Игра на памет

„Запомни последователността“

Изготвили:

Кристиана Стоянова и Виктор Иванов

# Съдържание

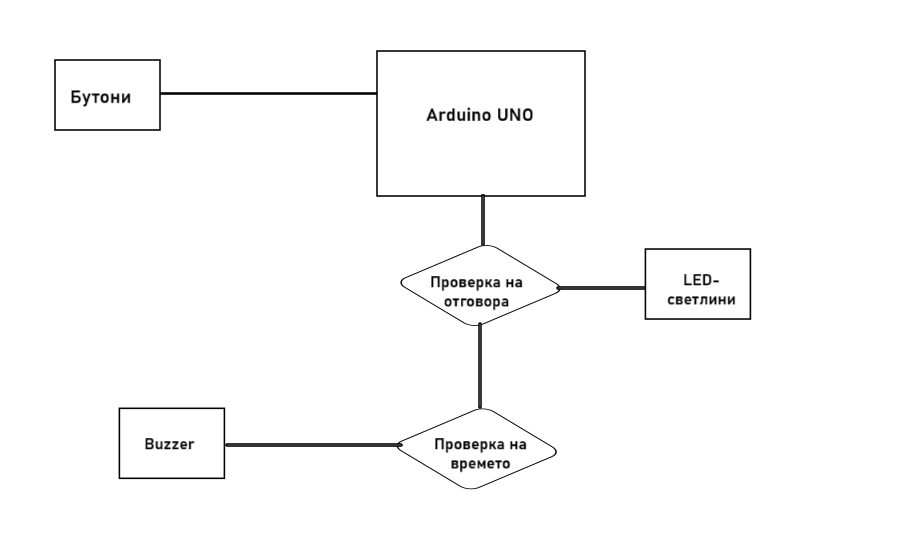
1. Описание на проекта  
2. Блокова схема  
3. Електрическа схема  
4. Списък съставни части  
5. Сорс код – описание на функционалността  
6. Заключение

# 1. Описание на проекта

Проектът представлява електронна игра, подобна на "*Саймън казва"* или на английски: *Simon Says*.

Целта е играчът да запомни и повтори последователност от светлини (LED), които се добавят една по една след всеки правилен отговор. Ако играчът допусне грешка или изтече времето за отговор, играта се рестартира. Проектът използва Arduino Uno, светодиоди, бутони и Пиезозумер за звукова обратна връзка.

# 2. Блокова схема



# 3. Електрическа схема

Описание:  
- 4 бутона свързани с пинове 2, 8, 10, 13  
- 4 LED светодиода – към същите пинове, но през резистори (~220Ω)  
- Пиезозумер (buzzer) – свързан към пин 5

# 4. Списък съставни части

|  |  |  |
| --- | --- | --- |
| № | Част | Количество |
| 1 | Arduino Uno | 1 бр. |
| 2 | Бутони | 4 бр. |
| 3 | LED светодиоди | 4 бр. |
| 4 | Резистори (220Ω за LED) | 4 бр. |
| 5 | Резистори (1kΩ за бутони) | 4 бр. |
| 6 | Пиезозумер (buzzer) | 1 бр. |
| 7 | Кабели | мн. бр. |
| 8 | Breadboard | 1 бр. |

# 5. Сорс код – описание на функционалността

#define PLAYER\_WAIT\_TIME 5000 - Дефинира се константа – максимално време в **милисекунди** (5 секунди), в което играчът трябва да натисне бутон.

byte sequence[100]; - Масив, в който се съхранява поредността на натисканите бутони

byte currentSequenceLength = 0; - Показва колко дълга е текущата последователност от бутони.

byte playerInputCount = 0;- Брой на коректните натискания от играча в настоящия ход.

byte lastPlayerInput = 0; - Последният бутон, натиснат от играча.

byte expectedButton = 0;- Бутонът, който играчът трябва да натисне в текущия момент.

bool isButtonPressed = false;- показва дали в момента има натиснат бутон.

bool isWaitingForPlayer = false;- дали чакаме вход от играча.

bool hasPlayerLost = false;- дали играчът е загубил.

byte speakerPin = 5;- Пинът, към който е свързан buzzer-а

byte totalButtons = 4;

byte buttonPins[] = {2, 13, 10, 8};- Определя колко бутона (и светодиода) има, и на кои **пинове** на Arduino са свързани.

long inputStartTime = 0;- ТАймер за измерване колко време е минало от последния вход от играча.

void setup()

{

delay(3000);- изчаква 3 сек след стратиране

resetGame();-започва нова игра

}

void setAllPinModes(byte mode)

{

for (byte i = 0; i < totalButtons; i++)

{

pinMode(buttonPins[i], mode);

}

}- Установява всички LED-ове като INPUT или OUTPUT в зависимост от това дали чакаме вход или извеждаме светлинен сигнал.

void writeToAllPins(byte value)

{

for (byte i = 0; i < totalButtons; i++)

{

digitalWrite(buttonPins[i], value);

}

}- Изпраща същата стойност (HIGH или LOW) към всички LED пинове.

void resetGame()

{

flashAllLeds(500);

currentSequenceLength = 0;

playerInputCount = 0;

lastPlayerInput = 0;

expectedButton = 0;

isButtonPressed = false;

isWaitingForPlayer = false;

hasPlayerLost = false;

}- рестартира играта като светва всички LED-ове 5 пъти и нулира променливите

void showSequence()

{

for (int i = 0; i < currentSequenceLength; i++)

{

digitalWrite(sequence[i], HIGH);

delay(500);

digitalWrite(sequence[i], LOW);

delay(250);

}

}- Показва текущата последователност като включва светодиода на съответния пин. Изчаква, след което го изключва.

void flashAllLeds(short interval)

{

setAllPinModes(OUTPUT);

for (int i = 0; i < 5; i++)

{

writeToAllPins(HIGH);

playBeep(50);

delay(interval);

writeToAllPins(LOW);

delay(interval);

}

}- Светва всички LED-ове 5 пъти заедно с кратък звук.

void playBeep(byte frequency)

{

analogWrite(speakerPin, 2);

delay(frequency);

analogWrite(speakerPin, 0);

delay(frequency);

}- Генерира звук от пиезобъзера

void indicateLoss()

{

flashAllLeds(50);

}- Индикатор за загуба – бързо мигане със всички LED-ове

void processLoss()

{

indicateLoss();

delay(1000);

showSequence();

delay(1000);

resetGame();

}- Последователност при загуба:

1. Светка бързо всички LED-ове.

2. Показва последната последователност. //идеята е да се преброи резултата и да се види грешката

3. Изчаква и рестартира играта.

void loop()

{

  if (!isWaitingForPlayer)

  {

    setAllPinModes(OUTPUT); - Нагласяне на пиновете за извеждане на светлинен сигнал

    randomSeed(analogRead(A0));

    sequence[currentSequenceLength] = buttonPins[random(0, totalButtons)]; - Поставете нова произволна стойност на следващата позиция в последователността

    currentSequenceLength++; - Увеличава дължината на последователността

    showSequence(); - Показва последователността на играча

    playBeep(50); - Издава звук означаващ начало на време за повтаряне на комбинацията

    isWaitingForPlayer = true; - Задаване на true тъй като е ред на играча

    inputStartTime = millis(); - Съхранява времето за измерване на времето за реакция на играча

  }

else

{

    setAllPinModes(INPUT); - Нагласяне на пиновете за вход на сигнал

    if (millis() - inputStartTime > PLAYER\_WAIT\_TIME)

{

      processLoss();

      return;

    }- Проверка за установяване на загуба на база времето на играча

    if (!isButtonPressed)

{

      expectedButton = sequence[playerInputCount]; - Намира стойността която очакваме от играча

      Serial.print("Expected: ");

      Serial.println(expectedButton);

      for (int i = 0; i < totalButtons; i++) – Цикъл преминаващ през всички пинове

{

        if (buttonPins[i] == expectedButton) – Пропуска правилния пин

          continue;

        if (digitalRead(buttonPins[i]) == HIGH) – Проверява дали всеки от бутоните е натиснат

{

          lastPlayerInput = buttonPins[i];

          hasPlayerLost = true; - Това показва, че сме загубили

          isButtonPressed = true; - Това ще попречи на програмата да прави едно и също отново и отново

          Serial.print("Read: ");

          Serial.println(lastPlayerInput);

        } – При натиснат бутон различен от правилния влизаме в тялото на If-а тоест сме загубили

      }

    }

    if (digitalRead(expectedButton) == 1 && !isButtonPressed) – Играчът е натиснал правилния бутон

{

      inputStartTime = millis();

      lastPlayerInput = expectedButton;

      playerInputCount++; - Отброява рекорда на играча от правилно натиснат бутон

      isButtonPressed = true; - Това ще попречи на програмата да прави едно и също отново и отново

      Serial.print("Read: ");

      Serial.println(lastPlayerInput);

    }

else

{

      if (isButtonPressed && digitalRead(lastPlayerInput) == LOW) – Проверява дали играчът е пуснал бутона

{

        isButtonPressed = false;

        delay(20);

        if (hasPlayerLost) - Това ще се задейства ако по-горе сме загубили

{

          processLoss(); - Изпълнение на процеса по загуба

        }

else

{

          if (playerInputCount == currentSequenceLength) – Проверява дали играчът е завършил последователността

{

            isWaitingForPlayer = false; - чрез това ще се върнем към първия If в основния цикъл

            playerInputCount = 0; - Нулиране на броя натискания на бутони от играча

            delay(1500);

          }

        }

      }

    }

  }

}

# 6. Заключение

Проектът „Запомни последователността“ успешно реализира интерактивна електронна игра, базирана на платформата Arduino Uno. Чрез използването на светодиоди, бутони и звукова обратна връзка (пиезозумер), създадохме система, която стимулира паметта и концентрацията на играча по забавен и предизвикателен начин. По време на разработката бяха приложени както хардуерни, така и софтуерни умения – изградихме електрическа схема, програмирахме логиката на играта на езика C++ и осигурихме потребителско изживяване, близко до оригиналната идея на играта „Simon Says“.

Линк към симулация на проекта: <https://wokwi.com/projects/433738084871874561>

Линк към GitHub: <https://github.com/kristianaas5/project8m>