УДК 004.9:681.3

В. Г. МИХАЙЛОВ

ЗАПИСЬ БИНАРНЫХ ДАННЫХ НА SD КАРТУ ARDUINO DUE

ООО «Мидивисана», г. Минск

*Дан краткий обзор микроконтроллеров семейства Arduino, их характеристик и областей применения. Отмечена важность записи параметров исследуемого объекта для отладки систем управления на микроконтроллерах Arduino. Единственной возможностью регистрации параметров в семействе Arduino является запись на SD-карту в текстовом режиме с использованием функций **print()**, **write()**. Рассмотрены проблемы, связанные с записью бинарных данных на SD-карту на микроконтроллере Arduino Due. Проведен анализ способов записи бинарных данных на SD-карту Arduino Due, возникающих проблем с неочисткой памяти от предыдущей программы, приводящей к возможности дублирования данных на SD-карте, наличие ошибочной точки зрения об ограничении объемов записи данных и необходимости использования устаревших SD-карт. Рассмотрены пути устранения отмеченных недостатков. Проведена оценка быстродействия различных подходов записи информации на SD-карту. На основании проведенных исследований предложен подход уплотнения записываемой информации за счет преобразования бинарных данных побайтно в символьный массив в коде ASCI без увеличения их объема и записи блоками по 240 байт. Это позволяет максимально использовать возможности стандартной функции **write()** Arduino и специфику организации памяти SD-карт и увеличить быстродействие более чем в 1100 раз по сравнению с записью в символьном виде по одному байту.*

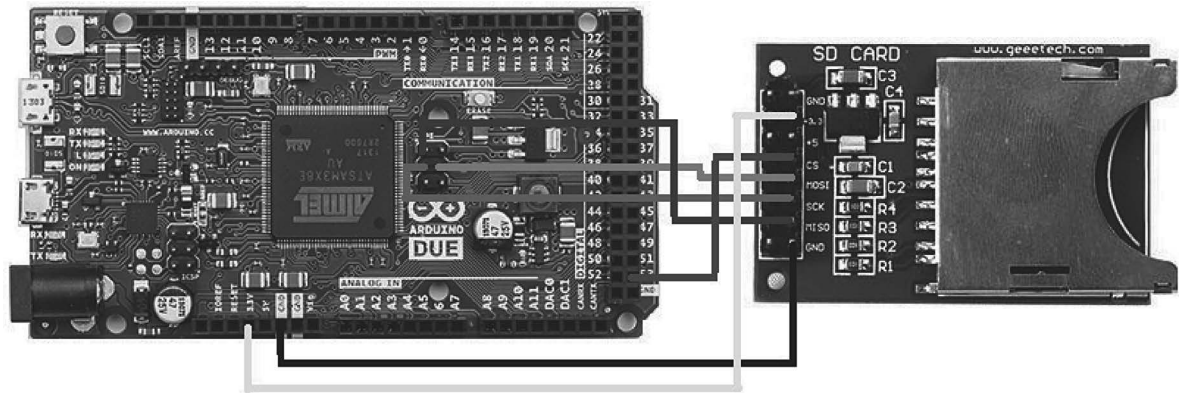
Отмечено, что использование предлагаемых на форумах решений исключения дублирования данных из-за неочистки памяти не обеспечивает полноты их устранения. Для Arduino Due для очистки памяти необходимо использования специального программатора или установка новой программы загрузки.

Ключевые слова: *микроконтроллер Arduino Due, SD-карта, запись бинарных данных, программа.*

Введение

Сейчас все более широко для задач автоматизации различных объектов применяются микроконтроллеры (МК) семейства Arduino [1, 2]. В следствии их производства в КНР, они имеют низкую стоимость (15–25 $), благодаря чему они получают все более широкое применение.

Линейка микроконтроллеров Arduino включает 8-ми, 32-ти разрядные модули. Они имеют малые габариты и малое потребление тока. И широко используются в различных системах управления игрушек, роботов, квадрокоптеров, систем умного дома и других устройств. Имеют неплохие перспективы для использования и в других областях. Наиболее производительным является Arduino Due, представляющий собой 32-ти разрядный микроконтроллер с тактовой частотой 84 МГц, показанный на рисунок.



Общий вид Arduino Due с SD-модулем и схема их соединения

Он имеет габариты 53x103x15 мм, масса – 35 г. В Arduino Due имеется 12 аналоговых входов АЦП и два аналоговых выхода 12-битного цифро-аналогового преобразователя. Необходимо также отметить, что плата Arduino Due имеет еще 54 цифровых вывода, которые могут работать в качестве входа или выхода сигналов, а также выводить 8-битные аналоговые значения в виде ШИМ-сигнала. Программирование модулей Arduino осуществляется с помощью языка C, C++ в среде Windows, Linux, Perl. Имеются большое количество различных датчиков и устройств для Arduino и бесплатных библиотек для них. Вместе с тем следует, отметить, что программные средства написаны в основном под 8-ми битные микроконтроллеры. Это не в полной мере пока позволяет использовать возможности 32-ти разрядного Arduino Due.

Кроме того для Arduino Due нет стандартного устройства (SD Shield), устанавливаемого сверху на плату. Из-за чего приходится использовать более простое устройство SD-карты с подключением проводами. Следует также отметить, что на Arduino Due не работает очистка памяти от предыдущей программы, что приводит к ее запуску и возможно дублирование данных в одноименных файлах.

Важную роль в процессе отладки систем управления на микроконтроллерах играет регистрация параметров исследуемого объекта, которую можно осуществить путем записи их на SD-карту. Однако запись бинарных данных на SD-карту [3–9] на микроконтроллерах представляет определенные трудности из-за ограниченных возможностей стандартной библиотеки SD, более рассчитанной на использование старых карт объемом 2, 4 Гб класса 2–4 и запись текстовой информации. А сейчас все больше распространены SD-карты (10 класс) стандарта SDHC и SDXC объемом соответственно до 32 Гб и более 64 Гб.

В Arduino запись на SD-карту в текстовом режиме производится с использованием функций **print()**, **write()** [3–8]. Максимальная длина записываемой строки соответственно 127, 255 байт. В тоже время многие регистрируемые параметры имеют бинарный вид (**int** 2 байта на 8-ми и 4 байта на 32 битных МК и **float, long** – 4 байта. Низкая скорость записи на SD-карту существенно ограничивает производительность системы управления. Если их (например, float) преобразовать в текстовый формат, то 4 байта **float** будут в среднем соответствовать 11–12 байтам текста с разделителем. И если необходимо произвести запись блока из 4 параметров, то реально в текстовом режиме можно записать блок кратный только 5-ти. В тоже время при использовании бинарных данных кратность могла быть равна 15-ти (при записи блока 240 байт).

Для управления быстродействующими объектами, такими как БЛА, квадрокоптеры важную роль играет быстродействие системы управления, зависящее, как от тактовой частоты микроконтроллера, его разрядности, так и времени записи на SD-карту. По данным работы [3], чтобы считать 6 байт из 3-х осевого акселерометра, библиотека гироскопа посылает адрес читаемого регистра и затем ждет. Шина I2C большинства гироскопов и акселерометров/магнетометров работает на частоте 100 КГц. В то же время сами сенсоры могут отдавать данные, например, 5000 раз в секунду. При такой частоте опроса сенсоров на 8-битном МК с рабочей частотой 16 МГц (Arduino Uno, Mega и другие) времени на обработку данных сенсоров, поступающих на огромной скорости практически не остается. При большой частоте опроса гироскопа или акселерометра Arduino можно использовать только как транслятор сигналов от сенсора через UART (аппаратно-последовательный порт) на внешний компьютер. Рассчитывать на выполнение каких-либо 32-ти битных вычислений, включая численное интегрирование и фильтрацию сигнала с датчиков ускорений гироскопа MPU6050 для определения угла наклона на самой Arduino (16 МГц, 8-ми битной) в такой ситуации не приходится. Поэтому возникла необходимость поиска путей повышения скорости записи на SD-карту, чтобы последняя не тормозила процесс управления быстродействующим объектом. Исходя из соображений быстродействия в своих исследованиях ориентировался на Arduino Due (84 МГц). Учитывалась тенденция, что в ближайшем будущем полностью произойдет переход на 32-разрядные микропроцессоры, чья большая вычислительная мощность позволит создавать новые интересные устройства, платы типа Arduino Due станут еще более востребованными.

С учетом всего этого в работе было акцентировано внимание на исследование путей записи бинарных данных на SD и возможности повышения производительности процесса записи, представляющих существенный практический интерес.

1. Особенности записи данных на SD-карты для МК Arduino

Особенностью записей на большинство SD-картах на МК, не использующих файловую систему, является специфическая организация их структуры хранения данных, при которой в начале ищется свободный блок (512 Кб либо 1024 Кб), в него вносятся данные и затем перезаписывается весь этот блок. С учетом этого целесообразно готовить и записывать данные на SD-карту блоком 512 Кб, а не по одному байту. Также следует учитывать, что используемые в Arduino функции **write()**, **print()** могут записывать информацию только в символьном виде. Функции **write()** может записать символьную строку объемом только 255 байт, а **print()** – 127 байт в коде ASCI. Это связано с использование в функции **size_t write (const uint8_t *buf, size_t size)** переменной **uint8_t *buf (**8-ми битного числа). Исходя из этого вытекает, что информацию целесообразно записывать с помощью функции **write()** блоком максимальным близким к 255 байт. Запись по одному байту, как предлагается в работах [7–10] нерациональна и непроизводительна. Она не обеспечивает требуемое минимальное время регистрации данных и управления быстродействующим объектом, например, квадрокоптером – 26 ms. И тем более 4-х, 6-ти, 8-ми параметров. Поэтому рассмотрим возможные пути уплотнения информации и повышения быстродействия записи на SD-карте, тем более что большинство регистрируемых параметров носят бинарный характер, а символьную информацию или режим можно закодировать в цифровом виде.

2. Запись информации в символьном виде

Для записи информации в символьном виде на один параметр (float, long) требуется 11–12 символов с разделителем. Отсюда получается, в строке 255 байт можно записать только 5 блоков по 4 параметра (4x11–12 = 220–240 символов). Наиболее худшим вариантом является запись по одному байту, что снижает производительность более чем в 1100 раз, т. к. для каждого символа требуется вначале найти блок, изменить его, а затем перезаписать. Это не позволяет использовать такой подход при регистрации данных и управления квадрокоптером, БЛА, роботом, станками.

3. Запись информации в бинарном виде

Бинарные данные (float, long, int) на 32-разрядном Arduino Due занимают 4 байта. Более целесообразным представляется для них использовать запись в символьный массив с преобразованием бинарных данных путем считывания побайтно в коде ASCI, при котором float, long, int будет занимать те же 4 байта, а четыре параметра 16 байт. Благодаря чему можно сократить объем информации в 3 раза по сравнению с символьным видом. Дополнительно можно уплотнить запись, если записывать информацию блоками, максимально приближаясь к объему символьной строки 255 байт. Для float (long, int) это 240 байт. Преобразование можно реализовать с помощью разработанной функции **zfl_bs** (см. исходный текст программы).

При заполнении символьной строки до максимально возможного (240 байт), она записывается на SD-карту с помощью стандартной функции **write()** и цикл повторяется.

Для удобства преобразования данные организованы в виде структуры **struct**. Это позволяет облегчить их обработку, преобразование через ссылочный механизм, не увеличивая объема информации.

Записанный на SD-карте массив в дальнейшем может быть считан с помощью функции языка С **fread()** и занесен в структуру. И далее при необходимости соответствующий образом обработан и использован. Например, использован в качестве исходных данных и сравнения при моделировании на Matlab/Simulink. И таким образом может быть проверена и отлажена система управления реальным объектом.

Программа реализации считывания 4-х параметров, записи их на SD-карту приведена ниже.

Алгоритм программы следующий: считываются данные с датчиков (в данном случае имитатора), их параметры заносятся в структуру, преобразуются и записываются в символьный массив блоками по 16 байт. При заполнении массива 240 байт производится запись на SD-карту и цикл повторяется. При достижении определенного объема файла, он

Таблица 1. **Программа записи 4-х значений float на SD-карту**

```
#include <SPI.h>
#include <SD.h>
#include <string.h>
#include <stdio.h>
struct dd {
float d1;
float d2;
float d3;
float d4;
byte bs[240];
int blz;
int kbz;
int zsb;
long kbf;
};
struct dd fd;
void zfl_bs(struct dd *d);
void zbl_sd(struct dd *d);
int j, ibl, kbf, kw,nf,mkf, kd;
//kf count write file
unsigned long time1, time2;
File mF;
char nfl[24];
char s1[24];
//------------------------------
void setup() {
//------------------
int i;

//----  Clear memory MC
/* Serial.begin(1200);
 while(Serial.available())
  Serial.read();
   Serial.end();
delay(5000);
*/

j=0;
nf=0; //number file
kbf=500; // count block in file
mkf=3; //  max count file
kw=0; // count write
fd.blz=240;
fd.kbz=16; // 4*4;
time1=0;
kd=0;
pinMode(53, OUTPUT);
//fd.bs[fd.blz]=0;
// Open serial communications and wait for port to open:
//---------------------
void loop() {
// my program
char s1[20];
int i;
//--------------------
//---------- incrise number file
while (j<=mkf)
{
if (kd==0)
{
  nfl[0]='\0';
itoa(j,nfl,10);
strcat(nfl,"test.txt");
mF = SD.open(nfl, FILE_WRITE);
kd=1;
Serial.println("File open");
if (j>0)
fd.d1=j;
}

//if (kw%kbf==0&&kw==kbf)
if (kw==kbf)
{
j++;
//nf=0;
mF.close();
Serial.println("File close");
kd=0;
kw=0;
time2 = millis();
Serial.print("Time write - ");
Serial.println(time2);
}
 fd.d1=1;
 fd.d2=2;
 fd.d3=3;
 fd.d4=4;
//------------
//fd.bs[0]='\0';
//fd.zsb=0;
//------------

for (i=0;i<ibl;i++)
{
fd.d1+=0.001;
fd.d2+=0.001;
fd.d3+=0.001;
fd.d4+=0.001;
zfl_bs(&fd);
```

```
 Serial.begin(250000);
 while (!Serial) {
    ; // wait for serial port to connect. Needed for
native USB port only
  }
  Serial.print("Initializing SD card...");
   if (!SD.begin(53)) {
    Serial.println("initialization failed!");
    return;
  }
  else
  Serial.println("initialization done.");
//while (!Serial);
time2=0;
ibl=fd.blz/fd.kbz;
sprintf(s1,"%ld %ld %ld\n",ibl,fd.zsb,fd.blz);
Serial.print(s1);

//-------------
if (j==mkf)
{
time2 = millis();
Serial.print("Time write - ");
Serial.println(time2);
Serial.println("Record File done.");
 }
} // End Setup
```

```
}
zbl_sd(&fd);
//if (kj%kbf==0)
//mF.close();
//j++;
 }
//mF.close();
 }

//---------------------------
// Record float in struct bs
void zfl_bs(struct dd *d)
{
int i;
byte *yf, *ys;
yf=(byte *)&d->d1;
ys=(byte *)&d->bs[0] + d->zsb;
if (d->zsb!=d->blz)
{
for (i=0;i<d->kbz;i++)
*ys++=*yf++;
d->zsb= d->zsb + d->kbz;
}
else
{
d->bs[0]='\0';
d->zsb=0;
}
}

//----------------
// Record data in File
void zbl_sd(struct dd *d)
{
int i,t,m;
byte *yf, *ys;
byte z;
//bl=d->blz;
z=240;
 if (d->zsb==d->blz)
 {
mF.write((const uint8_t *)d->bs, z);
kw++;
} //  if (d->zsb==d->blz)
}
//---------------------------
```

```
#include <SPI. h>
#include <SD. h>
#include <string. h>
#include <stdio. h>
struct dd {
```

```
// ————————-
void loop() {
// my program
char s1[20];
int i;
```

```
float d1;
float d2;
float d3;
float d4;
byte bs[240];
int blz;
int kbz;
int zsb;
long kbf;
};
struct dd fd;
void zfl_bs(struct dd *d);
void zbl_sd(struct dd *d);
int j, ibl, kbf, kw, nf, mkf, kd;
// kf count write file
unsigned long time1, time2;
File mF;
char nfl[24];
char s1[24];
// ——————————
void setup() {
// ————————
int i;

// — Clear memory MC
/* Serial. begin(1200);
while(Serial. available())
Serial. read();
Serial. end();
delay(5000);
*/

j= 0;
nf= 0; // number file
kbf= 500; // count block in file
mkf= 3; // max count file
kw= 0; // count write
fd. blz= 240;
fd. kbz= 16; // 4*4;
time1 = 0;
kd= 0;
pinMode(53, OUTPUT);
// fd. bs[fd. blz]= 0;
// Open serial communications and wait for port to open:
Serial. begin(250000);
while (! Serial) {
; // wait for serial port to connect. Needed for native USB port only
}
Serial. print(«Initializing SD card...»);
// ————————
// ——— incrise number file
while (j<=mkf)
{
if (kd== 0)
{
nfl[0]='\0';
itoa(j, nfl,10);
strcat(nfl,»test. txt»);
mF = SD. open(nfl, FILE_WRITE);
kd= 1;
Serial. println(«File open»);
if (j>0)
fd. d1 =j;
}

// if (kw%kbf== 0&&kw==kbf)
if (kw==kbf)
{
j++;
// nf= 0;
mF. close();
Serial. println(«File close»);
kd= 0;
kw= 0;
time2 = millis();
Serial. print(«Time write – «);
Serial. println(time2);
}
fd. d1 = 1;
fd. d2 = 2;
fd. d3 = 3;
fd. d4 = 4;
// ———
// fd. bs[0]='\0';
// fd. zsb= 0;
// ———

for (i= 0; i<ibl; i++)
{
fd. d1 += 0.001;
fd. d2 += 0.001;
fd. d3 += 0.001;
fd. d4 += 0.001;
zfl_bs(&fd);
}

zbl_sd(&fd);
// if (kj%kbf== 0)
// mF. close();
// j++;
```

```
if (! SD. begin(53)) {
Serial. println(«initialization failed!»);
return;
}
else
Serial. println(«initialization done.»);
// while (! Serial);
time2 = 0;
ibl=fd. blz/fd. kbz;
sprintf(s1,»%ld%ld%ld\n», ibl, fd. zsb, fd. blz);
Serial. print(s1);

// ———-
if (j==mkf)
{
time2 = millis();
Serial. print(«Time write – «);
Serial. println(time2);
Serial. println(«Record File done.»);
}
} // End Setup
```

```
}
// mF. close();
}

// ————————-
// Record float in struct bs
void zfl_bs(struct dd *d)
{
int i;
byte *yf, *ys;
yf=(byte *)&d->d1;
ys=(byte *)&d->bs[0] + d->zsb;
if (d->zsb!=d->blz)
{
for (i= 0; i<d->kbz; i++)
*ys++=*yf++;
d->zsb= d->zsb + d->kbz;
}
else
{
d->bs[0]='\0';
d->zsb= 0;
}
}

// ————————
// Record data in File
void zbl_sd(struct dd *d)
{
int i, t, m;
byte *yf, *ys;
byte z;
// bl=d->blz;
z= 240;
if (d->zsb==d->blz)
{
mF. write((const uint8_t *)d->bs, z);
kw++;
} // if (d->zsb==d->blz)
}
// ————————-
```

закрывается, открывается новый файл с соответствующим номером и так далее. Такой подход позволяет повысить надежность регистрации данных и сохранить большую часть информации при сбоях аппаратуры.

4. Оценка эффективности предлагаемого способа записи данных на SD-карту

Исследование проводилось с помощью SD-карт 2 класса (512 кб, неизвестный производитель, 2004 г. вып), 4 класса (4 Гб, 133 mb/c, Transcend, 2008 г. вып.), 10 класса (32 Гб, 200 mb/c, Transcend, 2016 г. вып.).

Результаты предлагаемого способа записи бинарных данных на SD-карту сведены в таблицу. Здесь же для сравнения приведены данные по записям в виде символьной информации 4-х параметров и влияние типа SD-карты на быстродействие записи.

Таблица 2. **Сравнительные результаты записи на SD-карты**

Время записи файла 240000 байт, ms	В бинарном виде блоком 16 байт * 15, 60000 параметров, ms	В символьном виде, блоком 4 параметра (46 байт) * 5, 20000 параметров, ms	В символьном виде по одному байту, для файла 240000 байт, ms
На SD-карте 32 Гб			
Полное время, ms	2581	2620	1032400
Приведенное время записи на один параметр float, ms	0,043	0,131	47,3
На SD-карте 4 Гб			
Полное время, ms	7701	8020	3080400
Приведенное время записи на один параметр float, ms	0,128	0,401	141,2
На SD-карте 512 кб			
Полное время, ms	10258	10270	4103200
Приведенное время записи на один параметр float, ms	0,172	0,51	188

Как видно из таблицы наиболее эффективным способом является запись бинарных данных блоком кратным 15-ти. Приведенное время на один параметр составляет 0,043 ms. При символьной форме записи время одного параметра составляет 0,128 ms. При бинарном и символьном способах записи данных блоком время минимально и не нарущает управление быстродействующим объектом (например, квадрокоптером – 26 ms.).

Как было выяснено существенное влияние на время записи оказывает тип и класс используемой CD-карты. Наиболее лучшие результаты показывает SD-карта 10 класса. У SD-карты 2 класса (512 мб) наиболее низкие результаты. SD-карта 4 класса (4 Гб) имеет несколько лучшие результаты по сравнению с SD-картой 4 класса (4 Гб). И все же они (SD карты 2, 4 класса) считаю менее годятся для регистрации быстродействующих процессов управления (БЛА, квадрокоптерами и т. п.). Последние лучше использовать для регистрации более медленных процессов, например, станками, роботами.

Как показали проведенные эксперименты SDHC карты 10 класса (32 Гб, 200 ms/c) прекрасно работают с модулями Arduino. И непонятна ошибочная точка зрения, что для Arduino подходят только устаревшие SD-карты до 4 Гб, которые сейчас уже не выпускаются. Возможно причиной такого утверждения является неправильный подсчет объема памяти из-за ограничения значности типа long [6] при использовании стандартной программы CardInfo.ino, показывающей объем памяти менее 2 Гб на картах более 4 Гб. Необходимо также учитывать, что SDHC-карты рассчитаны на напряжение питания 3,3 В, в то время как устаревшие на 5 В. И чтобы не вывести из строя SD-карту необходимо питать ее от отдельного стабилизатора либо использовать делитель напряжения на сопротивлениях. У Arduino Due имеется собственный стабилизатор на 3,3 В, с помощью которого питался блок SD. На SDHC-карты можно записать до 32 Гб информации, что вполне достаточно для многих задач. SDXC-карты (> 64 Гб) на Arduino не работают.

Необходимо упомянуть об имеющейся нерешенной проблеме очистки памяти от ранее загруженной программы в Arduino Due [9]. На сайтах для очистки памяти от ранее загруженной предыдущей программы (применительно в основном для Arduino Uno) предлагалось:

- использование кнопки очистки памяти, удерживая ее 5 сек;
- введение в программу задержки delay (5000);
- запуск программы-пустышки;
- использовать USB-порт для программирования в силу некоторых особенностей процесса очистки памяти микроконтроллера.

Но на Arduino Due все это не работает. Единственным решением является использование специального программатора для очистки памяти либо обновление программы загрузки МК [3]. В качестве временного решения можно рекомендовать использование записи в различные каталоги для отладочной и рабочей программы.

Заключение

С модулями Arduino целесообразно использовать современные карты SDHC 10 класса, 32 Гб, которые обеспечивают наибольшее быстродействие и могут записывать до 32 Гб информации.

2. Устаревшие SD карт 2, 4 класса имеют более низкое быстродействие (в 3–4 раза ниже чем у SDHC карт).

3. Запись информации на SD карты в символьном виде менее эффективна. Она приводит к снижению быстродействия в 3 раза по сравнению с бинарным способом.

4. На основании проведенных исследований предлагается подход уплотнения записываемой информации за счет преобразования бинарных данных побайтно в символьный массив в коде ASCI без увеличения их объема и записи блоками по 240 байт. Это позволяет максимально использовать возможности стандартной функции write() Arduino и специфику организации памяти SD карт и увеличить быстродействие более чем в 1100 раз по сравнению с записью в символьном виде по одному байту.

5. Предложена и апробирована программа циклического считывания информации с датчиков и записи их параметров на SD-карты в бинарном виде, которая обеспечивает наибольшее быстродействие их записи и уплотнение информации.

Литература

1. **32-разрядные** платы Arduino [Электронный ресурс]. – 2016. – Режим доступа:http://mcucpu.ru/index.php/platformy-8-bit/arduino/95-arduino32bit – Дата доступа: 12.05.2016.

2. **Arduino** Due Общие сведения [Электронный ресурс]. – 2016. – Режим доступа: http://arduino.ua/ru/hardware/Due. – Дата доступа: 17.05.2016

3. **Материалы** форума Вопросы по гироскопу MPU 6050 [Электронный ресурс]. – 2010. – Режим доступа: http://forum.amperka.ru/threads/%D0%92%D0%BE%D0%BF%D1%80%D0%BE%D1%81%D1%8B-%D0%BF%D0%BE-%D0%B3%D0%B8%D1%80%D0%BE%D1%81%D0%BA%D0%BE%D0%BF%D1%83-mpu-6050.1467/. – Дата доступа: 22.05.2015..

4. **Модуль** SD карты и Arduino [Электронный ресурс]. – 2016. – Режим доступа: http://arduino-diy.com/arduino-SD-karta. – Дата доступа: 19.05.2016.

5. **Материалы** форума. Запись данных с датчика DHT11 на SD карту? [Электронный ресурс]. – 2016. – Режим доступа: http://arduino.ru/forum/programmirovanie/zapis-dannykh-s-datchika-dht11-na-sd-kartu. – Дата доступа: 25.10.2016.

6. **Подключение** и использование SD карты с Arduino [Электронный ресурс]. – 2016. – Режим доступа: https://uscr.ru/podklyuchenie-i-ispolzovanie-sd-karty-c-arduino/. – Дата доступа: 24.05.2016.

7. **Запись**/чтение на SD-карту с Arduino [Электронный ресурс]. – 2016. – Режим доступа: http://cxem.net/arduino/arduino2.php – Дата доступа: 24.05.2016.

8. **Подключение** SD карт к Arduino [Электронный ресурс]. – 2016. – Режим доступа: http://www.poprobot.ru/home/podkluceniesdkartkarduino – Дата доступа: 24.05.2016

9. **Проблема** загрузки в плату. Помощь по загрузке: [Электронный ресурс]. – 2016. – Режим доступа: http://www.arduino.cc/en/Guide/Troubleshooting#upload. – Дата доступа: 20.05.2016.

10. **Arduino:** Работаем с SD картами [Электронный ресурс]. – 2016. – Режим доступа: http://zhitenev.ru/arduino-rabotaem-s-sd-kartami/. – Дата доступа: 20.05.2016.

References

1. **32-bit** boards Arduino [the Electronic resource]. – 2016. – the Mode доступа:http://mcucpu.ru/index.php/platformy-8-bit/arduino/95-arduino32bit. – Access Date: 5/12/2016.

2. **Arduino** Due the general convergence [the Electronic resource]. – 2016. – the Mode доступа: http://arduino.ua/ru/hardware/Due – Access Date: 5/17/2016.

3. **Materials** of a forum Questions on gyroscope MPU 6050 [the Electronic resource]. – 2010. – the Access mode: http://forum.amperka.ru/threads/%D0%92%D0%BE%D0%BF%D1%80%D0%BE%D1%81%D1%8B-%D0%BF%D0%BE-%D0%B3%D0%B8%D1%80%D0%BE%D1%81%D0%BA%D0%BE%D0%BF%D1%83-mpu-6050.1467/. – Access Date: 5/22/2015.

4. **SD** unit of a card and Arduino [the Electronic resource]. – 2016. – the Mode доступа: http://arduino-diy.com/arduino-SD-karta. – Access Date: 5/19/2016.

5. **Materials** of a forum Data record from sensor DHT11 on SD a card? [An electronic resource]. – 2016. – the Mode доступа: http://arduino.ru/forum/programmirovanie/zapis-dannykh-s-datchika-dht11-na-sd-kartu. – Access Date: 10/25/2016.

6. **Connection** and usage SD of a card with Arduino [the Electronic resource]. – 2016. – the Access mode: https://uscr.ru/podklyuchenie-i-ispolzovanie-sd-karty-c-arduino/. – access Date: 5/24/2016.

7. **Record**/reading on a SD-card with Arduino [the Electronic resource]. – 2016. – the Access mode: http://cxem.net/arduino/arduino2.php. – Access Date: 5/24/2016.

8. **Connection** SD of cards to Arduino [the Electronic resource]. – 2016. – the Access mode: <http://www.poprobot.ru/home/podkluceniesdkartkarduino>. – Access Date: 5/24/2016.

9. **A loading** problem in a board of Arduino. The help on loading: [the Electronic resource]. – 2016. – the Access mode: http://www.arduino.cc/en/Guide/Troubleshooting#upload. – Access Date: 5/20/2016.

10. **Arduino:** Working with SD Card: [the Electronic resource]. – 2016. – the Access mode: http://zhitenev.ru/arduino-rabotaem-s-sd-kartami/. – Access Date: 5/20/2016.

Поступила	*После доработки*	*Принята к печати*
06.09.2016	*12.09.2016*	*15.09.2016*

Vladimir Mikhailov

CAND. TECH. SCI. (ENGINEERING), LEADING ENGINEER OF OPEN STOCK CO «MIDIVISANA» MINSK, REPUBLIC BELARUS

The short review of microcontrollers of family Arduino, their characteristics and application fields is given. Importance of record of parameters of researched object is marked to produce debugging of control systems on microcontrollers Arduino. Unique possibility of registration of parameters in family Arduino is record on SD a card in an alpha mode with usage of functions ***print ()****,* ***write ()****. The problems connected to record of the binary data on SD a card on microcontroller Arduino Due are considered. The analysis of methods of record of the binary data on SD card Arduino Due, originating problems with neocleaning of memory from the previous program leading to possibility of duplication of the data on SD to a card, presence of the erratic point of view about restriction of volumes of data record and necessity of usage become outdated SD cards is carried out. Ways of elimination of the marked lacks are considered. The estimation of high-speed performance of various approaches of a data recording on SD a card is led. On the basis of the led researches the approach of multiplexing of the writeable information at the expense of conversion of the binary data is offered is byte-serial in a character array in code ASCI without magnification of their volume and record by units on 240 byte. It allows to use as much as possible standard function possibilities write () Arduino and specificity of the organization of memory SD of cards and to increase high-speed performance more than in 1100 times in comparison with record in a character type on one byte.*

It is marked that usage of decisions of an exception of duplication of the data offered at forums does not provide completeness of their elimination. For Arduino Due for storage cleaning it is necessary usages of the special programmator or setting of the new program of loading.

Keywords: *microcontroller Arduino Due, SD Card, record of the binary data, the program.*

Владимир Георгиевич Михайлов канд. техн. наук 05.05.03, 220005, Минск, ул. Пугачевская 24–8, ведущий инженер ООО «Мидивисана», г. Минск.

Специалист в области разработки систем CALS/PLM (PDM, ERP), программирования, автомобилестроения, моделирования динамических систем в пакетах MATLAB\SIMULIK, оценки напряженно-деформированного состояния в пакете ANSYS, испытаниям подвесок, рам ТС, пневматики, гидравлики, тензометрирования,

Tel.: + 375-(029)785–09–16. E-mail: sapr7@mail.ru.

Mikhailov Vladimir (S'72–M'76–SM'80) received the D. degree (Cand. Tech. Sci., mechanical Engineering) from the Belarussian Polytechnical institute of Belarus, Minsk, in 1982.

He was a Senior Research and Engineer-designer at Minsk Automobile plant, from 1972 to 1984, Leading Research, Chief of Research laboratory in CenterSystem, Minsk (design and development ERP) from 1984 to 1991, and was leading Engineer-designer at Minsk Wheel plant from 1994 to 2010, now a leading engineer of Open Stock Company «Midivisana», Republic Belarus, Minsk. His research interests include design and development of Software PDM, ERP, application Oracle on C++, PL/SQL, Java, programming MC, modeling dynamic systems, vibration.

The expert in the field of system engineering CALS/PLM (PDM, ERP), programming MC, motor industry, modeling of dynamic systems in packets MATLAB\SIMULIK (S-Function Builder), estimations of the intense-deformed state in packet ANSYS, to tests of suspension, frames of the vechicle, a pneumatics, hydraulics.