**Простая индикаторная панель для Ардуино**

*Библиотека Ардуино для простой индикаторной панели на зуммере и светодиодах*

**Предисловие**

В победившей на конкурсе "STроим вмеSTе" статье "[Самодельный электромеханический замок "ANTAR SL21"](https://cxem.net/guard/3-99.php)" была описана конструкция замка, в котором помимо основных функций, таких как закрывание/открывание, была реализована индикация различных состояний устройства на базе двухцветного светодиода и миниатюрного электромагнитного зуммера.

В последствии подобная индикация не раз потребовалась в различных конструкциях и самоделках и было решено вынести данный функционал в отдельный класс в виде библиотеки для Ардуино. В итоге получился подключаемый модуль, который очень легко и удобно использовать в любых проектах на платформе Arduino.

Не хотелось бы углубляться и очень подробно описывать все нюансы и внутренний механизм библиотеки, так как в процессе совершенствования может многое измениться за очень короткое время. Далее поверхностно будут рассмотрены только основные возможности, которые представляют собой воплощаемые технические требования к предоставляемым функциям и которые не будут сильно меняться от версии к версии. Алгоритмы же и логика работы самого программного кода класса могут меняться радикально по мере необходимости. Код библиотеки предоставляется в открытом доступе на платформе GitHub: <https://github.com/TarAndr/AnTar_SIP>.

**Описание**

Назначение библиотеки - индикация разнообразных, заранее определённых состояний или событий устройства различными светозвуковыми сигналами. Для начала рассмотрим аппаратную составляющую, необходимую для обеспечения нужного функционала. Выходными компонентами являются светодиоды и миниатюрный зуммер, хотя по желанию можно конечно использовать лампочки и/или громкоговоритель с соответствующими управляющими элементами. Оповещятельные сигналы не представляют собой большой сложности и являются импульсами определённой длительности с заданным числом повторений.

Данные импульсы управляют свечением светодиодов и звучанием зуммера. К моменту написания статьи использовался двухцветный светодиод с красным и зелёным составляющими, хотя можно применить и отдельные светодиоды, а в перспективе при надобности можно доработать библиотеку и для работы с трёхцветным RGB-светодиодом для определения широкого спектра различных цветов свечения.

Светодиоды и зуммер подключаются к любым выводам платы Arduino, которые задаются программно:

*Блок-схема подключений исполнительных узлов к плате Arduino*

Также в качестве управляющего устройства можно использовать микроконтроллер без отладочной платы, предварительно обеспечив для него компиляцию и загрузку исходного кода:

*Принципиальная электрическая схема индикаторной панели на микроконтроллере*

В данном конкретном случае на обеих схемах двухцветный светодиод подключен через один общий токоограничительный резистор и работает нормально, так как напряжение питания составляет 5 Вольт, при более низком же напряжении питания рекомендуется использовать отдельные токоограничивающие резисторы для каждого кристалла, так как при смешанном свечении один цвет будет перекрывать другой из-за различия падения напряжений на светодиодах разного цвета свечения.

Для использования функций библиотеки сначала её нужно скачать и установить, а затем подключить к исходному тексту своей программы.

Скачать библиотеку можно с хранилища системы контроля версий. Рекомендуется всегда скачивать стабильную версию со страницы репозитория в разделе <https://github.com/TarAndr/AnTar_SIP/releases>. Но при желании или затруднениях можно использовать последнюю опубликованную версию на главной ветке репозитория  по ссылке <https://github.com/TarAndr/AnTar_SIP/archive/main.zip>. Так же здесь на сайте в файлах для скачивания прикреплён архив с первой рабочей версией к моменту написания статьи.

После скачивания архив нужно распаковать в папку библиотек Arduino, например для операционной системы Windows это может быть папка "Documents\Arduino\libraries". Скачанный архив так же можно интегрировать через саму среду разработки, что наверное будет более правильным решением. Для этого в разделе "Sketch" из списка "Include Library" нужно выбрать пункт "Add ZIP Library..." и указать файл архива библиотеки:

*Sketch >> Include Library >> Add ZIP Library...*

При успешном завершении указанных действий в информационном окне должно отобразиться сообщение "Library installed", а в примерах установленных библиотек должны появиться примеры для работы с только что добавленной библиотекой:

*Примеры для работы с библиотекой*

Открыв выбранный пример можно на его базе начать собственный проект или же скопировать части кода примера в уже имеющийся начатый ранее проект. Далее рассмотрим какие шаги нужно предпринять для этого.

**Работа с библиотекой**

В первую очередь файл библиотеки необходимо подключить директивой **#include** как показано ниже:

#include <AnTar\_SIP.h>

Далее создаётся объект в виде экземпляра класса библиотеки, и здесь же в конструктор передаются параметры для установки подключаемых выводов панели:

SIP myPanel(redLedPin, greenLedPin, buzzerPin, buzzerFreq);

Здесь имя объекта может быть любым допустимым, а не обязательно "**myPanel**". В качестве параметров передаются номера выводов соответственно для красного светодиода, зелёного светодиода и зуммера. Последним параметром передаётся резонансная частота используемого зуммера, и если её не задать явно не передавая этого параметра в конструктор, то частота будет установлена в значение по умолчанию, что можно посмотреть и переопределить в исходном коде библиотеки "AnTar\_SIP.h".

Частоту звучания зуммера так же можно изменить во время работы программы, а в указанном файле исходного кода можно подробно посмотреть имеющиеся методы и константы для работы с библиотекой. Например для изменения частоты звучания определён следующий метод:

myPanel.setBuzzerFreq(buzzerFreq);

Где в качестве параметра **buzzerFreq** передаётся вновь устанавливаемая частота в Герцах. Метод **setBuzzerFreq();** вызванный без параметров так же устанавливает частоту звучания зуммера в значение по умолчанию.

Главным методом для вывода информации на панель в виде световых и звуковых сигналов является метод **signal(mode);** где в качестве параметра передаётся номер заранее определённого режима индикации, например так:

myPanel.signal(4);

Это может быть короткий двойной писк с одновременным зажиганием зелёного светодиода, или три раза повторяющиеся протяжённые гудки вместе со свечением красного светодиода. Но так как помнить номера режимов наизусть неудобно, да и не эстетично, поскольку в программном коде появляются магические числа, то в файле библиотеки (но не в самом классе) был определён перечень номеров режимов с понятными именами.

Но поскольку библиотека создана из идеи сигналов индикации электронного замка, то и названия режимов здесь соответствующие. К моменту написания статьи перечень режимов выглядел следующим образом:

enum signalMode {

shortBeep = 0,

lowPowerAlarm,

emptyPowerAlarm,

lockAlarm,

lockOpened,

lockClosed,

toLockAlarm,

modesNum // this is to determine the number of modes

};

Как видно из перечисления всего здесь семь режимов сигнала индикации, последнее значение используется для определения количества режимов, которое всегда можно увеличить по своему усмотрению. Для этого понадобиться немного поработать над структурой параметров режимов, где задаются продолжительность и число повторений сигнала, а так же указывается какие светодиоды задействованы в том или ином режиме.

С помощью данного перечисления можно например включить индикацию полного разряда батареи питания, не задумываясь о номере необходимого сигнала, вызвав соответствующий метод следующим образом:

myPanel.signal(emptyPowerAlarm);

Кроме главного метода в библиотеке определён вспомогательный метод, предназначенный для отключения и/или включения звучания зуммера:

mute(state);

Если вызвать метод mute(); без параметров или со значением true, то во время индикации зуммер не будет задействован. Для примера использования:

myPanel.mute();

// or

myPanel.mute(true);

Для последующего задействования зуммера нужно будет вызвать указанный метод со значением false, как в данном примере:

myPanel.mute(false);

И под конец статьи приведём полный базовый пример для поочерёдной индикации всех имеющихся режимов с небольшой задержкой между переходами и повторением. В теле основного цикла подряд перебираются все имеющиеся предустановленные состояния воспроизводя соответствующие световые и звуковые сигналы:

// Подключение библиотеки

#include <AnTar\_SIP.h>

// Номера контактов:

const int redLedPin = 13;

const int greenLedPin = 12;

const int buzzerPin = A2;

const long buzzerFreq = 4400;

// создание экземпляра класса панели, указав контакты,

// к которым она подключена, и/или частоту звукового сигнала

SIP myPanel(redLedPin, greenLedPin, buzzerPin, buzzerFreq);

// без задания частоты (со значением по умолчанию):

//SIP myPanel(redLedPin, greenLedPin, buzzerPin);

void setup() {

// установка частоты зуммера на 4400 Гц

//myPanel.setBuzzerFreq(buzzerFreq);

// Установка частоты звукового сигнала по умолчанию

//myPanel.setBuzzerFreq();

}

void loop() {

// переключение режимов

for(int mode = shortBeep; mode < modesNum; ++mode) {

myPanel.signal(mode);

delay(1000);

}

delay(2000);

}

**В заключение**

Хотелось бы отметить, что кроме больших положительных качеств объектно-ориентированного подхода, в виде удобства использования и возможности создания и одновременной работы нескольких объектов, имеются и некоторые недостатки, в частности для микроконтроллерных устройств с ограниченными ресурсами. Это конечно же более высокие требования к объёму памяти, как программной, так и оперативной - ведь создаваемый экземпляр класса постоянно находится в памяти независимо от того, к каким данным идёт обращение в настоящий момент. Учитывая это ограничение иногда вынуждено приходится писать программу в процедурном стиле, а некоторые даже до сих пор пишут на Ассемблере.