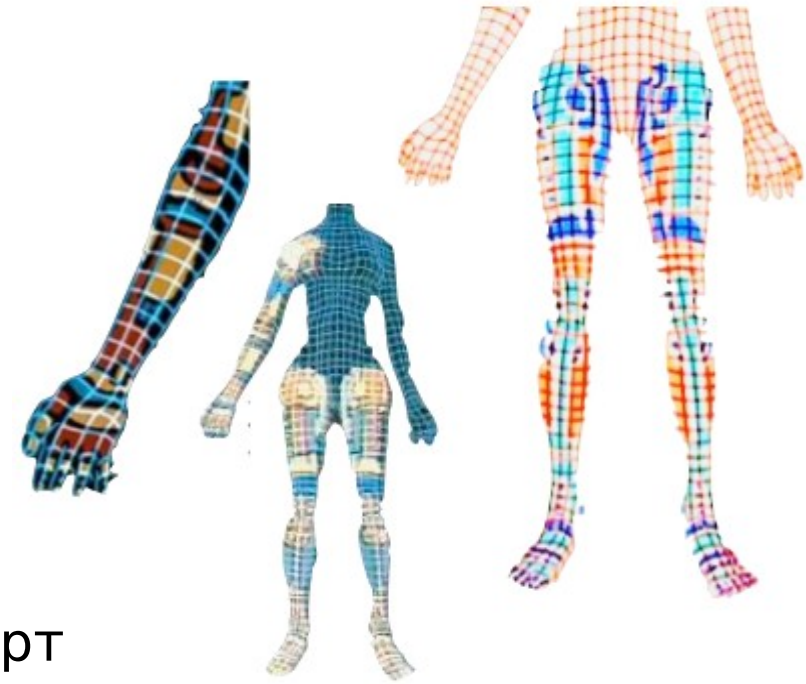
Arduino и бионика

Введение в микроконтроллеры с Arduino



Занятие 4

20 ноября 2007 - machineproject — Тод Е. Курт

Перевод на русский язык
13 апреля 2012 — robofreak.ru — Татьяна Волкова

Программа на сегодня

- Про ШИМ
- Управление сервомашинками
- Про шину I2C
- Использование I2C с Arduino
- Про акселерометры
- Нунчак Nintendo Wii как устройство ввода

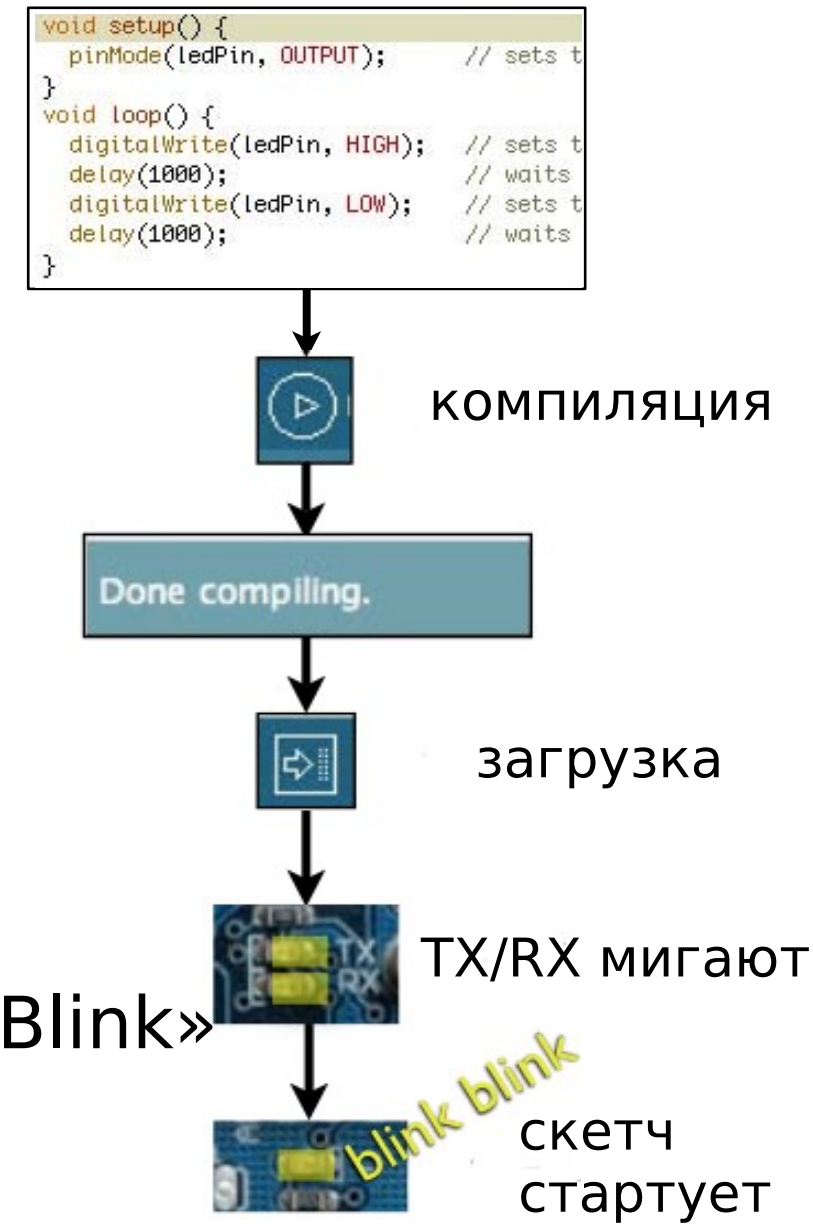
Повторение: мигающий светодиод

Удостоверьтесь, что всё по-прежнему работает

```
int ledPin = 13;                  // LED connected to digital pin 13

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```



Загрузите «File/Sketchbook/Examples/Digital/Blink»

Измените значение в «delay()», чтобы изменить частоту мигания

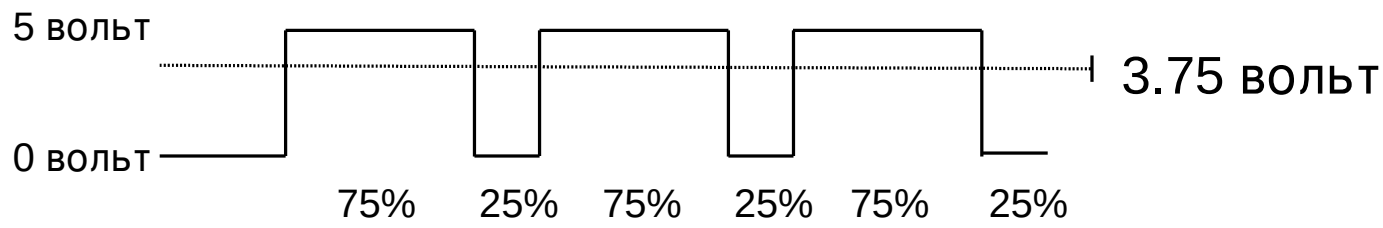
Широтно-импульсная модуляция

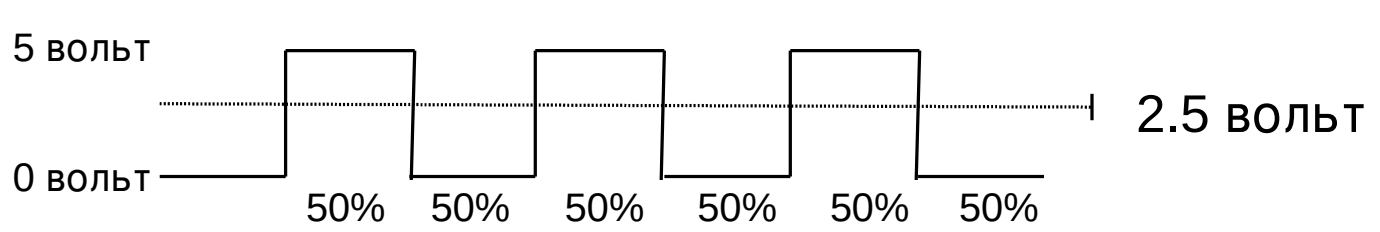
- Часто называется просто «ШИМ»
- Компьютеры не могут выдавать аналоговое напряжение
  - Только цифровое (0 вольт либо 5 вольт)
- Но вы можете его изобразить
  - если усредните цифровой сигнал, скачущий между напряжениями
- Например...

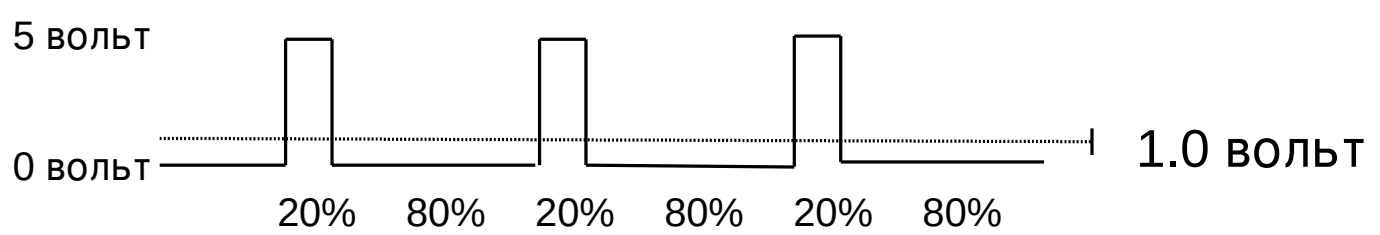
ШИМ

Выходное напряжение — среднее между временем включения и выключения

выходное напряжение = (время_вкл / время_выкл) * максимальное_напряжение







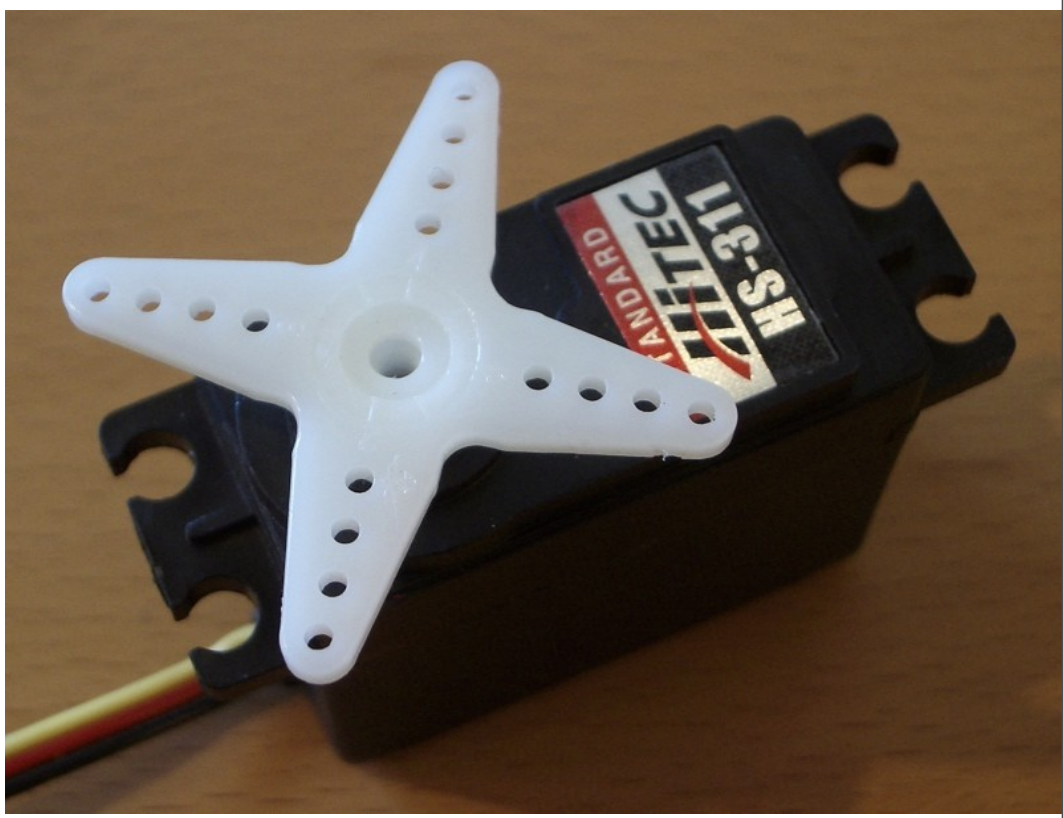
ШИМ

- Используется везде
  - Регуляторы яркости ламп, управление скоростью мотора, подача питания, создание звука
- Три характеристики ШИМ-сигнала
  - Ширина импульса (мин/макс)
  - Период импульса (= 1/число импульсов в сек.)
  - Уровни напряжения (к примеру, 0-5В)

Вы уже делали схемы с использованием ШИМ.

Сервомоторы

- Могут принимать положение от 0 до 180º (обычно)
- Внутренняя схема с обратной связью и редуктором сама делает всё сложное за Вас
- Простой интерфейс ШИМ 5В из 3-х проводов



Точнее, это сервомашинки, используемые в авиамоделировании, энтузиастами в качестве хобби.
Вообще говоря, «сервомотор» - это мотор с внутренним механизмом обратной связи, который позволяет отправлять команды «принять позицию» без необходимости Вам считывать текущую позицию.

Сервомашинки прекрасны

- Мотор постоянного тока
- Редуктор с большим моментом
- Потенциометр, считывающий позицию
- Схема с обратной связью считывает потенциометр и управляет мотором
- Всё встроено, Вам только нужно скормить ей ШИМ-сигнал

У таких небольших синих сервомашинок можно посмотреть на внутренности.

Для чего хороши сервы?

- Они в ходу у робототехников, авторов киноспецэффектов и марионеточников
- Везде, где Вам нужно управляемое, повторяемое движение.
- Можно превратить вращение в линейное движение при помощи хитрых механических рычагов.

Даже в одежде сейчас используются сервы:
http://www.technologyreview.com/read_article.aspx?id=17639&ch=infotech:

Сервомашинки

- Бывают всех размеров
  - От сверх-миниатюрных
  - до двигающих-машину
- Но у всех один и тот же 3-проводной интерфейс
- Сервы определяются через:

Вес: 9 г
Скорость: 12с/60° @ 6В
Момент: 1.5кг/см @ 6В
Напряжение: 4.6~6В
Размер: 21x11x28 мм

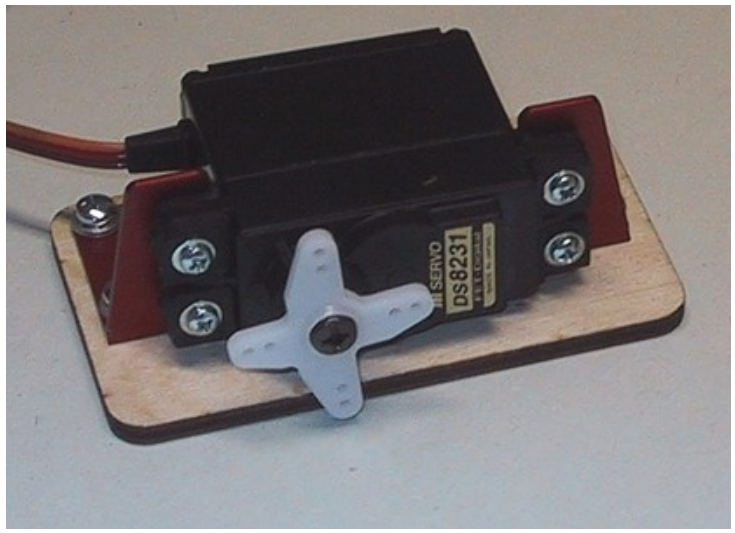
9 г

157 г

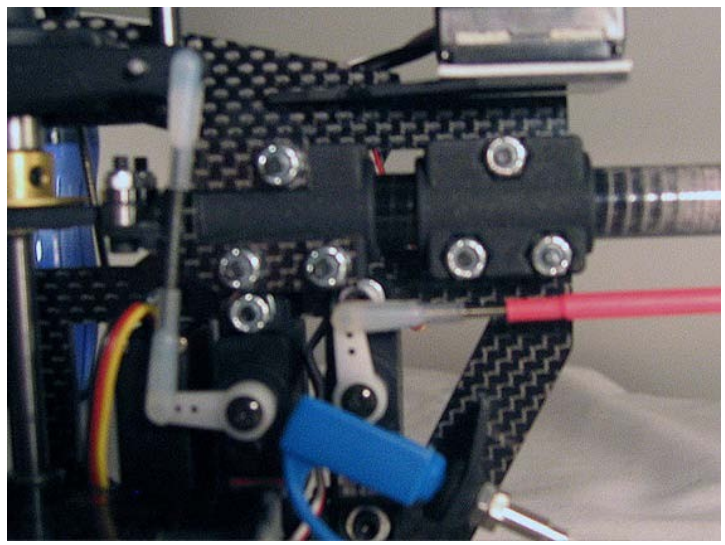
http://rctoys.com/
http://hobbypeople.net/

Крепление и удлинение сервы

Множество способов прикрепить серву

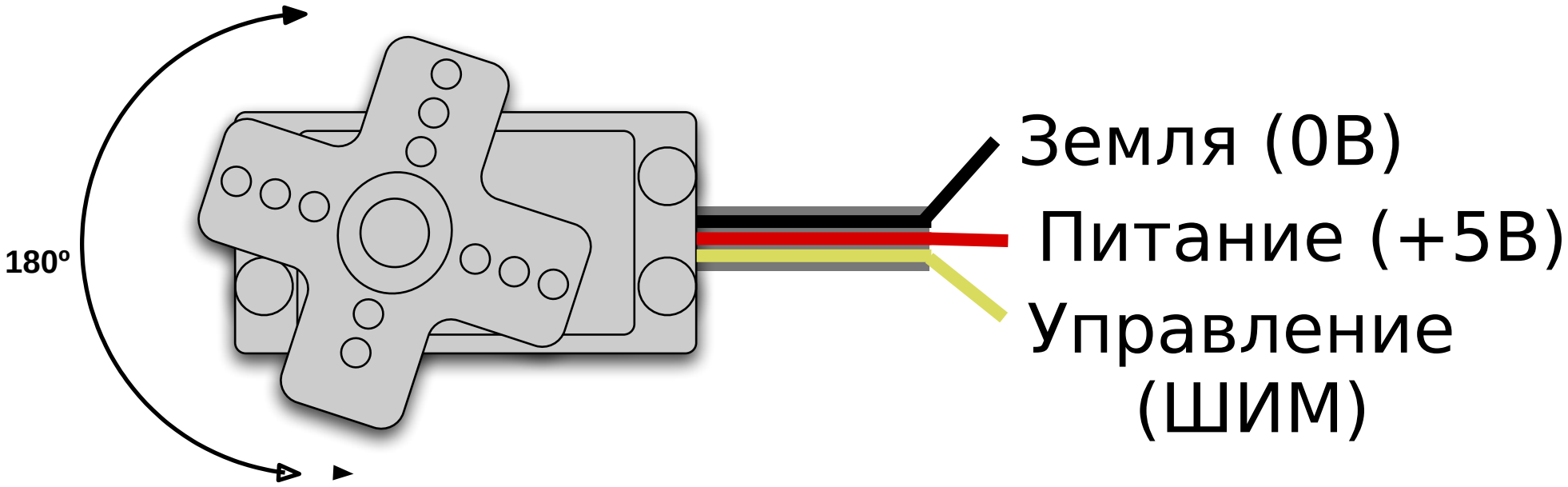


И превратить её вращательное движение в другие типы движения



Скоба для крепления: http://www.sierragiant.com/prod28.html

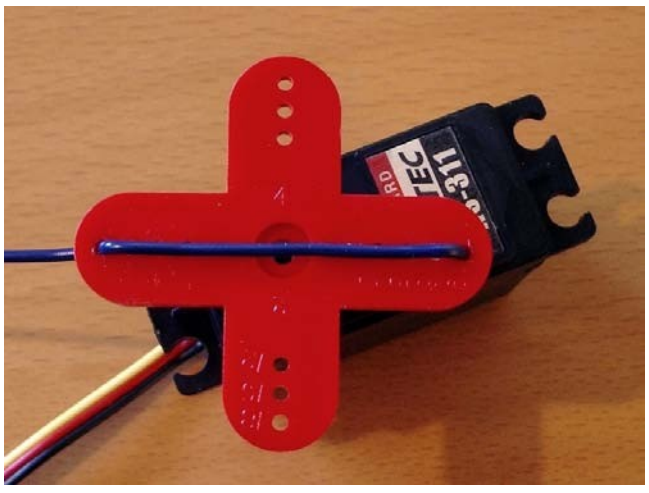
Управление сервой



- Частота ШИМ - 50 Гц (каждые 20 мс)
- Ширина импульса от 1 до 2 миллисекунд
  - 1 мс = крайняя позиция против часовой
  - 2 мс = крайняя позиция по часовой

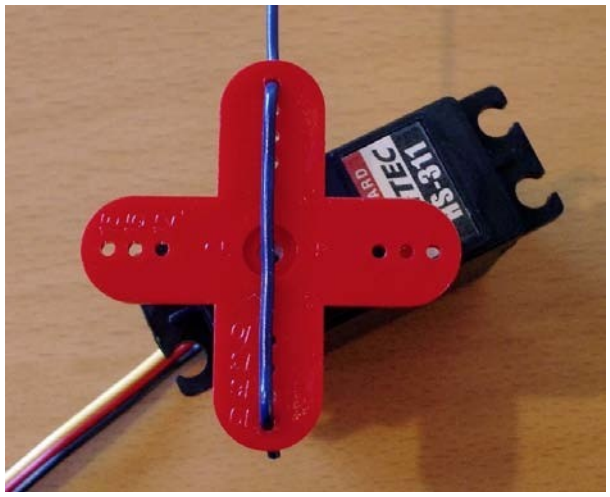
Движение сервы

0 градусов



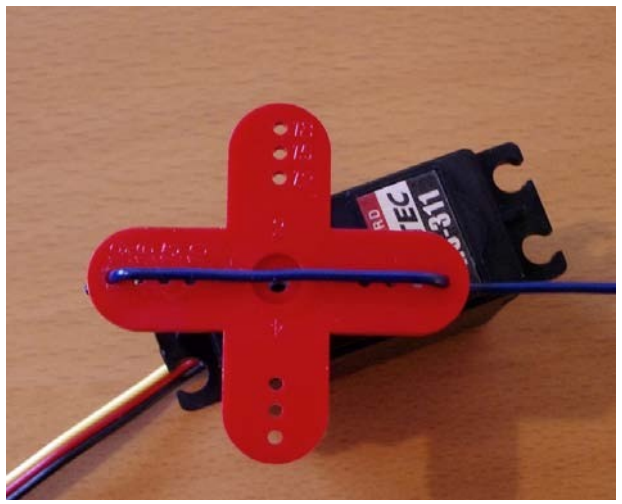
1000 микросекунд

90 градусов



1500 микросекунд

180 градусов



2000 микросекунд

На практике ширина импульса может быть от 500 до 2500 микросекунд

(добавим метку из провода к серве, как на картинке выше)

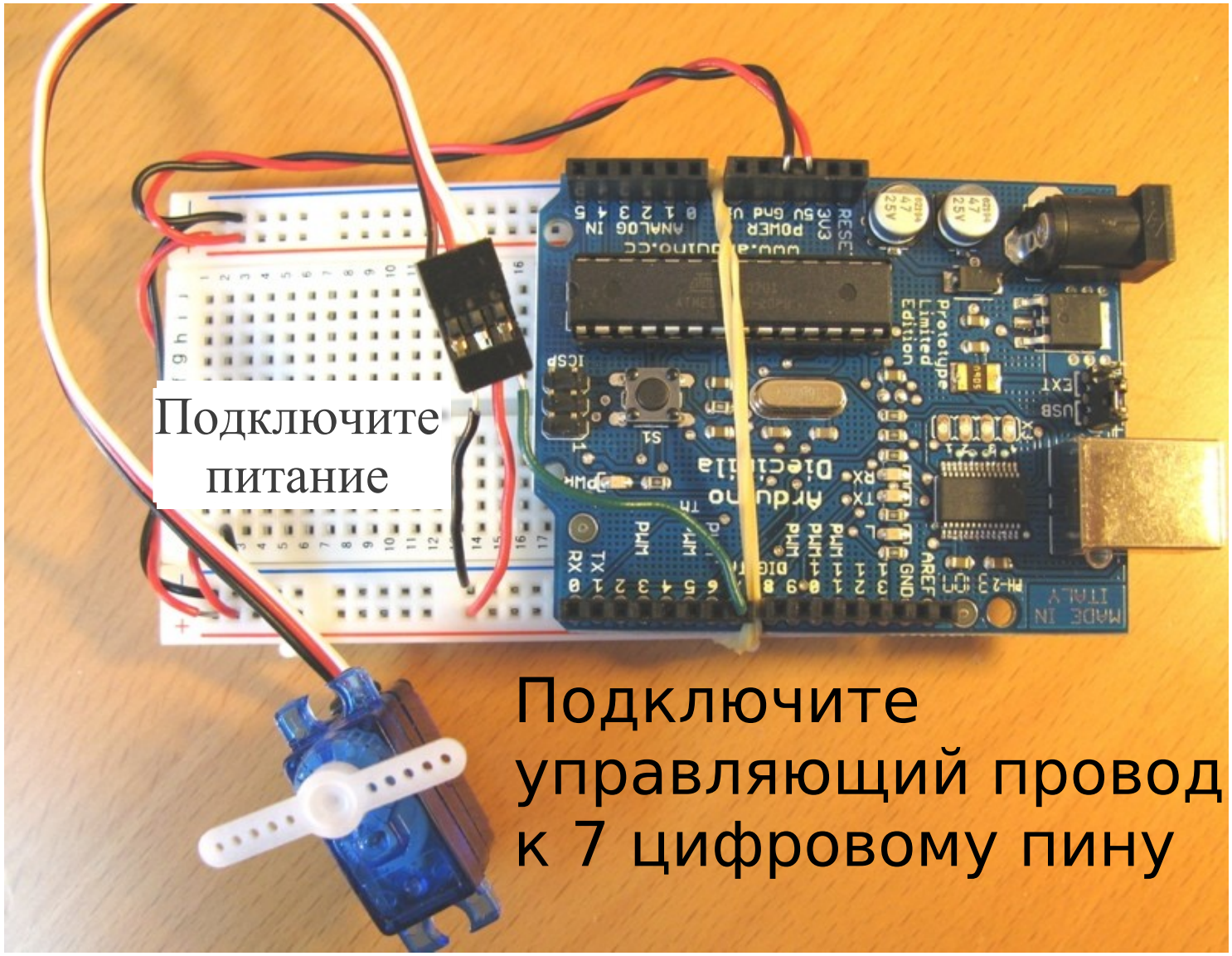
Поставьте красную «руку» на Вашу серву. Для этого нужна отвёртка.
Многие покупные серводрайверы имеют калибровку, чтобы справляться с различными сервами.

Серва и Arduino

Добавьте несколько проводов к разъёму сервы

Советую подбирать цвета проводов согласно общепринятой маркировке, насколько это возможно.

Серва и Arduino



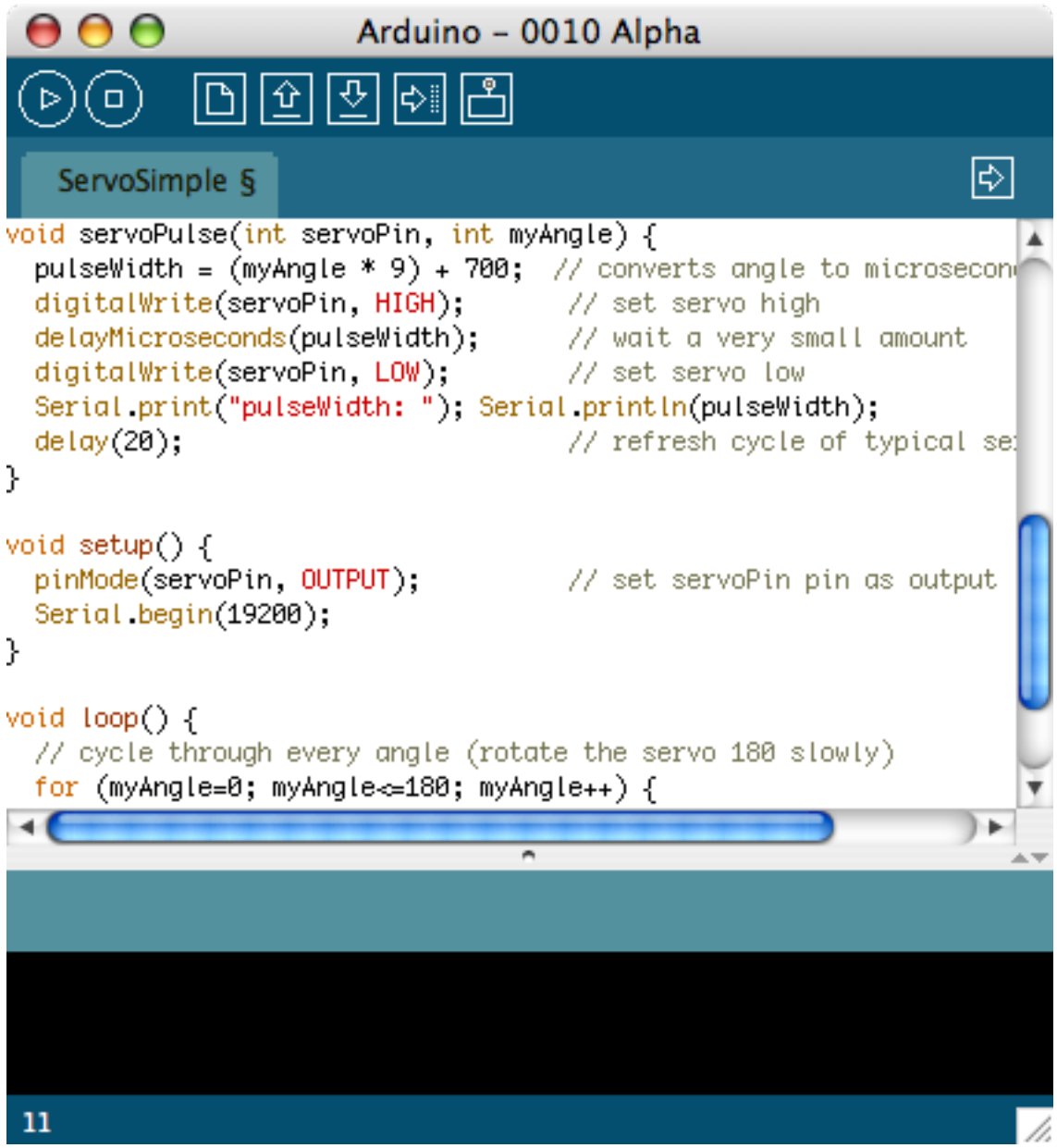
Двигаем сервомашинку

“ServoSimple”

Перемещает серво-машинку в доступных ей пределах

Функцией
`delayMicroseconds()`
задаётся ширина импульса

Функцией
`delay()`
задаётся частота импульса



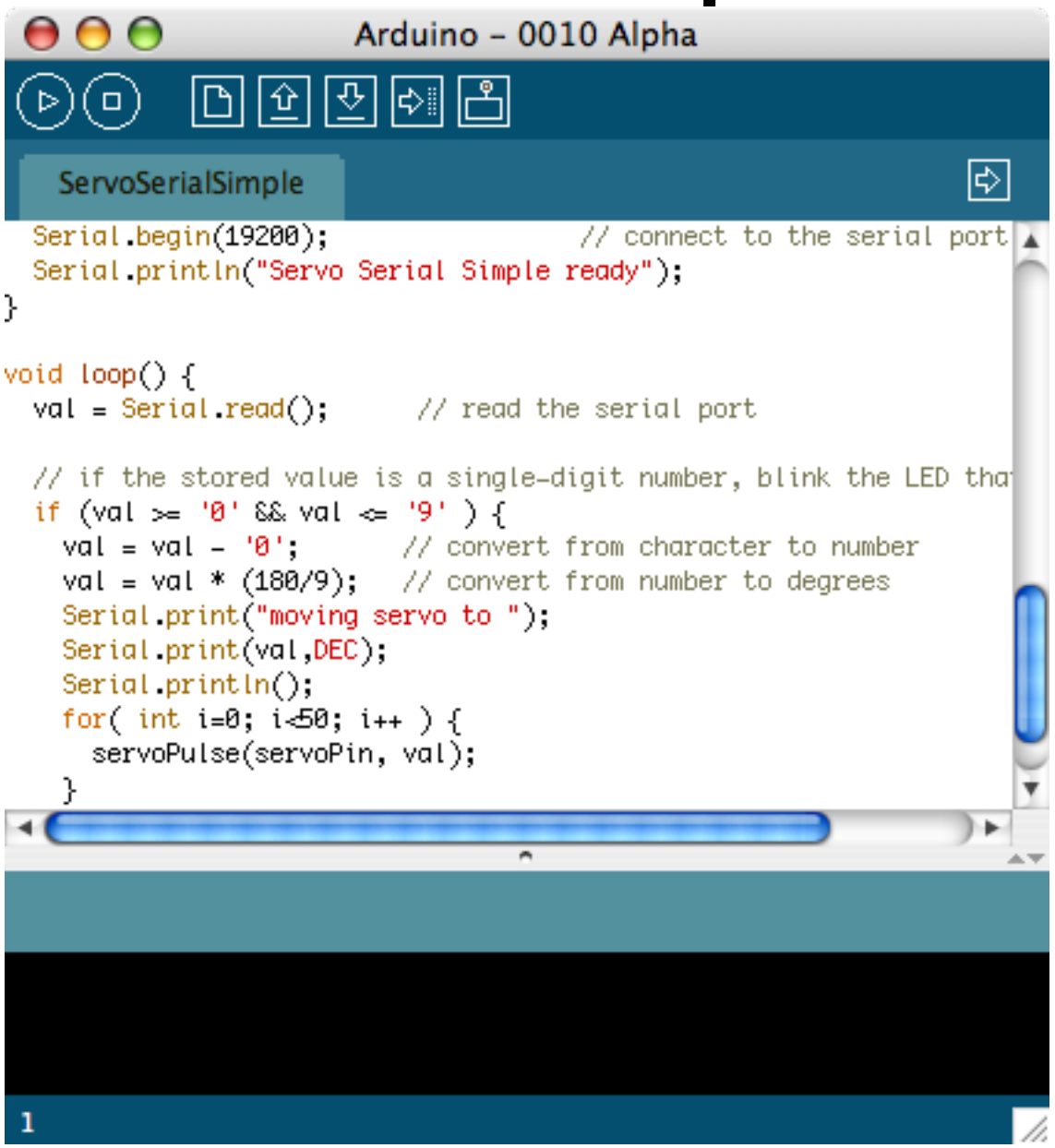
Скетч есть в раздаточном материале
Написана специальная функция для выдачи импульсов на сервомашинку.
Новая функция “delayMicroseconds()”. Как “delay()”, но задержка в микросекундах (µs), а не миллисекундах (ms).
На самом деле, просто ждать 20 миллисекунд не вполне правильно. Должно быть:
20 — (ширина_импульса/1000)
(1000 микросекунд = 1 миллисекунда, и 1000 миллисекунд = 1 секунда)

Серва, управляемая через последовательный порт

“ServoSerialSimple”

Управляйте сервомашинкой, нажимая на клавиши с цифрами.

Последний пример с сервомашинкой, к которому добавлен стандартный последовательный ввод



Скетч есть в раздаточном материале.
Зачем этот цикл «for»? Потому что сервомашинке требуется время, чтобы перейти на нужную позицию, а памяти у неё нет.

Шаг в сторону: управление Arduino

- Любая программа на компьютере, не только программы для Arduino, могут управлять платой Arduino
- В Unix-системах, таких как Mac OS X и Linux, можно даже из командной строки:

```
demo% export PORT=/dev/tty.usbserial-A3000Xv0
demo% stty -f $PORT 9600 raw -parenb -parodd cs8 -hupcl -cstopb clocal
demo% printf "1" > $PORT   # rotate servo left
demo% printf "5" > $PORT   # go to middle
demo% printf "9" > $PORT   # rotate servo right
```

Unix — это нереально круто.

Идея роботизированной игрушки для кошки

Удостоверьтесь, что надёжно закрепили сервомашинку перед тестовыми запусками. У кошек очень хорошо получается разбирать электронные прототипы.

Проблемы со временем

- Две проблемы с последним скетчем
  - Когда выполняется `servoPulse()`, ничто другое выполняться не может
  - Серве не выдаются периодические импульсы для удержания позиции
- Нужно выполнять две разных “задачи”:
  - Считывание с последовательного порта
  - Управление сервомашинкой

Если серве не говорят постоянно, что делать, она расслабляется и не поднимает/толкает/тянет.

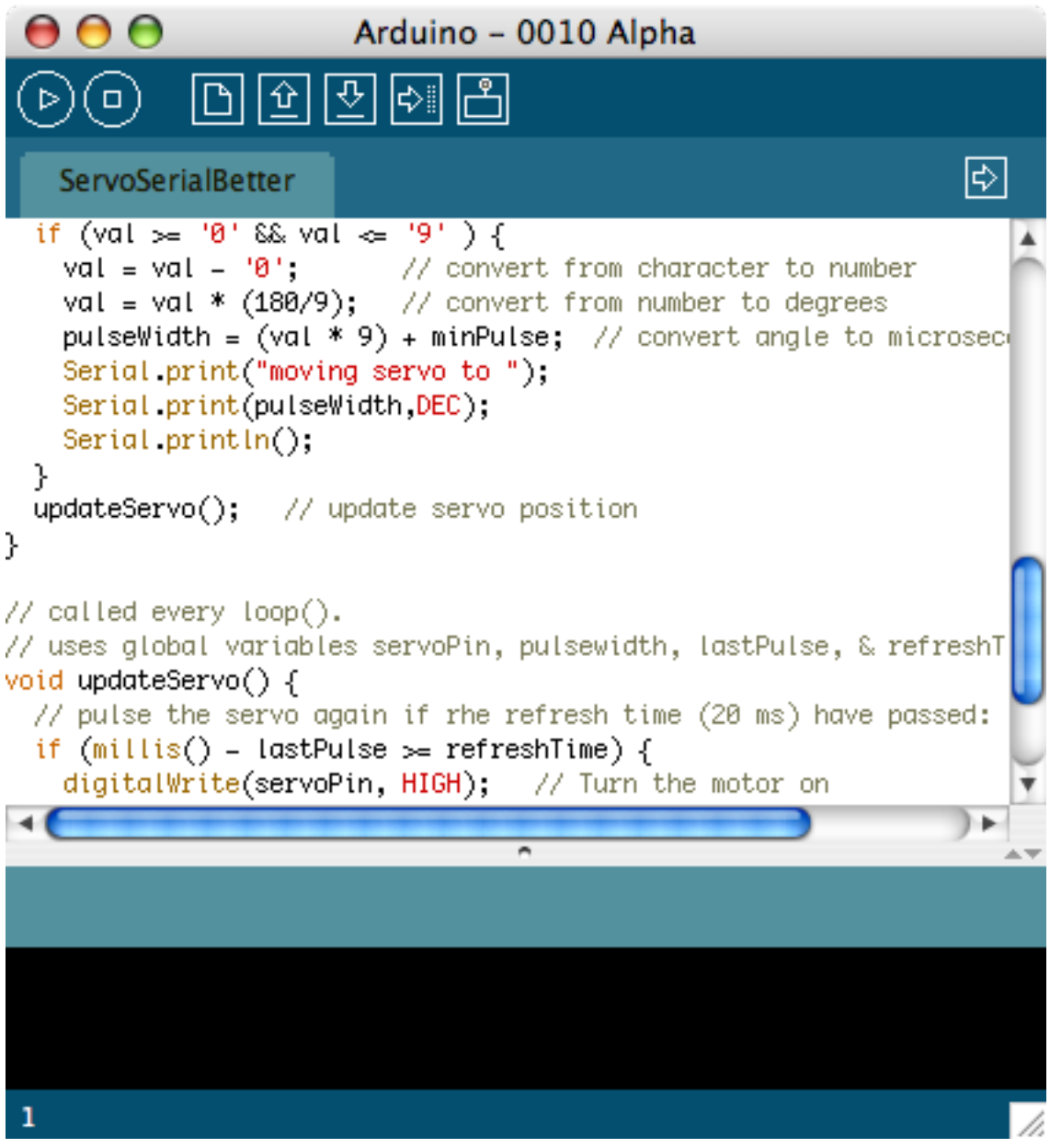
Более продвинутый вариант управления сервой

“ServoSerialBetter”

Работает как
ServoSerialSimple
(но лучше)

Обновляет позицию сервы, когда нужно, а не просто в момент вызова.

Вызывает “millis()”, чтобы узнать время



Скетч в раздаточном материале.
Жертвует временем (дополнительные переменные) ради более удобной логики.
Можно вызывать updateServo() как угодно часто, серва перемещается только тогда, когда это необходимо.

Несколько сервомашинок

- Технология `updateServo()` может применяться к нескольким сервам
- Единственное ограничение — число свободных цифровых выходов
- Проблемы начинаются, когда количество сервомашинок превышает 8

«Многозадачность»

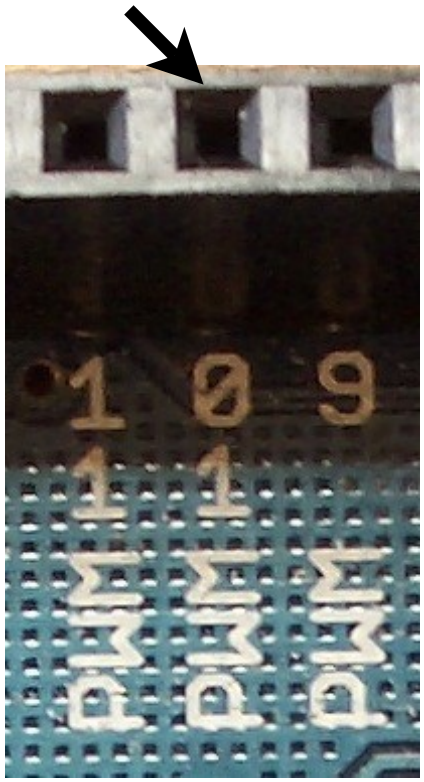
Концепция `updateServo()` полезна, когда нужно «делать много дел одновременно» в скетче Arduino:

- Определите задачу
- Разбейте её на несколько частей, основываясь на времени выполнения («кванты времени»)
- Поместите эти участки кода в отдельные функции
- С помощью `millis()` определяйте, какой участок должен сейчас выполняться
- Вызывайте функции из `loop()`

Внутри участков кода избегайте использования delay(), циклов for и других языковых конструкций, которые заставляют код выполняться слишком долго внутри кванта времени. Это называется “кооперативная многозадачность”, и так работали ОС в 80-е.

ШИМ в Arduino

почему всё программно, разве в Arduino нет ШИМ?



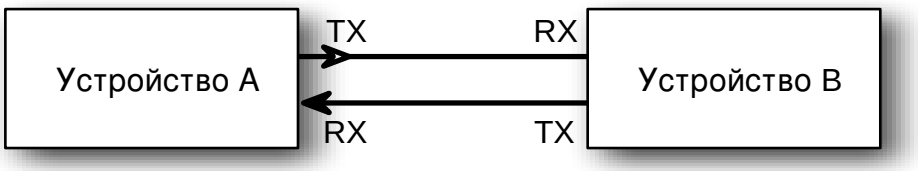
- В Arduino есть встроенный ШИМ
- На выходах 9, 10, 11
- Используйте `analogWrite(pin,value)`
- Работает на высокой, фиксированной частоте (поэтому не годен для серв)
- Но отлично подходит для светодиодов и моторов
- Использует встроенные электрические цепи ШИМ чипа ATmega8 -» не нужно дополнительного программирования

Частота ШИМ, используемая в analogWrite() в настоящий момент по умолчанию равна 450Гц или 30 кГц. Я точно не помню, но её нельзя изменить, пока Вы не поймёте на более глубоком уровне работу AVR.
Так что при программировании AVR-чипов вне среды Arduino, скорость ШИМ может быть назначена практически любой.

Перерыв

Последовательная коммуникация

Асинхронная коммуникация

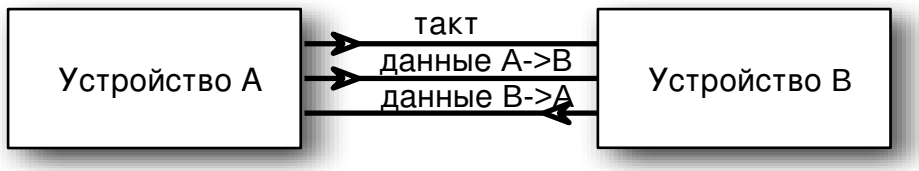


асинхронный – без тактирования
Данные передаются установкой уровней HIGH/LOW в нужное время

Разные провода для передачи и приёма

Оба устройства должны иметь устойчивый «ритм»

Синхронная коммуникация



синхронный – с тактированием
Данные передаются установкой HIGH/LOW, когда изменяется тактовый сигнал

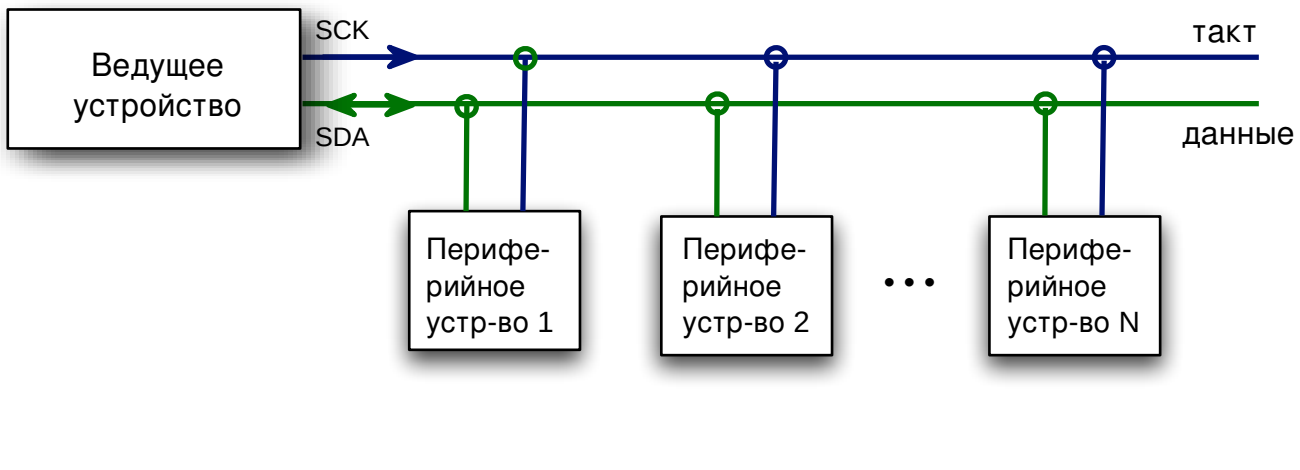
Один провод для тактирования и провод для каждого направления, передачи данных, как и раньше

Ни одному из устройств не нужен устойчивый ритм, но одно из них является дирижёром

Какой из них лучше? Зависит от конкретного приложения. Асинхронный хорош, когда имеются только два устройства, и они заранее настроены так, чтобы быть согласованными по скорости (как в ваших скетчах Arduino).
Синхронный в целом лучше для высоких скоростей (потому что не требуется точное тактирование, достаточно лишь следить за тактовым проводом).

I2C, также известная как «Два провода» (Two-wire)

Синхронная последовательная шина
с общей линией данных
маленькая сеть для Ваших устройств

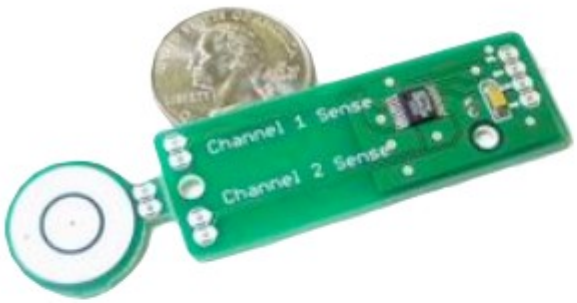


- До 127 устройств на одной шине
- Скорость передачи данных до 1Мб/сек
- Очень простой протокол (по сравнению с USB,Ethernet, и т.д)
- Встроен во многие микроконтроллеры

Общая линия данных означает, что устройства должны соглашаться о том, когда их очередь «говорить». Как в радиосвязи гражданского диапазона (CB), когда Вы говорите «Приём», чтобы показать, что Вы закончили говорить и готовы слушать собеседника.

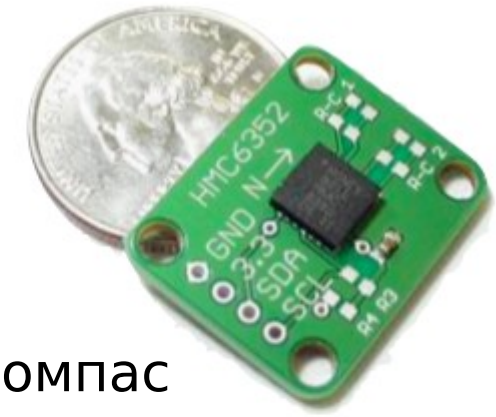
См. “Введение в I2C”: http://www.embedded.com/story/OEG20010718S0073
“I2C” расшифровывается как “Inter-Integrated Circuit”, но никто её так не называет
И если Ваш микроконтроллер не имеет встроенной аппаратной поддержки I2C, можно запрограммировать её вручную
(это всё равно приходится делать для управляющих устройств)

Множество устройств с I2C



датчик
касания

энергонезависимая
память



компас

FM-передатчик

ЖК-дисплей

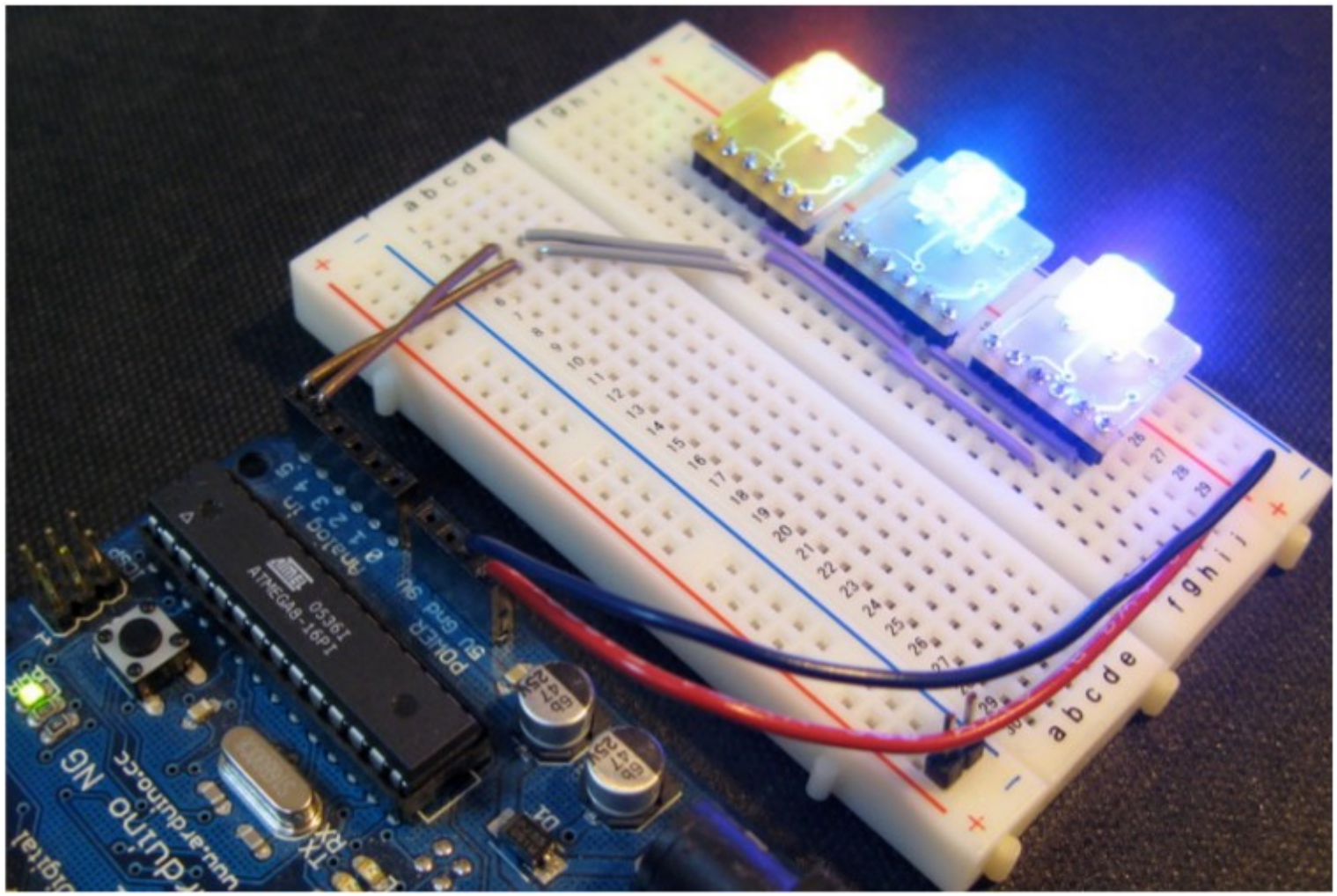
И многие другие
(гироскопы,клавиатуры,
моторы,...)

датчик
температуры
и влажности

Изображения со Sparkfun.com, кроме ЖК-экрана с matrixorbital.com

Обязательная реклама BlinkM

Умный светодиод с I2C



Сам делает всю сложную работу по генерации ШИМ

Это можно купить со Sparkfun.com.

Нунчак Nintendo Wii

- Стандартный интерфейс I2C
- 3-осевой акселерометр
  с 10-битным разрешением
- 2-осевой аналоговый джойстик
  с 8-битным А/Ц преобразователем
- 2 кнопки
- $20

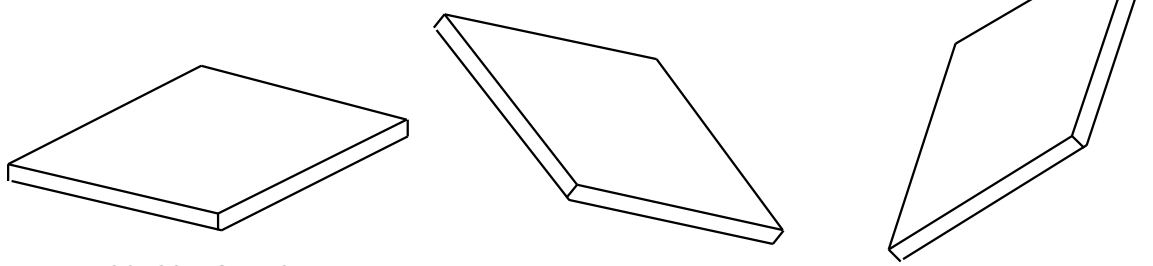
Если Вы взглянете на архитектуру Nintendo Wii и периферийные устройства, Вы увидите нехарактерное для Nintendo следование стандартам. Wii-контроллер - самый яркий пример. Подключение к нему осуществляется через стандартную шину I2C.
Дистанционные Wii-устройства общаются через Bluetooth HID с приставкой Wii (или Mac, или PC).
Нунчак легко соединить с Arduino, Basic Stamp и с большинством других микроконтроллеров, так как он использует стандартный интерфейс I2C.

См: http://www.wiili.org/index.php/Wiimote/Extension_Controllers/Nunchuk
и: http://www.windmeadow.com/node/42
и: http://todbot.com/blog/2007/10/25/boarduino-wii-nunchuck-servo/

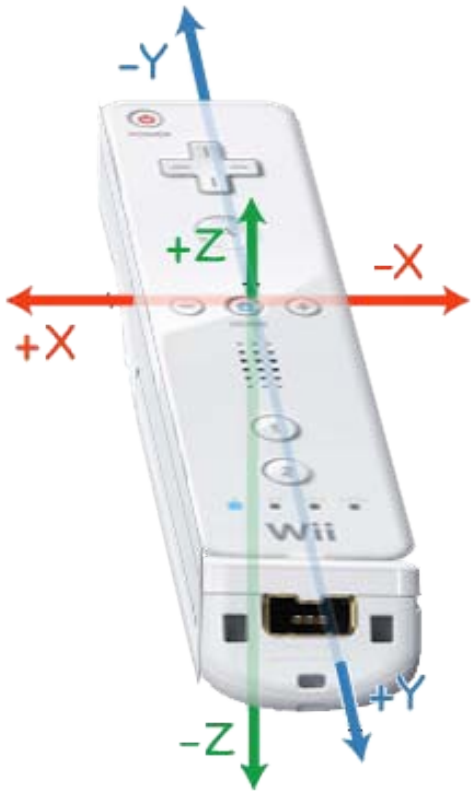
Обратите внимание на Wii Remote - кроме Bluetooth HID-интерфейса, в нём есть ещё акселерометры, кнопки, динамик, память, и он является ведущим (master) в шине I2C.

Акселерометр?

- Измеряет ускорение (изменение скорости)
- Как в машине, когда вас вдавливает в кресло
- Сила притяжения — тоже ускорение
- Поэтому, в том числе, измеряет наклон



Акселерометр в Нунчаке



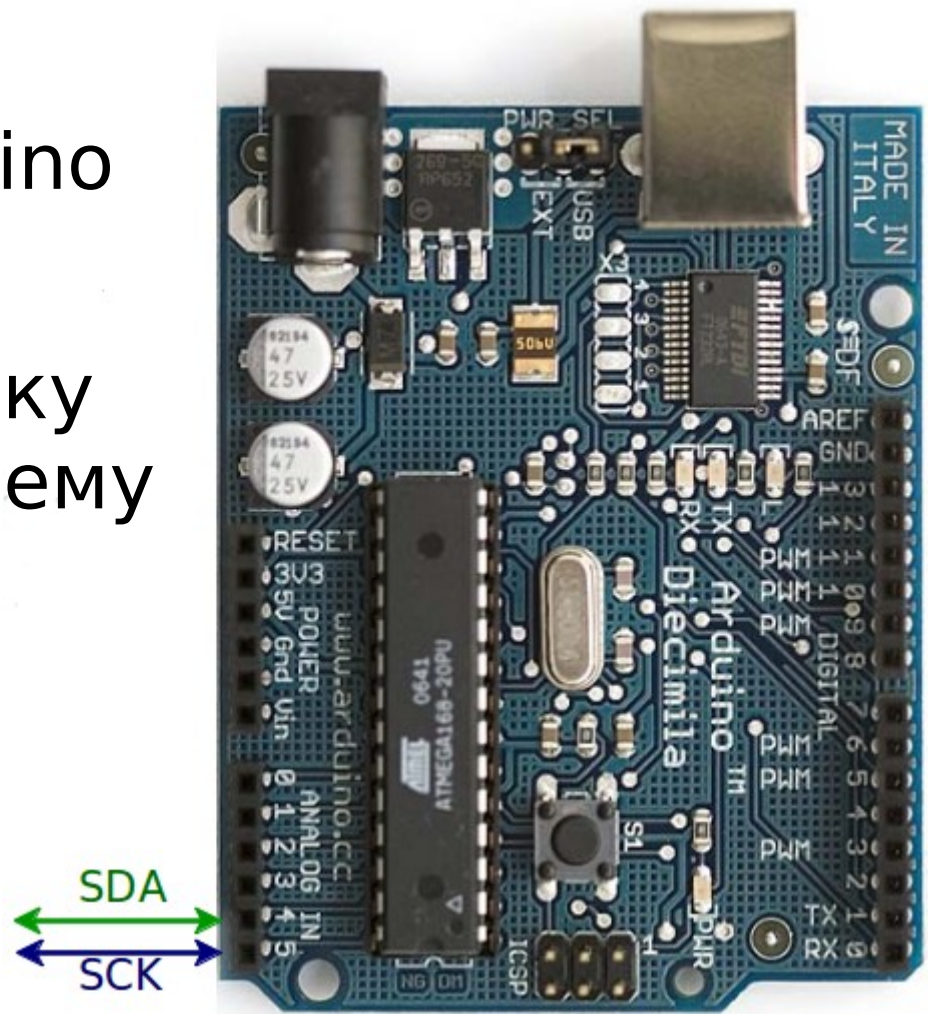
Оси акселерометра
Wii-пульта и Нунчака

Я не уверен, что правильно изобразил оси Нунчака..

Изображение осей Wiimote с сайта: http://www.wiili.org/index.php/Wiimote

I2C в Arduino

- I2C встроен в чип Arduino ATmega168
- Используйте библиотеку “Wire” для доступа к нему
- Аналоговый вход 4 — сигнал SDA
- Аналоговый вход 5 — сигнал SCK



Библиотека Arduino “Wire”

Отправление данных

Загрузить библиотеку Wire

Подключиться к шине I2C (как ведущий (master))

Начать посылку

Послать данные

Закончить посылку

```
#include <Wire.h>

void setup() {
  Wire.begin(); // join i2c bus (address optional for master)
}

byte x = 0;

void loop() {
  Wire.beginTransmission(4); // transmit to device #4
  Wire.send("x is ");        // sends five bytes
  Wire.send(x);              // sends one byte
  Wire.endTransmission();    // stop transmitting

  x++;
  delay(500);
}
```

О том, что делают разные команды, можно прочитать в инструкции/документации к конкретному устройству.

Библиотека Arduino “Wire”

Получение данных

Подключиться к шине I2C (как ведущий (master))

Запросить данные с устройства

Получить данные

```
#include <Wire.h>

void setup() {
  Wire.begin();        // join i2c bus (address optional for master)
  Serial.begin(9600);  // start serial for output
}

void loop() {
  Wire.requestFrom(2, 6);    // request 6 bytes from slave device #2

  while(Wire.available()) {   // slave may send less than requested
    char c = Wire.receive(); // receive a byte as character
    Serial.print(c);         // print the character
  }

  delay(500);
}
```

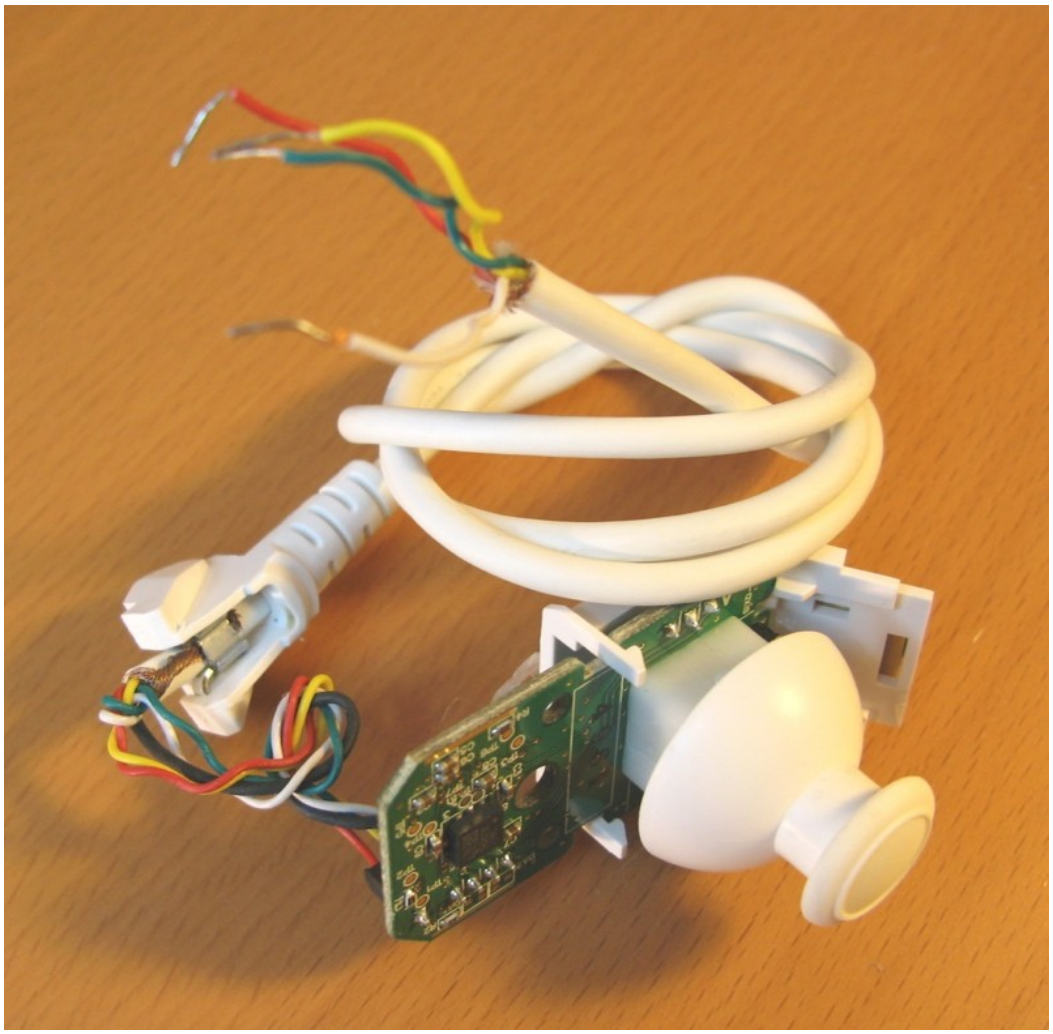
Типы возможных взаимодействий зависят от устройства, с которым Вы общаетесь

У большинства устройств есть несколько “команд”

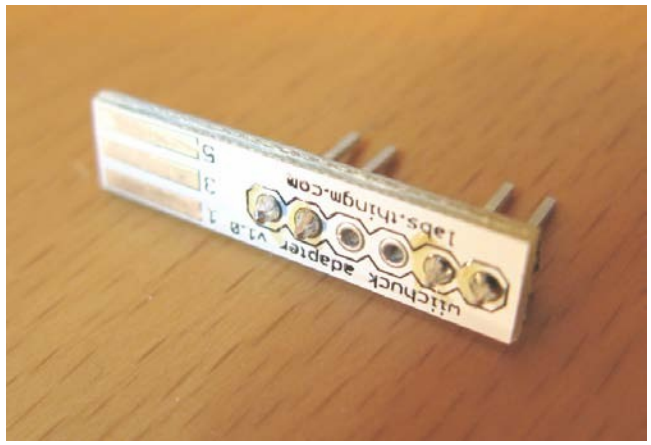
А о том, что делают разные команды, можно прочитать в инструкции/документации к конкретному устройству.

Подключение Нунчака

Мы можем отрезать разъём и подключиться непосредственно к проводам

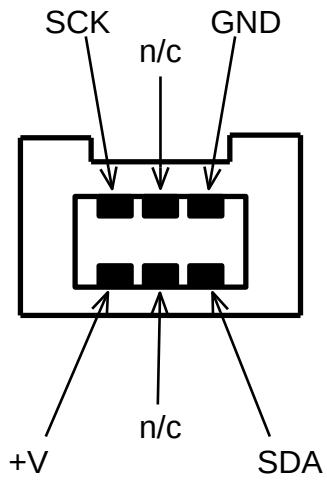


Но вместо этого, используем эту маленькую плату-адаптер



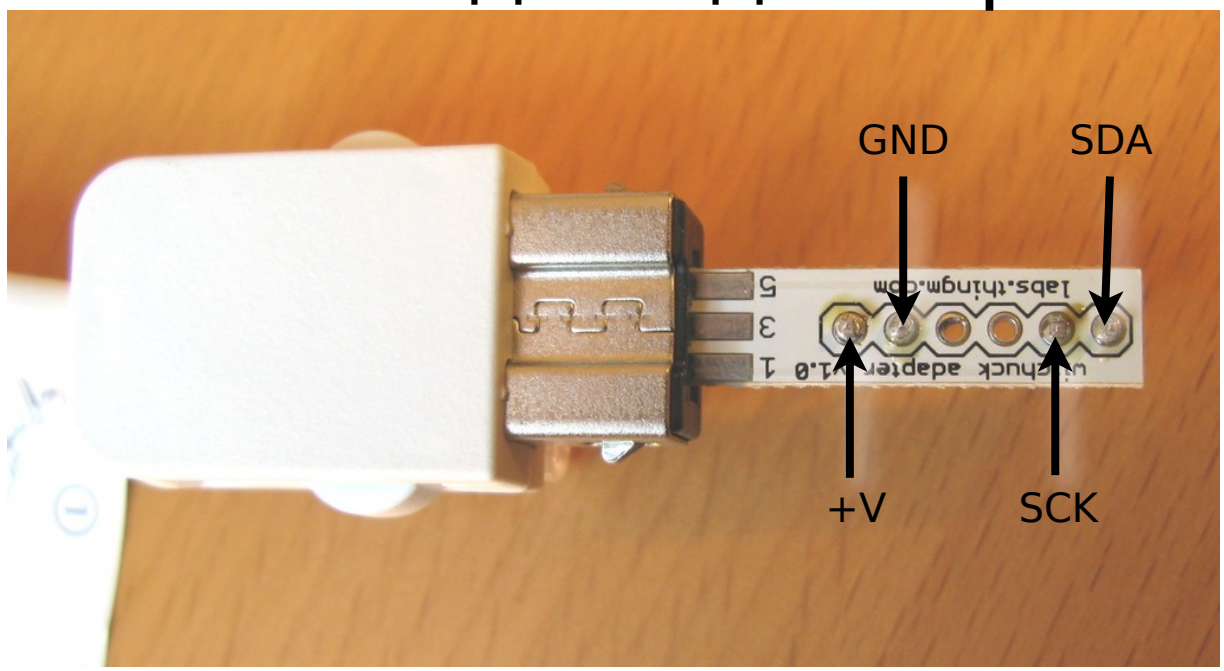
Адаптер Wii-Нунчака

Расположение выводов Нунчака



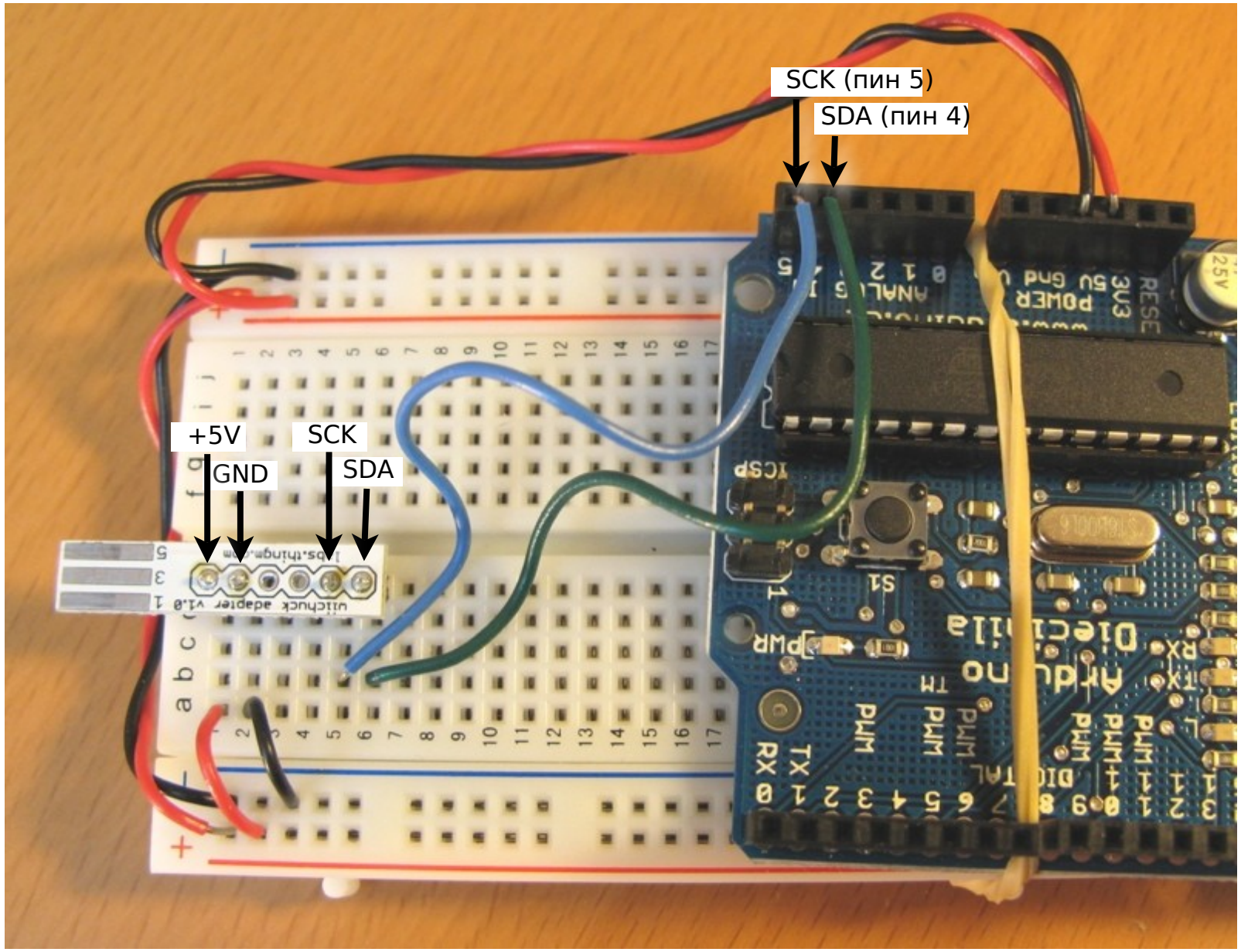
(если смотреть внутрь разъёма Нунчака)

Расположение выводов адаптера

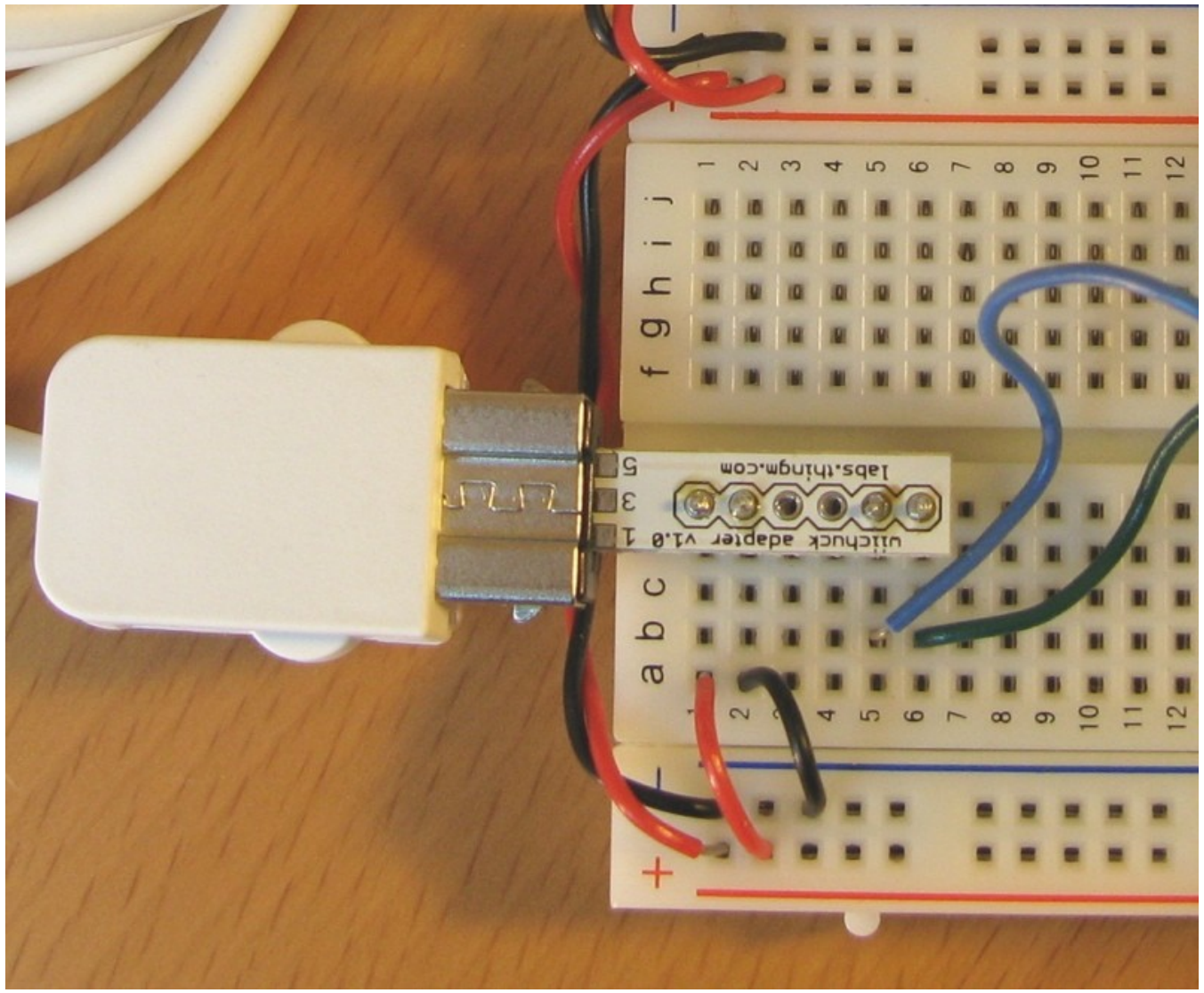


Внимание! На адаптере есть обозначения, но они неправильные. Так что Вам придётся положиться на вышеприведённые диаграммы.

Сборка схемы



Подключение Нунчака

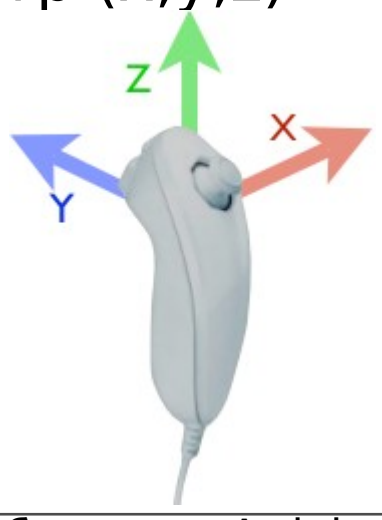


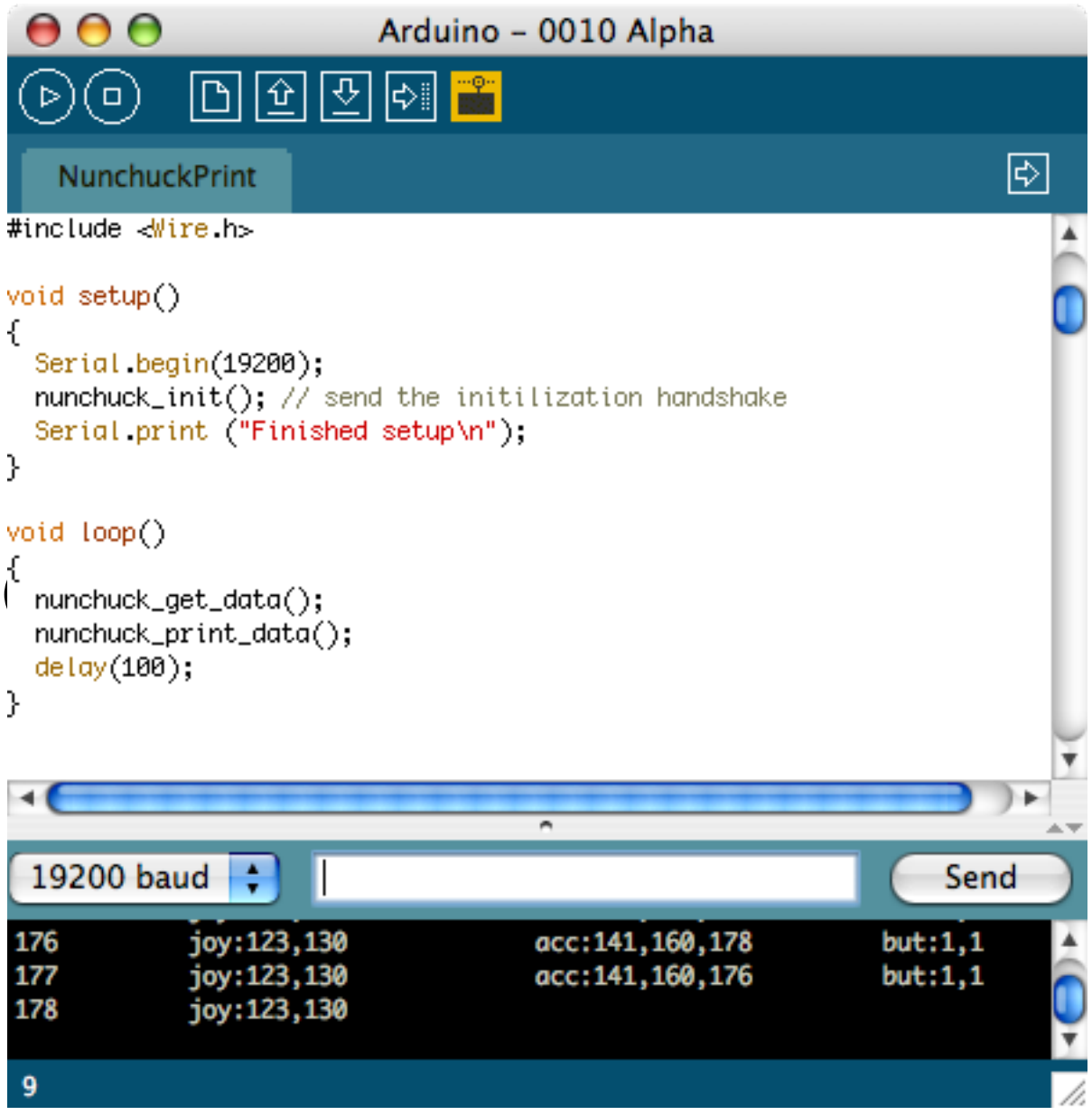
Пробуем Нунчак

“NunchuckPrint”

Считываем показания Нунчака каждую 1/10-ю секунды и выводим на экран все данные:

- положение джойстика
- акселерометр (x,y,z)
- кнопки Z,C





Использует основы библиотеки Arduino, которую я сейчас пишу.

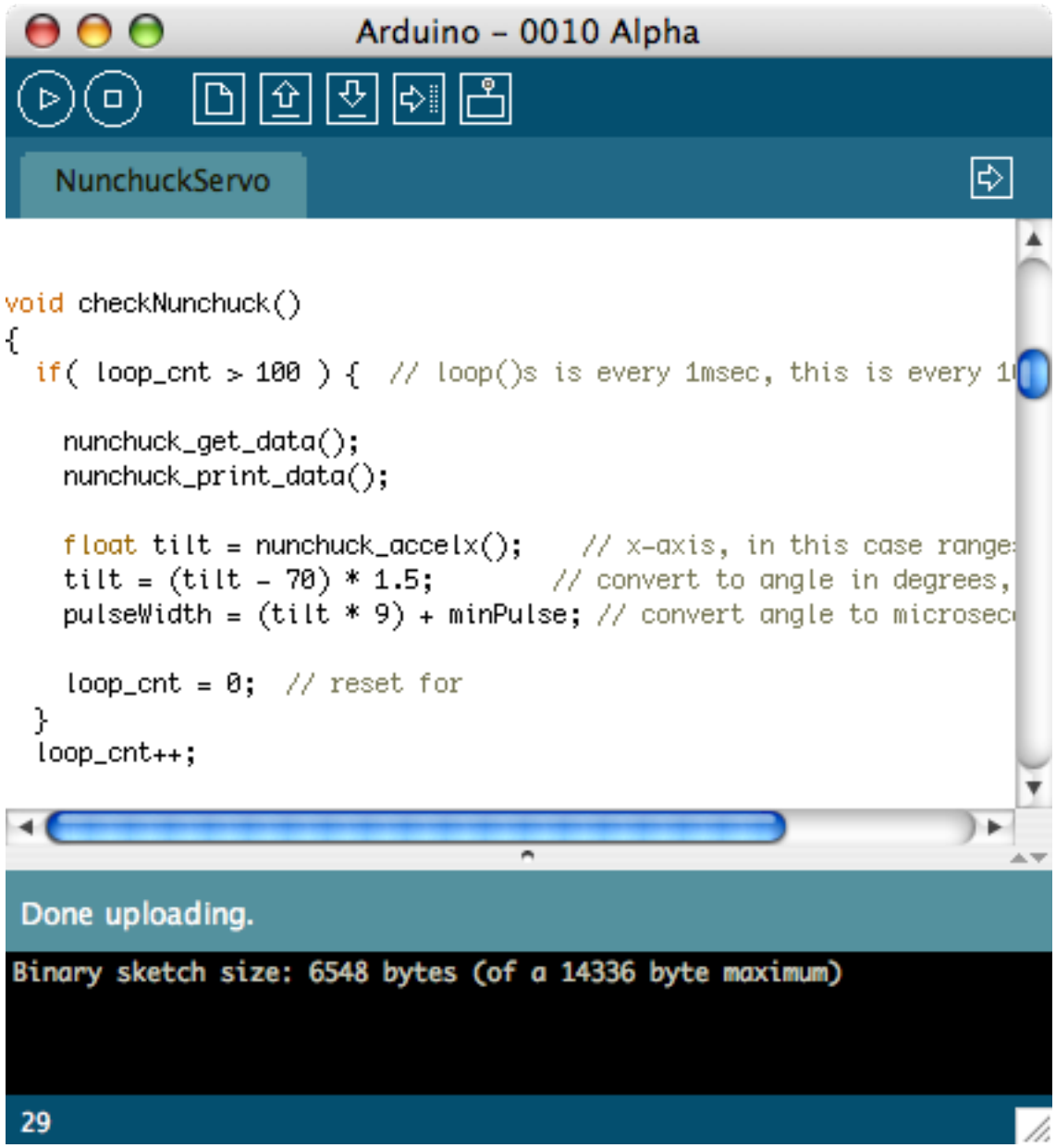
Добавляем сервомашинку

“NunchuckServo”

Перемещайте серву движением своей руки

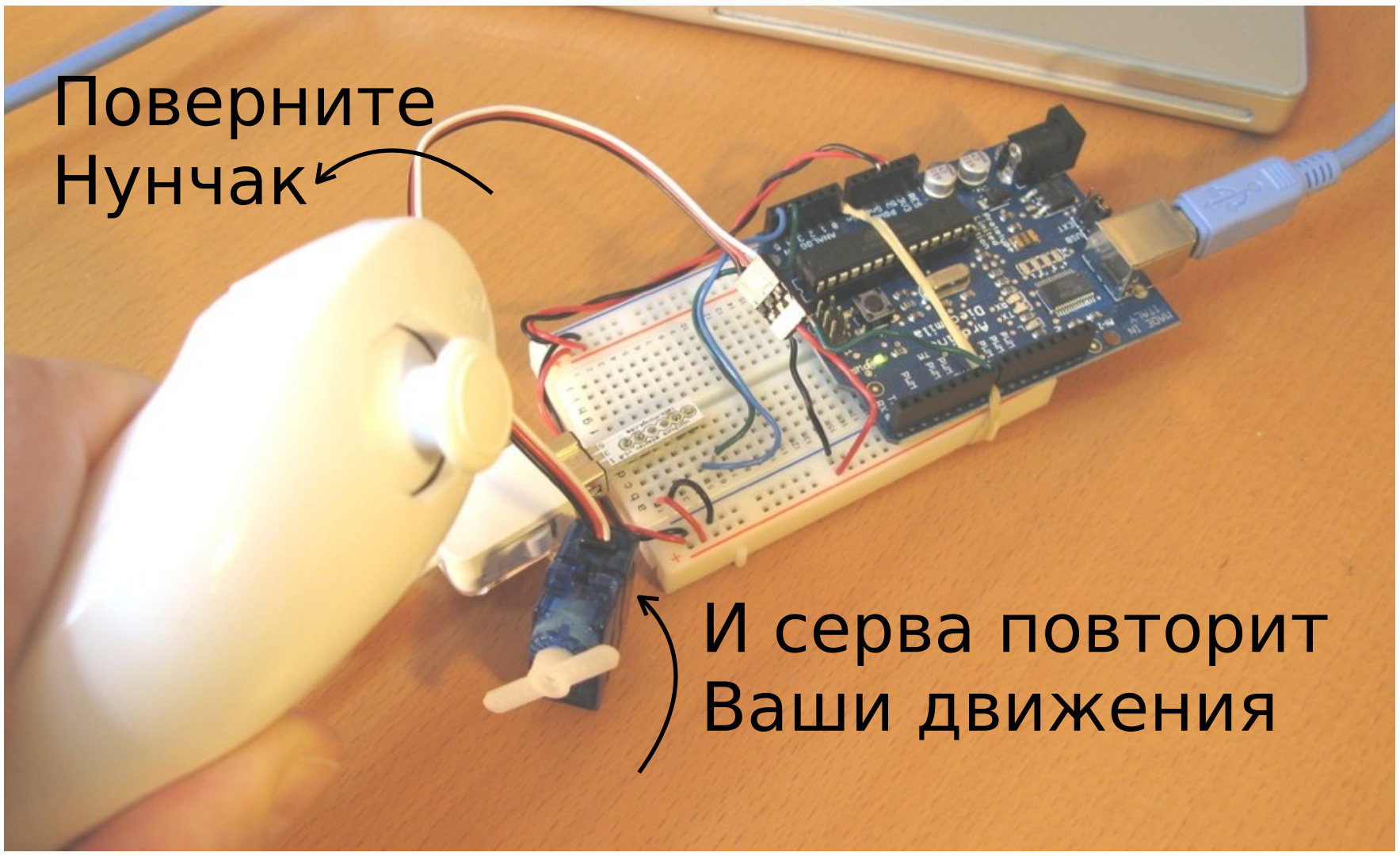
Вы - киборг!

Нажимайте кнопку Z, чтобы мигать светодиодом на 13-м выводе



Используется разделение на кванты времени, упомянутое ранее.

Нунчак и сервомашинка



Эмулятор Сегвэя

В основе — тот же код, что и в NunchuckServo.
Подробнее: http://todbot.com/blog/2007/10/25/boarduino-wii-nunchuck-servo/

Идём дальше

- Сервомашинки
  - Соедините их вместе - получится рука робота с несколькими степенями свободы.
  - Сделайте “серво-запись”: переводите движения руки в положения сервы, и затем проигрывайте эти записи.
  - Отлично годится для развлекательной аниматроники

Идём дальше

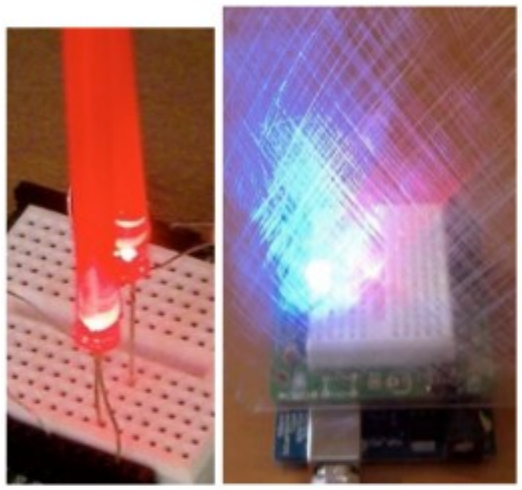
- Устройства с I2C
  - Попробуйте другие устройства
  - Подключите их к тем же проводам, что и Нунчак.
- Кооперативная многозадачность
  - Попробуйте сделать терменвокс из Нунчака и пьезоэлемента
  - Можно ли заставить предыдущие примеры реагировать быстрее?

Идём дальше

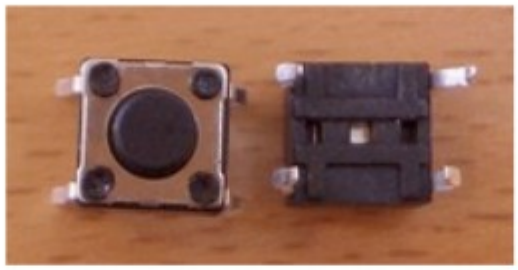
- Нунчак
  - Пространственный датчик движения. Управляйте чем угодно, как если бы Вы махали волшебной палочкой!
  - Как насчёт джойстика? У нас даже не было времени с ним поиграть.
  - Альтернативное устройство ввода для Вашего компьютера: управляйте программами на Processing и т.д.

В итоге

Вы изучили множество разных физических деталей



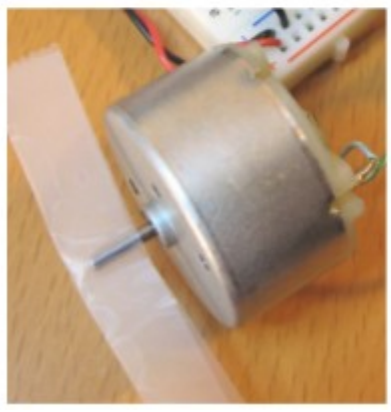
светодиоды



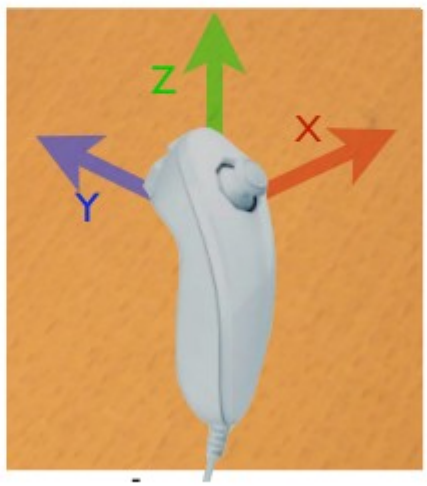
переключатели/кнопки

резистивные датчики

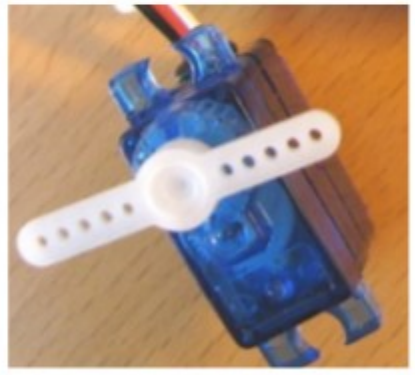
пьезоэлементы



моторы



акселерометры



сервомашинки

В итоге

И вы изучили множество программных деталей

коммуникация через последовательный порт

широтно-импульсная модуляция

I2C

аналоговый ввод-вывод

управляемый данными код (data driven code)

цифровой ввод-вывод

частотная модуляция

многозадачность

В итоге

Надеемся, Вам понравилось, продолжайте играть с Arduino!

Не стесняйтесь писать мне,
чтобы пообщаться обо всем этом.

КОНЕЦ занятия 4

http://todbot.com/blog/bionicarduino/

Тод Е. Курт

tod@todbot.com

Не стесняйтесь писать мне на почту, если есть вопросы