# ***ЛР 10. Программирование Автомат уличного освещения***

1. Изучить Автомат уличного освещения («Программирование микроконтроллерных плат Arduino» Улли Соммер, стр. 151)
2. Собрать схему выключателя уличного освещения
3. Создать скетч Автомат уличного освещения
4. Написать комментарии ко всем операторам цикла void loop()
5. Продемонстрировать преподавателю работу Автомата уличного освещения
6. Скриншоты всех этапов работы записать в отчет

int LED=13;  
int LDR=0;  
int Poti=1;  
int cnt=0;

void setup()  
{  
Serial.begin(9600);//открываем последовательное соединение  
pinMode(LED,OUTPUT);//со скоростью 9600 бод  
}

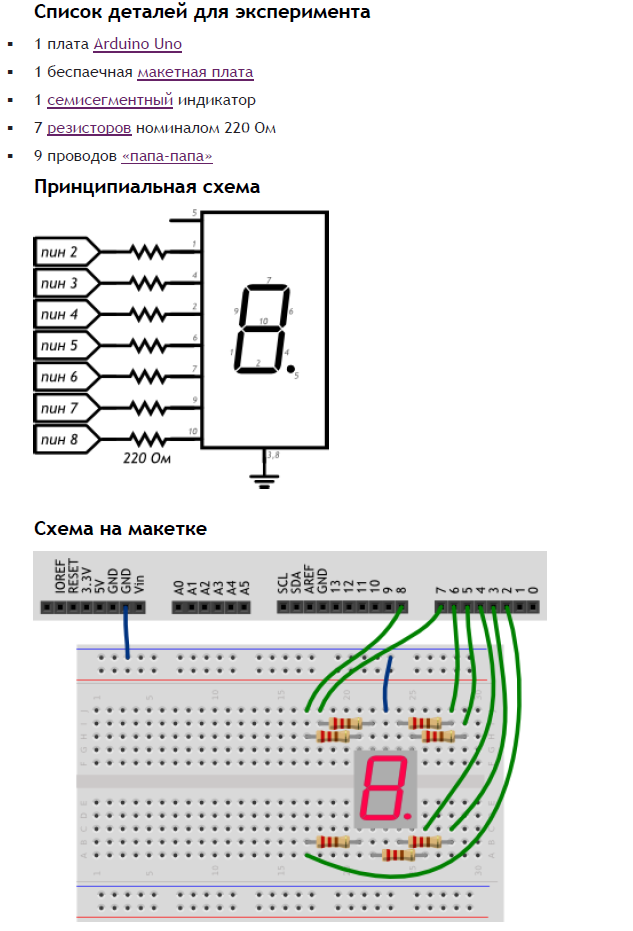
void loop()  
{  
if(analogRead(LDR)>analogRead(Poti))cnt=0;  
if(analogRead(LDR)<analogRead(Poti))cnt++;  
Serial.print(analogRead(LDR));  
Serial.print(" ");  
Serial.println(analogRead(Poti));  
if(cnt>300)  
{  
digitalWrite(LED,HIGH);  
do  
{  
delay(1000);  
Serial.print("Wnile");  
Serial.print(analogRead(LDR));  
Serial.print(" ");  
Serial.println(analogRead(Poti));  
}while(analogRead(LDR)<analogRead(Poti));  
cnt=0;  
//digitalWrite(LED,LOW);  
}  
else{ digitalWrite(LED,LOW); }  
  
delay(1000);  
}

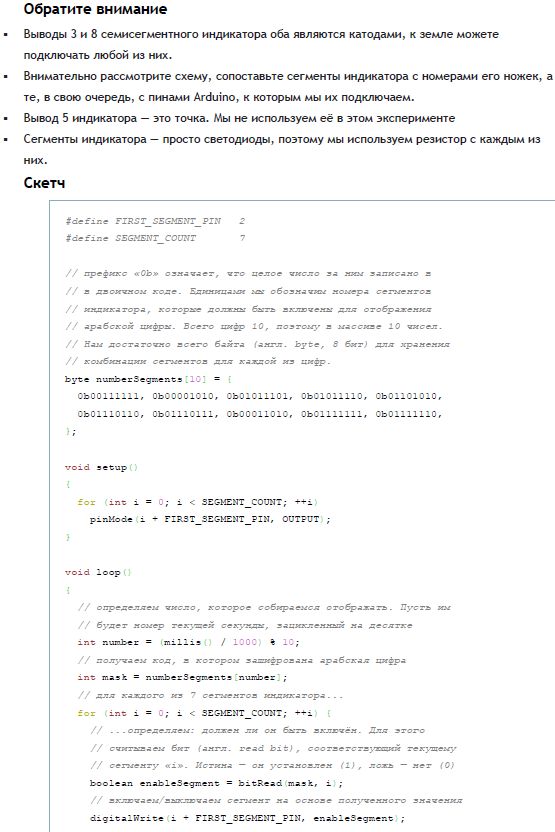
Комментарии к операторам цикла void loop() данного кода:

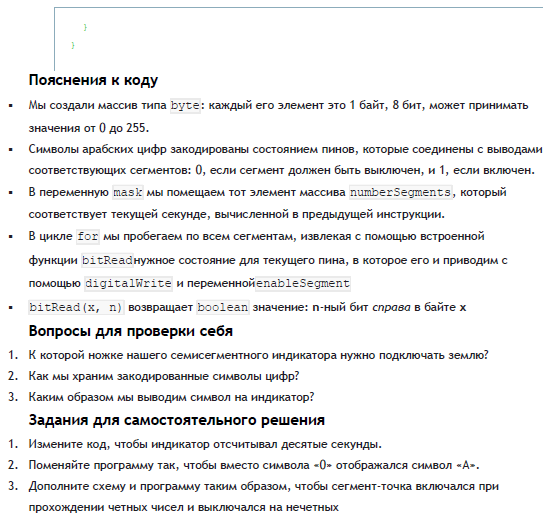
1. if(analogRead(LDR)>analogRead(Poti))cnt=0; - Если значение, считанное с датчика света (LDR), больше значения, считанного с потенциометра (Poti), то счетчик cnt устанавливается в 0.
2. if(analogRead(LDR)<analogRead(Poti))cnt++; - Если значение, считанное с LDR, меньше значения, считанного с Poti, то к счетчику cnt добавляется 1.
3. Serial.print(analogRead(LDR)); - Выводит значение, считанное с LDR, на последовательный порт.
4. Serial.print(" "); - Выводит пробел на последовательный порт.
5. Serial.println(analogRead(Poti)); - Выводит значение, считанное с Poti, на последовательный порт с переводом строки.
6. if(cnt>300) - Если значение счетчика cnt больше 300, выполняется следующий блок кода.
7. digitalWrite(LED,HIGH); - Устанавливает высокий уровень на пине LED, включая светодиод.
8. do { ... } while(analogRead(LDR)<analogRead(Poti)); - Выполняет блок кода внутри do-while цикла, пока значение считанное с LDR меньше значения считанного с Poti.
9. cnt=0; - Устанавливает значение счетчика cnt в 0.
10. digitalWrite(LED,LOW); - Устанавливает низкий уровень на пине LED, выключая светодиод.
11. else { digitalWrite(LED,LOW); } - Если условие в пункте 6 не выполняется, устанавливает низкий уровень на пине LED, выключая светодиод.
12. delay(1000); - Задержка в 1 секунду перед повторением цикла.

# ***ЛР 11. Программирование Секундомер***

1. Изучить Эксперимент 13. Секундомер (Конспект хакера Эксперименты)
2. Собрать схему Секундомер
3. Создать скетч Секундомер
4. Написать комментарии ко всем операторам
5. Продемонстрировать преподавателю работу Секундомер
6. Скриншоты всех этапов работы записать в отчет







#define FIRST\_SEGMENT\_PIN 2

#define SEGMENT\_COUNT 7

// префикс «0b» означает, что целое число за ним записано в

// в двоичном коде. Единицами мы обозначим номера сегментов

// индикатора, которые должны быть включены для отображения

// арабской цифры. Всего цифр 10, поэтому в массиве 10 чисел.

// Нам достаточно всего байта (англ. byte, 8 бит) для хранения

// комбинации сегментов для каждой из цифр.

byte numberSegments[10] = {

0b00111111, 0b00001010, 0b01011101, 0b01011110, 0b01101010,

0b01110110, 0b01110111, 0b00011010, 0b01111111, 0b01111110,

};

void setup()

{

for (int i = 0; i < SEGMENT\_COUNT; ++i)

pinMode(i + FIRST\_SEGMENT\_PIN, OUTPUT);

}

void loop()

{

// определяем число, которое собираемся отображать. Пусть им

// будет номер текущей секунды, зацикленный на десятке

int number = (millis() / 1000) % 10;

// получаем код, в котором зашифрована арабская цифра

int mask = numberSegments[number];

// для каждого из 7 сегментов индикатора...

for (int i = 0; i < SEGMENT\_COUNT; ++i) {

// ...определяем: должен ли он быть включён. Для этого

// считываем бит (англ. read bit), соответствующий текущему

// сегменту «i». Истина — он установлен (1), ложь — нет (0)

boolean enableSegment = bitRead(mask, i);

// включаем/выключаем сегмент на основе полученного значения

digitalWrite(i + FIRST\_SEGMENT\_PIN, enableSegment);

}

}

Комментарии к операторам данного кода:

1. #define FIRST\_SEGMENT\_PIN 2 - Определяет пин, на котором подключен первый сегмент индикатора.
2. #define SEGMENT\_COUNT 7 - Определяет количество сегментов в индикаторе.
3. byte numberSegments[10] = {...}; - Определяет массив, в котором хранятся комбинации сегментов для каждой из арабских цифр от 0 до 9.
4. void setup() - Начало функции настройки.
5. for (int i = 0; i < SEGMENT\_COUNT; ++i) - Цикл, который выполняется для каждого сегмента индикатора.
6. pinMode(i + FIRST\_SEGMENT\_PIN, OUTPUT); - Устанавливает пин с номером i + FIRST\_SEGMENT\_PIN в режим вывода.
7. void loop() - Начало функции цикла.
8. int number = (millis() / 1000) % 10; - Вычисляет значение переменной number как остаток от деления текущего времени в миллисекундах на 1000, затем берет остаток от деления этого значения на 10. Это позволяет получить номер текущей секунды, зацикленный на десятке.
9. int mask = numberSegments[number]; - Присваивает переменной mask значение из массива numberSegments, соответствующее номеру текущей секунды.
10. for (int i = 0; i < SEGMENT\_COUNT; ++i) - Цикл, который выполняется для каждого сегмента индикатора.
11. boolean enableSegment = bitRead(mask, i); - Считывает бит из переменной mask, соответствующий текущему сегменту "i", и присваивает его значение переменной enableSegment. Если бит равен 1, то enableSegment будет true, если бит равен 0, то enableSegment будет false.
12. digitalWrite(i + FIRST\_SEGMENT\_PIN, enableSegment); - Включает или выключает сегмент на пине с номером i + FIRST\_SEGMENT\_PIN в зависимости от значения переменной enableSegment.

# ***ЛР 12. Работа с текстовым дисплеем***

1. Изучить Эксперимент 18. Тестер батареек (Конспект хакера Эксперименты)
2. Собрать схему Тестер батареек - БЕЗ клемм
3. Создать скетч, который будет выводить на дисплей ФИО студента кириллицей (бегущая строка)
4. Написать комментарии ко всем операторам
5. Продемонстрировать преподавателю работу - бегущая строка
6. Скриншоты всех этапов работы записать в отчет

// Подключаем библиотеку для работы с жидкокристаллическим

// экраном (англ. Liquid Crystal Display или просто LCD)

#include <LiquidCrystal.h>

// на диоде, защищающем от неверной полярности, падает доля

// напряжения (англ. voltage drop). Необходимо это учитывать

#define DIODE\_DROP 0.7

// Объявляем объект, для управления дисплеем. Для его создания

// необходимо указать номера пинов, к которым он подключен в

// порядке: RS E DB4 DB5 DB6 DB7

LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

void setup()

{

// начинаем работу с экраном. Сообщаем объекту количество

// строк и столбцов. Опять же, вызывать pinMode не требуется:

// функция begin сделает всё за нас

lcd.begin(16, 2);

// печатаем сообщение на первой строке

lcd.print("Battery voltage:");

}

void loop()

{

// высчитываем напряжение подключенной батарейки

float voltage = analogRead(A0) / 1024.0 \* 10.0;

// если напряжение на делителе напряжения было зафиксировано,

// нужно прибавить напряжение на диоде, т.к. оно было съедено

if (voltage > 0.1)

voltage += DIODE\_DROP;

// устанавливаем курсор, колонку 0, строку 1. На деле — это

// левый квадрат 2-й строки, т.к. нумерация начинается с нуля

lcd.setCursor(0, 1);

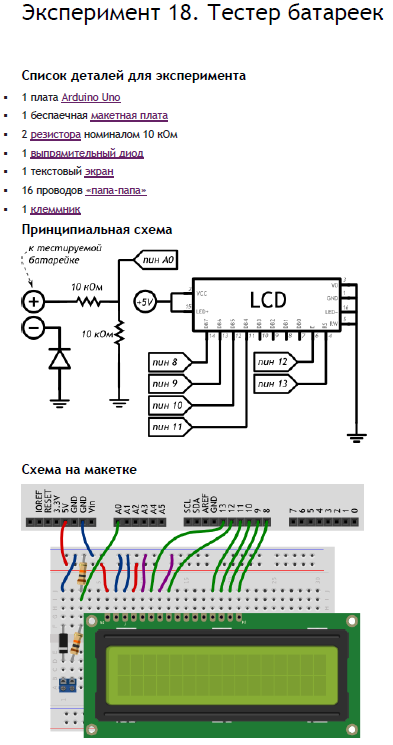
// печатаем напряжение в батарейке с точностью до сотых долей

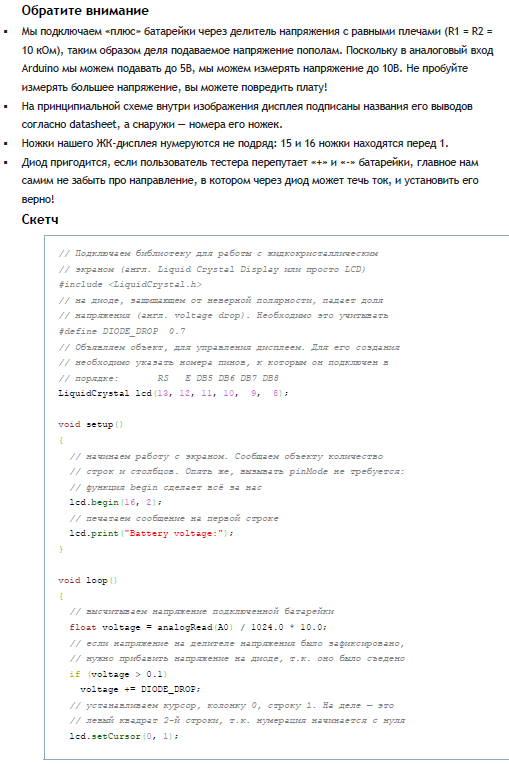
lcd.print(voltage, 2);

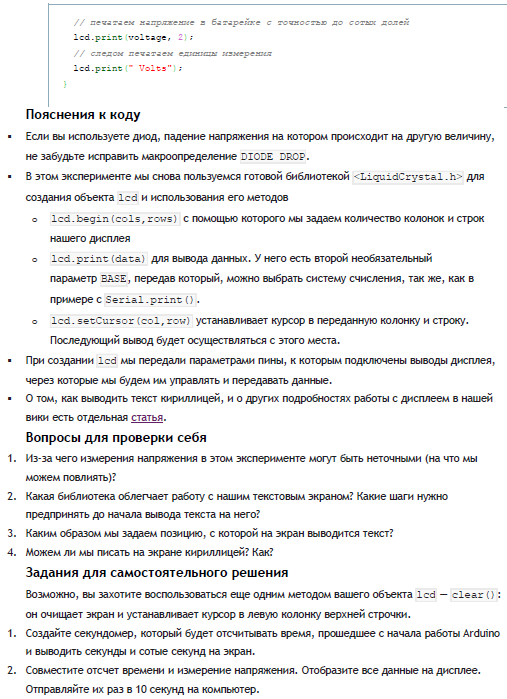
// следом печатаем единицы измерения

lcd.print(" Volts");

}









#include <Wire.h>   
#include <LiquidCrystalRus.h>  
  
#define DIODE\_DROP 0.7  
  
LiquidCrystalRus lcd(13, 12, 11, 10, 9, 8);  
  
byte heart[8] = { 0b00000, 0b01010, 0b11111, 0b11111, 0b11111, 0b01110, 0b00100, 0b00000 };  
  
void setup() {  
// Устанавливаем размер экрана  
// Количество столбцов и строк  
lcd.createChar(1, heart);  
}  
  
void loop() {  
  
for (int i = -18; i < 18; i++) {  
lcd.setCursor(i, 0);  
lcd.print(char(1));  
lcd.print("Гринцевич Юлия");  
lcd.print(char(1));  
delay(200);  
lcd.clear(); // очищаем экран  
  
}   
// передвигаем надпись влево  
for (int i = -18; i < 18; i++) {  
lcd.setCursor(i, 0);  
lcd.print(char(1));  
lcd.print("Чаган Алина ");  
lcd.print(char(1));  
delay(200);  
lcd.clear(); // очищаем экран  
}  
  
}

# ***ЛР 13. Программирование Бегущий огонек***

1. Изучить Эксперимент 7. Бегущий огонек (Конспект хакера Эксперименты)
2. Собрать схему Бегущий огонек
3. Создать скетч Бегущий огонек
4. Написать комментарии ко всем операторам
5. Продемонстрировать преподавателю работу Бегущий огонек
6. Изменить скетч так, чтобы светодиоды переключались раз в секунду.
7. Продемонстрировать преподавателю работу Бегущий огонек
8. Скриншоты всех этапов работы записать в отчет

// светодиодная шкала подключена к группе пинов расположенных

// подряд. Даём понятные имена первому и последнему пинам

#define FIRST\_LED\_PIN 2

#define LAST\_LED\_PIN 11

void setup()

{

// в шкале 10 светодиодов. Мы бы могли написать pinMode 10

// раз: для каждого из пинов, но это бы раздуло код и

// сделало его изменение более проблематичным.

// Поэтому лучше воспользоваться циклом. Мы выполняем

// pinMode для (англ. for) каждого пина (переменная pin)

// от первого (= FIRST\_LED\_PIN) до последнего включительно

// (<= LAST\_LED\_PIN), всякий раз продвигаясь к следующему

// (++pin увеличивает значение pin на единицу)

// Так все пины от 2-го по 11-й друг за другом станут выходами

for (int pin = FIRST\_LED\_PIN; pin <= LAST\_LED\_PIN; ++pin)

pinMode(pin, OUTPUT);

}

void loop()

{

// получаем время в миллисекундах, прошедшее с момента

// включения микроконтроллера

unsigned int ms = millis();

// нехитрой арифметикой вычисляем, какой светодиод

// должен гореть именно сейчас. Смена будет происходить

// каждые 120 миллисекунд. Y % X — это остаток от

// деления Y на X; плюс, минус, скобки — как в алгебре.

int pin = FIRST\_LED\_PIN + (ms / 120) % 10;

// включаем нужный светодиод на 10 миллисекунд, затем —

// выключаем. На следующем проходе цикла он снова включится,

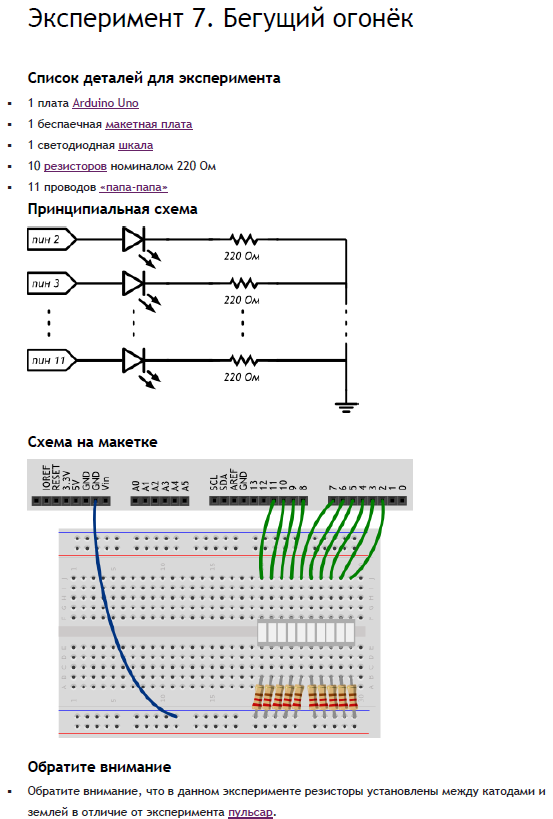
// если гореть его черёд, и мы вообще не заметим отключения

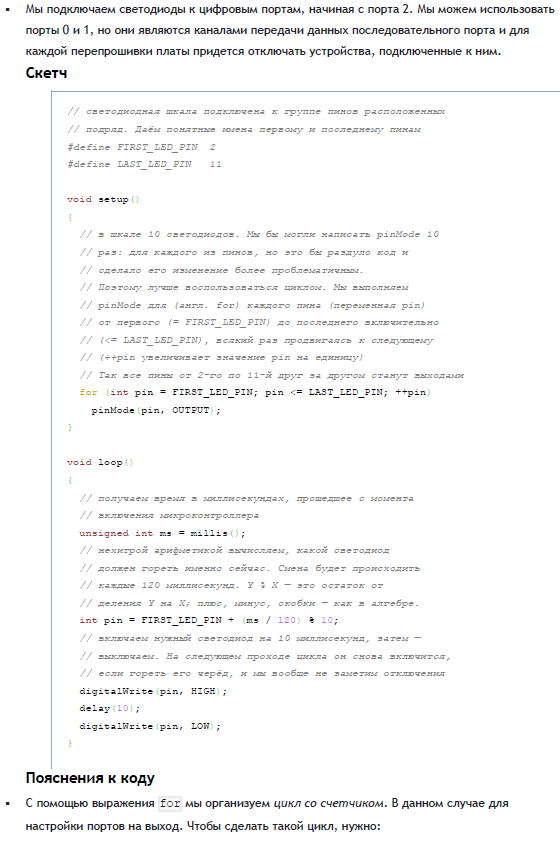
digitalWrite(pin, HIGH);

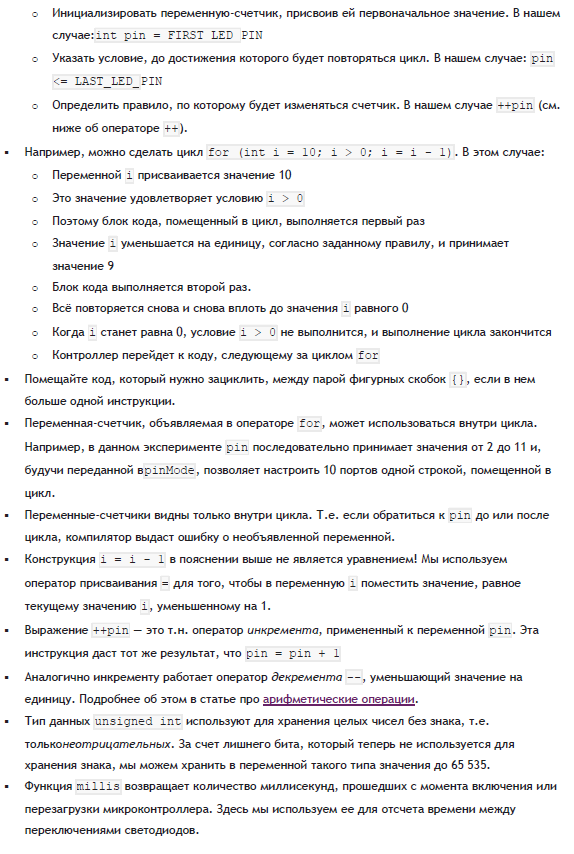
delay(10);

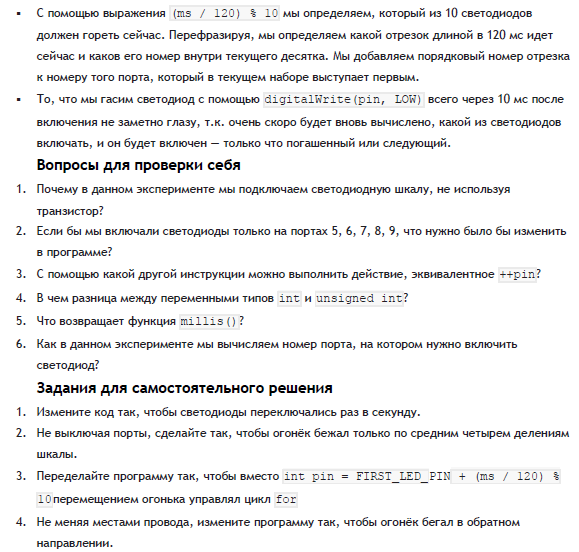
digitalWrite(pin, LOW);

}









Я попробую изменить следующий код, чтобы светодиоды переключались раз в секунду. Вот мой вариант:

#define FIRST\_LED\_PIN 2

#define LAST\_LED\_PIN 11

void setup()

{

// в шкале 10 светодиодов. Мы бы могли написать pinMode 10

// раз: для каждого из пинов, но это бы раздуло код и

// сделало его изменение более проблематичным.

// Поэтому лучше воспользоваться циклом. Мы выполняем

// pinMode для (англ. for) каждого пина (переменная pin)

// от первого (= FIRST\_LED\_PIN) до последнего включительно

// (<= LAST\_LED\_PIN), всякий раз продвигаясь к следующему

// (++pin увеличивает значение pin на единицу)

// Так все пины от 2-го по 11-й друг за другом станут выходами

for (int pin = FIRST\_LED\_PIN; pin <= LAST\_LED\_PIN; ++pin)

pinMode(pin, OUTPUT);

}

void loop()

{

// получаем время в миллисекундах, прошедшее с момента

// включения микроконтроллера

unsigned int ms = millis();

// нехитрой арифметикой вычисляем, какой светодиод

// должен гореть именно сейчас. Смена будет происходить

// каждые 1000 миллисекунд. Y % X — это остаток от

// деления Y на X; плюс, минус, скобки — как в алгебре.

int pin = FIRST\_LED\_PIN + (ms / 1000) % 10;

// включаем нужный светодиод на 10 миллисекунд, затем —

// выключаем. На следующем проходе цикла он снова включится,

// если гореть его черёд, и мы вообще не заметим отключения

digitalWrite(pin, HIGH);

delay(10);

digitalWrite(pin, LOW);

}

Надеюсь, это то, что вы хотели. 😊

Вот измененный код, чтобы светодиоды переключались раз в секунду:

```c++

#define FIRST\_LED\_PIN 2

#define LAST\_LED\_PIN 11

void setup()

{

for (int pin = FIRST\_LED\_PIN; pin <= LAST\_LED\_PIN; ++pin)

pinMode(pin, OUTPUT);

}

void loop()

{

unsigned int ms = millis();

int pin = FIRST\_LED\_PIN + (ms / 1000) % 10; // изменено значение 120 на 1000 для переключения раз в секунду

digitalWrite(pin, HIGH);

delay(500); // изменено значение 10 на 500 для задержки в полсекунды

digitalWrite(pin, LOW);

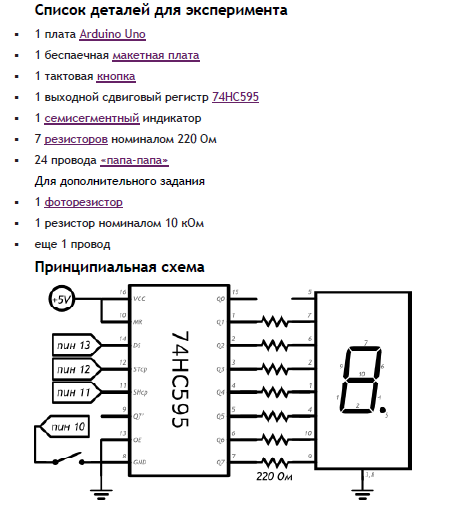
delay(500); // изменено значение 10 на 500 для задержки в полсекунды

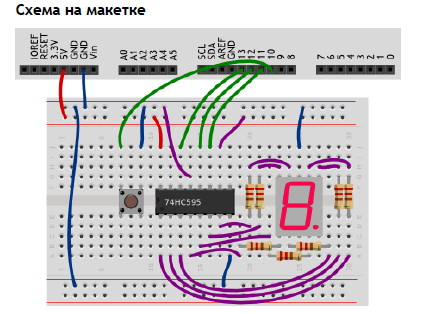
}

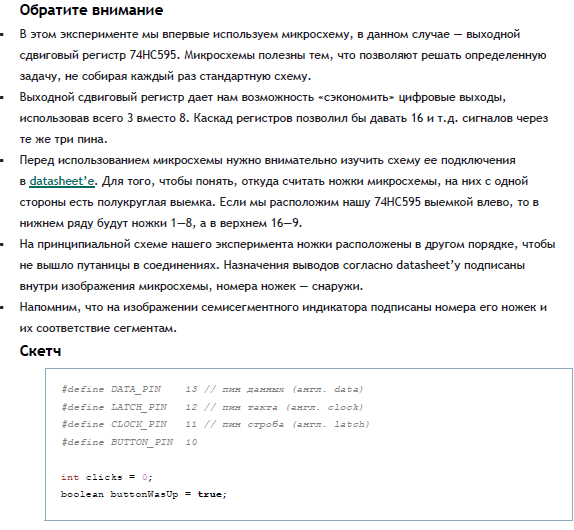
```

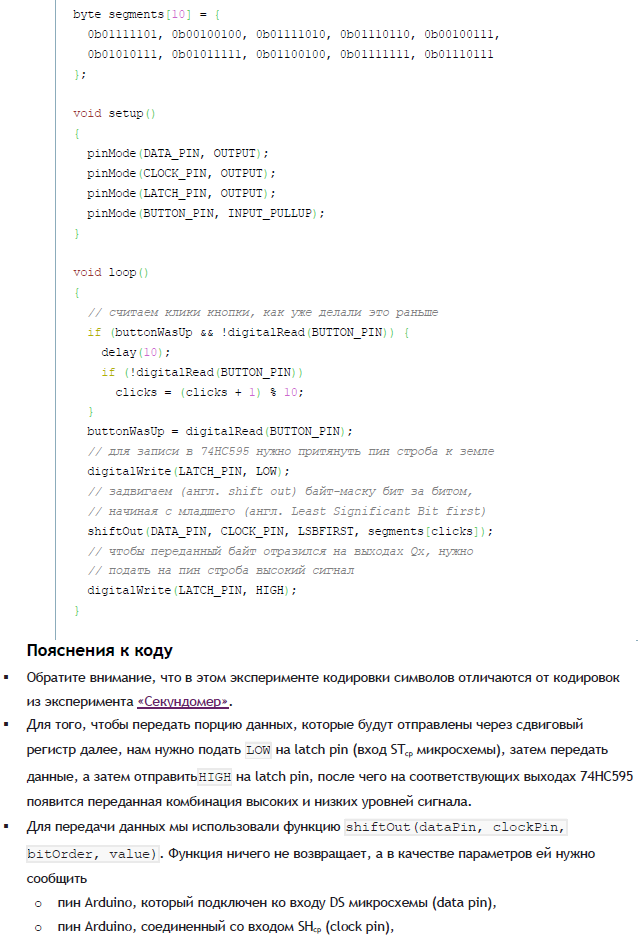
В этом измененном коде светодиоды будут переключаться каждую секунду. Сначала будет включен светодиод, соответствующий пину `pin`, затем он будет выключен, и будет задержка в полсекунды перед следующим переключением.

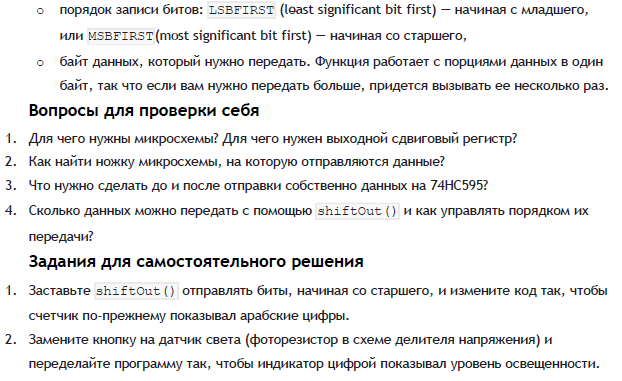
# ***ЛР 14. Счетчик нажатий***











#define DATA\_PIN 13 // пин данных (англ. data)

#define LATCH\_PIN 12 // пин строба (англ. latch)

#define CLOCK\_PIN 11 // пин такта (англ. clock)

#define BUTTON\_PIN 10

int clicks = 0;

boolean buttonWasUp = true;

byte segments[10] = {

0b01111101, 0b00100100, 0b01111010, 0b01110110, 0b00100111,

0b01010111, 0b01011111, 0b01100100, 0b01111111, 0b01110111

};

void setup()

{

pinMode(DATA\_PIN, OUTPUT);

pinMode(CLOCK\_PIN, OUTPUT);

pinMode(LATCH\_PIN, OUTPUT);

pinMode(BUTTON\_PIN, INPUT\_PULLUP);

}

void loop()

{

// считаем клики кнопки, как уже делали это раньше

if (buttonWasUp && !digitalRead(BUTTON\_PIN)) {

delay(10);

if (!digitalRead(BUTTON\_PIN))

clicks = (clicks + 1) % 10;

}

buttonWasUp = digitalRead(BUTTON\_PIN);

// для записи в 74HC595 нужно притянуть пин строба к земле

digitalWrite(LATCH\_PIN, LOW);

// задвигаем (англ. shift out) байт-маску бит за битом,

// начиная с младшего (англ. Least Significant Bit first)

shiftOut(DATA\_PIN, CLOCK\_PIN, LSBFIRST, segments[clicks]);

// чтобы переданный байт отразился на выходах Qx, нужно

// подать на пин строба высокий сигнал

digitalWrite(LATCH\_PIN, HIGH);

}