**Документация**

по проект **WAVEsound**

към програма ИТ кариера

1. Тема/идея на проекта:

**WAVEsound –** модул за визуализиране на нивото на аудио сигнал, управляваш LED лента спрямо силата на звук (музика), приета през микрофон (MIC) или звуков вход (AUX).

1. Автори:

* Мариан Балтов;
* Христо Ковачев;
* Елица Лазарова;
* Павел Драганов;
* Теодор Николов;

1. Структура на проекта:
   1. Хардуер(елементи): LED лента(WS2812), Arduino UNO, аудио формировател състоящ се от микрофонен предусилва-тел, линеен предусилвател, регулатори на ниво, суматор и изходен буфер
   2. Принцип: приема се звук през микрофона или звуковия вход. След приемането звука се усилва до необходимите нива за нормална работа с Arduino. Приетите сигнали се дозират по желание на опериращия и се сумират в общ сигнал. Полученият сигнал се измества спрямо нужното ниво за опериране с Arduino от където продължава софтуерната обработка на сигнала в основния компонент на проекта – Arduino. От там се изпраща сигнал към LED лентата, който се преобразува с светлинен.

Amplifierr

Amplifierierr

Attenuation

Attenuation

SUM

Level shifter

MIC

AUX

Dynamic LED control

Arduino UNO

LEDs x 30

Ain 2

Ain 1

Последователност:

* Измерват се стойността на канал 2, който определя динамиката на опресняване на светодиодната лента;
* Приетата стойност се премащабира към обхват 1/20 и се присвоява на променлива, която определя броя на калкулираните RMS стойности, а те от своя страна определят скоростта на опресняване на светодиодната лента;
* Измерване на стойността от канал 1 на ADC;
* Корекция на постоянно токовата съставка (измерената стойност - 512);
* Вдигане на квадрат на коригираната стойност;
* Прибавяне на квадрата към сумата

1. Софтуер

***Github:***

<https://github.com/icokovachev/WaveSound>

***Легенда:***

***А - библиотеки***

***А - дефиниции***

***А - константи***

***А - променливи***

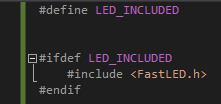
***А – функции***

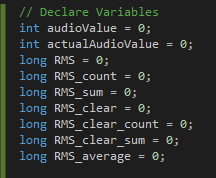
***А- методи***

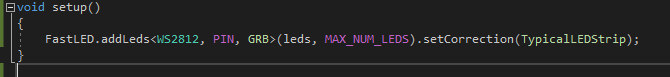
.ino файлът, съдържащ кода е поместен в : ~/WaveSound/WaveSound/WaveSound.ino

Използвани са ресурси от сайта на Arduino, FastLED, и форуми.

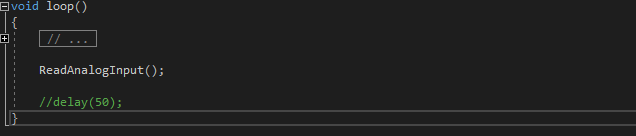
1. Библиотеки и глобални променливи
   1. Библиотеки
      1. FastLED - <http://fastled.io/>
         1. Библиотека за контрол на различни видове LED ленти. /При нас се използва за контол на WS2812B LED летна/
         2. Библиотеката е дефинирана в ifdef, защото ако нямаме дефинирана лед лента /#define LED\_INCLUDED/, библиотеката не се използва



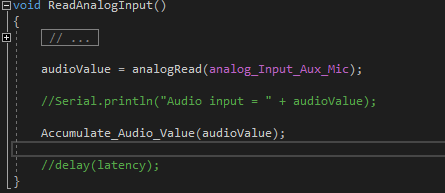
* 1. Дефинирани основни елементи
     1. #define LED\_INCLUDED – дефинира се за да декларира използването на LED лента
     2. #define analog\_Input\_Aux\_Mic A1 - задава се пинът, от който ще четем аналогов сигнал от микрофона и AUX вхова
     3. #define analog\_Input\_Refresh\_Rate\_LED A2 – задава се пинът, от който ще четем аналогов сигнал подаден от потенциометъра, който контролира обновяването на LED лентата
     4. #define latency 50 – задава се забавяне между четенията от аналоговите сигнали /време за изпълнение на останалата част от функционалностите/
     5. #define MAX\_NUM\_LEDS 30 – дефинират се броя на пикселите върху LED лентата /30 на брой в конкретния проект/
     6. #define PIN 8 – задава се пин за предаване на информацията към LED лентата
  2. Деклариране на глобални променливи
     1. Константи
        1. long RMS\_count\_max=50; - Задава броя на RMS, които ще бъдат прочетени и дадени на RMS\_clear за усредняване.
        2. long RMS\_clear\_count\_max = 20; - задава броя на RMS\_clear, които ще бъдат сметнати.
     2. Променливи
        1. int audioValue = 0; - Прочита число в порядъка 0-1023 /10 bit/ от пин А1 с функцията analogRead() като използваме ADC /Analog-to-Digital Converter/
        2. int actualAudioValue = 0; - Тъй като „нулата“ ни е 512 /+2,5V -2,5V/, за да получим реален инпут трябва да извадим от полученото число 512.
        3. long RMS = 0; - 1 RMS = actualAudioValue \* actualAudioValue
        4. long RMS\_count = 0; - брояч, който следи колко RMS стойности са приети /Ресетва се, когато стигне RMS\_count\_max/
        5. long RMS\_sum = 0; - Сумата на стойностите на RMS; RMS\_sum += RMS;
        6. long RMS\_clear = 0; - Получава се от средната стойност на буфера – RMS\_sum / RMS\_count. Използва се да контролира колко входни данни ще дадем на LED лентата в последствие.
        7. long RMS\_clear\_count = 0; - брояч, който следи колко стойности на RMS\_clear са събрани. /Ресетва се, когато стигне RMS\_clear\_count\_max/
        8. long RMS\_clear\_sum = 0; - Сумата на стойностите на RMS\_clear; RMS\_clear\_sum += RMS\_clear;
        9. long RMS\_average = 0; - Получава се от средната стойност на RMS\_clear – RMS\_average = RMS\_clear\_sum / RMS\_clear\_count;
  3. void setup(){}
     1. В setup() си инициализираме LED лентата според изискванията на FastLED библиотеката.



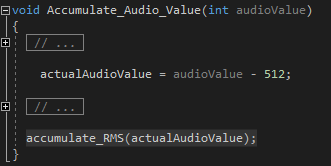
* 1. void loop() {}
     1. В loop() имаме извикване на ReadAnalogInput() метода.



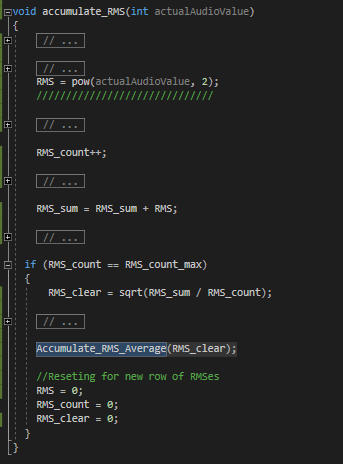
* 1. void ReadAnalogInput(){}
     1. ReadAnalogInput() метода инициализира audioValue променливата и ѝ дава стойност, която получава от analogRead(analoig\_Input\_Aux\_Mic) функцията. След което извиква Accumulate\_Audio\_Value(audioValue) като за входни данни му дава audioValue.



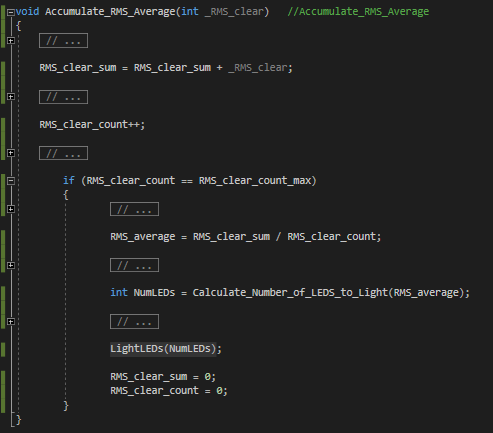
* 1. void Accumulate\_Audio\_Value(int audioValue)
     1. void Accumulate\_Audio\_Value(int audioValue) взима генерираната от ReadAnalogInput() audioValue и изважда от нея 512, за да получи чистия инпут като цяло число и я записва в actualAudioValue, след което вика accumulate\_RMS(actualAudioValue); метода и му дава получената променлива.



* 1. void accumulate\_RMS(int actualAudioValue)
     1. void accumulate\_RMS(int actualAudioValue) взима получената от предходния метод actualAudioValue и като я повдигне на квадрат получава RMS. След това броячът RMS\_count се увеличава с 1. Сегашната получена стойност за RMS се добавя към RMS\_sum. Ако условието на функцията if(RMS\_count == RMS\_count\_max) e изпълнено, се пресмята RMS\_clear като се коренува средното на RMS\_sum - (). След това се извиква метода Accumulate\_RMS\_Average(RMS\_clear) и му се подава RMS\_clear. Зануляват се променливите RMS, RMS\_count, RMS\_clear.



* 1. void Accumulate\_RMS\_Average(int \_RMS\_clear)
     1. void Accumulate\_RMS\_Average(int \_RMS\_clear) взима RMS\_clear – променливата, която се дава от void accumulate\_RMS(int actualAudioValue) и начислява сумата на получената променлива към RMS\_clear\_sum, където се съхранява сбора на всички получени до сега от RMS\_clear стойности. Броячът RMS\_clear\_count се увеличава с 1 всеки път, когато методът е извикан. Следва функцията if (RMS\_clear\_count == RMS\_clear\_count\_max). Ако условието ѝ е удовлетворено се смята средната стойност на сборът на получените до сега RMS\_clear стойности - и получената стойност записва като RMS\_average. Тази стойност на променливата се подава на Calculate\_Number\_of\_LEDS\_to\_Light(RMS\_average) метода. Полученото число след извикването на този метод се записва като int NumLEDs, която се подава на LightLEDs(NumLEDs) метода. Зануляват се RMS\_clear\_sum и RMS\_clear\_count.



* 1. int Calculate\_Number\_of\_LEDS\_to\_Light(int RMS\_average)
     1. int Calculate\_Number\_of\_LEDS\_to\_Light(int RMS\_average) връща int като стойност. Приема RMS\_average като входни данни и пресмята (RMS\_average \* MAX\_NUM\_LEDS)/1024. Стойността от пресмятането е записана в int leds\_to\_light.

1. Логика

