ТУЕС към ТУ-София

Компютърни архитектури

Курсова работа

“Мини конзола” с Arduino Uno

Изготвил: Костадин Костадинов XIг клас, випуск 2020г.

София, 27.04.2019

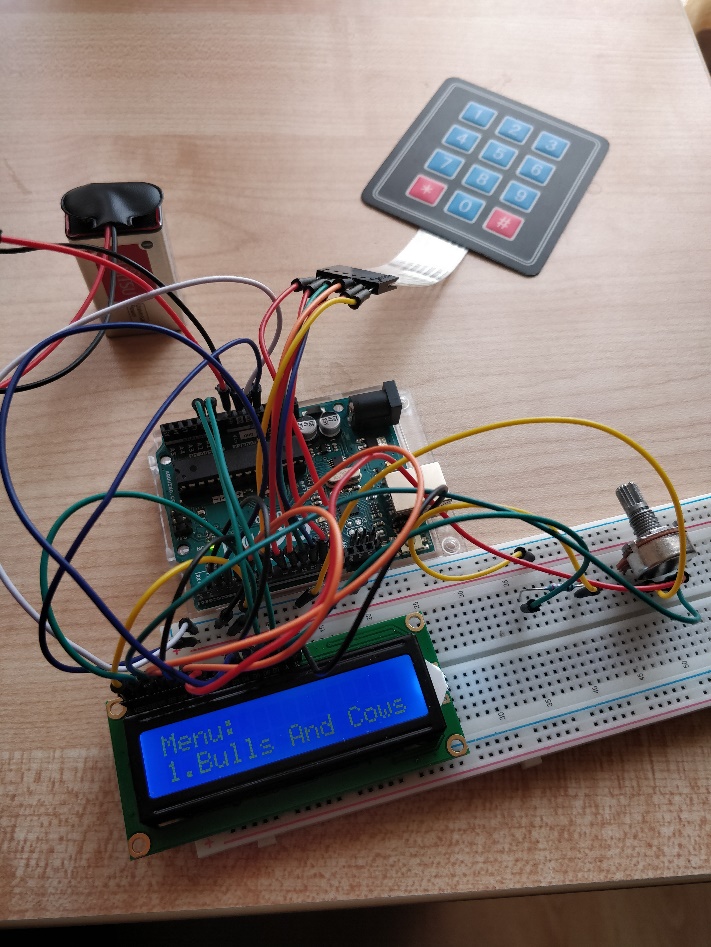
**Съдържание**

1. **Идея на проекта**
2. **Ресурси**
   1. Използвани ел.компоненти
   2. Използван софтуер, езици за програмиране и библиотеки
3. **Хардуер**
   1. Принципна схема на устройството
   2. Графичен оригинал на печатната платка
   3. Устройство и работа
4. **Софтуер**
   1. Блок схемa
   2. UML диаграми
   3. Документация
5. **Идея на проекта**

Проектът представлява мини конзола разработена чрез Arduino Uno.

Конзолата съдържа две игри, стандартна версия на известната игра „Бикове и крави“ и играта „Динозавър“, малко по-различна версия от играта за Chrome браузерите. Възможно е да се пишат игри и да се вкарват лесно в конзолата без голяма модификация на сорс код.

Управлението се извършва чрез един пад с 12 бутона. Конзолата се захранва от 9V батерия, а цялата интеракция между юзера и конзолата е посредством 16x2 LCD дисплей. В схемата има и един потенциометър за регулиране на яркоста на дисплея, както и един 330 ом-ов резистор за да предпазва диода, който служи за подсветка на дисплея, от прекалено високо напрежение.



**2.Ресурси**

2.1 Използвани ел.компоненти

Използваните електронни компоненти са:

* [Arduino Uno](https://erelement.com/arduino-uno)
* [16x2 LCD Display](https://erelement.com/displays/lcd-1602-blue-i2c?zenid=hhedp19neo84obbs1gtaka8fi4)
* [4x3 Keypad](https://www.aliexpress.com/item/50PCS-New-4-4-Matrix-Array-Matrix-Keyboard-16-Key-Membrane-Switch-Keypad-for-arduino/1807645184.html?spm=2114.search0104.3.9.2bbc3561z1M07n&ws_ab_test=searchweb0_0,searchweb201602_1_10065_10068_319_10059_10884_317_10887_10696_321_322_10084_453_10083_454_10103_10618_10304_10307_10820_10821_537_10302_536,searchweb201603_50,ppcSwitch_0&algo_expid=a342a724-a6a1-43da-bc05-bc951c2e43a9-1&algo_pvid=a342a724-a6a1-43da-bc05-bc951c2e43a9)
* [9V Battery](https://www.aliexpress.com/item/1PCS-100-Original-Panasonic-Greencell-PP3-6F22-6LR61-MN1604-9V-Block-Heavy-Duty-Cell-Battery/32949796712.html?spm=2114.search0104.3.15.6700582flk0Cqa&ws_ab_test=searchweb0_0,searchweb201602_1_10065_10068_319_10059_10884_317_10887_10696_321_322_10084_453_10083_454_10103_10618_10304_10307_10820_10821_537_10302_536,searchweb201603_50,ppcSwitch_0&algo_expid=1d93c657-370c-4532-abf9-cee21b421d15-2&algo_pvid=1d93c657-370c-4532-abf9-cee21b421d15)
* [10kΩ потенциометър](https://www.aliexpress.com/item/RV24YN20S-B103-RV24YN-2W-10K-OHM-Single-Turn-Rotary-Carbon-Potentiometer-x-10PCS/32804026446.html?spm=2114.search0104.3.78.aac4537e3xdrXJ&s=p&ws_ab_test=searchweb0_0,searchweb201602_1_10065_10068_319_10059_10884_317_10887_10696_321_322_10084_453_10083_454_10103_10618_10304_10307_10820_10821_537_10302_536,searchweb201603_50,ppcSwitch_0&algo_expid=500a9c2d-f9a6-4019-9f3d-34d15b270b9d-11&algo_pvid=500a9c2d-f9a6-4019-9f3d-34d15b270b9d)
* [330Ω резистор](https://www.aliexpress.com/item/20pcs-2W-Metal-Film-Resistor-2W-1-0R-2-2M-0-2-2-10-100-120/32958363818.html?spm=2114.search0104.3.16.2490312f2CFb6R&ws_ab_test=searchweb0_0,searchweb201602_1_10065_10068_319_10059_10884_317_10887_10696_321_322_10084_453_10083_454_10103_10618_10304_10307_10820_10821_537_10302_536,searchweb201603_50,ppcSwitch_0&algo_expid=7138974d-1ff1-4137-aed4-bd8d375912b5-2&algo_pvid=7138974d-1ff1-4137-aed4-bd8d375912b5)
* [Ардуино джъмпери](https://www.aliexpress.com/item/Dupont-Jumper-wire-10CM-20CM-30CM-Male-to-Male-Female-to-Male-Female-to-Female-Jumper/32962785036.html?spm=2114.search0104.3.1.cf625d1bGq1yKE&ws_ab_test=searchweb0_0,searchweb201602_1_10065_10068_319_10059_10884_317_10887_10696_321_322_10084_453_10083_454_10103_10618_10304_10307_10820_10821_537_10302_536,searchweb201603_50,ppcSwitch_0&algo_expid=b4685a81-25c3-44aa-9bb9-d7c407e19ea8-0&algo_pvid=b4685a81-25c3-44aa-9bb9-d7c407e19ea8)
* [Бредборд](https://www.aliexpress.com/item/Breadboard-830-Point-Solderless-PCB-Bread-Board-MB-102-MB102-Test-Develop-DIY/32701019904.html?spm=2114.search0104.3.25.be723e44SFup7f&ws_ab_test=searchweb0_0,searchweb201602_1_10065_10068_319_10059_10884_317_10887_10696_321_322_10084_453_10083_454_10103_10618_10304_10307_10820_10821_537_10302_536,searchweb201603_50,ppcSwitch_0&algo_expid=6edd0145-2a6c-48b2-b18a-5115af0f2b0d-3&algo_pvid=6edd0145-2a6c-48b2-b18a-5115af0f2b0d)

2.2 Използван софтуер, езици за програмиране и библиотеки

**Използван софтуер**:

[Arduino IDE](https://www.arduino.cc/en/main/software) – безплатна стандартна среда за разработка за Arduino.

Главният файл на конзолата main.ino е написан изцяло на Arduino IDE.

[Visual Studio Code](https://code.visualstudio.com/) – безплатен текстови едитор разработен от Microsoft.

С негова помощ са написани всички игри и цялата логика на конзолата.

[GitHub](https://github.com/) – сорс контрол система

Линк към курсовата работа: <https://github.com/KostadinovK/ELSYS-Arduino-Project>

**Използвани езици за програмиране**:

Курсовата работа е написана изцяло на C++.

**Използвани библиотеки и файлове:**

Сорс кодът не е изцяло авторски. Използвани са външни библиотеки разрешени за ползване.

Arduino.h – главния Ардуино хедър файл, който дефинира всички функции специфични за Ардуиното.

LiquidCrystal.h – хедър файл дефиниращ функции за използването и контролирането на LCD дисплея.

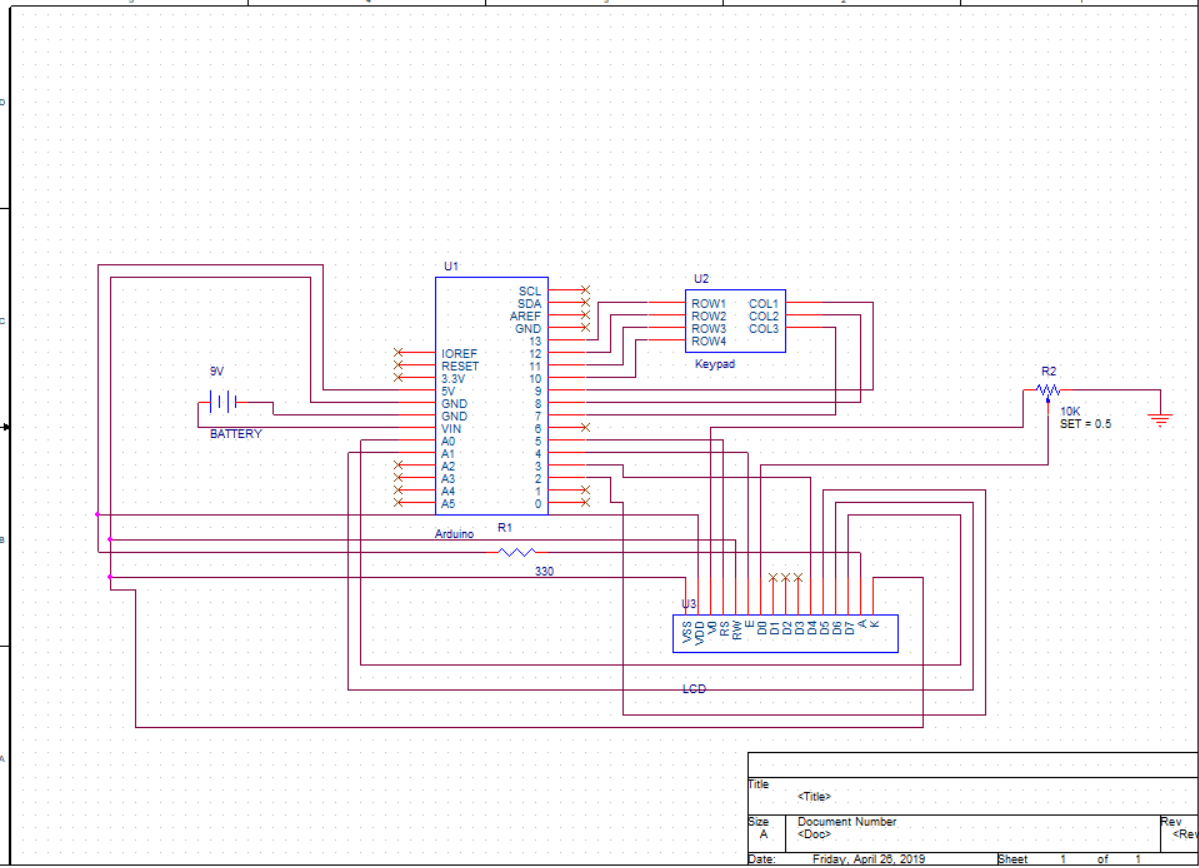
[Keypad](https://playground.arduino.cc/Code/Keypad/) - библиотека съдържаща функционалност за контролирането и използването на 4x3 кейпада, служещ като контролер за конзолата.

Другите използвани библиотеки са изцяло авторски.

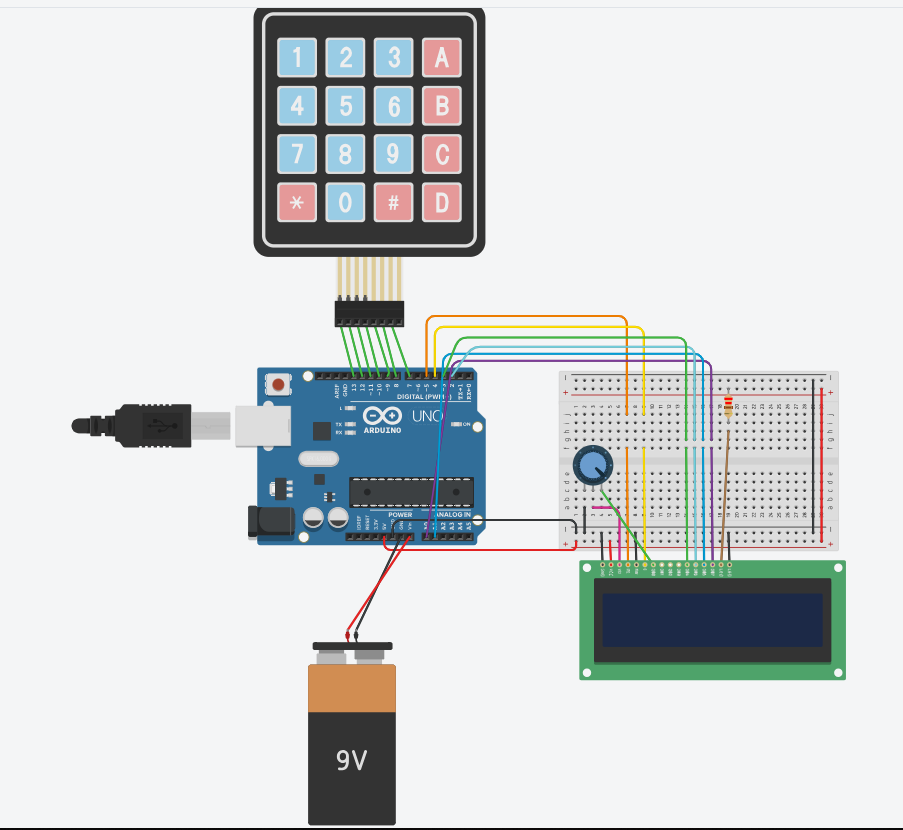
**3.Хардуер**

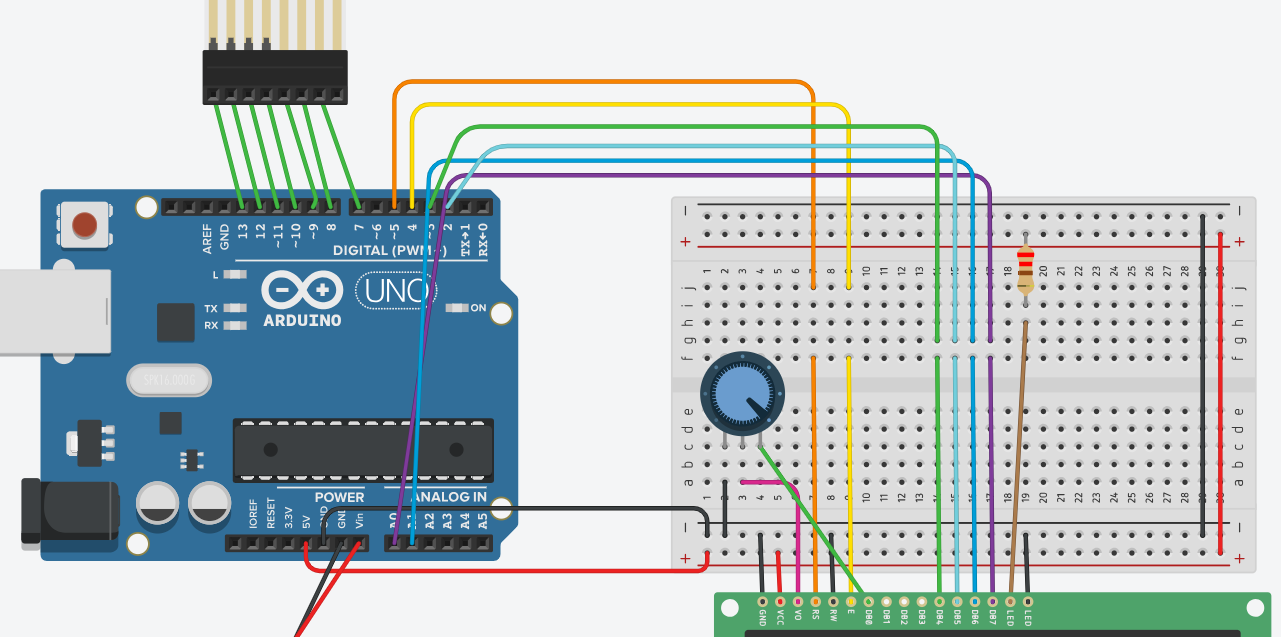
3.1 Принципна схема на устройството

Принципната схема е начертана на OrCAD Capture 10.5.



Същата схемата, но представена по-добре визуално, начертана на ThinkerCad:



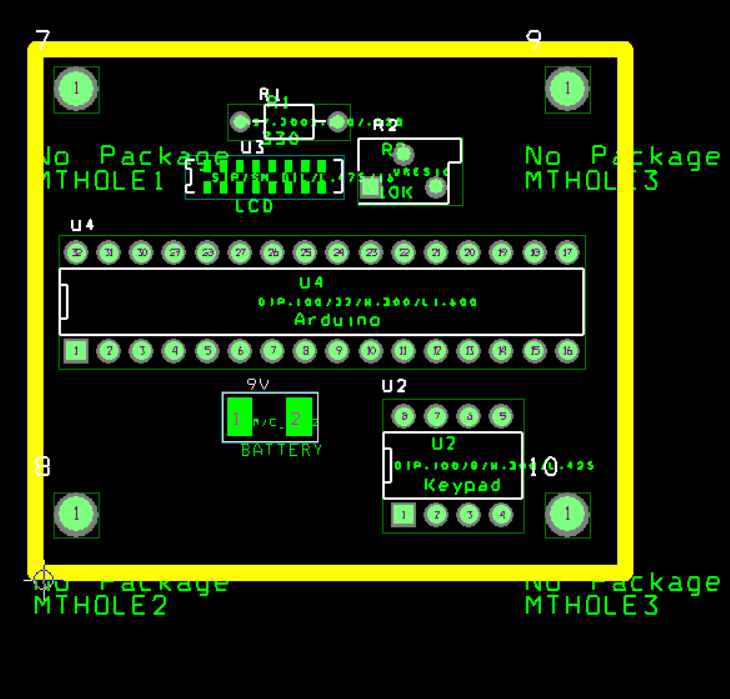


Поради това, че в ThinkerCAD няма 4x3 кейпад е използвам единствено в схемата 4x4. Двата кейпада са абсолютно идентични от гледна точка на използване, просто втория има с четири бутона повече.

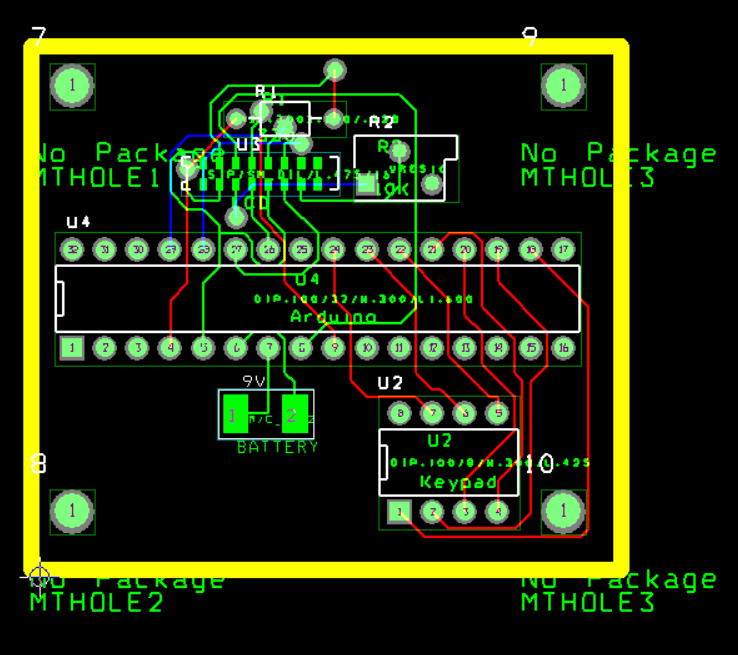
3.2 Графичен оригинал на печатната платка

Графичния оригинал на печатната платка е изработен с OrCAD Layout 10.5

Неопроводен графичен оригинал:



Опроводен графичен оригинал:

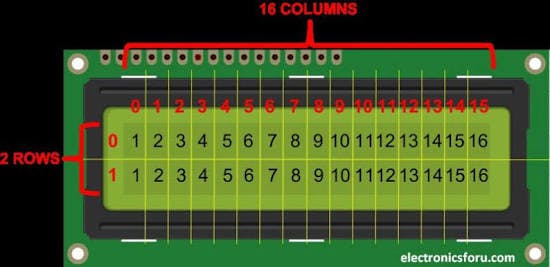


* 1. Устройство и работа

Батерията захранва Ардуиното като „+“ се върже към Vin пина, а пък „-“ към един от GND пиновете.

От своя страна Ардуиното захранва бредборда чрез 5V и GND.

LCD дисплей – 16x2 дисплея се състои от 32 чара, като има 16 на първия ред и 16 на втория. Всеки чар се състои от 40 пиксела(8 \* 5). За щастие има вграден контролер (HD44780) в самия дисплей, чиято работа е да контролира тези пиксели.



Дисплеят има 16 пина, като два от тях са за подсветката. Тя може да се включи по избор. Сред тези 14 оставащи пина има 8 пина за данни(D0-D7), 2 пина за захранване(VSS и VDD), 1 пин за контраст на дисплея(V0) – контролираме го с потенциометъра и последните три пина са за контрол(RS, RW и E).

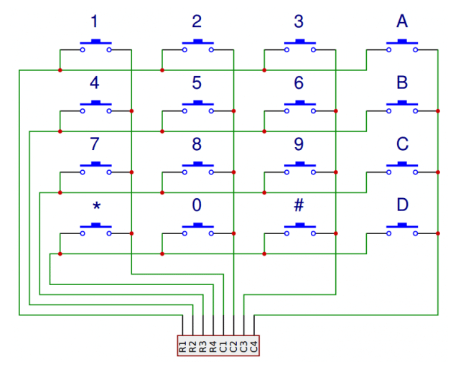
RW пина, който служи за преминаване на дисплея от Read в Write режим и обратното не се използва често затова може да се заземи, това настройва LCD-то в режим за четене. Така трябва просто да се контролира E(enable) пина и RS пина за да изпращаме правилно данни към дисплея.

* Заземяваме VSS.
* Връзваме VDD към+5V**.**
* Връзваме V0 към потенциометъра.
* Връзваме RS*(Register Selection)* към 5-ти пин на Ардуиното.
* Заземяваме RW *(Read/Write)* .
* Връзваме E *(Enable)* към 4-ти пин.
* Връзваме D4 към 3-ти пин.
* Връзваме D5 към 2-ри пин.
* Връзваме D6 към А1 пина.
* Връзваме D7 към А0 пина**.**
* Връзваме анода(А пин) на диода за задна подсветка към захранването заедно с последователно свързан към него 330 ом-ов резистор
* И връзваме катода към земята**.**

Не се използват пиновете D0, D1, D2 и D3 защото Ардуиното е в четири битов режим и за да се по-лесно синхронизирането между екрана и микроконтролера се използват само четири пина за данни.

След това вече Ардуиното може да използва дисплея по този начин, като вкара хедър файла и го инициализира.

4x3 кейпад – Всъщност кейпадът представлява 12 мембранни бутона свързани помежду си във формата на матрица(редове и колони).



Ардуиното засича кой бутон е натиснат спрямо това на кой ред и коя колона се намира.

Намирането на реда и колоната се случва главно чрез четири стъпки:

1. Когато нито един бутон не е натиснат всички пинове отговарящи за колоните са „1“, а всички пинове отговарящи за редовете са „0“.

2. Когато бутонът е натиснат пин на съответната колона пада в „0“.

3. Вече Ардуиното знае колоната затова превключва всеки един от пиновете отговарящи за редове в „1“, като в същото време чете всички колони и следи коя ще се върне в „1“.

4. Когато съответната колона премине в „1“, Ардуиното открива бутона.

За по-лесна работа с кейпада е хубаво да се изтегли библиотеката Keypad за Ардуино(автори – Марк Стенли и Александър Бревиг). Библиотеката се грижи за настройването на пиновете.

* 1. **Софтуер**

Целият код на проекта е достъпен на този адрес <https://github.com/KostadinovK/ELSYS-Arduino-Project>

4.1 Блок схемa

Блок схема на main.ino файла, начертана на [Creately](https://creately.com/):



**Блок „Импортване на нужни библиотеки“** – импортват се всички нужни бибиотеки за да работи конзолата, а имено библиотеките за дисплея, кейпада и всички други свързани с менюто и игрите.



**Блок „Дефиниране на пинове“ –** дефинират се всички нужни пинове, които ще трябват за правилно връзване на схемата и функциониране.



**Блок „Инициализация на екран и кейпад“ –**



Дефинират се чаровете на кейпада(матрица), както и пиновете към които са свързани редовете и колоните и се инициализират екрана и кейпада(чрез функцията makeKeymap(char[][])).

**Блок „Инициализация на GameEngine пойнтерите“ –** всеки един пойнтер сочи към място в памета, в което е дефиниран наследник на GameEngine класа. GameEngine е абстрактен клас, който се грижи за подкарването на игра, в себе си той има пойнтери към инстанция на кейпад и екран. Всяка игра се пише като се наследи този базов клас.



**Блок „Създаване на меню“ –** Менюто, както и MenuItem класа са дефинирани в Menu.h. Менюто съдържа един масив от MenuItem. Един MenuItem има стринг с името на опцията на която отговаря(играта) и един пойнтер сочещ към съответния GameEngine, нужен на играта да тръгне. Дефинират се тези MenuItem-и и потребителят може да избира между тях.

За да работи менюто трябва да се подадат екран и кейпад в конструктора му.



**Блок „Инициализиране на GameController, който стартира избраната игра“**

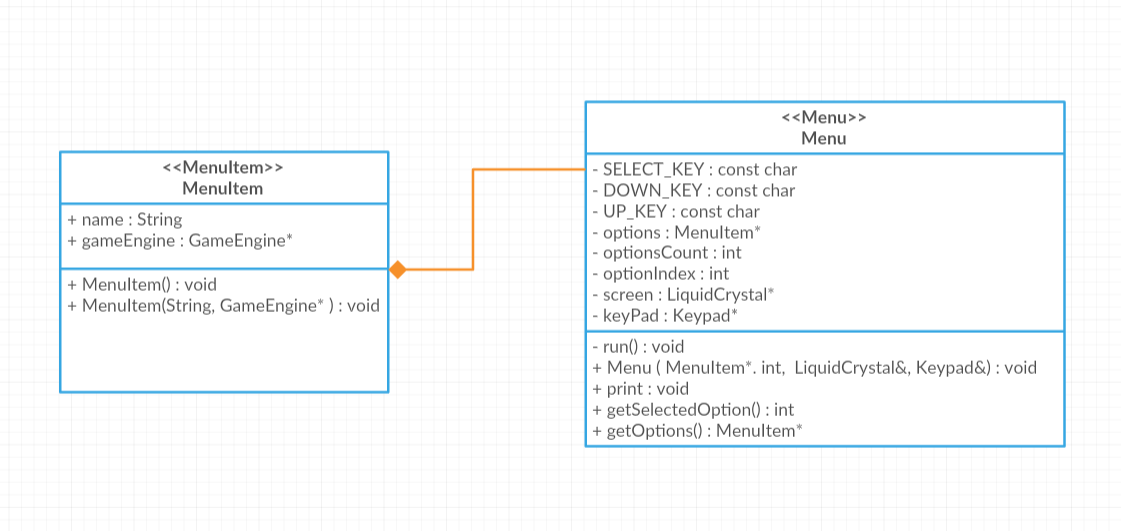
GameController класа е дефиниран в GameController.h неговата работа е да стартира избран GameEngine от потребителя. Той получава в конструктора си при инициализация пойнтери към кейпада и екрана, както и целия масив от MenuItem от менюто и номера на избраната игра. Играта се стартира чрез executeSelectedOption() функцията на GameController-а. Това е прави в setup фунцията на Ардуиното заедно със самото принтиране на менюто.



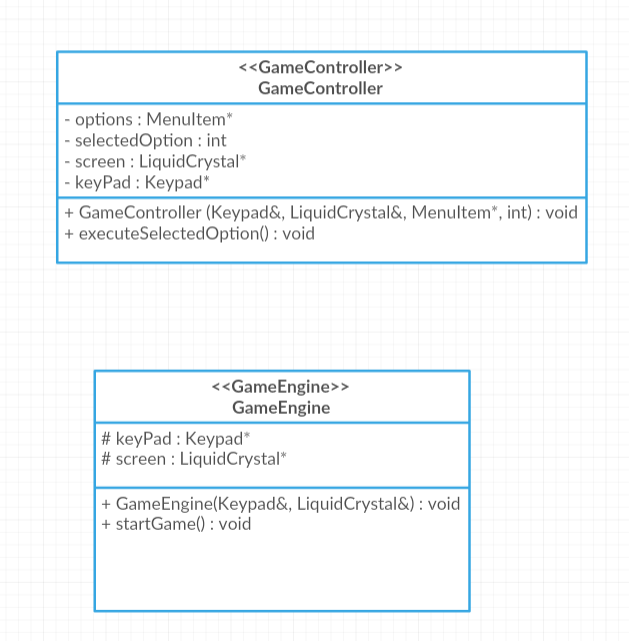
* 1. UML диаграми

Диаграмите са начертани на [Creately](https://creately.com/).

Диаграма на Menu библиотеката:



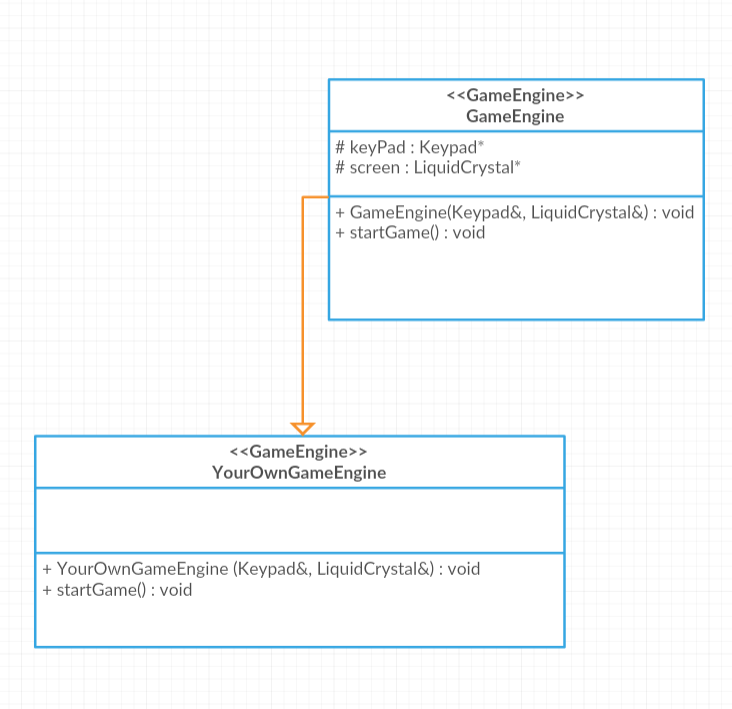
Диаграма на GameController библиотеката:



GameController-а се грижи да стартира избраната игра от потребителя през менюто с метода executeSelectedOption().

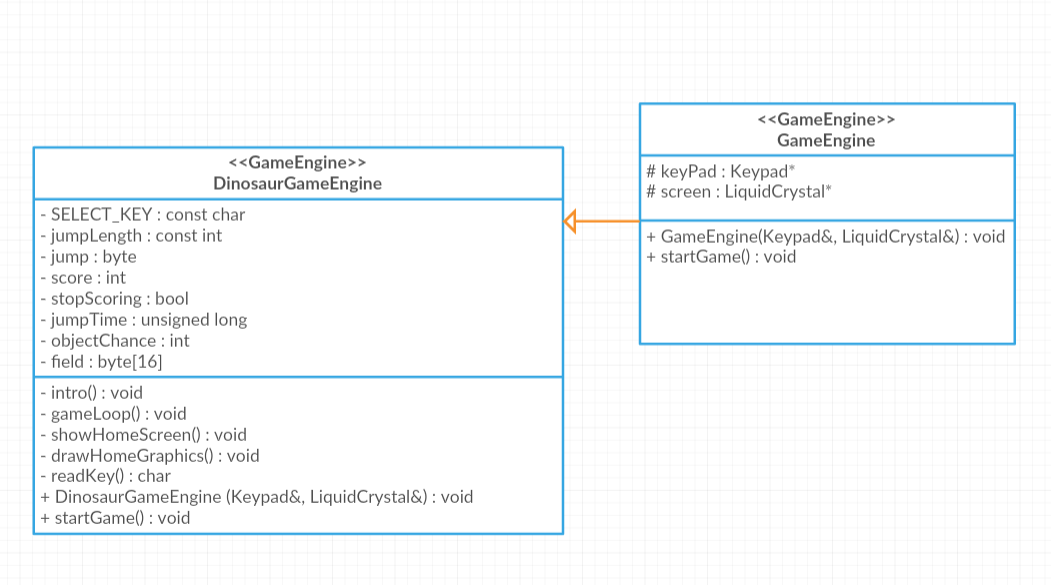
GameEngine е абстрактен клас, дефиниращ базовите неща, които всеки енджин трябва да има за да се стартира безпроблемно от GameController-а.

Диаграма на YourOwnGameEngine:



YourOwnGameEngine е клас който просто принтира на конзолата, че тук можеш да напишеш играта си.

Диаграма на DinosaurGame:

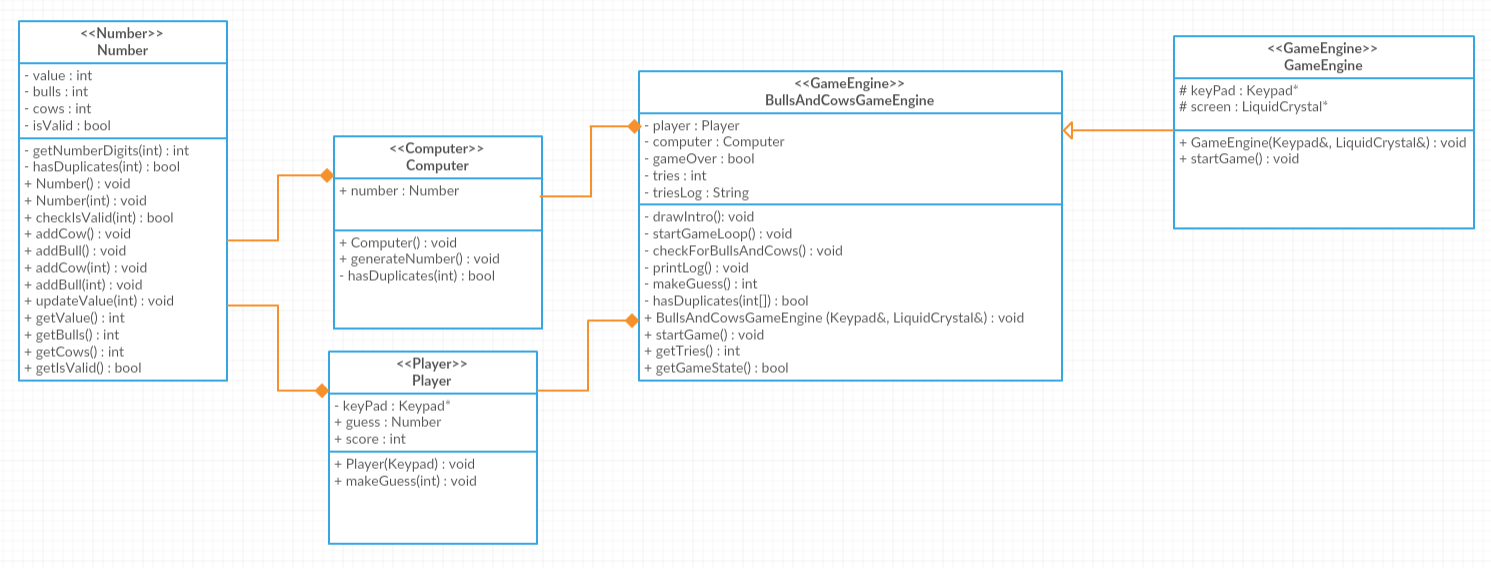


Цялата логика на „Динозавър“ играта е в класа DinosaurGameEngine, който наследява GameEngine класа.

Играта представлява малко по-различно копие на версията за Google Chrome. Ходова е за разлика от другата и е доста по-проста.

Оригиналната игра: <https://chromedino.com/>

Диаграма на BullsAndCowsGame:



Играта се състои от четири класа, играч, компютър, клас за числото и самия енджин. Целта е играча да познае генерираното от компютъра число, което е четирицифрено и е с четири различни цифри.

В зависимост от това какво е числото се добавят нужните крави и бикове към него, когато числото на играча получи 4 бика, то значи, че съвпада с генерираното число и играчът побеждава.

Подробни правила: <https://en.wikipedia.org/wiki/Bulls_and_Cows>

4.3 Документация

**Функции на Menu класа:**

**void run() –** при извикване менюто става интерактивно като може да се скролва при натискане на определените за това бутони и могат да се избират опции. Сменя се индекса на MenuItem-а, реално този индекс е избраната опция.

**Menu(MenuItem\* options, int optionsCount, LiquidCrystal& screen, Keypad& keyPad)** - конструктор, приемащ пойнтер сочещ към масив от MenuItem инстанции, бройка на инстанциите и екран и кейпад подадени по референция.

**void print() –** извежда на конзолата началния екран и инструкциите за работа с менюто.

**int getSelectedOption() –** връща индекса на който се намира избраната от потребителя опция.

**MenuItem\* getOptions() –** връща пойнтер който сочи към първия елемент на MenuItem масива.

**Функции на MenuItem класа:**

**MenuItem() –** дефоултен конструктор

**MenuItem(String name, GameEngine\* gameEngine)** – конструктор приемащ като параметри стринг с името на опцията и пойнтер сочещ към съответния GameEngine.

**Функции на абстрактния GameEngine клас:**

**GameEngine(Keypad& keyPad, LiquidCrystal& screen) –** конструкторприемащ като параметри екран и кейпад подадени по референция.

**virtual void startGame() = 0 –** абстрактна функция, стартураща играта за която GameEngine-а е отговорен. Трябва да се имплементира ит наследник на класа.

**Функции на GameController класа:**

**GameController(Keypad& keyPad, LiquidCrystal& screen, MenuItem\* options, int option) –** конструктор, приемащ референция към кейпад, екран, пойнтер сочещ към масива с MenuItems и индекса на избраната от потребителя опция(игра).

**void executeSelectedOption() –** вика startGame() функцията на съответния гейм енджин отговарящ на индекса на избраната игра.

**Функции на DinosaurGameEngine класа:**

Класът наследява GameEngine класа.

**void intro() –** дефинира всички графики в играта и вика showHomeScreen() функцията.

**void gameLoop() –** върти игровия цикъл с цялата логика на играта. Вика readKey() функцията.

**void showHomeScreen() –** принтирана дисплея началния екран и принтира инструкции за играта, както и името й. Извиква drawHomeGraphics() функцията.

**void drawHomeGraphics()** – отговорна е за принтирането на графиките, дефинирани в intro() функцията.

**char readKey() –** отговаря за натискането на бутон, като функцията връща чара на бутона, който е бил натиснат.

**DinosaurGameEngine(Keypad& keyPad, LiquidCrystal& screen) –** конструктор, приемащ референция към кейпада и екрана.

**void startGame() –** стартира играта като вика функцията intro() и gameLoop().

**Функции на Number класа:**

**Number() –** дефоултен конструктор.

**Number(int num) –** конструктор, сетваш value = num и биковете и кравите на 0.

**bool checkIsValid(int num) –** проверява дали числото е валидно, вика getNumberDigits() и hasDuplicates().

**void addCow() –** Добавя крава към числото.

**void addBull() –** Добавя бик към числото.

**void addCow(int cows) –** Като параметър идва броя на кравите, които се добавят към числото.

**void addBulls(int bulls) –** Като параметър идва броя на биковете, които се добавят към числото.

**void updateValue(int value) –** Сменя стойноста на числото с подадената като параметър стойност и нулира броя на биковете и кравите.

**int getNumberDigits(int num) –** Подава се число като параметър и връща броя на цифрите в числото.

**bool hasDuplicates(int num) –** Връща булева стойности дали подаденото число като параметър съдържа дублиращи се цифри.

**int getValue() –** връща стойноста на числото.

**int getBulls() –** връща колко бикове има числото.

**int getCows() –** връща колко крави има числото.

**bool getIsValid() –** връща булева стойност дали числото е валидно, дали съдържа дублиращи се цифри.

**Функции на Player класа:**

**Player(Keypad keyPad) –** конструктор, приемащ кейпада като параметър.

**void makeGuess(int value) –** Приема стойност като параметър и я присвоява на Number инстанцията на Player класа.

**Функции на Computer класа:**

**Computer() –** конструктор, който дефинира рандъм сийда за генератора на числа.

**void generateNumber()** – генерира число между 1000 и 9998 като генерира число докато то не бъде валидно. Вика hasDuplicates() за да провери дали числото има дублиращи се цифри.

**bool hasDuplicates(int num) –** проверява дали числото подадено като пареметър има дублиращи се цифри.

**Функции на BullsAndCowsGameEngine класа:**

**void drawIntro() –** принтира името на играта, главното меню на играта и инструкциите за игра.

**void startGameLoop() –** стартира главния игрови цикъл, който контролира цялата логика на играта.

**void checkForBullsAndCows() –** проверява колко крави и бикове има въведеното от играча число и добавя биковете и кравите към него.

**void printLog() –** Изпринтира на конзолата всички опити на играча да познае числото във формат {число} B:{бикове} C:{крави}.

**int makeGuess() –** Изчаква потребителят да въведе валидно число. При невалидно число потребителят трябва да въвежда отново докато не въведе валидно число.

**bool hasDuplicates(int value[]) –** Като параметър функцията приема масив от четири цифри, цифрите на числото, и ги проверява дали има сред тях повтарящи се цифри. Резултатът се връща като булева променлива.

**BullsAndCowsGameEngine(Keypad& keyPad, LiquidCrystal& screen) –** Конструктор приемащ кейпад и екран.

**void startGame() –** Стартира играта, извиквайки drawIntro() и startGameLoop().

**int getTries() –** Връща броя на опитите направени от играча.

**bool getGameState() –** Връща булева стойност дали играта е приключила или не.