МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное учреждение высшего образования
ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт математики, механики и компьютерных наук имени И. И. Воровича

Направление подготовки 01.03.02 — «Прикладная математика и информатика»

Кафедра прикладной математики и программирования

Югай К.Ю., 3 курс, 2 группа

Курсовая работа

ИССЛЕДОВАНИЕ ПОДХОДОВ К УПРАВЛЕНИЮ УСТРОЙСТВАМИ НА ОСНОВЕ ARDUINO ИЗ PASCALABC.NET

Научный руководитель:
ст. преподаватель Пучкин М.В.

85 (отл.)
оценка (рейтинг)

подпись руководителя

Ростов-на-Дону

2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Arduino – одна из самых распространенных платформ для создания прототипов электронных устройств.

Существуют различные способы взаимодействия приложений и программ с устройствами на основе Arduino. Мы можем управлять устройством с помощью программ на различных языках посредством различных каналов связи. Для этого используются определенные протоколы взаимодействия, которые мы и рассмотрим.

Цель данной курсовой работы – рассмотреть способы взаимодействия приложений для платформы .NET и устройств на основе платформы Arduino, уделив особое внимание протоколам передачи данных посредством виртуальных COM-портов поверх USB. Изучить и провести сравнительный анализ различных протоколов взаимодействия с устройствами Arduino, предоставляющих полный контроль устройства, и разработать библиотеку для управления различными проектами на основе Arduino.

ПОСТАНОВКА ЗАДАЧИ

– изучить особенности взаимодействия с устройствами на основе Arduino – посредством Wi-Fi и USB, основное внимание уделив использованию виртуальных COM-портов поверх USB;

– рассмотреть и подготовить сравнительный обзор протоколов управления Arduino-устройствами, ориентированными на контроль состояния платы «извне»;

– сформулировать требования к сторонней библиотеке, учитывающие как временные задержки при взаимодействии с платой, так и особенности передачи данных;

– реализовать демонстрационную систему, позволяющую выполнять простейшие действия из среды разработки PascalABC.NET.

1. Исследование предметной области

1.1 Программирование Arduino

Arduino - это платформа с открытым исходным кодом, которая состоит из аппаратного и программного обеспечения, позволяющая быстро разрабатывать различные интерактивные проекты. Аппаратная часть включает в себя большое количество видов плат Arduino со встроенными программируемыми микроконтроллерами, а также дополнительные модули. Программная часть состоит из среды разработки, упрощенного языка программирования, огромного множества готовых функций и библиотек.

Остановимся на второй части, а именно – рассмотрим, как же программируется Arduino.

Arduino использует свой собственный язык программирования, который очень напоминает язык C++. Мы можем управлять платой, написав код в среде разработки Arduino IDE на этом языке, а затем загрузив полученный скетч – программу – посредством USB-соединения или Wi-Fi.

В любой программе для Arduino есть две части: подготовительная часть и основной цикл.

В подготовительной части пишем, чего от нас ожидать: какие порты настроить на вход, какие на выход, что как называется. Вся часть с подготовкой выполняется один раз при старте контроллера. Контроллер всё запоминает и переходит в основной цикл.

Основной цикл – это то, что происходит в функции loop(). Arduino последовательно выполняет команды внутри этой функции бесконечное количество раз.

В основном цикле мы описываем все, что должен делать контроллер: считывать данные, мигать лампами, включать и выключать моторы и т.д.

Пример 1. Код для Arduino.

```
int led = 13;
void setup() {
  pinMode(led, OUTPUT);
}
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Если мы подключим к пину 13 на плате светодиод, то данный код реализует мигание этого светодиода. В первой строчке инициализируем глобальную переменную led значением 13, далее в функции setup() обозначим переменную led, а точнее пин 13, как выход. Инициализация глобальной переменной и функция setup() – подготовительная часть, код, который выполняется один раз при запуске или включении платы. Далее начинается основной цикл – функция loop(). Отправляем на пин 13 значение HIGH, тем самым включив светодиод, немного ждем, а затем посылаем LOW, и светодиод гаснет. Так как код выполняется циклично, светодиод будет загораться и гаснуть бесконечное количество раз.

1.2 Взаимодействие с Arduino и протоколы передачи данных

Существует несколько подходов к управлению платформой Arduino с компьютера. Один из таких подходов – запуск основной программы на ПК и использование последовательного соединения через USB-кабель.

Также мы можем взаимодействовать с платой посредством Wi-Fi соединения, однако для этого необходим специальный модуль ESP8266, который позволит Arduino обмениваться данными с другими модулями по беспроводным сетям. Сейчас существуют платы со встроенным Wi-Fi модулем, но его при необходимости можно подключить к плате и отдельно.

Далее мы будем рассматривать первый подход – использование виртуальных COM-портов поверх USB.

Мы уже знаем, что Arduino использует свой собственный язык программирования, он необходим, если мы хотим прошить плату из Arduino IDE. Однако, Arduino можно использовать и с другими языками. Мы можем запустить основную программу практически на любом языке программирования. Возникает логичный вопрос – как же Arduino будет понимать нас на другом языке?

Для взаимодействия Arduino с ПК необходим протокол. Он нужен для того, чтобы платформа понимала отправляемые сигналы и команды с ПК, а также могла предоставить необходимую информацию о состоянии платы или информацию с датчиков так, чтобы ПК мог эту информацию принять и обработать.

Пример 2.

Приведен код главной формы приложения на языке C#. (см. приложение 1)

Данный код реализует приложение для управления сервоприводом. Для начала программа считывает доступные COM-порты и, если таковых нет, выдает сообщение, что доступные порты не найдены. Если же они есть, формирует список доступных портов для дальнейшей работы. Далее запрограммированы действия при нажатии определенных кнопок и соответственно подаются сигналы на последовательный порт:

`port.Open();` - Открываем порт для связи с Arduino.

`port.Close();` - Закрываем порт – разрываем связь.

Остальные кнопки передают значения на порт:

```
port.Write("#x|");
port.Write("#w|");
port.Write(UgolServo);
```

Arduino в свою очередь считывает сигнал с порта и выполняет команды, однако для того, чтобы она эти команды понимала, разработан скетч – протокол для связи программы на C# и Arduino. (см. приложение 2)

Этот код обеспечивает правильное понимание команд платформой. В скетче прописаны действия уже на понятном для Arduino языке. Эти команды подаются в соответствии с принятыми данными с порта. То есть, принятый с порта пакет данных "#x|" обрабатывается и на выходе задано для этого пакета определенное действие. Конкретно в нашем примере выполняется команда digitalWrite(13, 1); - мы передаем пину 13 значение 1(в нашем случае включаем лампочку).

Код, описанный выше (см. приложение 2), это прошивка под конкретную задачу. Мы можем использовать ее только при условии, что мы знаем обозначения пакетов данных и какие конкретно действия выполняются.

Эти действия и пакеты мы можем запрограммировать сами, создать определенные соглашения между программами, тем самым обеспечив корректную связь между ПК и Arduino. Эти соглашения и есть протокол. Таким образом мы можем использовать любые языки и библиотеки и писать программы на основе разработанного протокола. Эти протоколы можно разрабатывать самостоятельно, однако, к счастью, существуют стандартные протоколы. Одним из таких протоколов является протокол Firmata.

1.3 Протокол и библиотека Firmata

Firmata – это базовый протокол, который предоставляет возможность для связи между микроконтроллером и ПО, работающим на компьютере. Этот протокол устанавливает формат последовательной связи, который позволяет считывать цифровые и аналоговые входы, а также отправлять информацию на цифровые и аналоговые выходы.

На основе этого протокола существует библиотека Firmata, которая уже есть в составе Arduino IDE. Чтобы использовать библиотеку, мы можем

просто открыть ее в Arduino IDE и загрузить скетч на плату. Мы можем начать управление с помощью основной программы, которая подает и принимает сигналы с порта в соответствии с протоколом.

Библиотека Firmata содержит различные функции для работы с платой. Например:

`void sendAnalog(byte pin, int value);` - отправка аналогового сообщения

`void available(void);` - проверка наличия данных в буфере

`void processInput(void);` - прочитать данные и отправить всем зарегистрированным функциям-обработчикам

Пример 3. Использование библиотеки Firmata: программа для одновременного управления сразу двумя сервоприводами.

```
#include <Firmata.h>
#include <Servo.h>
Servo servo9;
Servo servo10;
void analogWriteCallback(byte pin, int value){
    if(pin == 9)
      servo9.write(value);
    if(pin == 10)
      servo10.write(value);
}
void setup() {
    Firmata.setFirmwareVersion(0, 2);
    Firmata.attach(ANALOG_MESSAGE, analogWriteCallback);
    servo9.attach(9);
    servo10.attach(10);
    Firmata.begin(9600);
}
void loop() {
    while(Firmata.available())
        Firmata.processInput();
   }
```

Данная программа получает пакет данных, отправляемых с компьютера через последовательное соединение, и управляет сервоприводами при

помощи библиотеки Firmata. Эти самые данные, отправляемые с компьютера, должны соответствовать протоколу.

Пример 4. Применение протокола Firmata на C#: установка состояния порта.

```
byte [] buf = new byte[3];
buf[0] = 0xF4;
buf[1] = pin;
buf[2] = mode;
port.Write(buf, 0, 3);
```

Здесь `port` – экземпляр класса `Serial` для работы с COM-портом, `pin` – номер пина ардуины, `mode` – состояние порта. Методом `Write` отправляем данные на порт. С помощью `0xF4` отправляем тип сообщения – установка режима пина согласно протоколу. `mode` – одно из

```
MODE_INPUT    0x00
MODE_OUTPUT   0x01
MODE_ANALOG   0x02
MODE_PWM      0x03
MODE_SERVO    0x04
MODE_SHIFT    0x05
MODE_I2C      0x06
```

2. Реализация демонстрационной системы

Необходимо реализовать демонстрационную систему, позволяющую выполнять простейшие действия из среды разработки PascalABC.NET. На PascalABC.NET сложно писать программу для Arduino.

Реализуем требуемую систему с помощью сторонней библиотеки на языке C#. Необходима библиотека классов. Библиотека должна содержать в себе функции, позволяющие принимать данные с платы и отправлять необходимые команды на плату. Все функции библиотеки должны соответствовать протоколу.

Благодаря протоколу нам не нужно для реализации нашей задачи писать собственную прошивку, мы будем использовать универсальные функции, которые будут передавать данные в соответствии с Firmata.

Библиотеку подключим к программе на PascalABC.NET, откуда и будем использовать функции библиотеки. То есть сама программа на PascalABC.NET не будет напрямую обращаться к портам, мы будем делать это с помощью библиотеки.

2.1.Примеры простейших задач для реализации

Сформулируем несколько простых задач по управлению платой. Это необходимо, чтобы понять, какие функции следует реализовать в библиотеке.

1. Реализовать постоянное мигание светодиода. Для этого необходимо передавать на светодиод сигналы 0 и 1 с определенным интервалом.

2. Переключать состояние светодиода при каждом нажатии на кнопку. Так как каждый раз требуется менять состояние светодиода при нажатии на одну и ту же кнопку, необходимо считывать текущее состояние. Помимо этого, во избежание дребезга контактов, следует запоминать предыдущее состояние.

3. Управление степенью яркости светодиода в зависимости от степени освещенности. Определять уровень освещенности с помощью фоторезистора и менять аналоговый сигнал, передающийся на светодиод.

4. Использовать датчик звука. Считывать данные с датчика, при получении сигнала включать или выключать светодиод.

5. Управление электродвигателем. Использовать ШИМ (широтно-импульсную модуляцию) для получения изменяющегося аналогового значения посредством цифровых сигналов. Регулировать скорость с помощью потенциометра.

6. Управление сервоприводом. Он должен поворачиваться на указанное количество градусов. Также можно регулировать с помощью потенциометра.

7. Управление двумя сервоприводами с помощью джойстика. Джойстик позволяет отслеживать отклонения от нулевой точки. Получать с него данные и передавать сервоприводам.

8. Управление шаговым двигателем с помощью кнопок. При нажатии на первую кнопку, шаговый двигатель должен делать заданное количество шагов в одну сторону, при нажатии на вторую – в другую.

9. Использовать ультразвуковой датчик расстояния. Получать данные с датчика - расстояния до предметов. Можно использовать эти данные для управления моторчиком. Если расстояние больше заданного, моторчик вращается, иначе – прекращает работу.

10.Использовать датчика цвета. Принимать данные, какой цвет распознан и, в зависимости от этого, передавать команды моторчику. Например, если распознан зеленый цвет – моторчик вращается, красный – останавливается.

2.2.Анализ взаимодействия PascalABC.NET и Arduino

Перечисленные выше задачи можно реализовать в ArduinoIDE, там уже определены необходимые нам функции и методы, с их помощью легко запрограммировать плату на решение этих задач. Ранее уже рассмотрен этот способ – плата будет прошита скетчем под конкретную задачу и сможет выполнять только то, что было заложено в ее программу.

Однако, поставлена задача реализации системы, позволяющей выполнять простые действия из среды разработки PascalABC.NET. Основная программа должна быть написана на PascalABC.NET, а на выходе действия выполняет Arduino. Необходимо установить связь между этими двумя составляющими.

Для реализации требуемой связи необходимо учитывать особенности передачи данных. Ранее уже были рассмотрены протоколы, в частности,

протокол Firmata для Arduino, который идеально подойдет для решения данной задачи. Конечно, чтобы передавать данные согласно протоколу, требуется его изучить. Например, протокол Firmata для выполнения каких-либо действий требует данных, переданных в виде массивов байтов, которые подразумевают собой конкретную информацию. Каждый раз передавать такие данные, требующие детальных знаний о протоколе и особенностях передачи, довольно нелегкая задача. Целью же нашей работы является создание системы с простыми командами, без необходимости для пользователя углубляться в то, как передаются данные.

Существуют различные клиенты и библиотеки, которые решают за нас эту проблему. Используя их, нам достаточно знать функции и методы, которые они реализуют, для выполнения требуемых действий. Можно не знать, как именно эти функции работают внутри, но использовать их, зная, какие данные в них нужно передать и что они делают.

Для того, чтобы пользоваться библиотекой, даже не нужно знать, на каком языке программирования она написана. Библиотеки на одном языке программирования мы можем использовать в программах на другом языке. DLL библиотеки являются отдельно компилируемым исполняемым кодом, который подключается к вызывающей программе во время ее исполнения. Мы можем подключить dll-библиотеку, написанную на C# к программе на PascalABC.NET и использовать функции из библиотеки так, будто они написаны на одном языке.

Итак, имеем основную программу на PascalABC.NET, плату Arduino, подключенную к ПК через последовательный порт, а также dll-библиотеку, основанную на протоколе Firmata для Arduino. Но это еще не все, плата без прошивки не сможет выполнять никаких действий, ей обязательно нужна программа.

Так как для решения задачи был выбран протокол Firmata, то со стороны Arduino также необходимо настроиться на этот протокол. Существуют уже готовые прошивки под этот протокол, его просто нужно загрузить на плату. Для этого в ArduinoIDE нужно перейти в Файл > Примеры > Firmata > StandartFirmata. Данный скетч компилируется и загружается в плату.

Для использования библиотеки из PascalABC.NET требуется знать необходимые функции. Их можно использовать, прописав к ним полный путь с учетом всех пространств имен и классов. То есть, чтобы пользоваться библиотекой, пользователь должен знать ее структуру. Это делает код сложным и громоздким.

Пример 5. Использование библиотеки на C# в проекте на PascalABC.NET.

```
   var Arduino := new
Solid.Arduino.ArduinoSession(Solid.Arduino.EnhancedSerialConnect
ion.Find());
   Arduino.SetDigitalPinMode(13,Solid.Arduino.Firmata.PinMode.Dig
italOutput);
   Arduino.SetDigitalPin(13, 1);
```

В данном примере мы создаем новый объект типа `ArduinoSession`, используем функцию `EnhancedSerialConnection.Find()`, которая ищет порт, к которому и подсоединена плата Aduino Uno через USB-кабель. Далее, устанавливаем цифровой режим вывода на 13 пин и устанавливаем значение 1. При подключении к плате на 13 пин светодиода и запуске данной программы, реализуется включение светодиода.

Для упрощения работы пользователя с программой на PascalABC.NET можно отдельный блок кода, в котором будут прописаны все необходимые пути, чтобы не было нужды прописывать каждый раз полный путь при вызове какой-либо функции. Это будет некой оберткой библиотеки в более простой для понимания и написания код. На PascalABC.NET для реализации данной задачи подойдет модуль.

Модули позволяют разбивать программу на несколько самостоятельных файлов, компилируемых раздельно. В модуле мы можем описать различные объекты программы – типы, классы, переменные, процедуры и функции, которые мы сможем использовать в основной программе.

Итак, основная программа использует модуль, модуль, в свою очередь, использует dll-библиотеку, которая передает данные на порт согласно протоколу Firmata, эти данные считываются и преобразуются в определенные протоколом команды, которые Arduino выполняет. Загрузка скетча в Arduino уже рассмотрена ранее, рассмотрим подробнее подключение библиотеки, а также реализацию модуля и основной программы.

2.3.Подключение библиотеки на C# в PascalABC.NET

Для связи с Arduino используется уже существующая библиотека на C# – SolidSoils4Arduino [5] – это клиентская библиотека, построенная на .NET Framework 4.5 и обеспечивающая простой способ взаимодействия с платами Arduino. Библиотека реализует основные коммуникационные протоколы, первым из которых является протокол Firmata.

Для того, чтобы использовать данную библиотеку, необходимо получить dll-файл. Для этого нужно запустить проект Solid.Arduino. Для корректной работы с ним требуется установить NuGet пакет. После установки проект станет запускаемым. После сборки dll-файл будет лежать в папке запускаемого проекта.

Dll-файл подключается к программе, в которой требуются функции библиотеки, в нашем случае такой программой является проект на PascalABC.NET. Чтобы использовать библиотеку необходимо перенести dll-файл в папку этого проекта и к подключенным сборкам добавить его из поиска сборок на компьютере, а затем подключить:

```
{$reference 'Solid.Arduino.dll'}
```

2.4.Реализация и использование модуля на PascalABC.NET

Итак, реализация модуля – обертки для библиотеки `Solid.Arduino`. Именно здесь с самого начала подключается библиотека.

Программный код модуля открывает строка с ключевым словом unit, которая также объявляет имя модуля и оно обязано совпадать с именем файла. Назовем модуль ArduinoUno: `unit ArduinoUno;`

Здесь описываются необходимые объекты с описанием полного пути в библиотеке классов. Модуль содержит 2 раздела – интерфейс и реализация. Интерфейс содержит объявления всех имен, присутствующих в модуле, которые будут видны в других модулях и программах при подключении его к ним с помощью `uses`. В начале описываются основные типы, которые используются далее в программе:

```
///Последовательное соединение с Arduino
type ardSession = Solid.Arduino.ArduinoSession;
///Режим пина
type mode = Solid.Arduino.Firmata.PinMode;
///Состояние пина
type pinstate = Solid.Arduino.Firmata.PinState;
///Режимы пина
type mode = (DigitalOutput, DigitalInput, AnalogInput,
AnalogOutput);
///Состояния пина
type state = (HIGH, LOW);
```

Далее объявлен и описан класс для работы с Arduino. Уже здесь можно использовать типы, которые были объявлены ранее. Для этого класса и объявлены все основные необходимые функции.

```
///Класс для работы с ArduinoUno
Arduino = class
  private
    ///Последовательное соединение с Arduino
    _conn : ardSession;
    ///Номер пина
    pinNumber : integer;
    ///Процедура для установки номера пина
```

```
    procedure PinNS(index : integer);

  public
    ///Конструктор класса Arduino для установки соединения
    constructor Create();
    ///Функция для установки номера пина
    function Pin(index : integer): Arduino;
    ///Процедура для установки режима пина
    procedure SetMode(_mode: mode);
    ///Процедура для установки цифрового состояния пина
    procedure SetState(_state: state);
    ///Процедура для установки цифрового состояния пина
    procedure SetState(_state: boolean);
    ///Процедура для установки аналогового состояния пина
    procedure SetState(_state: integer);
    ///Функция для считывания режима и состояния пина
    function GetState(): pinstate;

end;
```

В следующем разделе – разделе реализации – описаны программы конструкторов, функций и методов, которые объявлены в разделе интерфейса.

Внутри этих функций используются функции внешней библиотеки, например:

```
procedure Arduino.SetMode(_mode: mode);
begin
  _conn.SetDigitalPinMode(pinNumber, _mode);
end;
```

Итак, данный модуль использует функции и методы dll-библиотеки, реализует основной функционал для работы с Arduino, позволяет нам не прописывать полный путь каждый раз и управлять платой с помощью простых команд, не углубляясь в код и принципы работы внутри программы.

Для того, чтобы использовать модуль, нужно создать новый проект. В папку проекта переносится полученный модуль – файл ArduinoUno.pas и библиотеку Solid.Arduino.dll. Подключается модуль с помощью следующей строки:

```
uses ArduinoUno;
```

Здесь, в основной программе, можно создать экземпляр класса Arduino и использовать функции и методы модуля, тем самым управляя платой.

Пример 6. Основная программа. Использование модуля на PascalABC.NET.

```
var Uno := new Arduino; //Создаем экземпляр класса Arduino. 
Производится последовательное соединение с платой.

Uno.Pin(13).SetMode(DigitalOutput); //Устанавливаем режим 13 
пина как цифровой выход
Uno.Pin(13).SetState(HIGH); //Устанавливаем на 13 пин значение 
HIGH

var state: pinstate := Uno.Pin(13).GetState(); //Считываем 
состояние и значение пина 13 и записываем в переменную state
println(state); //Печатаем результат

Uno.Pin(13).SetState(LOW); //Устанавливаем на 13 пин значение 
LOW
```

Данный пример при подключении на 13 пин светодиода реализует включение светодиода, считывание состояния и выключение светодиода.

Таким образом, получилось реализовать управление платой Arduino Uno из PascalABC.NET.

Пример 7. Основная программа. Использование модуля на PascalABC.NET.

Необходимо реализовать решение задачи 1 из пункта 2.1: мигание светодиода с заданным интервалом. Для примера задан интервал 1 секунда.

```
repeat
Uno.Pin(13).SetState(true);
delay(1000);
Uno.Pin(13).SetState(false);
delay(1000);
until KeyPressed;
```

Непрерывное повторение реализовано с помощью оператора цикла repeat-until. Условием выхода является нажатие любой клавиши на клавиатуре. Пока клавиша не нажата, светодиод непрерывно мигает с заданным функцией delay(1000) интервалом, аргументом является количество миллисекунд.

Пример 8. Основная программа. Использование модуля на PascalABC.NET. Управление сервоприводом.

Реализуется поворот сервопривода на 360 градусов и обратно. Также циклично, как в примере 7.

```
repeat
Uno.Pin(11).SetState(360);
delay(500);
Uno.Pin(11).SetState(0);
delay(500);
until KeyPressed;
```

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены способы взаимодействия программ и приложений с платформой Arduino, в особенности, связь посредством COM-портов поверх USB.

Изучены протоколы передачи данных и их назначение. Рассмотрен протокол взаимодействия с Arduino Firmata и примеры его использования.

Разработан модуль на PascalABC.NET для управления состоянием платы Arduino с использованием сторонней dll-библиотеки на языке C#. Полученная демонстрационная система позволяет выполнять простейшие действия из PascalABC.NET.

СПИСОК ЛИТЕРАТУРЫ

[1] Arduino official site. – URL: https://www.arduino.cc

[2] Абрамян М. Э. Visual C# на примерах. – СПб.: БХВ-Петербург, 2008. – 482 с.

[3] Осипов А. В. PascalABC.NET: Введение в современное программирование. – Ростов-на-Дону, 2019 – 572с.

[4] Firmata Protocol Documentation. – URL: https://github.com/firmata/protocol

[5] SolidSoils4Arduino. – URL: https://github.com/SolidSoils/Arduino

ПРИЛОЖЕНИЯ

Приложение 1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
namespace WindowsServoTester{
    public partial class Form1 : Form{
        bool isConnected = false;
        SerialPort port;
        public Form1(){
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e){
            comboBox1.Items.Clear();
            // Получаем список COM портов доступных в системе
            string[] portnames = SerialPort.GetPortNames();
            // Проверяем есть ли доступные
            if (portnames.Length == 0){
                MessageBox.Show("COM PORT not found");
            }
            foreach (string s in portnames){
                //добавляем доступные COM порты в
список
                comboBox1.Items.Add(s);
            }
        }
        private void connectToArduino(){
            isConnected = true;
            string
selectedPort = comboBox1.GetItemText(comboBox1.SelectedItem);
            port = new SerialPort(selectedPort, 9600,
Parity.None, 8, StopBits.One);
            port.Open();
            button2.Text = "Disconnect";
        }
        private void disconnectFromArduino(){
            isConnected = false;
            port.Close();
            button2.Text = "Connect";
        }
        private void button2_Click(object sender, EventArgs e){
            if (!isConnected){
                connectToArduino();
            }
```

```
            else{
                disconnectFromArduino();
            }
        }
        private void button3_Click(object sender, EventArgs e){
            port.Write("#x|");
        }
        private void button4_Click(object sender, EventArgs e){
            port.Write("#w|");
        }
        private void trackBar1_Scroll(object sender, EventArgs
e){
            String UgolServo = "#A";
            UgolServo = UgolServo + Convert.ToString(trackBar1.
Value, 16);
            UgolServo = UgolServo + "|";
            port.Write(UgolServo);
            label1.Text = String.Format("Угол сервопривода:
{0}", trackBar1.Value);
        }
    }
}
```

Приложение 2

```
const char StaPack = '#';
const char EndPack = '|';
const uint16_t TimeOut = 500;
#include <Servo.h>
Servo servo;
void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  servo.attach(10);
  servo.write(0);
}
byte hexToByte (String StrControlHex) {
  uint8_t  HEX16 = 0;
  uint8_t  exp16 = 1;
  uint8_t  decBy = 0;
  StrControlHex.remove(0, 2);
  int i = StrControlHex.indexOf('|');
  if (i == -1) return 0;
  if (i > 2)   return 0;
  StrControlHex.remove(i, 1);
  for (int j = StrControlHex.length() - 1; j >= 0; j--) {
    HEX16 = StrControlHex.charAt(j);
    if (HEX16 >= 48 && HEX16 <= 57) HEX16 = map(HEX16, 48, 57,
0, 9);
    if (HEX16 >= 65 && HEX16 <= 70) HEX16 = map(HEX16, 65, 70,
10, 15);
    if (HEX16 >= 97 && HEX16 <= 102) HEX16 = map(HEX16, 97, 102
```

```
, 10, 15);
    decBy = decBy + HEX16 * exp16;
    exp16 = exp16 * 16;
  }
  return decBy;
}
void loop() {
clearPack:
  char   IncomChar;
  String StrControl = "";
  while (Serial.available() > 0) {
    IncomChar = Serial.read();
    if (IncomChar == StaPack) {
      StrControl += IncomChar;
      unsigned long currentTime = millis();
      ReceptionPacket:
      if (Serial.available() > 0) {
        IncomChar = Serial.read();
        StrControl += IncomChar;
        if (IncomChar == EndPack) break;
      }
      if (millis() - currentTime >= TimeOut) goto clearPack;
      goto ReceptionPacket;
    }
  }
  if (StrControl != "") {
  //Serial.println(StrControl);
  switch (StrControl.charAt(1)) {
    case 'w':
      digitalWrite(13, 0);
      break;
    case 'x':
      digitalWrite(13, 1);
      break;
    case 'A':
      servo.write(hexToByte(StrControl));
      break;
  }
  }
}
```