# Вступление

Всего разработано и имеется описание 5 функций:

showWhithout1  
setDivergenceRand  
divergenceMeter  
pushButton  
navigate

Таблица глобальных переменных

|  |  |
| --- | --- |
| Имена переменных | Семантика переменных |
| out1 – out8 | Пины дешифратора |
| key1 – key6 | Пины ламп |
| lampKeys | Массив номеров пинов ламп |
| digits | Массив для текущего значения времени на шесть цифр |
| mainMode | True - режим часов, false - режим календаря |
| AntiRattlingTimer | Таймер для устранения дребезга кнопки |
| pushedKey | Нажатая кнопка |
| blinkTimer | Массив времени начала мигания для каждой цифры |
| flagBlink | Индикатор мигания для каждой цифры |
| previousNumber | Предыдущее значение для каждой цифры |
| delaysArr | Массив для мигания индикаторов (хранит время задержки на пол-секунды мигания) |

# Описание функций

Таблица функций

|  |  |
| --- | --- |
| Идентификатор | Семантика |
| navigate | Используется для переключения режима работы часов, принимает в качестве параметра номер режима и вызывает соответствующую функцию, для режима часов возвращается в функцию *loop*. |
| setDivergenceRand | Устанавливает случайные значения для индикаторов 3 – 6, и рассчитывает значение первого индикатора (тоже случайное, но с расчетной вероятностью от 10% до 1% в зависимости от значения 3-го индикатора). |
| divergenceMeter | Режим дивергенсметра, значение обновляется при нажатии кнопки 1. |
| pushButton | Возвращает номер нажатой кнопки или 0, если нажатия нет, а так же защищает от дребезга кнопок. |
| showWhithout1 | Функция используется для отображения цифр на индикаторах без использования 2-го индикатора (при отсчете с 1). |

# Алгоритмы функций

Описание функции "setDivergenceRand"

В функцию передаются:

* *maxTimeLoop* - максимальное время перебора значений на индикаторах в миллисекундах
* *stepUpdate* – время задержки до следующего обновления значения индикатора

1. Объявление переменных
   1. *timeOff* - время перебора значений для индикаторов 3-6
   2. *realMaxTime* – максимальное значение из массива *timeOff*
2. Заполнение в цикле массива *timeOff* с помощью функции *random(maxTimeLoop/4, maxTimeLoop)* и запись максимального в переменную *realMaxTime*
3. Запускается таймер *startLoop*.
4. Запускается таймер *updateLoop*.
5. Если *startLoop* меньше максимального времени обновления индикатора
   1. Если *updateLoop* меньше *stepUpdate*
      1. Перезапуск таймера *updateLoop*
      2. *digits[0] = random(0, 10);*
      3. Перебор индикаторов 3-6
         1. Если *startLoop* меньше времени обновления для данного индикатора
            1. Индикатору присваивается случайное значение диапазона 1-9
   2. Вывод значений на индикаторы (*showWhithout1()*)
6. Рассчет значения первого индикатора (*digits[0] = (random(0, 101) > 90 + digits[2])? 1:0*)
7. Вывод текущего значения дивергенсметра в течении 160 миллисекунд
8. Задержка на 80 миллисекунд
9. Выход из функции

# Код функций с комментариями

**navigate**

void navigate(int funcKey){

while(funcKey){

if(funcKey == diverKey)

funcKey = divergenceMeter();

else if(funcKey == setupKey)

funcKey = setTime\_Date();

}

if(mainMode)

updateTime();

else

updateDate();

}

**setDivergenceRand**

void setDivergenceRand(int maxTimeLoop = random(800, 2500), int stepUpdate = 80){

int timeOff[4]; // Время перебора значений для индикаторов 3-6

int realMaxTime = 0;

for(int i = 0; i < 4; i++){

timeOff[i] = random(maxTimeLoop/4, maxTimeLoop);

if(realMaxTime < timeOff[i])

realMaxTime = timeOff[i];

}

unsigned long startLoop = millis();

unsigned long updateLoop = startLoop - stepUpdate;

while(millis() - startLoop < realMaxTime){

if(millis() - updateLoop > stepUpdate){

updateLoop = millis();

digits[0] = random(0, 10);

for(int i = 2; i < 6; i++){

if(millis() - startLoop < timeOff[i-2])

digits[i] = random(0, 10);

}

}

showWhithout1();

}

digits[0] = (random(0, 101) > 90 + digits[2])? 1:0;

**divergenceMeter**

int divergenceMeter(){

setDivergenceRand();

while(true){

showWhithout1();

if(pushedKey = pushButton()){

if(pushedKey == 1)

setDivergenceRand();

else if(pushedKey == 2)

return loopKey;

else if(pushedKey == 3)

return setupKey;

}

}

}

**pushButton**

int pushButton(){

int MyButtonValue = analogRead(keyb);

// наиболее вероятно, что кнопка не нажата

if(MyButtonValue < 200)

return 0;

// устраняем дребезжание кнопки

if(millis() - AntiRattlingTimer < 250)

return 0;

AntiRattlingTimer = millis();

if(MyButtonValue > 900 && MyButtonValue < 1100)

return 1;

else if(MyButtonValue > 600 && MyButtonValue < 800)

return 2;

else if(MyButtonValue > 300 && MyButtonValue < 500)

return 3;

else

return 0;

}

**showWhithout1**

void showWhithout1(int a = 1){

for(int i = 0 ; i < 6; i++){

if(i != a){

setNumber(digits[i]);

digitalWrite(lampKeys[i],HIGH);

delay(2);

digitalWrite(lampKeys[i],LOW);

delay(1);

}

else

delay(3);

}

}