Практическое занятие №3

Изучение принципов работы с семисегментным индикатором и матричной клавиатурой в САПР Proteus и Arduino IDE

Цель работы:

1) изучить основные принципы работы со матричной клавиатурой 4x3 и семисегментным индикатором в САПР Proteus и Arduino IDE;
2) научиться разрабатывать алгоритмы управления устройствами индикации с использованием матричной клавиатуры на примере семисегментного индикатора;
3) научиться моделировать устройства индикации в САПР Proteus.

1 Краткие теоретические сведения

1.1 Микросхема MAX7219

Микросхема MAX7219 – это драйвер для светодиодной индикации. Используется для управления семисегментными и матричными индикаторами.

Драйвер MAX7219 управляется по трехпроводной последовательной шине Microwire (3-Wire). Драйвер допускает каскадирование для управления большим числом индикаторов (до восьми семисегментных индикатора с точкой, либо отдельно 64 светодиода в LED матрицах 8x8 с общим катодом). Каждый из разрядов индикатора имеет независимую адресацию и его содержимое может быть обновлено без необходимости перезаписи всего индикатора. MAX7219 также позволяет пользователю определять режим декодирования каждого разряда. Кроме того, драйвер MAX7219 имеет спящий режим с запоминанием информации, аналоговое и цифровое управление яркостью подключенных индикаторов и тестовый режим, включающий все LED сегменты.

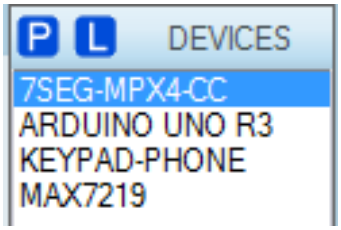
Даташит данной микросхемы:

https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf

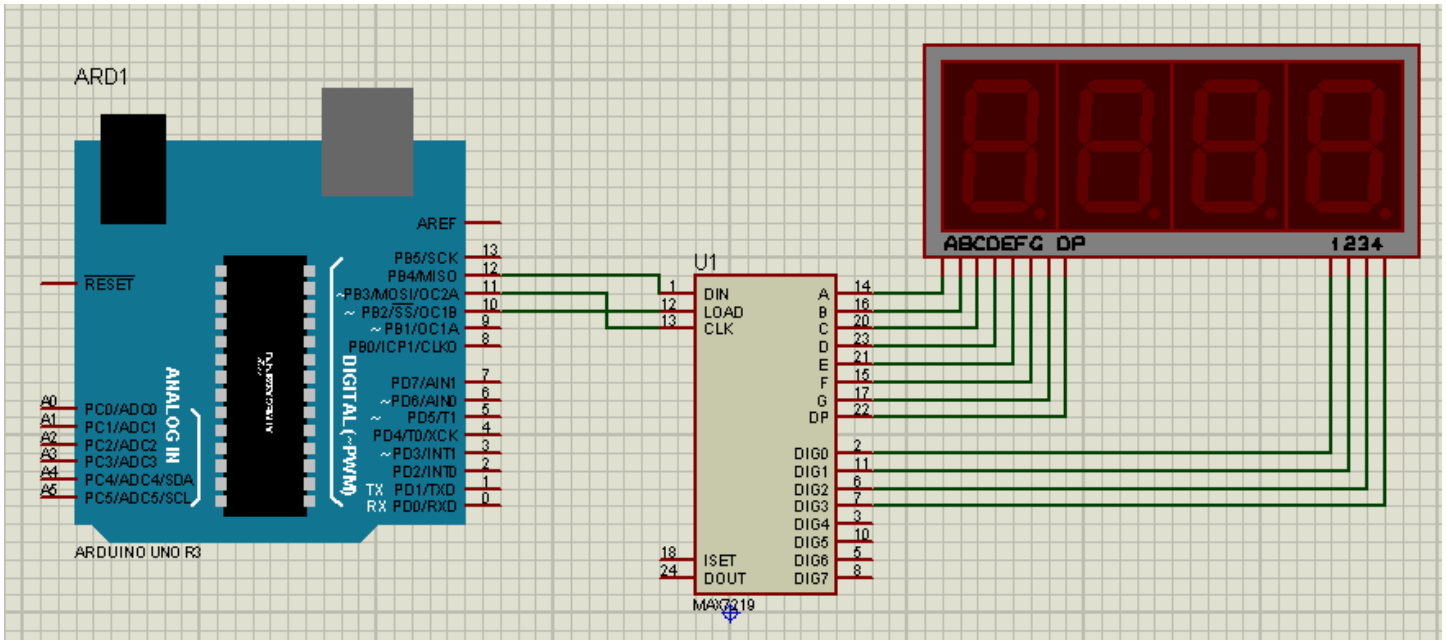
1.2 Схема подключения матричной клавиатуры и семимегментного индикатора к Arduino в Proteus

Для того чтобы собрать данную схему, понадобятся следующие компоненты:

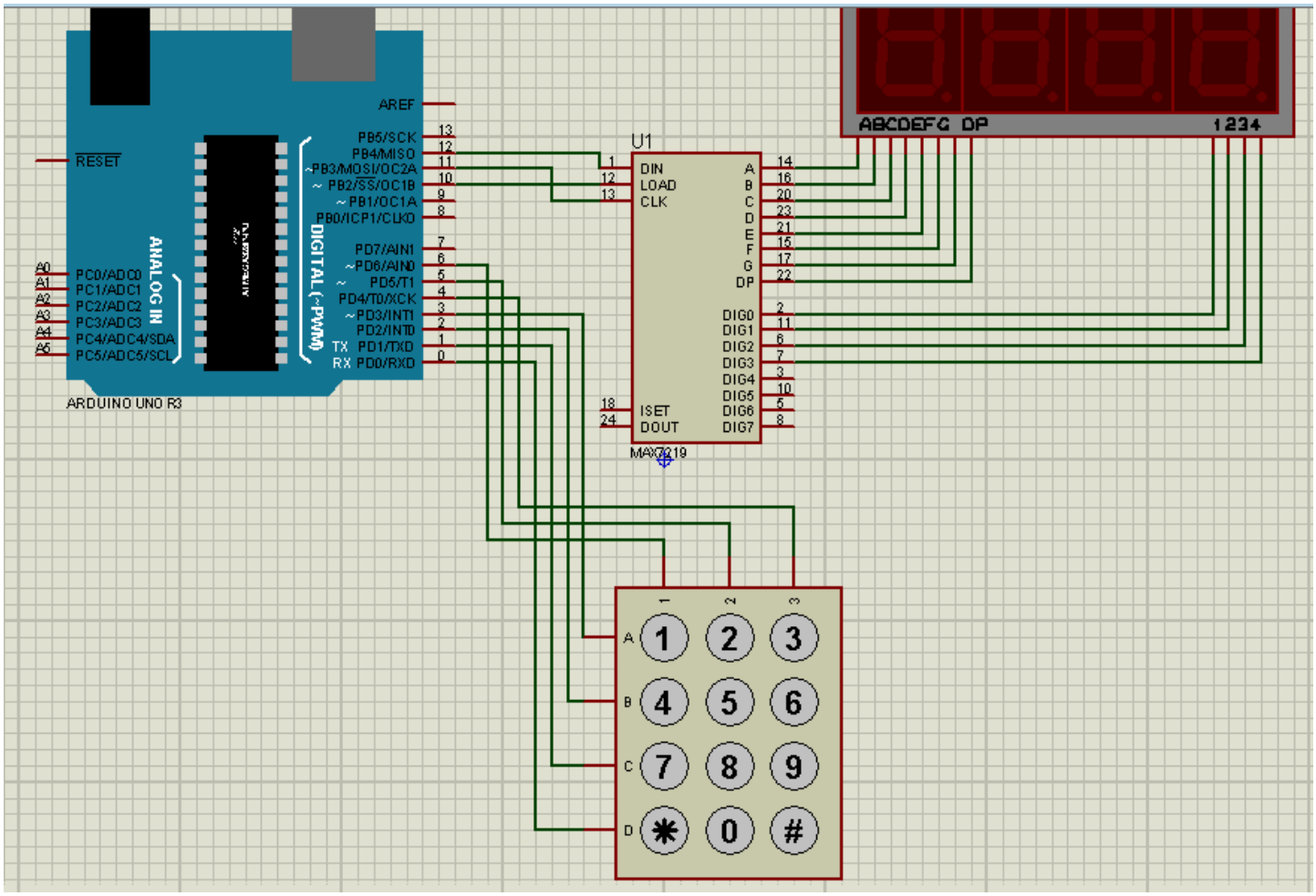
1) Arduino Uno
2) MAX7219
3) 7SEG-MPX4-CC
4) KEYPAD-PHONE



Подключение драйвера MAX7219 к Arduino осуществляется через выводы 10 (SS), 11 (MOSI) и 12 (MISO). Типовая схема подключения семисегментного индикатора к Arduino приведена на следующем изображении:

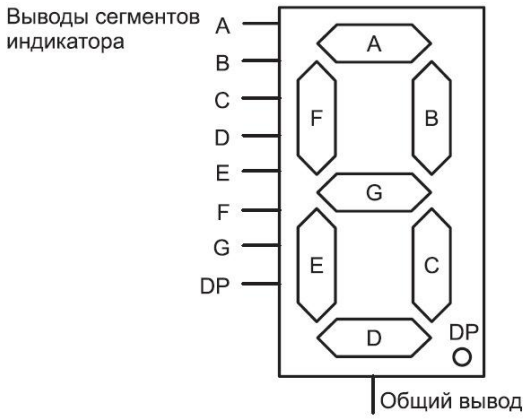


Матричная клавиатура 4x3 подключается к любым семи цифровым выводам Arduino. Пример подключения показан на следующем рисунке:

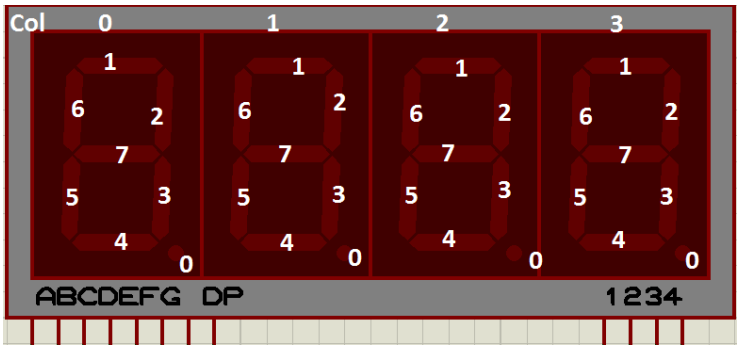


1.3 Механизм работы с семисегментным индикатором

Светодиодный семисегментный индикатор представляет собой группу светодиодов, расположенных в определенном порядке и объединенных конструктивно. Светодиодные контакты промаркированы метками от a до g (и дополнительно dp – для отображения десятичной точки), и один общий вывод, который определяет тип подключения индикатора (схема с общим анодом ОА, или общим катодом ОК). Зажигая одновременно несколько светодиодов, можно формировать на индикаторе символы цифр. Схема одноразрядного семисегментного индикатора показана на следующем рисунке.



Ниже приведена схема четырёхразрядного семисегментного индикатора:



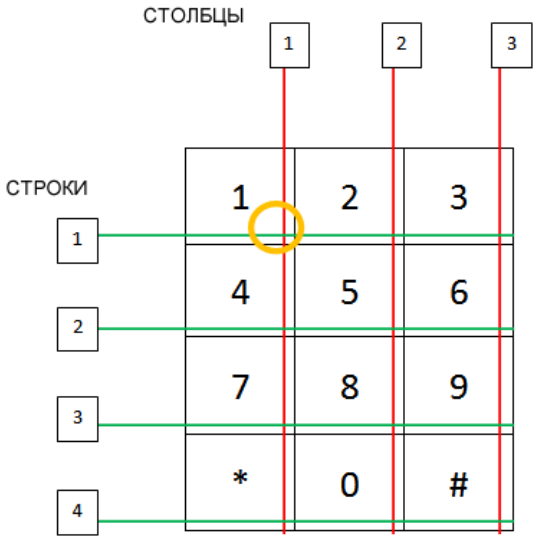
Каждый сегмент в модуле индикатора мультиплексирован, то есть он разделяет одну анодную/катодную точку соединения с другими сегментами своего разряда. И каждый из четырех разрядов в модуле имеет собственную точку подключения с общим катодом/анадом. Это позволяет каждую цифру включать или выключать независимо. Кроме того, такой метод мультиплексирования позволяет микроконтроллеру использовать только одиннадцать или двенадцать выводов вместо тридцати двух (при подключении напрямую к МК).

1.4 Механизм работы с матричной клавиатурой

Клавиатуры позволяют пользователям вводить данные во время выполнения программы. Она часто требуется для обеспечения ввода данных в систему на Arduino, и матричные клавиатуры являются экономичным решением для многих приложений. Они довольно тонкие и могут быть легко установлены везде, где они необходимы.

Клавиатура 4x3 имеет три столбца и четыре строки. Нажатие кнопки замыкает вывод одной из строк с выводом одного из столбцов. Из этой информации Arduino может

определить, какая кнопка была нажата. Например, когда нажата кнопка 1, замкнуты столбец 1 и строка 1. Arduino определит это и введет в программу 1.
На рисунке ниже показано, как внутри клавиатуры расположены строки и столбцы.



1.5 Библиотека LedControl

Данная библиотека позволяет управлять подключёнными к MAX7219 устройствами (например, семисегментным индикатором или светодиодной матрицей) с помощью Arduino. Поскольку библиотека LedControl поддерживает до восьми каскадно подключенных MAX7219, это позволяет индивидуально управлять 512 светодиодами, и для этого нужно по-прежнему всего три цифровых контакта Arduino.
К Arduino IDE подключить библиотеку можно добавив папку LedControl (*Docs\Вычислительные машины\Практическое занятие №3\ Библиотека\LedControl*) в соответствующий каталог библиотек Arduino IDE, по умолчанию находящийся по пути:
C:\Users\<Имя пользователя>\Documents\Arduino\libraries
К проекту библиотека подключается следующим образом:

```
#include "LedControl.h"
```

Все функции библиотеки вызываются через переменную типа **LedControl**. Создание экземпляра класса выполняется следующим образом:

```
int DIN = 12;
int LOAD =  10;
int CLK = 11;
int dev_count = 3;
LedControl lc = LedControl (DIN, CLK, LOAD, dev_count);
```

Инициализация нескольких/одного MAX7219 выполняется следующим образом:

```
for (byte i = 0; i < dev_count; i++) { // i – адрес MAX7219
lc.shutdown(i, false); // спящий режим выключен
lc.setIntensity(i, 15); // яркостьна максимум (значение от 0 до 15)
lc.clearDisplay(i); // очистка светодиодной матрицы
}
```

Основные функции данной библиотеки:

void setLed(int addr, int row, int col, boolean state); *// Задает статус отдельного светодиода на семисегментном индикаторе*

Параметр	Описание
int addr	Адрес дисплея, которому нужно передать команду
int row	Позиция, в которой расположен светодиод (0 … 7)
int col	Столбец, в котором расположен светодиод (0 … 3)
boolean state	Если «true», то светодиод включается, а если «false», то выключается

void setChar(int addr, int digit, char value, boolean dp); *// Позволяет отобразить символ на индикаторе*

Доступные символы:

- 0 1 2 3 4 5 6 7 8 9
- A a (будет печататься заглавная буква)
- B b (будет печататься заглавная буква)
- C c (будет печататься заглавная буква)
- D d (будет печататься заглавная буква)
- E e (будет печататься заглавная буква)
- F f (будет печататься заглавная буква)
- H h (будет печататься заглавная буква)
- L l (будет печататься заглавная буква)
- P p (будет печататься заглавная буква)
- - (знак минуса)
- ., (десятичная запятая)
- _ (нижнее подчеркивание)
- <SPACE> (пустое место или пробел)

Параметр	Описание
int addr	Адрес дисплея, которому нужно передать команду
int digit	Столбец, в котором расположен светодиод (0 … 3)
char value	Отображаемый символ
boolean dp	Отображение точки. Если «true», то она включается, а если «false», то нет

void setDigit(int addr, int digit, byte value, boolean dp); *// Отобразить цифру на индикаторе (в шестнадцатеричном виде: 0x00..0x0F)*

Параметр	Описание
int addr	Адрес дисплея, которому нужно передать команду
int digit	Столбец, в котором расположен светодиод (0 … 3)
byte value	Значение, которое нужно вывести в шестнадцатеричном виде: 0x00..0x0F
boolean dp	Отображение точки. Если «true», то она включается, а если «false», то нет

Примеры:

lc.setLed(0,2,3,true); // включаем светодиод, находящийся в позиции №2 на третьей цифре индикатора

lc.setChar(0, 0, 'H', false); // выводим в первый столбец символ 'H'
lc.setDigit (0, 0, 0x0A, false); // выводим в первый столбец цифру A в шестнадцатеричном виде

Подробное описание библиотеки можно прочитать здесь:
http://wikihandbk.com/wiki/Arduino:Библиотеки/LedControl

1.6 Библиотека Keypad

Библиотека Keypad служит для использования совместно с Arduino клавиатур матричного типа. Текущая версия библиотеки (3.1) поддерживает множественные нажатия. Данная библиотека была создана для создания уровня абстракции для аппаратного обеспечения. Она улучшает читаемость кода, скрывая от пользователя вызовы функций *pinMode* и *digitalRead*. Нет необходимости использовать внешние резисторы или диоды, так как библиотека использует внутренние подтягивающие резисторы в микроконтроллере на Arduino и дополнительно обеспечивает высокое входное сопротивление на всех неиспользуемых выводах столбцов.
К Arduino IDE подключить библиотеку можно добавив папку LedControl (*Docs\Вычислительные машины\Практическое занятие №3\ Библиотека\Keypad*) в соответствующий каталог библиотек Arduino IDE, по умолчанию находящийся по пути:
C:\Users\<Имя пользователя>\Documents\Arduino\libraries
К проекту библиотека подключается следующим образом:
#include <Keypad.h>
Все функции библиотеки вызываются через переменную типа ***Keypad***. Создание экземпляра класса выполняется следующим образом: *Keypad(makeKeymap(userKeymap), row[], col[], rows, cols)*
Ниже приведён фрагмент инициализации и настройки Arduino для работы с матричной клавиатурой 4x3:

```
const byte rows = 4; // четыре строки
const byte cols = 3; // три столбца
char keys[rows][cols] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'#','0','*'}
};
byte rowPins[rows] = {5, 4, 3, 2}; // подключить к выводам строк клавиатуры
byte colPins[cols] = {8, 7, 6};    // подключить к выводам столбцов клавиатуры
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, rows, cols );
```

Данный фрагмент кода создает объект Keypad, который использует выводы 5, 4, 3, 2 как выводы строк и выводы 8, 7, 6 как выводы столбцов. Эта клавиатура имеет 4 строки и 3 столбца, в итоге 12 кнопок.
Основные функции данной библиотеки:

void begin(makeKeymap(userKeymap)); *// Инициализирует внутреннюю раскладку клавиатуры, чтобы та соотвествовала userKeymap*
char waitForKey(); *// Данная функция будет ждать бесконечно, пока кто-нибудь не нажмет клавишу. Предупреждение: она блокирует весь остальной кода, пока кнопка не будет нажата. Это означает, что не будет происходить ни мигание светодиода, ни обновление LCD дисплея, ничего, кроме выполнения функций прерываний.*
char getKey(); *// Возвращает кнопку, которая нажата, если таковая имеется. Данная функция является неблокирующей.*
KeyState getState(); *// Возвращает текущее состояние любой из клавиш. Возвращается одно из четырех состояний: IDLE, PRESSED, RELEASED и HOLD.*
boolean keyStateChanged(); *// Дает вам знать, когда изменилось состояние кнопки. Например, вместо проверки нужной клавиши, вы можете проверить, когда клавиша была нажата.*
setHoldTime(unsigned int time); *// Устанавливает количество миллисекунд, которое пользователь должен удерживать кнопку нажатой, чтобы было вызвано состояние HOLD.*
setDebounceTime(unsigned int time); *// Устанавливает количество миллисекунд, которое клавиатура ждет перед тем, как применить новое нажатие кнопки или событие кнопки. Это «время задержки» в методе обработки дребезга контактов.*
addEventListener(keypadEvent); *// Вызывает событие, если используется клавиатура.*

Методы класса **Keypad** и определения для работы с множественными нажатиями:
Key key[LIST_MAX]; *// Список активных клавиш. LIST_MAX (равен 10) задает максимальное количество клавиш в активном списке.*
bool getKeys(); *// Заполняет массив key активных клавиш (до 10 значений). Возвращает true при наличии любых активных клавиш.*
bool isPressed(char keyChar); *// Возвращает true, если кнопка с кодом keyChar нажата.*
int findInList(char keyChar); *// Поиск клавиши по коду в списке активных клавиш. Возвращает -1, если клавиша не найдена, или индекс в массиве активных клавиш.*

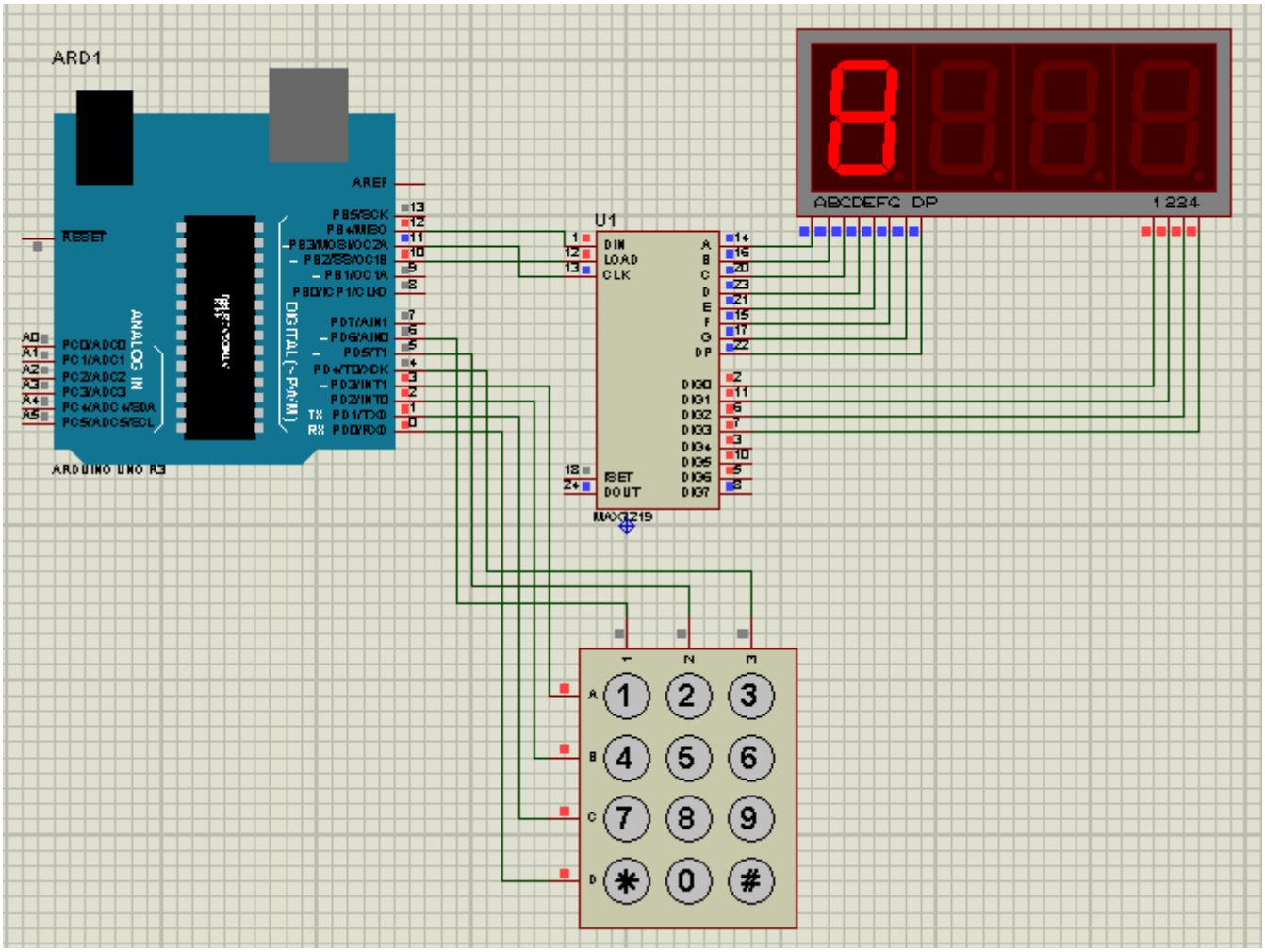
Пример:

```
char customKey = customKeypad.getKey(); // считываем нажатую клавишу
  if (customKey) {
// ……..обработка ……..
                        }
```

Подробное описание библиотеки можно прочитать здесь:
https://radioprog.ru/post/146

1.7 Пример работы

В папке с заданием находится тестовая программа и схема. В программе считывается нажатая клавиша и её код выводится в первый столбец семисегментного индикатора. По нажатию на клавишу «#» индикатор очищается (см. рисунок ниже).

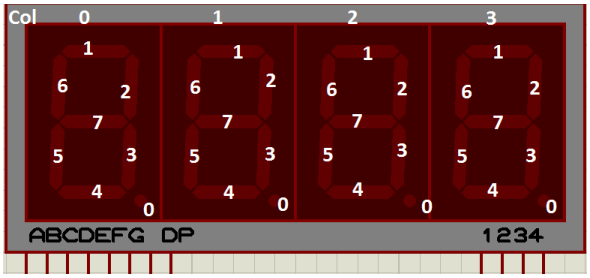


2 Выполнение практического задания

2.1 Задание на практическое занятие

В данном практическом задании необходимо выполнить следующее задание:

1) Начиная с крайней левой позиции индикатора (столбец Col 0) включать по отдельности элементы индикатора по часовой стрелке, начиная с элемента № 1 и заканчивая № 7. Алгоритм повторить для всех позиций индикатора. Задержку между включением элементов индикатора установить равной 100 мс. Далее, повторить алгоритм, но в обратном порядке, т.е. начать с крайней правой позиции индикатора (столбец Col 3) и выключать элементы индикатора против часовой стрелки, двигаясь справа налево (см. видео *Docs\Вычислительные машины\Практическое занятие №3\ Пример работы.avi*).



2) Слева направо выводить на семисегментный индикатор код нажатой на клавиатуре клавиши. После заполнения всех четырёх полей индикатора код последующих нажатых клавиш выводить на индикатор также слева направо. Кнопка «#» должна очищать индикатор и устанавливать курсор в первоначальное положение, т.е. при

нажатии на клавишу, её код должен выводиться в крайнюю левую позицию индикатора. При нажатии на кнопку «*» должен выполняться алгоритм, в соответствии с вариантом:

Вариант	Задание
1	Реализовать счётчик от 0 до 15 в двоичной системе счисления в коде «8421». На индикатор выводить текущее значение счётчика. Время между переключениями значений счётчика выбрать из интервала 50 мс … 500 мс. Для организации задержки использовать функцию **delay (*X мс*);**
2	Реализовать счётчик от 0 до 9 в двоичной системе счисления в коде «5421». На индикатор выводить текущее значение счётчика. Время между переключениями значений счётчика выбрать из интервала 50 мс … 500 мс. Для организации задержки использовать функцию **delay (*X мс*);**
3	Реализовать счётчик от 0 до 9 в двоичной системе счисления в коде Грея. На индикатор выводить текущее значение счётчика. Время между переключениями значений счётчика выбрать из интервала 50 мс … 500 мс. Для организации задержки использовать функцию **delay (*X мс*);**
4	Реализовать счётчик от 0 до 9 в двоичной системе счисления в коде «2 из 5». На индикатор выводить текущее значение счётчика. Время между переключениями значений счётчика выбрать из интервала 50 мс … 500 мс. Для организации задержки использовать функцию **delay (*X мс*);**
5	Реализовать счётчик от 0 до 9 в двоичной системе счисления в коде «дополнение до 9». На индикатор выводить текущее значение счётчика. Время между переключениями значений счётчика выбрать из интервала 50 мс … 500 мс. Для организации задержки использовать функцию **delay (*X мс*);**
6	Реализовать счётчик от 0 до 9 в двоичной системе счисления в коде «2421». На индикатор выводить текущее значение счётчика. Время между переключениями значений счётчика выбрать из интервала 50 мс … 500 мс. Для организации задержки использовать функцию **delay (*X мс*);**
7	Реализовать счётчик от 0 до 4 и с 4 до 0 в двоичной системе счисления в коде Джонсона. На индикатор выводить текущее значение счётчика. Время между переключениями значений счётчика выбрать из интервала 50 мс … 500 мс. Для организации задержки использовать функцию **delay (*X мс*);**

3 Результаты выполнения практического задания

В результате выполнения данного практического задания необходимо составить отчёт (можно в электронном виде), содержащий следующие пункты:

1) Титульный лист
2) Цель практического занятия
3) Задание
4) Ход выполнения практического задания (программный код)
5) Результат выполнения практического задания (скриншоты)
6) Выводы