Практическое занятие №
Тема: «Программирование интерфейсов с помощью энкодера»

Цель работы: приобрести практические навыки по подключению и программированию энкодеров на платформе Arduino.

Последовательность выполнения работы:

– Собрать схемы на макетной плате, иначе при отсутствии набора Arduino в web-приложениях (https://wokwi.com/projects/new/arduino-uno или https://www.tinkercad.com/) для приведенных примеров.
– Запрограммировать микроконтроллер согласно заданию в примере.
– Выполнить задание для самостоятельной работы.

Содержание отчета:

– Название практического занятия, его цель.
– Фото или скриншоты собранной схемы.
– Написанный программный код вставить текстом, Courier New, 12 кегль, одинарный отступ без абзацев.
– Вывод о проделанной работе.
– Файл Fritzing с принципиальной и монтажной схемой.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Энкодер (энкодер поворота, rotary encoder) – это электромеханическое устройство, преобразующее механическое вращение в цифровые сигналы. Это аналог потенциометра, но с важными отличиями.

Основные типы энкодеров
1. Инкрементальный энкодер (Incremental Encoder)
Определяет только изменение положения, имеет два выхода (A и B) с квадратурными сигналами, не запоминает абсолютное положение после отключения питания.

2. Абсолютный энкодер (Absolute Encoder)
Определяет абсолютное положение, имеет уникальный код для каждого положения, запоминает положение после отключения питаниядороже и сложнее в подключении,

Как работает энкодер:
При вращении вала диск с прорезями вращается, ИК-светодиод светит через прорези, фототранзисторы улавливают прерывистый свет и тем самым генерируются импульсы на выходах S1 и S2

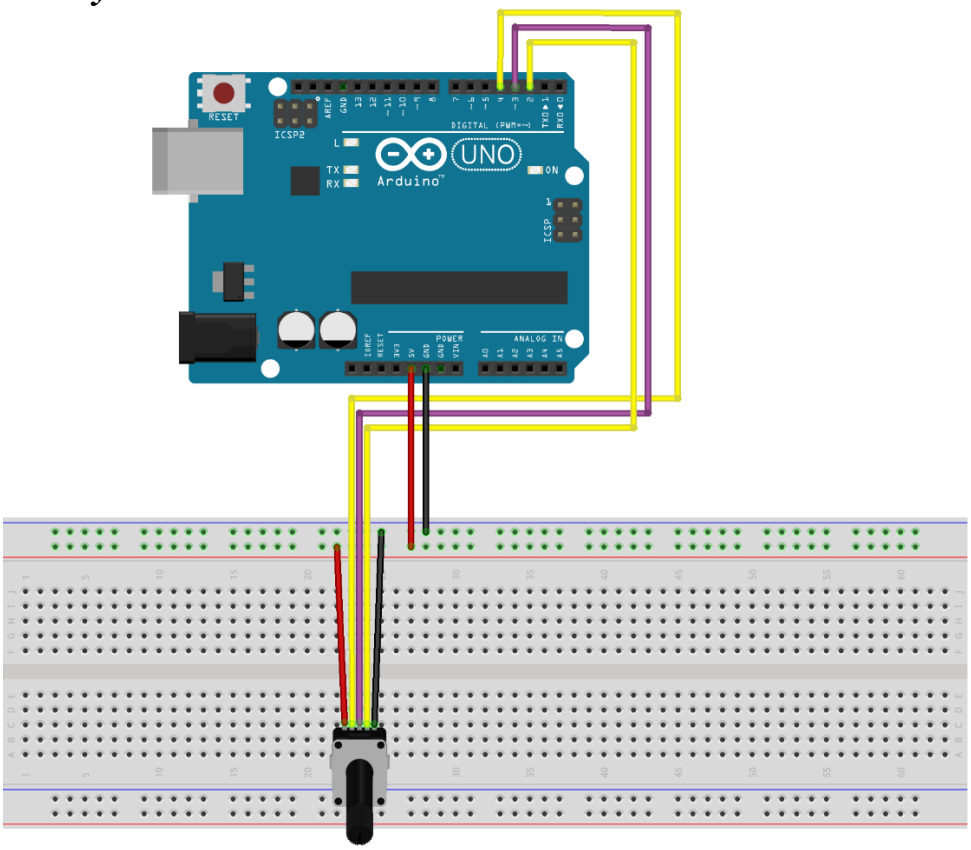
Рисунок 1 – Внешний вид Энкодеров

Назначение выводов:

Вывод	Назначение	Описание
5V	питание	напряжение питания 3.3V-5V
GND	земля	общий провод
S1	канал A (CLK)	первый квадратурный сигнал
S2	канал B (DT)	второй квадратурный сигнал
KEY	кнопка (SW)	тактовая кнопка

ЗАДАНИЯ

Собрать схему:



Написать программный код:

В данной программе энкодер используется в качестве переключателя между значениями диапазона из 20 чисел при помощи вращения. Смена диапазона значений происходит по нажатию.

```
#include <Wire.h>

// ========== НАСТРОЙКИ ПИНОВ ==========
#define ENCODER_CLK 2       // S1 энкодера
#define ENCODER_DT 3        // S2 энкодера
#define ENCODER_SW 4        // KEY энкодера

// ========== ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ ==========
volatile int encoderPosition = 0;
volatile int lastCLK = HIGH;
int lastEncoderPosition = -1;

// Переменные кнопки
bool lastButtonState = HIGH;
bool buttonState = HIGH;
```

```
unsigned long lastDebounceTime = 0;
unsigned long buttonPressTime = 0;
const unsigned long debounceDelay = 50;
const unsigned long longPressDelay = 1000;

// Диапазоны значений
enum Range {
  RANGE_0_20,
  RANGE_21_40,
  RANGE_41_60,
  RANGE_61_80,
  RANGE_81_100
};

Range currentRange = RANGE_0_20;
bool reverseMode = false; // false = 0-100, true = 100-0

// ========== НАСТРОЙКА ==========
void setup() {
  Serial.begin(9600);

  // Настройка пинов энкодера
  pinMode(ENCODER_CLK, INPUT_PULLUP);
  pinMode(ENCODER_DT, INPUT_PULLUP);
  pinMode(ENCODER_SW, INPUT_PULLUP);

  // Настройка прерывания
  lastCLK = digitalRead(ENCODER_CLK);
  attachInterrupt(digitalPinToInterrupt(ENCODER_CLK),
encoderISR, CHANGE);

  displayInfo();
}

// ========== ГЛАВНЫЙ ЦИКЛ ==========
void loop() {
  // Обработка кнопки энкодера
  handleButton();

  // Обновление отображения при изменении положения
  if (encoderPosition != lastEncoderPosition) {
    lastEncoderPosition = encoderPosition;
    displayInfo();
  }
```

```
  delay(10);
}

// ========== ОБРАБОТКА ЭНКОДЕРА ==========
void encoderISR() {
  int clkState = digitalRead(ENCODER_CLK);
  int dtState = digitalRead(ENCODER_DT);

  if (clkState != lastCLK) {
    if (dtState != clkState) {
      encoderPosition = reverseMode ? encoderPosition - 1 :
encoderPosition + 1;
    } else {
      encoderPosition = reverseMode ? encoderPosition + 1 :
encoderPosition - 1;
    }

    // Ограничение значений в зависимости от диапазона
    limitToCurrentRange();

    lastCLK = clkState;
  }
}

void limitToCurrentRange() {
  int minVal, maxVal;
  getRangeLimits(minVal, maxVal);

  if (encoderPosition < minVal) encoderPosition = minVal;
  if (encoderPosition > maxVal) encoderPosition = maxVal;
}

void getRangeLimits(int &minVal, int &maxVal) {
  switch(currentRange) {
    case RANGE_0_20: minVal = 0; maxVal = 20; break;
    case RANGE_21_40: minVal = 21; maxVal = 40; break;
    case RANGE_41_60: minVal = 41; maxVal = 60; break;
    case RANGE_61_80: minVal = 61; maxVal = 80; break;
    case RANGE_81_100: minVal = 81; maxVal = 100; break;
  }
}

// ========== ОБРАБОТКА КНОПКИ ==========
```

```
void handleButton() {
  int reading = digitalRead(ENCODER_SW);

  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != buttonState) {
      buttonState = reading;

      if (buttonState == LOW) {
        buttonPressTime = millis();
      } else {
        if (buttonPressTime > 0 && (millis() -
buttonPressTime) < longPressDelay) {
          shortPress();
        }
        buttonPressTime = 0;
      }
    }
  }

  // Проверка длинного нажатия
  if (buttonPressTime > 0 && (millis() - buttonPressTime) >=
longPressDelay) {
    longPress();
    buttonPressTime = 0;
    delay(300);
  }

  lastButtonState = reading;
}

void shortPress() {
  // Переход к следующему диапазону
  if (reverseMode) {
    // Движение назад по диапазонам
    currentRange = (Range)((currentRange - 1 + 5) % 5);
  } else {
    // Движение вперед по диапазонам
    currentRange = (Range)((currentRange + 1) % 5);
  }
```

```
  // Установка значения в середину диапазона
  int minVal, maxVal;
  getRangeLimits(minVal, maxVal);
  encoderPosition = (minVal + maxVal) / 2;

  displayInfo();
}

void longPress() {
  // Переключение направления движения по диапазонам
  reverseMode = !reverseMode;
  displayInfo();
}

// ========== ОТОБРАЖЕНИЕ В МОНИТОРЕ ПОРТА ==========
void displayInfo() {
  Serial.print("Значение: ");
  Serial.print(encoderPosition);
  Serial.print(" | Диапазон: ");
  Serial.print(currentRange + 1);
  Serial.print(" | Инверсия: ");
  Serial.println(reverseMode ? "Да" : "Нет");
}

// функция возврата диапазона
String getRangeName() {
  switch(currentRange) {
    case RANGE_0_20: return "0-20";
    case RANGE_21_40: return "21-40";
    case RANGE_41_60: return "41-60";
    case RANGE_61_80: return "61-80";
    case RANGE_81_100: return "81-100";
    default: return "Неизвестное значение";
  }
}
```