Урок 7. Прерывания

1. Сделать “светофор” через прерывания, чтобы он выключался-включался по прерыванию с кнопки
2. * Воспользуйтесь аппаратным прерыванием и “поймайте” сигнал с датчика..

Задание 2*. Необходимо сделать прерывание по таймеру, и при этом, чтобы каждый таймер “ловил” свое событие. Программа должна взаимодействовать как минимум с двумя таймерами. Можно использовать любые доступные библиотеки.

1. Включение ночного режима светофора нажатием кнопки. (не удалось выполнить принудительное завершение функции nightMode)

```
const int inputPin = 2;
volatile boolean flag = 1;
void setup() {
  Serial.begin(9600);
  pinMode(5, OUTPUT);          // будем мигать
  pinMode(3, OUTPUT);          // будем мигать
  pinMode(4, OUTPUT);          // будем мигать
  pinMode(inputPin, INPUT_PULLUP);
  attachInterrupt(0, nightMode, LOW); // контакт D2
}

void nightMode(){
  flag = !flag;
}

void loop() {
digitalWrite(5, 1*flag);
delay(2000*flag);
```

```
digitalWrite(5, LOW);
delay(500*flag);
digitalWrite(5, 1*flag);
delay(500*flag);
digitalWrite(5, LOW);
digitalWrite(3, 1*flag);
delay(1000*flag);
digitalWrite(3, LOW);
digitalWrite(4, 1*flag);
delay(2000*flag);
digitalWrite(3, 1*flag);
delay(1000*flag);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
digitalWrite(3, 1*!flag);
delay(500*!flag);
digitalWrite(3, LOW);
delay(500*!flag);
}
```

Работа кода показана на видео Traffic_light.mp4

2. Аппаратное прерывание с датчика APDS9930:

```
// Подключаем библиотеки:                                    //
#include <Wire.h>                                  // Для работы с шиной I2C
#include <APDS9930.h>                              // Для работы с датчиком APDS-9930
APDS9930 apds = APDS9930();                        // Определяем объект apds, экземпляр класса APDS9930
                                                   //
// Объявляем выводы, флаги и функции для прерываний:          //
uint8_t  pinINT = 2;                               // Определяем № вывода Arduino к которому подключен вывод INT датчика
uint8_t  numINT;                                   // Объявляем переменную для хранения № внешнего прерывания для вывода pinINT
bool     flgINT;                                   // Объявляем флаг указывающий на то, что сработало прерывание
void     funINT(void){flgINT=1;}                   // Определяем функцию, которая будет устанавливать флаг flgINT при каждом её вызове
                                                   //
// Объявляем переменные:                                     //
uint16_t proximityIntHigh = 50;                    // Определяем переменную для хранения верхнего порога приближения, ниже которого прерывания выводиться не будут
uint16_t proximityIntLow  = 0;                     // Определяем переменную для хранения нижнего  порога приближения, выше которого прерывания выводиться не будут
uint16_t proximityData    = 0;                     // Определяем переменную для хранения значения приближения
                                                   //
void setup() {                                     //
  Serial.begin(9600);                              // Инициируем передачу данных в монитор последовательного порта на скорости 9600 бит/сек
```

```
//  Подготавливаем переменные и функции для прерываний:        //
   pinMode(pinINT, INPUT);                                  // Переводим вывод pinINT в режим входа
   numINT = digitalPinToInterrupt(pinINT);                  // Определяем № внешнего прерывания для вывода pinINT
   attachInterrupt(numINT, funINT, FALLING);                // Задаём функцию funINT для обработки прерывания numINT. FALLING значит, что функция funINT будет вызываться при каждом спаде уровня сигнала на выводе pinINT с «1» в «0».
   if(numINT>=0){                                           // Если у вывода pinINT есть внешнее прерывание, то ...
        Serial.println("Pin interrupt OK!");                // Выводим сообщение об успешном выборе вывода прерывания
   }else{Serial.println("Pin interrupt ERROR!");}           // Иначе, выводим сообщение об ошибке выбранного вывода прерывания
                                                            //
//  Инициируем работу датчика:                              //
   if(apds.init()){                                         // Если инициализация прошла успешно, то ...
        Serial.println("Initialization OK!");               // Выводим сообщение об успешной инициализации датчика
   }else{Serial.println("Initialization ERROR!");}          // Иначе, выводим сообщение об ошибке инициализации датчика
                                                            //
//  Устанавливаем коэффициент усиления приёмника:           // Доступные значения: 1x, 2x, 4x, 8x (PGAIN_1X, PGAIN_2X, PGAIN_4X, PGAIN_8X). Чем выше коэффициент тем выше чувствительность
   if(apds.setProximityGain(PGAIN_2X)){                     // Если установлен коэффициент усиления приёмника в режиме определения расстояния, то ...
        Serial.println("Set gain OK!");                     // Выводим сообщение об успешной установке коэффициента усиления приёмника
```

}else{Serial.println("Set gain ERROR!");} // Иначе, выводим сообщение об ошибке при установке коэффициента усиления приёмника

// Прочитать установленный коэффициент усиления приёмника можно так: uint8_t i = apds.getProximityGain(); // в переменную i сохранится значение: PGAIN_1X, или PGAIN_2X, или PGAIN_4X, или PGAIN_8X

// Устанавливаем силу тока драйвера ИК-светодиода: // Доступные значения: 100мА, 50мА, 25мА, 12.5мА (LED_DRIVE_100MA, LED_DRIVE_50MA, LED_DRIVE_25MA, LED_DRIVE_12_5MA). Чем выше сила тока, тем выше чувствительность.

if(apds.setProximityDiode(LED_DRIVE_25MA)){ // Если установлена сила тока драйвера (яркость) ИК-светодиода для обнаружения приближения, то ...

Serial.println("Set LED drive OK!"); // Выводим сообщение об успешной установке силы тока драйвера

}else{Serial.println("Set LED drive ERROR!");} // Иначе, выводим сообщение об ошибке при установке силы тока драйвера

// Прочитать установленную силу тока можно так: uint8_t i = apds.getProximityDiode(); // в переменную i сохранится значение: LED_DRIVE_100MA, или LED_DRIVE_50MA, или LED_DRIVE_25MA, или LED_DRIVE_12_5MA

// Устанавливаем нижний порог определения приближения: // Значения приближения выше данного порога не будут приводить к возникновению прерываний на выводе INT

if(apds.setProximityIntLowThreshold(proximityIntLow)){ // Если установлен нижний порог прерываний, то ...

Serial.println("Set proximity low OK!"); // Выводим сообщение об успешной установке нижнего порога

}else{Serial.println("Set proximity low ERROR!");} // Иначе, выводим сообщение об ошибке при установке нижнего порога

// Прочитать нижний установленный порог можно так: int i; bool j = apds.getProximityIntLowThreshold(i); // в переменную i запишется порог, а в переменную j результат выполнения чтения (true/false)

// Устанавливаем верхний порог определения приближения: // Значения приближения ниже данного порога не будут приводить к возникновению прерываний на выводе INT

if(apds.setProximityIntHighThreshold(proximityIntHigh)){ // Если установлен верхний порог прерываний, то ...

Serial.println("Set proximity high OK!"); // Выводим сообщение об успешной установке верхнего порога

}else{Serial.println("Set proximity high ERROR!");} // Иначе, выводим сообщение об ошибке при установке верхнего порога

// Прочитать верхний установленный порог можно так: int i; bool j = apds.getProximityIntHighThreshold(i); // в переменную i запишется порог, а в переменную j результат выполнения чтения (true/false)

// Разрешаем режим определения приближения: //

if(apds.enableProximitySensor(true)){ // Если механизм определения приближения (true - с прерыванием на выходе INT) запущен, то ...

Serial.println("Start proximity sensor OK!"); // Выводим сообщение об успешном запуске механизма определения приближения

}else{Serial.println("Start proximity sensor ERROR!");} // Иначе, выводим сообщение об ошибке запуска механизма определения приближения

// Запретить работу механизма определения приближения можно так: bool j = apds.disableProximitySensor(); // в переменную j сохранится результат выполнения функции (true/false)

// Запрет или разрешение прерываний при определения приближения//

// apds.setProximityIntEnable(false); // Запрет разрешённых ранее прерываний от механизма определения приближения. Данная функция, как и представленные выше, так же возвращает true при успехе и false при неудаче

```
//              apds.setProximityIntEnable(true);              // Разрешение запрещённых ранее прерываний от механизма определения приближения. Данная функция, как и представленные выше, так же возвращает true при успехе и false при неудаче

//  uint8_t i = apds.getProximityIntEnable();                  // Чтение разрешены ли прерывания от механизма определения приближения. В переменную i запишется значение 0 или 1

                                                               //

//  Ждём завершение инициализации и калибровки:                //

   delay(500);                                                 //

}                                                              //

void loop(){                                                   //

   if(flgINT){ flgINT=0;                                       // Если установлен флаг flgINT (указывающий о том, что сработало прерывание), то сбрасываем его и ...

//  Читаем определённое датчиком значение приближения:         //

      if(apds.readProximity(proximityData)){                   // Если значение приближения корректно прочитано в переменную proximityData, то ..

         Serial.println((String) "Proximity="+proximityData);  // Выводим значение приближения

      }else{Serial.println("Reading proximity value ERROR!");}// Иначе, выводим сообщение об ошибке чтения приближения

//     Сообщаем модулю, сбросить прерывание с выхода INT:      //

      if(!apds.clearProximityInt()){                           // Если модуль НЕ сбросил прерывание с выхода INT после его установки как реакцию на приближение, то ...

         Serial.println("Clearing interrupt ERROR!");          // Выводим сообщение о том, что прерывание не сброшено

      }                                                        //

   }                                                           //

}
```

Работа кода показана на видео APDS9930+INT.mp4

Задание 2*:

```
// Пример простой генерации прерываний аппаратным таймером
#include "GyverTimers.h"
 int counter1Tim2 = 0;
 int counter2Tim2 = 0;
void setup() {
 Serial.begin(9600);
 //Timer1.setFrequency(2);             // Высокоточный таймер 1 для первого прерывания, частота - 3 Герца
 //Timer1.setPeriod(333333);          // то же самое! Частота 3 Гц это период 333 333 микросекунд
 Timer1.setFrequencyFloat(0.33);    // Если нужна дробная частота в Гц
 Timer1.enableISR();                 // Запускаем прерывание (по умолч. канал A)
 // запустим второй таймер
 Timer2.setPeriod(10000);      // Устанавливаем период таймера 10000 мкс -> 100 гц
 Timer2.enableISR(CHANNEL_A);   // Или просто .enableISR(), запускаем прерывание на канале A таймера 2
 pinMode(13, OUTPUT);          // будем мигать
 pinMode(2, OUTPUT);          // будем мигать
 pinMode(3, OUTPUT);          // будем мигать
 pinMode(4, OUTPUT);          // будем мигать
}
void loop() {}

// Прерывание A таймера 1
ISR(TIMER1_A) {
   digitalWrite(4, !digitalRead(4));
}
```

```
// Прерывание А таймера 2
ISR(TIMER2_A) {   // мигаем
  if (counter1Tim2 < 100){
  counter1Tim2++;
  }
  else
  { counter1Tim2 = 0;
    digitalWrite(2, !digitalRead(2));
  }
    if (counter2Tim2 < 200){
  counter2Tim2++;
  }
  else
  { counter2Tim2 = 0;
    digitalWrite(3, !digitalRead(3));
  }
}
```

Работа кода показана на видео Timers2.mp4