ВИСШЕ ВОЕННОМОРСКО УЧИЛИЩЕ
“Никола Йонков Вапцаров” гр. Варна

ДОКУМЕНТАЦИЯ НА IOT ПРОЕКТ № 55 “URBAN NOISE MONITORING – TRACK NOISE LEVELS IN DIFFERENT AREAS”

Антон Евгениев Атанасов

Гр. Варна
2024

1. Увод

Проектът за следене на градския шум има за цел да проследява и анализира нивата на шум в различни градски райони. Тази инициатива е създадена, за да идентифицира зони с високо ниво на шумово замърсяване, като по този начин подпомага градското планиране и подобрява качеството на живот на жителите. За изпълнението на проекта са необходими няколко ключови компонента.

Основната обработваща единица за всяка мониторингова станция ще бъде микроконтролер, като например Arduino Uno. За измерване на околното шумово ниво ще се използват звукомери, оборудвани със сензори за ниво на звука, като микрофон. Точната локация на измерванията на шума ще се записва чрез GPS модули. Данните ще се предават към централен сървър посредством комуникационни модули, като Wi-Fi или клетъчни модули. За захранване на устройствата ще се използват батерии или соларни панели, а за защитата на оборудването ще се осигуря водоустойчив корпус. Съхранението и обработката на данните ще се осъществяват чрез облачен базиран или локален сървър.

Проектът обхваща няколко основни аспекта. Непрекъснатото измерване на нивата на шума ще се извършва на различни градски локации. Автоматизираното събиране и предаване на данни за шума, заедно с информация за местоположението, ще се изпраща до централен сървър. Събраните данни ще се обработват и анализират, за да се идентифицират модели и зони с високо ниво на шум. Проектът също така ще предоставя основателни данни за кампании за повишаване на обществената осведоменост и за вземане на решения при градското планиране.

2. Системен дизайн и архитектура

2.1. Микроконтролер

За целта на проекта бе използвана оригиналната платформа Arduino Uno R3, основана около Atmega328p микроконтролера, чиято функция е да измерва шум, като го разделя на съставните му честоти посредством трансформация на Фурие. Обработените сигнали, точното време на измерване и нивото на захранване се съхраняват на постоянен носител Micro SD карта и се предават безжично през WiFi до мрежови TCP сървър, от който устройството следи и за дистанционно подаване на команди.

2.2. Измерване на звука

За акустични измервания, бе използван кит K002H, предлаган от верига магазини Elimex. Устройството представлява усилвател на звуков сигнал с ниво подходящо за смесителен пулт или крайно стъпало. То е реализирано с транзисторен предусилвател, на входа на който е включен електретен микрофон. Микрофонният модул споделя източника за захранване на микроконтролера. Измерванията се изпълняват посредством 10-битовият аналогово-цифров преобразувател (АЦП), свързан към извод A0 на Uno платформата. Микроконтролерът разчита на 16 MHz кварцов осцилатор за генериране на тактовата честота за процесора, която също се използва и от честотен делител, отговорен за времевата дискретизация на сигналите през АЦП изводите. По подразбиране, честотният делител намалява честотата от основният осцилатор 128 пъти преди да подаде такт към АЦП. Считайки, че архитектурата на микроконтролера отнема приблизително 13 процесорни такта за да обработи сигнала през АЦП, това позволява времева дискретизция от 9600 Hz, която е сравнително ниска за анализ на шумове в човешкия слухов обхват. Посредством манипулация на група регистри ADCSRA, честотният

делител бе преконфигуриран да намалява основните тактове 64 пъти, а резултатната честота на дискретизация (ЧД) се повишава до 19200 Hz. По-високи стойности на дискретизираща честота са възможни чрез настройване на делителя на шестнадесет кратно или еднократно делене, които обаче значително намаляват резолюцията на измерванията. Извличане на честотите от измерванията са изпълнени с помощта на arduinoFFT библиотеката на GitHub потребител kosme.

2.3. Времеброене

Звуковите измервания са съчетани със следене на точно време, което дава възможност да се прави статистика на вида шум и потенциалните му източници в зависимост от гражданската активност в различните периоди от денонощието и годишното време. За целта, бе използван часовник за реално време, основан на интегрална схема DS1307. Връзката с микроконтролера бе установена чрез I^2C комуникационният протокол и библиотеката RTCLib, предоставена от Adafruit. Конкретният модул представлява кит K2018, който съдържа в себе си цифровият часовник и 32K EEPROM памет, задължително захранвани от собствена батерия стандарт CR2032. Програмно бе зададено времето на компилация в локалната часова зона, което при първото стартиране на модула се прехвърля в DS1307 часовника. След това, бордовият осцилатор и собствената батерия на позволяват непрекъснато отмерване на времето, което микроконтролера прочита от модула при поискване от програмата.

2.4. Съхранение на данни

Локалното съхранение на данни като част от заданието изисква свързването на постоянен носител за голям обем от данни. Широкото разпространение, голям капацитет и ниска цена на Micro SD носителите на памет позволява с ниски усилия да бъде реализирано локалното съхранение на

данни в малък форм фактор. С помощта на библиотеката SdFat от Adafruit, са установени директно четене и запис на устройства от всички стандарти SD, за които SPI комуникационният протокол е нативен - изводите на самите паметоносители директно кореспондират на изводи за SPI. Изводът за активиране на модула е свързан с цифров извод 10 на микроконтролера. Важно е да се отчете, че SD устройствата работят на захранване и логическо ниво от 3.3V, докато повечето устройства от Arduino платформата използват 5V за логическо ниво и захранване. За целта, е необходим преобразувател на логическо ниво или модул за SD карта, често срещани на пазара, като този проект разчита на кит K2162 за интерфейс със Micro SD.

2.5. Комуникация

Основен елемент от този проект, е безжичната комуникация, която е осъществена чрез ESP-01S WiFi модул, базиран на ESP8266 микроконтролер. Разчита на серийна комуникация между основния Arduino микроконтролер и ESP-01S модула чрез вградената вградената библиотека SoftwareSerial. Библиотекат позволява връзка с модула чрез двойка дигитални изводи за получаване и предаване на данни. ESP-01S модулът изисква 3.3V захранване, което е осигурено от специализиран извод с бордови регулатор на напрежение на Arduino Uno устройството. Серийният извод Rx бе свързан към дигитален извод 6 на Arduino Uno, а Tx към дигитален извод 7. За да се намалят изискванията върху обема на програмата, се използва вграденият фърмуер на ESP-01S, който позволява подаването на Esp-AT команди за ниско ниво на управление на възможностите на модула чрез серийна комуникация без софтуерна разработка. С помощта на AT команди за конфигурация на WiFi връзка и свързване и подаване на данни към TCP сървър, системата може да изпраща измервания към мрежови хост и да бъде управлявана дистанционно

от него. Конфигурациите на мрежата и сървъра трябва да бъдат зададени в програмата.

2.6. Местоположение

За да се определи позицията на устройството по време на работа дори в движение, бе планиран GPS модул NEO-6M, който не бе осъществен. В замяна на това, бе имплементирана възможността географската ширина и дължина да могат да бъдат задавани чрез дистанционни команди за управление на системата, описани в съответната секция от този документ.

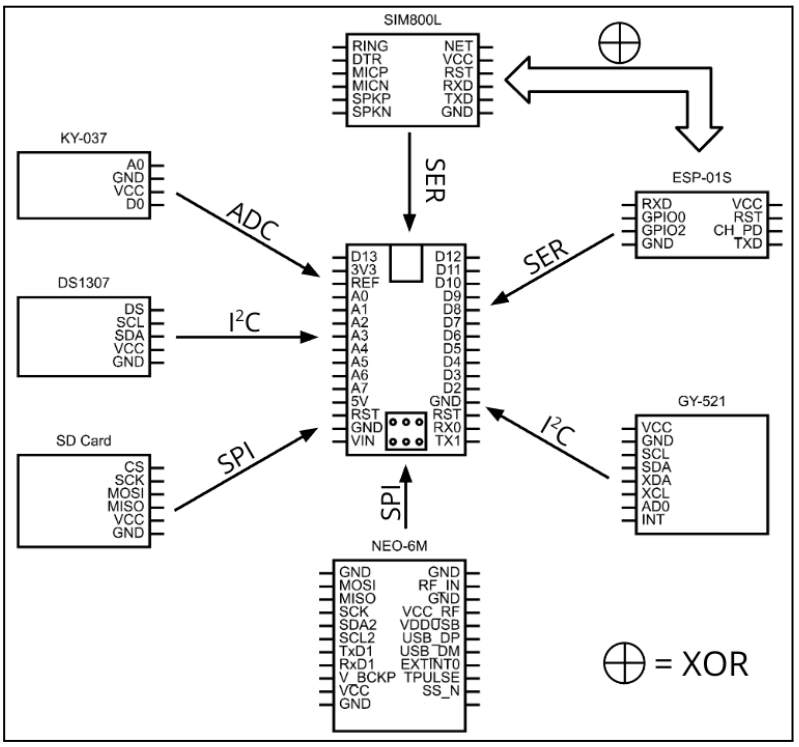
2.7. Захранване

Системата разполага с литиево-полимерна (Li-Po) клетка с оценено напрежение 3.7V и капацитет 550 mAh за самостоятелно захранване. За да бъде постигнато работното ниво от 5V на системата, бе използван кит K561 повишаващ конвертор и регулатор с входен обхват 2.6-4.2V и изходно стабилизирано напрежение 5V до 0.6A. За зареждане на клетката бе използван кит K548 зарядно за литиево-йонни или литиево-полинери клетки с USB-C интерфейс. Външен източник може да бъде използван през зарядния модул, като например соларен панел с регулатор на напрежение за 5V. Нивото на батерия се следи от микроконтролера, посредством АЦП, което следи напрежението на клетката. Поради липсата на време за детайлни опити и на необходими инструменти, животът на системата при работа на самостоятелно захранване не е известно.

2.8. Защитен корпус

За употреба във външни условия, за системата може да бъде проектирана платка, която да осигури устойчива връзка между компонентите, тъй като фактори подобни на вибрации, вятър и физически взаимодействия могат да повлияят на електрическите връзки. За предпочитане е корпусът да има клас за

защита от проникване на вода и прах не по-нисък от IP34 и степен на вандалоустойчивост (извън обхвата на този проект).



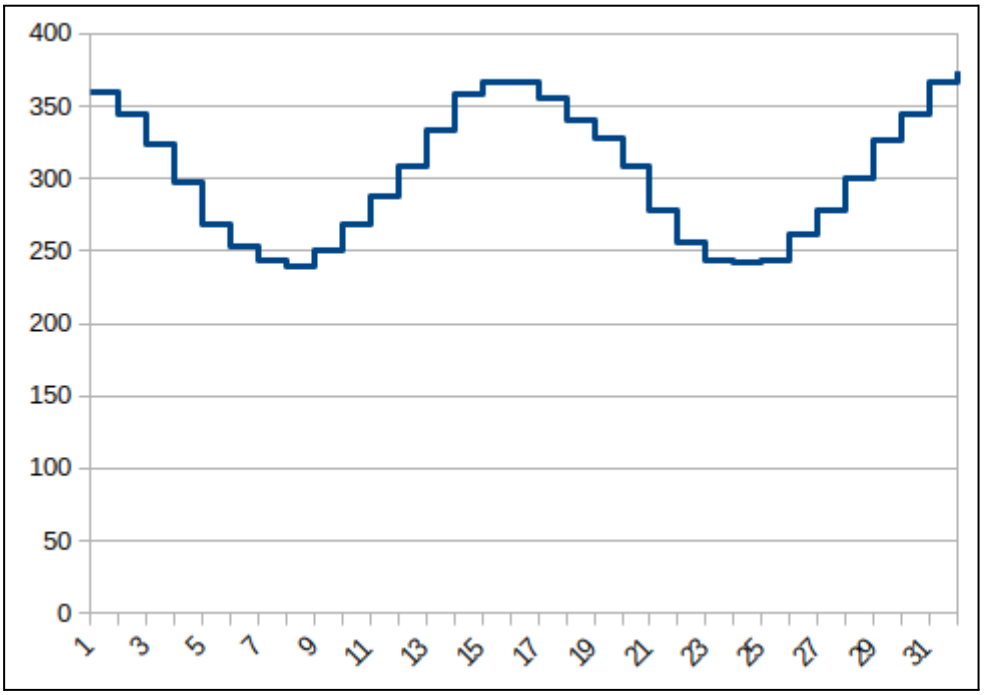
Фиг. 1. Микроконтролер и планирани модули

3. Калибрация на сензорите

3.1. Микрофонен модул и анализ на честота

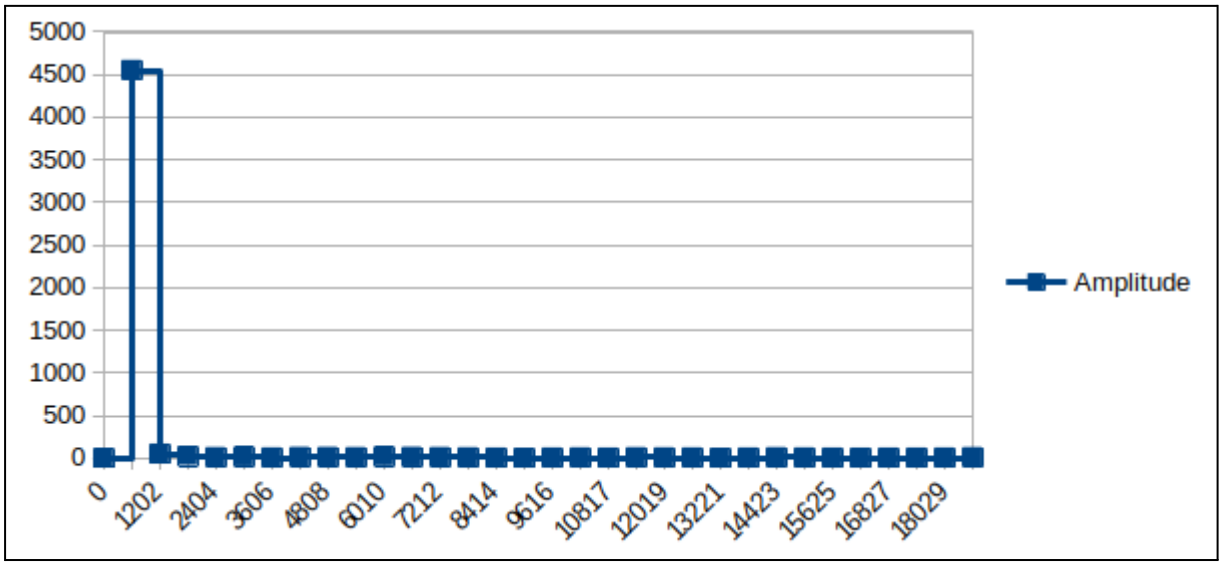
При избраната работна ЧД 19200 Hz, емпирично беше измерен ефикасният обхват на измервания микроконтролерът със съответният микрофонен модул. Поради сравнително малкият капацитет оперативна памет на микроконтролера (2 kB SRAM), изерванията представляваха от 16 до 128 (винаги степен на 2 заради метода за анализ чрез бърза двоична трансформация на Фурие) последователни отчитания на нивото на напрежението [Фиг.2], подадено от микрофонният предусилвател кум АЦП,

които се записват в буферен масив от 32-битови дробни числа с плаваща запетая.



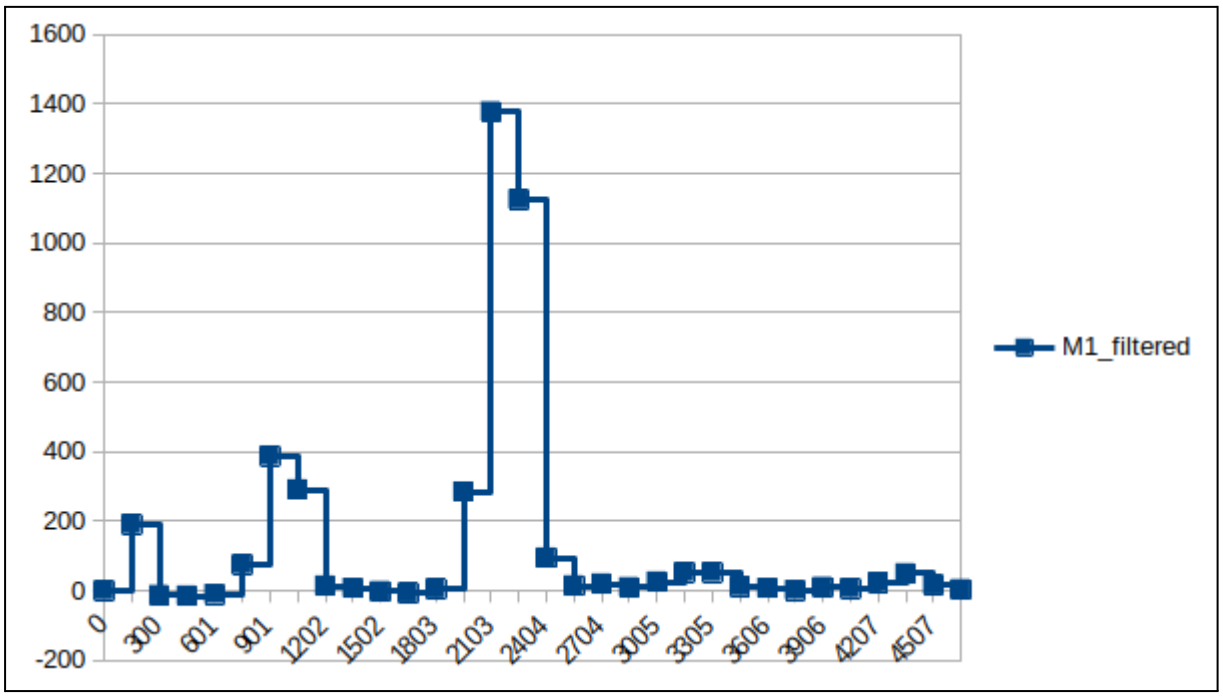
Фиг. 2. Направени 32 измервания на напрежението на микрофонния предусилвател при времева дискретизация 9600 Hz по време на излъчен сигнал 2 kHz с интензитет 60 dB

По време на запис, с помощта на стерео усилвател за компютърно аудио бе генериран синусоидален сигнал с различни честоти. След това, резултатите от анализа на сигнала бяха сравнени. Както в присъствието на звук, така по време на шум не по-висок от 35 dBA, бяха забелязани постоянна нула в първата измерена честота и пик с висок интензитет (зависещ от броя измервания) във втората [Фиг. 3].



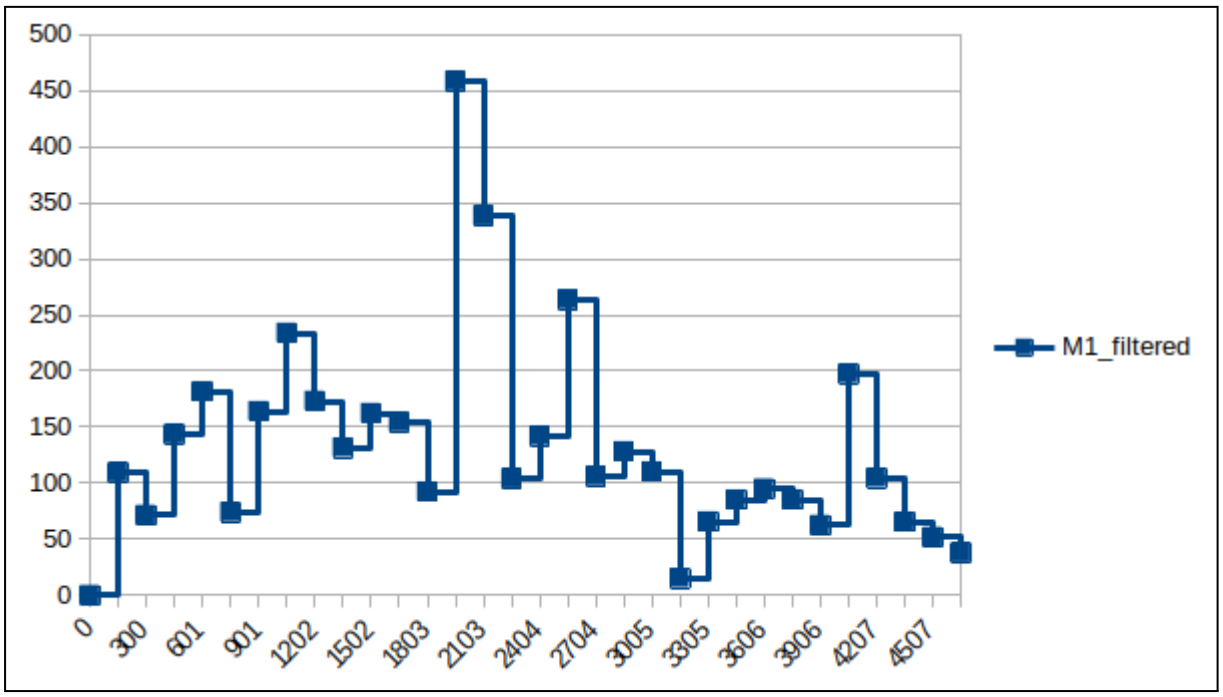
Фиг. 3. Абсолютна нула в първата позиция и фалшив пик във втората позиция присъстват във всички необработени измервания

С цел да се премахне фалшифия пик на втора позиция, както и други статични колебания от равновесното положение на сигнала, за всяка комбинация от ЧД и брой измервания, може да се направи неколкократен анализ на Фурие, след което средноаритметичната стойност на анализите да бъде извадена от по-нататъшните резултати. С този метод, бяха изследвани набор от честоти за да се удостовери надеждността на корекцията [Фиг.4]. Също така, бе забелязано нелинейно разминаване между честотите на измерения и генерирания сигнал, което е постоянно и може да бъде по-нататък коригиран с полиномна функция.



Фиг. 4. Анализ на акустична вълна със съставни честоти 440 Hz, 1000 Hz и 2500 kHz, филтрирани със средноаритметичната стойност на пет измервания по време на ниво на шум под 35 dBA

Чрез опити бе установено, че надеждно използваемите честоти от са между 150 Hz и 6500 Hz, след което се наблюдава силно затихване в сигнала на микрофонния модул.



Фиг. 5. Анализ на бял шум с интензитет 60 dB, филтриран със средноаритметичната стойност на пет измервания по време на ниво на шум под 35 dBA

3.2. Модул за времеброене

С помощта на вградени програмни функции в Arduino платформата, е възможно да се отмерва време в милисекунди или микросекунди в 32-битов формат, но с високо натрупване на грешка, което нараства при нестабилно захранване или натоварване при работа на микроконтролера, което може да доведе до грешка от 30 секунди на денонощие. Вдопълнение, 32-битовият формат се препълва след малко над 49 дни, което само по себе си поставя изисквания на алгоритъма. Пореден проблем със самостоятелното времеброене е загубата на захранване или рестарт на микроконтролера, което води до загуба на информацията или пауза във време броенето. За да бъде избегнато това, се разчита на външен модул за следене на реално време със собствено независимо захранване. За да се калибрира модулът за времеброене,

благодарение на вградени променливи в програмната среда, програмно бе зададено времето на компилация в локалната часова зона. В програмната инициализация на системата се изпълнява проверка ако последният запаметен час е изостанал след компилацията, като само в този случай в модулът се присвоява момента на компилация като настоящо време.

3.3. Модул за съхранение на данни

Поради лимитациите за програмно пространство (32 kB flash) и оперативна памет на микроконтролера, предварителната подготовка на работни файлове и структура на файловата система са избегнати, като устройството очаква на паметоносителя да присъства Fat32 файлова система и текстов файл с име “measurements.csv”. В допълнение, това е прост файл за записване на табличен файл, в който се записват колоните “Amplitudes, Sampling Frequency, Time, Location, Battery Voltage”, които могат да бъдат изрично изписани на първи ред във файла. Използва се Comma Separated Values (CSV) формата за отделяне на табличните колони, като в тях съответно се записват линейните амплитуди на прихванатите честоти, разделени с разстояие, честотата на дискретизация, момента на измерване в UNIX формат (броят секунди, изминали от 1ви чнуари 1970 г.), географската ширина и дължина, разделени с разстояния и напрежението на захранващата батерия, описана по-нататък. За поддръжка на SD карти с нисък стандарт за скорост на четене и запис, изрично бе зададена по-ниска честота на SPI комуникацията, между 1 MHz и 4 MHz, тъй като за това устройства не е необходима висока скорост на данните.

3.4. Модул за безжична комуникация

По подразбиране модулът очаква честота на сигнали за пренос на данни от 115200 Hz, в резултат на която се среща голяма вероятност за грешка в

предадените символи по серийният протокол чрез неспециализирани изводи като дигитални изводи 0 и 1 в Arduino платформата, които са заети за качване на програмата и серийна комуникация с компютър по време на изпълнение. За да се избегне това, честотата на комуникация бе намалена до 9600 Hz чрез AT команда UART_DEF.

3.5. Заряд на батерията

Нивото на батерията може да се следи чрез дистанционен достъп до устройството и също се записва на SD картата. Напрежението на батерията се наблюдава чрез АЦП на микроконтролера. Поради проблеми с работното напрежение на чувствителния кварцов осцилатор на модула за времеброене, се наблюдава затруднение на източници с напрежение, надвишаващо 4 V, като зареденото ниво на литиево-йонни и полимерни клетки често достига 4.2 V. Това доведе до нуждата от използването на обикновен резисторен делител на напрежение за свързване с АЦП. С помощта на волтметър бе установено разминаването между измереното и реалното напрежение, което може да бъде програмно въведено като потребителски параметър. В бъдеще може да се използват и възможностите на Arduino платформата за стабилизиране на аналоговите измервания с вътрешния регулатор на напрежение или чрез свързването на стабилизатора на захранването с извод AREF.

4. Софтуер

В началото на основния файл main.cpp е разположена секция за потребителските параметри, която предоставя възможност на незапознат ползвател да настрои системата по своя преценка с предварително поддържани променливи [Фиг. 6].

```
// User configuration
#define SERIAL_BAUD 115200
#define BATTERY_CALIBRATION_OFFSET -0.2
#define MEASUREMENTS_FILE "measurements.csv"
#define NUMBER_OF_SAMPLES 16
#define PRESCALE_MULTIPLIER 64
#define MEASUREMENT_DELAY 30
#define WIFI_CMD "AT+CWJAP=\"SSID\",\"PWD\""
#define SERVER_CMD "AT+CIPSTART=\"TCP\",\"IP\",PORT"
#define ENABLE_COMMAND_LOG_AT_SETUP false
```

Фиг. 6. Секция за потребителските параметри

Параметрите позволяват да се промени скоростта на серийния протокол за комуникация SERIAL_BAUD, грешката в измереното напрежение на батерията BATTERY_CALIBRATION_OFFSET, името на файла в SD картата, където се записва следената информация MEASUREMENTS_FILE. Важен избираем параметър е и броя измервания NUMBER_OF_SAMPLES, тъй като от него силно зависи качеството на анализа на шума. Този параметър трябва да бъде със стойност степен на двойката поради спецификата на бързата трансформация на Фурие, която се използва за анализ на честотите. Резултатните честотни ленти винаги представляват дробни стойности два пъти по малко на брой от измерванията. Страничен ефект е голямата употреба на оперативна памет, като за всяко измерване са нужни 64 бита. Възможно е да се задава и степента на честотия делител PRESCALE_MULTIPLIER, от който зависи честотата на дискретизация, като поддържаните стойности са съответно 128 с ЧД 9600 Hz, 64 с ЧД 19200 Hz, 16 с ЧД 76800 Hz и 1 с ЧД 125 kHz. Също по преценка на потребителя се задава периода от време между автоматичните измервания на звука MEASUREMENT_DELAY в секунди, като стойност от 0 индикра на програмата да не се извършват измервания автоматично. Това

действие може да бъде изпълнено и с дистанционна команда чрез сървъра. За връзка със сървъра, трябва да бъдат правилно конфигурирани командите за безжичният модул WIFI_CMD и SERVER_CMD като SSID се заменя с името на безжична точка за достъп с мрежова свързаност, PWD е паролата за достъп на точката, а IP и PORT трябва да бъдат заменени с адреса и порта, на които отворен TCP сокет следи за връзка с устройствата. Включен е и параметър ENABLE_COMMAND_LOG_AT_SETUP, който определя дали е разрешено на устройството да извежда съобщения от периферните модули по серойна връзка по време на инициализация. Това е по подразбиране забранено с цел повишаване на сигурността, тъй като при комуникация между микроконтролера и безжичния модул явно се предават съобщения за конфигурация на мрежата и сървъра.

Следващата секция от кода позволява при смяна на микроконтролера или според изисквания на потребителя лесно да се промени конфигурацията на свързаност между модулите [Фиг. 7].

```
// Wiring
#define MIC_ANALOG_OUT_PIN A0
#define BATTERY_MONITOR_PIN A5
#define SD_CS_PIN 10
#define ESP01_WIFI_RX_PIN 6
#define ESP01_WIFI_TX_PIN 7
```

Фиг. 7. Секция за свързване на модулите

Показаните редове описват свързаните изводи на периферните устройства: Аналоговият извод на микрофонният предусилвател и батерията трябва да бъдат свързани с аналогови изводи на микроконтролера, означени с MIC_ANALOG_OUT_PIN и BATTERY_MONITOR_PIN, изводът за активиране на SPI устройство трябва да отговаря на зададеният извод

SD_CS_PIN, а модулът за безжична комуникация свързва Rx и Tx едноименно с изводите ESP01_WIFI_RX_PIN и ESP01_WIFI_TX_PIN. Всички останали връзки са според стандартните изводи за съответния комуникационен протокол в Arduino платформата.

Стандартно в Arduino платформата, еднократно се изпълнява функцията setup(), която инициализира периферните модули и конфигурацията на микроконтролера, след което се повтаря функцията loop(), съдържаща инструкциите за дистанционно командване и за автоматично измерване на звука. По време на изпълнение на програмата са използвани и изредените функции [Фиг.8-12].

```
// Initialization functions
bool initSerial();
void initPins(uint8_t prescaleMultiplier);
bool initWlSer();
bool initRTC();
bool initSD();
```

Фиг. 8. Програмни функции за инициализация

Първият набор от функции за инициализация съдържат нужните инструкции за еднократно повикване, с които се конфигурират периферните модули. Повечето от тях връщат булева стойност, която отговаря на успешното зареждане на съответния компонент.

```
// FFT functions
void readSamples();
void computeFFT();
void printFFTResult();
```

Фиг. 9. Програмни за трансформация на Фурие

Следващата секция отговаря на функциите за анализ на звук. Чрез readSamples() се записват последователнии измервания в дробният масив vReal. При повикване на computeFFT(), тези измервания се преобразуват в честоти и се копират в същия масив. С помощта на printFFTResult резултата от честотния анализ може да бъде изведен през серийната комуникация.

```
// Serial functions
void readHwSerial();
void readSwSerial();
void processCommandFromStr(String str);
bool wlCommand(const __FlashStringHelper *cmd, String resp = "OK");
bool sendToServer(String str = "OK", bool newLine = true);
```

Фиг. 10. Програмни функции за серийно и безжично предаване на данни

Тук са представени функциите за управление на серийна и дистанционна комуникация. Функцията readHwSerial() е отговорна за обработване на команди от потребителя, подадени през серийната комуникация с компютър. Подбно на нея, readSwSerial() има за цел да приема и обработва съобщенията, получени от безжичният модул. Всяка от двете разполага със самостоятелен буфер за запис на символите, които преминават през серийните протоколи. И двете функции са насочени към общия набор от инструкции за обработка на команди processCommandFromStr(), който приема низ и определя дали той съответства на предварително дефинирана команда и изпълнява съответните действия, като дефинираните команди са описани в експлоатационно ръководство на този документ. Следва функцията wlCommand(), която подава ESP-AT команди към безжичния модул и следи дали неговия отговор съвпада с изискваната ключова дума (стандартно изпълнението на AT командите завършват с OK за успех и ERROR или FAILED за неуспех, или busy ако

устройството не е завършило процес на предишна коанда). Последната функция от тази секция е sendToServer(), чиято цел е дистанционно да предава текст към сървъра, накрая завършващ или не с терминатор за нов ред.

```
// SD card functions
void writeToSD();
void seekSDLines(int lines);
void readFromSD(int lines,bool seekOnly=false);
```

Фиг. 11. Програмни функции за управление на SD картата

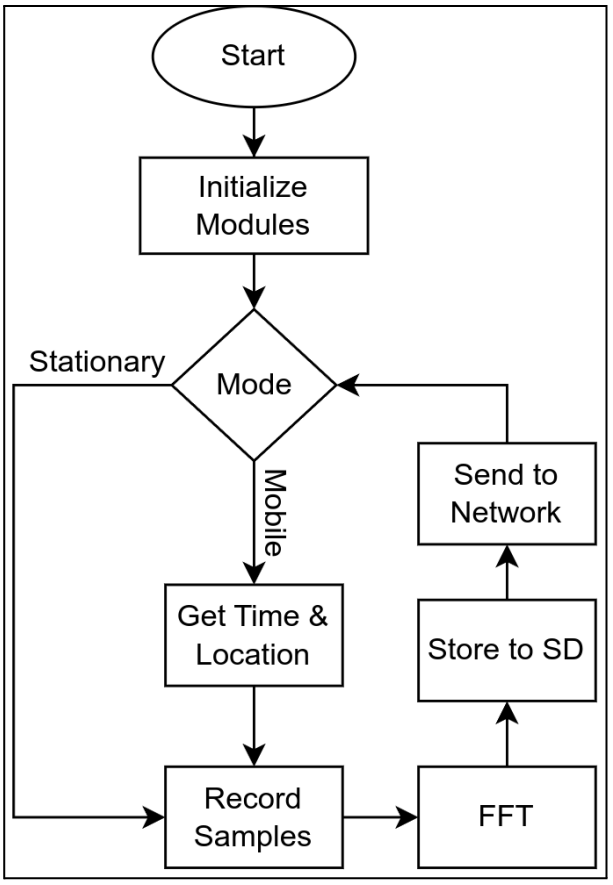
Този набор от инструкции обработва данните на SD картата, като writeToSD() автоматично записва масива от честоти от последната трансформация на Фурие, заедно с използваната честота на дискретизация, настоящия час, местоположението и напрежението на захранващата батерия. Функцията seekSDLines() се използва за да се премести четеца на данни до определен ред от табличния файл за запис. Отрицателни стойности поставят показалеца след последния записан ред, където се очаква записа на актуални данни. Изчитането на записаните данни е възможно чрез readFromSD(), на която се подават брой редове за изчитане и булева променлива, която по подразбиране е изключена. При активиране, броят редове няма да бъде изчетен, а прескочен като само се измества относително показалеца. Изчетените данни се предават до сървъра.

```
// Miscellaneous functions
float vbatt();
void geoloc(String str, float &coord);
void printTime();
```

Фиг. 12. Програмни функции за управление и справка на променливи

Последният набор от функции отговаря за справка на състоянието на системата, като vbatt() подава напрежението на батерията в дробен формат, printTime() извежда времето в UNIX формат, а geoloc() се използва за задаването на географска ширина и дължина чрез низова команда през сървъра или кабелната серийна комуникация с микроконтролера. В препратката се подава съответната променлива latitude или longitude, която трябва да бъде променена.

Програмният код, използван за изграждане на проекта е достъпен в платформата GitHub на следният адрес:
https://github.com/antonAtananasov/55_Urban_Noise_Monitoring



Фиг. 13. Схема на програмата

5. Инструкции за експлоатация

За да стартира успешно, при инициализация системата трябва да се свърже с точката за достъп и сървъра според потребителски зададените параметри. Сървърът представлява отворен TCP сокет на зададен порт върху мрежови хост. За да се стартира такъв може да бъде използван например netcat с командата “nc -lk <PORT>”, в защитната стена на хоста трябва да бъде пропуснат съответния порт. След като потребителят може да потвърди, че сървърът е достъпен, системата може да бъде включена. При успешна инициализация, в сървърът се появява съобщение “[READY]”. След това съобщение, устройството може да бъде управлявано с набор от предварително дефинирани команди дистанционно или чрез кабелна серийна връзка:

- marco - използва се за да се тества успешна връзка със сървъра и обработка на команди. Като отговор на командата, в сървъра трябва да се покаже съобщение “polo”
- tt - изпраща настоящото време в UNIX формат към сървъра
- bt - изпраща настоящото напрежение на батерията към сървъра
- lat<LATITUDE> - задава географската ширина, като <LATITUDE> е положително число в градуси с до 4 цифри след десетичната запетая
- lon<LONGITUDE> - задава географската дължина, като <LONGITUDE> е положително число в градуси с до 4 цифри след десетичната запетая
- ff - изпълнява запис на последователни измервания на звука, изпраща стойностите им към сървъра, след което изпълнява трансформация на Фурие и изпраща нейните резултати към сървъра, след което записва получените данни в SD картата
- rd<N> - изчита редове от файла на SD картата и ги изпраща към сървъра, като <N> е броят редове за отпечатване

- sd<N> - премества показалеца за четене на файла на SD картата на определен ред, като <N> е номерът на реда. При подаване на отрицателна стойност, показалецът се поставя на края на файла
- > - символът индикира текста след него да бъде изпратен към сървъра
- < - символът индикира текста след него да бъде подаден към безжичния модул

При липса на съобщение за готовност в период на повече от 10 секунди след стартиране на системата, е възможно да има грешка в иициализацията на някой от модулите или проблем с физическата връзка помежду им. В такъв случай кабелната връзка може да бъде използвана като средство за диагностика. Тя приема същите команди, но и извежда допълнителна информация. При грешка в инициализацията може да бъде забелязано съобщение “RTC err”, “SD err”, “WL err”, индикирайки грешка с посочения модул, съответно модула за време броене, модула за SD картата, или модула за безжична комуникация. При подаване на команда с начален символ “<”, може да се предават иструкции директно към ESP-01S модула чрез Esp-AT стандарта.

6. Заключение

Този проект не развива пълния потенциал на системата главно поради недостиг на ресурсите на избрания микроконтролер, но може да демонстрира как подобен продукт може да бъде използван от кампании за повишаване на качеството на живота на жителите в различни райони.

В градската среда присъстват множество източници на шум, които допринасят към натоварването на нервната система, повишават стреса и умората. Също така, високото шумозамърсяване представлява индикатор за

проблеми с трафика, аварии или лошо планиране или неефикасна архитектура на засегнатия район. Чрез анализ на честотите, които съставят шума, може да се направи извод за техния източник и статистика за неговото разпространение във времето.

Системите за планиране на трафика и гражданска навигация могат да използват шума като показател за натовареността на различните маршрути в различни периоди от денонощието.

Хората, които планират закупуването на недвижим имот могат да се информират за качеството на звуковото замърсяване в града и дали е породено от работна, човешка или автомобилна дейност.

Озвучителни тела и звукови сигнали като напр. средства за незрящи могат да бъдат съобразени с нивото на околния шум така, че да бъдат ясно отчетливи над фона без да предизвикват излишно дразнение от прекомерно висока сила.

Може да бъде създадена уеб платформа за гражданска информираност, която представлява оцветена карта със статистика за нивото на шума на клчючови зони от града.

Архитектите и гражданските инженери могат да оптимизират организацията и структурата на жилищните комплекси като вземат предвид разпространението на акустични вълни за да намалят тяхното достигане до жилищните площи.

Системата може да бъде разърната с помощта на микроконтролер с повече ресурси, което да позволи по-голям брой измервания при анализ на честотите, АЦП с по-висока резолюция и повече програмно пространство, с което да бъдат имплементирани разширени възможности.

7. Литература

- Stoter, J., de Kluijver, H., & Kurakula, V. (2008). 3D noise mapping in urban areas. International Journal of Geographical Information Science, 22(8), 907–924. https://doi.org/10.1080/13658810701739039
- Erlend. (2019, June 4). Acoustic quantities, part 4: Quantities in noise regulations. Erlend M. Viggen. https://erlend-viggen.no/acoustic-quantities-4/
- Google Earth: Browser-based animation tool for Google Earth’s 3D and satellite imagery. (n.d.). Retrieved April 17, 2024, from https://earth.google.com
- 32-Band Audio Spectrum Visualizer Analyzer. (n.d.). projecthub.arduino.cc. https://projecthub.arduino.cc/shajeeb/32-band-audio-spectrum-visualizer-analyzer-924af
- Libraries - Arduino reference. (n.d.). https://www.arduino.cc/reference/en/libraries
- Elimex, online store for electronics, batteries and accumulators, meters, LED bulbs, sound.(2023, November 20). https://elimex.bg/
- emag.bg - Wide range of products online store. (n.d.). eMAG.bg. https://www.emag.bg/
- alldatasheet.com. (n.d.). KY-037 Datasheet(PDF). JOY-IT. https://www.alldatasheet.com/datasheet-pdf/pdf/1284505/JOY-IT/KY-037.html
- Arduino - DS1307 RTC Module | Arduino getting Started. (n.d.). Arduino Getting Started. https://arduinogetstarted.com/tutorials/arduino-ds1307-rtc-module
- alldatasheet.com. (n.d.-b). NEO-6M PDF. https://pdf1.alldatasheet.com/datasheet-pdf/view/1283987/U-BLOX/NEO-6M.html
- LME Editorial Staff. (2022, June 26). Interface ublox NEO-6M GPS Module with Arduino. Last Minute Engineers. https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/
- LME Editorial Staff. (2023, November 30). Send Receive SMS & Call with SIM800L GSM Module & Arduino. Last Minute Engineers. https://lastminuteengineers.com/sim800l-gsm-module-arduino-tutorial/
- LME Editorial Staff. (2022b, October 10). Interface DS1307 RTC Module with Arduino. Last Minute Engineers. https://lastminuteengineers.com/ds1307-rtc-arduino-tutorial/
- ESP8266 Wi-Fi module. (n.d.). https://www.taydaelectronics.com/datasheets/files/ESP-01S.pdf
- AT Command Set - ESP32 - — ESP-AT User Guide latest documentation. (n.d.). https://docs.espressif.com/projects/esp-at/en/latest/esp32/AT_Command_Set/index.html
- Instructables. (2019, July 17). Arduino timing methods with Millis(). Instructables. https://www.instructables.com/Arduino-Timing-Methods-With-Millis/
- Instructables. (2017, September 30). Getting started with the ESP8266 ESP-01. Instructables. https://www.instructables.com/Getting-Started-With-the-ESP8266-ESP-01/
- Instructables. (2017a, September 25). The simple guide to flashing your ESP8266 firmware. Instructables. https://www.instructables.com/The-Simple-Guide-to-Flashing-Your-ESP8266-Firmware/

- Instructables. (2020, January 8). Restore or upgrade firmware on ESP8266 (ESP-01) module using Arduino UNO. Instructables. https://www.instructables.com/Flash-or-Upgrade-Firmware-on-ESP8266-ESP-01-Module/
- Instructables. (2019b, July 31). Getting started with ESP 8266 ESP-01 with Arduino IDE | installing ESP boards in Arduino IDE and programming ESP. Instructables. https://www.instructables.com/Getting-Started-With-Esp-8266-Esp-01-With-Arduino-/
- Instructables. (2019c, December 13). Totoro Project - IoT & MQTT & ESP01. Instructables. https://www.instructables.com/Totoro-Project-IoT-MQTT-ESP01/
- Instructables. (2017c, September 30). Using ESP-01 and arduino UNO. Instructables. https://www.instructables.com/Using-ESP-01-and-Arduino-UNO/
- Instructables. (2021, February 11). EasyFFT: Fast fourier transform (FFT) for Arduino. Instructables. https://www.instructables.com/EasyFFT-Frequency-Transform-for-Arduino/
- RobTillaart. (n.d.). GitHub - RobTillaart/float16: Arduino library to implement float16 data type. GitHub. https://github.com/RobTillaart/float16
- Edragon. (n.d.). esp_firmware/Firmware at master · Edragon/esp_firmware. GitHub. https://github.com/Edragon/esp_firmware/tree/master/Firmware
- espressif. (n.d.). GitHub - espressif/esp-at: AT application for ESP32/ESP32-C2/ESP32-C3/ESP32-C6/ESP8266. GitHub. https://github.com/espressif/esp-at
- Binaryupdates. (n.d.). esp01-firmware/esp8266_flasher.exe at main · binaryupdates/esp01-firmware. GitHub. https://github.com/binaryupdates/esp01-firmware/blob/main/esp8266_flasher.exe
- Ekstrand. (n.d.). GitHub - ekstrand/ESP8266wifi: ESP8266 Arduino library with built in reconnect functionality. GitHub. https://github.com/ekstrand/ESP8266wifi
- ESP8266 ESP-01 relay control using Web server or MQTT. (n.d.). Gist. https://gist.github.com/jeroavf/07f48ebd6994a4b913d0
- Adafruit. (n.d.). GitHub - adafruit/RTClib: A fork of Jeelab’s fantastic RTC Arduino library. GitHub. https://github.com/adafruit/RTClib
- Mmoller2k. (n.d.). Float64/Float64.h at master · mmoller2k/Float64. GitHub. https://github.com/mmoller2k/Float64/blob/master/Float64.h
- Kosme. (n.d.). GitHub - kosme/arduinoFFT: Fast Fourier Transform for Arduino. GitHub. https://github.com/kosme/arduinoFFT
- A 2 byte float? (2023, September 20). Arduino Forum. https://forum.arduino.cc/t/a-2-byte-float/1170014/50
- Playing with PROGMEM and __FlashStringHelper. (2017, December 10). Arduino Forum. https://forum.arduino.cc/t/playing-with-progmem-and-__flashstringhelper/496026/6
- ESP8266 platform index for Arduino IDE, https://arduino.esp8266.com/stable/package_esp8266com_index.json

- 16 bit floats? (2015, June 4). Arduino Forum. https://forum.arduino.cc/t/16-bit-floats/315961/8
- Half precision Floating point numbers. (2010, October 10). Arduino Forum. https://forum.arduino.cc/t/half-precision-floating-point-numbers/5927/3
- Analog to digital converter - clock prescaler tests, obsevations and questions. (2015, April 25). Arduino Forum. https://forum.arduino.cc/t/analog-to-digital-converter-clock-prescaler-tests-obsevations-and-questions/307353
- Super High Performance PIN I/O technique. (2014, April 18). Arduino Forum. https://forum.arduino.cc/t/super-high-performance-pin-i-o-technique/224363/19
- Faster analog read? (2008, April 21). Arduino Forum. https://forum.arduino.cc/t/faster-analog-read/6604/6