Отчет по проекту «Шкатулка на сенсорном замке»

Студенты 2 курса 1 группы:

Александров Артём Сергеевич,

Васильев Владислав Васильевич

**Описание проекта:**

Замок с сервоприводом и датчиком звука/вибрации/касания

"Запоминает" секретный стук, а именно - время между "ударами"

Умеет распознавать слишком быстрое нажатие, слишком медленное, ну и само собой "правильное" нажатие

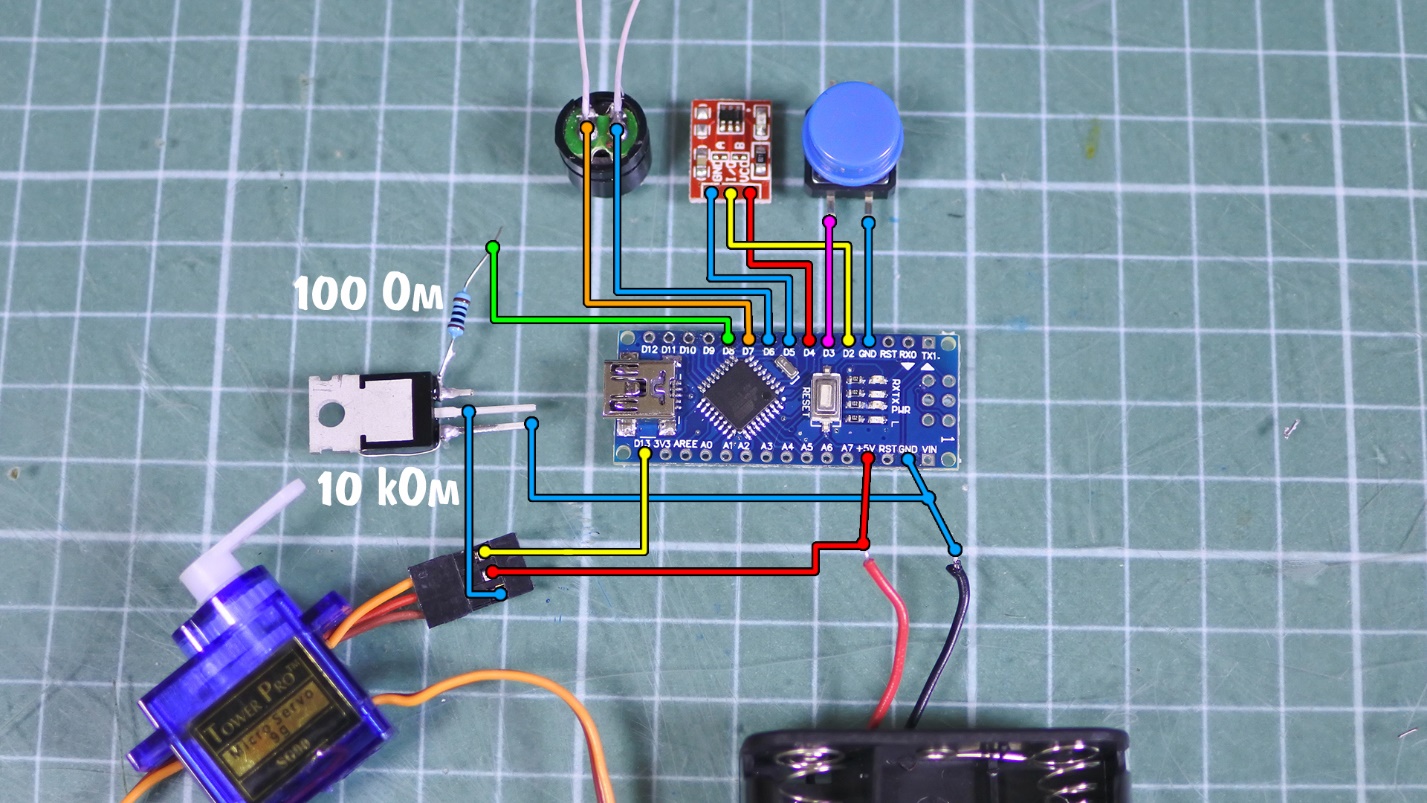
Работает от аппаратного прерывания, что даёт очень стабильную и чёткую отработку "стуков"

Использована библиотека сна, потребление в режиме ожидания около 0.1 мА

**Компоненты:**

* Плата Arduino Nano
* Сервопривод
* Батарейный отсек
* Сенсорная кнопка
* Резистор 100 Ом
* Резистор 10 кОм
* Пьезодинамик
* Кнопка

Схема:



**Код:**

//----- НАСТРОЙКИ -----

#define difficulty 100 // миллисекунд на реакцию (плюс минус)

#define max\_knock 30 // число запоминаемых "стуков"

#define close\_angle 85 // угол закрытия

#define open\_angle 180 // угол открытия

#define debug 1 // режим отладки - вывод в порт информации о процессе игры

//----- НАСТРОЙКИ -----

#include <Servo.h>

Servo servo;

#define buzzPin 7 // пин пищалки

#define buzzGND 6 // земля пищалки

#define sensGND 5 // земля сенсора

#define sensVCC 4 // питание сенсора

#define servoVCC 8 // питание серво

#define servoPin 11 // серво

#include <TimerOne.h>

#include <LowPower.h> // библиотека сна

byte fade\_count, knock;

volatile byte mode;

boolean cap\_flag, write\_start;

volatile boolean debonce\_flag, threshold\_flag;

volatile unsigned long debounce\_time;

unsigned long last\_fade, last\_try, last\_knock, knock\_time, button\_time;

byte count, try\_count;

int wait\_time[max\_knock], min\_wait[max\_knock], max\_wait[max\_knock];

void setup() {

Serial.begin(9600);

if (debug) Serial.println("system start");

delay(50);

pinMode(2, INPUT); // пин датчика вибрации или кнопка

pinMode(3, INPUT\_PULLUP); // пин датчика вибрации или кнопка

servo.attach(servoPin);

// настраиваем пины питания как выходы

pinMode(buzzPin, OUTPUT);

pinMode(buzzGND, OUTPUT);

pinMode(sensGND, OUTPUT);

pinMode(sensVCC, OUTPUT);

pinMode(servoVCC, OUTPUT);

// подаём нужные сигналы

digitalWrite(buzzPin, 0);

digitalWrite(buzzGND, 0);

digitalWrite(sensGND, 0);

digitalWrite(sensVCC, 1);

digitalWrite(servoVCC, 1);

servo.write(open\_angle);

delay(1000);

digitalWrite(servoVCC, 0);

attachInterrupt(0, threshold, RISING); // прерывание датчика

attachInterrupt(1, buttonPress, FALLING); // прерывание кнопки выключить

threshold\_flag = 0;

knockWrite();

delay(50);

close\_cap();

good\_night(); // сразу спать

}

void knockWrite() { // режим записи стука

if (debug) Serial.println("knock write mode");

last\_knock = millis();

knock = 0;

while (1) { // ждём первого удара

if (millis() - last\_knock > 5000) { // если 5 секунд не ударяли

write\_start = 0; // всё сбросить, выйти из режима

break;

}

if (threshold\_flag) { // если ударили

write\_start = 1; // разрешить запись

tone(buzzPin, 400, 50); // пикнуть дрыгнуть

last\_knock = millis();

threshold\_flag = 0;

if (debug) Serial.println("knock");

break;

}

}

if (write\_start) { // если запись пошла

while (1) {

if (threshold\_flag) {

knock\_time = millis() - last\_knock; // расчёт времени между стуками

wait\_time[knock] = knock\_time; // записать

min\_wait[knock] = knock\_time - difficulty; // определить время с учётом времени реакции

max\_wait[knock] = knock\_time + difficulty; // определить время с учётом времени реакции

knock++; // перейти к следующему

tone(buzzPin, 400, 50); // пикнуть дрыгнуть

last\_knock = millis();

threshold\_flag = 0;

if (debug) Serial.println("knock");

}

if (millis() - last\_knock > 3000) {

break;

}

}

// показать комбинацию "раунда"

tone(buzzPin, 400, 50); // пыхнуть светодиодом

for (byte i = 0; i < knock; i++) {

delay(wait\_time[i]); // ждать время шага одного хода

tone(buzzPin, 400, 50); // пыхнуть светодиодом

if (debug) Serial.println(wait\_time[i]);

}

mode = 0; // перейти в режим игры

}

}

void loop() {

if (threshold\_flag && mode == 0) {

threshold\_flag = 0;

if (knock == 0) {

mode = 3;

goto openCap;

}

debounce\_time = millis();

last\_try = millis(); // обнулить таймер

tone(buzzPin, 400, 50);

try\_count = 0;

threshold\_flag = 0;

while (1) {

// если не нажал в установленное время (проигрыш)

if (millis() - last\_try > max\_wait[try\_count]) {

// мигнуть красным два раза

tone(buzzPin, 400, 50);

delay(1000);

mode = 0; // перейти в начало! Это начало нового раунда

if (debug) Serial.println("too slow");

threshold\_flag = 0;

break;

}

if (threshold\_flag) {

// если нажатие попало во временной диапазон (правильное нажатие)

if (millis() - last\_try > min\_wait[try\_count] && millis() - last\_try < max\_wait[try\_count]) {

tone(buzzPin, 400, 50); // мигнуть

try\_count++; // увеличить счётчик правильных нажатий

last\_try = millis(); // ВОТ ТУТ СЧЁТЧИК СБРАСЫВАЕТСЯ, ЧТОБЫ УБРАТЬ ВЛИЯНИЕ ЗАДЕРЖЕК!

threshold\_flag = 0; // сбросить флаг

if (debug) Serial.println("good");

// если нажал слишком рано (проигрыш)

} else if (millis() - last\_try < min\_wait[try\_count] && threshold\_flag) {

tone(buzzPin, 400, 50);

delay(100);

tone(buzzPin, 400, 50); // мигнуть красным дважды

delay(1000);

mode = 0; // перейти в начало! Это начало нового раунда

if (debug) Serial.println("too fast");

threshold\_flag = 0;

break;

}

// если число правильных нажатий совпало с нужным для раунда (выигрыш)

if (try\_count == knock) {

// мигнуть 3 раза

delay(200);

tone(buzzPin, 400, 50);

delay(200);

tone(buzzPin, 400, 50);

delay(200);

tone(buzzPin, 400, 50);

delay(200);

mode = 3; // перейти к действию при выигрыше

if (debug) Serial.println("victory");

break;

}

}

}

}

if (mode == 3) {

openCap:

mode = 4;

delay(500);

open\_cap();

good\_night();

}

if ((threshold\_flag && mode == 4) || mode == 5) {

mode = 0;

delay(500);

close\_cap();

good\_night();

}

if (millis() - debounce\_time > 10000 || mode == 4) {

good\_night();

}

}

void threshold() {

if (millis() - debounce\_time > 50) debonce\_flag = 1;

if (debonce\_flag) {

debounce\_time = millis();

threshold\_flag = 1;

debonce\_flag = 0;

}

}

void buttonPress() {

if (mode == 4) {

mode = 5;

}

}

void good\_night() {

if (debug) {

Serial.println("good night");

delay(50);

}

delay(5);

LowPower.powerDown(SLEEP\_FOREVER, ADC\_OFF, BOD\_OFF); // спать. mode POWER\_OFF, АЦП выкл

}

void open\_cap() {

if (debug) Serial.println("open cap");

digitalWrite(servoVCC, 1);

for (int i = close\_angle; i < open\_angle; i++) {

servo.write(i);

Serial.println(i);

delay(12);

}

delay(100);

digitalWrite(servoVCC, 0);

}

void close\_cap() {

if (debug) Serial.println("close cap");

digitalWrite(servoVCC, 1);

for (int i = open\_angle; i > close\_angle; i--) {

servo.write(i);

Serial.println(i);

delay(12);

}

delay(100);

digitalWrite(servoVCC, 0);

}