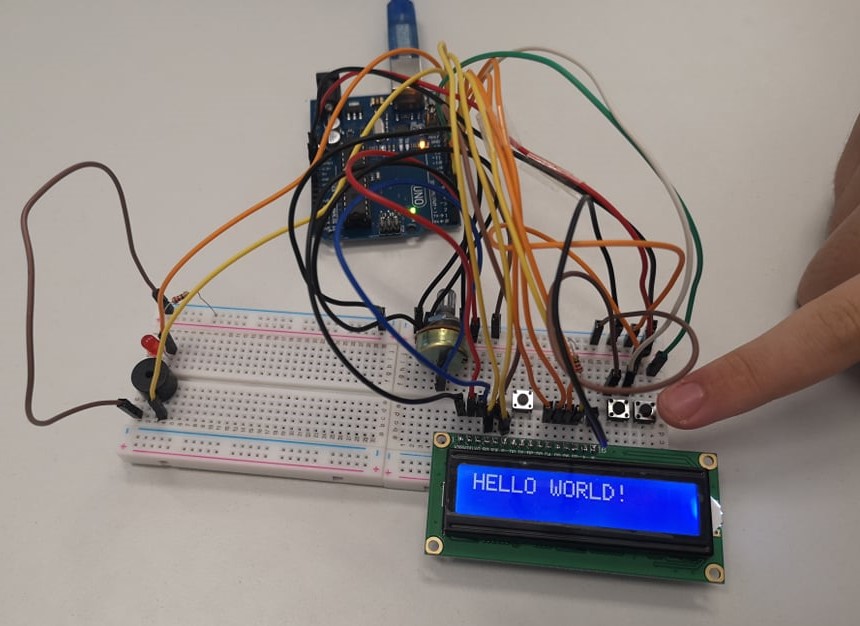
**Морзов код с Arduino**

Преводач и инструктор за Морзов код

Изготвили:

Димитър Казаков, Димитър Дюлгеров, Михаил Буков, Николай Миладинов



Съдържание:

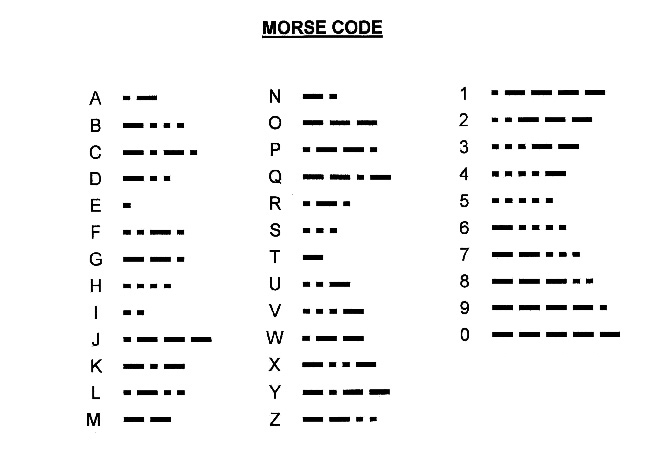
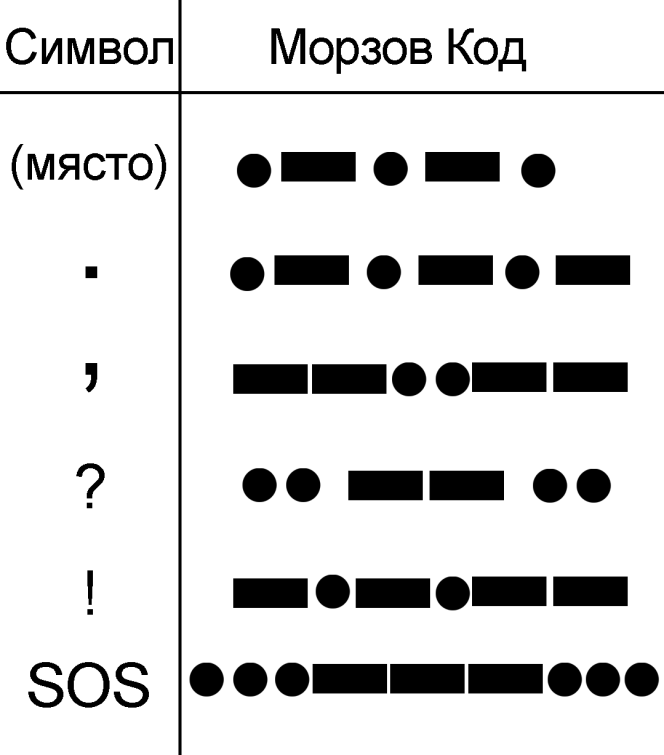
* Описание на проекта
* Съставни части
* Блокова схема
* Електрическа схема
* Сглобяване
* Програмиране и сорс код
* Заключение и бъдещи планове

Описание на проекта

Идеята на проекта е да направи превеждането и научаването на Морзов код лесно и забавно. Използването му е ограничено, но все пак е нещо полезно, което може да послужи на всекиго в определени условия, когато това е единственият начин за комуникация. Едно от големите предимства на Морзовия код е, че пренасянето му е значително по просто и лесно и следователно изисква по-малко сложна екипировка. Честотната му лента също е много по-малка: 100 - 150 Hz в сравнение с около 2400 Hz на гласовото предаване.

Главните функционалности на преводача са следните:

* Превеждане от Морзов код към английски букви, числа и специални символи (разпознаваемите символи и знаци са описани по-долу);



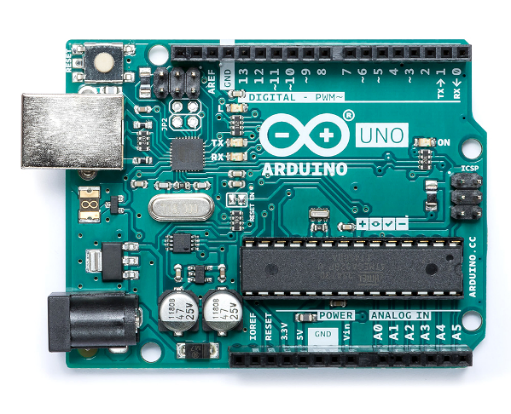
* Режим на учене на азбуката, който демонстрира всяка буква по ред, чрез светлина и звук, съответно от червено led и piezo buzzer;
* Режим на учене на прости думи, буква по буква.

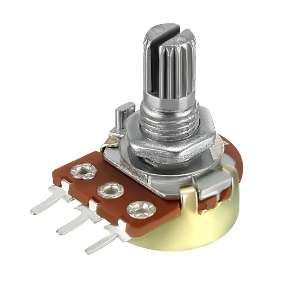
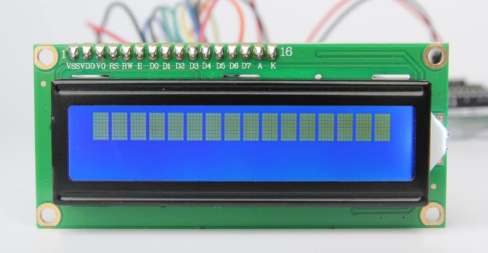
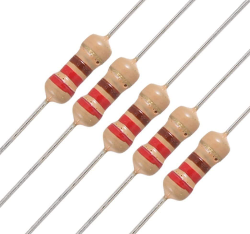
Преводачът е приложим за лица от всякаква възрастова група, най-вече подрастващи.

Съставни части

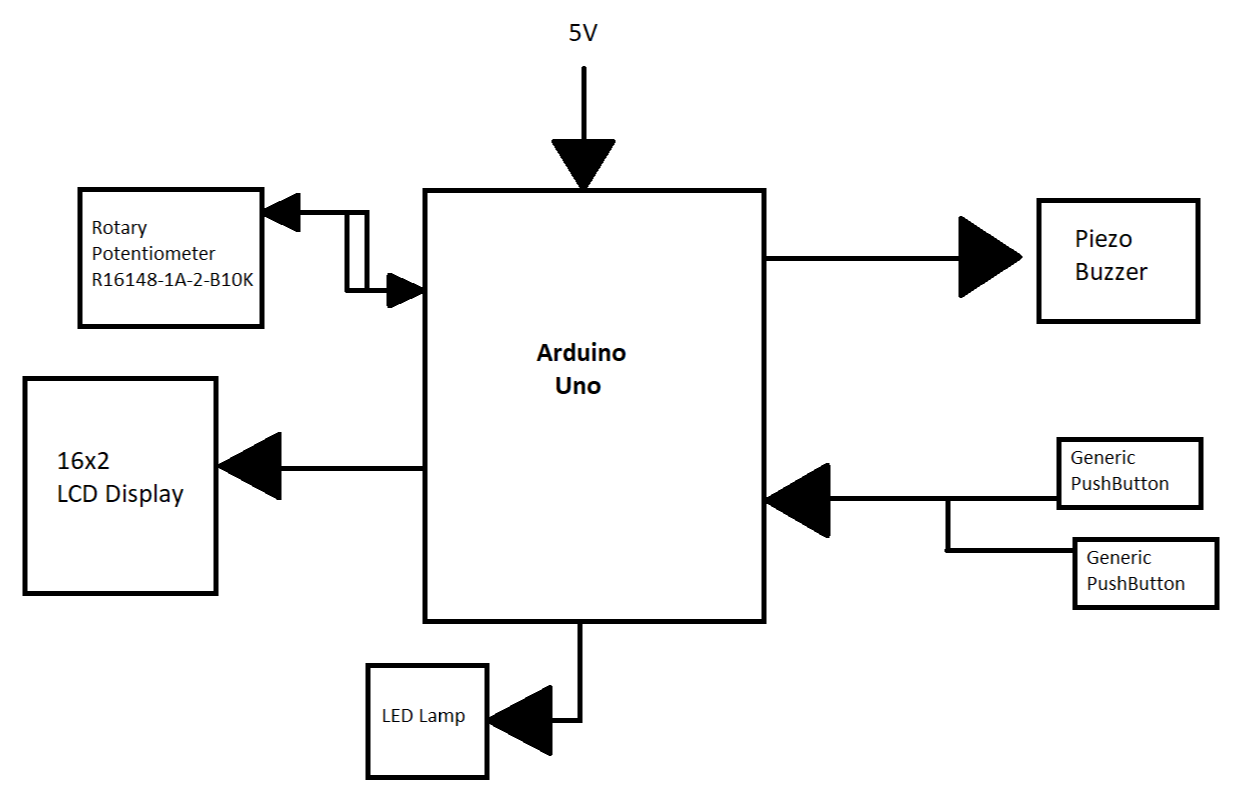
В проекта са използвани следните части:

* Arduino UNO Rev3 [1]
* Piezo Buzzer [2]
* Червено LED [3]
* Pushbutton x2 [4]
* LCD Display [5]
* Въртящ се потенциометър [6]
* 220 Ω транзистор x2 [7]
* Свързващи кабели [8]

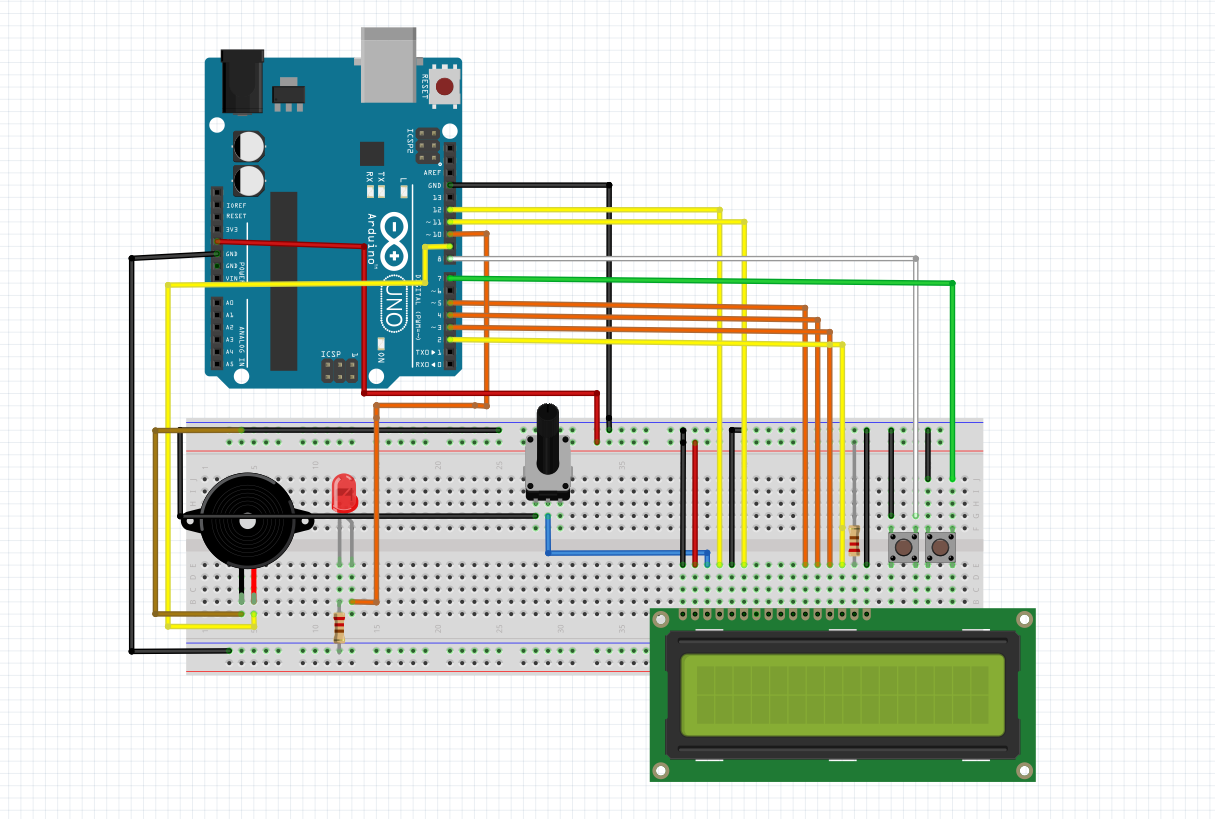
[1] [2] [3] [4]

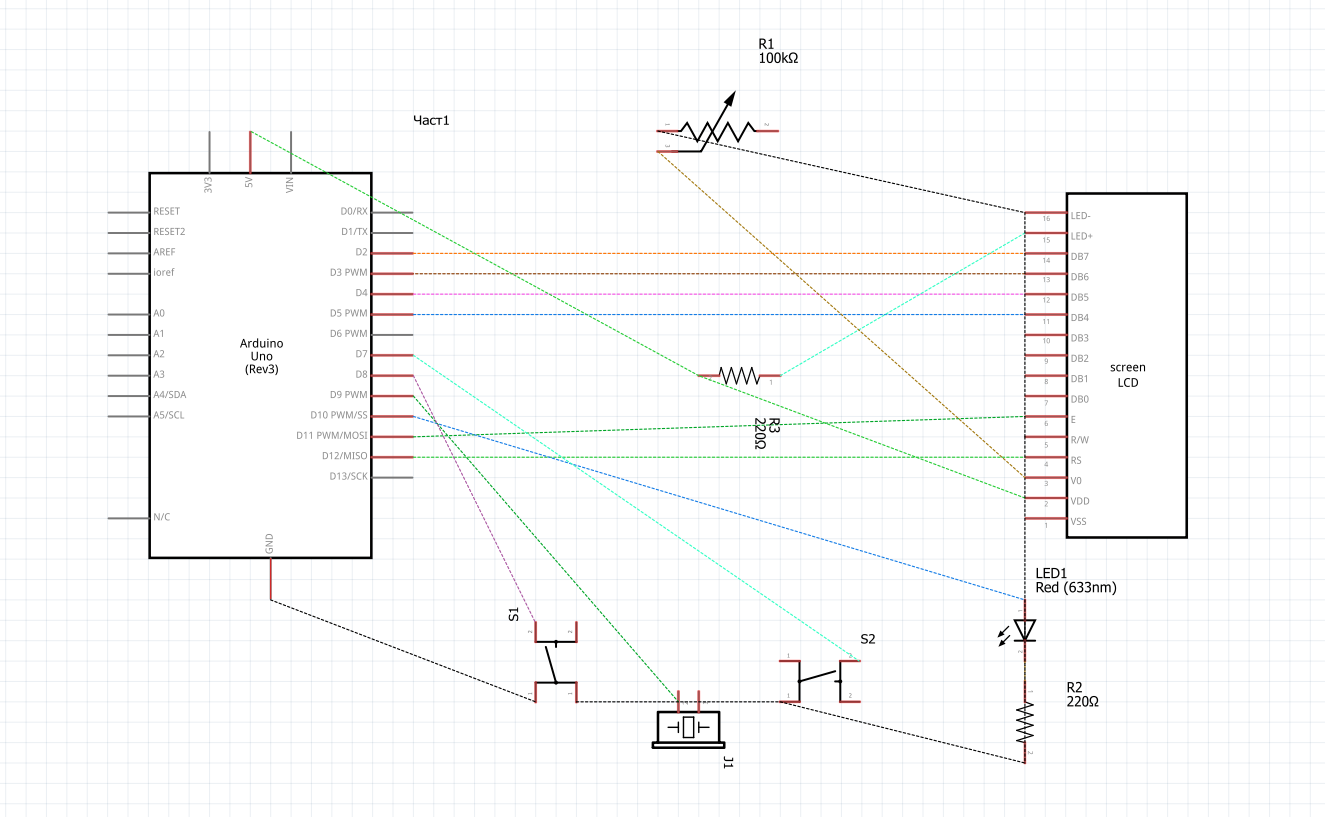
[5] [6] [7] [8]

Блокова схема



Електрическа схема





Програмиране и сорс код

Главната функционалност на преводача се извършва от кода. Идеята е, че чрез вход, взет от разположения най-вдясно бутон, се приема точка или тире, в зависимост от продължителността на задържането на бутона. След това, кодът бива преведен, с помощта на прост алгоритъм и масиви със предефинирани стойности и бива изписан на LCD дисплей, както и на Serial monitor-а.

Начинът на работа на двете обучителни игри е сходен – програмата превежда дадена дума в морзов код и сравнява входът на потребителя с очаквания. Ако е правилен, програмата продължава, ако ли не – буквата/думата се повтаря, докато не бъде въведена правилно.

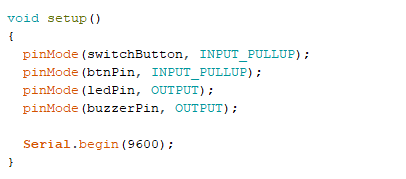
Потребителят може да преминава от един начин на действие (Translator -> Learn the alphabet game -> Learn words game) към друг, като натиска втория бутон, намиращ се отляво на този за въвеждане.

Това е общата концепция на кода.

Сорс кодът може да намерите на:

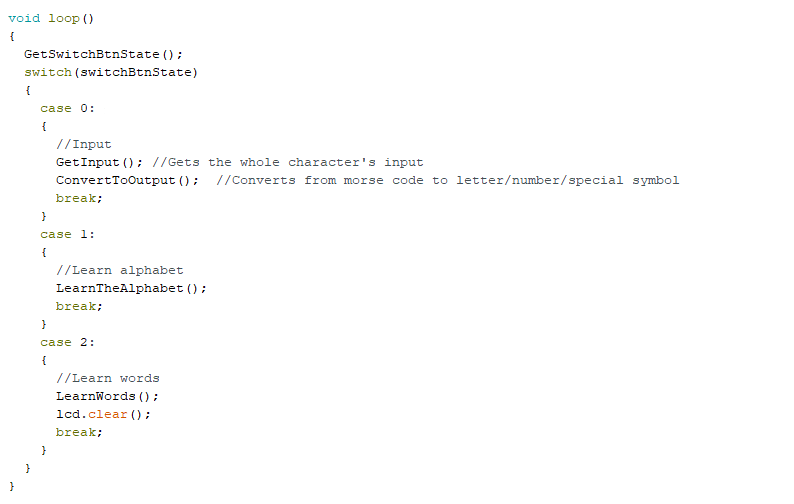
<https://github.com/the-crazyguy/Morse-Code-With-Arduino/blob/master/Morse_Code_V6.ino>

В началото се дефинират глобалните променливи и масиви:

В setup() функцията се дефинират главните пинове и тяхната функционалност:

Забележка: бутоните са зададени с INPUT\_PULLUP, тоест се използват вградените резистори в Arduino, което също така прави така, че бутонът се счита за натиснат при digitalRead(btnPIN) == LOW, a ненатиснат при digitalRead(btnPin) == HIGH.

В loop() функцията, както името подсказва, изпълняваща се многократно, се проверява статусът на switchBtnState чрез switch() и се определя коя част от програмата трябва да се изпълни:

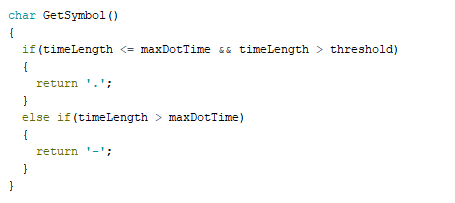


В случай, че бутонът за превключване на режим е в state = 0, и по подразбиране при първо пускане, се изпълнява функцията за превеждане на Морзов код към английската азбука, числа и знаци.

Функциите **GetInput()** и **GetDotDash()**:



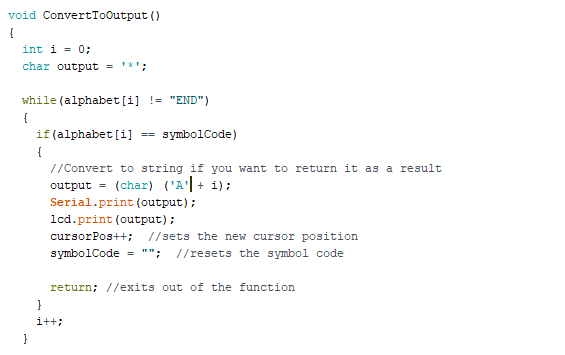
Ролята на GetDotDash() е да разбере за колко време е натиснат бутона и да запише тази информация в timeLength, като изважда времето, когато е натиснат бутона от времето, когато е отпуснат. След това записва точка или тире в **symbolCode**. Това става с помощта на **GetSymbol()** функцията:



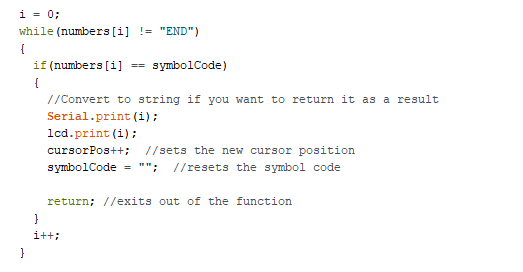
Нейната роля е да провери дали времето на задържане на бутона съответства с точка или с тире. Ролята на **threshold** променливата е да елиминира неточности, ако бутонът, по някаква причина, бъде натиснат погрешка за < 50 ms (смущения в сигнала, etc.).

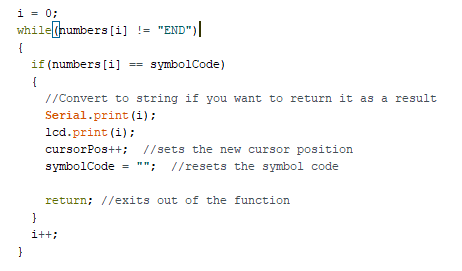
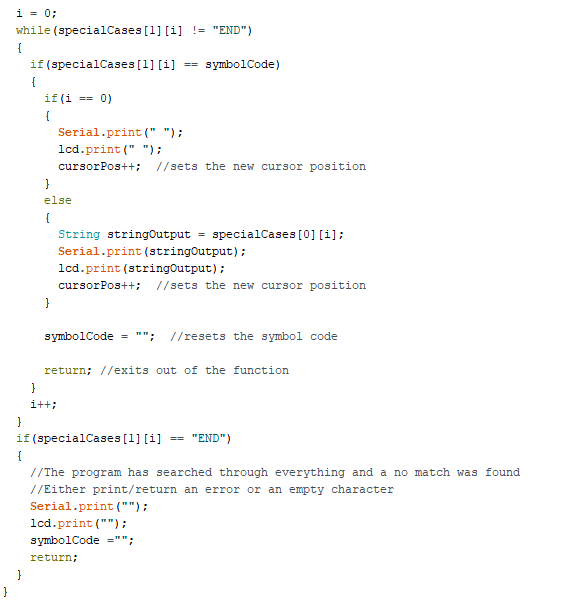
GetDotDash() се извиква повторно, докато не измине определено време (**timeBetweenLetters)** от последното отпускане на бутона. Когато това стане се приема, че въвеждането на символи е приключило и програмата отива в **ConvertToOutput()**, която може да се раздели на 3 основни части: превод на буква, превод на число и превод на специален знак.

Нека да разгледаме превода на буква:



Тук имаме брояч **i**, служещ за обхождане на масивите, **output** за запазване на получения знак.Чрез while() цикъл се обхожда alphabet[], докато не бъде намерен търсения знак. Ако това стане, програмата го запазва в output, принтира го на Serial Monitor-а и на LED екрана, премествайки курсора му с един знак напред, и най-накрая занулява symbolCode, за да може да мжое се използва отново. Най-накрая, чрез return, излизаме от функцията, тъй като сме намерили това, което търсим.

В случай, че не сме, функцията продължава да търси, но този път в масива с числа numbers[], по абсолютно аналогичен начин:

За търсене на специални символи се използва двумерен масив (матрица), съставена от 2 реда. На ред с индекс 0 се намират специалните знаци и символи, а на реда с индекс 1 – тяхната форма в Морзов код. Останалата част е аналогична.

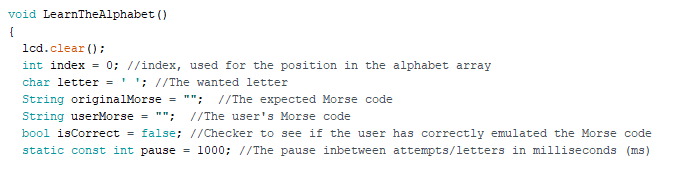
Ако не бъде намерено съответствие и в масива със специални знаци се приема, че не съществува знак със зададения от потребителя Морзов код и не се отпечатва нищо.

Функцията **PlayMorseCode(String morseCode)**:



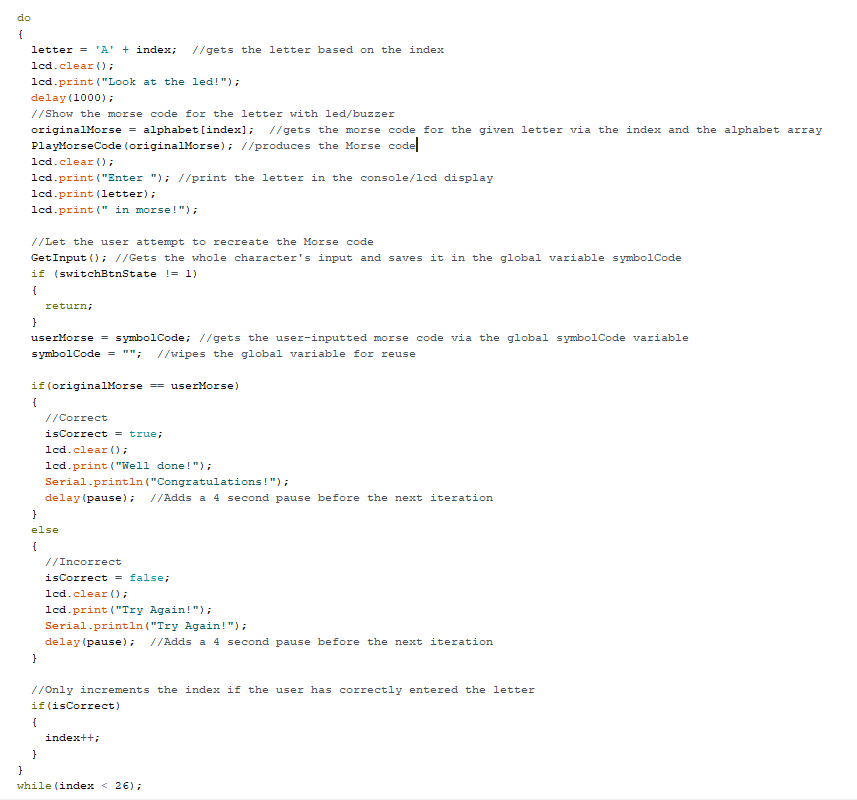
Нейното предназначение е да „покаже“ морзовия код, зададен като параметър на функцията, чрез led и piezo buzzer. взема дължината на променливата от тип String с .length() и завърта for-цикъла толкова пъти, колкото знаци има. За всеки знак проверява дали е точка или тире и съответно включва led и piezo buzzer-а за съответното време.

Функцията **LearnTheAlphabet()**:



В началото се дефинират основните променливи, които ще са ни необходими за реализирането на функцията, описание на които може да се види от предоставените коментари.

Тук започва истинската функционалност както на функцията, така и на режимът за учене на азбуката:



В началото, на променливата **letter** се задава стойност, базирана на мястото, на което сме в азбуката, започвайки от 0. Това се случва по следната формула: **letter = ‘A’ + index;** Понеже letter е от тип char, можем да използваме факта, че буквите от азбуката са подредени последователно в ASCII таблицата, следователно добавяйки **index** към ‘A’ (първата буква от азбуката), можем да получим търсената. Например ‘A’ + 3 = ‘D’. След като знаем търсената буква, взимаме морзовия ѝ код от масива с азбуката и го запазваме в променливата, с която ще проверяваме въведения от потребителя input. Чрез функцията PlayMorseCode(String morseCode) се показва изисквания input. Променливата index се инкрементира всеки път, когато потребителят въведе морзовия код за показаната буква правилно. Има пауза от 4 секунди преди или следващата буква да се покаже, или да се направи повторна демонстрация и въвеждане. Функцията приключва или при преминаване на цялата азбука, или при сменяне на режима, чрез съответния бутон.

Заключение и бъдещи планове

В заключение, идеята на проекта ни е да направи ученето на Морзов код лесно и забавно. Наличието на начин за учене по интерактивен начин прави процеса много по приятен и интуитивен, давайки истинска демонстрация на това как Морзовия код би се използвал.

* Добавяне на режим за обучение на числа и символи.
* Добавяне на бутон за изтриване на последния въведен знак (буква, число, знак)