Практическое занятие №1
Тема: «Типовые алгоритмы и работа с системой прерываний МК на высокоуровневом языке»

Цель работы: приобрести практические навыки по разработке типовых алгоритмов с системой прерываний МК на высокоуровневом языке.

Последовательность выполнения работы:

– Изучить теоретические сведения, приведенные в практическом занятии.

– Сделать монтажную и принципиальную схему в программе Fritzing. (напоминание: принципиальная схема формируется автоматически после создания монтажной).

– Собрать схемы на макетной плате, иначе при отсутствии набора Arduino в web-приложениях (https://wokwi.com/projects/new/arduino-uno или https://www.tinkercad.com/) для приведенных примеров.

– Запрограммировать микроконтроллер согласно тексту, указанному в примере.

– Выполнить задание для самостоятельной работы.

СОДЕРЖАНИЕ ОТЧЕТА

– Название практического занятия, его цель.

– Принципиальная и монтажная схема подключения к микроконтроллеру: скриншоты, в т.ч. если выполнение в web-приложении или фотографии, в случае наличия необходимых компонентов у вас (в наборе присутствуют).

– Написанный программный код для скетчей: вставить в отчёт текстом, Courier New, 12 одинарный отступ без абзацев.

– Вывод о проделанной работе.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Прерывания можно объяснить на простом жизненном примере. Например, вы ждете очень важное письмо на электронной почте и, поэтому проверяете свой почтовый ящик каждые 10 минут. Это приводит к трате времени и ваша работа работа становится менее продуктивной. В случае с программированием Ардуино — это равносильно постоянному опросу датчика, а на это будет тратиться время и ресурсы процессора.

Второй вариант — это включить оповещение, таким образом, вы сможете заниматься своими делами, не проверяя почтовый ящик. Когда придет оповещение, то вы прервете свою работу и прочитаете письмо. Прерывания в Arduino работают аналогичным образом. Вместо отслеживания

состояния кнопки, можно использовать аппаратное прерывание и как только кнопка будет нажата, запустится функция прерывания.

В следующей таблице приведены пины, которые могут быть использованы для прерываний. Если какой-либо вывод на плате не совместим с функцией прерывания, то программа выполняться не будет. Прерывания запускаются по изменению входного сигнала на пине платы. Вы можете выбрать точное изменение сигнала, которое хотите контролировать. Для этого используется третий параметр в attachInterrupt().

платы Arduino	**пины, поддерживающие прерываания**
Arduino Uno, Nano, Mini	2, 3
Arduino Mega	2, 3, 18, 19, 20, 21
Arduino Micro, Leonardo	0, 1, 2, 3, 7
Arduino Zero	все цифровые пины, кроме 4
Arduino Due	все цифровые пины

Синтаксис вызова функции:

attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);

digitalPinToInterrupt(pin) — в этом параметре необходимо указать конкретный пин, который будет использоваться для обработки внешнего сигнала прерывания.

ISR — имя функции, которая будет вызвана во время прерывания

mode — тип обрабатываемого перехода (изменения состояния), например, нижний уровень, высокий уровень и т.д. Возможны следующие значения:

RISING: при изменении уровня сигнала с НИЗКОГО на ВЫСОКИЙ

FALLING: при изменении уровня сигнала с ВЫСОКОГО на НИЗКИЙ

CHANGE: каждый раз, когда сигнал изменяется с LOW на HIGH или с HIGH на LOW

LOW: срабатывает всякий раз, когда сигнал находится в состоянии LOW

Что учесть при использовании прерываний

Делайте функцию прерывания максимально быстрой:

Функция прерывания должна быть как можно более быстрой, поскольку она прерывает выполнение основного кода Вашей программы. Если при возникновении внешнего прерывания необходимо выполнить длительные действия, то их следует оставить в void loop(). А внутри функций прерывания следует модифицировать только те переменные состояния, от которых зависят требуемые операции.

Не используйте в прерываниях функции времени:

Команда delay() в функции прерывания не будет работать. Если использовать счетчик времени на millis(), то при выходе из функции прерывания будет получено последнее записанное значение, так как в функции прерывания отсчет времени не идет. Использование функции millis() может предотвратить дрожание кнопки, но лучше использовать его в главном цикле void loop().

Не используйте монитор порта в прерываниях:

Когда вы находитесь внутри прерывания, данные, отправленные или полученные командами Serial, могут быть потеряны. Для отладки программы можно использовать Serial.print() внутри функции прерывания и убедиться, что прерывание сработало, но при этом можно столкнуться с проблемами. Предпочтительнее устанавливать флаг внутри прерывания и запрашивать этот флаг внутри void loop().

Используйте volatile для объявления переменных:

Переменные, используемые в функциях ISR и обычных функциях, должны быть объявлены с использованием переменной volatile. Это сообщает компилятору, что переменные данного типа могут измениться в любой момент и микроконтроллер должен перезагружать переменную при каждом обращении к ней. В противном случае алгоритмы оптимизации могут полностью удалить переменную из файла.

Использование функции attachInterrupt в языке Arduino очень полезно, когда необходимо убедиться, что микроконтроллер не пропустит никаких изменений в контролируемом сигнале. Однако при использовании прерываний по кнопке в Arduino нужно быть осторожным и не использовать их слишком часто. Иногда простой опрос датчика может оказаться более простым и лучшим вариантом.

ЗАДАНИЕ

Для этого задания потребуется:

- Arduino Uno / Arduino Nano / Arduino Mega;
- тактовая кнопка;
- светодиоды и резисторы;
- макетная плата;
- провода «папа-папа».

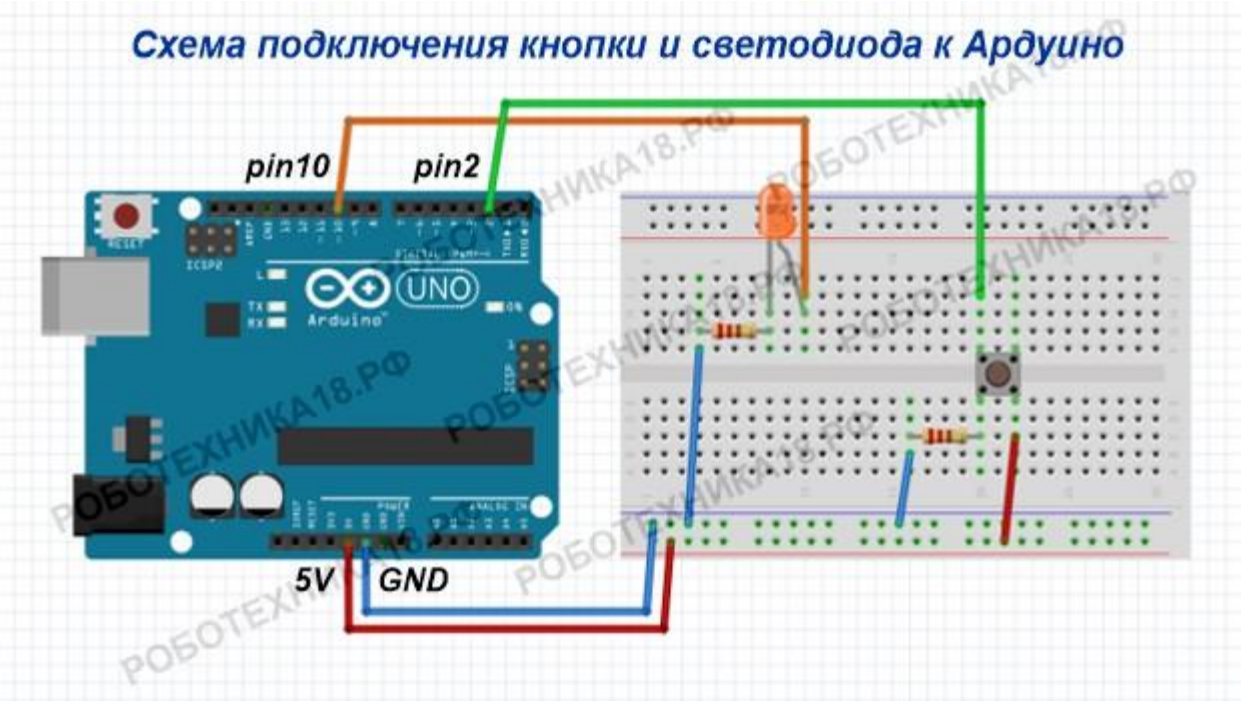

Рисунок 1 – Монтажная схема к заданию

В первом примере микроконтроллер постоянно следит за состоянием кнопки, и при ее нажатии загорается светодиод. Но если использовать цифровой вход 2 в режиме прерывания, то микроконтроллеру не нужно опрашивать кнопку. Если цифровой сигнал на выводе, активированном в режиме прерывания, изменится, то Arduino прекратит выполнение кода основной программы, не опрашивая кнопку постоянно.

Скетч для опроса кнопки без прерывания

```
#define LED 10
#define BUTTON 2

byte ledState = LOW;

void setup() {
    pinMode(LED, OUTPUT);
    pinMode(BUTTON, INPUT);
}

void loop() {
    if (digitalRead(BUTTON) == HIGH) {
        ledState = !ledState;
    }

    digitalWrite(LED, ledState);
}
```

Пояснения к коду: при нажатии на кнопку — включится светодиод.

Скетч для внешнего прерывания по кнопке:

```
#define LED 10
#define BUTTON 2

volatile byte ledState = LOW;

void setup() {
    pinMode(LED, OUTPUT);
    pinMode(BUTTON, INPUT);

    attachInterrupt(digitalPinToInterrupt(BUTTON), blinkLed, RISING);
}

void loop() {
    //
}

void blinkLed() {
    ledState = !ledState;
    digitalWrite(LED, ledState);
}
```

Пояснения к коду: в основном цикле void loop() нет никаких команд, однако при нажатии на кнопку — сработает функция прерывания blinkLed() и светодиод включится.

САМОСТОЯТЕЛЬНАЯ РАБОТА

Ответьте на контрольные вопросы:

1. Что такое прерывание?
2. Какие функции используются для работы с прерываниями в Arduino?
3. Как правильно использовать attachInterrupt() в Arduino?
4. Какие пины используются для прерываний на разных платах Arduino?
5. В каких случаях полезны прерывания в Arduino?
6. Как работают функции millis() и micros() при использовании прерываний?

Собрать и запрограммировать схему аналогичной заданию, но с двумя светодиодами и двумя кнопками, подключенных к 2 и 3 пину.