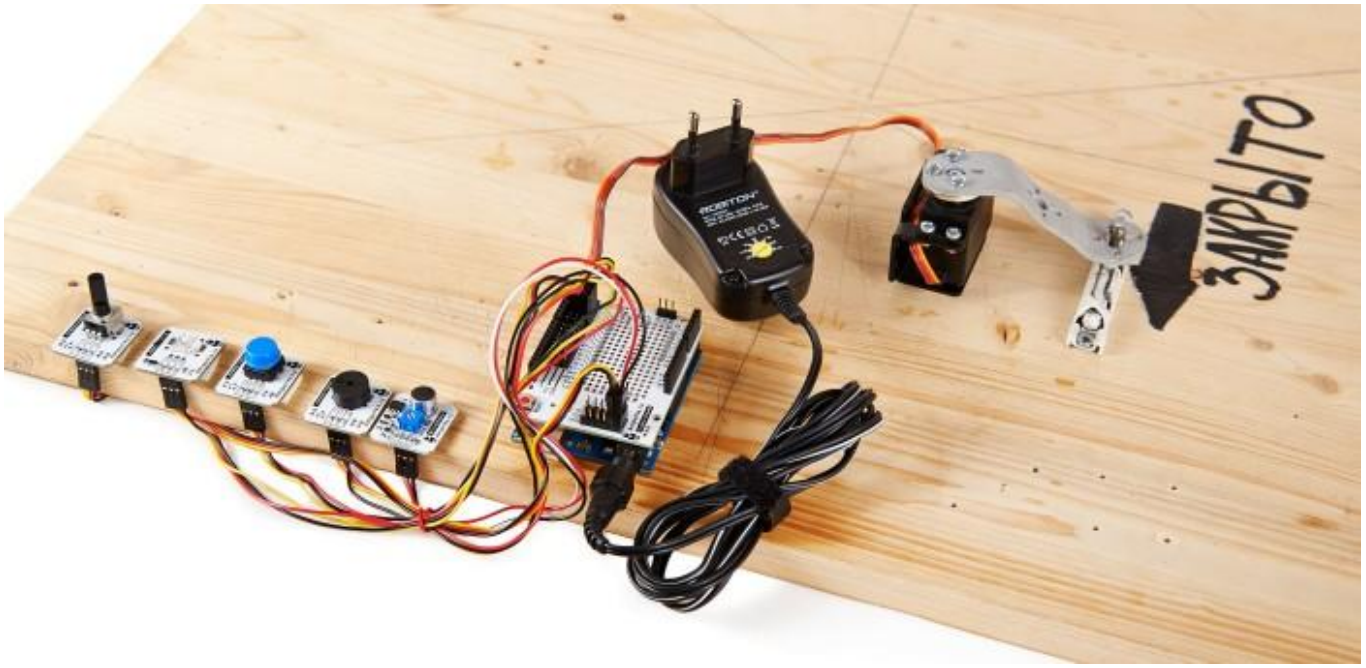
Кодовый замок «Тук-тук»

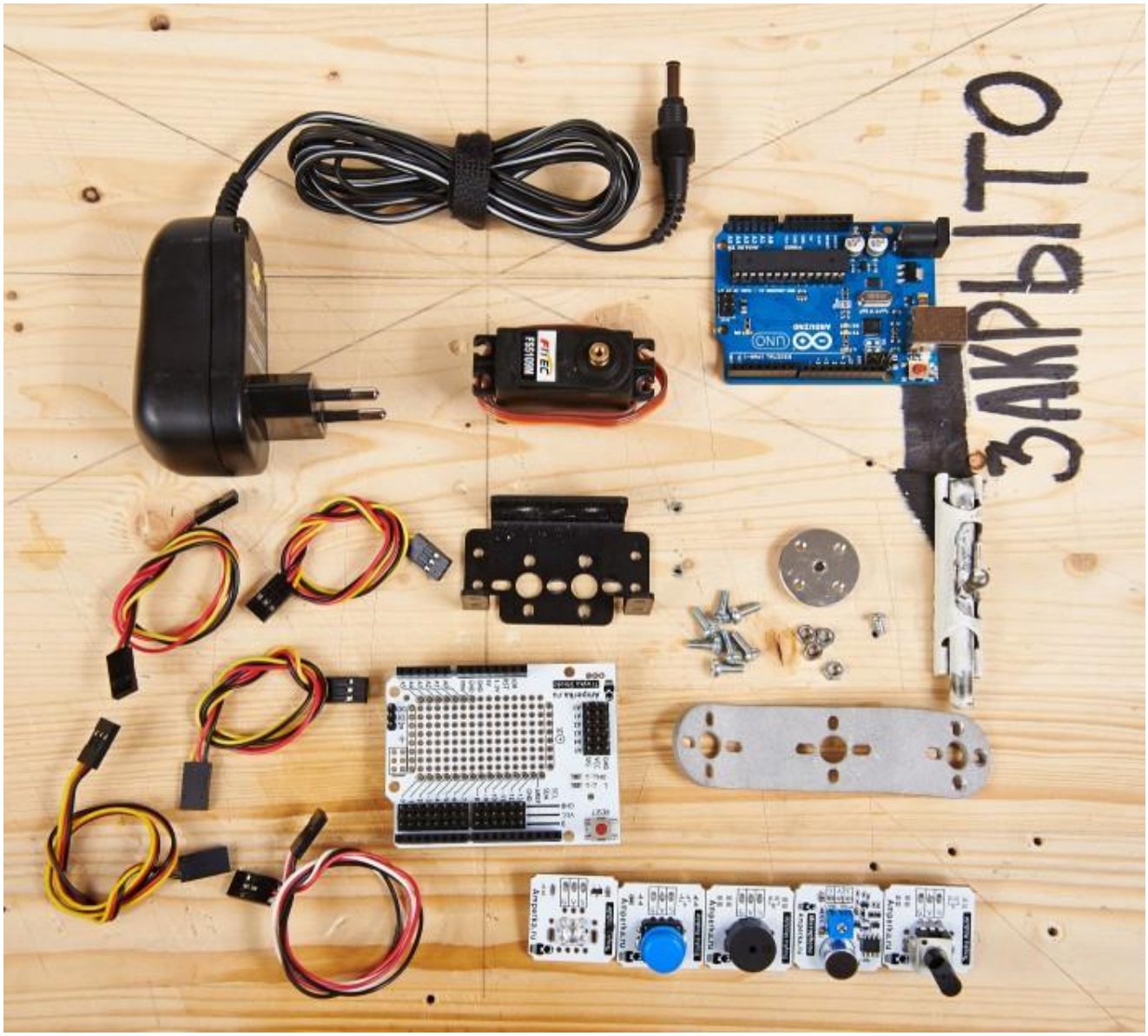


- Платформы: Arduino Uno
- Языки программирования: Arduino (C++)
- Тэги: замок, микрофон, аудиозамок

Что это такое?

В этой статье мы соберём замок, который будет открываться как в шпионских фильмах — по специальному шифру-стуку. Для этого мы воспользуемся модулем, показывающим уровень шума. Этот модуль не отличает высоких звуков от низких, для него важна только амплитуда, т.е. громкость звука. Наше устройство будет иметь режим записи кодовой последовательности, т.к. код можно будет изменить в любой момент.

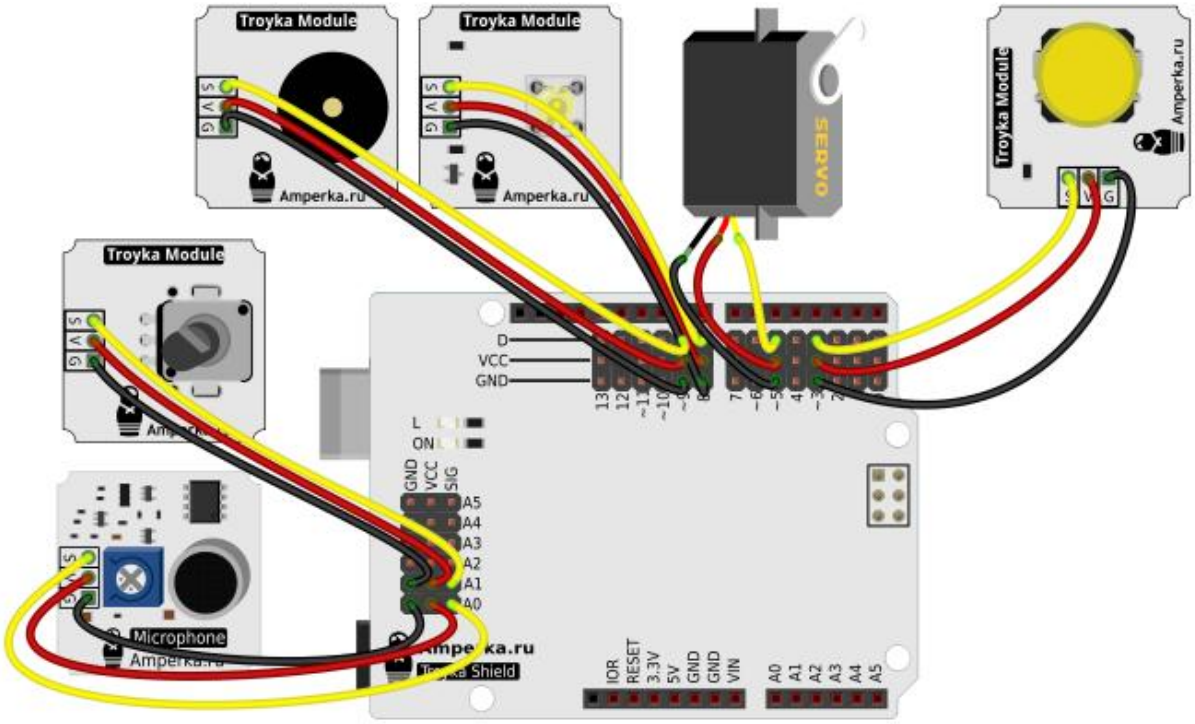
Что нам понадобится?



1. Arduino Uno
2. Troyka Shield
3. Потенциометр (Troyka-модуль)
4. Кнопка (Troyka-модуль)
5. Зуммер (Troyka-модуль)
6. Светодиод «Пиранья» (Troyka-модуль)
7. Микрофон (Troyka-модуль)
8. Сервопривод
9. Импульсный блок питания
10. Алюминиевое полуплечо с фиксатором для сервопривода

Как это собрать?

Соберите следующую схему:



Вам нужно будет придумать, как сделать механическую часть открывания/закрывания. Мы решили проблему таким образом:

Прошейте Arduino Uno скетчем, приведённым ниже.

Исходный код

knocklock.ino

```
#include <Servo.h>

// Задекларируем номера используемых пинов
#define KNOCK_PIN    A0
#define KNOCKSEN_PIN A1
#define BTN_PIN      3
#define BUZZER_PIN   9
#define LED_PIN      8
#define SERVO_PIN    5

#define KNOCK_HIGH 300  // Темп стука
// Если стука не было больше KNOCK_TIMEOUT, значит последовательность
ударов
// закончена
#define KNOCK_TIMEOUT 1000

// Длительности срабатывания кнопки закрывания/открывания замка
// Длинное нажатие - переход в режим программирования кода
// Короткое нажатие - закрыть дверь
#define BTN_SHORT_MIN 200
#define BTN_SHORT_MAX 2000
#define BTN_LONG_MIN 3000
#define BTN_LONG_MAX 10000

// Значения углов сервопривода в открытом и закрытом состояниях.
Значения с
// индексом 2 устанавливаются через небольшую паузу после безиндексных.
Это
// нужно для того, чтобы убрать усилие с сервоприводов.
#define OPEN 107
#define OPEN2 103
#define CLOSE 85
#define CLOSE2 88

// Длина кодовой последовательности в количестве кодов
// Тук и пауза - это "1"
// Тук-тук - это "0"
#define CODE_LEN 5

// Состояния системы
enum State
{
    OPENED,
    CLOSED,
    SETPWD,
    CNFPWD,
    INIT
};

// Стартовый код
bool code[CODE_LEN] = { 1, 0, 1, 0, 1 };
// Полученная кодовая последовательность
bool input[CODE_LEN] = { 0, 0, 0, 0, 0 };
// Чувствительность замка
unsigned int knockLevel = 0;

State state;
Servo srv;
```

```
void setup()
{
    Serial.begin(115200);

    pinMode(BTN_PIN, INPUT);
    pinMode(LED_PIN, OUTPUT);

    digitalWrite(LED_PIN, LOW);

    state = INIT;
    Serial.println("Started.");
}

bool btn(void)
{
    return digitalRead(BTN_PIN);
}

// Функция трансформирует аналоговый сигнал с микрофона в бинарный:
// 1 - громко
// 0 - тихо
// При этом используется простенькая фильтрация
// (сигнал принимается только если он не менялся более 1 мс).
bool getknk(void)
{
    bool b1, b2;

    while (1) {
        b1 = analogRead(KNOCK_PIN) > knockLevel;
        delay(1);
        b2 = analogRead(KNOCK_PIN) > knockLevel;

        if (b1 == b2) return b1;
        delay(1);
    }

    return false;
}

// Функция возвращает код стука (0 или 1) или -1,
// если в течении KNOCK_TIMEOUT никто не стучал.
char getch_knk(void)
{
    unsigned long tm = millis();

    while (!getknk() && millis()-tm < KNOCK_TIMEOUT) {}

    if (millis()-tm >= KNOCK_TIMEOUT) return -1;

    tm = millis();

    while (getknk()) {}

    while (!getknk() && millis()-tm < KNOCK_HIGH) {}

    tm = millis()-tm;

    while (getknk()) {}

    if (tm < KNOCK_HIGH) return 0;

    return 1;
}
```

```
// Функция считывает кодовую последовательность и возвращает:
// 0 - код ошибочен
// 1 - код верен
// -1 - никто не стучал или количество стуков меньше кодовой
последовательности
char readCode(void)
{
    int i;
    char ch;

    for (i = 0; i < CODE_LEN; i++) {
        ch = getch_knk();

        if (ch < 0) return -1;

        input[i] = ch;
        Serial.print((int)ch);
    }

    Serial.println("");

    for (i = 0; i < CODE_LEN; i++) {
        if (input[i] != code[i]) return 0;
    }


    return 1;
}

// Функция считывает программируемый код
// (режим задания кодовой последовательности).
void readPwd(void)
{
    int i;
    char ch;

    for (i = 0; i < CODE_LEN; i++) {
        while ((ch = getch_knk()) < 0) {}
        input[i] = ch;

        if (ch == 0) {
            ledOn();
            delay(100);
            ledOff();
            delay(100);
            ledOn();
            delay(100);
            ledOff();
        } else {
            ledOn();
            delay(100);
            ledOff();
        }

        Serial.print((int)ch);
    }
}

void ledOn(void)
{
    digitalWrite(LED_PIN, HIGH);
}

void ledOff(void)
```

```
{
    digitalWrite(LED_PIN, LOW);
}

// Функция отпирает замок.
void open(void)
{
    srv.attach(SERVO_PIN);
    srv.write(OPEN);
    delay(500);
    srv.write(OPEN2);
    delay(100);
    srv.detach();

    state = OPENED;
    Serial.println("Opened.");
}

// Функция запирает замок.
void close(void)
{
    srv.attach(SERVO_PIN);
    srv.write(CLOSE);
    delay(500);
    srv.write(CLOSE2);
    delay(100);
    srv.detach();

    state = CLOSED;
    Serial.println("Closed.");
}


void loop()
{
    int i;
    char ch;
    unsigned long tm;

    // Актуализируем выставленную потенциометром чувствительность
замка.
    knockLevel = analogRead(KNOCKSEN_PIN);

    switch (state) {
    case OPENED:  // Дверь открыта
        while (btn()) {}
        tm = millis();
        while (!btn()) {}
        tm = millis()-tm;

        if (tm > BTN_SHORT_MIN && tm < BTN_SHORT_MAX) close();
        else if (tm > BTN_LONG_MIN && tm < BTN_LONG_MAX)
            state = SETPWD, ledOn();
        break;

    case CLOSED:  // Дверь закрыта
        ch = readCode();  // Считываем код

        if (ch < 0) {  // Никто не стучал
            Serial.println("FAIL");
            break;
        }

        if (ch == 1) {
```

```
            open();  // Ура! Стук правильный
        } else {  // Стук неправильный, пропипикаем, чтобы сообщить об
этом
            tone(BUZZER_PIN, 500, 200);
            delay(200);
            tone(BUZZER_PIN, 500, 200);
            delay(200);
            tone(BUZZER_PIN, 500, 200);
            delay(200);
        }
        break;

    case SETPWD:  // Задаём кодовую последовательность
        delay(2000);

        // Поморгаем, чтобы показать момент начала записи
        ledOn();
        delay(100);
        ledOff();
        delay(100);
        ledOn();
        delay(100);
        ledOff();

        Serial.print("Input code:");
        readPwd();  // Получаем код

        // Сохраняем код
        for (i = 0; i < CODE_LEN; i++) code[i] = input[i];

        Serial.println("");
        state = CNFPWD;  // Уходим на подтверждение
        break;

    case CNFPWD:  // Подтверждаем кодовую последовательность
        Serial.print("Confirm code:");
        readPwd();  // Получаем код

        // Сравниваем с сохранённым
        for (i = 0; i < CODE_LEN; i++) {
            if (code[i] != input[i]) break;
        }

        Serial.println("");

        // Если всё ок, выходим из процедуры и тушим светодиод
        if (i == CODE_LEN) ledOff(), state = OPENED;
        else
            state = SETPWD;  // Иначе - повторяем ввод кода

        break;

    case INIT: open(); break;  // Начальное состояние
    }
}
```