# Универсальный пульт управления из старого телефона


- Платформы: Iskra Neo / Arduino Leonardo
- Языки программирования: C++
- Версия Arduino IDE: 1.6.9
- Тэги: умный дом, ИК-управление, пульт дистанционного управления, bluetooth, RemoteXY, android.

## Что это?

У каждого есть целая коллекция пультов управления от бытовых приборов. Часто мы нажимаем одни и те же последовательности кнопок сначала на одном пульте, затем на другом. Было бы здорово иметь для таких случаев только одну кнопку. А чтобы не искать эту кнопку по всей квартире, мы поместим её в смартфон, который всегда с собой.

## Что нам понадобится?

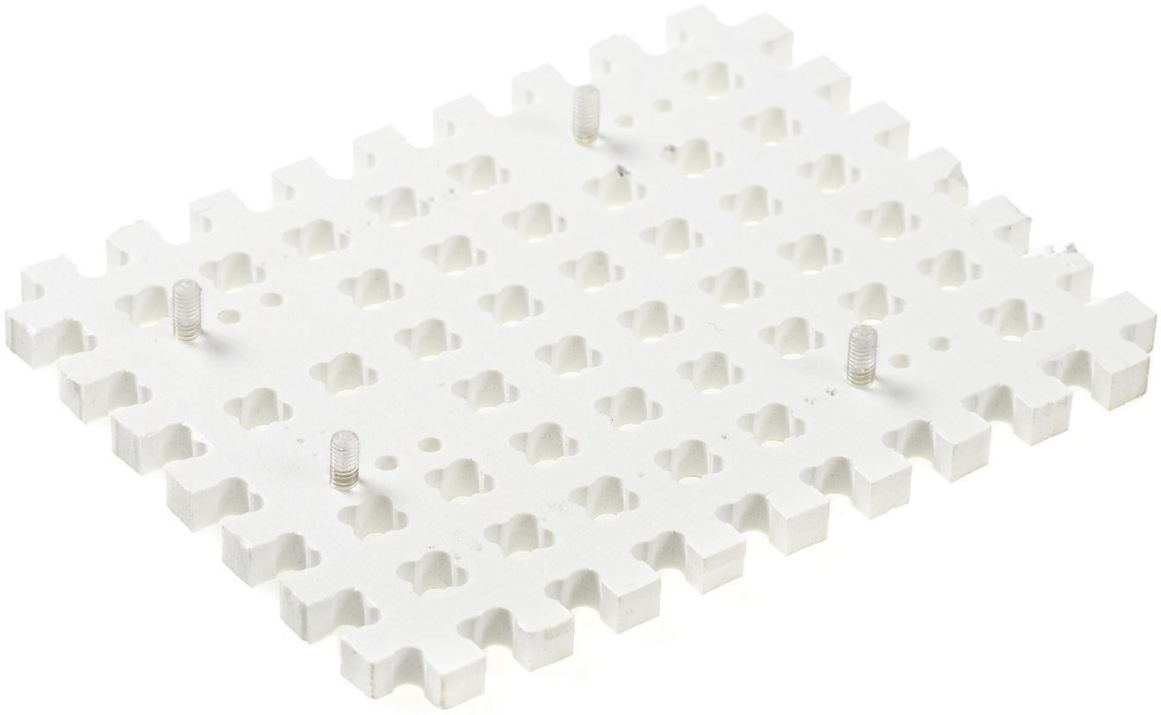
1. Iskra Neo
2. Troyka Slot Shield
3. Bluetooth HC-05 (Troyka-модуль)
4. ИК-передатчик (Troyka-модуль)
5. ИК-приёмник (Troyka-модуль)
6. Slot Box (#Структор)
7. Кабель USB (A — Micro USB)
8. Импульсный блок питания с USB-разъёмом (5 В, 2100 мА)
9. 8 × Винт M3
10. Канцелярский нож
11. Android-смартфон
12. Приложение RemoteXY

Все электронные компоненты собраны в коллекцию «ИК-репитер»

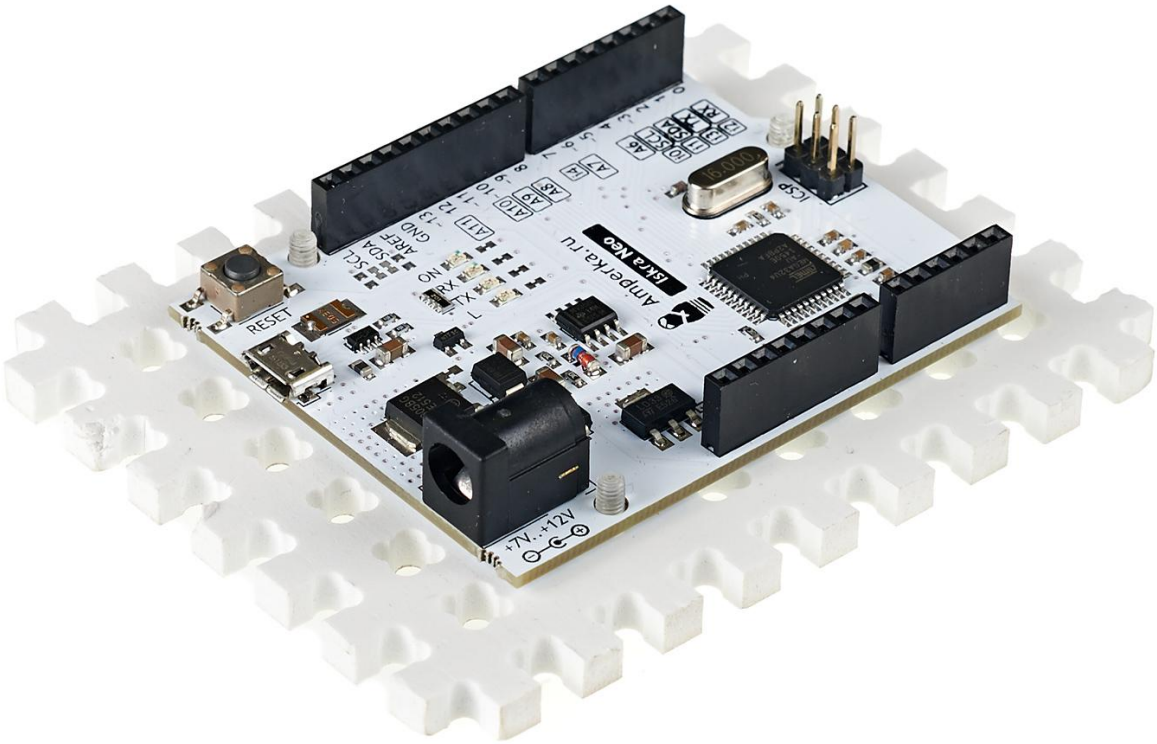
# Как собрать?

1. Используя канцелярский нож, извлеките детали #Slot Box-а.

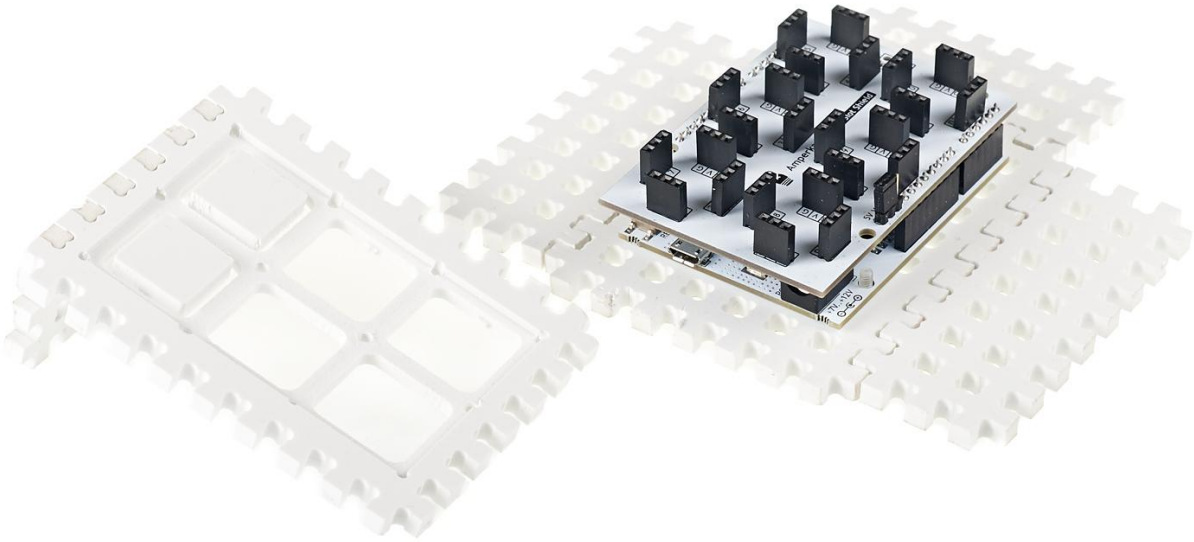
2. Вкрутите винты М3 в плашку #структора для крепления Arduino.



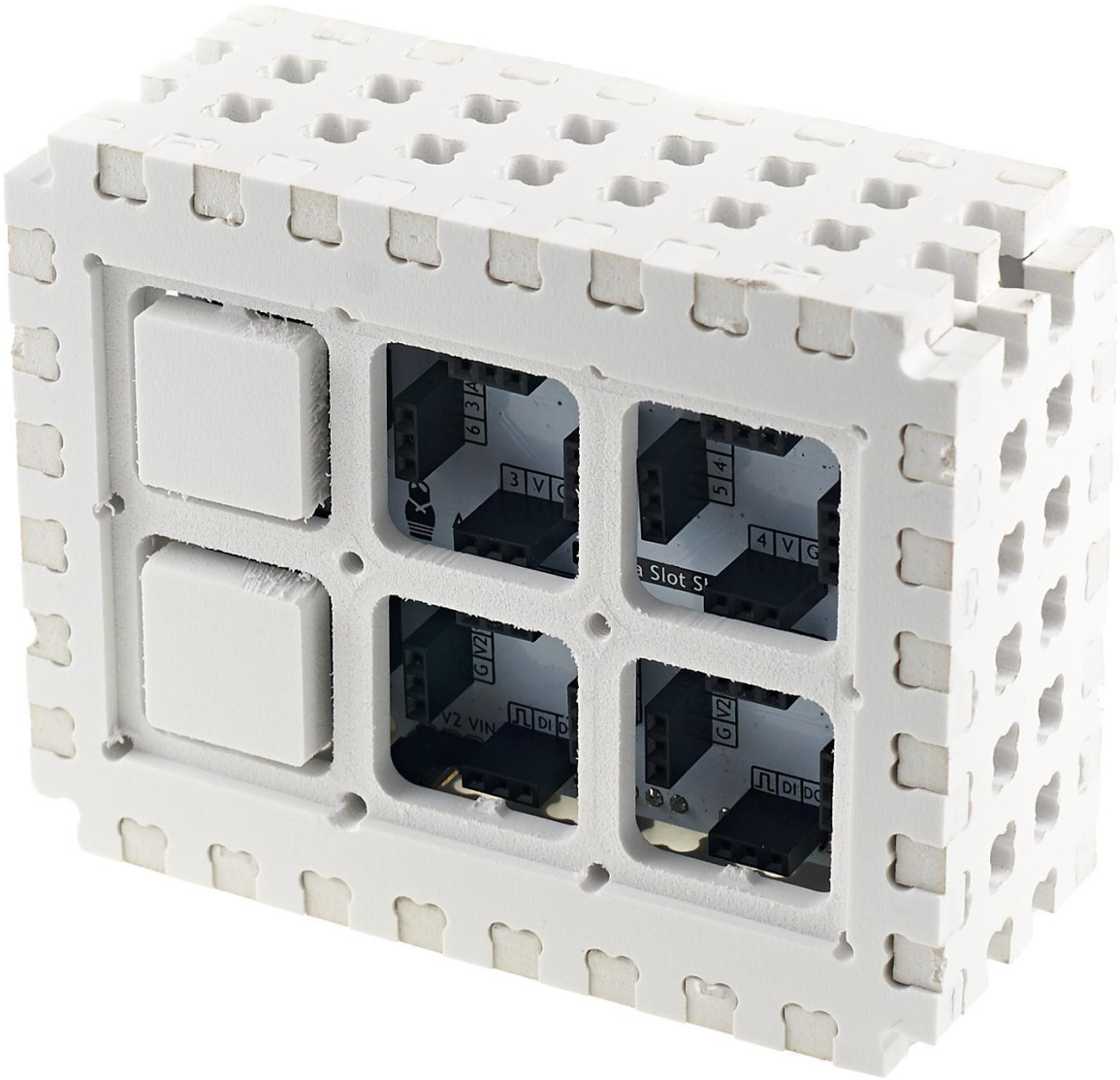
3. Разместите Iskra Neo на винтах



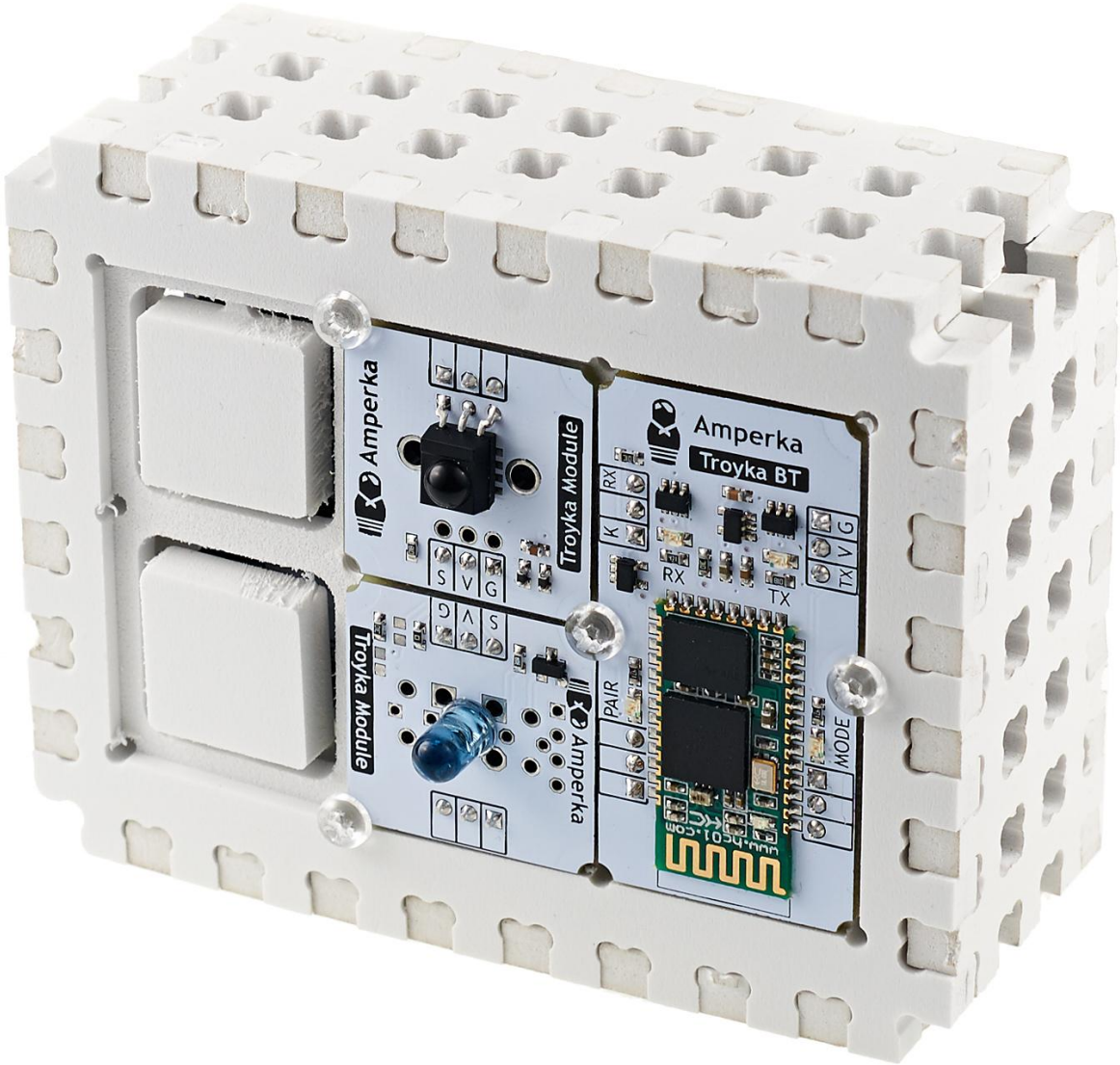
4. Установите Troyka Slot Shield на Arduino. Установите боковые стенки корпуса.



5. Установите лицевую панель на #Slot Box



6. Соедините ИК-приёмник с пином `3`, ИК-передатчик с пином `9`. Соедините пин `TX` Bluetooth-модуля с пином `0`, а контакт `RX` с пином `1`. Закрепите модули винтами М3




# Исходный код

Мы разделили код по функционалу на отдельные файлы. Проще всего скачать проект целиком из репозитория на GitHub.

### Скетч

IR-repeater.ino

```
// Подключаем код для работы ИК-приёмника и передатчика
#include "IRremote_header.h"
// Подключаем код для работы с приложением remoteXY
#include "remoteXY_header.h"
// Подключаем код с логикой работы с энергонезависимой памятью
#include "eeprom_logic.h"
// Подключаем код с логикой работы с ИК-командами
#include "IRremote_logic.h"

// Переменная для хранения состояния "запись ИК-сигналов"
bool recordState = false;

// Переменная для хранения состояния кнопки записи
```

```
bool recButtonLastState = false;

// Переменная для хранения номера последеней нажатой кнопки на телефоне
byte buttonLast = 0;

void setup() {
  // Инициализируем remoteXY
  RemoteXY_Init ();
  // Запускаем работу ИК-приёмника (IRremote_header.h)
  irrecv.enableIRIn();
  // Если у нас есть записанные комбинации ИК-сигналов - достаём их
(eeprom_logic.h)
  eepromDataInit();
}

void loop() {
  // Опрашиваем remoteXY. Не пришли ли новые команды по bluetooth?
  RemoteXY_Handler ();
  // Запоминаем состояние кнопки record на смартфоне.
  bool recButtonState = RemoteXY.record;
  // Если состояние кнопки изменилось...
  if (recButtonLastState != recButtonState) {
    // ... запомним новое состояние этой кнопки.
    recButtonLastState = recButtonState;
    // Если кнопка record нажата...
    if (recButtonState) {
      // меняем состояние "запись ИК-сигналов" на противоположное
      recordState = !recordState;
    }
    // Если мы вошли в состояние записи...
    if (recordState) {
      // ... зажгём красный "светодиод" на смартфоне.
      RemoteXY.recMode_r = 255;
      // Если мы вышли из состояния записи
    } else {
      // ... сохраним записанные данные ИК-пульта
      saveAll();
      // и потушим светодиод
      RemoteXY.recMode_r = 0;
    }
  }
  // Если на смартфоне была нажата новая кнопка...
  if (RemoteXY.buttons != buttonLast) {
    // ... сохраним её значение.
    buttonLast = RemoteXY.buttons;
    // Еслим мы в состоянии записи — сохраним записанные данные ИК-
пульта,
    // если нет — отправим ИК-сигналы, соответствующие этой кнопке
(IRremote_logic.h)
    recordState ? saveAll() : sendButton(buttonLast);
  }
  // Если мы в состоянии записи, и пришла новая команда с ИК-пульта...
  if ((recordState) && (irrecv.decode(&results))) {
    // сохраним команду с ИК-пульта (IRremote_logic.h) в
последовательность команд текущей кнопки
    storeCode(&results, buttonLast);
    // перезапустим ИК-приёмник
    irrecv.resume();
  }
}
// Функция для сохраняем данных с ИК-пульта
void saveAll() {
  // Если была принята хотя бы одна команда с ИК-пульта
  if (macroPos > 0) {
```

```
    // записываем принятую последовательность команд (eeprom_logic.h)
    saveIRToEEPROM();
    // обнуляем счётчик команд
    macroPos = 0;
  }
}
```

### Код инициализации ИК-приёмника и передатчика

IRremote_header.h

```
// Библиотека для работы с ИК-приёмником и передатчиком
#include <IRremote.h>

// Имя пина, которому подключен приёмник
#define RECV_PIN 3
// Имя пина, к которому подключен передатчик
#define SEND_PIN 9

// Максимальная длина последовательности комманд ИК-пульта
#define IR_BUTTONS_COUNT 15
// Максимальное количество таких последовательностей
// Должно быть равно количеству кнопок в remoteXY
#define MACROS_COUNT 5

// Создаём объект для ИК-приёмника
IRrecv irrecv(RECV_PIN);

// Создаём объект для ИК-передатчика
IRsend irsend;

// Создаём объект для декодированных команд с ИК-пульта
decode_results results;

// Структура для хранения команд с ИК-пульта
struct irData {
  // Тип принятой команды: UNKNOWN, NEC, SONY, RC5, ...
  decode_type_t           decode_type = UNKNOWN;
  // Адрес устройства. Используется в пультах Panasonic и Sharp
  unsigned int            address = 0;
  // Значение декодированной ИК-команды
  unsigned long           value = 0;
  // Длина команды в битах
  int                     bits = 0;
};

// Массив для хранения последовательностей ИК-команд
irData irSygnals[MACROS_COUNT][IR_BUTTONS_COUNT];
// Счётчик принятых команд в текущей последовательности
byte macroPos = 0;
```

### Логика работы с ИК-командами

IRremote_logic.h

```
// Функция для отправки последовательности команд
// принимает на вход номер последовательности
void sendButton(byte button)
{
  // Переменная для хранения бита переключения. Нужна в протоколах RC5,
RC6
  int toggle = 0;

  // Перебираем в цикле всю последовательность команд
  for (int i = 0; i < IR_BUTTONS_COUNT; ++i) {
```

```
    // Сохраняем текущую команду во вспомогательные переменные
    decode_type_t decode_type = irSygnals[button][i].decode_type;
    unsigned int address = irSygnals[button][i].address;
    unsigned long codeValue = irSygnals[button][i].value;
    int codeLen = irSygnals[button][i].bits;

    // Если протокол команды известен
    if (decode_type != UNKNOWN) {

      // Если протокол команды RC5 или RC6
      if (decode_type == RC5 || decode_type == RC6) {
        // Инвертируем бит переключения,
        toggle ^= 1;
        // обнулим бит переключения в самом значении команды,
        codeValue = codeValue & ~(1 << (codeLen - 1));
        // и запишем в это место наш бит переключения... Вот такой уж
протокол:)
        codeValue = codeValue | (toggle << (codeLen - 1));
        // Если протокол RC5,
        if (decode_type == RC5) {
          // отправим ИК-команду по протоколу RC5
          irsend.sendRC5(codeValue, codeLen);
          // Если протокол RC6,
        } else {
          // отправим ИК-команду по протоколу RC6
          irsend.sendRC6(codeValue, codeLen);
        }
      }
      // В зависимости от протокола команды, используем соответствующие
      // этому протоколу функции из библиотеки IRremote
      switch (decode_type)
      {
        case NEC:
          irsend.sendNEC(codeValue, codeLen);
          break;
        case SONY:
          irsend.sendSony(codeValue, codeLen);
          break;
        case PANASONIC:
          irsend.sendPanasonic(address, codeValue);
          break;
        case JVC:
          irsend.sendJVC(codeValue, codeLen, false);
          break;
        case SAMSUNG:
          irsend.sendSAMSUNG(codeValue, codeLen);
          break;
        case WHYNTER:
          irsend.sendWhynter(codeValue, codeLen);
          break;
        case LG:
          irsend.sendLG(codeValue, codeLen);
          break;
        case DISH:
          irsend.sendDISH(codeValue, codeLen);
          break;
        case SHARP:
          irsend.sendSharp(address, codeValue);
      }
      // Подождём выполнения команды бытовыми приборами
      delay(1000);
      // если команда неизвестна — выходим из функции.
    } else {
```

```
      return;
    }
  }
}

// Функция для сохранения принятой ИК-команды
// принимает на вход декодированную ИК-команду, номер текущей
последовательности
// (т.е. номер нажатой кнопки) и позицию команды в текущей
последовательности
void storeCode(decode_results *results, int buttonNumber)
{
  // Если позиция команды меньше, чем максимальное количество команд в
последовательности
  if (macroPos < IR_BUTTONS_COUNT) {
    // Если протокол команды известен, и это не команда повтора команды
    if ((results->decode_type != UNKNOWN) && (results->value !=
REPEAT)) {
      // Сохраняем команду в массив хранения последовательностей команд
      irSygnals[buttonNumber][macroPos].decode_type = results-
>decode_type;
      irSygnals[buttonNumber][macroPos].address = results->address;
      irSygnals[buttonNumber][macroPos].value = results->value;
      irSygnals[buttonNumber][macroPos].bits = results->bits;
      //увеличиваем счётчик принятых комманд
      macroPos += 1;
       // Пометим следующую команду как неизвестную, чтобы
      // остановится на этом месте в функции отправки
последовательностей команд
      if (macroPos < IR_BUTTONS_COUNT)
        irSygnals[buttonNumber][macroPos].decode_type = UNKNOWN;
    }
  }
}
```

### Логика работы с энергонезависимой памятью

eeprom_logic.h

```
// Подключаем библиотеку для работы с энергонезависимой памятью
#include <EEPROM.h>

// Функция для загрузки сохранённых последовательностей ИК-команд
void eepromDataInit() {
  // Волшебное число
  int notFirstStartVal = 12345;
  // Переменная для считанного из EEPROM волшебного числа
  int val;
  // Если число, считанное по адресу 0 не равно волшебному...
  if (EEPROM.get(0, val) != notFirstStartVal) {
    // ... значит мы только что прошили скетч, и в EEPROM лежит мусор
    // Кладём в EEPROM волшебное число по адресу 0
    EEPROM.put(0, notFirstStartVal);
    // Кладём в EEPROM пустой массив для хранения последовательностей
ИК-команд
    // начиная с адреса 0 + размер волшебного числа
    EEPROM.put(sizeof(notFirstStartVal), irSygnals);
  } else {
    // Если волшебное число лежит в EEPROM, значит в EEPROM не мусор
    // Достаём записанные последовательности ИК-команд
    EEPROM.get(sizeof(notFirstStartVal), irSygnals);
  }
}
```

```
// Функция для записи последовательностей ИК-команд в энергонезависимую
память
void saveIRToEEPROM()
{
  // Записываем последовательности, начиная с адреса 0 + размер
волшебного числа
  EEPROM.put(sizeof(int), irSygnals);
}
```

### Автоматически сгенерированный код редактора RemoteXY

RemoteXY предоставляет инструмент online-конструктора интерфейсов для смартфона. Это кнопки, тумблеры, джойстики и другие элементы управления. Конструктор нашего интерфейса можно посмотреть и отредактировать на свой вкус по ссылке.

remoteXY_header.h

```
// Файл автоматически сгенерирован online-конструктором
// форм для смартфона remoteXY
// http://remotexy.com/en/editor/a1cf60078a90eb8434b694227961932b/

//////////////////////////////////////////////
//        RemoteXY include library          //
//     use  library  version 2.2.5 or up    //
//   use ANDROID app version 3.7.1 or up    //
//////////////////////////////////////////////

/* RemoteXY select connection mode and include library */
#define REMOTEXY_MODE__HC05_HARDSERIAL

#include <RemoteXY.h>

/* RemoteXY connection settings */
#define REMOTEXY_SERIAL Serial1
#define REMOTEXY_SERIAL_SPEED 9600


/* RemoteXY configurate  */
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =
  { 2,1,26,0,6,5,0,3,133,1
  ,10,98,11,2,1,0,63,40,12,12
  ,1,82,0,65,4,25,40,12,12,2
   };

/* this structure defines all the variables of your control interface
*/
struct {

    /* input variable */
  uint8_t buttons; /* =0 if select position A, =1 if position B, =2 if
position C, ... */
  uint8_t record; /* =1 if button pressed, else =0 */

    /* output variable */
  uint8_t recMode_r; /* =0..255 LED Red brightness */

    /* other variable */
  uint8_t connect_flag;  /* =1 if wire connected, else =0 */

} RemoteXY;
#pragma pack(pop)
```

```
//////////////////////////////////////////
//           END RemoteXY include          //
//////////////////////////////////////////
```

# Как пользоваться?


- Включите устройство.
- Запустите приложение remoteXY на Android-смартфоне.
- Нажмите на кнопку выбора последовательности ИК-команд.
- Нажмите кнопку записи. Загорится индикатор записи.
- Наведите на устройство ИК-пульт, и нажмите несколько кнопок.
- Повторно нажмите кнопку записи. Индикатор записи погаснет.
- Нажмите кнопку выбора последовательности, с записанной последовательностью. Устройство воспроизведёт последовательность ИК-команд.

# Альтернативный вариант сборки

Бывает так, что #Структор не подходит к интерьеру. Попробуем собрать устройство в стиле IKEA.

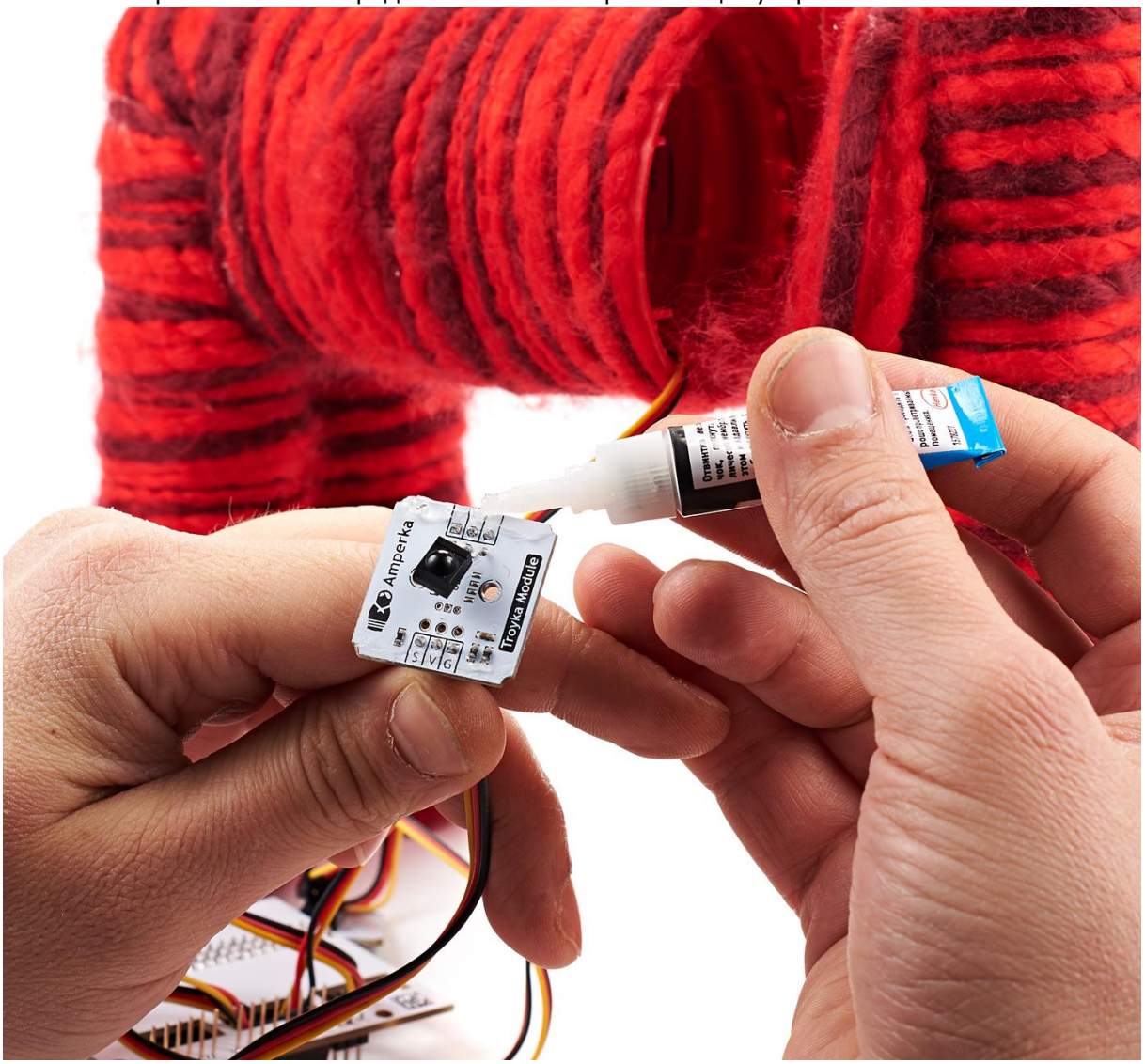
## Что нам понадобится?

1. Iskra Neo
2. Troyka Shield LP
3. Bluetooth HC-05 (Troyka-модуль)
4. ИК-передатчик (Troyka-модуль)
5. ИК-приёмник (Troyka-модуль)
6. Тумблер
7. Батарейка Крона
8. Кабель питания от батарейки Крона
9. Суперклей
10. Канцелярский нож
11. Android-смартфон
12. Приложение RemoteXY
13. Козлик из IKEA

## Как собрать?

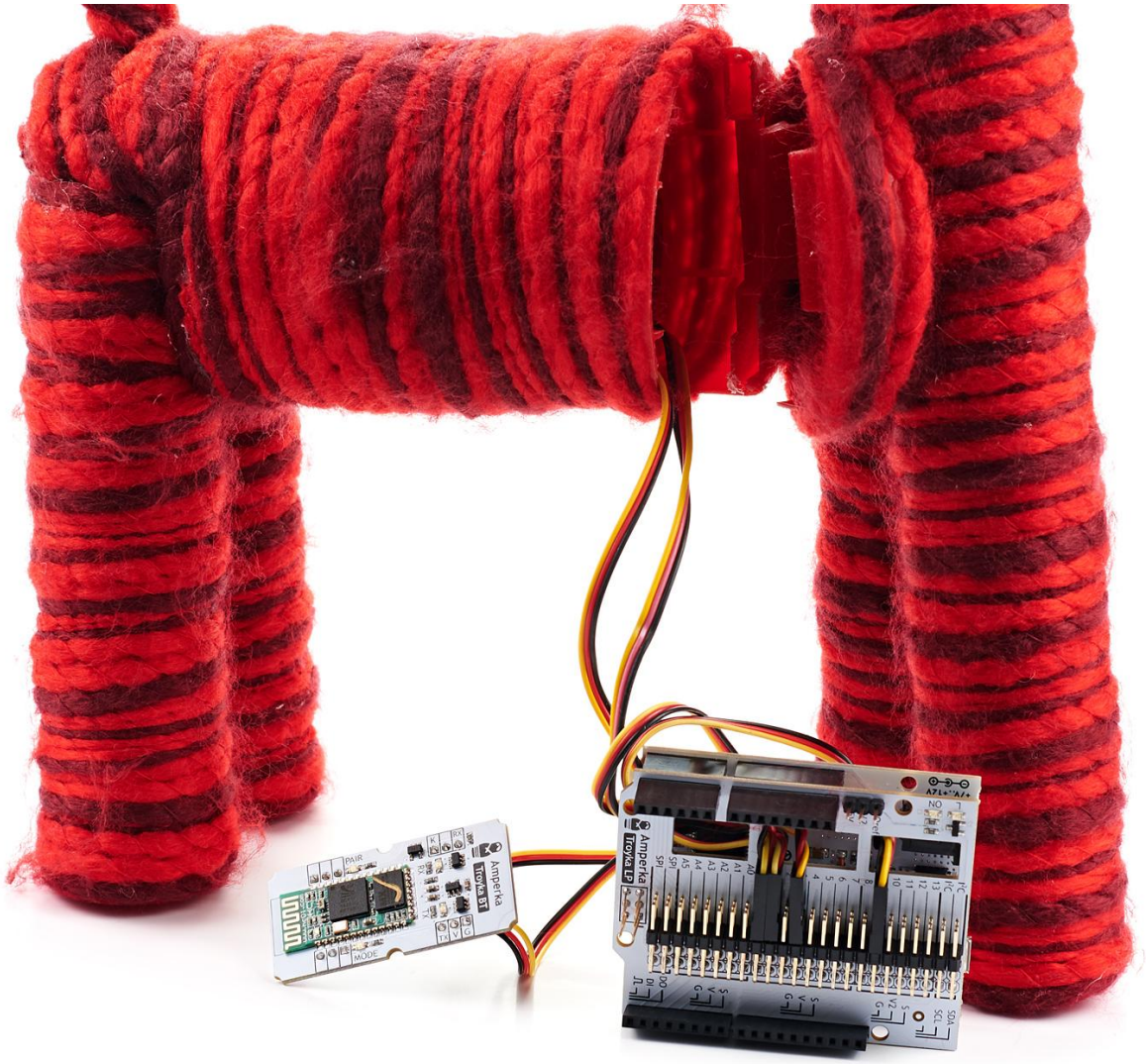
1. Распотрошим козлика канцелярским ножом. Порежем пополам, сделаем отверстия под ИК-приёмник, ИК-передатчик и тумблер

2. Вклеим ИК-приёмник и ИК-передатчик в козлика при помощи суперклея

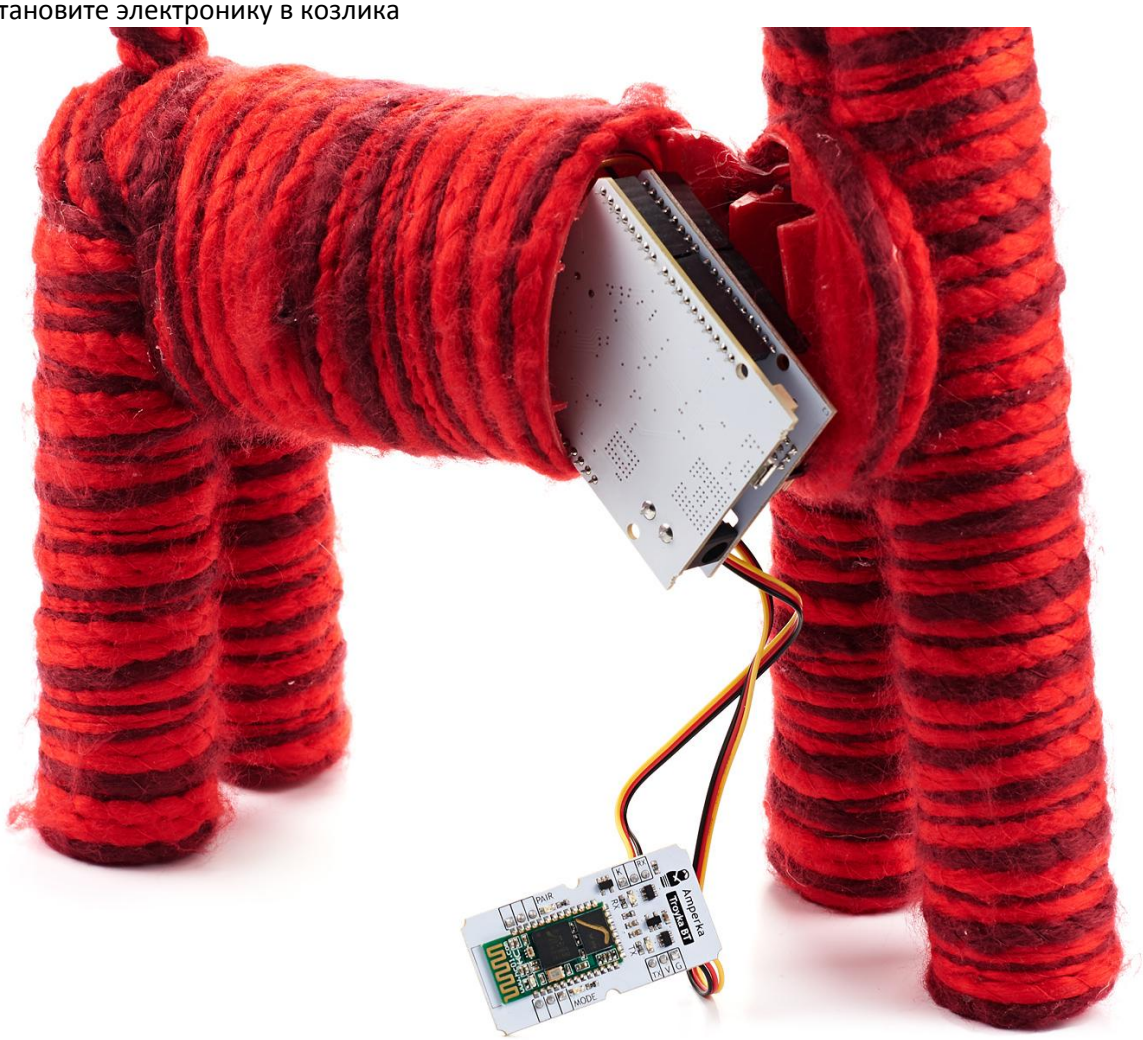


3. Установите Troyka Shield LP на Iskra Neo. Соедините ИК-приёмник с пином `3`, ИК-передатчик с пином `9`. Соедините пин `TX` Bluetooth-модуля с пином `0`, а контакт `RX` с

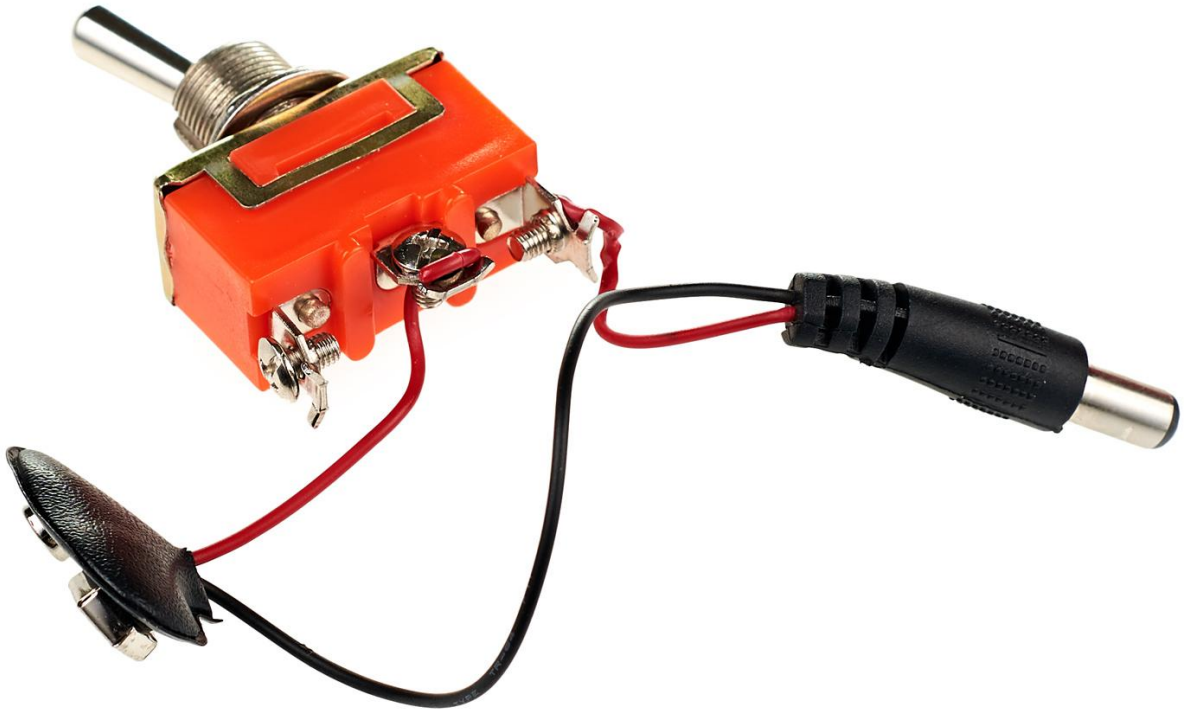
пином 1.



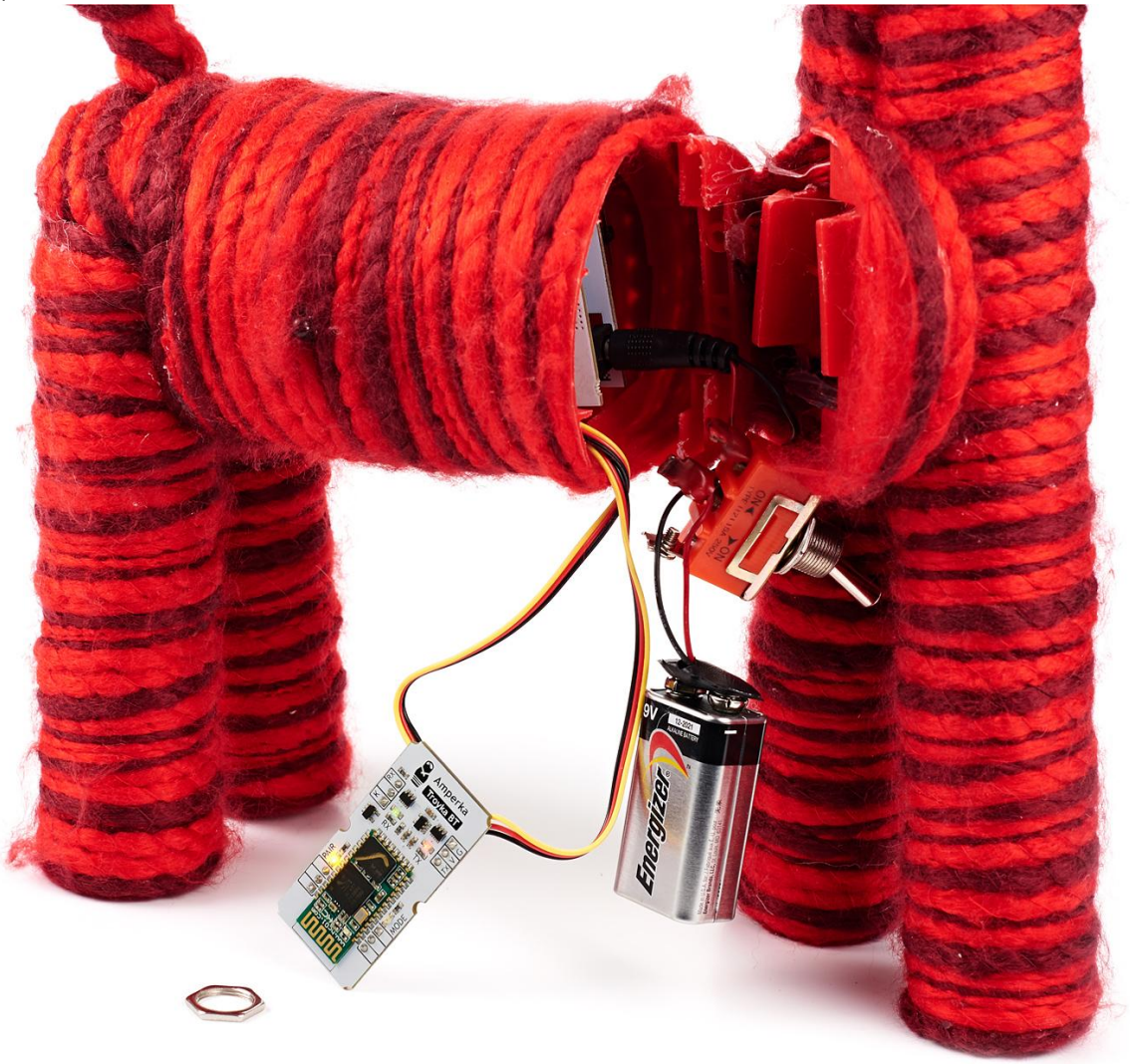
4. Установите электронику в козлика




5. Разрежьте красный провод кабеля питания от батарейки Крона и зачистите концы. Прикрутите контакты к тумблеру



6. Подключите питание к Iskra Neo.



7. Установите тумблер в отверстие для него и зафиксируйте гайкой