Ръководство за курсово проектиране по микропроцесорна схемотехника

ас. маг. инж. Н. Кадиев

2026

Съдържание

Въведение

Настоящото ръководство е предназначено за студентите от Факултет Електронна Техника и Технологии, специалности „Електроника“, „Електронни Информационни Системи“ и „Автомобилна Електроника“, но може да се използва от ученици и студенти от други направления. Основната му цел е да зададе опорни точки при разработването на курсов проект по предметите „Микропроцесорна схемотехника“ и „Микропроцесорна системотехника“, чрез поетапно изработване на примерна тема. С леки изменения, материала може да послужи като структура за изготвяне на дипломна работа от бакалаври и магистри.

Изисквания:

- шрифт Times New Roman;
- размер на шрифта 14pt;
- 1.15 междуредие;
- двустранно подравняване на текста;
- PDF формат;
- номерация на страниците;
- физическа реализация на макет не е нужна.

Структура:

- Заглавна страница;
- Задание;
- Съдържание;
- Литературно проучване;
- Избор на компоненти;
- Разработване на хардуер и фърмуер;
- Заключение;
- Използвани източници.

Примерно задание:
Тема: Система за източване на 10 л. воден резервоар
Използвана периферия: GPIO, прекъсвания

1. Литературно проучване

В тази глава се помества литературно проучване по темата на проекта. Започва се с ясно посочване на целта на работата. След това се разписват въпросите:

Какво е предназначението на проектираното устройство?

В коя част на бита може да намери приложение?

Какви са наличните пазарни решения?

Какви са подходите за реализиране на подобен тип устройство?

В тази и следващите глави се цитират източниците, които сте използвали по време на проектирането. Цитирането се извършва с използването на [X] като в тях се поставя поредния номер на източника. След това, източника се поставя в главата „Използвана литература“, „Литературни източници“ или само „Литература“, където източниците се подреждат по номера (1, 2, 3 ...). Ако цитирания източник е учебник, книга или статия, то се посочват авторите, заглавието и къде е публикувано / отпечатано. Ако използвания източник е онлайн „tutorial“ или сайт, то се цитира заглавието и линк към сайта.

Когато се добавят изображения (фигури), таблици или графики, те се центрират на листа и под тях се добавя текст („caption“) в следния формат - **<вид> <пореден номер>: <кратко описание>**, например *Фиг. 1: Помпено отделение на цистерна*, *Таблица 1: Електрически характеристики на помпени инсталации*. Поредния номер може да е общия номер на вмъкнатото съдържание или да се свърже с главата, по която се работи: *Глава 1 – Фиг.1.1, Фиг. 1.2 ... Фиг. 1.25, Глава 2 – Фиг.2.1, Фиг. 2.2 ... Фиг. 2.5 и т.н*. По кой от двата начина ще се работи е индивидуално решение, но щом веднъж се избере, то трябва да се спазва от начало до край. Ако дадено съдържание е взето на готово, след описанието се цитира мястото, от където е взето: *Фиг. 1: Помпено отделение на цистерна [3]* и ако източника не е добавен в списъка с цитати, то той се добавя по посочения начин.

2. Избор на компоненти

В тази глава се прави основния подбор на компонентите. След направата на литературното проучване са на лице най-често използваните методи и компоненти. Избраната тук последователност дава оптимален начин за решаване на голяма част от инженерните задачи. Алтернативен начин е първо да се избере микроконтролер (с който сте най-добре запознати или с който разполагате) и след това да се подберат сензори и захранване. Без значение от избраната последователност, водещи са изходните данни от заданието. Възможно е след първоначално реализиране на устройството, то да не отговаря изцяло на всички изисквания. В този случай се налага преработка на даден блок или на целия прототип. С оглед, че тук става въпрос за курсов проект, заложените изисквания са сравнително леки и лесно се удовлетворяват.

2.1 Избор на типа сензори и актуатори

Отправна точка при избора на сензори е заданието. Когато в него ясно са уточнени вида на сензорите и актуаторите, избора е сравнително бърз и лесен. Но в случаите когато „темата е отворена“ се налага да проявим малко логическо мислене и находчивост.

Нека видим темата на проекта - „Система за източване на 10 л. воден резервоар“. Това не ни казва много, но ни дава достатъчна информация за отправна точка. Знаем, че става въпрос за 10 л. резервоар, който трябва да се източи и че въпросната течност е вода. Това означава, че в даден момент от време трябва да сме наясно дали резервоара е пълен, празен или в междинно състояние. Следователно трябва да знаем количеството вода в него. Това може да се реализира чрез следене на теглото на резервоара или на нивото на течността. Нека разгледаме методите за реализация на тези два варианта.

За измерване на теглото на съда обикновено се използва мостова схема с два тензорезистора. Те се монтират на метален лост, единия край на който е фиксиран, а върху другия се поставя измервания обект. В следствие на масата на измервания обект се получава деформация на лоста, която причинява изменение на съпротивлението на тензорезисторите. Промяната на съпротивлението, води до промяна на напрежението в диагоналите на моста. Този сигнал се усилва с диференциален или прецизен усилвател и микроконтролера може да определи точното тегло на обекта. На пазара се предлагат готови модули съдържащи моста, интегрална схема за управление с

цифров изход и монтираните сензори с метално рамо. Тук обаче се появяват някои ограничения от към максимално измерваното тегло (теглото на напълнения резервоар), както и необходимостта микроконтролера постоянно да следи теглото.

Варианта с измерване на ниво на течността е доста по-проста – трябва да знаем кога съда е напълнен и кога се е изпразнил. Тук имаме няколко варианта за обмисляне:

- поставяме утра-звуков сензор за измерване на разстоянието до повърхността на течността. Когато течността премине предварително зададена стойност, стартираме изпразването. Това решение е лесно за постигане, но отново се налага микроконтролера постоянно (или през определен време-интервал) да проверява нивото на течността.
- поставяме два сензора – единия на дъното, а другия на нивото на течността при обем 10 л. Горният сензор ще се свърже към извод на микроконтролера и ще предизвика прекъсване при напълване. Това прекъсване ще стартира процеса на източване, който ще бъде прекратен при задействане на сензора на дъното. При този вариант, микроконтролера може да се вкара в режим на ниска консумация и да се включва само при напълнен съд.

Избираме варианта с два сензора и сега трябва да изберем самите сензори. Нека за простота ограничим изборите до капацитивни сензори за ниво и магнитни сензори с поплавък. И в двата случая сензорите ще се монтират в или на повърхността на резервоара, и двата вида сензори могат да бъдат намерени като модули с цифров изход, но за още по-голямо опростяване на задачата нека изберем две рид-ампули (фиг. 1) и магнитен поплавък. Обърнете внимание, че този избор не е в разрез с поставените от заданието условия – не е зададен вида на съда и можем да изберем или изработим резервоар със слот за движение на поплавък и места за монтиране на сензорите.

Фиг. 1: Рид-ампула

След като избрахме метода за определяне обема на течността, е време да изберем и метода за източването ѝ. Стандартно това се прави с електромагнитен клапан или с помпа. Използването на клапан разчита на гравитацията за изместване на течността. Това означава, че резервоара трябва да се разположи на определено ниво над земята. От друга страна, помпата използва електрическата енергия за да измести течността и не се влияе от позицията на съда (в определени и допустими граници). За нашия случай ще изберем решение с използване на 12V помпа (перисталптична или за аквариум) – фиг. 2. Това означава, че се нуждаем от електронен ключ за пускане и спиране на помпата. Тъй като тук ще работим с ниско напрежение, можем директно да използваме транзистор, но в случаите с работа с мрежово напрежение (230V) по удачно е използването на реле или тиристор с галванично разделяне.

Фиг. 2: Мембранна водна помпа
12V/1A

За да изберем подходящ транзистор се водим от големината на тока през товара и напрежението, което ще превключваме. Добра практика е да се използва запас от 1,5 до 2 пъти стойността на желан параметър. Нека изберем да използваме N-канален MOS транзистор, който да е в корпус TO-220. В повечето случай, този вид транзистори са с достатъчно високо напрежение и водещ ще е тока. Ако тока през помпата е 1A, то със запас 2 пъти, търсения транзистор трябва да има максимален ток поне 2A. След известно ровене, решаваме, че ще използваме транзистор IRF540N, който е с U_{DSS} = 100V и I_D =22A.

2.2 Избор на подходящ микроконтролер

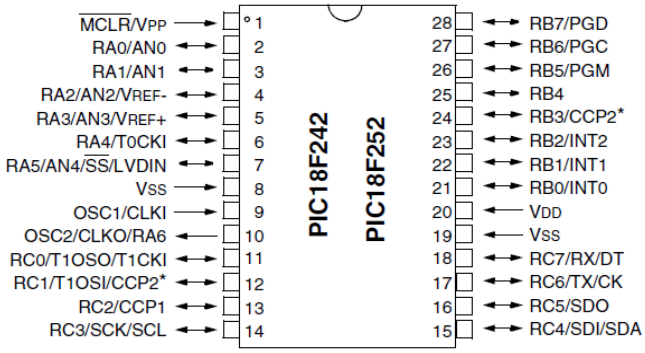
При избора на използвания контролер се ограничаваме до достъпните на пазара модели, които да отговарят на следните условия:

- можете да ползвате 8, 16 или 32 битов микроконтролер;

- микроконтролера може да е: MSP430xx (като този от лабораторните), PICxxFxxx (Microchip) или контролери на Atmel (Microchip), STM32xxx (STMicroelectronics), LPC21/20/17/13 (NXP), ESP32 (Espressif) или вариантите му, RASPBERRY PI PICO (Raspberry Pi Foundation), RASPBERRY PI PICO 2 или RASPBERRY PI PICO W;
- може да се използва Arduino и Arduino IDE при условие, че кода е написан на ниво регистър или чрез използване на базови периферни библиотеки (SPI, I2C, UART, ADC и т.н.) за получаване на крайния резултат. Обърнете внимание, че ардуино е името на развойните платка и платформа, а не на използваните микроконтролери!

След като се избере типа на микроконтролера, се избира подходящ чип (освен ако не се ползва ESP32 или RASPBERRY PI PICO). Това се прави като отново се вземат под внимание изискванията в заданието. Ако става въпрос за сравнително лека задача, без необходимост от голяма изчислителна способност, 8 или 16-битов микроконтролер ще свърши достатъчно добра работа. В противен случай трябва да се използва 32-битов. Друго изискване е общата консумация на устройството. Отново, ако целта е периодична проверка и вземане на решение, може да се използва по-мощен микроконтролер. Но ако става дума за постоянна работа – тогава е необходим по-строг подбор или компромис.

За целта на настоящата тема ще изберем 8-битовия PIC18F252 на Microchip (фиг. 3), който е с 5V захранване, но има и възможността да работи на 3,3V. Той се явява подходящ, тъй като нашата задача е да следим два сензора и да пускаме и спираме помпа. По-важното в случая е, че той дава възможност за по-пълни обяснения в следващите глави.



Фиг. 3: Различните функции за изводите на PIC18F252

2.3 Избор на вид захранване

На последно място, но не и по важност, е избора на подходящо захранване. На база избраните сензори – 12V/1A помпа и рид-ампули (за които напрежението не е от съществено значение) и избрания микроконтролер – 5V/(няколко-стотин μA до 1-2mA) PIC, можем спокойно да изберем захранващ източник с напрежение 12V и ток 2A – в случая може да е 12V адаптер.

Това напрежение е високо за микроконтролера и трябва да използваме стабилизатор на напрежение за да го понижим. За тази цел можем да използваме импулсен (понижаващ преобразовател) или линеен стабилизатор. Този избор зависи от мощността, която ще се отделя в стабилизатора:

$$P_{отд} = \left(U_{вх} - U_{изх}\right) \cdot I_{изх}$$

Ориентировъчно, ако $P_{отд}$ надвишава 1W, импулсните стабилизатори са за предпочитане. В противен случай, линейните стабилизатори са по-добрия избор. Това правило е ориентировъчно и трабва да е съобразено с изискванията на решаваната задача. Основното съображение тук е, че линейните стабилизатори разсейват тази мощност във формата на топлина и изискват охлаждане, с радиатор например. Колкото по-голяма е отделяната топлина, толкова повече се загрява стабилизатори и следователно по-голям трябва да е радиатора.

В нашия случай:

$$P_{отд} = \left(U_{вх} - U_{изх}\right) \cdot I_{изх} = \left(12 - 5\right) \cdot 1\,mA = 7\,mW$$

което ни позволява да използваме обикновен линеен стабилизатор. Нека изберем стабилизатор 7805.

3. Разработване на принципна схема и фърмуер

След като сме направили първоначалния подбор на компоненти, можем да започнем проектирането на принципната схема на устройството. Първоначално можем да нахвърляме идеи на лист хартия и да преминем към изчертаването в последствие.

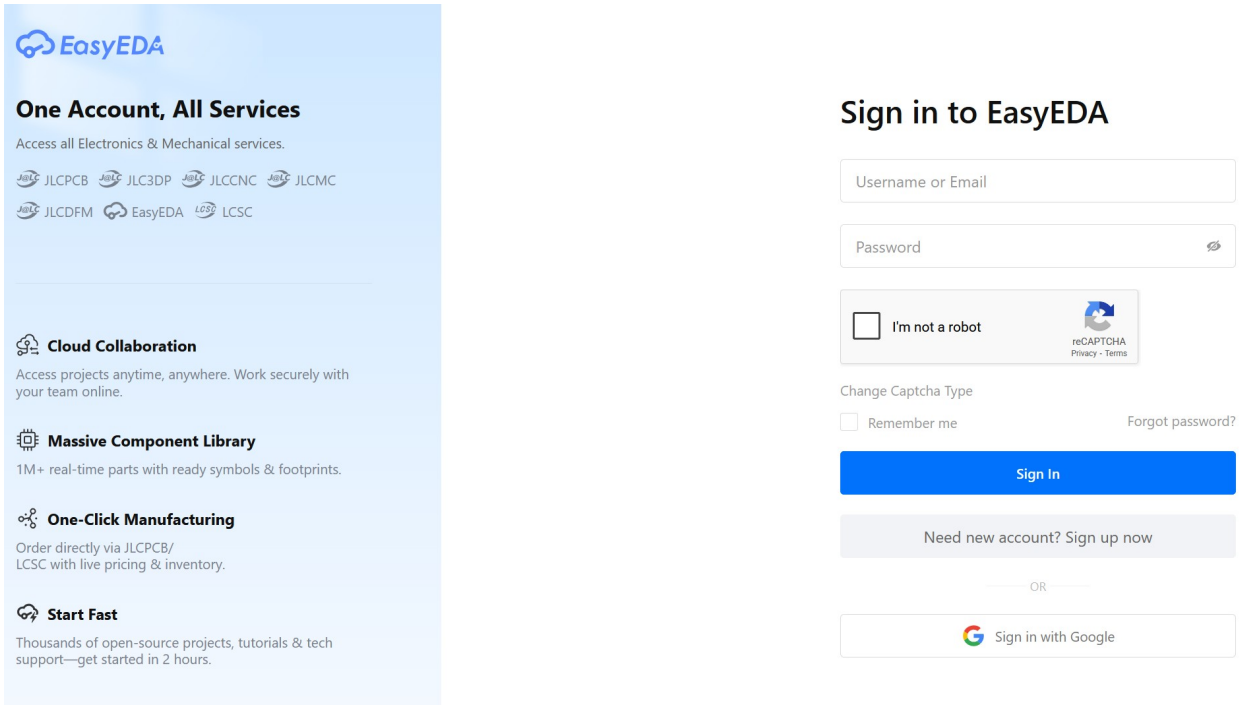
3.1 Изработване на принципна схема

За изработване на принципната схема има множество продукти. Altium Designer и Mentor Graphics са сравнително мощни среди, които се използват от множество фирми. Недостатъка им е, че са платени и изискват повече компютърен ресурс. Ако разполагате с някоя от двете, използвайте я без проблем. За целите на курсовото проектиране ще се спрем на две достъпни програми с отворен код – Kicad и EasyEDA.

EasyEDA може да се намери на https://easyeda.com/. Тя е онлайн базирана среда с възможност за изтегляне локално. Разполага със сравнително голяма компонентна база благодарение на своите потребители. За да я използвате отворете линка и изберете показаното на фиг. 4 поле в десния ъгъл.

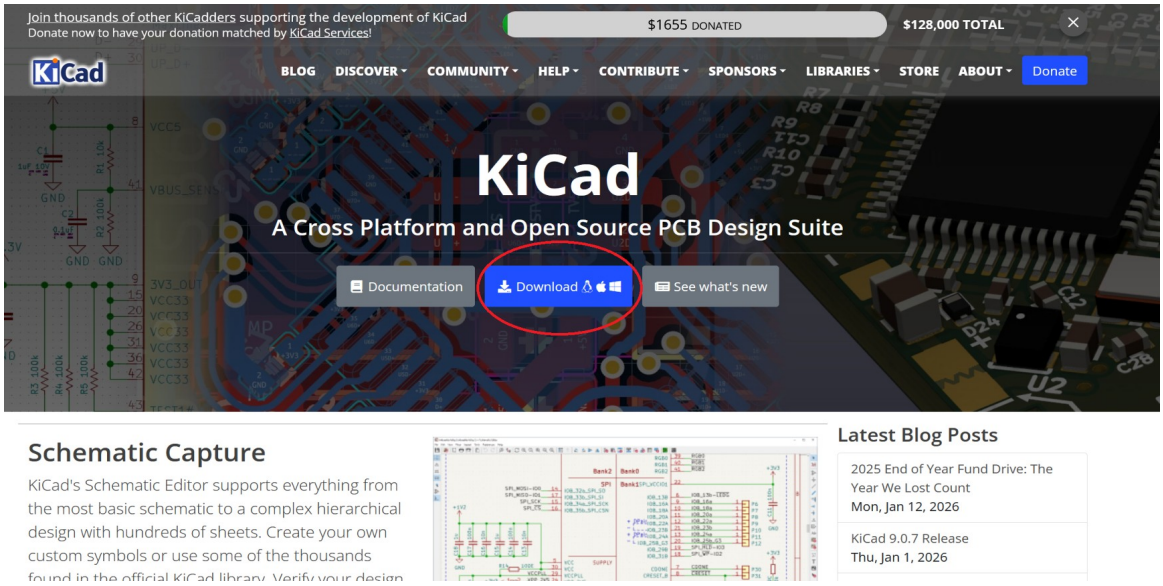
Фиг. 4: Начална страница на сайта

След това имате възможността да се регистрирате или да се впишете с Google акаунт (фиг. 5).



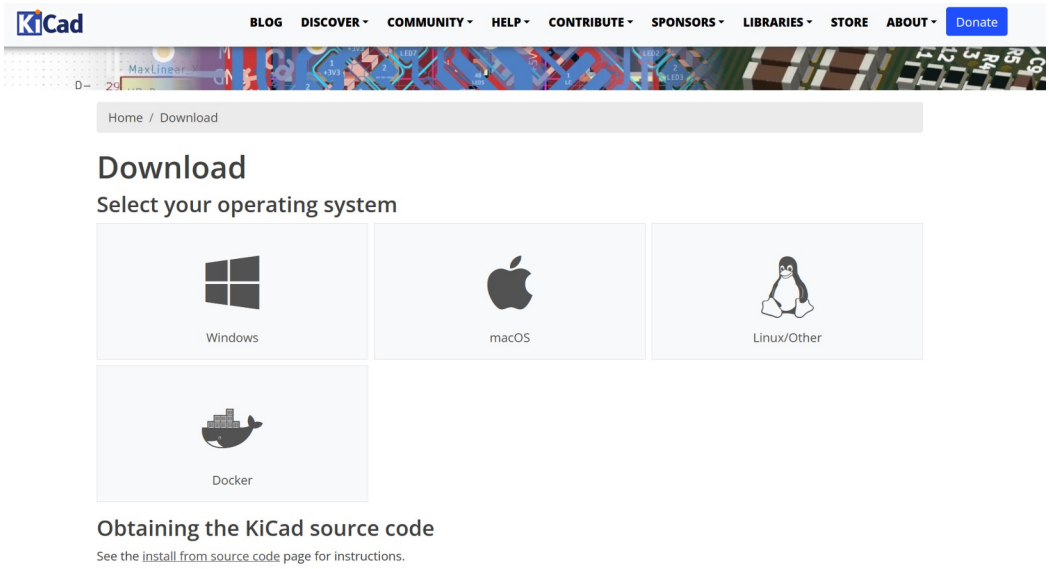
Фиг. 5: Изглед на страница за вписване

Kicad можете да намерите на https://www.kicad.org/. Тя изисква сваляне и инсталиране, но не натоварва значително компютъра. Въпреки, че не разполага с толкова обширна компонентна база както EasyEDA, необходимите компоненти могат да се изгенерират ръчно или да се свалят от https://www.snapeda.com. Когато отворите посочения сайт, директно ви се появява бутон за сваляне – фиг. 6.



Фиг. 6: Начален изглед на сайта

След това избирате версията за вашата операционна система (фиг. 7), сваляте и инсталирате.



Фиг. 7: Изглед на страница за сваляне на различни версии

Как точно се използват двете програми, можете да разберете на следните линкове:

https://www.youtube.com/watch?v=4Gtd7xY6zS4&list=PL3bNyZYHcRSUhUXUt51W6nKvxx2ORvUQB&index=3

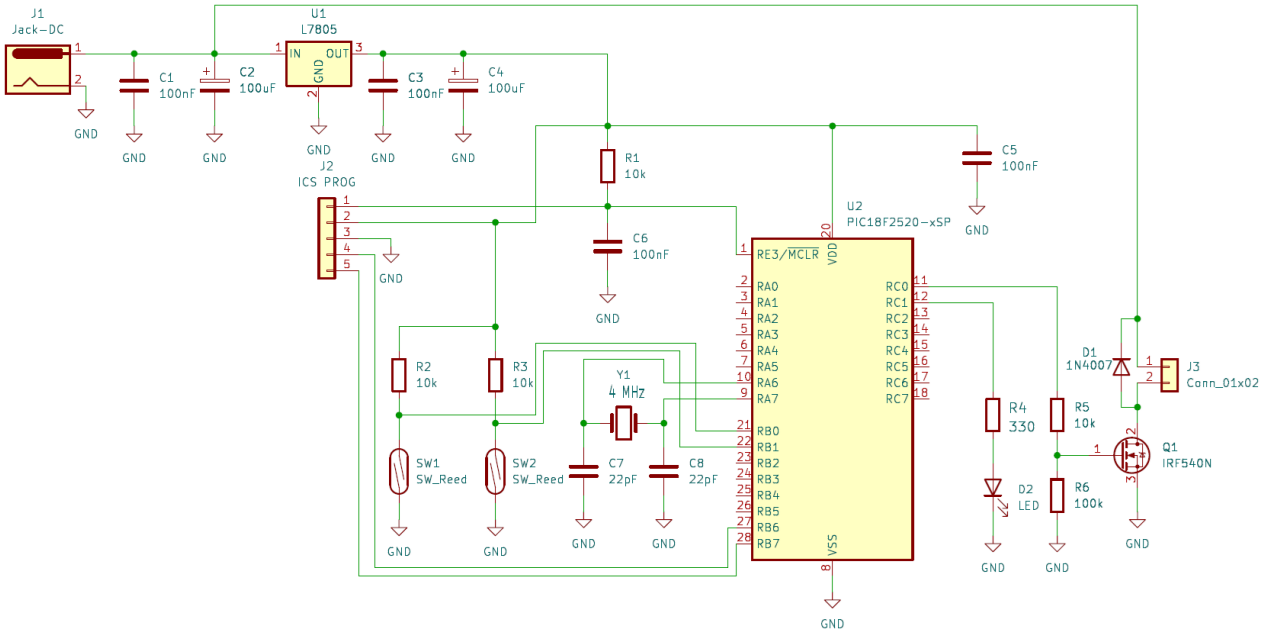
https://www.kicad.org/help/learning-resources/

https://www.youtube.com/watch?v=utBQqcuOt9U

https://docs.easyeda.com/en/Introduction/Introduction-to-EasyEDA/

ИЗПОЛЗВАНЕТО НА PAINT И ПОДОБНИ ПРОГРАМИ ЗА ИЗЧЕРТАВАНЕ НА ПРИНЦИПНИ СХЕМИ Е ЗАБРАНЕНО, СМЕШНО И НЯМА ДА СЕ ПРИЕМЕ!

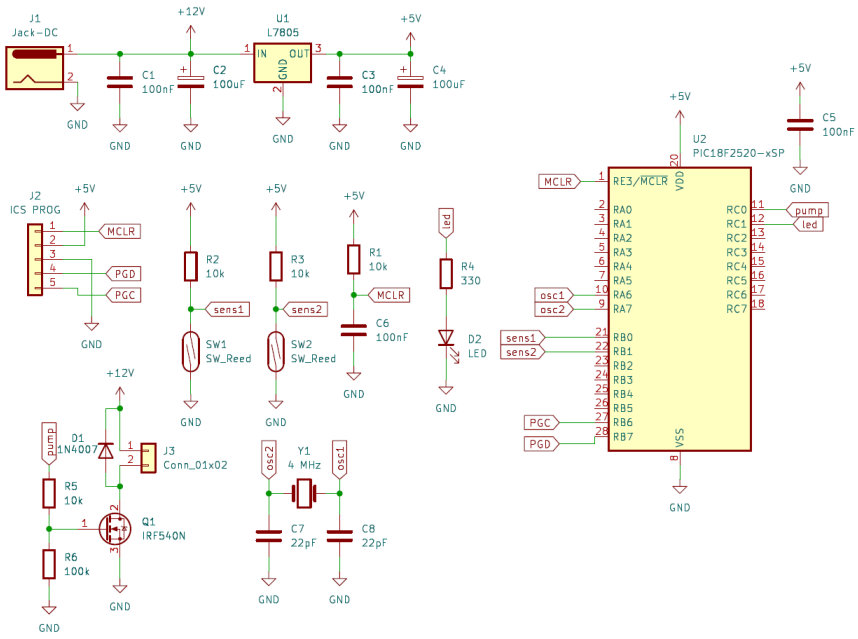
Нека чрез Kicad да начертаем нашата принципна схема – фиг. 8.



Фиг. 8: Принципна схема на реализираното устройство

Получената схема включва няколко допълнителни компоненти. Захранващата букса J1 служи за връзка с 12V адаптер. Кондензатори C1 до C4 са поставени по препоръка на производителя и за допълнително изглаждане на напрежението. Конектор J2 се използва за връзка с програматора – бърза и лесна смяна на кода. Кондензатор C5 е препоръчителен. Той трябва да се постави максимално близо да захранващия извод на микроконтролера, с което да доизглади високочестоните смущения. Кварца с кондензатори C7 и C8 изграждат тактовия генератор на микроконтролера. Групата R1-C6 се използва за начално установяване на микроконтролера. R1-D2 индикират, че водната помпа работи и резервоара се източва. Конектор J3 е за връзка с помпата. Диод D1 служи за предпазване на транзистора при комутация на помпата. Делителя R5-R6 служи за предотвратява задействането на транзистора при краткотрайни импулси, например при начална инициализация на микроконтролера.

Когато схемата е сравнително малка, тя може да се остави и в този вид, но е препоръчително използването на етикети („labels“). Тяхната роля е да заместят свързващите проводници на схемата и да позволят разместването на отделни компоненти. Нашата схема ще изглежда по начина показан фиг. 9 след използването на етикети.

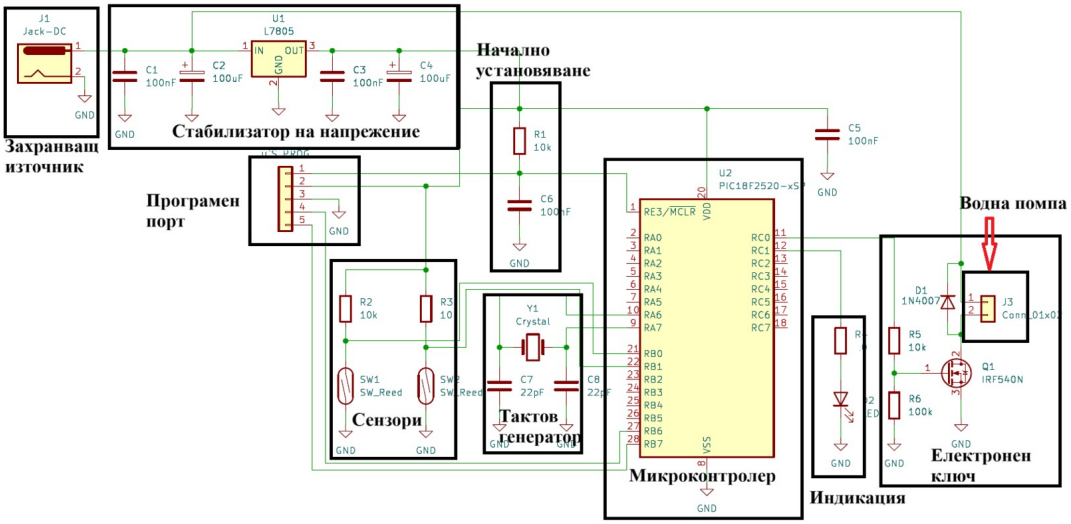


Фиг. 9: Принципна схема с използване на етикети

3.2 Изработване на блокова схема

Блоковата схема представлява опростено представяне на основните градивни блокове на схемата и тяхната взаимна връзка. Тя може да се изведе от готовата принципна схема или да се изготви предварително.

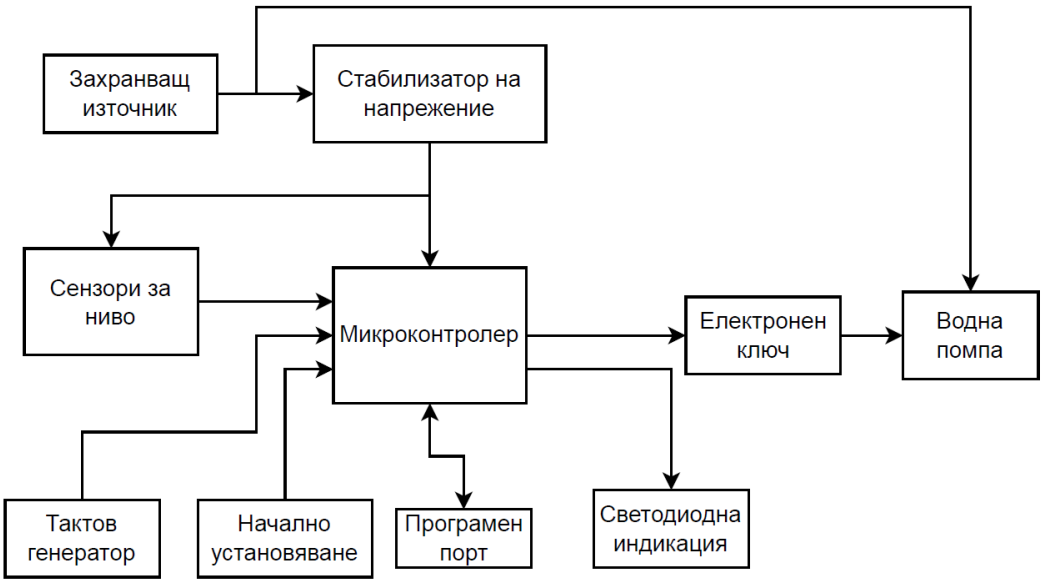
Нека вземем нашата принципна схема и оградим основните ѝ блокове – фиг. 10.



Фиг. 10: Основни блокове на устройството

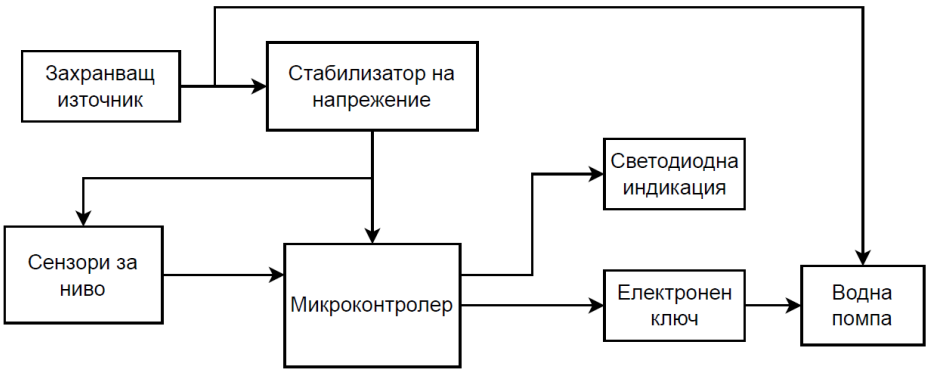
След като сме отделили основните блокове, можем да преминем към изчертаване на блоковата схема. За тази цел може да се използва https://app.diagrams.net/, който е онлайн базирана среда за изчертаване на различни блокови схеми или да се използват различните форми и стрелки от MS или Libreoffice пакета. Посоката на стрелките показва посоката на данни или сигнали. Те винаги трябва да са прави или ако отделните блокове са на различни нива, да са пречупени на 90°.

Ето как изглежда нашата блокова схема – фиг. 11.



Фиг. 11: Пълна блокова схема

Реално погледнато, блокове „Начално установяване“, „Програмен порт“ и „Тактов генератор“ могат да се вкарат в блок „Микроконтролер“ (фиг. 12). Това се дължи на факта, че те се реализират с прости схеми, но ако например тактовия генератор или началното установяване се изработват от специализирани интегрални схеми, то тогава те си остават отделни блокове.



Фиг. 12: Съкратена блокова схема

3.3 Написване на програмен код

За написване на програмния код за микроконтролера се използва средата, предлагана от производителя. Тя може да се свали от един от дадените линкове, в зависимост от избрания от вас микроконтролер.

За Microchip:
https://www.microchip.com/en-us/tools-resources/develop/mplab-x-ide

https://www.microchip.com/en-us/tools-resources/develop/mplab-xc-compilers

За STMicroelectronics:
https://www.st.com/en/development-tools/stm32cubeide.html

За NXP:
https://www.nxp.com/design/design-center/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE

За Espressif:
https://code.visualstudio.com/download

https://github.com/espressif/vscode-esp-idf-extension/blob/master/docs/tutorial/install.md

https://www.youtube.com/watch?v=XDDcS7HQNlI

За Raspberry Pi Foundation:
https://code.visualstudio.com/download

Нов вариант:
https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf

Стар вариант:
https://shawnhymel.com/2096/how-to-set-up-raspberry-pi-pico-c-c-toolchain-on-windows-with-vs-code/

https://www.youtube.com/watch?v=B5rQSoOmR5w&list=PLEBQazB0HUyQO6rJxKr2umPCgmfAU-cqR&index=2

За Arduino IDE:
https://www.arduino.cc/en/software

След инсталиране на избраната среда, направете нов проект като посочите правилния микроконтролер. Преди да започнем да пишем кода е добре да изясним какво точно ще извършва той.

В нашата примерна тема, микроконтролера трябва да следи сензор за ниво, показващ дали резервоара е пълен. В такъв случай, той трябва да пусне помпата и да изчака изпомпването на цялата вода, индикирано от втория сензор за ниво. Докато микроконтролера изчаква напълването на съда, той може да се вкара в режим на ниска консумация. Допълнително трябва да включва светодиод, който показва, че помпата работи. За да се изпълнят тези задачи, трябва да настроим два от изводите като нормални изходи, два извода като входове, като към един от тях прикачим прекъсване по спадащ фронт. Всичко това се извършва от следния код:

```
// PIC18F252 Configuration Bit Settings
// 'C' source line config statements
// CONFIG1H
#pragma config OSC = XT         // Oscillator Selection bits (XT oscillator)
#pragma config OSCS = OFF       // Oscillator System Clock Switch Enable bit (Oscillator system clock switch option is disabled (main oscillator is source))
// CONFIG2L
#pragma config PWRT = ON        // Power-up Timer Enable bit (PWRT enabled)
```

```
 #pragma config BOR = ON          // Brown-out Reset Enable bit (Brown-out Reset enabled)
 #pragma config BORV = 20         // Brown-out Reset Voltage bits (VBOR set to 2.0V)
 // CONFIG2H
 #pragma config WDT = OFF         // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))
 #pragma config WDTPS = 128       // Watchdog Timer Postscale Select bits (1:128)
 // CONFIG3H
 #pragma config CCP2MUX = ON      // CCP2 Mux bit (CCP2 input/output is multiplexed with RC1)
 // CONFIG4L
 #pragma config STVR = ON         // Stack Full/Underflow Reset Enable bit (Stack Full/Underflow will cause RESET)
 #pragma config LVP = OFF         // Low Voltage ICSP Enable bit (Low Voltage ICSP disabled)
 // CONFIG5L
 #pragma config CP0 = OFF         // Code Protection bit (Block 0 (000200-001FFFh) not code protected)
 #pragma config CP1 = OFF         // Code Protection bit (Block 1 (002000-003FFFh) not code protected)
 #pragma config CP2 = OFF         // Code Protection bit (Block 2 (004000-005FFFh) not code protected)
 #pragma config CP3 = OFF         // Code Protection bit (Block 3 (006000-007FFFh) not code protected)
 // CONFIG5H
 #pragma config CPB = OFF         // Boot Block Code Protection bit (Boot Block (000000-0001FFh) not code protected)
 #pragma config CPD = OFF         // Data EEPROM Code Protection bit (Data EEPROM not code protected)
 // CONFIG6L
 #pragma config WRT0 = OFF        // Write Protection bit (Block 0 (000200-001FFFh) not write protected)
 #pragma config WRT1 = OFF        // Write Protection bit (Block 1 (002000-003FFFh) not write protected)
 #pragma config WRT2 = OFF        // Write Protection bit (Block 2 (004000-005FFFh) not write protected)
```

```
 #pragma config WRT3 = OFF          // Write Protection bit (Block 3 (006000-007FFFh) not write protected)
 // CONFIG6H
 #pragma config WRTC = OFF         // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write protected)
 #pragma config WRTB = OFF         // Boot Block Write Protection bit (Boot Block (000000-0001FFh) not write protected)
 #pragma config WRTD = OFF         // Data EEPROM Write Protection bit (Data EEPROM not write protected)
 // CONFIG7L
 #pragma config EBTR0 = OFF        // Table Read Protection bit (Block 0 (000200-001FFFh) not protected from Table Reads executed in other blocks)
 #pragma config EBTR1 = OFF        // Table Read Protection bit (Block 1 (002000-003FFFh) not protected from Table Reads executed in other blocks)
 #pragma config EBTR2 = OFF        // Table Read Protection bit (Block 2 (004000-005FFFh) not protected from Table Reads executed in other blocks)
 #pragma config EBTR3 = OFF        // Table Read Protection bit (Block 3 (006000-007FFFh) not protected from Table Reads executed in other blocks)
 // CONFIG7H
 #pragma config EBTRB = OFF        // Boot Block Table Read Protection bit (Boot Block (000000-0001FFh) not protected from Table Reads executed in other blocks)
 // #pragma config statements should precede project file includes.
 // Use project enums instead of #define for ON and OFF.

 #include <xc.h>

 #define _XTAL_FREQ 4000000  //Задай 4MHz тактова честота

 #define LED_PIN LATCbits.LC1  //дефиниция за извод за светодиод
 #define PUMP_PIN LATCbits.LC0  //дефиниция за извод за помпа
 #define SENS1_PIN PORTBbits.RB0  //дефиниция за извод за горен сензор
 #define SENS2_PIN PORTBbits.RB1  //дефиниция за сензор на дъното

 void sys_init (void){   //функция за инициализация на микроконтролера
    TRISA = 0x00;  //превключи всички изводи на PORTA в изходи
    TRISB = 0x03;  //превключи изводи 0 и 1 на PORTB във входове
    TRISC = 0x00;  //превключи всички изводи на PORTC в изходи
```

```
    LED_PIN = 0;  //задай ниско логическо ниво
    PUMP_PIN = 0;  //задай ниско логическо ниво

    INTCONbits.INT0IE = 1;  //позволи прекъсване за извод RB0
    INTCONbits.INT0IF = 0;  //нулирай флага за прекъсване
    INTCON2bits.INTEDG0 = 0;//задай спадащ фронт за прекъсване
    RCONbits.IPEN = 0;  //изключи приоритет на прекъсванията
        INTCONbits.PEIE = 1;        //позволи всички не маскирани периферни прекъсвания
     INTCONbits.GIE = 0; //не прави преход към подпрограма за обслужване на прекъсвания след изход от режим на ниска консумация

    return;
 }

 void pump_ctrl (void){  //функция за управление на помпата
    LED_PIN = 1;  //задай високо логическо ниво
    PUMP_PIN = 1;  //задай високо логическо ниво

    while(SENS2_PIN == 1){ //изчакай изпразването на резервоара
       __delay_ms(100);
    }

    LED_PIN = 0;  //задай ниско логическо ниво
    PUMP_PIN = 0;  //задай ниско логическо ниво

    return;
 }

 void main(void) {   //главна функция
    sys_init(); //инициализирай микроконтролера

    while(1){
       if(SENS2_PIN == 1){ //провери дали в резервоара има течност
          pump_ctrl();
       }
       SLEEP();    //влез в режим на ниска консумация
```

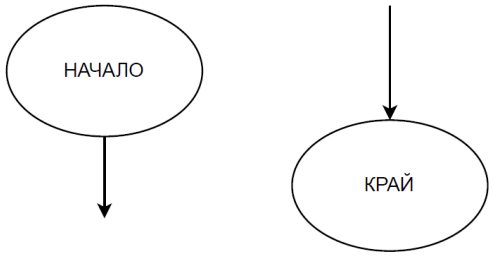
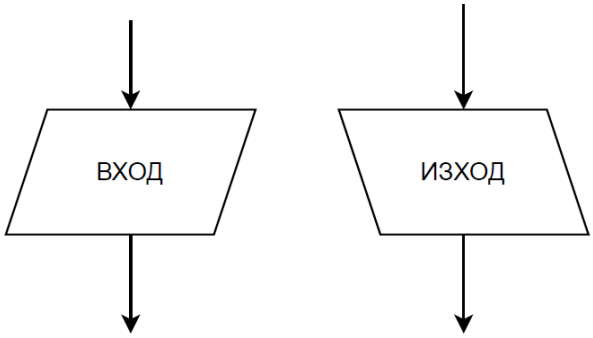
```
        INTCONbits.INT0IF = 0;  //нулирай флага за прекъсване
    }

    return;
}
```

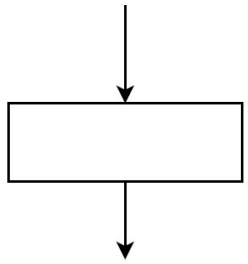
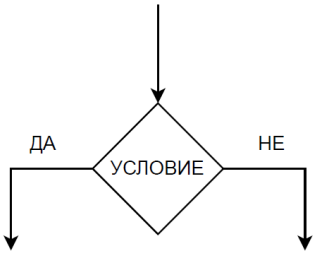
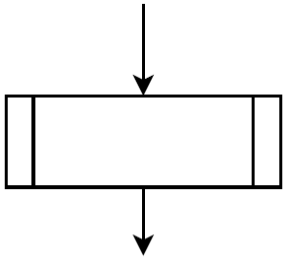
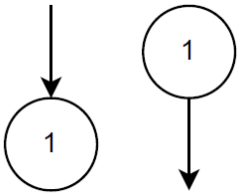
Добра практика е добавянето на коментари към кода. По този начин, той става по-добре разбираем за други разработчици, а и помага на самите вас да си спомните какво и как сте правили.

3.4 Изработване на блок-алгоритъм на кода

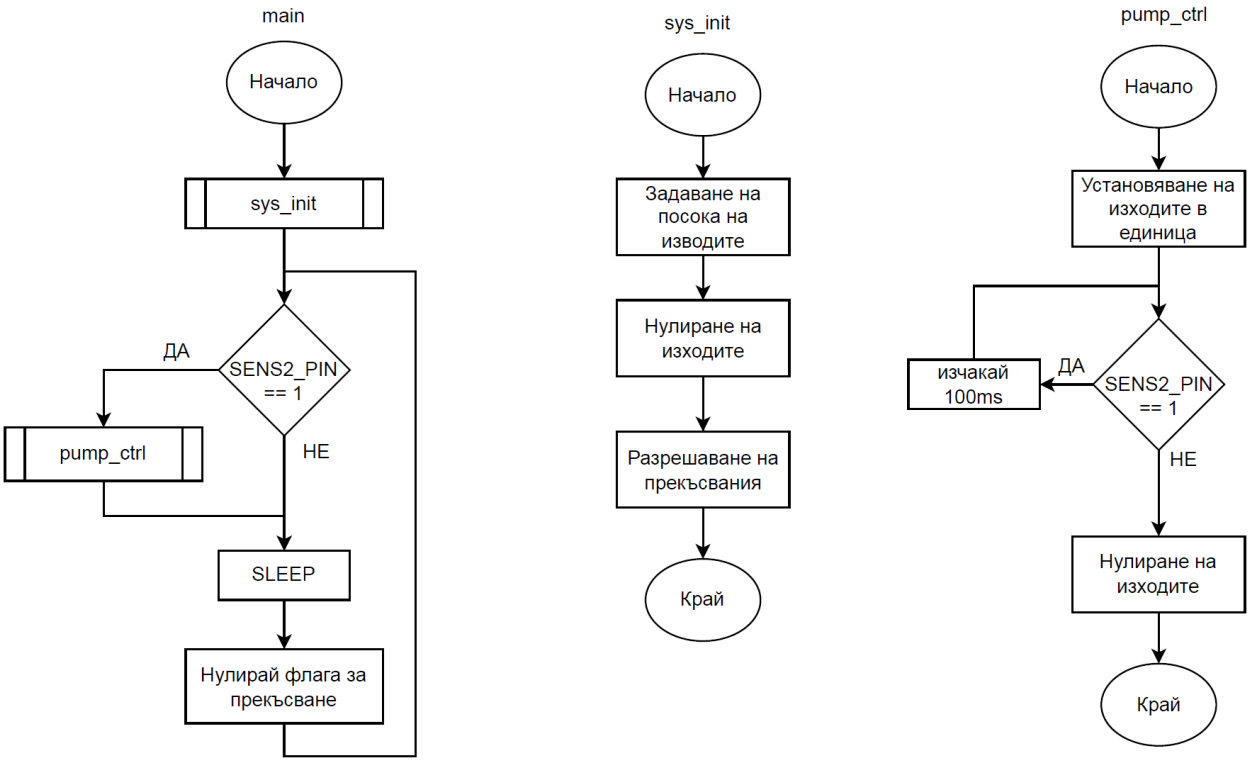
Блок-алгоритъма представлява графично изобразяване на работата на написания код. За тази цел се използват специални блокове, показни в таблица 1.

Таблица 1: Основни означения за изчертаване на блок-алгоритъм

	За начало – определя мястото, от където стартира изпълнението на кода; от него излиза само една стрелка. За край – определя къде приключва изпълнението на кода; в него влизат произволен брой стрелки.
	Блок за вход – описват се входни данни за кода. Блок за изход – извеждане на крайни и междинни резултати от работата на алгоритъма.

	Блок за обработка – описват се произволен брой безусловни операции.
	Условен блок – задава необходимото логическо условие; ако то е вряно, кода продължава изпълнението си по клон „да“, в противен случай се поема по клон „не“
	Блок за извикване на подпрограма (функция) – използва се когато кода се обръща към написана от нас функция.
	Междустраничен съединител – използват се когато блок-алгоритъма е сравнително голям и трябва да се изчертае на няколко страници.

След като знаем какви блокове сме използвали в кода, нека изчертаем нашия блок-алгоритъм – фиг. 13



Фиг. 13: Блок-алгоритъм на изготвения код

Обърнете внимание, че така написания код е в безкраен цикъл и поради това липсва блока „Край“. Това не е най-добрата практика! Препоръчително е да се постави условие, което да прекрати изпълнението на кода и с това да рестартира микроконтролера.

3.5 Описание на програмния код

В тази част се прави описание на работата на целия код и детайлно описание на използваните функции. Може да се използват само имената на функциите или да се изпишат цялостно прототипите им.

Ето описанието на нашия код:

void sys_init (void) – функцията прави начална инициализация на необходимата периферия на микроконтролера. Първо се посочва посоката на изводите – RB0

и RB1 се конфигурират като входове, а RC0 и RC1 – като изходи. След това изводите се нулират – гарантира се, че помпата е изключена. Разрешават се прекъсвания по спадащ фронт на RB0 и се разрешават всички немаскирани периферни прекъсвания. За да не се прави преход към подпрограма за обслужване на прекъсвания след изхода от режим на ниска консумация, бит GIE се нулира.

void pump_ctrl (void) – функцията реализира управление на водната помпа. При извикването ѝ, тя включва помпата и светва светодиода. Започва да следи състоянието на сензора на дъното на всеки 100ms. Когато състоянието на сензора стане логическа нула, то резервоара е празен и помпата и светодиода се изключват.

void main(void) – основната функция на кода. Първо извиква функцията за инициализация на микроконтролера, след това преминава към проверка на състоянието на сензорите. Ако сензора на дъното е в логическа единица се извиква функция за източване на резервоара, което гарантира започване от изцяло празно състояние. След това, микроконтролера се вкарва в режим на ниска консумация и се изчаква напълването на резервоара. Процеса се завърта до спиране на захранването.

4. Заключение

В тази глава се прави кратко обобщение на извършената работа – в рамките на 2 или 3 абзаца. Уточняват се срещнатите проблеми при разработването на устройството и неговата евентуална реализация. Прави се цялостна оценка на устройството и се дават примери за за подобряване на дизайна – схемотехника и код.

Помощна информация

Магазини от където бихте могли да закупите микроконтролер и сензори/модули/периферия:

https://elimex.bg/

https://elimex.bg/category/hobby-electronics-atmel

https://store.comet.bg/Catalogue/

https://www.olimex.com/Products/

https://radev96.com/

https://erelement.com/

https://www.robotev.com/index.php

Помощна литература:

https://store.comet.bg/Catalogue/Product/6798/

https://store.comet.bg/Catalogue/Product/25310/

https://store.comet.bg/Catalogue/Product/16008/

https://store.comet.bg/download-file.php?id=1596

https://storage.composity.com/files/b47a2a73

https://www.jameco.com/Jameco/workshop/Howitworks/how-servo-motors-work.html

https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/

https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/

https://lastminuteengineers.com/433mhz-rf-wireless-arduino-tutorial/

https://www.youtube.com/watch?v=cUNb_XtwIZU

https://www.youtube.com/@greatscottlab/videos

https://www.youtube.com/@HowToMechatronics/videos

https://www.youtube.com/@ELECTRONOOBS/videos

https://www.youtube.com/@SineLab/videos

Критерии за оценяване

За Среден 3:

Спазване на заложените изисквания и структура на курсовия проект, задоволително изпълнение на глави 1, 2, 4, минимално изпълнение на глави 3.1, 3.2, 3.3 и защита на курсовия проект

За Добър 4:

Спазване на заложените изисквания и структура на курсовия проект, задоволително изпълнение на глави 1, 2, 3.1, 3.2, 3.3, 3.4, 4 и защита на курсовия проект

За Много добър 5 и Отличен 6:

Спазване на заложените изисквания и структура на курсовия проект, задоволително изпълнение на глави 1, 2, 3, 4, съответствие с работата по време на лабораторните упражнения и защита на курсовия проект

При съмнение за плагиатство / предаване на чужда работа и/или нерегламентирано използване на изкуствен интелект, предадената работа ще бъде върната!