции setup() мы инициализируем, иными словами сообщаем программе о существовании мотора, считывателя радио меток и подключаем программу через порт, задавая скорость обмена данными в 9600 бит/сек, т.к. она наиболее оптимальная. Заключительной функцией в нашей программе является функция loop, которая “отлавливает” все необходимые события. Функция постоянно ищет какую-либо метку и если находит, то выдает считывает ее серийный номер и сравнивает его с заданным. Если совпадает, то происходит открытие замка с помощью функции Flipping(), после этого программа ждет 5 секунд и закрывает замок с помощью той же Flipping.

# II.3 Создание детали вертушки.

Вертушка была выращена на 3D принтере, чертеж детали был выполнен в системе автоматизированного проектирования SolidWorks (рисунок 3). Деталь имеет размеры 13x2.5x0.5 см. Готовая деталь представлена на рисунке 4.

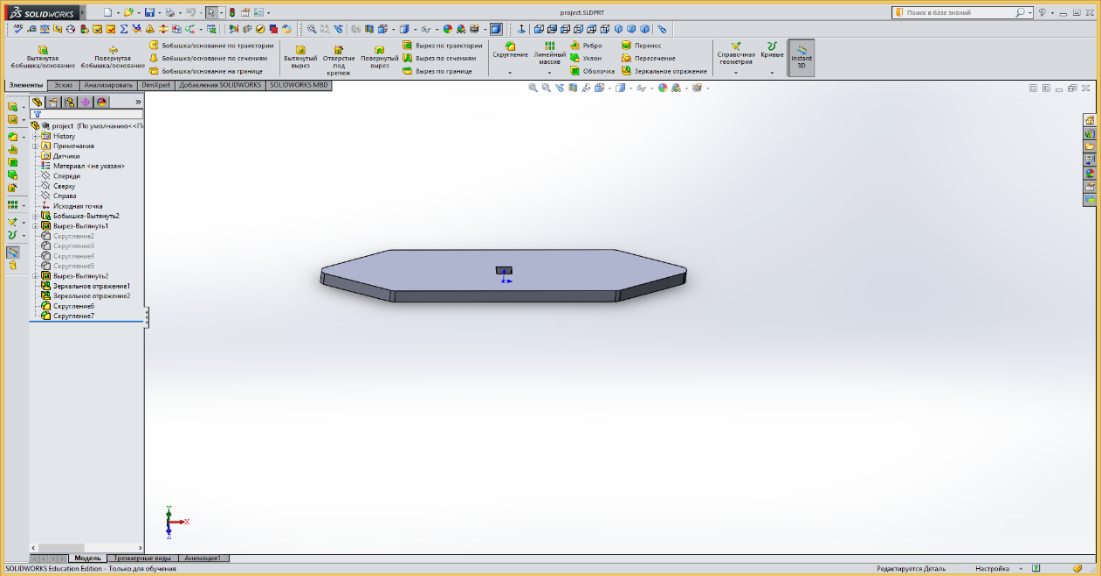


Рисунок 3. Деталь в SolidWorks.

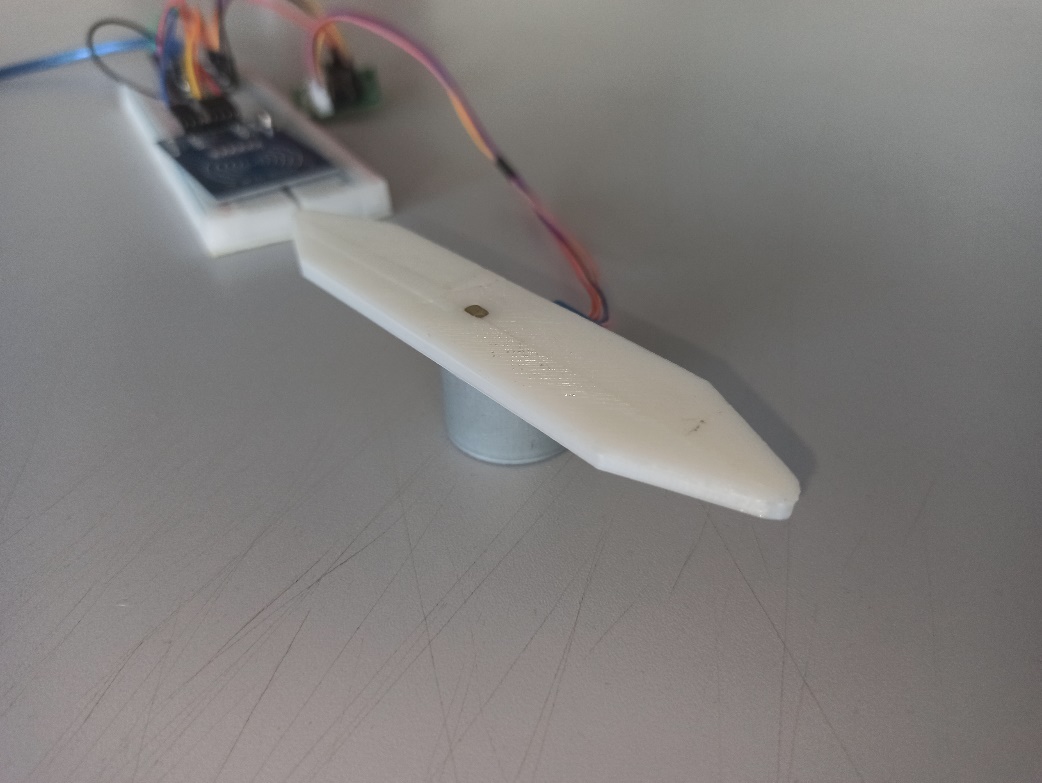


Рисунок 4. Выращенная на 3D принтере деталь.

# II.4 Сборка на макетной плате.

Изначально на макетную плату была установлена плата Arduino Nano, затем был подсоединен RFID считыватель, после этого был подключен шаговой двигатель при помощи драйвера, в последнюю очередь на двигателе была закреплена вертушка. Питание для работы конструкции подается через USB провод, подключенный к Arduino Nano. Устройство в готов виде показано на рисунке 5.

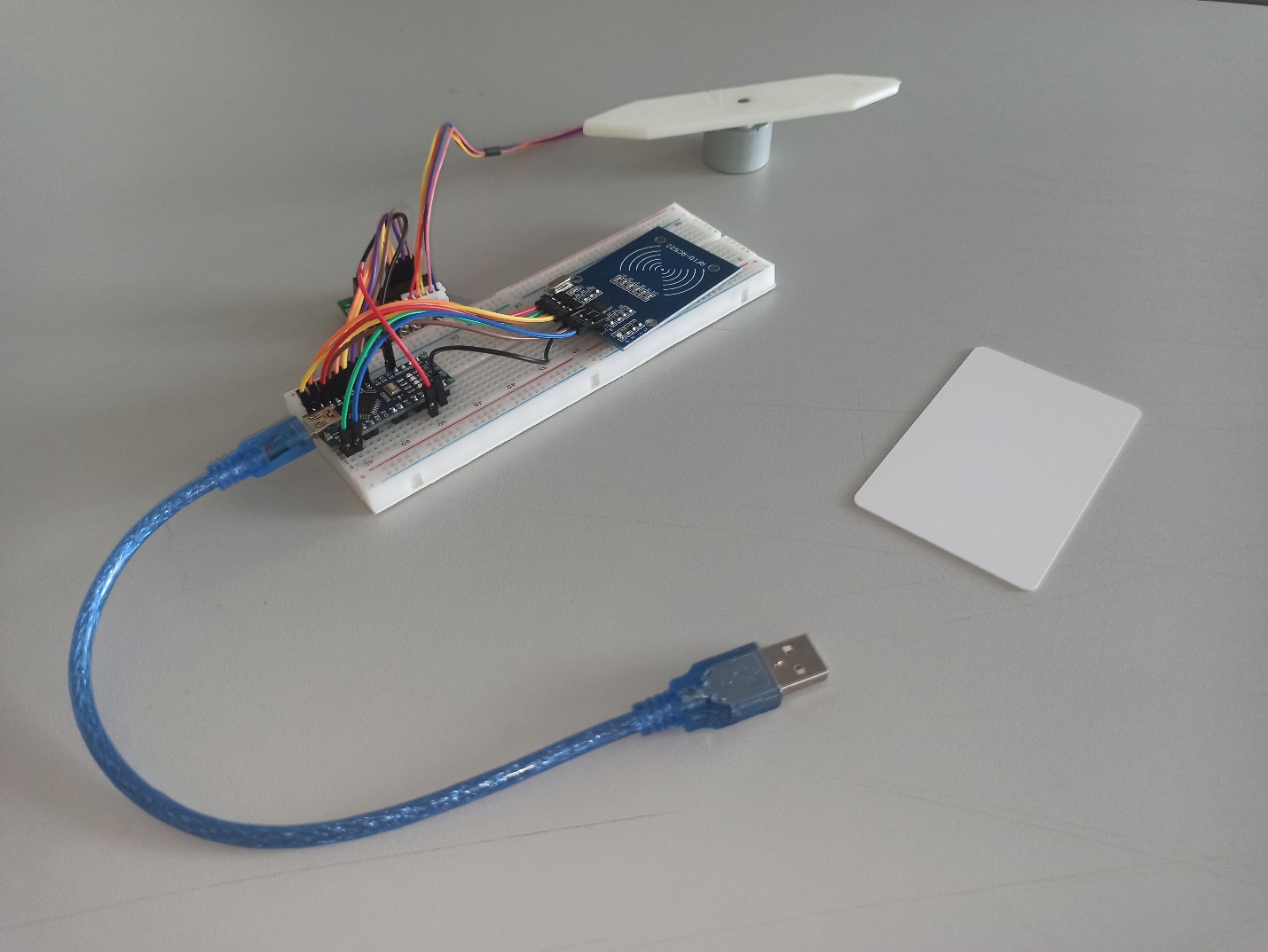


Рисунок 5. Собранное устройство.

**ЗАКЛЮЧЕНИЕ**

Подводя итог, я считаю, что можно сказать, что я достиг поставленной цели, я собрал прототип системы интеллектуального доступа на основе RFID. Моя система определяет, подходит ли обнаруженная ей метка и открывает замок, в случае, если она подошла. Затрагивая вещи, которые можно доработать, можно добавить базу данных и возможность добавлять новые метки, а также разграничивать доступ у различных меток, не затрагивая программный код.

**ПРИЛОЖЕНИЯ.**

|  |
| --- |
| #include <SPI.h> |
|  | #include <MFRC522.h> // библиотека "RFID". |
|  | #define SS\_PIN 10 |
|  | #define RST\_PIN 9 |
|  | MFRC522 mfrc522(SS\_PIN, RST\_PIN); |
|  | unsigned long uidDec, uidDecTemp; // для храниения номера метки в десятичном формате |
|  |  |
|  |  |
|  | bool f = false; |
|  | bool flip = false; |
|  |  |
|  | int MotorPins[4] = {5, 6, 7, 8}; |
|  | /\*Целочисленная константа, показывающая количество фаз подачи сигналов для одного шага мотора. Для полушагового режима - 8 |
|  | Для шагового - 4\*/ |
|  | const int OneTurnPhasesCount = 8; |
|  |  |
|  | /\*Целочисленная константа, показывающая задержку в миллисекундах между фазами подачи сигналов мотору. Для полушагового режима - 2, |
|  | для шагового - 3\*/ |
|  | const int TurnPhasesDelay = 2; |
|  |  |
|  | /\*Целочисленная константа, показывающая количество шагов, которые должен выполнить двигатель за полный оборот на 90 град. |
|  | Внутренний вал мотора совершает 64 шага за полный оборот, с учётом передаточного числа редуктора 64:1, то мотор должен совершать 64x64/4=1024 шагов\*/ |
|  | const int CountStepsOneDirection = 1024; |
|  |  |
|  | /\*Целочисленная переменная, показывающая количество шагов, которые выполнил двигатель в одном направлении\*/ |
|  | int CurrentStepOneDirection = 0; |
|  |  |
|  | /\*Целочисленная переменная, показывающая номер текущей фазы\*/ |
|  | int CurrentPhase = 0; |
|  |  |
|  | /\*Целочисленная переменная, показывающая направление вращения мотора: 1 - по часовой стрелке, -1 - против\*/ |
|  | int TurnDirection = 1; |
|  |  |
|  | // Для полушагового режима |
|  |  |
|  | /\*Массив, в котором указано какие сигналы подавать на контакты мотора в той или иной фазе. [фаза][контакт]. Контакты даются в порядке, перечисленном в массиве MotorPins - оранж., жёлт., розов., син. 0 - нет сигнала, 1 - есть сигнал\*/ |
|  | bool MotorTurnPhases[8][4] = { |
|  | { 1, 0, 0, 0}, |
|  | { 1, 1, 0, 0}, |
|  | { 0, 1, 0, 0}, |
|  | { 0, 1, 1, 0}, |
|  | { 0, 0, 1, 0}, |
|  | { 0, 0, 1, 1}, |
|  | { 0, 0, 0, 1}, |
|  | { 1, 0, 0, 1} |
|  | }; |
|  |  |
|  | /\*Функция CheckLastPhase проверяет не вышел ли номер текущей фазы за пределы размера массива MotorTurnPhases, который определяется переменной OneTurnPhasesCount и не пора ли поменять направление вращения\*/ |
|  | void CheckLastPhase() |
|  | { |
|  | if (CurrentPhase >= OneTurnPhasesCount) |
|  | { |
|  | CurrentPhase = 0; |
|  | } |
|  | if (CurrentPhase < 0) |
|  | { |
|  | CurrentPhase = (OneTurnPhasesCount - 1); |
|  | } |
|  |  |
|  | //Увеличиваем шаг на 1 |
|  | CurrentStepOneDirection++; |
|  |  |
|  | //проверяем не совершил ли мотор полный оборот |
|  | if (CurrentStepOneDirection == CountStepsOneDirection) |
|  | { |
|  | CurrentStepOneDirection = 0; |
|  | TurnDirection \*= -1; |
|  | flip = true; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | void Flipping() { |
|  | while (not flip) { |
|  | /\*подаём напряжения на контакты мотора соответственно фазе, заданной в массиве MotorTurnPhases\*/ |
|  | for (int i = 0; i < 4; i++) { |
|  | digitalWrite(MotorPins[i], ( (MotorTurnPhases[CurrentPhase][i] == 1) ? HIGH : LOW) ); |
|  | } |
|  |  |
|  | //переходим к другой фазе |
|  | CurrentPhase += TurnDirection; |
|  |  |
|  | // Пауза между фазами |
|  | delay(TurnPhasesDelay); |
|  | CheckLastPhase(); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | void setup() { |
|  | Serial.println("Start"); |
|  | Serial.begin(9600); |
|  | SPI.begin(); // инициализация SPI / Init SPI bus. |
|  | mfrc522.PCD\_Init(); // инициализация MFRC522 / Init MFRC522 card. |
|  | for (int i = 0; i < 4; i++) |
|  | pinMode(MotorPins[i], OUTPUT); |
|  | } |
|  |  |
|  |  |
|  | void loop() { |
|  | // Поиск новой метки |
|  | if ( ! mfrc522.PICC\_IsNewCardPresent()) { |
|  | return; |
|  | } |
|  | // Выбор метки |
|  | if ( ! mfrc522.PICC\_ReadCardSerial()) { |
|  | return; |
|  | } |
|  |  |
|  | uidDec = 0; |
|  | // Выдача серийного номера метки. |
|  | for (byte i = 0; i < mfrc522.uid.size; i++) |
|  | { |
|  | uidDecTemp = mfrc522.uid.uidByte[i]; |
|  | uidDec = uidDec \* 256 + uidDecTemp; |
|  | } |
|  | Serial.println("Card UID: "); |
|  | Serial.println(uidDec); // Выводим UID метки в консоль. |
|  | if ((uidDec == 1199345248) and (not f)) { |
|  | Serial.println("Podoshlo"); |
|  | flip = false; |
|  | f = true; |
|  | Flipping(); |
|  | flip = false; |
|  | delay(5000); |
|  | Serial.println("Back"); |
|  | Flipping(); |
|  | } |
|  | else |
|  | f = false; |
|  | } |