Курсова работа по „Вградени микрокомпютърни системи“

Име на курсовата работа:
Подслушвателно устройство

Изготвящи курсовата работа:
1. Виктор Димитров Димитров, XI Б клас, № 5 (*изготвил хардуерната част от проекта и участвал при закупуването на частите му*)
2. Теодор Славчев Дишански, XI Б клас, № 23 (*изготвил софтуерната част от проекта и участвал при закупуването на частите му*)

1. Произход на идеята за темата на курсовата работа

В началото на учебната година ни беше казано, че за завършване по учебния предмет „Вградени микрокомпютърни системи“ ще бъде необходима курсова работа и то на такава тема, по която ни се е искало винаги да работим и с която би ни било интересно да се занимаваме като идея и концепция. Ние решихме да направим подслушвателно устройство. Идеята ни дойде простичко и естествено – и двамата си падаме по книги и филми на криминална и научно-фантастична тематика. Щом имахме възможността да направим нещо, което да работи коректно и от което накрая да сме доволни, че сме положили усилия върху него, то ние нямахме съмнения, че идеята ни е точната. След като се консултирахме с преподавателя ни по предмета господин Росен Витанов относно това дали ще можем да реализираме подобно нещо като проект и след като получихме одобрение за идеята ни, ние вече имахме готовността да започнем да работим по въпросната курсова работа.

2. Какво има вече произведено на пазара и как стои въпросът там с продукти като този от нашата идея?

Подслушвателните устройства в Република България от гледна точка на регламентите на ДАТО (Държавна агенция „Технически операции“) и законите на Националното бюро за контрол на специалните разузнавателни средства (СРС) могат да бъдат вкарани в експлоатация, било то пряка или косвена, единствено чрез съдебна заповед или специално служебно разрешение и не от кого да е, а само от хора, които са на служба към Министерство на вътрешните работи (МВР), Държавна агенция „Разузнаване“ (ДАР), Държавна агенция „Национална сигурност“ (ДАНС), Главна дирекция за борба с организираната престъпност (ГДБОП), Служба „Военно разузнаване“ (СВР), както и от Комисията за противодействие на корупцията и за отнемане на незаконно придобитото имущество (КПКОНПИ).

На световно ниво подслушвателните устройства влизат в законна експлоатация единствено когато се използват от специалните служби за сигурност и отбрана, разузнавателните управления, службите, занимаващи се с контратероризъм и при специални оперативни дейности на конкретни лица по време на конкретни техни мисии, регламентирани в длъжностната им характеристика. Анологично и в останалите страни по света подслушвателните устройства се използват единствено чрез документално нареждане или заповед.

Наличието на тези клаузи значително възпрепятства достъпа до информация от тип на това, какви са например характеристиките, спрямо които се избира един от няколко вида подслушвателни устройства, но това, до което може да се достигне, е, че най-често едно подслушвателно устройство бива вкарано в употреба в зависимост от специфичната ситуация и в зависимост от това какъв точно е исканият резултат от въпросното специално резузнавателно средство. Именно това определя и изборът на дадено устройство, като главна роля играят неговите специфични характеристики и парадигма на действие. Въпреки трудностите по сдобиване с информация за видовете подслушвателни устройства, които се предлагат на пазара и които се предлагат за точно спецефични цели, все пак могат да се диференциират няколко основни типа.

1. Подслушвателни устройства, съдържащи SIM карта: Това са подслушвателни устройства, които съдържат възможност за запис. Жаргонното им наименование в средите, в които се използват, е „бръмбари“. Тези устройства са може би най-разпространените

подслушвателни устройства от всички типове, тъй като могат да предават акустична информация на предварително зададен номер, използвайки GSM мрежата. Те могат да бъдат контролирани дистанционно, чрез специфични SMS команди или най-просто предварително могат да бъдат програмирани от компютър. Захранването обикновено е автономно (литиеви батерии), но могат и да бъдат използвани и ако са свързани към електрическата инсталация в помещението, в което ще се използват (Фигура 1).

Тъждество на едно такова устройство би могъл да бъде и съвременен мобилен телефон или смартфон, ако на него има незаконно инсталиран софтуер за подслушване. Контролът отново е чрез SMS съобщения, а захранването идва от батерията на самия телефон. Жаргонното име на телефони, специално използвани за тази цел, е „тракери“ (това понятие е различно от понятието GPS тракери, които нямат нищо общо с подслушването, тъй като GPS тракерите обикновено нямат вграден микрофон в себе си) (Фигура 2).

Фигура 1. Подслушвателно устройство с възможност за запис и подслушвателно устройство с възможност за гласов контрол

Фигура 2. Смартфон с инсталиран софтуер за подслушване и изпращане на запис до дадена мрежа посредством 3G

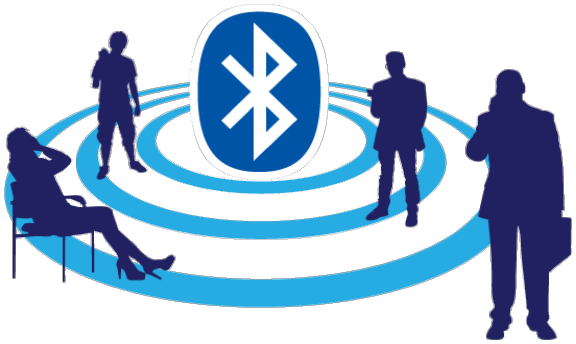
2. Подслушвателни устройства с главна специфика тяхната възможност за запис: Това са класическите подлушвателни устройства, претендиращи най-вече с превъзходната си опция за запис, чието съдържание е изчистено от ненужни шумове и съдържа информация, която е прихваната в конкретен диапазон. Фактът, по който този тип подслушвателни устройства се различават от първия тип, е незадължителното условие самото устройство да съдържа SIM карта (Фигура 3).

Подслушвателните устройства, използващи Bluetooth за трансфер на записа, са типичен представител за този тип подслушватели. Тези устройства могат да бъдат изключително малки, като фактор за тяхната големина в повечето случаи е вида на захранването – дали разполагат с батерия или са свързани директно към електрическата мрежа. Ако едно подобно устройство не използва батерия, то може да има размери 1 x 1 x 0.5 cm. Именно заради своите миниатюрни размери, в комбинация с това, че работят във високочестотен обхват, не предават постоянно сигнал и имат много малко потребление на електроенергия, те са изключително трудни за откриване. Друго важно тяхно предимство е, че правят запис върху собствена памет, а самият запис може да бъде предаден в определен

часови диапазон, което автоматично означава, че по времето на създаване на записа те няма да излъчват никакъв радио сигнал. Телефоните, използващи Bluetooth също са способни да работят като подслушвателните устройства от този тип (Фигура 4).

Фигура 3. Класическо подслушвателно устройства с възможност за запис (на снимката – GPS тракер, съдържащ микрофон)

Фигура 4. Благодарение на големия обхват на Bluetooth, използващите го подслушвателни устройства са способни на много

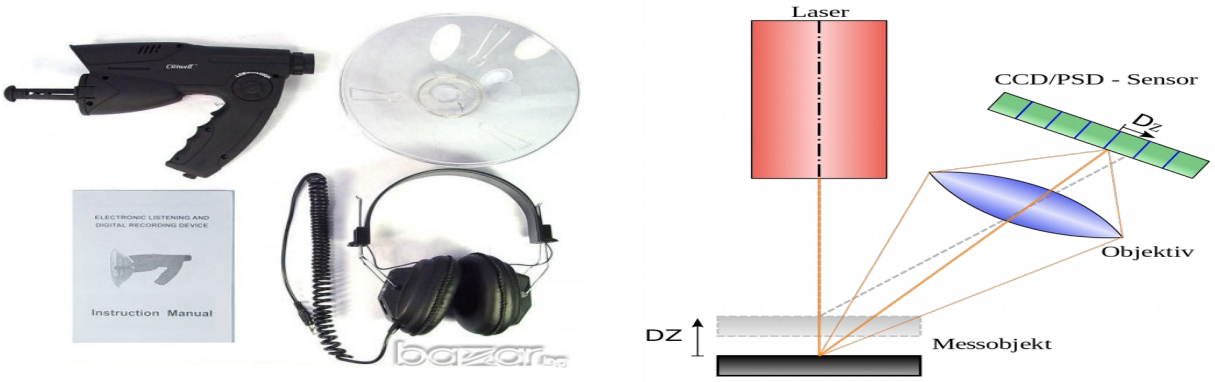


3. Подслушвателни устройства с главна специфика тяхната възможност за видеонаблюдение: Това са устройства, които, макар и да спадат към групата на подслушвателните, се отнасят по-скоро в групата на устройствата, свързани с видеонаблюдението, тъй като това, в което са специализирани, е отличната им възможност за видеонаблюдение на пространството и последващ запис от това видеонаблюдение. Независимо обаче дали правят аудио запис или видеонаблюдение, въпросният запис се запазва на microSD карта, като аудио записът най-често се записва в .amr формат, а аудио-видео файловете – в .avi формат. Стандартният размер на файловете при най-професионалните устройства е от 700 MB.

4. Подслушвателни устройства с AGPS – По същество това са устройства, съдържащи AGPS (Assisted GPS или Augmented GPS). Тяхното предимство е, че към себе си имат вграден GPS, което допълнително спомага за позиционирането на самото устройство, както и показване на местонахождението на даден обект. Някои, подобно на първия тип подслушвателни устройства, работят с микро SIM карта с цел

потреблението на мобилен Интернет, през който се изпращат данните към Интернет базирана платформа. Други стигат дотам, че притежават и два отделни датчика, служещи за уведомление при задействане на устройството, като първият датчик е вибрационен (когато устройство се разклати), а вторият е звуков (при преминаване на точно определена граница на шум – обикновено се произвеждат с граница от 60 dB, съществуват и устройства, чиято граница за превишаване на шума се движи в диапазона от +20 dB до -20 dB).

5. Устройства с лазерно подслушване – Тези подслушвателни устройства са снабдени с лазерен микрофон, който използва лазерен лъч за откриване на звукови вибрации в отдалечен обект. Въпросните лазерни устройства са и най-надеждните, тъй като чрез тях може да се подслушва с минимален шанс за разкритие на самото устройство.



Фигура 5. Устройство с лазерно подслушване и схема на принципния му начин на действие

На международния пазар ситуацията с видовете подслушвателни устройства е аналогична. Почти всяко едно устройство в днешно време има SIM карта в себе си. Важно е да се отбележи, че тази SIM карта не служи чак толкова за запазване на даден вид запис, а по-скоро е потребна за комуникация с потребителя. Комуникацията се осъществява посредством SMS съобщения, които се изпращат от потребителя до SIM картата на подслушвателното устройство или от подслушвателното устройство до потребителя, когато последният се налага да бъде уведомен за конкретно състояние на самото устройство. Тук е редно да се спомене, че няма сигурност при експлоатацията на кое да е подслушвателно устройство, тъй като при тяхното поставяне на конкретна позиция се залага на това самото устройство да не бъде разкрито като физически съществуващ обект. Към днешна дата най-сигурни от гледна точка на разкритие са именно лазерните устройства, тъй като лазерът е много по-

трудно прихващаем, което прави и самото подслушвателно устройство почти незабележимо.

Разбира се, не е изключена и уникалността при производството на подслушвателни устройства. Исторически погледнато съществуват много случаи на подслушвателни устройства, чиято цел е била да се използват за изключително специфична ситуация, като по-познати разработчици на такива технически решения са били Централното разузнавателно управление (ЦРУ) в Америка, Главното разузнавателно управление (ГРУ) в Русия, както и Института за разследване и спациални задачи (Мосад) в Израел и други.

3. Реализация на курсовата работа

3.1. Структурни компоненти в блок-схемата

В подслушвателното устройство са използвани микрофон, усилвател, Bluetooth модул, Arduino микроконтролер и други спомагателни части. Микрофонът е кондензаторен, а видът на Bluetooth модула е HC05. За захранване на веригата са използвани две батерии с напрежение от 9 V. Операционният усилвател е деференциален.

3.2. Arduino Uno

В подслушвателното устройство е използвана платката за микроконтролер Arduino Uno.

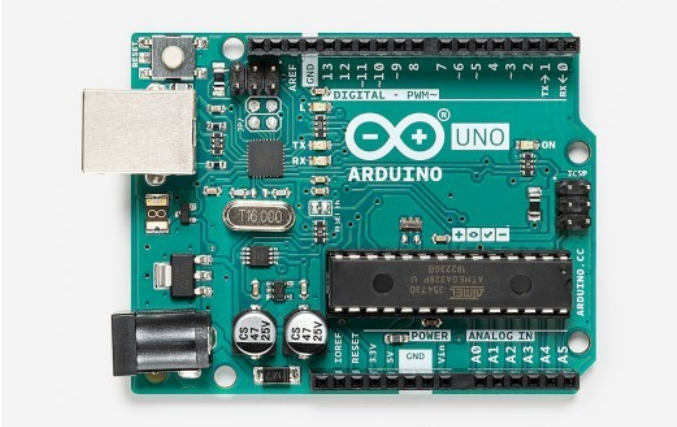
Arduino е платформа с отворен код, с голяма обществена подкрепа и обширен набор от библиотеки и примерни проекти. За създаване на приложения и програмиране на платките Arduino се използва свободен софтуер ArduinoIDE (IDE – Integrated Development Environment, в превод - интегрирана среда за разработка), който е изключително удобен и лесен за употреба (макар че за набиране на кода могат да бъдат използвани и други среди за разработка, които имат поддръжка на същата концепция, като например Platform IO, който може да бъде изтеглен и инсталиран посредством Visaul Stduio Code или Visual Studio Insiders – софтуерни продукти на Microsoft). Именно в средата за разработка се набира и кодът, който ще управлява Arduino Uno. В официалния сайт има подробна информация за платформата и множество примерни проекти за взаимодействието му с периферни устройства – сензори, модули, изпълнителни механизми, Arduino Shields и други.

Arduino Uno е микроконтролерна развойна платка, изградена с микроконтролер Microchip Atmega328P, официално разработена от Arduino.cc (https://www.arduino.cc/). Платката е оборудвана с 14 цифрови входно-изходни (I/O – input/output) порта, 6 аналогови входа, 16 MHz кварцов резонатор, четири светодиода (един потребителски, свързан на 13-ти цифров входно-изходен порт, и три, които индикират работата на платката: ON, Tx и Rx), USB конектор, захранващ куплунг, бутон за рестартиране и ICSP конектор. Пиновете имат възможността да бъдат свързани към различни разширителни платки и други схеми. Шест от цифровите входно-изходни порта могат да се използват като PWM (PWM – Pulse-width modulation, в превод – ШИМ – широчинно импулсно модулирани) изходи. Свързването с компютър се осъществява посредством USB кабел USB A – USB B.

Uno може да се захранва през USB порта на компютъра или от външен източник, като превключването между различните начини за захранване е автоматично. Външният източник на захранване може да е и DC адаптер 7 – 12 V батерия.

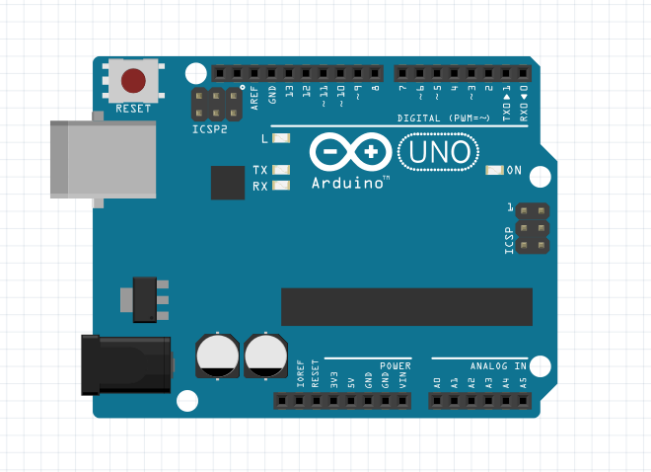
В Uno Revision 3 микроконтролера Atmega8U2, програмиран като USB-to-Serial адаптер, служещ за USB връзката, е сменен с Atmega16U2. След пина Aref са добавени два допълнителни пина за сигналите SDA и SCL (I2C протокола). Бутонът за рестартиране е преместен до USB конектора, а след пина има нов порт IOREF.

Референтният дизайн на хардуера се разпространява под лиценз Creative Commons Atrributions Share-Alike 2.5 и е достъпен на посочения сайт на Ардуино. Налични са и оформление и производствени файлове за някои версии на хардуера. Цялата хардуерна реализация може да бъде достъпена на следния линк: https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf



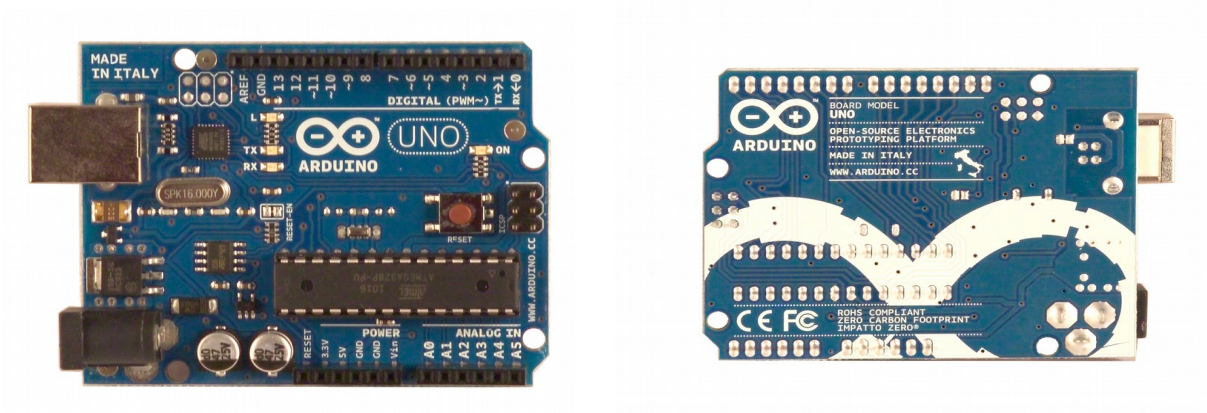
Фигура 6. Arduino Uno Revision 3

Фигура 7. Arduino Uno Rev3, визуализиран през симулатора TinkerCad



Технически характеристики на Arduino Uno:

Работно напрежение: 5 V;
Входно напрежение (препоръчително): от 7 V до 20 V;
Входно напрежение (работни граници): от 6 до 20 V;
Цифрови входно-изходни пинове: 14 на брой;
Аналогови входно-издохни пинове: 6 на брой;
DC Current за входно-изходен пин: 40 mA;
DC Current за 3.3 V пин: 50 mA;
Флаш памет: 32 KB, от които 0.5 KB се използват за стартиране;
SRAM (Static RAM): 2 KB;
EEPROM (Electrically Erasable Programming ROM): 1 KB;
Clock Speed: 16 MHz;
UART (Universal synchronous receiver-trasnmitter): 1;
I2C: 1; *SPPI:* 1;
Първо измерение: 68.6 mm;
Второ измерение: 53.4 mm;
Маса: 25 g;



Фигура 8. Предна и задна страна на Arduino Uno Revision 3

Вход и изход:

Всеки един от 14-те цифрови пина може да бъде използван като вход и изход за микроконтролера, като потребителят използва функциите *pinMode(), digitalWrite()* и *digitalRead()*. Те оперират при работно напрежение от 5 V. Всеки един пин също така може да получи и изпрати максимален ток от 40 mA. В допълнение, някои пинове имат и допълнителни функции:

1. Сериен: Пинове с номера 0 (RX) и 1 (TX). Тези пинове се използват съответно за получаване и трансфер на серийни данни от тип TTL (Time to Live, в превод – време за живот на данните). Тези пинове се

свързват към съответните им пинове от ATmega8U2 USB-to-TTL Serial chip.

2. Външни прекъсвания: Пинове с номера 2 и 3. Тези пинове могат да бъдат конфигурирани при поява на прекъсване при по-ниски стойности или въобще при промяна в стойностите (за повече подробности може да се направи справка с функцията *attachInterrupt()*).

3. PWM: Пинове с номера 3, 5, 6, 9, 10 и 11. Тези пинове осигуряват 8-битов широчинно-импулсно модулиран изход посредством функцията *analogWrite()*.

4. SPI: Пинове с номера 10 (SS), 11 (MOSI), 12 (MISO) и 13 (SCK). Тези пинове отговарят за поддръжката на SPI комуникацията, използвайки стандартната SPI библиотека.

5. LED: Пин с номер 13. В самият микроконтролер има вграден LED, който е свързан с този пин. Когато пинът приема висока стойност (пинът е в режим на HIGH), въпросният LED е включен, а когато приема ниска стойност (пинът е в режим на LOW), същият LED е изключен.

6. Аналогови входове: Всички те са обозначени с A0, A1, A2, A3, A4 и A5. Всеки от тях разполага с 10 бита за резолюция (тоест 2^{10} или 1024 различни стойности). По подразбиране те измерват напрежението от маса до максимално допустимите за микроконтролера 5 V, макар че е възможно горната граница за определяне на тези стойности да бъде променена, използвайки се пина с обозначение AREF и функцията *analogReference()*. Допълнително някои пинове имат и специална функционалност. Това са A4 или SDA пин и A5 или SCL пин. Поддържат TWI комуникация, използвайки библиотеката Wire.

7. AREF – Този пин реферира работното напрежение за аналоговия вход. Използва функцията *analogReference()*.

8. Reset – Променя режима на целия микроконтролер на LOW, като по този начин микроконтролерът се нулира. Обикновено този пин се използва с това свое предназначение за добавяне на бутон, който да успее да нулира всички останали използвани shields, както и да блокира shield-а на платката.

<u>Комуникация:</u>

Arduino Uno има набор от помощни средства за комуникация, които му позволяват да комуникира с компютър, друго Arduino или въобще с други микроконтролери. ATmega328 поддържа UART TTL (5 V) серийна комуникация, която е достъпна за дигиталните пинове с номера 0 и 1, съответно RX и TX. ATmega16U2 върху платката свързва тази серийна комуникация посредством USB и се появява като виртуален порт към друг софтуер или компютър. Софтуерът за микроконтролера включва и сериен монитор, който позволява прости текстови данни да бъдат изпратени към и

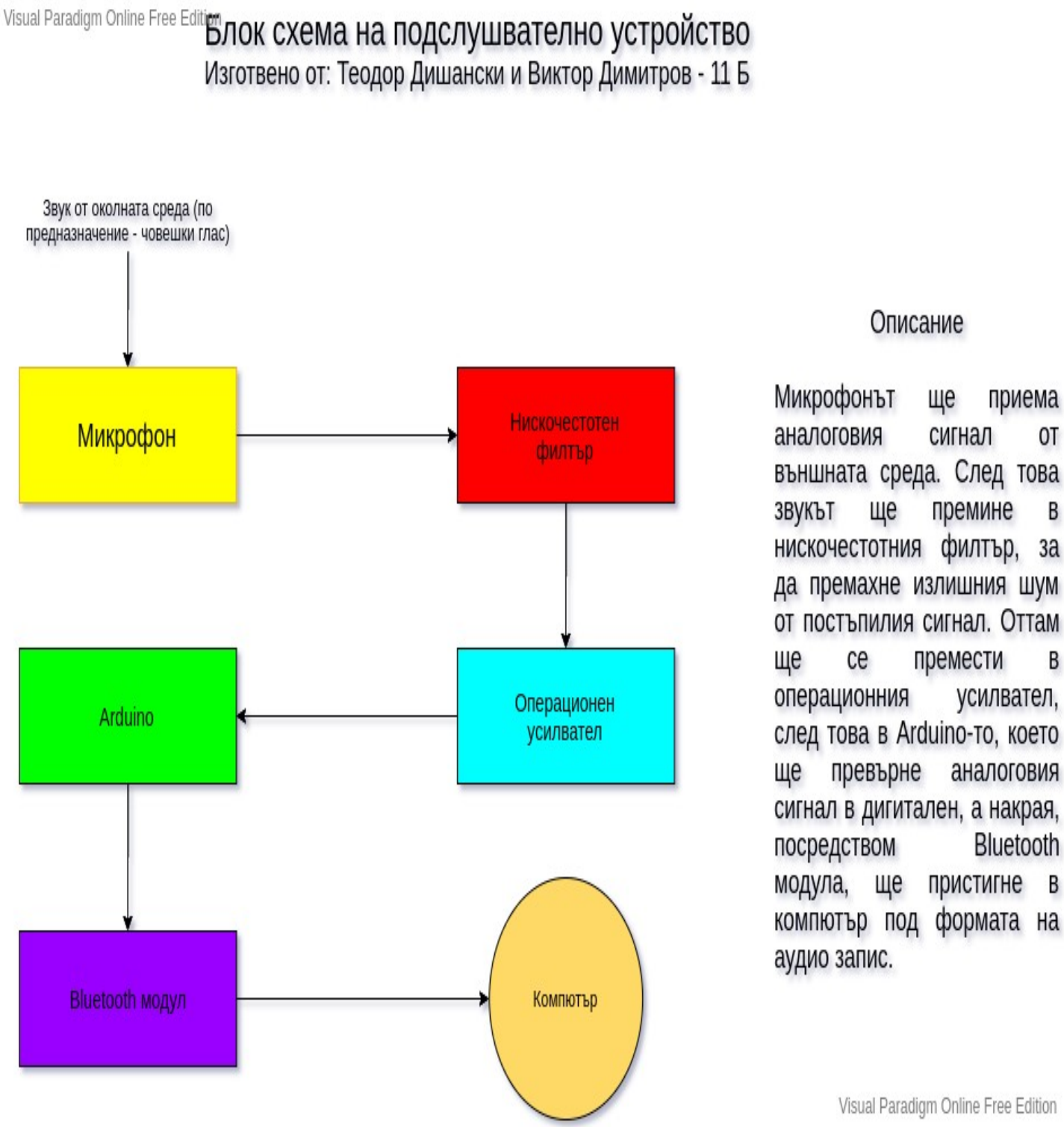
от платка на Arduino. RX и TX LEDs на платката ще изчезнат щом данните вече са изпратени с USB-to-serial чипът и USB връзката към компютъра За софтуерна комуникация обикновено се използва библиотеката SofwareSerial, която между другото позволява тази софтуерна комуникация да се осъществи от кой да е дигитален пин от Arduino Uno. ATmega328 също така поддържа I2C (TWI) или SPI комуникация. За олесняване употребата на I2C комуникацията, се използва и вече споменатата Wire библиотека. На практика при една стандартна SPI комуникация се използва SPI библиотеката.

За повече информация по софтуерните възможности или хардуерната реализация на Arduino Uno Revision 3, може да погледнете и тук:
https://www.farnell.com/datasheets/1682209.pdf – Datasheet на контролера
https://www.arduino.cc/en/Main/Boards – Разновидности на Arduino
https://www.arduino.cc/en/Reference/SPI – Менажиране на библиотеката SPI
https://www.arduino.cc/en/Reference/Wire – Менажиране на библиотеката Wire
https://www.arduino.cc/en/Hacking/PinMapping168 - „Карта“ на пиновете
https://www.arduino.cc/en/software – Изтегляне на Arduino IDE
https://www.arduino.cc/reference/en/ - Документация на функциите
https://www.arduino.cc/en/Tutorial/HomePage – Уроци за програмиране с Arduino

3.3. Блокова схема на устройството – хардуер. Начин на свързване на отделните компоненти в схемата на устройството. Хардуерна реализация.

На фигура 9 може да се види блоковата схема на подслушвателното устройство. Първото звено, което е необходимо, за да работи подслушвателното устройство, е наличието на звук от околната среда. По предназначение това би трябвало да бъде човешки глас. Звуковият сигнал от външната среда ще бъде прихванат от микрофон. Всеки един звуков сигнал по същество е аналогов сигнал. Това означава, че той няма точно определена (ни най-малко пък константна) амплитуда, както и че останалите характеристика на сигнала са изключително променливи и зависими от самия звук. Именно заради това е необходимо и следващото звено – нискочестотният филтър. Неговата работа е да изчисти постъпилия аналогов сигнал от така наречения шум – това е нежелан и необработен звук (често смятан за неприятен, а в случая с устройството – даже променящ съдържанието на оригиналния сигнал). За да се избегне грешката от дискретизация, като следващо звено използваме операционен

усилвател, който ще усили вече изчистения от шума сигнал. С негова помощ, като краен резултат, след преминаването си през трите звена, даден звуков сигнал ще се възвърне в своята „оригинална форма“, тоест – ще премине успешно изчистването от шума и усилването си, така че да премине към нататъшното звено, което се явява и най-важното, а именно – Arduino Uno микроконтролера.



Фигура 9. Блокова схема на подслушвателното устройство

Неговата работа ще е да превърне аналаговия сигнал в дигитален или цифров, тъй като без неговата помощ нито едно от звената след това не би работило коректно, защото данните не биха могли да се съхраняват в аналогов вид, а единствено в цифров. Другата задача на микроконтролера е да препрати вече превърнатия в дигитален сигнал на Bluetooth модула, който ще отговаря за това да осъществи комуникацията посредством Bluetooth връзка. Това е неговата приоритетна задача, тъй като в основата на концепцията на самата курсова работа е информацията да се предава с Bluetooth. Накрая сигналът постъпва в компютър, който ще го преработи в аудио запис.

3.4. Алгоритъм на софтуерното решение и работата на кода. Софтуерна реализация

Кодът на задачата е написан на езика ца програмиране C++. Това е мощен език за програмиране, който поддържа и концепцията за ООП (обектно ориентирано програмиране). Кодът би работил и при копирането му във файл, зареждащ код на езика за програмиране C, тъй като Arduino поддържа програмирането и на двата възможни езика.

По начало включваме заглавните файлове (Header files) Arduino.h и SoftwareSerial.h. Тези файлове са предпроцесорни директиви към стандартните библиотеки, поддържащи работата с Arduino и комуникацията от тип Software Serial на Arduino с други звена, както вече беше казано по-горе. Тук е важно да се спомене, че, ако се работи през Arduino IDE, тези заглавни файлове с техния ресурсен код са включени автоматично и тяхното менажиране от потребителя не е необходимо. Но ако се работи през текстов редактор, който се използва само с цел набиране на кода, то тези заглавни фалйове, заедно с ресурсния им код, трябва да бъдат изтеглени допълнително от официалния сайта на производителя на викроконтролера и да се поставят в същата папка, в която се намира и ресурсният код за стартиране на проекта.

```
#include <Arduino.h>
#include <SoftwareSerial.h>
```

Фигура 10. Включване на заглавните файлове

Следващото нещо, което правим, е да дефинираме пиновене които ще бъдат използвани като входно-изходни пинове. Тези пинове са три на брой. Първият е пинът с обозначение A0 – това е аналоговият пин на Arduino Uno. Другите два пина са с обозначения 8 и 9. Тези пинове са дигитални и те ще осъществяват действията, които осъществяват и RX и TX. Благодарение на цифровите пинове след това можем да осъществим връзката си от тип Software Serial като създадем обект от този тип. Неговото име е mySerial, който ще използва посочените пинове с номера 8 и 9.

```
#define analogPin A0
#define pinArduinoRX 8
#define pinArduinoTX 9
```

Фигура 11. Дефиниране на изполваните пинове

```
SoftwareSerial mySerial(pinArduinoRX, pinArduinoTX);
```

Фигура 12. Инициализиране на обект от тип SoftwareSerial

Всяко управление на Arduino се организира с помощта на две основни функции. Те се казват *setup()* и *loop()*. И двете са от тип void и не връщат резултат. Първата функция ще се изпълни само един път и това ще се случи тогава, когато кодът се зареди в микроконтролера. Втората функция ще се изпълнява многократно до момента, в който не се спре цялата работа на микроконтролера (с помощта например на бутон) или до момента, в който не се прекрати захранването на Arduino (което е силно непрепоръчително поради възможност от изгаряне на уредите, които се използват към момента на използване на устройството). Тази функция може да се разглежда и като своеобразен цикъл, който се изпълнява постоянно при работата на микроконтролера.

```
void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);
}
```

Фигура 13. Функция setup()

Функцията *setup()* изпълнява две команди. Първата команда се отнася до това да дефинира скоростта на предаване на данните вътре в самия микроконтролер. Тази скорост на предаване се измерва в битове за секунда (в чуждоезикова литература често споменавано и като baud rate). По същество това е предаване на серийни данни. В случая с подслушвателното устройство въпросната скорост е 9600. Това означава, че скоростта на предаване на сигнала ще бъде 9600 бита за секунда. Тъй като mySerial ще е необходим за осъществяването на комуникацията, неговата скорост на предаване трябва да е същата. В противен случай биха могли да възникнат проблеми. Ако скоростта на предаване на сигнала при комуникацията е по-малък, това означава, че би се получило загуба на информация, тъй като сигналът ще пристига по-бавно до компютъра, а нов сигнал вече ще е постъпил в самия микроконтролер. Обратно, ако скоростта на предаване на сигнала при комуникацията е по-голяма, това пък означава, че ще се получи накъсаност на информацията, от което след това не става съвсем ясно в какъв вид ще пристигнат цифровите стойности на постъпилия сигнал, което също не е алтернатива при работата на едно коректно изпълнено техническо решение на подслушвателно устройство.

```
void loop() {
  int readValue = analogRead(analogPin);
  float voltage = readValue * 5.0 / 1024.0;
  mySerial.println(voltage);

  delay(20);
}
```

Фигура 14. Функция loop()

Функцията *loop()* в своята пълнота ще изпълнява преносът на данните и ще повтаря цикъла по постъпване на сигнал и изпращане на сигнал многократно. Първата команда във функцията прочита стойността от аналоговия пин и записва тази стойност в целочислената променлива с име *readValue*. Тази стойност след това се умножава по максималното допустимо работно напрежение от 5 волта, а получената стойност се разделя на всички възможни 1024 стойности, които могат да постъпят като аналагов вход, както бе посочено в точка **3.2. Arduino Uno.** Тъй като се осъществява операцията деление, то записаната новополучена стойност се съхранява в променлива от тип число с плаваща запетая с единична точност или другояче казано – променливата с име *voltage*. След това тази променлива се извежда на екрана на серийния монитор на Bluetooth вградената система с използването на вградената функция за обекта, който

създадохме за комуникаци, която носи името *println()*. Това също така означава, че след всяка изведена стойност, автоматично ще се извежда и знака *\n*, което означава, че всяка една стойност ще се вижда на нов ред при извеждането си. След извършването на всеки три такива команди, за онагледяване на искания резултате се извършва и четвъртата команда *delay(20)*, която осъществява изчакване от 20 милисекунди преди извеждането на новата стойност. Например: след като е изведена първата обработена стойност, ще се изчака период от време 20 милисекунди, след това ще се изведе втората обработена стойност, после ще се изчака още един период от време 20 милисекунди, после ще се изведе третата обработена стойност, отново ще се изчака един период от време 20 милисекунди и така нататък. Разбира се, това не е задължителна команда и нейното използване е представено, за да може да се онагледи ситуацията при изписването на различните стойности. В оригинала на кода тази команда е изтрита.

Кодът от Фигури с номера 10, 11, 12, 13 и 14 може да бъде запазен като скица (sketch) в Arduino IDE. В оригинала на курсовата работа тази скица носи името *sketch_apr16a.ino*. При други обстоятелства, използвайки се единствено текстов редактор, а не среда за разработка, самият код може да бъде запаметен в паметта на компютъра и като *main.cpp*, но в такъв случай ще се наложи в същата директория на този файл да се намират и допълнителните файлове *SoftwareSerial.cpp* и *SoftwareSerial.h*, които ще са необходими при работата с библиотеката SoftwareSerial.

За онагледявене на работата на програмата дотук, както и за визуална комуникация с Bluetooth модула на програмата, е използван допълнителен софтуерен продукт с името Bluetooth Serial Terminal (https://www.microsoft.com/en-us/p/bluetooth-serial-terminal/9wzdncrdfst8?activetab=pivot:overviewtab), който се заема с това да принтира всички стойности, които се извеждат на изхода на Bluetooth. Резултат от това е показан на Фигура 15.

Фигура 15. Резултат от работата на програмата

Както може да се види, всяка една стойност се извежда на отделен нов ред. За да работи този софтуерен продукт, първо трябва да е активирана опцията *Terminal ON*, а на падащото меню с устройствата за Bluetooth комуникация е необходимо да се избере *HC-05*, което на практика е Bluetooth модулът, използван в курсовата работа.

```
import processing.serial.*;

Serial mySerial;
PrintWriter output;
```

Фигура 16. Начало на програмата за четене на стандартния изход от Bluetooth Serial Terminal

След като веднъж вече стойностите са изведени, то остава те да се запишат в текстов файл. За целта е необходимо да се напише допълнителна програма, която единствена цел ще е да прави именно това – да прочете всичко, което се изписва на изхода на терминала на комуникация в отделен файл. Тъй като езикът за програмиране C++ е базов за създаването на езика за програмиране Java, то не е трудно за досещане човек да предпочете да напише една такава програма именно на Java.

За да се създаде едно такова приложение обаче е необходима и още една допълнителна среда за разработка, която се казва Processing IDE (https://processing.org/reference/environment/), която ще се заеме с това да стартира програмата за извършването на вече посочените действия по зададените горепосочени цели. На практика тази среда за разработка не използва точно Java, а езикът за програмиране Processing, който поддържа парадигмата за обектно ориентирано програмиране, както и възможност за работа с микроконтролер (в частност – използвания в курсовата работа Arduino Uno Revision 3). Програмният код в тази среда за разработка се запазва във формата *.pde*.

На Фигура 16 е показана началната част от кода, която е необходима за допълнителната програма. На първо място изпращаме съобщение до средата за разработка, че ще използваме класът *serial*, както и всички негови методи. Този клас се използва за изпращане и получаване на данни посредством протокола за серийна комуникация. За да го използваме обаче е необходимо и да създадем обект от типа *Serial*, който, както може да се види на картинката по-горе, е наречен с аналогичното име *mySerial*. Скоростта на предаване на данните на този обект за серийна комуникация по подразбиране е 9600 бита за секунда. Това отговаря и на нашата скорост на предаване на серийната комуникация с микроконтролера и затова не е необходимо да променяма самата скорост. За повече информация какво прави и как се използва този клас, може да посетите този линк: https://processing.org/reference/libraries/serial/Serial.html. Вторият обект, който създаваме е инстанция на класа *PrintWriter*, който позволява на символите да бъдат изведени като тектово-изходен поток. При нас този обект се нарича *output*. За повече информация какво прави и как се използва този клас, може да посетите този линк: https://processing.org/reference/PrintWriter.html.

```
void setup() {
   String normalPort = Serial.list()[0];
   String bluetoothPort = Serial.list()[2];

   mySerial = new Serial(this, bluetoothPort, 9600);
   output = createWriter("data.txt");
   mySerial.bufferUntil('\n');
}
```

Фигура 17. Функция setup(), използвана в програмата за четене на стойностите от изхода на Bluetooth Serial Terminal

Основната функция, която се използва за програмата, е *setup()*. Аналогично и тук тя е от тип *void* и не връща никакъв резултат. Първите две операции, които се извършват в нея, са най-значимите, за да може програмата да работи коректно, а именно – да се обозначат портовете, които ще се използват за комуникация с микроконтролера и портовете за комуникация с модула. Чрез метода *list()* на класа *Serial* взимаме портове с номера 1 и 3. Тъй като методът в действителност не връща само исканият порт, а лист от всички възможни портове, то ние трябва да вземем желаният от нас порт на съответия индекс от списъка, на който се намира той. За повече информация за този метод, вижте тук: https://processing.org/reference/libraries/serial/Serial_list_.html. Двата обекта от тип *String* с имена *normalPort* и *bluetoothPort* са имената на нужните ни портове.

На следващо място трябва да осъществим комуникация с порта за Bluetooth комуникация. За целта дефинираме инстанцията на класа *Serial* с инстанция конструктор за родителя му (което и същата инстанция, затова като първи параметър е написано *this*), порта за комуникацията (което е текстовата стойност с име *bluetoothPort*) и вече спомената скорост на предаване по подразбиране от 9600 бита за секунда. На следващо място ни е необходимо да създадем файл, в който ще записваме стойностите, които ще използваме. За да направим това, създаваме нов обект от типа на PrintWriter, което се осъществява с вградения метод *createWriter()*, който получава като аргумент път към файла. В случая на програмата този файл се казва *data.txt*. На последно място поставяме нов ред при принтиране на стойностите в конзолата на всеки определен брой принтирани символи, което пък се осъществява с вградения метод *bufferUntil()* на класа *Serial*.

Всичките тези операции са показани на Фигура 17.

```
void draw() {
    if (mySerial.available() > 0 ) {
          String value = mySerial.readString();
          output.println(value);
          printArray(value);
    }
}
```

Фигура 18. Функция draw(), използвана в програмата за четене на стойностите от изхода на Bluetooth Serial Terminal

Другата основна функция носи името *draw()*. Тя е аналогична на функцията *loop()* от езика за програмиране C++, тъй като работата ѝ е за продължителен период от време да изпълнява специфичен блок от кода, заключен именно в гялото на функцията до момента, в който програмата не се терминира или не приключи поради някакви външни обстоятелства. Важно е да се смопене, че тази функция никога не бива да се извиква експлицитно, защото всички програми, използващи езика за програмиране Processing рестартират своя екран след едно приключване на функцията *draw()*, а не по-рано. В тялото на функцията четем символите на изхода на Bluetooth Serial Terminal. За да направим това, първо трябва да проверим дали въобще има някакви прочетени байтове, за което използваме вградения метод *available()* на класа Serial. Ако такива има, то ние запаметяваме прочетената стойност с вградения метод *readString()* под формата на обект от типа *String*, която ще изведем на екрана. Тъй като стойностите са в текстов формат, то е задължително те да бъдат изведени на нов ред и това се получава с функцията *printArray()*, която получава като параметър променливата от тип *String* с име *value*.

```
void keyPressed() {
    output.flush();  // Writes the remaining data to the file
    output.close();  // Finishes the file
    exit();  // Stops the program
}
```

Фигура 19. Функция keyPressed(), използвана в програмата за четене на стойностите от изхода на Bluetooth Serial Terminal

На последно място създаваме функцията *keyPressed()*. По подразбиране тази функция се извиква всеки път, когато е натиснат бутон от клавиатурата и символът от този бутон се съхранява в променливата key. В нашия случая обаче в тази функция се извкват два метода и една функция. И двата метода са вградени за класа *PrintWriter*. Първият метод записва оставащите данни, които сме съхранили, във файла, който вече сме създали. Вторият метод затваря файловия поток и самият обект от тип *PrintWriter*. Накрая извикваме функцията *exit()*, която независимо терминира програмата. С това нашата програма може да се сметне за напълно готова. Реализацията на функцията е показана на Фигура 19.

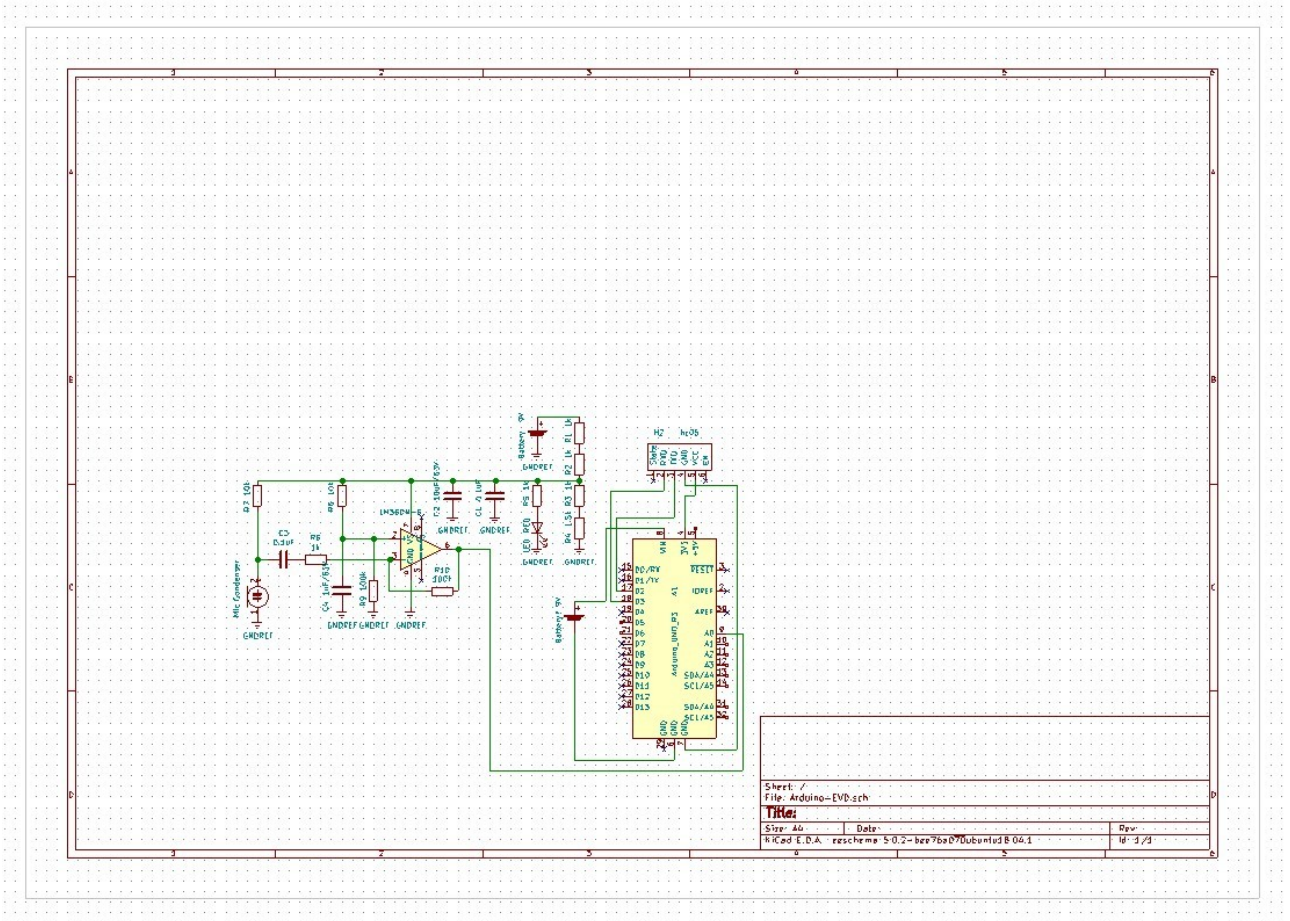
Файлът със запаметения код (в оригинала на курсовата работа – носещ името *sketch_210512a.pde*) се налага да се намира в една и съща

директория с горепосочения *.ino* файл, за да може програмата да се изпълни коректно. Създаденият текстов файл при работата на програмата също ще се намира в същата директория, което е разбираемо, имайки се в предвид, че програмата се изпълнява в същото местоположение на операционната система. Целият ресурсен код е достъпен в Github Repository-то на курсовата работа.

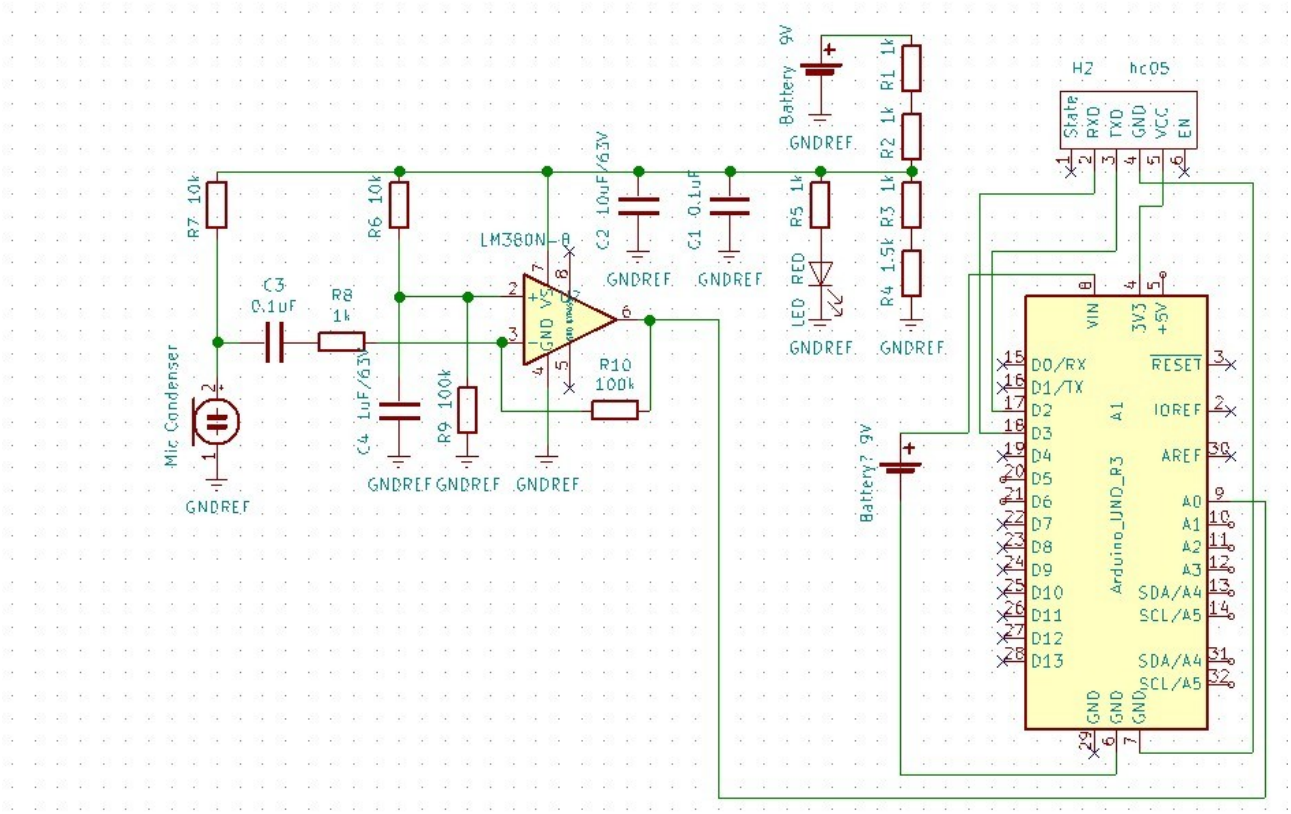
4. Принципна електрическа схема, нейните компоненти и графични оригинали на печатната платка

4.1. Принципна електрическа схема

Принципната електрическа схема е направена на приложението KiCad. Това е безплатен софтуерен продукт, който се използва за автоматизация на електронни дизайни при разработването на устройства, схеми, връзки между компоненти и графични оригинали (софтуерният продукт може да бъде изтеглен оттук: https://www.kicad.org/download/). Въпросната среда за разработка дава достъп до голям набор от инструменти, които могат да бъдат използвани при разработването на някакъв по-голям и мащабен проект.



Фигура 20. Умален модел на принципната електрическа схема на курсовата работа, хардуерно реализирана на KiCad

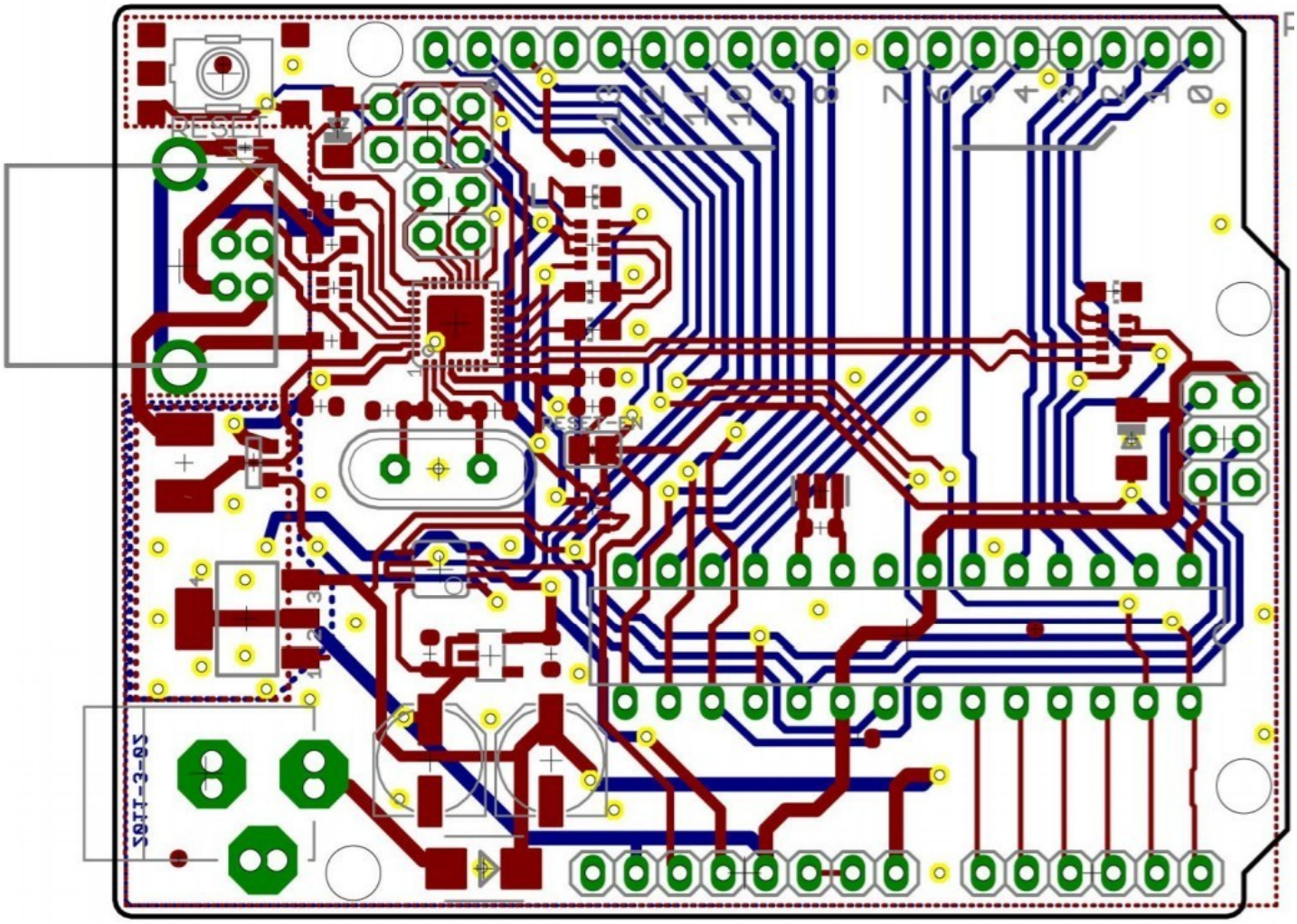


Фигура 21. Увеличен модел на принципната електрическа схема на курсовата работа, хардуерно реализирана на KiCad

Както вече стана ясно, за захранване са използвани две батерии с напрежение от 9 V. Едната батерия е за захранване на операционния усилвател, а другата е за захранване на микроконтролера Arduino Uno Revision 3. В схемата също така са използвани и няколко резистора с различно съротивление – 1 kΩ, 1.5 kΩ, 10 kΩ, 100 kΩ, както и няколко кондензатора – от 0.1 μF и 10 μF. Използван е и LED диод, светещ в червена светлина.

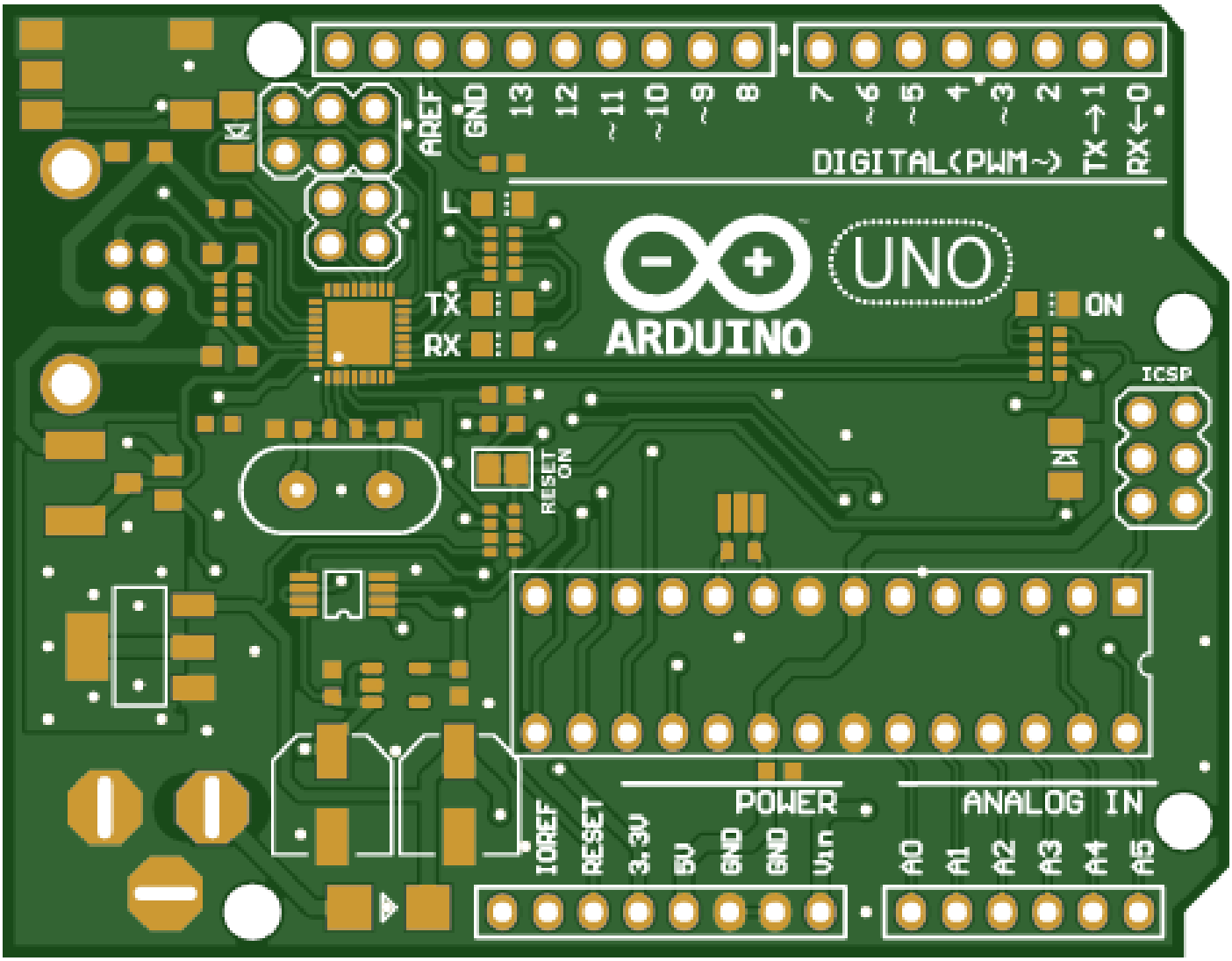
4.2. Графични оригинали на печатната платка

За да се онагледят по възможно най-добър начин всички „пътища“ в печатната платка, хубаво е те да бъдат онагледени върху специална схема, която се нарича графичен оригинал на печатната платка. Този оригинал показва всички възможни свързвания и компоненти върху платката на Arduino Uno Revision 3.



Фигура 22. Пътищата, онагледени върху графичния оригинал на печатната платка (картината е в своя уголемен размер, за да се онагледи възможно най-добре всичко от изображението)

Както може да се забележи, зелените правилни геометрични фигури всъщност описват пиновете, както и местата, през които на практика ще протича някакъв ток. Именно затова те са обозначени с по-различен цветови код върху схемата.



Фигура 23. Различните компоненти върху платката на Arduino Uno Revision 3 (може да се забележи приликата с Фигура 22, при която се описват пътищата, докато тук се описват само и единствено компонентите)

Аналогичен е и графичният оригинал на платката, при който се виждат компонентите, но не и пътищата. В действителност разликата между Фигура 22 и Фигура 23 е в това, че първата фигура описва пътищата върху платката на Arduino Uno Revision 3, докато втората описва детайлно компонентите на същата тази платка.

5. Реализация на курсовата работа. Възникнали трудности при изработването ѝ и намирането на решения на възникналите проблеми

При реализацията на курсовата работа възникнаха немалък брой трудности, които изискваха бързо решение и адекватен подход, след като такива възпрепятствия бяха регистрирани. По-значимите трудности, за които си заслужава да се отдели специално внимание, бяха хардуерните форми на възпрепятсване, които се обособяваха в това да създаде напълно коректно работеща верига, така че в края на краищата да не изгори някой от компонентите. За тази цел бяха извършени множество от калкулации по няколко пъти. Друга хардуерна трудност е поставянето и последващото премахване на сензора HX711, който по начало се оказа, че може да бъде полезен, но в края на краищата се оказа напълно безсмислен, тъй като това е сензор за тежести. Възникна също и известно объркване при свързването на микроконтролера с компютъра и проблемът се констатираше с това, че пиновете RX и TX не биваше да бъдат свързани с нищо по време на работата на подслушвателното устройство. По-значимите трудности от софтуерната част се състояха в това да се четат правилно стойностите от изхода на Bluetooth Serial Terminal и да се записват в отделен текстов файл. Тези проблеми бяха разрешени, както стана ясно, чрез написването на допълнителна програма, чиято основна цел е именно това. Друг проблем, който възникна по време на писане на кода бе и правилното записване на тези стойности, тъй като по начало ги записвахме неправилно и неправилността идваше от факта, че не извеждахме всяка стойност на нов ред, което допълнително ни объркваше, но и което разрешихме бързо.

Разбира се, без помощта на господин Росен Витанов, никога не бихме се справили, защото той винаги беше отворен за въпроси по курсовите работи на всички ученици и ни оказа безкрайна помощ с напътствията си.

6. Какво ново научихме, докато работехме по курсовата работа?

Много. Изключително много. И двамата за пръв път използвахме микроконтролер в нещо толкова мащабно като курсовата работа по учебната дисциплина. За пръв път запоявахме с поялник и свързвахме реални компоненти в истинска електрическа верига (а не както досега сме се учели да правим това само на теория). Тук се научихме и да управляваме микроконтролер чрез код на C / C++ и Java / Processing. Макар е да бяхме наясно до известна степен с тези езици за програмиране, тук се сблъскахме с напълно непозната територия, както се казва, която обаче успешо завладяхме с дълги нощи пред компютрите и воля при подготовката на курсовата работа по предмета „Вградени микрокомпютърни системи“. Също така именно с тази курсова работа видяхме реални проблеми, с които може да се сблъска един електроинженер или системен програмист и отново с курсовата работа търсихме и намирахме решение на възникналите проблеми, които се появяваха през времето на изготвянето на проекта.

7. Използвани ресурси за изработването на курсовата работа - софтуерни продукти и хардуерни елементи

За курсовата работа бяха използвани различни софтуерни и хардуерни елементи, без които ние, разработчиците, не бихме се справили, и на които производители благодарим.

1. Arduino IDE – https://www.arduino.cc/en/software
2. Microsoft Visual Studio Code - https://code.visualstudio.com/
3. Platform IO - https://platformio.org/
4. Processing IDE - https://processing.org/reference/environment/
5. KiCad EDA - https://www.kicad.org/download/
6. Arduino Uno Revision 3 - https://store.arduino.cc/arduino-uno-rev3
7. Радимекс БГ - https://radimexbg.com/
8. Елимекс БГ - https://elimex.bg/