Практическое занятие №
Тема: «Подключение и программирование модуля часов реального времени DS1302»

Цель работы: приобрести практические навыки по подключению и программированию модуля часов реального времени (Real time clock — RTC) DS1302 к 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.

Содержание отчета:

– название практического занятия, его цель;
– фото или скриншоты собранной схемы;
– написанный программный код вставить текстом;
– вывод о проделанной работе;
– файл Fritzing с принципиальной и монтажной схемой.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Основная и единственная микросхема на модуле, это DS1302, как еще называют «Real Time Clock» (Часы Реального Времени), сокращенно RTC. Данная микросхема поддерживает секунды, минуты, часы, день недели, дата, месяц и год информацию, а также следит за количеством дней в месяце и делает поправку на високосный год. Микросхема работает в 24-часовом или 12-часовом формате с индикатором AM/PM.

Рисунок 1 – Модуль RTS DS1302

На плате, кроме микросхемы DS1302, установлен кварц на 32,768 кГц и бокс для батарейки тика CR2032. Для связи и подачи питания, предусмотрен пяти контактный разъем, шагом 2,54 мм.

Назначение контактов:

– **Vcc и GND** — Питание модуля, 2 — 5.5 В;
– **CLK** — Вход тактовой частоты последовательных данных.
– **DAT** — Ввод/вывод последовательных данных
– **RST** — Включение логике RTC

В примере описывается, как подключить модуль DS1302 к Arduino UNO. В программе управления устанавливается текущие время и затем считвается, полученные данные отправляются в последовательный порт.

Подключение:

Схема не сложная, необходимо пять проводов, сначала подключаем шину Vcc и GND от модуля DS1302 к выводам Arduino 5В и GND (можно и запитать и от 3.3В), затем подключаются CLK, DAT, RST от модуля DS1302 к выводам Arduino 13, 11, 10 (при необходимости, выводы можно поменять).

ЗАДАНИЕ

Схема:

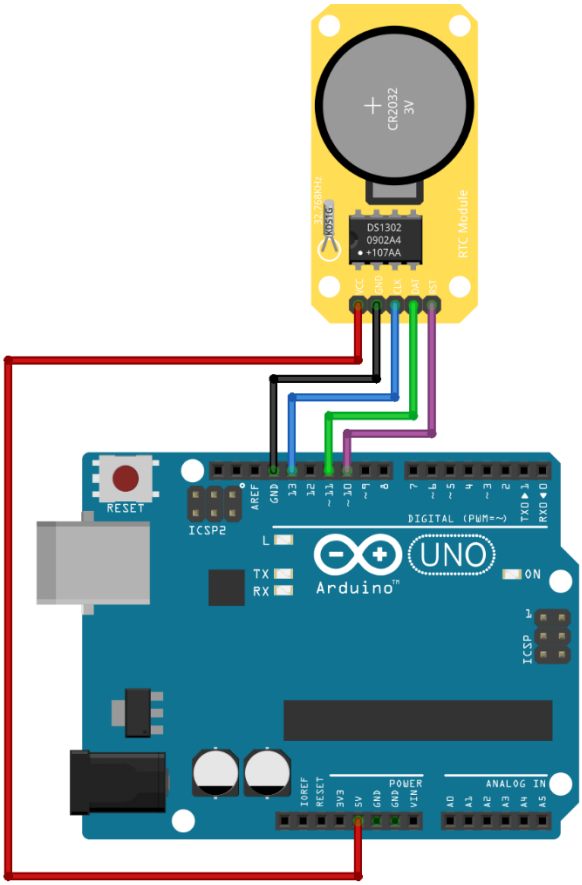

Рисунок 2 – Схема кодключения модуля RTC DS1302

Программа №1:

```
#include <ThreeWire.h> // Подключаем библиотеку ThreeWire
#include <RtcDS1302.h> // Подключаем библиотеку RtcDS1302

ThreeWire myWire(11,13,10); // Указываем вывода IO, SCLK, CE
RtcDS1302<ThreeWire> Rtc(myWire);

void setup ()
{
    Serial.begin(9600); // Установка последовательной связи
на скорости 9600
    Serial.print("Data: "); // Отправка данных на
последовательный порт
    Serial.println(__DATE__); // Получение даты и времени с
ПК
    Serial.print("Time: "); // Отправка данных на
последовательный порт
    Serial.println(__TIME__);
// Получение даты и времени с ПК

    Rtc.Begin();   // Инициализация RTC
    RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
// Копирование даты и времени в compiled

    Rtc.SetDateTime(compiled); // Установка времени
    Serial.println(); // Отправка данных на последовательный
порт
}

void loop ()
{
  Rtc.GetDateTime();
  RtcDateTime now = Rtc.GetDateTime();

  Serial.print("Дата RTC: ");// Отправка данных на
последовательный порт
  Serial.print(now.Month());// Отправка месяца
  Serial.print(".");  // Отправка данных на последовательный
порт
  Serial.print(now.Day());  // Отправка дня
  Serial.print(".");  // Отправка данных на последовательный
порт
  Serial.print(now.Year()); // Отправка года
```

```
  Serial.print(" TIME RTC "); // Отправка данных на
последовательный порт
  Serial.print(now.Hour()); // Отправка часа
  Serial.print(":");  // Отправка данных на последовательный
порт
  Serial.print(now.Minute()); // Отправка минут
  Serial.print(":");  // Отправка данных на последовательный
порт
  Serial.println(now.Second());     // Отправка секунд
  delay(1000);     // Пауза 2 с
}
```

Программа №2:

В программе 2 добавлено динамическое управление светодиодами. Пины 2-7 инициализируются как выходы. Количество горящих светодиодов зависит от десятков секунд (0-9 → 1 светодиод, 10-19 → 2 и т.д.).

Необходимо дополнить цепь 6 светодиодами на пинах 2-7 (через резисторы 220 Ом), написать программу ниже и протестировать работу на Arduino.

Светодиоды будут обновляться каждую секунду, показывая текущие десятки секунд в визуальной форме.

```
#include <ThreeWire.h>
#include <RtcDS1302.h>

ThreeWire myWire(11, 13, 10); // IO, SCLK, CE
RtcDS1302<ThreeWire> Rtc(myWire);

// Диапазоны секунд и соответствующие пины
const int LED_COUNT = 6;
const int LED_PINS[LED_COUNT] = {2, 3, 4, 5, 6, 7};

void setup() {
  Serial.begin(9600);

  // Инициализация светодиодов
  for (int i = 0; i < LED_COUNT; i++) {
    pinMode(LED_PINS[i], OUTPUT);
    digitalWrite(LED_PINS[i], LOW);
  }

  Rtc.Begin();
```

```
  // Проверка валидности времени RTC
  if (!Rtc.IsDateTimeValid()) {
    Serial.println("Время RTC не установлено!");
    // Установка времени компиляции только при необходимости
    RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
    Rtc.SetDateTime(compiled);
  }
}

void loop() {
  RtcDateTime now = Rtc.GetDateTime();

  // Получаем текущие секунды
  int seconds = now.Second();

  // Определяем количество активных светодиодов
  int activeLeds = (seconds / 10) + 1;
  if (activeLeds > LED_COUNT) activeLeds = LED_COUNT;

  // Управляем светодиодами
  for (int i = 0; i < LED_COUNT; i++) {
    digitalWrite(LED_PINS[i], (i < activeLeds) ? HIGH :
LOW);
  }

  // Вывод отладочной информации
  Serial.print("RTC Time: ");
  Serial.print(now.Hour());
  Serial.print(":");
  Serial.print(now.Minute());
  Serial.print(":");
  Serial.print(seconds);
  Serial.print(" - Active LEDs: ");
  Serial.println(activeLeds);

  delay(1000);
}
```