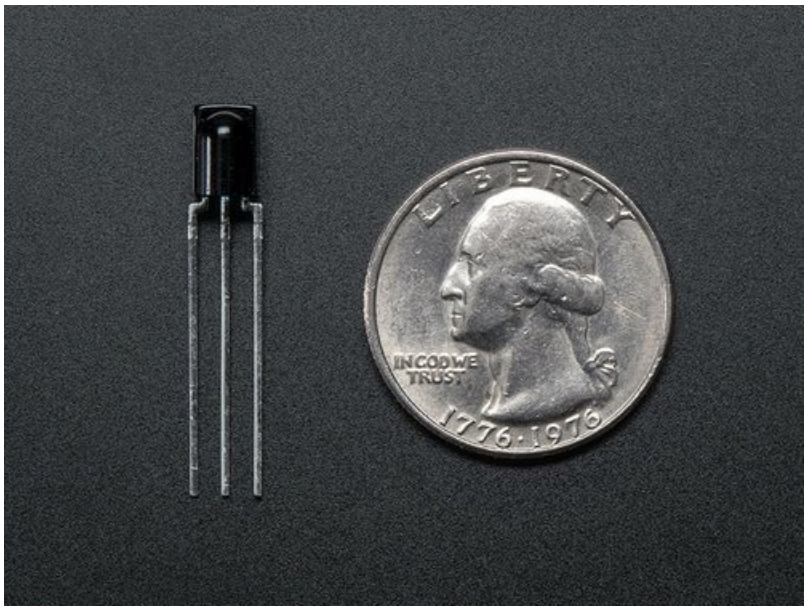
adafruit learning system

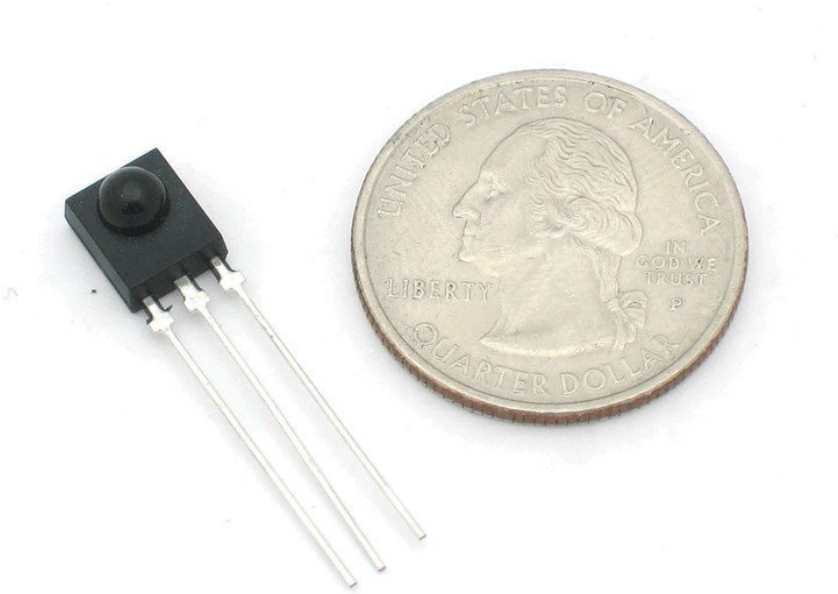
ИК-датчик

Создано леди Ад



Последнее обновление 2019-04-09 1:24:04 PM UTC

обзор



ИК-детекторы маленькие микрочип с помощью фотоэлемента, которые настроены для прослушивания инфракрасного света. Они почти всегда используются для обнаружения дистанционного управления - каждый телевизор и DVD-плеер имеет один из них в передней для прослушивания ИК-сигнал от кликера. Внутри пульта дистанционного управления является ИК-светодиод соответствия, который излучает импульсы ИК сказать телевизор, чтобы включить, выключить или изменить каналы. ИК свет не виден человеческому глазу, а значит, это займет немного больше работы, чтобы проверить установку.

Есть несколько разницы между ними и сказать CdS фотоэлементы (Https://adafru.it/aHA) :

- ИК-детекторы, специально фильтруется для инфракрасного света, они не хороши при обнаружении видимого света. С другой стороны, фотоэлементы хороша в обнаружении желтый / зеленый видимый свет, не очень хорошо инфракрасного света
- ИК-детекторы имеют демодулятор внутри, что ищет модулированного ИК при 38 кГц. Просто блестящий ИК-светодиод обыкновение быть обнаружен, он должен быть ШИМ мигает при 38kHz. Фотоэлементы не имеют какие-либо демодулятор и могут обнаружить любую частоту (в то числе DC) в пределах скорости отклика фотоэлемента (что составляет около 1 кГц) ИК-детекторы цифровых выхода - либо они обнаруживают 38KHz ИК-сигнал и выходной сигнал с низкими (0В), или они не обнаруживают
- какой-либо и высокий выход (5V). Фотоэлементы действуют как резисторы, сопротивление меняется в зависимости от того, сколько света они подвергаются

В этом уроке мы покажем, как

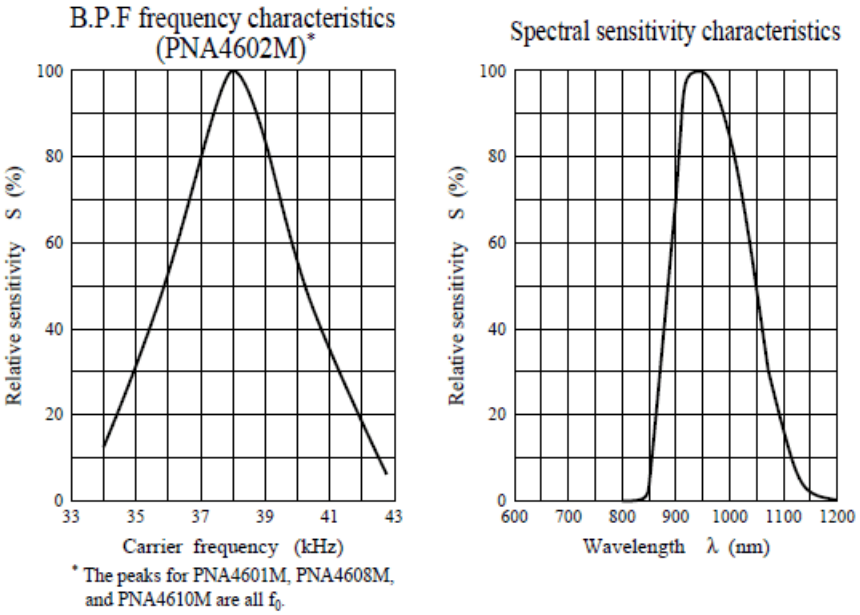
- Проверка ИК-датчик, чтобы убедиться, что его работа Read
- сырых ИК-коды в микроконтроллер Создания камеры
- интервалометра
- Прислушайтесь «команды» с пультом дистанционного управления на вашем микроконтроллероме

Некоторые статистики

Эти статистические данные являются для ИК-детектор в магазине Adafruit (Https://adafru.it/aIH) также известный как PNA4602. Почти все фотоэлементы будут иметь несколько различных спецификаций, хотя все они в значительной степени работают одинаково. Если есть техническое описание, вы хотите, чтобы обратиться к нему

- Размер: квадрат, 7 мм по площади 8мм детектора
- Выход: 0В (низкий) при обнаружении носителя 38KHz, 5V (высокий) в противном случае
- Диапазон чувствительности: 800 нм до 1100 нм с максимальной чувствительностью при 940нм. Диапазон воспроизводимых частот составляет 35KHz до 41KHz с пиковым детектированием при 38 кГц Напряжение питания: 3-5V DC 3mA
-
- PNA4602 Технический паспорт (Https://adafru.it/cm2) (В настоящее время прекращено) или GP1UX311QS (Https://adafru.it/cm3) или же TSOP38238 (Https://adafru.it/cm4) (пин-совместимые замены)

Что вы можете измерить

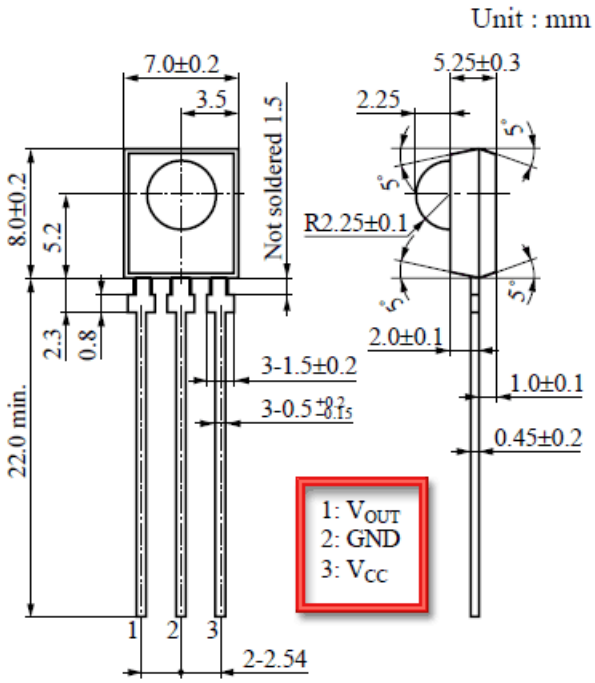


Как видно из этих графиков, техническое описание, определение пиковой частоты на 38 кГц, а пик СИД цвета 940 нм. Вы можете использовать примерно от 35 кГц до 41 кГц, а чувствительность упадет настолько, что он не будет обнаружить, а издалека. Кроме того, вы можете использовать 850 до 1100 нм светодиодов, но они не будут работать, а также от 900 до 1000нм поэтому убедитесь, чтобы получить соответствующие светодиоды! Проверьте спецификацию для вашей ИК-светодиода для проверки длины волны.

Попытка получить 940нм - помните, что 940nm не видимый свет (его Infra Red)!

Тестирование ИК-датчика

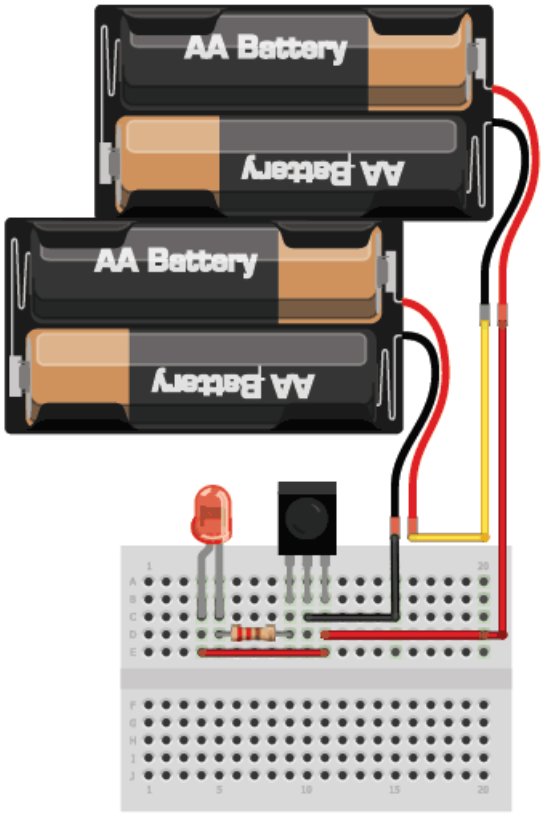
Потому что есть полупроводник / чип внутри датчика, он должен быть включен с 3 - 5V функционировать. Сравните это с фотоэлементами и CPЛCOM, где они действуют как резисторы и, таким образом, может быть просто протестированы с помощью мультиметра.



Здесь мы будем подключать детектор как таковой:

- Pin 1 является выходным, поэтому мы телеграфировать это видимый светодиод и резистор Pin 2
- размалывают
- Pin 3 является VCC, подключить к 3-5V

Когда детектор видит ИК-сигнал, он будет тянуть выход низкий, включение светодиода - так как светодиод красного цвета его гораздо легче для нас, чтобы увидеть, чем ИК!



Мы будем использовать 4xAA 1.3V батареи (я использую NiMH), так что напряжение питания датчика составляет около 4В.

2 батарейки (3V) может быть слишком мало. 3 Батареи должны быть хорошо, если у вас есть держатель тройной AA

Вы также можете получить 5V с микроконтроллером, как Arduino, если у вас есть один вокруг. Первый идет в центральный контакт.

Положительный (больше) глава красного светодиода подключается к контакту + 6В и отрицательному (короче, свинец) соединяется через 200 до 1000 Ом резистора к первому штифта на ИК-датчике.

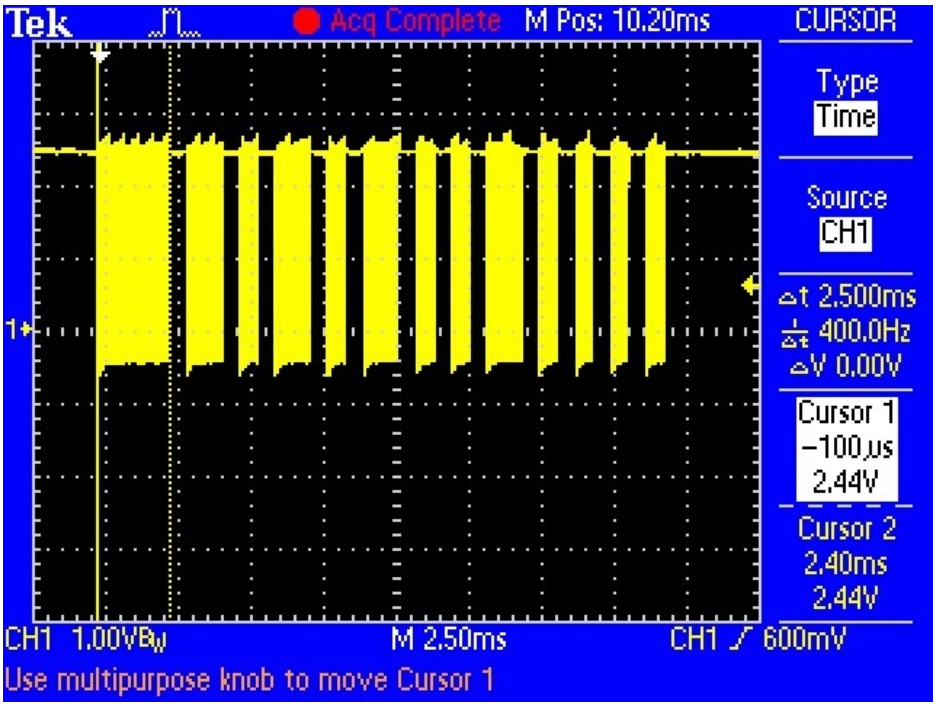
Теперь возьмите любой пульт дистанционного управления, как для телевизора, DVD, компьютер и т.д., и направьте его на датчик, нажимая несколько кнопок, вы должны увидеть индикатор мигает несколько раз при каждом нажатии пульта дистанционного управления.

ИК-сигналы пульта дистанционного

Теперь мы знаем, что датчик работает, мы хотим выяснить, Что посылается правильно? Но прежде чем мы это сделаем, давайте сначала рассмотрим, как именно данные передаются из ИК-пульта дистанционного управления (в руке) в ИК-датчик приема (на плате)

Для этого примера мы будем использовать мощности Sony включения / выключения ИК-кода с пульта ДУ телевизора Sony. Его очень просто и обычно документированы!

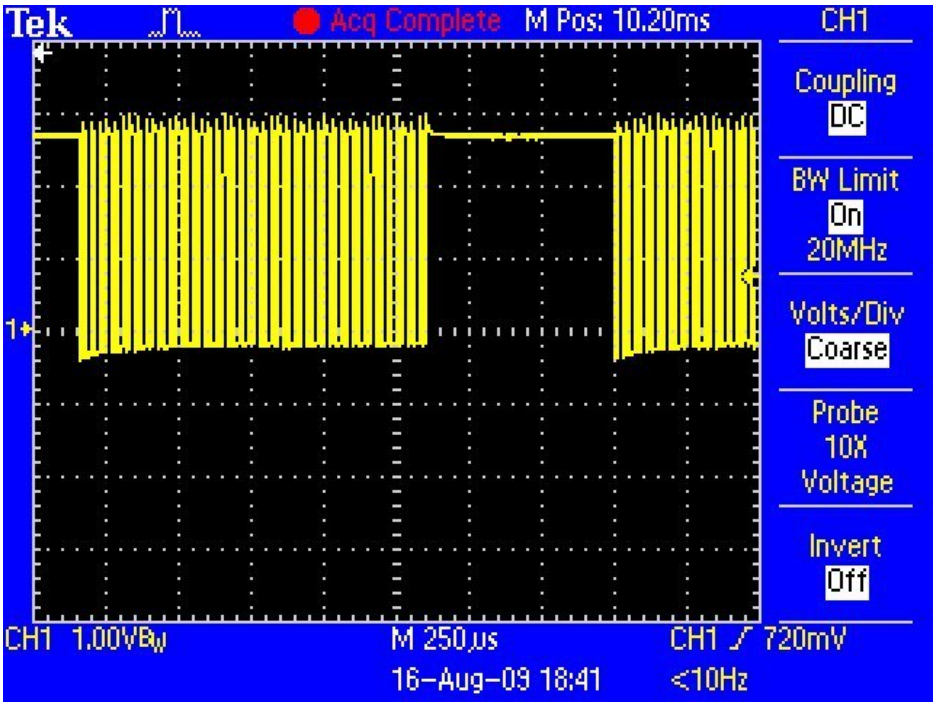
Позволяет делать вид, что есть пульт дистанционного управления Sony, и мы можем посмотреть на то, что свет идет выброшенным из ИК-светодиода. Мы Подвод основного датчика света (как основной фотоэлемент!) И слушать. Мы не будем использовать декодер, как PNA4602 (только пока), потому что мы хотим, чтобы увидеть недекодированную сигнал. То, что мы видим, состоит в следующем:



В основном мы видим импульсы или ИК-сигнал. желтые «блоки» являются, когда ИК-светодиод передачи и, когда есть только линия, ИК-светодиод выключен. (Обратите внимание, что напряжение будучи в 3Vdc только из-за того, как я подключил датчик, если бы я поменял подтяжку для спуска было бы на земле.)

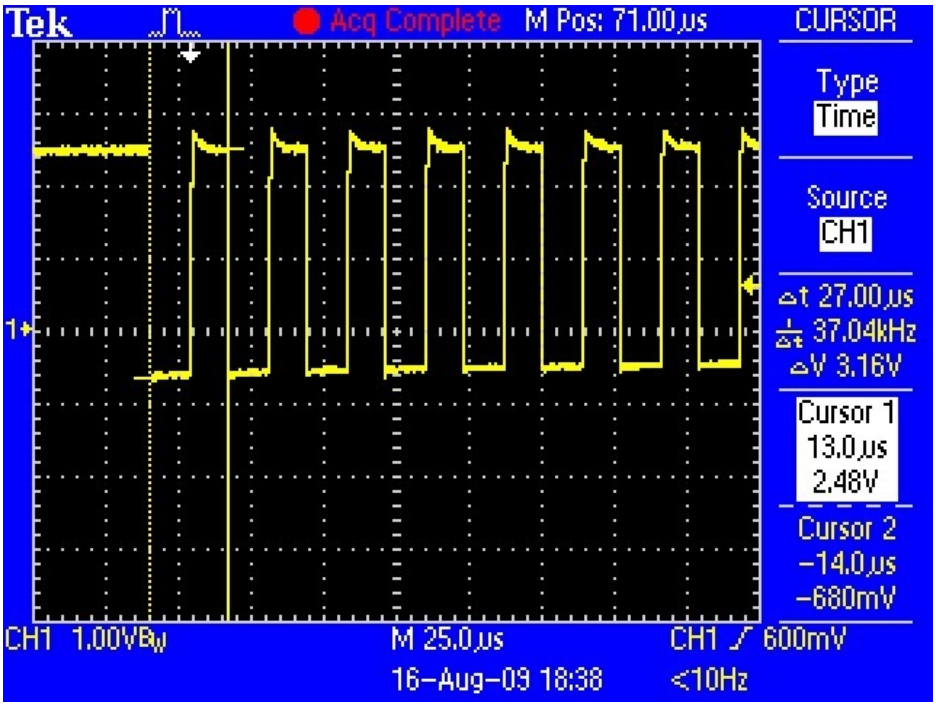
Первый «блок» составляет около 2,5 мс длиной (см курсоров и измерение на стороне)

Если вы увеличите в один из этих блоков ...



Вы видите, что они на самом деле не «блоки», но на самом деле очень быстрые импульсы!

Если вы увеличиваете всю дорогу ...



Можно измерить частоту ИК-импульсов. Как вы можете сказать, курсоры и измерений на стороне, частота составляет около 37.04KHz

ОК, так что теперь мы можем понять, как ИК-коды посылаются. СИД ИК-передатчик быстро импульсный (ШИМ - широтно-импульсной модуляцией) при высокой частоте 38 кГц, а затем, что ШИМ аналогичным образом в импульсном режиме и выключается гораздо медленнее, время от времени, которые имеют длину около 1-3 мс.

Почему нет светодиода просто и выключается? Почему PWM «носитель» пульсирует? Много причин!

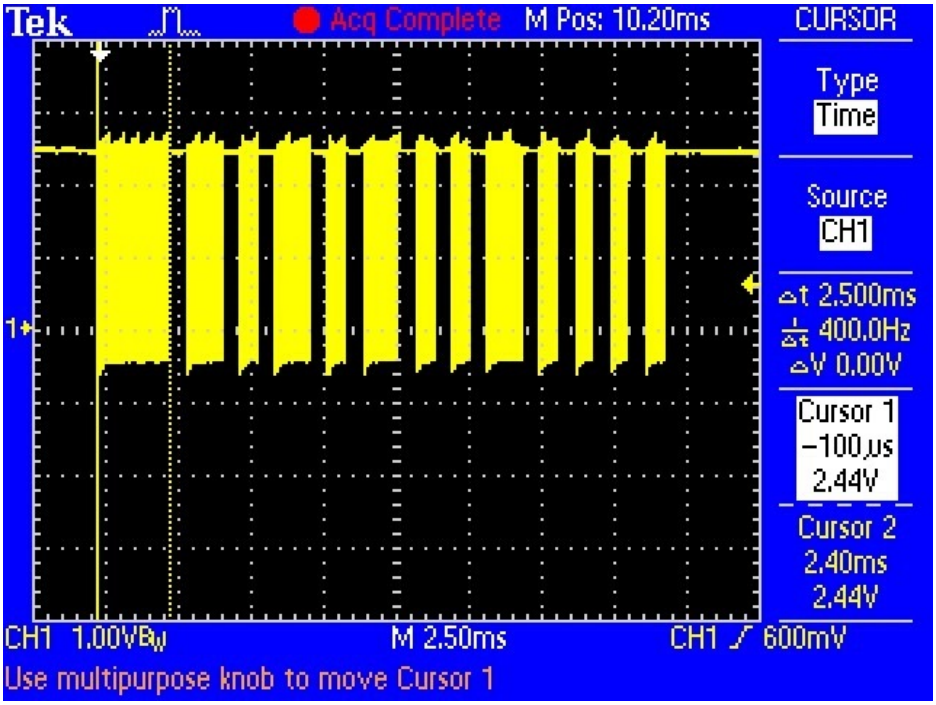
Одной из причин является то, что это позволяет светодиод остыть. ИК-светодиоды могут занять до 1 А (1000 миллиампер!) Тока. Большинство СИД только принимает 20 мА или около того. Это означает, что ИК-светодиоды предназначены для мощных взрывных работ, но они могут принять его только в течение нескольких микросекунд. По PWM'ing, вы дайте остыть СИД половину времени

Другая причина заключается в том, что телевизор будет слушать только определенные частоты ШИМ. Так Sony пульт дистанционного управления на 37KHz обыкновение иметь возможность работать с JVC DVD-плеер, который только хочет сказать 50кГц.

И, наконец, наиболее важной причиной является то, что пульсирует на несущую волну, вы уменьшить влияние окружающего освещения. Телевизор смотрит только на изменение уровня освещенности что часы в около 37KHz. Так же, как его легче для нас, чтобы сказать различие между аудиотонами чем придавить шаг precsise тонуса (ну, для большинства людей, по крайней мере)

ОК, так что теперь мы знаем несущую частоту. Его 37KHz. Далее позволяет найти ширины импульса!

Оглядываясь назад на первой области видимости изображения



Первый импульс 2,5 мс. Мы можем использовать курсоры для измерения оставшихся импульсов. Я избавлю вас 12 изображений и пусть вы знаете, что импульсы:

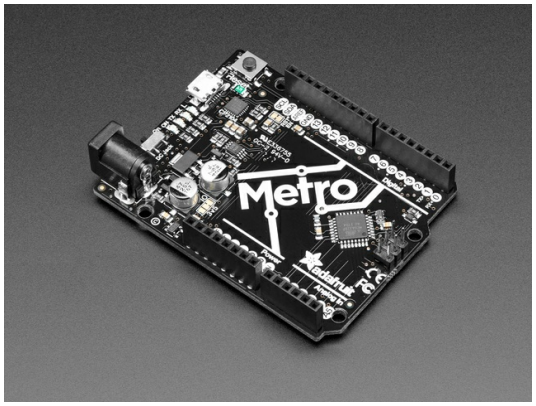
PWM ON	OFF
2,4 мс	0,6 мс
1,2 мс	0,6 мс
0,6 мс	0,6 мс
1,2 мс	0,6 мс
0,6 мс	0,6 мс
1,2 мс	0,6 мс
0,6 мс	0,6 мс

0,6 мс	0,6 мс
1,2 мс	0,6 мс
0,6 мс	0,6 мс
0,6 мс	0,6 мс
0,6 мс	0,6 мс
0,6 мс	270 мс

Так что позволяет сказать, что вы не имеете $ 1000 осциллографа, как еще вы можете читать эти сигналы? Ну ИК-декодер, такие как PNA4602 делает нам одолжение, это отфильтровывает "сигнал 38KHz, так что мы только получаем большие куски сигнала в диапазоне milliscond. Это гораздо проще для микроконтроллера для обработки. То, что мы будем делать в следующем разделе!

С помощью ИК-датчика

Хорошая новость заключается в том, что это очень легко подключить этот датчик. Просто подключите выход к цифровому выводу. Плохая новость заключается в том, что дружественная процедура в Arduino в digitalRead () является немного слишком медленно, чтобы надежно читать быстрый сигнал, как его приход. Таким образом мы используем функцию аппаратной шпильки чтения непосредственно из контактного D2, это то, что линия «IRpin_PIN & BV (Ирпеня))»делает. Этот трюк является специфичным для плат ATmega328 на основе таких как Arduino Uno, Adafruit метро и т.д.

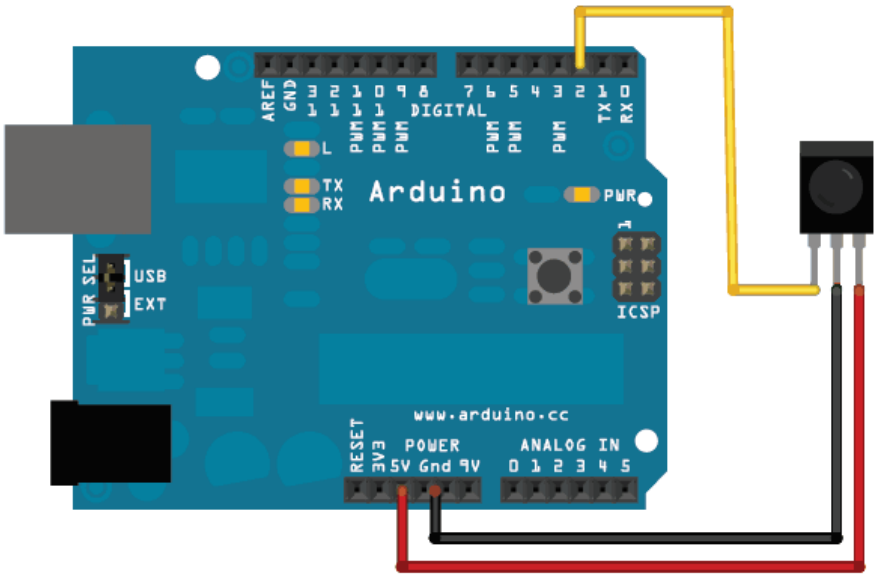


Adafruit METRO 328 Полностью Собранный - Arduino IDE совместимый

$ 17,50

В НАЛИЧИИ

ДОБАВИТЬ В КОРЗИНУ

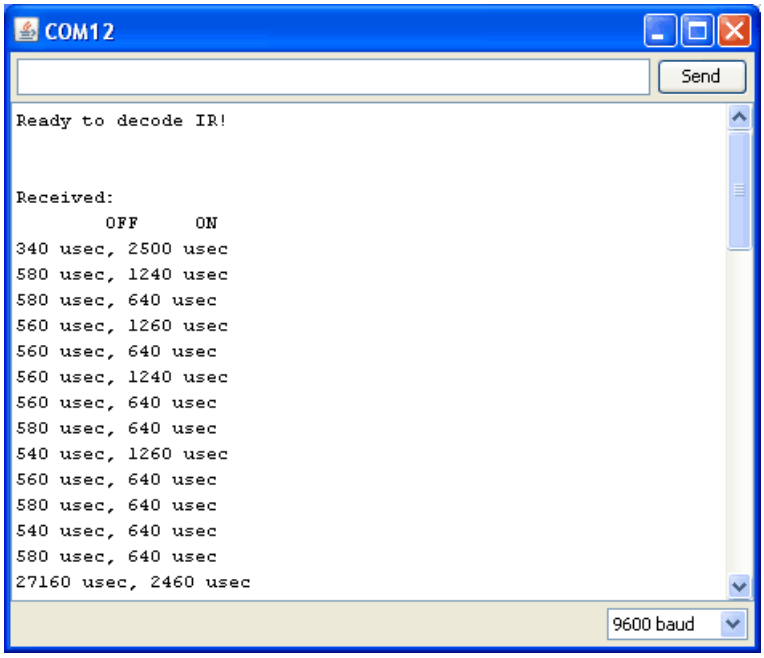



```
/ * Сырые ИК декодер эскиз!

  Этот эскиз / программа использует Arduino и PNA4602 для декодирования ИК получила. Это может
  быть использовано, чтобы сделать ИК-приемник (путем поиска определенного кода)

  или передатчик (импульсные спектры ИК светодиод на ~ 38 кГц для длительности обнаружено


  Код является общественным достоянием, проверить www.ladyada.net и adafruit.com для более учебников!

  * /

// Нам нужно использовать методы «сырые» шпилька чтения // потому что сроки очень важно здесь и
digitalRead ()
```

```
// потому что сроки очень важно здесь и digitalRead () // Процедура медленнее! // uint8_t Ирпеня = 2;


// Цифровой контактный # 2 так же, как видим Pin D2
// http://arduino.cc/en/Hacking/PinMapping168 для «сырого» отображения контактов
# определить IRpin_PIN PIND
# определить Ирпене 2

// максимальный импульс, мы будем слушать - 65 миллисекунд долгое время
# определить MaxPulse 65000

// то, что наша резолюция времени должно быть, больше, тем лучше // как его более «точным», - но
слишком большой, и вы не получите // точного времени

# определить РАЗРЕШЕНИЕ 20

// мы будем хранить до 100 пар импульсов (это -a lot-) uint16_t импульсы [100] [2]; // пара высокого и
низкого импульсов uint8_t currentpulse = 0; // индекс для импульсов мы хранящих


недействительные установки (недействительная) {
  Serial.begin (9600);
  Serial.println ( "Готово декодировать ИК!"); }


недействительные петли (недействительная) {
  uint16_t HighPULSE, lowpulse; // временного хранения времени HighPULSE = lowpulse = 0; // начать с
  длительностью импульса не


// пока (digitalRead (г.Ирпень)) {// это слишком медленно!
      в то время как (IRpin_PIN & (1 << Ирпеня)) {// штырьковый
        еще ВЫСОКИЙ

        // отсчитать еще несколько микросекунд HighPULSE ++;

        delayMicroseconds (разрешение);

        // Если пульс слишком долго, мы «не истекли» - либо ничего // был получен или код закончен, поэтому
        печати, что // мы схватились до сих пор, а затем сбрасывается, если ((HighPULSE> = MaxPulse) &&
        (currentpulse! = 0)) {

            printpulses (); currentpulse =
         0; вернуть; }}


  // мы не тайм-аут так позволяет копить импульсы чтения [currentpulse] [0] =
  HighPULSE;

  // то же самое, что и выше
  в то время как (! (IRpin_PIN & _BV (г.Ирпень))) {
        // контактный еще НИЗКИЙ
        lowpulse ++;
        delayMicroseconds (разрешение);
        если ((lowpulse> = MaxPulse) && (currentpulse! = 0)) {
            printpulses (); currentpulse =
          0; вернуть; }
```

```
    }
    импульсы [currentpulse] [1] = lowpulse;

    // мы читаем один высокий-низкий пульс успешно, продолжайте! currentpulse ++; }



недействительный printpulses (недействительный) {
    Serial.println ( "\ n \ r \ n \ rReceived: \ N \ ROFF \ Ton"); для (uint8_t = 0; r <currentpulse; я
    ++) {
        Serial.print (импульсы [I] [0] * РЕШЕНИЕ, декабрь); Serial.print ( "мксек,");

        Serial.print (импульсы [I] [1] * РЕШЕНИЕ, декабрь); Serial.println ( "мксек"); }



    // распечатать его в формате 'массива' Serial.println ( "ИНТ
    IRsignal [] = {"); Serial.println ( "// ON, OFF");

    для (uint8_t = 0; r <currentpulse-1; я ++) {
        //Serial.print("\t "); // вкладка Serial.print ( "pulseIR
        (");
        Serial.print (импульсы [I] [1] * РЕШЕНИЕ, декабрь); Serial.print ( ");"); Serial.println (
        ""); //Serial.print("\t ");


        Serial.print ( "delayMicroseconds ("); Serial.print (импульсы [+ 1] [0] * РЕШЕНИЕ,
        декабрь); Serial.println ( ");");


    }
    //Serial.print("\t "); // вкладка Serial.print ( "pulseIR
    (");
    Serial.print (импульсы [currentpulse-1] [1] * РЕШЕНИЕ, декабрь); Serial.print ( ");");


}
```

Если запустить это, указывая на Sony ИК пульт дистанционного управления и нажав кнопку ON, вы получите следующее ...



Если вы игнорируете первый импульс OFF (его только время с момента, когда Arduino включена в первом сигнале ИК получил) и последний импульс ON (это начало следующий код), вы найдете код питания Sony:

PWM ON	OFF
2,5 мс	0,6 мс
1,2 мс	0,6 мс
0,6 мс	0,6 мс
1,2 мс	0,6 мс
0,6 мс	0,6 мс
1,2 мс	0,6 мс
0,6 мс	0,6 мс
0,6 мс	0,6 мс
1,2 мс	0,6 мс
0,6 мс	0,6 мс
0,6 мс	0,6 мс
0,6 мс	0,6 мс
0,6 мс	27.2 мс

Создание Интервалометр

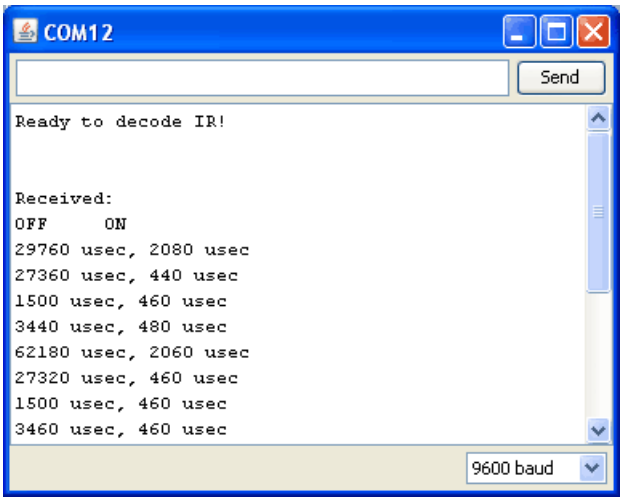
Хорошо, теперь, что мы можем читать ИК-коды, позволяет сделать основной проект. Первый из них мы будем делать это сделать интервалометр. Интервалометр в основном электронная штуковина, что делает камеру срабатывала каждые несколько минут или около того. Это может быть использовано для покадровых проектов или кайт Arial фотографии или других фотографии проектов.

Камера мы будем использовать имеет ИК-пульт дистанционного управления, вы можете использовать, чтобы установить его (большинство более дорогих камер имеют их).

Сначала мы выяснить коды путем считывания сигнала, посылаемого при нажатии кнопки. Тогда мы будем считать, что данные и

сделать Arduino выплюнуть этот код в ИК-светодиод один раз в минуту

ОК шаг один легко, направьте пульт дистанционного управления на ИК-датчик и нажмите на кнопку, мы получили следующее для нашего ML-L3 Nikon дистанционного управления.



Похоже, данные послали:

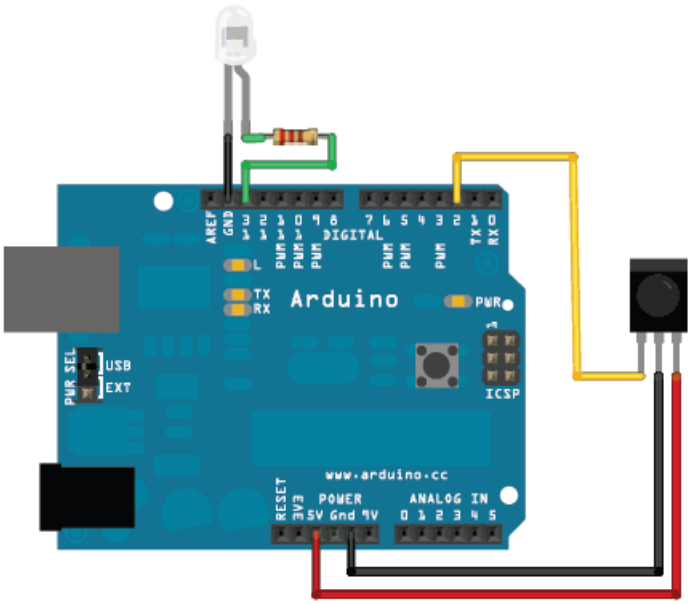
PWM ON	OFF
2,0 мс	27 мс
0,4 мс	1,5 мс
0,5 мс	3,5 мс
0,5 мс	62.2 мс
2,0 мс	27 мс
0,5 мс	1,5 мс
0,5 мс	3,5 мс
0,5 мс	

Если вы внимательно посмотрите, вы увидите его на самом деле просто

PWM ON	OFF
2,0 мс	27 мс
0,4 мс	1,5 мс
0,5 мс	3,5 мс
0,5 мс	62.2 мс

отправлен в два раза. Отправка и тот же сигнал дважды очень часто - удвоение, чтобы убедиться, что он получает получил

Далее мы должны подключить ИК 940нм светодиод на выходе Arduino



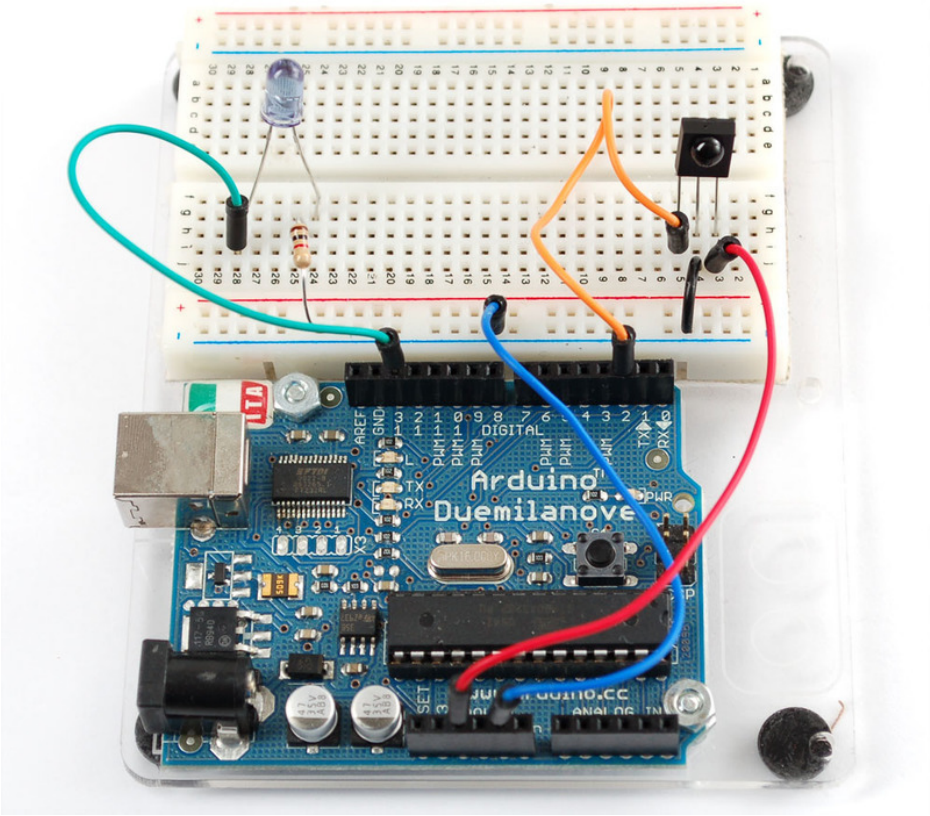
Тогда мы напишем эскиз, который будет пульсировать пин # 13 и выключается очень быстро в правильной последовательности кода.

```
// Этого эскиз будет посылать сигнал триггера D50 Nikon (вероятно, работает с большинством Никонов) // Смотрите полный учебник по
https://learn.adafruit.com/ir-sensor/making-an-intervalometer // MIT License, атрибуция оценили
                                                                    Limor Фрид, Adafruit Industries

INT IRledPin = 13;                                // светодиод подключен к цифровому контакту 13

// Метод установки () выполняется один раз, когда эскиз начинается

недействительные установки ()     {
    // инициализировать цифровой контактный ИК в качестве выходного сигнала:
    pinMode (IRledPin, OUTPUT);

    Serial.begin (9600); }


недействительным цикл () {

    Serial.println ( "Передача ИК-сигнала");

    SendNikonCode ();

    Задержка (60 * 1000); // подождите одну минуту (60 секунд * 1000 миллисекунд)}


// Эта процедура посылает 38kHz импульс к IRledPin
// для некоторых # микросекунд. Мы будем использовать это всякий раз, когда нам нужно отправлять коды пустот pulseIR (длинный microsecs) {

    // мы будем отсчитывать от числа микросекунд мы сказали ждать

    кли (); // это отключает любые фоновые прерывания

    в то время как (microsecs> 0) {
        // 38 кГц составляет около 13 мкс высоты и 13 мкс с низким digitalWrite (IRledPin, HIGH); // это занимает около 3 мкс произойдет
```

```
    digitalWrite (IRledPin, HIGH); // это занимает около 3 мкс случаться delayMicroseconds (10);
                                                    // болтаться в течение 10 микросекунд, вы можете также изменить это на 9, если ее не работает
    digitalWrite (IRledPin, LOW);                   // это также занимает около 3 мкс
    delayMicroseconds (10);                         // болтаться в течение 10 микросекунд, вы можете также изменить это на 9, если ее не работает

    // так 26 микросекунд в общей сложности microsecs - =
   26; }


  SEI (); // это превращает их обратно}


SendNikonCode недействительными () {
  // Это код для моего конкретного Nikon, для других использовать учебник // для «захвата» соответствующего кода с пульта
  дистанционного

  pulseIR (2080); задержка
  (27); pulseIR (440);

  delayMicroseconds (1500); pulseIR (460);

  delayMicroseconds (3440); pulseIR (480);



  задержка (65); // Ожидаем 65 миллисекунд, прежде чем отправить его еще раз

  pulseIR (2000); задержка
  (27); pulseIR (440);

  delayMicroseconds (1500); pulseIR (460);

  delayMicroseconds (3440); pulseIR (480); }
```

аннулированию pulseIR (длинный microsecs) наша процедура помощника, она будет создавать PWM ИК-сигнал, как мы видели раньше. Я использовал свою область, чтобы точно настроить его так, что задержки добавленного право. Мы используем не-часто обсуждаемые кли () а также SEI () процедуры отключить прерывания. Arduino делает пару вещей в фоновом режиме, как ищет последовательных данных для чтения или записи, отслеживания времени и т.д. Большую часть времени мы можем просто игнорировать это, но для тонких сигналов высокой скорости, как это мы хотим сохранить спокойствие, так что мы получаем хороший чистый сигнал

Если вы посмотрите на SendNikonCode () вы увидите код команды ИК, что мы выведенный в предыдущем проекте по времени импульсов от датчика ИК.



Мы проводной это и он работал большой, убедитесь, чтобы указать ИК-светодиод на камеру правильно.

Чтение ИК-команд

Для нашего окончательного проекта, мы будем использовать пульт дистанционного управления для отправки сообщений на микроконтроллер. Например, это может быть полезно для робота, который может быть направлен с ИК-пульта дистанционного управления. Она также может быть полезна для проектов, которые вы хотите контролировать издалека, без проводов.

Для дистанционного управления в этом примере мы будем с помощью кликера пульт Apple Remote. Вы можете использовать любой вид пульта дистанционного управления, который вы хотите, или вы можете украсть один из них от ничего не подозревающего хипстера.

Мы будем использовать код из нашего предыдущего эскиза для сырой ИК чтения, но на этот раз мы будем редактировать наш принтер-внешнее, чтобы он дает нам импульсы в массиве C, это будет сделать легче для нас, чтобы использовать для сопоставления с образцом.

```
недействительный printpulses (недействительный) {
    Serial.println ( "\ n \ r \ n \ rReceived: \ N \ ROFF \ Ton"); для (uint8_t = 0; r <currentpulse; я
    ++) {
        Serial.print (импульсы [I] [0] * РЕШЕНИЕ, декабрь); Serial.print ( "мксек,");

        Serial.print (импульсы [I] [1] * РЕШЕНИЕ, декабрь); Serial.println ( "мксек"); }



    // распечатать его в формате 'массива' Serial.println ( "ИНТ
    IRsignal [] = {");
    Serial.println ( "// ON, OFF (в 10-х микросекунд)"); для (uint8_t = 0; r <currentpulse-1; я ++) {

        Serial.print ( "\ т"); // вкладка
        Serial.print (импульсы [I] [1] * РЕШЕНИЕ / 10, декабрь); Serial.print ( "");

        Serial.print (импульсы [+ 1] [0] * РЕШЕНИЕ / 10, декабрь); Serial.println ( ""); }


    Serial.print ( "\ т"); // вкладка
    Serial.print (импульсы [currentpulse-1] [1] * РЕШЕНИЕ / 10, декабрь); Serial.print ( "0};"); }
```

Я загрузил новый эскиз и нажал кнопку воспроизведения на пульте дистанционного управления Apple, и получил следующее:

```
ИНТ IRsignal [] = {// ON, OFF (в 10-х микросекунд)
912, 438,
68, 48,
68, 158,
68, 158,
68, 158,
68, 48,
68, 158,
68, 158,
68, 158,
70, 156,
70, 158,
68, 158,
68, 48,
68, 46,
70, 46,
68, 46,
68, 160,
68, 158,
70, 46,
68, 158,
68, 46,
70, 46,
68, 48,
68, 46,
68, 48,
66, 48,
68, 48,
66, 160,
66, 50,
66, 160,
66, 52,
64, 160,
66, 48,
66, 3950,
908, 214,
66, 3012,
908, 212,
68, 0};
```

Мы будем стараться, чтобы обнаружить этот код.

Давайте начнем новый эскиз под названием IR Commander (вы можете загрузить окончательный код с GitHub на зеленую кнопку ниже или нажмите Zip Скачать Project в полном листинге кода).

https://adafru.it/Eta

https://adafru.it/Eta

```
/ * Сырые ИК командир

  Этот эскиз / программа использует Arduno и PNA4602 для декодирования ИК получила. Затем он пытается
  согласовать его с ранее записанным ИК-сигнала. Limor Фрид, Adafruit Industries



  MIT License, пожалуйста, приписывать
  проверить learn.adafruit.com для более учебников!
```

```
  проверить learn.adafruit.com для более учебников!
  * /

// Нам нужно использовать «сырую» методу шпильки чтения // потому что сроки очень важно здесь и
digitalRead () // Процедура медленнее! // uint8_t Ирпеня = 2;


// Цифровой контактный # 2 так же, как видим Pin D2
// http://arduino.cc/en/Hacking/PinMapping168 для «сырого» отображения контактов
# определить IRpin_PIN          PIND
# определить Ирпене             2

// максимальный импульс, мы будем слушать - 65 миллисекунд долгое время
# определить MaxPulse 65000
# определить NUMPULSES 50

// то, что наша резолюция времени должно быть, больше, тем лучше // как его более «точным», - но
слишком большой, и вы не получите // точного времени

# определить РАЗРЕШЕНИЕ 20

// Какой процент мы разрешим в изменении, чтобы соответствовать один и тот же код
# определить нечеткость 20

// мы будем хранить до 100 пар импульсов (это -a lot-) uint16_t импульсы [NUMPULSES] [2]; // пара высокого и
низкого импульсов uint8_t currentpulse = 0; // индекс для импульсов мы хранящих


# включают «ircommander.h»

недействительные установки (недействительная) {
    Serial.begin (9600);
    Serial.println ( "Готово декодировать ИК!"); }


недействительные петли (недействительная) {
    ИНТ numberpulses;

    numberpulses = listenForIR ();

    Serial.print ( "Слышал"); Serial.print
    (numberpulses);
    Serial.println ( "- импульсная длиной ИК-сигнал");
    если (IRcompare (numberpulses, ApplePlaySignal, SizeOf (ApplePlaySignal) / 4)) {
        Serial.println ( "PLAY"); }

        если (IRcompare (numberpulses, AppleRewindSignal, SizeOf (AppleRewindSignal) / 4)) {Serial.println ( "перемотку"); }


        если (IRcompare (numberpulses, AppleForwardSignal, SizeOf (AppleForwardSignal) / 4)) {Serial.println ( "ВПЕРЕД"); }


    задержки (500); }


// KGO: добавлен размер сравнения образца. сравнить только минимум двух булевых IRcompare (интермедиат numpulses, Int
Signal [], Int refsize) {
    INT кол = мин (numpulses, refsize); Serial.print ( "Count
    установлен в:"); Serial.println (количество);

    для (INT I = 0; г <кол-1; я ++) {
```

```
    для (INT I = 0; г <кол-1; я ++) {
        INT oncode = импульсы [я] [1] * РЕШЕНИЕ / 10; INT offcode = импульсы [+ 1] [0] *
        РЕШЕНИЕ / 10;

# IFDEF DEBUG
        Serial.print (oncode); // сигнал ON мы услышали Serial.print ( "-");

        Serial.print (Сигнал [я * 2 + 0]); // сигнал ON мы хотим
# ENDIF

        // проверить, чтобы убедиться, ошибка меньше размытости процентов, если (ABS (oncode - Сигнал [я * 2 + 0]) <= (Сигнал [я * 2 + 0] *
        размытость / 100)) {
# IFDEF DEBUG
            Serial.print ( "(OK)");
# ENDIF
        } Еще {
# IFDEF DEBUG
            Serial.print ( "(x)");
# ENDIF
            // мы не соответствовали совершенно, возвращению ложной матч вернуться ложной; }




# IFDEF DEBUG
        Serial.print ( "\ т"); // вкладка
        Serial.print (offcode); // сигнал OFF мы услышали Serial.print ( "-");

        Serial.print (Сигнал [я * 2 + 1]); // сигнал OFF мы хотим
# ENDIF

        если (ABS (offcode - Сигнал [я * 2 + 1]) <= (Сигнал [г * 2 + 1] * нечеткости / 100)) {
# IFDEF DEBUG
            Serial.print ( "(OK)");
# ENDIF
        } Еще {
# IFDEF DEBUG
            Serial.print ( "(x)");
# ENDIF
            // мы не соответствовали совершенно, возвращению ложной матч вернуться ложной; }



# IFDEF DEBUG
        Serial.println ();
# ENDIF}

    // Все соответствует! возвращает
  истину; }


INT listenForIR (аннулируются) {
    currentpulse = 0;

    в то время как (1) {
        uint16_t HighPULSE, lowpulse; // временного хранения времени HighPULSE = lowpulse = 0; // начать с
        длительностью импульса не

// пока (digitalRead (г.Ирпень)) {// это слишком медленно!
        в то время как (IRpin_PIN & (1 << Ирпеня)) {
```

```
    в то время как (IRpin_PIN & (1 << Ирпеня)) {
        // пин-прежнему высок

        // отсчитать еще несколько микросекунд HighPULSE ++;

        delayMicroseconds (разрешение);

        // Если пульс слишком долго, мы «не истекли» - либо ничего // был получен или код закончен, поэтому
        печати, что // мы схватились до сих пор, а затем сбросить


        // KGO: Добавлена проверка на конец приемного буфера
        если (((HighPULSE> = MaxPulse) && (currentpulse! = 0)) || currentpulse == NUMPULSES) {
            вернуться currentpulse; }}


    // мы не тайм-аут так позволяет копить импульсы чтения [currentpulse] [0] =
    HighPULSE;

    // то же самое, что и выше
    в то время как (! (IRpin_PIN & _BV (г.Ирпень))) {
        // контактный еще НИЗКИЙ
        lowpulse ++;
        delayMicroseconds (разрешение);
          // KGO: Добавлена проверка на конец приемного буфера
          если (! ((lowpulse> = MaxPulse) && (currentpulse = 0)) || currentpulse == NUMPULSES) {вернуться currentpulse; }}



    импульсы [currentpulse] [1] = lowpulse;

    // мы читаем один высокий-низкий пульс успешно, продолжайте! currentpulse ++; }}



недействительный printpulses (недействительный) {
    Serial.println ( "\ n \ r \ n \ rReceived: \ N \ ROFF \ Ton"); для (uint8_t = 0; r <currentpulse; я
    ++) {
        Serial.print (импульсы [I] [0] * РЕШЕНИЕ, декабрь); Serial.print ( "мксек,");

        Serial.print (импульсы [I] [1] * РЕШЕНИЕ, декабрь); Serial.println ( "мксек"); }



    // распечатать его в формате 'массива' Serial.println ( "ИНТ
    IRsignal [] = {");
    Serial.println ( "// ON, OFF (в 10-х микросекунд)"); для (uint8_t = 0; r <currentpulse-1; я ++) {

        Serial.print ( "\ t"); // вкладка
        Serial.print (импульсы [I] [1] * РЕШЕНИЕ / 10, декабрь); Serial.print ( "");

        Serial.print (импульсы [+ 1] [0] * РЕШЕНИЕ / 10, декабрь); Serial.println ( ""); }


    Serial.print ( "\ t"); // вкладка
    Serial.print (импульсы [currentpulse-1] [1] * РЕШЕНИЕ / 10, декабрь); Serial.print ( "0};"); }
```

```
/ ****************** наши коды ********** /

INT ApplePlaySignal [] = {
// ON, OFF (в 10-х микросекунд)
912, 438,
68, 48,
68, 158,
68, 158,
68, 158,
68, 48,
68, 158,
68, 158,
68, 158,
70, 156,
70, 158,
68, 158,
68, 48,
68, 46,
70, 46,
68, 46,
68, 160,
68, 158,
70, 46,
68, 158,
68, 46,
70, 46,
68, 48,
68, 46,
68, 48,
66, 48,
68, 48,
66, 160,
66, 50,
66, 160,
66, 50,
64, 160,
66, 50,
66, 3950,
908, 214,
66, 3012,
908, 212,
68, 0};

ИНТ AppleForwardSignal [] = {// ON, OFF (в 10-х микросекунд)

  908, 444,
  64, 50,
  66, 162,
  64, 162,
  64, 162,
  64, 52,
  64, 162,
  64, 162,
  64, 162,
  64, 164,
  62, 164,
  64, 162,
  64, 52,
  62, 52,
```

```
  62, 52,
  64, 52,
  64, 50,
  64, 164,
  64, 50,
  64, 164,
  64, 162,
  64, 50,
  66, 50,
  66, 50,
  64, 50,
  66, 50,
  64, 52,
  64, 50,
  66, 160,
  66, 50,
  64, 162,
  66, 50,
  64, 162,
  64, 50,
  66, 3938,
  906, 214,
  66, 3014,
  906, 214,
  64, 0};

ИНТ AppleRewindSignal [] = {// ON, OFF (в 10-х микросекунд)

  908, 442,
  66, 48,
  66, 162,
  66, 160,
  66, 160,
  66, 50,
  66, 160,
  66, 160,
  66, 160,
  68, 158,
  68, 160,
  66, 160,
  66, 50,
  66, 48,
  66, 50,
  66, 48,
  66, 162,
  66, 160,
  66, 48,
  68, 48,
  66, 160,
  66, 50,
  66, 50,
  66, 48,
  66, 50,
  66, 48,
  68, 48,
  66, 160,
  66, 50,
  66, 160,
  66, 50,
  66, 160,
```

```
66, 48,
68, 3936,
906, 214,
66, 0};
```

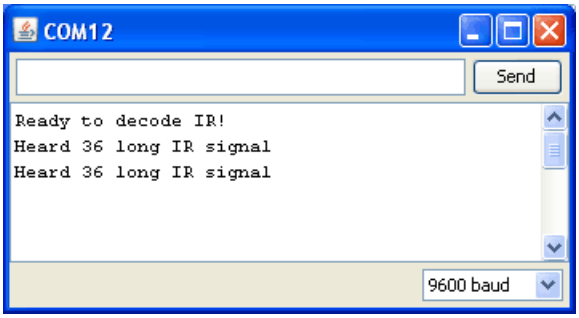
Этот код использует части нашего предыдущего эскиза. Первая часть мы будем сделать, это создать функцию, которая просто прослушивает ИК-кода с помещает тайминги импульсов в импульсы [] массива. Она возвращает число импульсов он услышал, как возвращаемое значение.

```
INT listenForIR (аннулируются) {
    currentpulse = 0;

    в то время как (1) {
        uint16_t HighPULSE, lowpulse; // временного хранения времени HighPULSE = lowpulse = 0; // начать с
        длительностью импульса не

// пока (digitalRead (г.Ирпень)) {// это слишком медленно!
        в то время как (IRpin_PIN & (1 << Ирпеня)) {
            // пин-прежнему высок

            // отсчитать еще несколько микросекунд HighPULSE ++;

            delayMicroseconds (разрешение);

            // Если пульс слишком долго, мы «не истекли» - либо ничего // был получен или код закончен, поэтому
            печати, что // мы схватились до сих пор, а затем сбрасывается, если ((HighPULSE> = MaxPulse) &&
            (currentpulse! = 0)) {

                вернуться currentpulse; }}


        // мы не тайм-аут так позволяет копить импульсы чтения [currentpulse] [0] =
        HighPULSE;

        // то же самое, что и выше
        в то время как (! (IRpin_PIN & _BV (г.Ирпень))) {
            // контактный еще НИЗКИЙ
            lowpulse ++;
            delayMicroseconds (разрешение);
            если ((lowpulse> = MaxPulse) && (currentpulse! = 0)) {
                вернуться currentpulse; }}


        импульсы [currentpulse] [1] = lowpulse;

        // мы читаем один высокий-низкий пульс успешно, продолжайте! currentpulse ++; }}
```

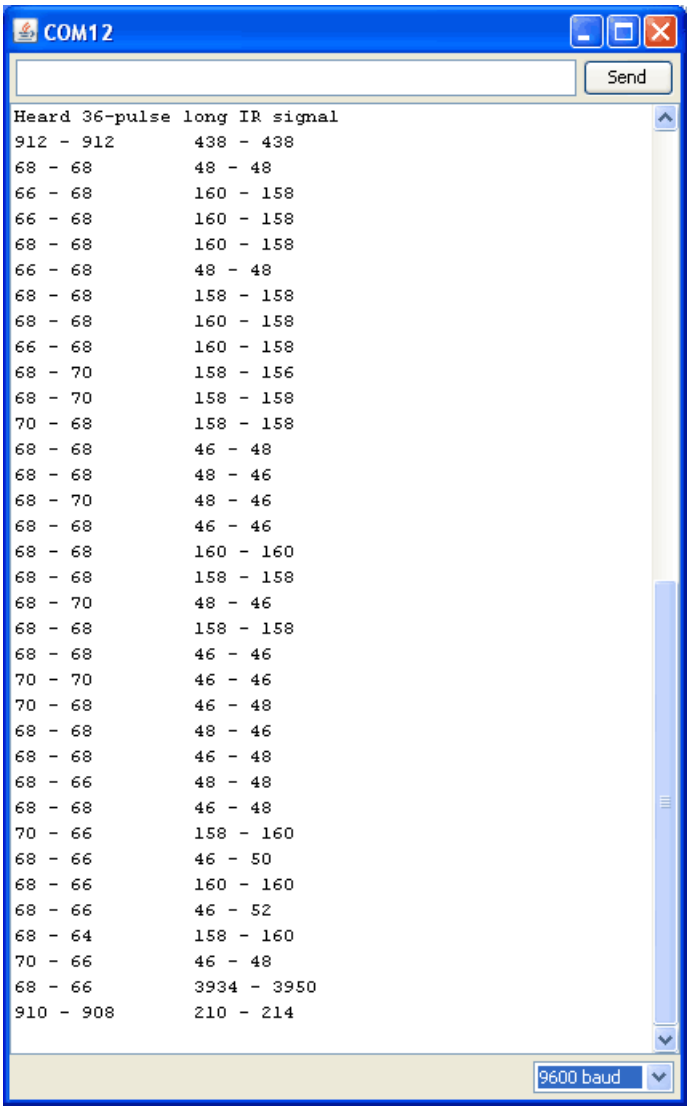
Наш новый цикл () будет начинать только прослушивание импульсов

```
недействительные петли (недействительная) {
    ИНТ numberpulses;

    numberpulses = listenForIR ();

    Serial.print ( "Слышал"); Serial.print
    (numberpulses);
    Serial.println ( "- импульсная длиной ИК-сигнал"); }
```

При запуске этого напечатает что-то вроде ...

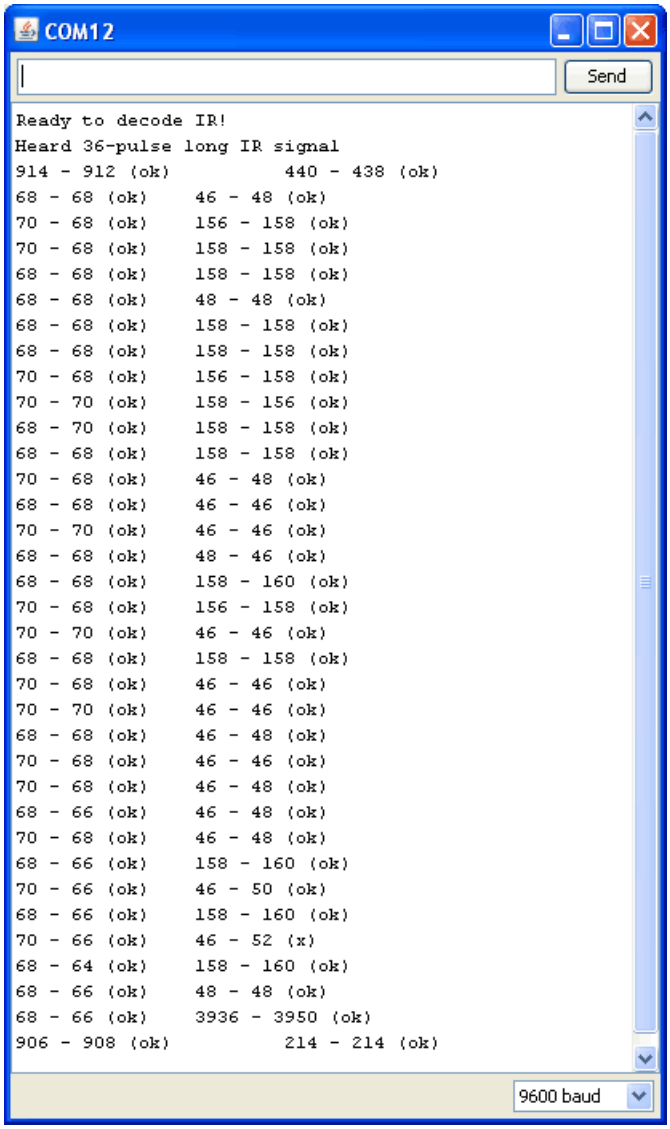


OK время, чтобы сделать эскиз сравнить то, что мы получили то, что мы имеем в нашем хранящемся массиве:



Как вы можете видеть, есть некоторые различия. Поэтому, когда мы делаем наше сравнение мы не можем искать preciesely то же значение, что мы должны быть немного «нечеткими». Мы будем говорить, что значения могут изменяться на 20% - это должно быть достаточно хорошо.

```
// Какой процент мы разрешим в изменении, чтобы соответствовать один и тот же код \\ #define нечеткость 20

недействительные петли (недействительная) {
  ИНТ numberpulses;

  numberpulses = listenForIR ();

  Serial.print ( "Слышал"); Serial.print
  (numberpulses);
  Serial.println ( "- импульсная длиной ИК-сигнал");

  для (INT I = 0; г <numberpulses-1; я ++) {
    INT oncode = импульсы [я] [1] * РЕШЕНИЕ / 10; INT offcode = импульсы [+ 1] [0] *
    РЕШЕНИЕ / 10;

    Serial.print (oncode); // сигнал ON мы услышали Serial.print ( "-");

    Serial.print (ApplePlaySignal [я * 2 + 0]); // сигнал ON мы хотим

    // проверить, чтобы убедиться, ошибка меньше размытости процентов, если (ABS (oncode - ApplePlaySignal [я * 2 + 0]) <= (oncode *
    размытость / 100)) {
      Serial.print ( "(OK)"); } Еще {

      Serial.print ( "(x)"); }

    Serial.print ( "\ т"); // вкладка

    Serial.print (offcode); // сигнал OFF мы услышали Serial.print ( "-");

    Serial.print (ApplePlaySignal [я * 2 + 1]); // сигнал OFF мы хотим

    если (ABS (offcode - ApplePlaySignal [я * 2 + 1]) <= (offcode * нечеткости / 100)) {
      Serial.print ( "(OK)"); } Еще {

      Serial.print ( "(x)"); }


    Serial.println (); }}
```



Этот цикл, как он проходит через каждый импульс, делает немного математики. Он сравнивает разницу абсолютного (абс ()) между кодом, который мы слышали, и код, который мы пытаемся соответствовать абс (oncode - ApplePlaySignal [я * 2 + 0]), а затем убеждается, что погрешность составляет менее нечеткость процентов длина кода (oncode * нечеткости / 100)

Мы обнаружили, мы должны были настроить сохраненные значения немного, чтобы они соответствовали до 100% каждый раз. ИК не precisiontimed протокол так того, чтобы сделать нечеткости 20% или более не плохо

Наконец, мы можем превратить петлю () в свою собственную функцию, которая возвращает истину или ложь в зависимости от того соответствует ли код мы спрашиваем его. Мы также закомментирована функции печати

```
булево IRcompare (интермедиат numpulses, Int Signal []) {

  для (INT I = 0; г <numpulses-1; я ++) {
    INT oncode = импульсы [я] [1] * РЕШЕНИЕ / 10; INT offcode = импульсы [+ 1] [0] *
    РЕШЕНИЕ / 10;

    / *
    Serial.print (oncode); // сигнал ON мы услышали Serial.print ( "-");

    Serial.print (Сигнал [я * 2 + 0]); // сигнал ON мы хотим
    * /

    // проверить, чтобы убедиться, ошибка меньше размытости процентов, если (ABS (oncode - Сигнал [я * 2 + 0]) <= (Сигнал [я * 2 + 0] *
    размытость / 100)) {
      //Serial.print ( "(OK)"); } Еще {

      //Serial.print ( "(x)");
      // мы не соответствовали совершенно, возвращению ложной матч вернуться ложной; }



    / *
    Serial.print ( "\ т"); // вкладка
    Serial.print (offcode); // сигнал OFF мы услышали Serial.print ( "-");

    Serial.print (Сигнал [я * 2 + 1]); // сигнал OFF мы хотим
    * /

    если (ABS (offcode - Сигнал [я * 2 + 1]) <= (Сигнал [г * 2 + 1] * нечеткости / 100)) {
      //Serial.print ( "(OK)"); } Еще {

      //Serial.print ( "(x)");
      // мы не соответствовали совершенно, возвращению ложной матч вернуться ложной; }



    //Serial.println (); }

  // Все соответствует! возвращает
истину; }
```

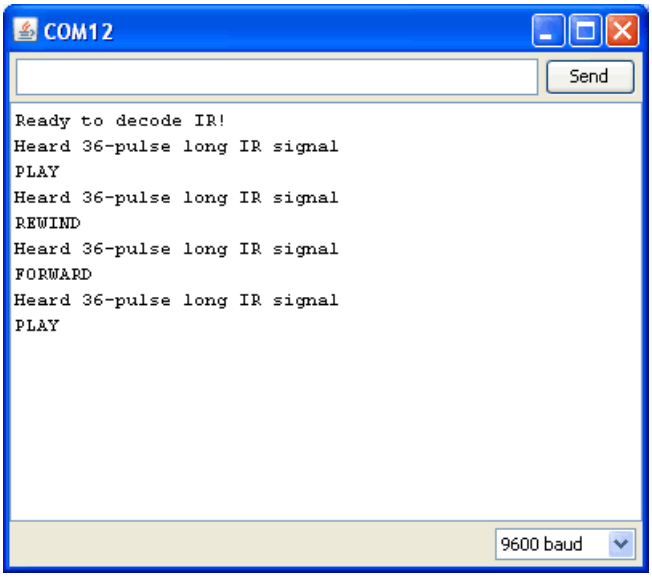
Затем мы взяли больше данных команд IR для кнопок и «перемотки» «FastForward» и поместить все данные кода массива в ircodes.h **сохранить основной эскиз от слишком долго, и нечитаемых** (Вы можете получить весь код с GitHub) (Https://adafru.it/aKg)

Наконец, главный цикл выглядит следующим образом:

```
недействительные петли (недействительная) {
    ИНТ numberpulses;

    numberpulses = listenForIR ();

    Serial.print ( "Слышал"); Serial.print
    (numberpulses);
    Serial.println ( "- импульсная длиной ИК-сигнал"); если (IRcompare (numberpulses,
    ApplePlaySignal)) {
        Serial.println ( "PLAY"); }

        если (IRcompare (numberpulses, AppleRewindSignal)) {Serial.println ( "перемотку"); }


        если (IRcompare (numberpulses, AppleForwardSignal)) {Serial.println ( "ВПЕРЕД"); }}
```

Мы проверяем против всех кодов мы знаем о и распечатать всякий раз, когда мы получаем матч. Теперь Вы можете взять этот код и превратить его в нечто иное, как робот, который движется в зависимости от того, что кнопка нажата.

После тестирования, успех!



CircuitPython

С CircuitPython вы можете легко прочитать ИК импульсы датчика из кода Python. Встроенный в CircuitPython является специальным `pulseio` модуль, который фактически делает большую часть работы, чтения быстрых импульсов ИК-приемник для вас. Еще лучше с кодом Python вы можете очень легко хранить и обрабатывать большие списки длительности импульсов. Там даже удобно Adafruit CircuitPython IRRemote (Https://adafru.it/BBm) модуль, который упрощает некоторые из логики обработки для чтения общих пультов дистанционного управления. CircuitPython делает его очень легко читать ИК-сигналы!

Оборудование и установки

Для чтения необработанных ИК-сигналов вам необходимо подключить ИК-датчик к вашей плате, как показано на предыдущих страницах. В этом примере мы предполагаем, что выходной сигнал датчика подключается к контакту D2 на доске.

Как уже упоминалось вам также необходимо установить Adafruit CircuitPython IRRemote (Https://adafru.it/BBm) библиотека на доске CircuitPython.

Сначала убедитесь, что вы производите Последняя версия Adafruit CircuitPython (Https://adafru.it/Amd) для доски.

Далее вам необходимо установить необходимые библиотеки, чтобы использовать аппаратные средства - внимательно следуйте инструкциям, чтобы найти и установить эти библиотеки из библиотека расслоение CircuitPython Adafruit в (Https://adafru.it/zdx) , Наше руководство имеет введение большая страница о том, как установить пакет библиотеки (Https://adafru.it/ABU) как для экспресса и не экспресс доски.

Помните, для не-экспресса плата, таких как, вам нужно вручную установить необходимые библиотеки из пакета:

- adafruit_irremote.mpy

Или загрузите файл последней версии на Adafruit CircuitPython IRRemote выпускает страницу (Https://adafru.it/BBn) ,

Перед продолжением убедитесь, что Lib папку или корневой файловой системы вашего платы имеет adafruit_irremote.mpy модуль скопировали.

использование

следующий подключения к последовательному РЕПЛ платы (Https://adafru.it/Awz) так что вы находитесь в CircuitPython >>> строке.

Затем импортировать необходимые платы и pulseio модули:

```
импорт доска импорта
pulseio
```

Теперь создадим экземпляр класса класс PulseIn (Https://adafru.it/BBo) который считывает импульсы с выхода ИК-датчика. Импульс просто переход от высоких к низким или наоборот, и классу PulseIn будет записывать микросекундную длительность каждого импульса. Давайте создадим импульсный вход, который может вспомнить продолжительность до 200 импульсов (достаточно для записи большинства кодов ДУ):

```
Импульсы = pulseio.PulseIn (board.D2, MaxLen = 200, idle_state = True)
```

Давайте разберем все параметры, передаваемые в инициализатор PulseIn:

- Совет контактный - Это обязательный параметр, который указывает, какой вывод подключен к выходу приемника ИК. MaxLen - указывает число длительностей
- импульсов записи. Для большинства пультов Д значение 200 будет более чем достаточно длительностей импульсов в магазин. Если установить слишком высоко вы можете использовать больше памяти, чем ваш совет уже доступны, поэтому будьте осторожны с тем, что вы цените выбрать.

- idle_state - это логическое значение, которое указывает «по умолчанию» или состояние ожидания импульсного штифта. Для получения ИК-приемников они, как правило, в режиме ожидания высокой логики или истинное положение так, устанавливающего idle_state к истинному показывает нормальное состояние высокого логического уровня.

Если у вас есть объект входного импульса вы можете взаимодействовать с ним, как если бы это был список значений длительности. Внутри класс PulseIn всегда прослушивает импульсы от штифта (т.е. перехода от текущего высокого / низкого логического уровня к противоположному уровне) и сохранения длительности импульса. Вы можете перечислить то количество принятых импульсов так же, как чтение длины списка:

```
LEN (импульсы)
```

```
>>> len(pulses)
0
>>>
```

Нулевое значение означает, что датчик еще не получил импульс. Попробуйте направить пульт дистанционного управления на датчик и нажать кнопку. Потом снова прочитать длительность импульса:

```
LEN (импульсы)
```

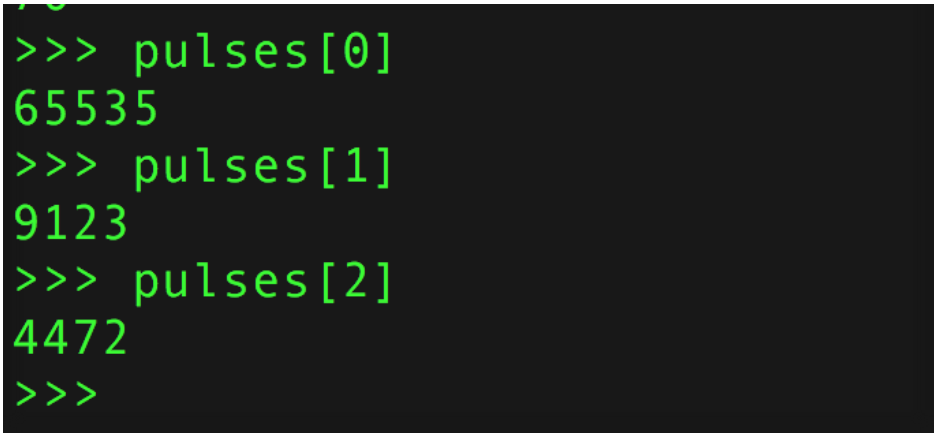
```
>>> len(pulses)
70
>>>
```

Теперь у нас есть несколько длительностей импульсов для расследования! Сначала давайте скажем класс импульсов для временной остановки прослушивания для импульсов. Это полезно, так что вы можете работать на последнем видели импульс без других импульсов добавляя больше шума или артефакты:

```
pulses.pause ()
```

Теперь исследуем некоторые из длительности импульсов путем считывания значения, если объект импульса был список. Например, чтобы прочитать первые три длительностей:

```
импульсов [0]
импульсов [1]
импульсов [2]
```

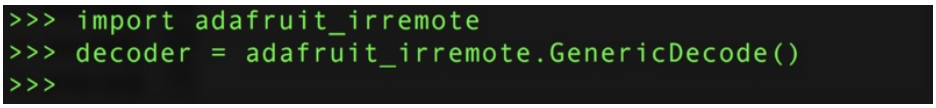


Каждая продолжительность времени в миллисекундах, что импульс был на определенном логическом уровне. Самое первый импульс представляет собой максимальное значение 65535, поскольку она представляет собой количество времени, датчик был ждет импульса запуска (то есть, как долго он был в уровне по умолчанию высоких логического состояния простоя). Так же, как с кодом Arduino на предыдущей странице вы можете игнорировать это первое значение.

Следующие два значения интересны, следующее значение импульса показывает датчик получает импульс, который был длиной около 9 миллисекунд (или ~ 9000 микросекунд). Затем датчик не получил импульс в течение около 4 мс. Эта пара значений представляет собой одиночный импульс и начало сигнала дистанционного управления. Это хорошо, чтобы увидеть значение ~ 9ms на и ~ 4м от, как это общая преамбула или начать для ИК-кодов!

Оказывается, эти пары импульсов настолько распространены между различными пультами дистанционного управления, что многие из них могут быть считаны с аналогичным кодом. Библиотека Adafruit CircuitPython IRRemote является очень простым ИК-пульта дистанционного управления библиотекой декодирования, который упрощает большую часть импульса и удаленной логики декодирования. Давайте использовать этот модуль для упрощения анализа импульсов, первый импортировать его, а затем создать удаленный декодер:

```
импорт adafruit_irremote
Декодер = adafruit_irremote.GenericDecode ()
```



Класс декодер позволяет легко ждать и прочитать список импульсов от управления нажмите дистанционного. Перед тем, как использовать это позволяет включить импульсный вход обратно (помните, что в настоящее время приостановлено) и очистить его предыдущий вход:

```
pulses.clear ()
pulses.resume ()
```

Теперь мы готовы использовать декодер ждать и возврата импульсов. Выполнив этот код и обратите внимание на РЕПЛЕ останавливается и ждет дальнейшего ввода:

```
Пульс = decoder.read_pulses (импульсы)
```

```
>>> pulse = decoder.read_pulses(pulses)
```

Цель вашего пульта дистанционного управления на приемник и нажмите кнопку. Вы должны увидеть REPL возврата к обычному режиму работы. Это означает, что декодер мог обнаружить сигнал дистанционной ИК и возвратил сырой список значений импульсов.

```
>>> pulse = decoder.read_pulses(pulses)
>>>
```

Этот перечень импульсов представляет собой массив, который содержит длину в микросекундах каждого высокого и низкого импульс от приемника. Например, вы можете проверить, сколько импульсов изменения были обнаружены и видеть их длины, используя стандартные операции длины массива и печати:

```
Len (пульс)
импульсы
```

```
>>> len(pulse)
67
>>> pulse
[9144, 4480, 602, 535, 600, 540, 595, 536, 599, 537, 600, 536, 596, 540, 595, 544, 591,
539, 596, 1668, 592, 1676, 593, 1667, 593, 1674, 596, 1670, 590, 1674, 595, 535, 590, 16
73, 597, 541, 595, 536, 597, 538, 597, 538, 597, 1666, 594, 541, 594, 541, 594, 540, 595
, 1668, 596, 1673, 592, 1668, 592, 1672, 601, 540, 592, 1669, 590, 1672, 598, 1667, 593]
>>>
```

Одна очень полезная вещь декодер делает внутренне обнаруживает и игнорируя шум или постороннее длительность импульса, как долго, начиная длительность импульса до того, как пульт дистанционного управление обнаружено. Это очень полезно, поскольку это упрощает код обработки ИК - вы можете сосредоточиться только на глядя на «очищенном» длительности импульса!

Попытка записи второго импульса:

```
pulse2 = decoder.read_pulses (импульсы)
```

```
>>> pulse2 = decoder.read_pulses(pulses)
```

Помните, что функция read_pulses будет ждать управления пресс дистанционного быть обнаружены (или если один ранее произошло, и не был обработан он будет захватить его, а). Нажмите ту же кнопку на пульте дистанционного управления, чтобы генерировать подобный импульс в качестве первого пресса:

```
>>> pulse2 = decoder.read_pulses(pulses)
>>>
```

Теперь давайте сравним первый и второй список импульсов, чтобы увидеть, если они совпадают. Простое сравнение может быть, чтобы проверить каждое значение в каждом списке и убедитесь, что они одинаковы. Давайте попробуем это с простой функцией Python мы определяем:

```
Защиту simple_pulse_compare (pulse1, pulse2):
    если Len (pulse1) = Len (pulse2):
        вернуться Ложные
    для г в диапазоне (LEN (pulse1)):
        если pulse1 [я] = pulse2 [я]:
            вернуться Ложные
    возвращающие

simple_pulse_compare (пульс, pulse2)
```

```
>>> def simple_pulse_compare(pulse1, pulse2):
...     if len(pulse1) != len(pulse2):
...         return False
...     for i in range(len(pulse1)):
...         if pulse1[i] != pulse2[i]:
...             return False
...     return True
...
>>> simple_pulse_compare(pulse, pulse2)
False
>>>
```

О, нет, сравнение не удалось, и вернулся ложь! То, что произошло, было не то же кнопка нажата? Оказывается, выбор времени между импульсами может варьироваться в небольших способами. Если посмотреть на отдельные длины импульса каждого массива вы увидите, что они близки, но не совсем то же самое. Если сравнить исходные импульсы вам нужно добавить «fuzzyness», который сравнивает значения, которые близки, но не совсем то же самое.

Давайте сделаем новый нечеткая функция сравнения, которая будет проверять наличие импульсов, которые близки друг к другу (в пределах 20% друг от друга, например):

```
Защиту fuzzy_pulse_compare (pulse1, pulse2, fuzzyness = 0,2):
    если Len (pulse1) = Len (pulse2):
        вернуться Ложные
    для г в диапазоне (LEN (pulse1)):
        порог = INT (pulse1 [я] * fuzzyness) если абс (pulse1 [я] - pulse2 [я])> порог:

            вернуться Ложные
    возвращающие

fuzzy_pulse_compare (пульс, pulse2)
```

```
>>> def fuzzy_pulse_compare(pulse1, pulse2, fuzzyness=0.2):
...     if len(pulse1) != len(pulse2):
...         return False
...     for i in range(len(pulse1)):
...         threshold = int(pulse1[i] * fuzzyness)
...         if abs(pulse1[i] - pulse2[i]) > threshold:
...             return False
...     return True
...
>>> fuzzy_pulse_compare(pulse, pulse2)
True
>>>
```

Успех! Оба импульса, как представляется, то же самое при использовании нечеткого сравнения. По умолчанию сравнение будет рассматривать импульсы так же, если они в пределах 20% друг от друга, но вы можете изменить это fuzzyness, установив fuzzyness

Ключевое слово на другое значение. Значение fuzzyness представляет собой процент от 0 до 1,0 (или от 0 до 100%), где импульсы должны находиться в пределах этого процента времени друг друга. Более низкие значения являются более жесткими и требуют более аналогичных импульсов, в то время как более высокие значения являются менее строгими, и может позволить шума или неправильные импульсы выглядят одинаково. В общем палочке с 20% fuzzyness, если не работать в более проблематичных ИК сигналов.

Давайте связать все вместе, делая полную программу, которая ждет кнопку выше, чтобы быть нажата и выводит сообщение.

Вы можете использовать список записанных импульсов в вашей программе, чтобы вспомнить ранее записанный пульс и сравнить новые против него. Для того, чтобы обнаружить различные нажатие клавиши просто записать его с указанными выше шагами и обновить список импульсов в коде.

Изменение списка импульсов в верхнем в коде ниже значение записанного (просто скопировать и вставить его из РЕПЛ) и сохранить его в качестве code.py на доске:

https://adafru.it/Et8

https://adafru.it/Et8

```
импорт доска импорта
pulseio
импорт adafruit_irremote

IR_PIN = board.D2 # контактные подключены к ИК-приемнику.

#  Ожидаемое пульс, вставили в предыдущей сессии из записи REPL: импульсный = [9144, 4480, 602, 535, 600, 540, 595,
536, 599, 537, 600, 536,
                596, 540, 595, 544, 591, 539, 596, 1668, 592, 1676, 593, 1667,
                593, 1674, 596, 1670, 590, 1674, 595, 535, 590, 1673, 597, 541,
                595, 536, 597, 538, 597, 538, 597, 1666, 594, 541, 594, 541, 594,
                540, 595, 1668, 596, 1673, 592, 1668, 592, 1672, 601, 540, 592,
                1669, 590, 1672, 598, 1667, 593]

Печать ( «ИК слушатель»)
#  Нечеткая функция сравнения импульсов:
Защиту fuzzy_pulse_compare (pulse1, pulse2, fuzzyness = 0,2):
    если Len (pulse1) = Len (pulse2):
        вернуться Ложные
    для г в диапазоне (LEN (pulse1)):
        порог = INT (pulse1 [я] * fuzzyness) если абс (pulse1 [я] - pulse2 [я])> порог:

            вернуться Ложные
    возвращающие

#  Создать импульсный вход и ИК-декодер.
Импульсы = pulseio.PulseIn (IR_PIN, MaxLen = 200, idle_state = True) Декодер =
adafruit_irremote.GenericDecode () pulses.clear () pulses.resume ()


#  Цикл ожидания приема импульсов. в то время как True:

    #  Подождите, импульс, чтобы быть обнаружены. Обнаруженные =
    decoder.read_pulses (импульсы) печати ( 'получил импульс ...')

    #  Получили импульс, теперь сравнить.
    если fuzzy_pulse_compare (пульс, обнаружен):
        печать ( «Полученное правильное управление пресс дистанционного!»)
```

Теперь при нажатии на кнопку пульта дистанционного управления, вы должны увидеть сообщение, напечатанное в РЕПЛ! Вот все, что основной сырьевой ИК обнаружения импульсов и сравнение с CircuitPython!

Код на этой странице может быть удобен для основного или неизвестного обнаружения протокола дистанционного управления. Однако следует помнить, что пульт дистанционного управления, на самом деле довольно продвинутый и иногда ведут себя не так, как вы ожидаете - как нажатие на кнопку несколько раз, возможно, не на самом деле отправить полный код каждый раз, вместо того, чтобы пульт мог бы послать более короткий код повтора! Это означает, что основное обнаружение сырой ИК показанное здесь могут не потому, что он не ожидает, что кода повторения, когда один виден.

Оказывается, общий ИК-пульт дистанционного обнаружение настолько совершенно, что лучше всего обрабатывается отдельной библиотекой, которая может декодировать повторяющиеся коды и **многие другие. Для CircuitPython ознакомьтесь модуль IRLibCP (Https://adafru.it/BBp) от Крис Янг, она имеет гораздо более полнофункциональный ИК поддержку удаленного декодирования!**

Python Docs

Python Docs (Https://adafru.it/C54)