```
001 /*
002   Микропроцессорное устройство для обнаружения металлических объектов
003   Разработчик: студент группы 150501 Климович А.Н.
004   Дата: 01.12.2023
005   Все права защищены (c) 2023 Алексей Климович
006 */
007
008 #include <LiquidCrystal_I2C.h>   // Библиотека для работы с LCD дисплеем
009 #include "iarduino_RTC.h"        // Библиотека для работы с модулем RTC
010
011 // ========================================================================
012 // ---------------ПОДКЛЮЧЕНИЕ ЭЛЕМЕНТОВ К МИКРОКОНТРОЛЛЕРУ---------------
013 // ========================================================================
014 #define RED_LED 8              // Номер контакта красного светодиода (D8)
015 #define BLUE_LED 7             // Номер контакта синего светодиода (D7)
016 #define YELLOW_LED 6           // Номер контакта желтого светодиода (D6)
017 #define CALIBRATE_BUTTON_GND 2 // Номер контакта кнопки калибровки (D2)
018 #define CALIBRATE_BUTTON 3     // Номер контакта кнопки калибровки (D3)
019 #define REGULATOR 3            // Номер контакта потенциометра (A3)
020 #define BLUE_BUTTON 9          // Номер контакта синей кнопки (D9)
021 #define WHITE_BUTTON 10        // Номер контакта белой кнопки (D10)
022 #define PIEZO_SOUND 4          // Номер контакта пьезоизлучателя (D4)
023 #define PIN_RST 11             // Номер контакта сброса модуля RTC (D11)
024 #define PIN_DAT 12             // Номер контакта данных RTC (D12)
025 #define PIN_CLK 13             // Номер входа тактовой частоты RTC (D13)
026
027 // ========================================================================
028 // ----------------------------КОНСТАНТЫ-----------------------------------
029 // ========================================================================
030 #define SOUND_DELAY 20             // Задержка между воспроизведениями
031 #define SOUND_DURATION 30          // Продолжительность одного звучания
032 #define SOUND_FREQUENCY 3000       // Частота воспроизводимого звука
033 #define STATIC_MODE 1              // Статический режим поиска
034 #define DYNAMIC_MODE 0             // Динамический режим поиска
035 #define DEFAULT_BAUD 9600          // Скорость передачи данных
036 #define DYNAMIC_TIMER_INTERVAL 300 // Время срабатывания динамич. таймера
037 #define LONG_PRESS_DURATION 1000   // Время долгого удержания кнопки
038 #define FREQUENCY_RST_CONST 0.5    // Коэффициент плавного сброса частоты
039
040 // ========================================================================
041 // ------------------МАКРОСЫ УСТАНОВКИ/ОЧИСТКИ БИТОВ--------------------
042 // ========================================================================
043 #define SET(x,y) (x |=(1<<y))
044 #define CLR(x,y) (x &= (~(1<<y)))
045 #define CHK(x,y) (x & (1<<y))
046 #define TOG(x,y) (x^=(1<<y))
047 // ========================================================================
048 // ----------------------ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ-------------------------
049 // ========================================================================
050
051 float sensitivity = 1000.0;      // Чувствительность устройства
052 unsigned long t0 = 0;            // Время последнего срабатывания Timer1
053 unsigned long dynamic_timer;           // Таймер динамического режима
054 int t = 0;                             // Время между прерываниями
055 unsigned char tflag = 0;               // Флаг готовности к изменению
056 int start_frequency = 0;               // Начальная частота
057 float f = 0;                           // Значение измерения
058 unsigned int FTW = 0;                  // Частота генератора писка
059 unsigned int PCW = 0;                  // Фаза генератора писка
060 unsigned long count_timer = 0;         // Счетный таймер
061 unsigned int currentMenu = BLUE_BUTTON; // Текущее меню
062 unsigned int mode = 1;                 // Режим работы устройства
063 unsigned int cursorRow = 1;            // Положение курсора в меню RTC
```

```
064 unsigned long pressStartTime = 0;       // Время начала удержания кнопки
065 LiquidCrystal_I2C lcd(0x27, 20, 4);     // Создание объекта дисплея LCD
066 iarduino_RTC rtc(RTC_DS1302,    // Создания объекта модуля RTC DS1302
067                  PIN_RST,
068                  PIN_CLK,
069                  PIN_DAT);
070 // =====================================================================
071
072 // Функция установки режимов работы контактов и инициализации переменных
073 void setup()
074 {
075   rtc.begin();          // Инициализация работы модуля RTC
076   lcd.init();           // Инициируем работу с LCD дисплеем
077   lcd.backlight();      // Включаем подсветку LCD дисплея
078   lcd.clear();          // Очищаем дисплей
079   setupPins();          // Настройка режимов работы контактов
080   showMainMenu();       // Отображение главного меню
081
082   // Установка счетчика 1 на счет на контакте D5
083   TCCR1A = 0;     // Устанавливаем значение в регистре TCCR1A в 0
084   TCCR1B = 0x07;  // Установка внешнего тактового источника на выводе T1.
085   SET(TIMSK1, OCF1A);   // Включить прерывание Timer1 overflow
086
087   // OCF1A: Output Compare Flag 1A - Флаг 1A совпадения выхода
088   // Бит OCF1A устанавливается при совпадении состояния счетчика1 и OCR1A
089   // Бит OCF1A очищается при обработке соответств. вектора прерывания.
090 }
091 // Главная бесконечная функция программы
092 void loop()
093 {
094   int pressedButton = getPressedButton();   // Получение нажатой кнопки
095   if(pressedButton > 0)                     // Если была нажата кнопка
096   {
097     if((pressedButton == WHITE_BUTTON) &&
098        ((currentMenu == BLUE_BUTTON) ||
099        (currentMenu == WHITE_BUTTON && cursorRow == 3)))
091 // Главная бесконечная функция программы
092 void loop()
093 {
094   int pressedButton = getPressedButton();   // Получение нажатой кнопки
095   if(pressedButton > 0)                     // Если была нажата кнопка
096   {
097     if((pressedButton == WHITE_BUTTON) && ((currentMenu == BLUE_BUTTON)
098         || (currentMenu == WHITE_BUTTON && cursorRow == 3)))
099     {
100       switchMenu();                         // Смена меню
101     }
102     else if(currentMenu == WHITE_BUTTON)
103       timeMenu(pressedButton);              // Выполнить функцию RTC
104   }
105   if (digitalRead(CALIBRATE_BUTTON) == LOW) // Нажата красная кнопка
106   {
107     if (pressStartTime == 0)
108       pressStartTime = millis();            // Отсчет времени удержания
109     else
110     { // Длинное удержание кнопки
111       if (millis() - pressStartTime > LONG_PRESS_DURATION)
112       { // Смена режимов работы устройства
113         mode = mode == STATIC_MODE ? DYNAMIC_MODE : STATIC_MODE;
114         pressStartTime = 0;       // Сбрасываем время удержания кнопки
115       }
116     }
117   }
```

```
118   else
119     pressStartTime = 0;           // Сбрасываем время удержания кнопки
120
121   if (tflag)                      // Установлена готовность к изменениями
122   {
123     if (mode == STATIC_MODE)    // Выбран статический режим
124       staticMode();
125     if (mode == DYNAMIC_MODE)   // Выбран динамический режим
126       dynamicMode();
127     f = f * 0.85 + absf(t - start_frequency) * 0.15;  // Фильтруем сигнал
128     sensitivity = map(analogRead(REGULATOR), 0, 1023, 10, 2000);
129     if(currentMenu == BLUE_BUTTON)   // Если выбрано главное меню
130     {
131       printSensitivity();    // Вывод чувствительности
132       printFrequency();      // Вывод частоты
133       printMetalScale();     // Вывод шкалы приближения к металлу
134     }
135     float clf = f * sensitivity;  // Конвертация изменения частоты в звук
136     if (clf > 10000)
137       clf = 10000;
138     FTW = clf;
139     tflag = 0;      // Сброс признака готовности к изменениям
140   }
141
142   if (millis() > count_timer)     // Если сработал счетный таймер
143   {
144     count_timer += 10;
145     PCW += FTW;
146     if (PCW & 0x8000)             // Если фаза достигнула макс. значения
147     {
148       PCW &= 0x7fff;
149       if(currentMenu == BLUE_BUTTON)    // Если выбрано главное меню
150       {
151         tone(PIEZO_SOUND, SOUND_FREQUENCY, SOUND_DURATION); // Звук
152         digitalWrite(BLUE_LED, HIGH);   // Включить светодиод
153         delay(SOUND_DELAY);             // Задержка
154       }
155     }
156     noTone(PIEZO_SOUND);                // Выключить пьезоизлучатель
157     digitalWrite(BLUE_LED, LOW);        // Выключаем светодиод
158   }
159 }
160
161 // Обработчик прерывания от Timer 1
162 ISR(TIMER1_COMPA_vect)
163 {
164   OCR1A += 1000;      // Срабатывает каждые 1000 импульсов с генератора
165   t = micros() - t0;  // Запоминаем время между прерываниями
166   t0 += t;            // Изменяем время последнего срабатывания Timer 1
167   tflag = 1;          // Установка признака готовности к изменениям
168 }
169
170 // Функция устанавки режима работы указанных контактов
171 void setupPins()
172 {
173   // Установка режима работы контакта для пьезоизлучателя
174   pinMode(PIEZO_SOUND, OUTPUT);
175
176   // Установка режима работы контакта для кнопки калибровки
177   pinMode(CALIBRATE_BUTTON, INPUT_PULLUP);
178   pinMode(CALIBRATE_BUTTON_GND, OUTPUT);
179   digitalWrite(CALIBRATE_BUTTON_GND, LOW);
180
```

```
181   // Установка режима работы контакта для красного светодиода
182   pinMode(RED_LED, OUTPUT);
183   for(int i = 0; i < 3; ++i)
184   {
185      digitalWrite(RED_LED, HIGH);   // зажигаем светодиод
186      delay(500);                    // ждем секунду
187      digitalWrite(RED_LED, LOW);    // выключаем светодиод
188      delay(500);                    // ждем секунду
189   }
190   digitalWrite(RED_LED, HIGH); // включаем светодиод
191
192 // Установка режима работы контакта для синего светодиода
193   pinMode(BLUE_LED, OUTPUT);
194   digitalWrite(BLUE_LED, HIGH); // выключаем светодиод
195
196   // Установка режима работы контакта для желтого светодиода
197   pinMode(YELLOW_LED, OUTPUT);
198   digitalWrite(YELLOW_LED, HIGH); // выключаем светодиод
199 }
200
201 // Функция модуля для типа float
202 float absf(float f)
203 {
204   return f < 0.0 ? -f : f;
205 }
206
207 // Функция статического режима
208 void staticMode()
209 {
210   digitalWrite(YELLOW_LED, HIGH);              // Включить светодиод
211   if (digitalRead(CALIBRATE_BUTTON) == LOW)    // Нажата кнопка сброса
212     start_frequency = t;  // Запомнить текущую частоту
213 }
214 // Функция динамического режима
215 void dynamicMode()
216 {
217   digitalWrite(YELLOW_LED, LOW);  // Выключаем светодиод
218   if (millis() - dynamic_timer > DYNAMIC_TIMER_INTERVAL)
219   {
220     // Запоминаем срабатывание таймера
221     dynamic_timer = millis();
222
223     // Плавно сбрасываем частоту
224     start_frequency = start_frequency * FREQUENCY_RST_CONST +
225                                         (1 - FREQUENCY_RST_CONST) * t;
226   }
227 }
228
229 // Функция смены меню
230 void switchMenu()
231 {
232   if(currentMenu == WHITE_BUTTON) currentMenu = BLUE_BUTTON;
233   else currentMenu = WHITE_BUTTON;
234   switch(currentMenu)
235   {
236     case BLUE_BUTTON:
237       showMainMenu();  // Отображение главного меню
238       break;
239     case WHITE_BUTTON:
240       showTimeMenu();  // Отображение меню времени
241       break;
242   }
243 }
```

```
244 // Функция показа главного меню для поиска металлических объектов
245 void showMainMenu()
246 {
247   lcd.clear();                        // Очистка экрана
248   lcd.setCursor(0, 0);                // Установка курсора в позицию (0, 0)
249   lcd.print("---- Main Menu -----");// Вывод строки
250   lcd.setCursor(0, 1);                // Установка курсора в позицию (0, 1)
251   lcd.print("                    ");// Вывод строки
252   lcd.setCursor(0, 2);                // Установка курсора в позицию (0, 2)
253   lcd.print("Frequency: ");           // Вывод строки пояснения
254   printFrequency();                   // Вывод частоты
255   lcd.setCursor(0, 3);                // Установка курсора в позицию (0, 3)
256   lcd.print("Sensitivity: ");         // Вывод строки пояснения
257   printSensitivity();                 // Вывод чувствительности
258 }
259
260 // Функция вывода частоты
261 void printFrequency()
262 {
263   lcd.setCursor(11, 2);           // Установка курсора в позицию (11, 2)
264   lcd.print("         ");         // Очистка строки
265   lcd.setCursor(11, 2);           // Установка курсора в позицию (11, 2)
266   lcd.print(f);                   // Вывод частоты
267 }
268
269 // Функция вывода чувствительности
270 void printSensitivity()
271 {
272   lcd.setCursor(13, 3);           // Установка курсора в позицию (13, 3)
273   lcd.print("       ");           // Очистка строки
274   lcd.setCursor(13, 3);           // Установка курсора в позицию (13, 3)
275   lcd.print(sensitivity);         // Вывод чувствительности
276 }
277
278 // Функция вывода шкалы приближения металлических объектов
279 void printMetalScale()
280 {
281   lcd.setCursor(0, 1);                // Установка курсора в позицию (0, 1)
282   lcd.print("                    ");// Очистка строки
283   lcd.setCursor(0, 1);                // Установка курсора в позицию (0, 1)
284   int squares = (int)f;               // Кол-во прямоугольников на шкале
285   if(f > 20)
286     squares = 20;                       // Корректировка прямоугольников
287   for(int i = 0; i < squares; ++i)
288     lcd.print(char(255));               // Вывод очередного прямоугольника
289 }
290 // Функция вывода меню RTC модуля
291 void showTimeMenu()
292 {
293   lcd.clear();                        // Очистка экрана
294   lcd.setCursor(0, 0);                // Установка курсора в позицию (0, 0)
295   lcd.print("---- Time Menu -----");// Вывод строки
296
297   lcd.setCursor(0, 1);                // Установка курсора в позицию (0, 1)
298   lcd.print("> Time: ");              // Вывод строки
299   lcd.print(rtc.gettime("H:i, D")); // Вывод времени
300
301   lcd.setCursor(0, 2);                // Установка курсора в позицию (0, 2)
302   lcd.print("  Date: ");              // Вывод строки
303   lcd.print(rtc.gettime("d.m.y"));  // Вывод даты
304
305   lcd.setCursor(0, 3);                // Установка курсора в позицию (0, 3)
306   lcd.print("  ------ Back ------");// Вывод строки
```

```
307
308   cursorRow = 1;                    // Начальное значение курсора
309 }
310
311 // Функция меню RTC
312 void timeMenu(int pressedButton)
313 {
314   delay(100);    // Задержка, чтобы уменьшить дребезг нажатия кнопок
315   if(pressedButton == BLUE_BUTTON)    // Нажата кнопка движения по меню
316   {
317     lcd.setCursor(0, cursorRow);      // Курсор в позицию (0, cursorRow)
318     lcd.print("  ");                  // Очистка курсора '>'
319     cursorRow += cursorRow == 3 ? -(cursorRow - 1) : 1;
320     lcd.setCursor(0, cursorRow);      // Курсор в позицию (0, cursorRow)
321     lcd.print("> ");                  // Вывод курсора
322   }
323   else if(pressedButton == WHITE_BUTTON)  // Нажата кнопка выбора
324   {
325     if(cursorRow == 1)       // Изменение времени
326     {
327       setHours();            // Изменение часов
328       setMinutes();          // Изменение минут
329       setWeekDay();          // Изменение дня недели
330       printTime();           // Печать времени
331     }
332     else if(cursorRow == 2)  // Изменение даты
333     {
334       setMonthDay();         // Изменение дня месяца
335       setMonth();            // Изменение месяца
336       setYear();             // Изменение года
337       printDate();           // Печать даты
338     }
339   }
340 }
341
342 // Вывод времени
343 void printTime()
344 {
345   lcd.setCursor(8, 1);              // Установка курсора в позицию (8, 1)
346   lcd.print("              ");      // Очистка строки
347   lcd.setCursor(8, 1);              // Установка курсора в позицию (8, 1)
348   lcd.print(rtc.gettime("H:i, D")); // Получение времени
349 }
350
351 // Вывод даты
352 void printDate()
353 {
354   lcd.setCursor(8, 2);              // Установка курсора в позицию (8, 2)
355   lcd.print("              ");      // Очистка строки
356   lcd.setCursor(8, 2);              // Установка курсора в позицию (8, 2)
357   lcd.print(rtc.gettime("d.m.y"));  // Получение даты
358 }
359
360 // Установка часов
361 void setHours()
362 {
363   int pressedButton = 0;            // Нажатая кнопка
364   do
365   {
366     delay(10);                      // Задержка для плавности изменения
367     int hours = map(analogRead(REGULATOR), 0, 1023, 0, 24);
368     if(hours == 24)
369       hours = 0;
```

```
370     rtc.settime(rtc.gettime("s"),  // Секунды
371                 rtc.gettime("i"),  // Минуты
372                 hours,             // Часы
373                 rtc.gettime("d"),  // День месяца
374                 rtc.gettime("m"),  // Месяц
375                 rtc.gettime("y"),  // Год
376                 rtc.gettime("D")); // День недели
377     printTime();                   // Вывод времени
378     pressedButton = getPressedButton();   // Получения нажатой кнопки
379   } while(pressedButton != WHITE_BUTTON); // Пока не подтвердили изменения
380 }
381 // Установка минут
382 void setMinutes()
383 {
384   int pressedButton = 0;           // Нажатая кнопка
385   do
386   {
387     delay(10);                     // Задержка для плавности изменения
388     int minutes = map(analogRead(REGULATOR), 0, 1023, 0, 60);
389     if(minutes == 60)              // Корректировка выбранного значения
390       minutes = 0;
391     rtc.settime(rtc.gettime("s"),  // Секунды
392                 minutes,           // Минуты
393                 rtc.gettime("H"),  // Часы
394                 rtc.gettime("d"),  // День месяца
395                 rtc.gettime("m"),  // Месяц
396                 rtc.gettime("y"),  // Год
397                 rtc.gettime("D")); // День недели
398     printTime();                   // Вывод времени
399     pressedButton = getPressedButton();   // Получение нажатой кнопки
400   } while(pressedButton != WHITE_BUTTON); // Пока не подтвердили изменения
401 }
402
403 // Установка дня месяца
404 void setMonthDay()
405 {
406   int pressedButton = 0;           // Нажатая кнопка
407   do
408   {
409     delay(10);                     // Задержка для плавности изменения
410     int monthDay = map(analogRead(REGULATOR), 0, 1023, 1, 32);
411     if(monthDay == 32)             // Корректировка выбранного значения
412       monthDay = 1;
413     rtc.settime(rtc.gettime("s"),  // Секунды
414                 rtc.gettime("i"),  // Минуты
415                 rtc.gettime("H"),  // Часы
416                 monthDay,          // День месяца
417                 rtc.gettime("m"),  // Месяц
418                 rtc.gettime("y"),  // Год
419                 rtc.gettime("D")); // День недели
420     printDate();                   // Вывод даты
421     pressedButton = getPressedButton();   // Получение нажатой кнопки
422   } while(pressedButton != WHITE_BUTTON); // Пока не подтвердили изменения
423 }
424
425 // Установка месяца
426 void setMonth()
427 {
428   int pressedButton = 0;           // Нажатая кнопка
429   do
430   {
431     delay(10);                     // Задержка для плавности изменения
432     int month = map(analogRead(REGULATOR), 0, 1023, 1, 13);
```

```
433     if(month == 13)                // Корректировка выбранного значения
434       month = 1;
435     rtc.settime(rtc.gettime("s"),  // Секунды
436                 rtc.gettime("i"),  // Минуты
437                 rtc.gettime("H"),  // Часы
438                 rtc.gettime("d"),  // День месяца
439                 month,             // Месяц
440                 rtc.gettime("y"),  // Год
441                 rtc.gettime("D")); // День недели
442     printDate();                   // Вывод даты
443     pressedButton = getPressedButton();   // Получение нажатой кнопки
444   } while(pressedButton != WHITE_BUTTON); // Пока не подтвердили изменения
445 }
446
447 // Установка года
448 void setYear()
449 {
450   int pressedButton = 0;           // Нажатая кнопка
451   do
452   {
453     delay(10);                     // Задержка для плавности изменения
454     int year = map(analogRead(REGULATOR), 0, 1023, 0, 100);
455     if(year == 100)                // Корректировка выбранного значения
456       year = 99;
457     rtc.settime(rtc.gettime("s"),  // Секунды
458                 rtc.gettime("i"),  // Минуты
459                 rtc.gettime("H"),  // Часы
460                 rtc.gettime("d"),  // День месяца
461                 rtc.gettime("m"),  // Месяц
462                 year,              // Год
463                 rtc.gettime("D")); // День недели
464     printDate();                   // Вывод даты
465     pressedButton = getPressedButton();   // Получение нажатой кнопки
466   } while(pressedButton != WHITE_BUTTON); // Пока не подтвердили изменения
467 }
468 // Установка дня недели
469 void setWeekDay()
470 {
471   int pressedButton = 0;           // Нажатая кнопка
472   do
473   {
474     delay(10);                     // Задержка для плавности изменения
475     int weekDay = map(analogRead(REGULATOR), 0, 1023, 0, 7);
476     if(weekDay == 7)               // Корректировка выбранного значения
477       weekDay = 6;
478     rtc.settime(rtc.gettime("s"),  // Секунды
479                 rtc.gettime("i"),  // Минуты
480                 rtc.gettime("H"),  // Часы
481                 rtc.gettime("d"),  // День месяца
482                 rtc.gettime("m"),  // Месяц
483                 rtc.gettime("y"),  // Год
484                 weekDay);          // День недели
485     printTime();                   // Вывод времени
486     pressedButton = getPressedButton();    // Получение нажатой кнопки
487   } while(pressedButton != WHITE_BUTTON);  // Пока не подтвердили изменения
488 }
489
490 // Функция получения нажатой кнопки
491 int getPressedButton()
492 {
493   int pressedButton = 0;                     // Нажатая кнопка
494   pressedButton = digitalRead(BLUE_BUTTON);  // Статус синей кнопки
495   if(pressedButton > 0)
```

```
496     return BLUE_BUTTON;
497   pressedButton = digitalRead(WHITE_BUTTON);  // Статус белой кнопки
498   if(pressedButton > 0)
499     return WHITE_BUTTON;
500   return 0;
501 }
```