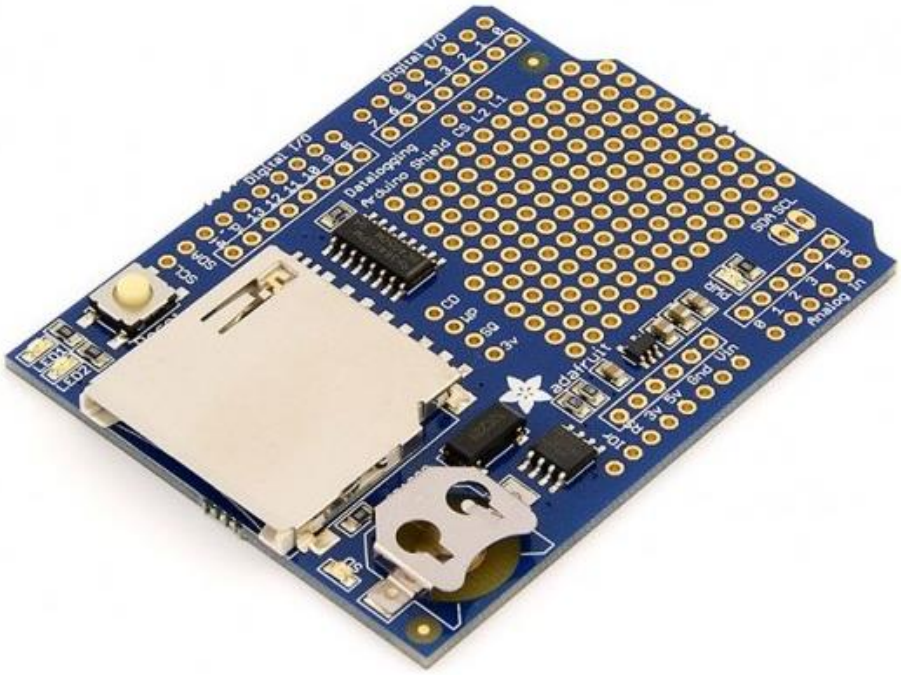
Модуль регистрации и хранения данных (плата дата логгера для Arduino)
Assembled Data Logging shield for Arduino



Модуль позволяет сохранять на SD карте результаты измерений дополнительно монтируемых датчиков. Модуль регистрации и хранения данных содержит микросхему часов реального времени, позволяющую фиксировать дату и время каждого измерения. На плате смонтирован держатель SD карты, контейнер для батареи питания. Батарея обеспечивает ход часов в течение нескольких лет. Макетное поле предназначено для монтажа датчиков и электрических цепей. Программное обеспечение Arduino, предоставляемое с сайта Adafruit, позволяет сохранять файлы данных на карте памяти в формате пригодном для программы Excel. Производится фирмой Adafruit.
Устройство дает возможность исследовать параметры различных процессов, развивающихся на протяжении длительного времени. Например, можно узнать какова температура в нескольких точках блока питания, работающего с максимальной гарантированной нагрузкой в течение недели и проверить эффективность средств охлаждения. Полученные данные представляются в виде файла стандартного формата, который можно на персональном компьютере преобразовать в график средствами Microsoft office.
Модуль найдет широкое применение при проведении испытаний техники и в научных экспериментах www.arduino-kit.ru .

Характеристики

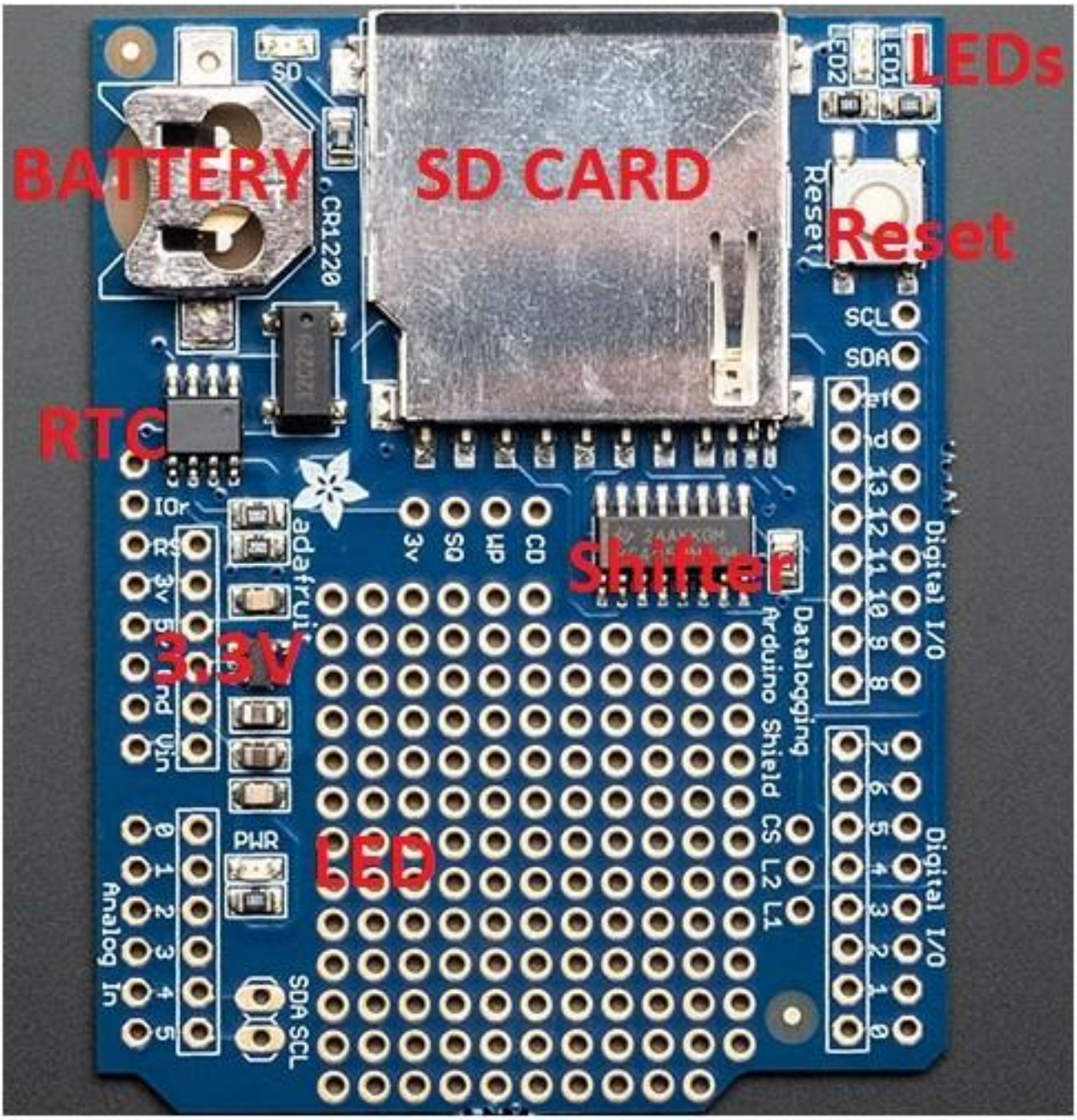
Напряжение питания 5 В
Размеры 70 х 53 х 17 мм
SD карта выступает на 10 мм

Особенности

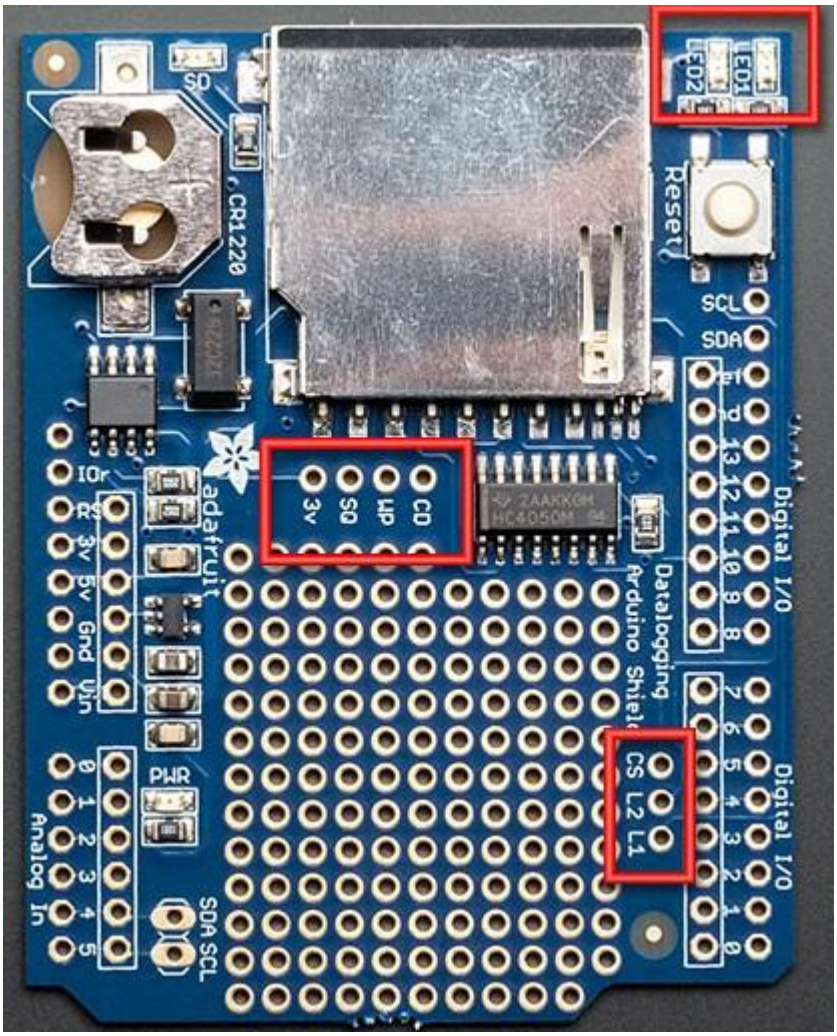
Модуль регистрации и хранения данных работает с картами памяти отформатированными в FAT16 или FAT32. Напряжение питания для SD карты 3,3 В обеспечивает микросхема RT9193-33 интегральный стабилизатор постоянного напряжения. Интересные особенности заключаются также в наличии часов реального времени (RTC) работают при отключении основного питания благодаря батарее. Есть программные библиотеки с примерами кода для карты памяти и часов реального

времени (RTC). Размещены 2 светодиода для использования на усмотрение инженера-исследователя. Последовательно со светодиодами включены резисторы 470 Ом.

Обзор



На фотографии модуля вверху слева виден контейнер батареи. Под ним [микросхема DS1307](#) часов реального времени (RTC), рядом с ней более крупный компонент – кварцевый резонатор. Ниже слева стабилизатор напряжения 3,3 В, еще ниже зеленый светодиод PWR, который светится при подаче питания. Вверху справа светодиоды для различного применения. Вверху в центре большой держатель для SD карты SD/MMC до 32 Гбайт. Если у вас micro-SD карта, то для ее установки применяется адаптер.



Кнопка Reset сбрасывает всю систему на базе центрального модуля Arduino, в которую включен модуль регистрации и хранения данных.
Под контейнером карты памяти расположены контакты, на которые выведены следующие сигналы.
3В – это выход стабилизатора напряжения 3,3 В, позволяет подключать нагрузку с потреблением тока до 50 мА.
SQ – прямоугольные импульсы с выхода часов. В программе должна быть специальная команда, чтобы включить этот сигнал. Используется в основном для тестирования.
WP – Write Protect сигнал логического уровня, сообщающий о наличии запрета записи карте памяти.
CD – низкий уровень сигнала сообщает о том, что модуль регистрации и хранения содержит карту памяти в держателе. Для этого сигнала рекомендуется использовать внутренние резисторы подтягивания к питанию внутри микроконтроллера.
В нижней части группа из трех контактов.
CS – сигнал интерфейса SD карты выбор устройства.
L2 и L1 – управление светодиодами, расположенными вверху. L1 для подключения LED1 и L2 для подключения LED2. Светодиоды работают при подаче высокого логического уровня.

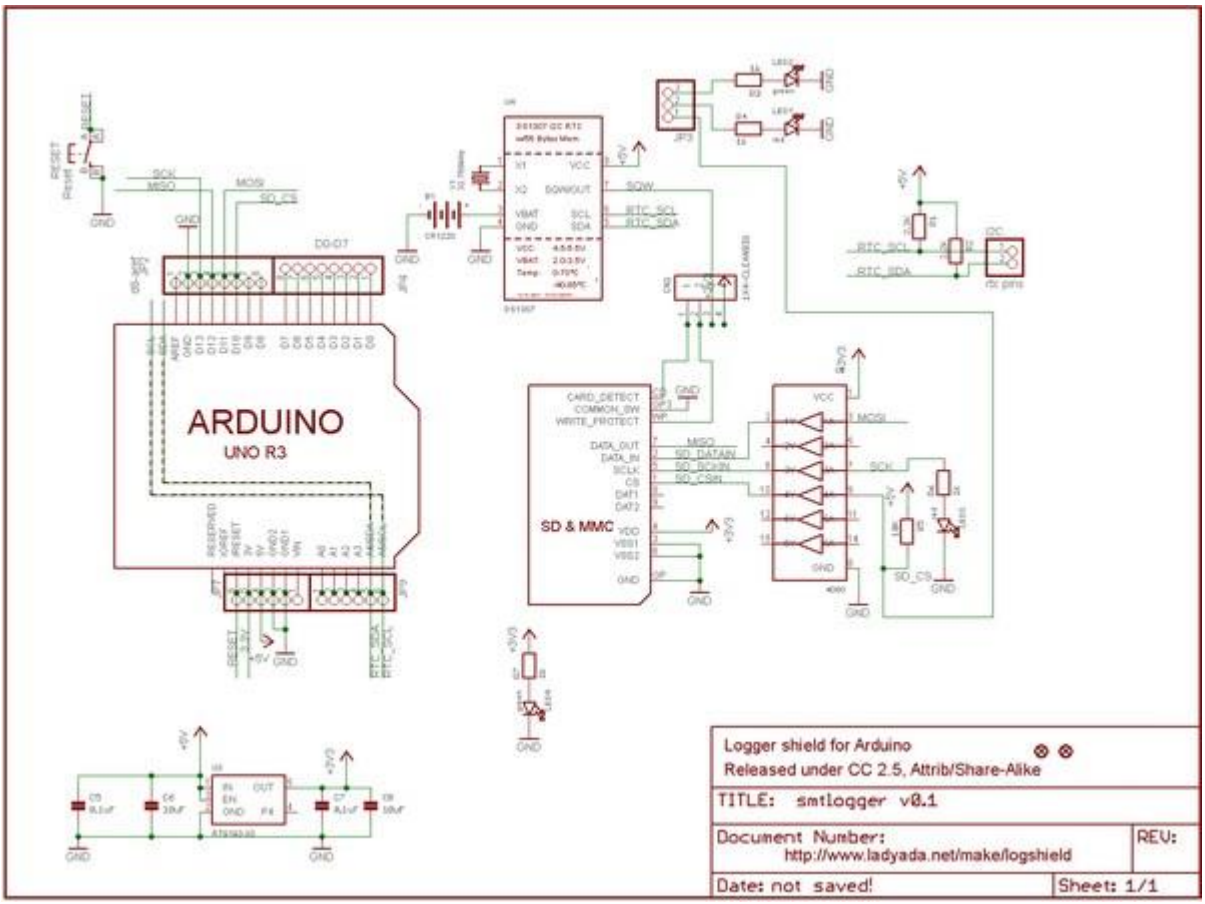
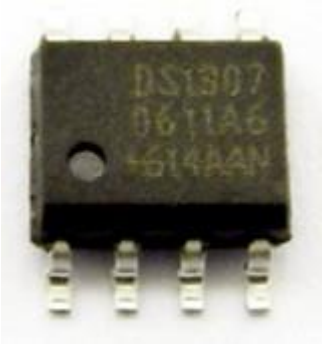


Схема модуля и подключение к Arduino UNO R3.

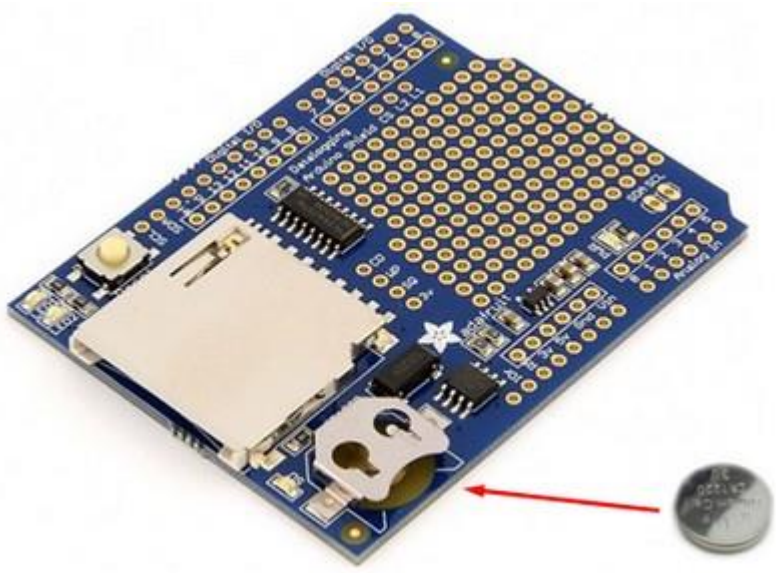
Использование часов реального времени

При сохранении данных события полезно знать дату и время произошедшего события. Arduino имеет встроенный хронометр, называемый Millis() и таймеры, встроенные в микроконтроллер, которые могут отслеживать длительные периоды времени. Зачем нужна отдельная микросхема RTC? Главная причина в том, что Millis() только отсчитывает временной интервал, а когда питание исчезает и появляется, таймер устанавливается на 0. Как у дешевых будильников: каждый раз, когда пропадает питание, они мигают и показывают 12:00.



Микросхема часов реального времени.

Пока работает батарея часы идут, их показания могут быть считаны, откорректированы или изменены в соответствии с часовым поясом.



Обмен данными с RTC

RTC имеет интерфейс I2C, который использует две информационные линии. Главный модуль Arduino имеет сигналы I2C выведенные на контакты 4 и 5, имеющие второе назначение – прием аналоговых сигналов. Пока начнем с этого, а позже разберемся, как изменить ситуацию. Для библиотеки RTC будем использовать библиотеку JeeLab в RTC. Ее скачать можно здесь. Затем установите ее в Arduino каталог, в папку под названием RTClib.

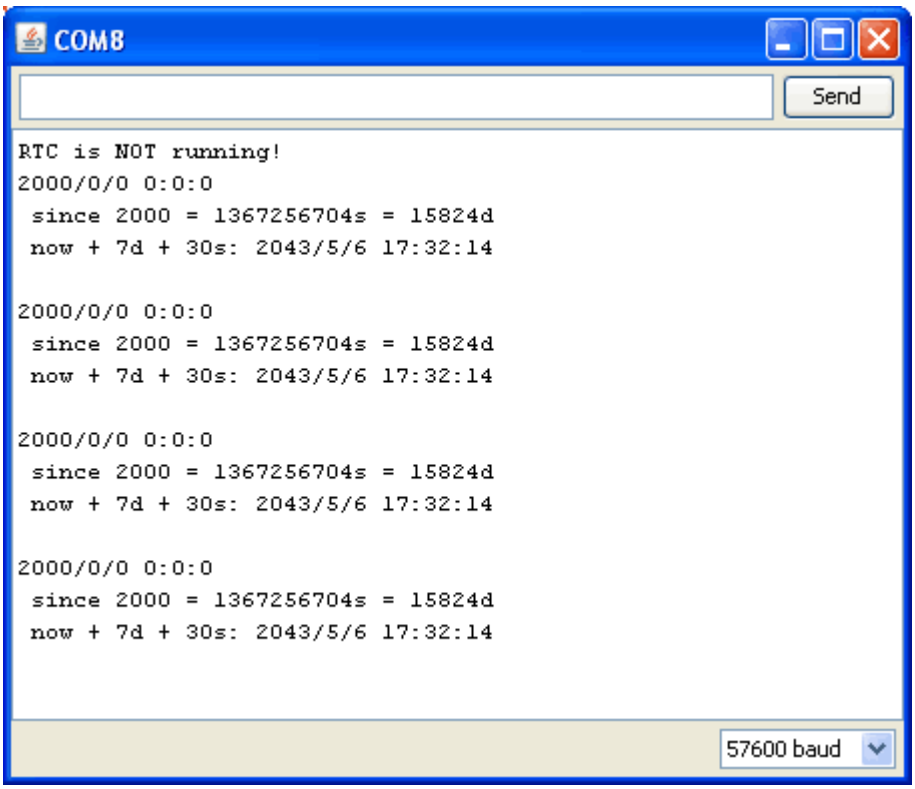
Проверка часов реального времени

Посмотрите скетч, который будет читать время от RTC раз в секунду. Чтобы начать, выньте батарею из контейнера. Подождите 3 секунды, а затем установите батарею. Это сбрасывает RTC. Теперь загрузите следующий скетч, который находится в Examples->RTClib->ds1307 в Arduino с установленным поверх модулем регистратора.

```
/ Date and time functions using a DS1307 RTC connected via I2C and Wire lib

#include
#include "RTClib.h"

RTC_DS1307 RTC;

void setup () {
 Serial.begin(57600);
 Wire.begin();
 RTC.begin();

 if (! RTC.isrunning()) {
 Serial.println("RTC is NOT running!");
 // following line sets the RTC to the date & time this sketch was compiled
 // uncomment it & upload to set the time, date and start run the RTC!
 //RTC.adjust(DateTime(__DATE__, __TIME__));
 }

}
```

```
void loop () {
 DateTime now = RTC.now();

 Serial.print(now.year(), DEC);
 Serial.print('/');
 Serial.print(now.month(), DEC);
 Serial.print('/');
 Serial.print(now.day(), DEC);
 Serial.print(' ');
 Serial.print(now.hour(), DEC);
 Serial.print(':');
 Serial.print(now.minute(), DEC);
 Serial.print(':');
 Serial.print(now.second(), DEC);
 Serial.println();

 Serial.print(" since 1970 = ");
 Serial.print(now.unixtime());
 Serial.print("s = ");
 Serial.print(now.unixtime() / 86400L);
 Serial.println("d");

 // calculate a date which is 7 days and 30 seconds into the future
 DateTime future (now.unixtime() + 7 * 86400L + 30);

 Serial.print(" now + 7d + 30s: ");
 Serial.print(future.year(), DEC);
 Serial.print('/');
 Serial.print(future.month(), DEC);
 Serial.print('/');
 Serial.print(future.day(), DEC);
 Serial.print(' ');
 Serial.print(future.hour(), DEC);
 Serial.print(':');
 Serial.print(future.minute(), DEC);
 Serial.print(':');
 Serial.print(future.second(), DEC);
 Serial.println();

 Serial.println();
 delay(3000);
}
```

Теперь запустите последовательный терминал и убедитесь, что скорость передачи данных установлена 57600 бод. Вы увидите следующее:

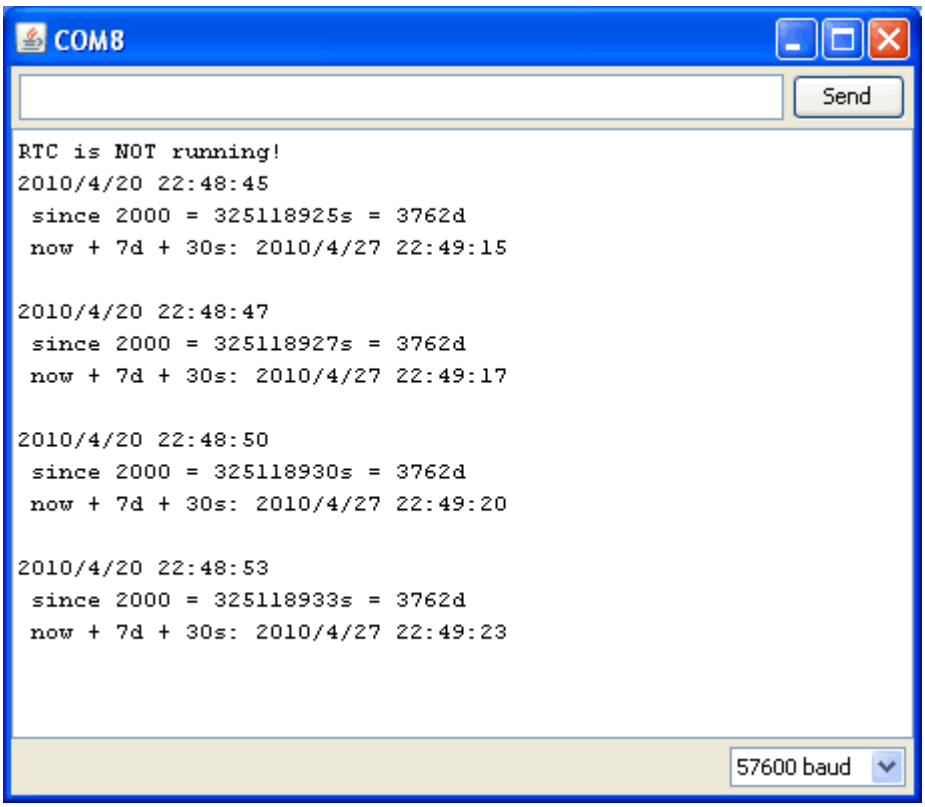


Установка времени

Если скетч загружен, то раскомментируйте строку, которая начинается с RTC.adjust:

// following line sets the RTC to the date & time this sketch was compiled
RTC.adjust(DateTime(__DATE__, __TIME__));

Эта строка позволяет принять дату и время компьютера (справа при компиляции кода). Затем откройте окно последовательного монитора.



Теперь не придется устанавливать время: батарея будет поддерживать часы 5 и более лет.

Чтение времени

Теперь RTC идут. Давайте посмотрим на скетч снова.

```
void loop () {
 DateTime now = RTC.now();

 Serial.print(now.year(), DEC);
 Serial.print('/');
 Serial.print(now.month(), DEC);
 Serial.print('/');
 Serial.print(now.day(), DEC);
 Serial.print(' ');
 Serial.print(now.hour(), DEC);
 Serial.print(':');
 Serial.print(now.minute(), DEC);
 Serial.print(':');
 Serial.print(now.second(), DEC);
 Serial.println();
```

Получим время, используя RTClib. Вызовем now(), функция возвращает объект DateTime, который содержит год, месяц, день, час, минуту и секунду.
Существуют библиотеки RTC.year() и RTC.hour() чтобы получить год и час. Тем не менее, есть одна проблема. Если во время показаний часов 3:14:59 получить данные о часе, а потом получить данные о минутах поотдельности, то существует вероятность ошибки на час, так как в итоге может быть получено ошибочное время 3:14:00.
Мы также можем получить данные из объекта DateTime связавшись с UnixTime который подсчитывает количество секунд (не считая leapseconds), начиная с полуночи, 1 января 1970.

```
Serial.print(" since 2000 = ");
Serial.print(now.unixtime());
Serial.print("s = ");
Serial.print(now.unixtime() / 86400L);
Serial.println("d");
```

В одном дне 60*60*24=86400 секунд. Исходя из этого можно подсчитать дату и время. Это может быть полезно, если вы хотите отслеживать, сколько времени прошло с момента последнего запроса данных от датчиков.

Использование SD карты

Микроконтроллер главного модуля Arduino способен долговременно хранить данные в EEPROM. Это все лишь пара сотен байт – крохотный объем по сравнению с возможностями SD карты. Она хорошо подходит для длительного хранения большого объема данных. В комплект модуля SD карта не входит. www.arduino-kit.ru Для работы в модуле подойдет множество типов SD карт.

Micro CD карта и адаптер.

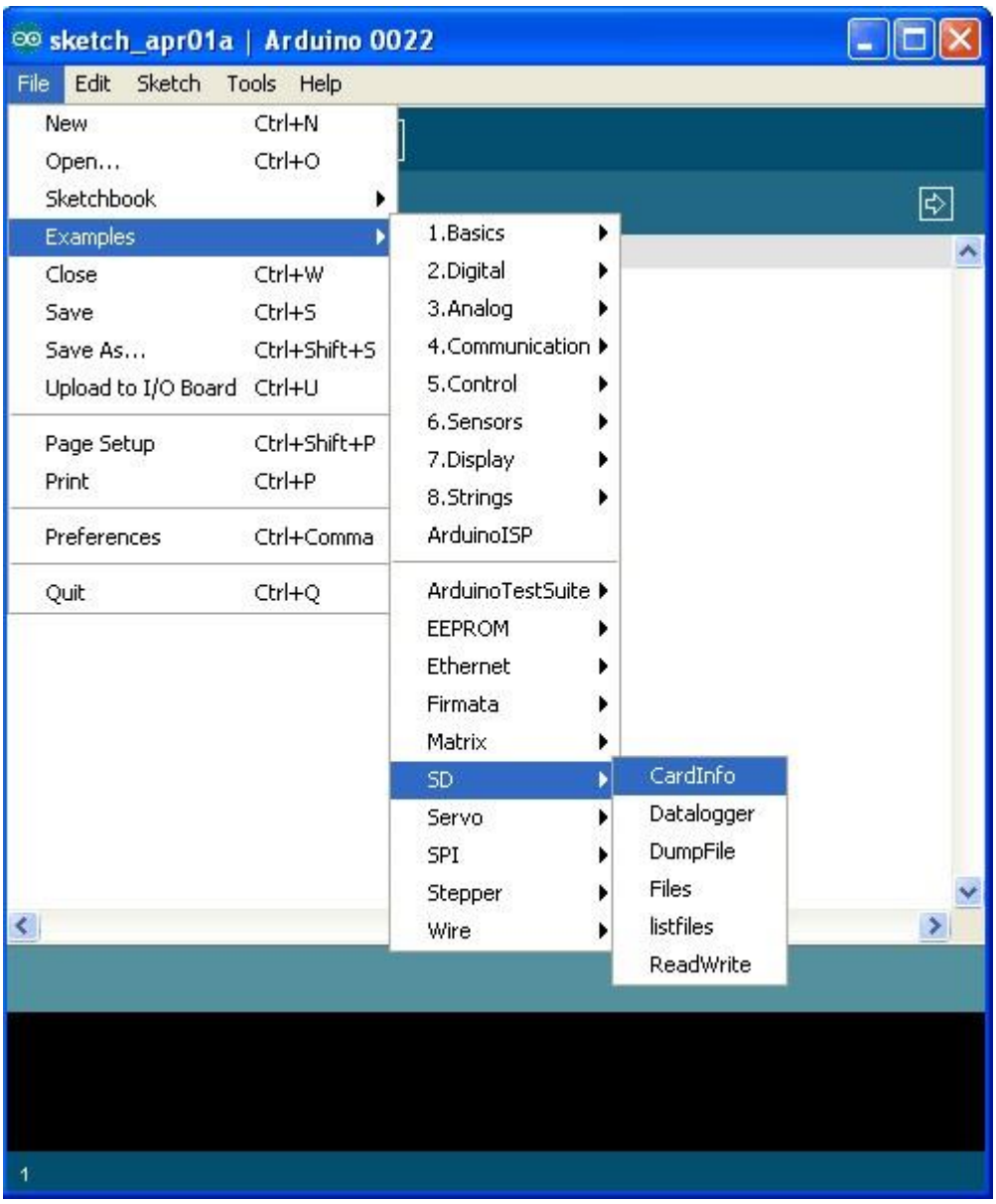
Есть возможность читать и записывать на карту. Вам потребуется ридер SD карты.

Форматирование для Windows и Mac

Если вы купили карту памяти, то скорее всего она отформатирована в FAT формате. Библиотека Arduino SD, которую мы используем, поддерживает файловые системы FAT16 и FAT32. Всегда лучше отформатировать карту перед использованием. Рекомендуем использовать официальную утилиту форматирования SD карты ассоциации SD.

Для пользователей Arduino Mega и Leonardo

Если Вы используете Arduino Mega и Leonardo, то придется обновить библиотеку SD карты добавив 'SD card on any pin' – поддержка карты через любые контакты. Для этого следуйте инструкциям на странице.
Затем выберите в меню File, в разделе Examples, пункт SD и далее CardInfo.

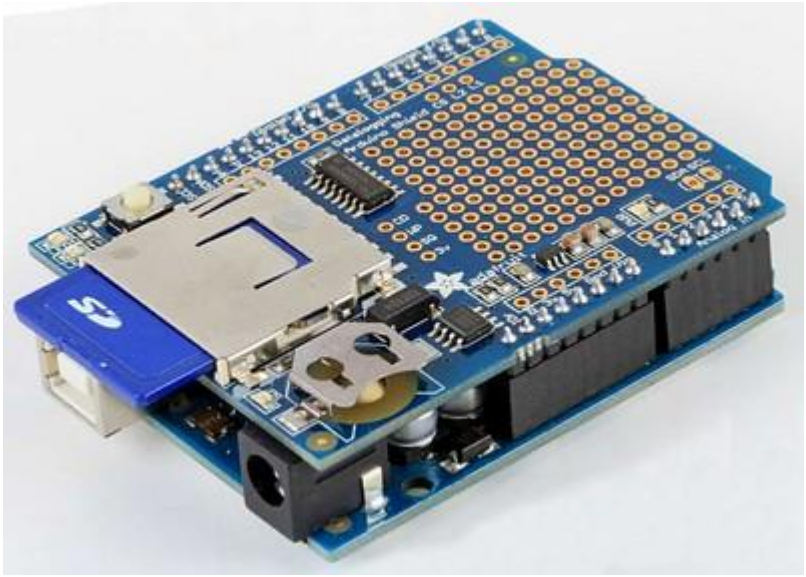


Этот скетч не будет сохранять данные на карту, а просто сообщит, удалось ли получить сведения о карте для проверки ее работоспособности. Перейдите к началу скетча и убедитесь, что строка chipSelect верная. Для модуля используется цифровой контакт, измените его номер на 10.

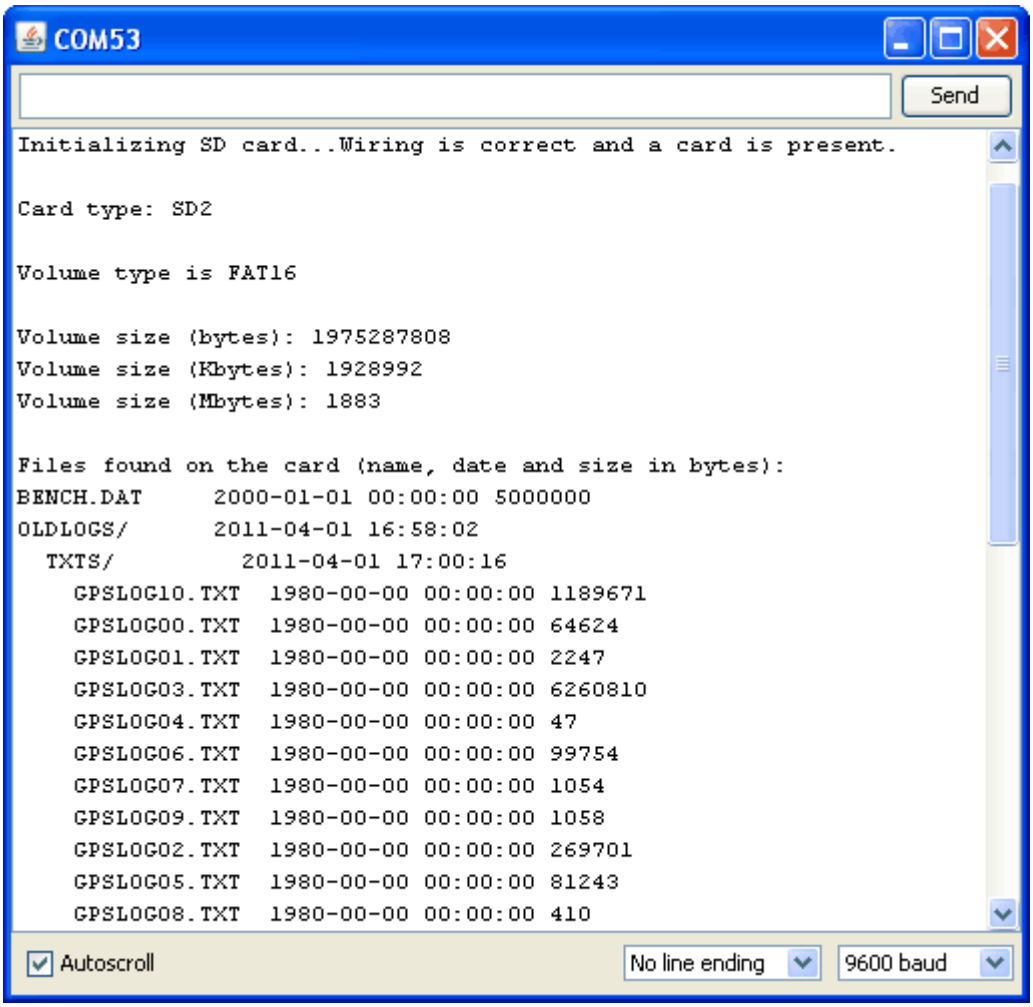
Для использования модуля регистрации и хранения совместно с Arduino Mega и Leonardo не забудьте изменить sd.begin(), чтобы задать номера контактов.

SD.begin (10,11,12,13);

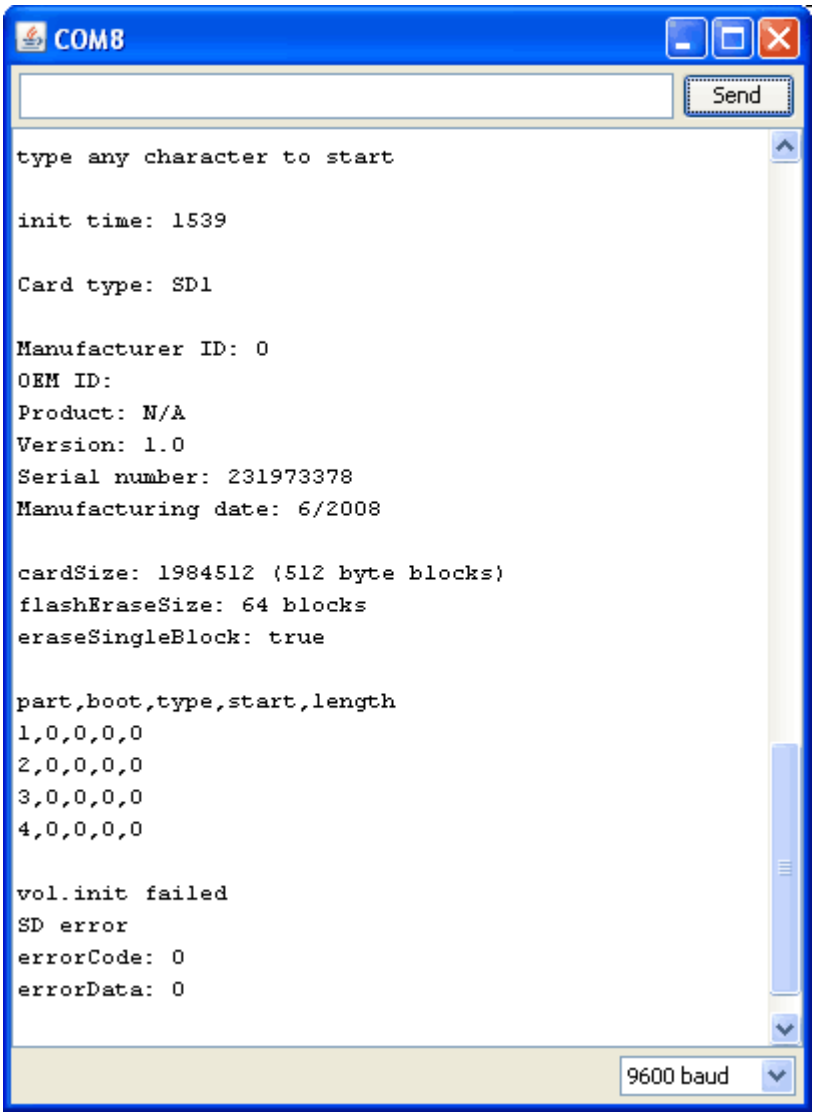
Вставьте SD карту в модуль регистрации и хранения данных и загрузите скетч.



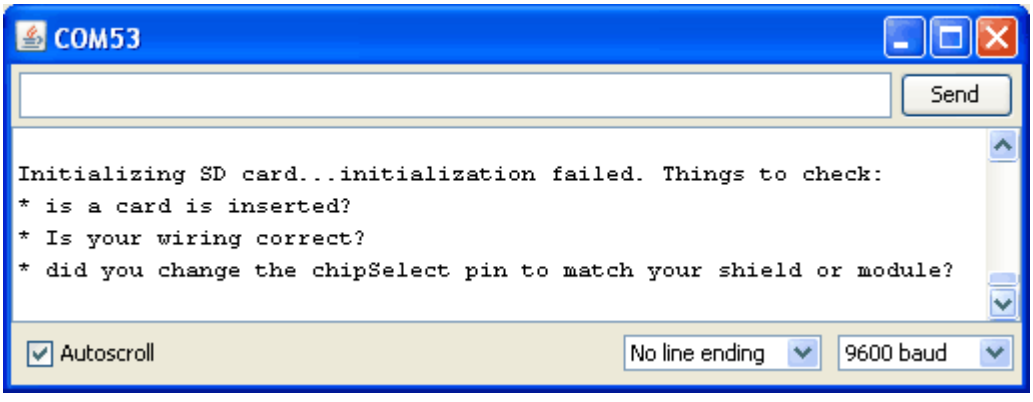
Откройте последовательный монитор и введите символ в текстовом поле и нажмите отправить. Вероятно получите что-то вроде:



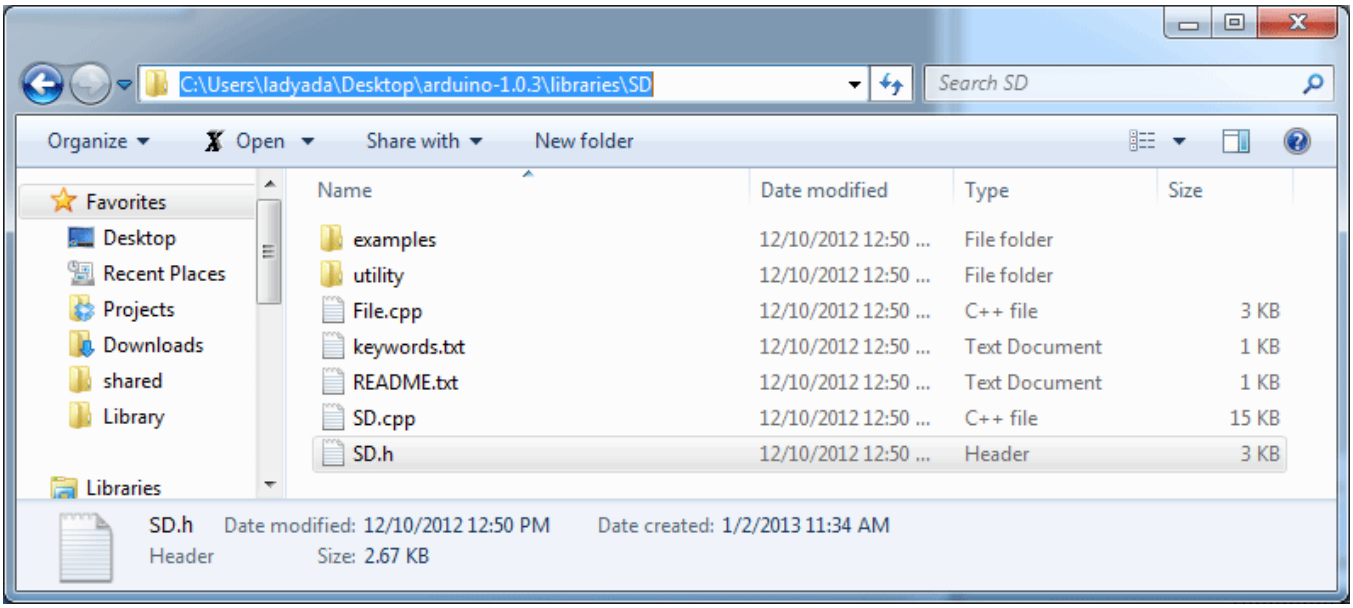
Эти данные содержат мало интересного, но их полезно посмотреть: Volume type is FAT16 – тип формата, а также размер карты и т.д. Если у вас плохая карта, то увидите:



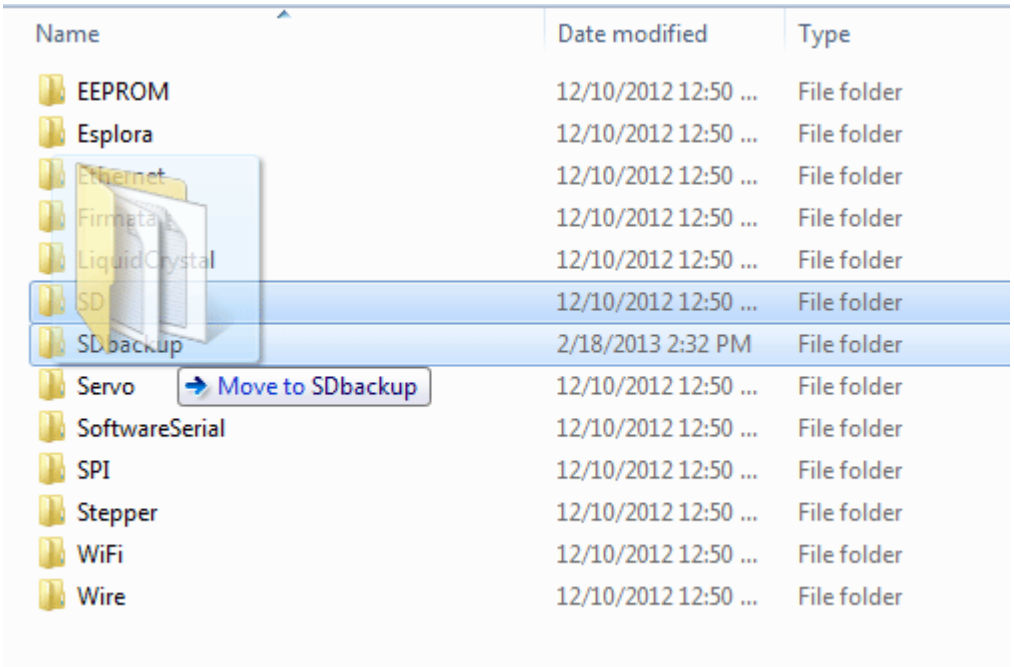
Выньте SD карту и снова запустите скетч, получите следующее:



Это также может произойти, если есть ошибка пайки или если карта действительно повреждена.
Если у вас Arduino Uno или Duemilanove или Diecimila не требуется обновлять библиотеку SD карты 'SD card on any pin' – поддержка карты через любые контакты.
Для Arduino Mega и Leonardo.
Во-первых, найдите папку основные библиотеки. Внутри увидите папку SD, внутри папки SD будет SD.cpp, SD.h и другое содержимое.



В папке библиотек, создайте новую папку под названием SDbackup. Затем перетащите SDfolder в SDbackup, чтобы "спрятать" старую библиотеку SD, не удаляя ее.



Теперь мы возьмем новую библиотеку SD здесь или здесь. Распакуйте и переименуйте несжатую папку SD. Убедитесь, что папка SD содержит SD.cpp и SD.h Поместите папку SD библиотека в своем альбоме папок библиотек. Возможно потребуется создать вложенные библиотеки. Для получения более подробной информации о том, как установить библиотеки нужно почитать здесь.

Использование библиотеки SD на Arduino Mega и Leonardo

Для Arduino Mega и Leonardo необходимо указать, какие контакты будут использоваться для интерфейса SPI карты. Для модуля они будут под номерами 10, 11, 12 и 13. Найдите в вашем скетче строку, содержащую SD.begin(), например:

```
// see if the card is present and can be initialized:
 if (!SD.begin(chipSelect)) {
```

и измените текст программы, указав номера контактов следующим образом:

```
// see if the card is present and can be initialized:
 if (!SD.begin(10, 11, 12, 13)) {
```

Информация о SD карте

Cardinfo скетч использует небольшую библиотеку на уровне общения с картой, она называется card.init() вместо SD.begin().

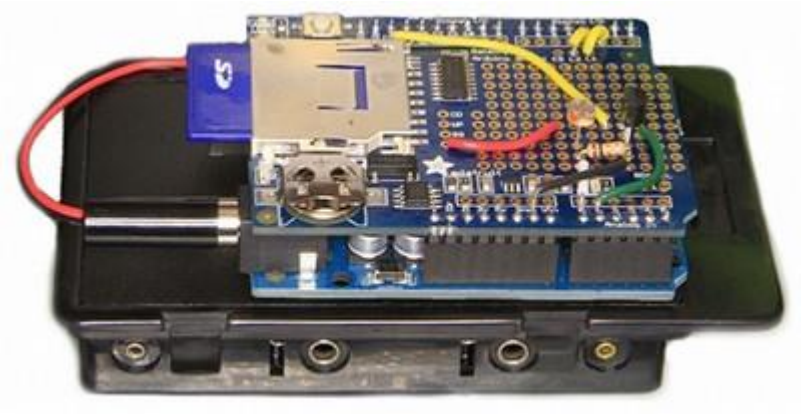
```
// we'll use the initialization code from the utility libraries
 // since we're just testing if the card is working!
 while (!card.init(SPI_HALF_SPEED, chipSelect)) {
```

При вызове card.init(), необходимо изменить предварительно заданные контакты SPI, а именно:

```
// we'll use the initialization code from the utility libraries
 // since we're just testing if the card is working!
 while (!card.init(SPI_HALF_SPEED, 10, 11, 12, 13)) {
```

Регистратор освещенности и температуры

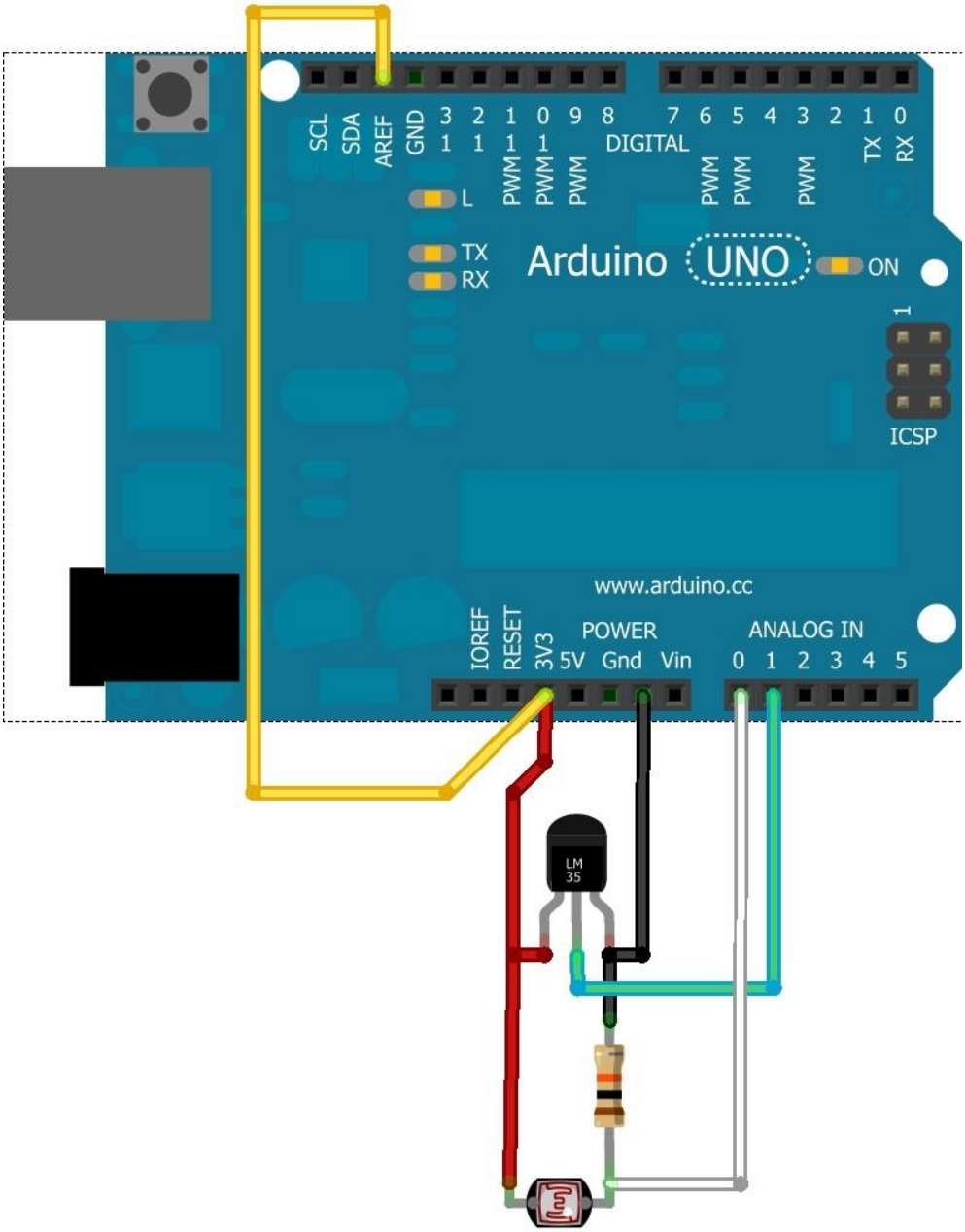
После освоения работы с часами и с SD картой можем перейти к регистрации данных. Будем фиксировать, что происходит в холодильнике. Как меняется температура в холодильнике, когда включается и выключается компрессор. Как влияет открывание двери на температуру? Сколько времени требуется на восстановление низкой температуры? Освещение внутри действительно отключается при закрывании двери?



Нам пригодится:
Arduino Uno R3. Модуль регистрации и хранения данных с SD-картой отформатированной для FAT и проверенный, используя наш пример, скетч CdS, фоторезистор и последовательно с ним включенный обычный резистор сопротивлением 10 кОм. Датчик температуры с аналоговым выходом, например, TMP36 Блок батарей.

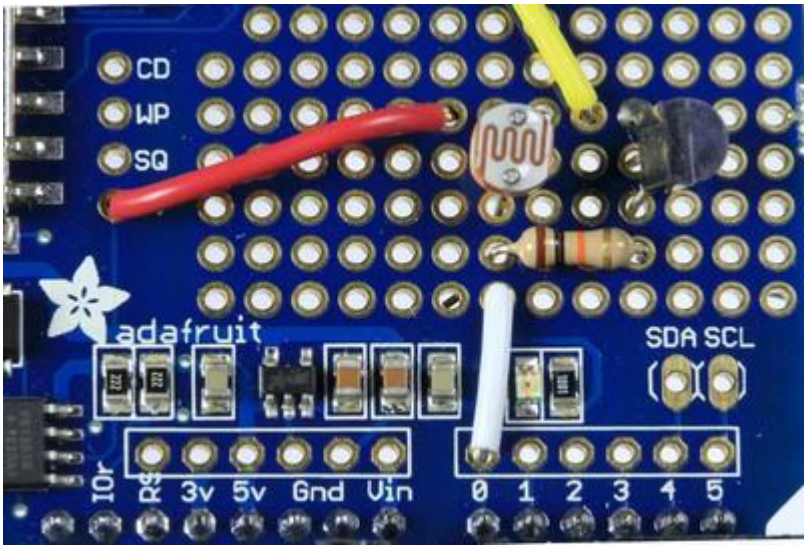
Датчики

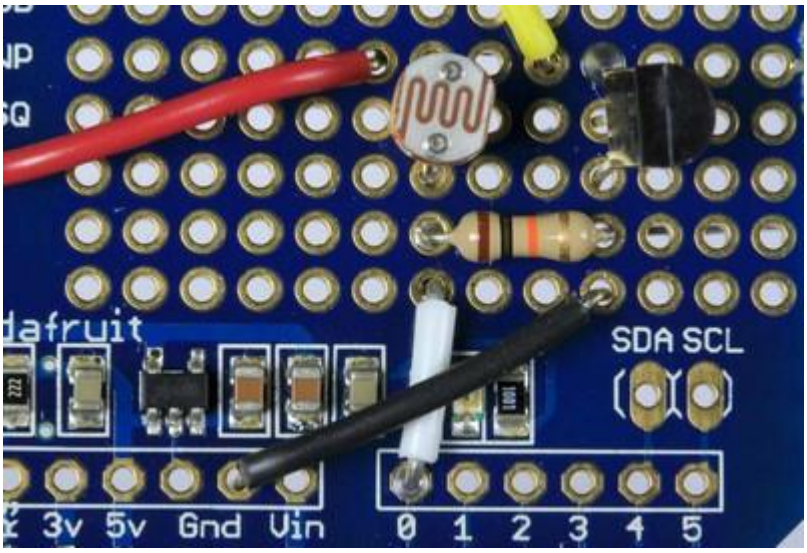
Будем использовать два датчика для регистрации данных. Adafruit рекомендует фоторезистор, имеющий сопротивление 200 кОм в темноте и 10 кОм при сильной освещенности. Проведем небольшое импортозамещение. В качестве датчика освещенности подойдут отечественные фоторезисторы ФСК-1 ФР-764 ФР-765 ФР1, ФР2, ФР3 и аналогичные.
Датчик температуры можно взять здесь http://arduino-kit.ru/catalog/id/datchik-temperaturyi-tmp36gt9z
Подключите датчики, как показано на рисунке. В качестве опорного потенциала для работы АЦП в микроконтроллере Arduino Uno R3 подаем на контакт AREF и одновременно для питания наших датчиков используем напряжение 3,3 В с платы Arduino Uno R3. В напряжении питания 5 В содержится множество шумов, всплесков и прочих помех мешающим датчикам достоверно фиксировать данные, поэтому его использовать крайне не рекомендуется.

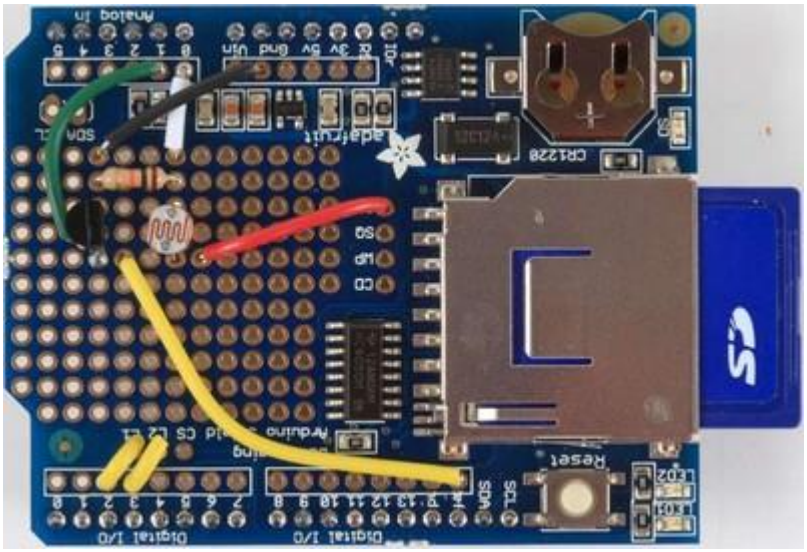


Сборка

Используем для монтажа отверстия платы. Будем использовать провода того же цвета, что и линии на схеме.







Провода соединений светодиодов.

От L1 к цифровому выводу 2 (желтый провод)
От L2 к цифровому выводу 3 (желтый провод)

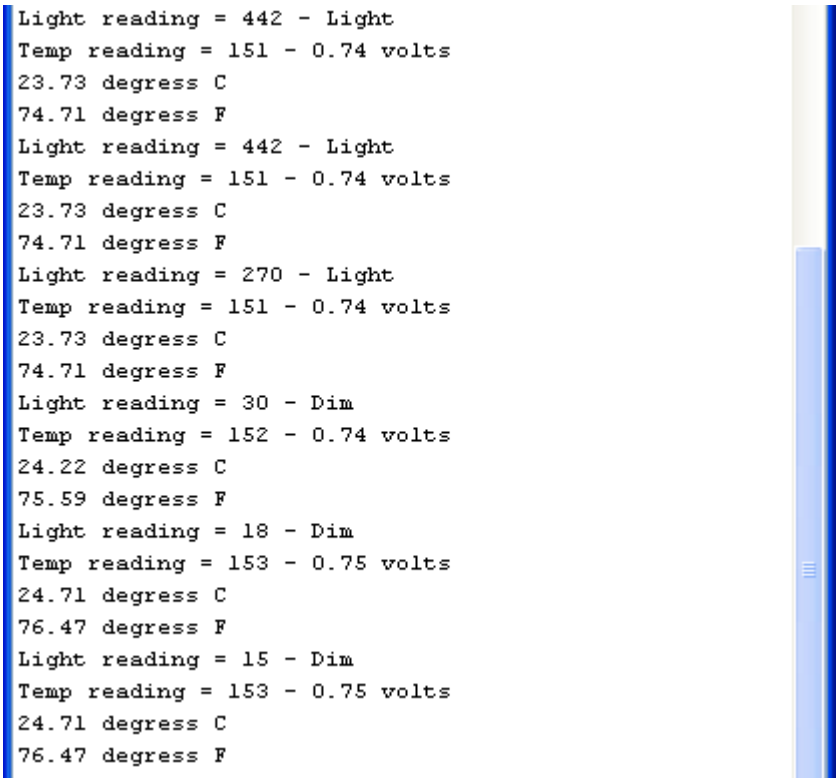
Проверка датчиков

Теперь проверим датчики, используя этот скетч

```
/* Sensor test sketch
 for more information see http://www.ladyada.net/make/logshield/lighttemp.html
 */

#define aref_voltage 3.3 // we tie 3.3V to ARef and measure it with a multimeter!

int photocellPin = 0; // the cell and 10K pulldown are connected to a0
int photocellReading; // the analog reading from the analog resistor divider

//TMP36 Pin Variables
int tempPin = 1; //the analog pin the TMP36's Vout (sense) pin is connected to
 //the resolution is 10 mV / degree centigrade with a
 //500 mV offset to allow for negative temperatures
int tempReading; // the analog reading from the sensor

void setup(void) {
 // We'll send debugging information via the Serial monitor
 Serial.begin(9600);

 // If you want to set the aref to something other than 5v
 analogReference(EXTERNAL);
}


void loop(void) {
 photocellReading = analogRead(photocellPin);

 Serial.print("Light reading = ");
 Serial.print(photocellReading); // the raw analog reading

 // We'll have a few threshholds, qualitatively determined
 if (photocellReading < 10) {
 Serial.println(" - Dark");
 } else if (photocellReading < 200) {
```

```
 Serial.println(" - Dim");
 } else if (photocellReading < 500) {
 Serial.println(" - Light");
 } else if (photocellReading < 800) {
 Serial.println(" - Bright");
 } else {
 Serial.println(" - Very bright");
 }

 tempReading = analogRead(tempPin);

 Serial.print("Temp reading = ");
 Serial.print(tempReading); // the raw analog reading

 // converting that reading to voltage, which is based off the reference voltage
 float voltage = tempReading * aref_voltage / 1024;

 // print out the voltage
 Serial.print(" - ");
 Serial.print(voltage); Serial.println(" volts");

 // now print out the temperature
 float temperatureC = (voltage - 0.5) * 100 ; //converting from 10 mv per degree wit 500 mV offset
 //to degrees ((volatge - 500mV) times 100)
 Serial.print(temperatureC); Serial.println(" degrees C");

 // now convert to Fahrenheight
 float temperatureF = (temperatureC * 9 / 5) + 32;
 Serial.print(temperatureF); Serial.println(" degrees F");

 delay(1000);
}
```

Загрузите эту программу и все проверьте, используя монитор последовательного порта.
Некоторые недавние версии IDE и библиотеки SD требуют, чтобы вы явно включили библиотеку SPI. Если вы получаете сообщение об ошибке компиляции "'SPI' was not declared in this scope", то просто добавьте "#include " в начале вашего скетча.



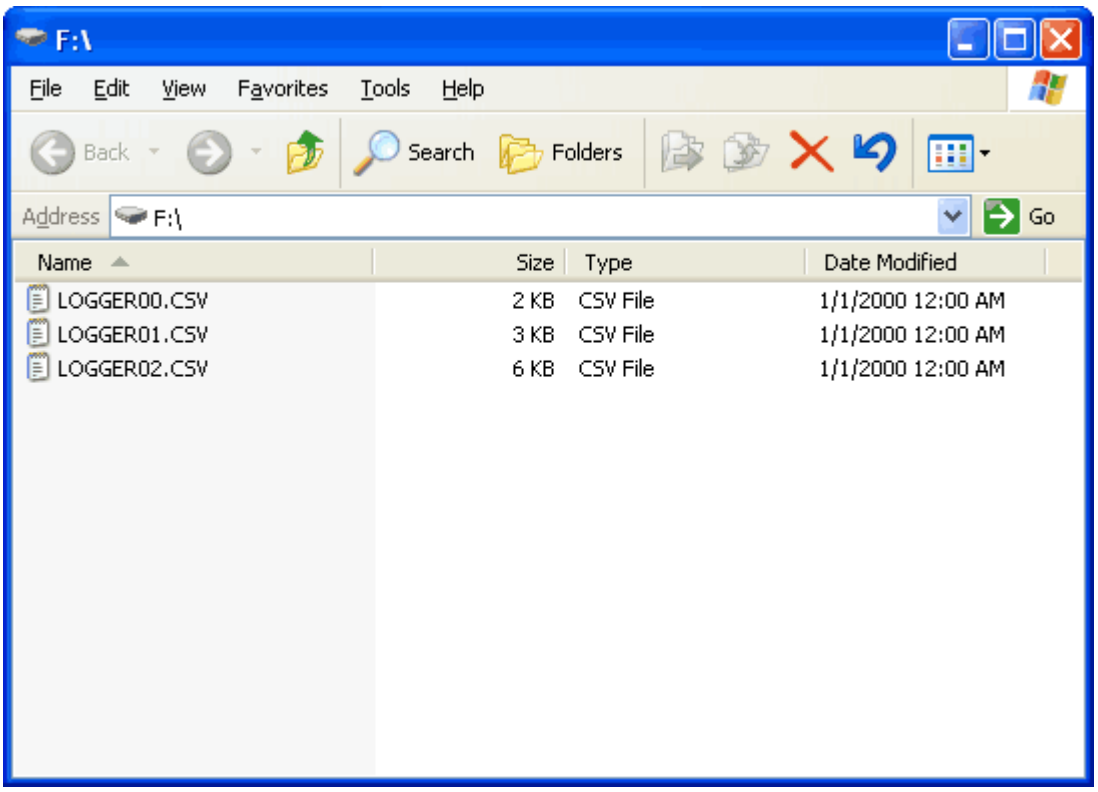
Помните, что примененные датчики не обладают большой точностью. После получения положительных результатов проверки переходите к экспериментам с регистрацией.

Программа регистрации

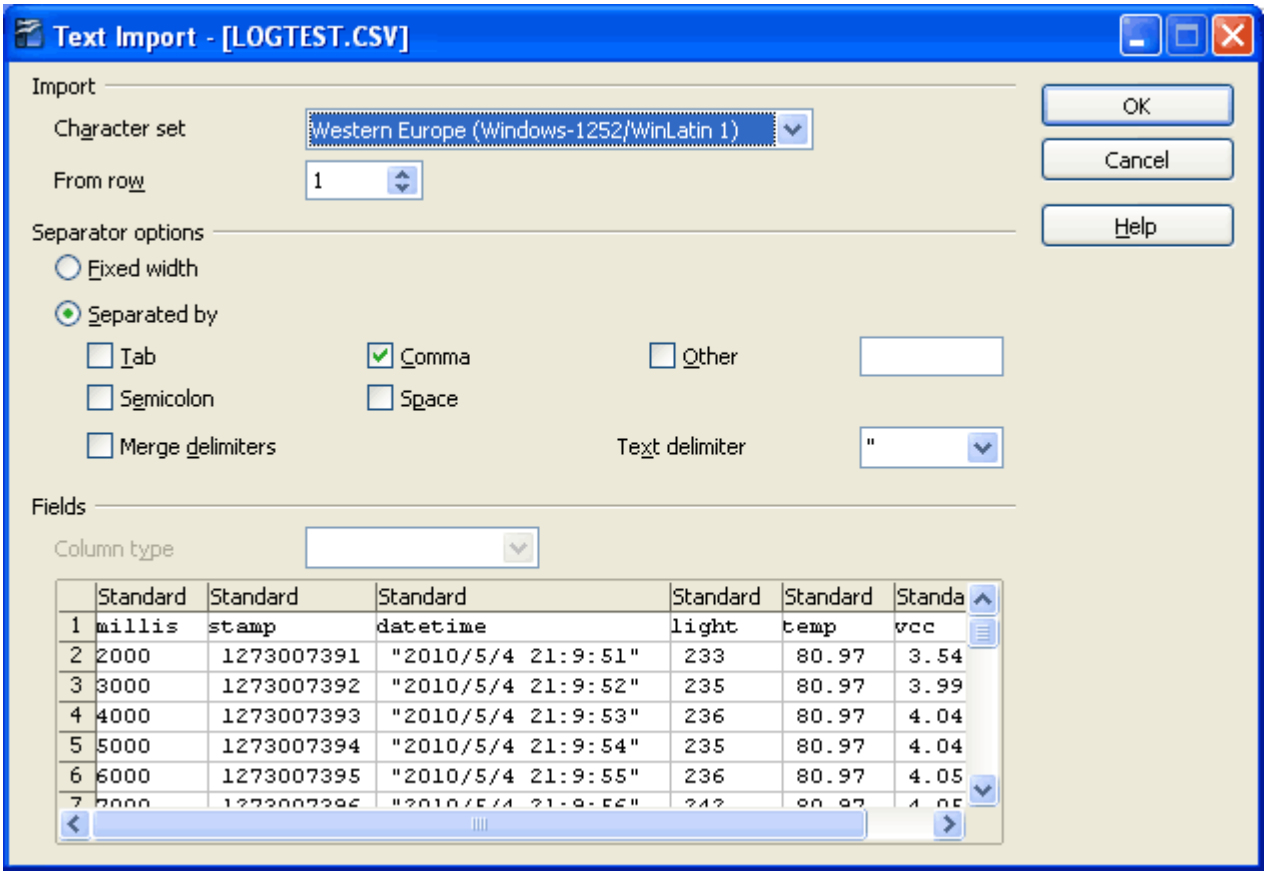
Скачайте программу ведения протокола освещенности и температуры. Вставьте SD-карту. Загрузить скетч в Arduino. Теперь проверим его связь с компьютером. В течение нескольких секунд закройте фотодатчик, а затем посветите на него фонариком. Чтобы нагреть датчик температуры сожмите его пальцами.

Построение электронной таблицы

Для проверки данных регистрации, отключите Arduino и вставьте SD карту в картридер компьютера. Вы увидите, по крайней мере 1, а возможно 2 или 3 файла, для каждого периода работы регистратора.



Откроем последний. Также можно посмотреть что должно получиться здесь. Самый быстрый способ для просмотра данных это использование OpenOffice или Excel, где можно открыть файл *.CSV и импортировать его в электронную таблицу.



Портативный регистратор

Можем сделать портативный регистратор мобильным при добавлении аккумуляторов или батарей. Самый дешевый способ получить хорошее суммарное напряжение это использовать 6 батарей типа AA. При использовании щелочных батарей можно получить 24 часа работы и более.

Описание текста программы

Скачать полный файл здесь

Включает в себя и определяет

```
#include "SD.h"
#include
#include "RTClib.h"
```

Три библиотеки включаются в начале файла: библиотека SD записывает на карту, библиотека Wire обеспечивает работу интерфейса I2C и библиотека RTClib служит для общения с часами реального времени.

```
// A simple data logger for the Arduino analog pins
#define LOG_INTERVAL 1000 // mills between entries
#define ECHO_TO_SERIAL 1 // echo data to serial port
#define WAIT_TO_START 0 // Wait for serial input in setup()

// the digital pins that connect to the LEDs
#define redLEDpin 3
#define greenLEDpin 4

// The analog pins that connect to the sensors
#define photocellPin 0 // analog 0
#define tempPin 1 // analog 1
```

Далее идут определения постоянных и констант.
LOG_INTERVAL устанавливает сколько миллисекунд между регистрациями состояний датчиков.
ECHO_TO_SERIAL определяет, следует ли отправить материалы, записываемые на карту также в последовательный монитор. Это замедляет работу регистратора, но с другой стороны полезно.

WAIT_TO_START означает, что вы должны послать символ в последовательный порт Arduino, чтобы начать регистрацию. Установлен 0 в настоящее время. Если хотите активировать включение от компьютера, то установите 1.
redLEDpin подключен контакт 3 к красному светодиоду регистратора.
greenLEDpin подключен контакт 4 к зеленому светодиоду регистратора.
photocellPin подключается аналоговый вход для фотодатчика.
tempPin подключается аналоговый вход для датчика температуры.

Объекты и error()

```
RTC_DS1307 RTC; // define the Real Time Clock object

// for the data logging shield, we use digital pin 10 for the SD cs line
const int chipSelect = 10;

// the logging file
File logfile;

void error(char *str)
{
 Serial.print("error: ");
 Serial.println(str);

 // red LED indicates error
 digitalWrite(redLEDpin, HIGH);

 while(1);
}
```

Мы получили все объекты для RTC и карты SD. Используем контакт 10 для SD-карты. Далее идет функция ошибки. Она выводит сообщение об ошибке в Serial Monitor, включается красный светодиод ошибки, а затем переходит в бесконечный цикл, приводящий к остановке.

Установки

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

#if WAIT_TO_START
 Serial.println("Type any character to start");
 while (!Serial.available());
#endif //WAIT_TO_START
```

Начинаем с инициализации последовательного порта со скоростью 9600 бод. Если мы установим WAIT_TO_START в 0, то Arduino не будет ждать, пока пользователь введет символ. В противном случае программа идет вперед к следующей части.

```
// initialize the SD card
 Serial.print("Initializing SD card...");
 // make sure that the default chip select pin is set to
```

```
 // output, even if you don't use it:
 pinMode(10, OUTPUT);

 // see if the card is present and can be initialized:
 if (!SD.begin(chipSelect)) {
 Serial.println("Card failed, or not present");
 // don't do anything more:
 return;
 }
 Serial.println("card initialized.");

 // create a new file
 char filename[] = "LOGGER00.CSV";
 for (uint8_t i = 0; i < 100; i++) {
 filename[6] = i/10 + '0';
 filename[7] = i%10 + '0';
 if (! SD.exists(filename)) {
 // only open a new file if it doesn't exist
 logfile = SD.open(filename, FILE_WRITE);
 break; // leave the loop!
 }
 }

 if (! logfile) {
 error("couldnt create file");
 }

 Serial.print("Logging to: ");
 Serial.println(filename);
```

Теперь программа начинает общаться с SD картой, пытается инициализировать карту и найти FAT16 или FAT32. Затем пытается создать файл журнала регистрации. Мы хотим, чтобы файлы назывались LOGGERnn.csv где nn номер. Начиная работу, регистратор создает файл LOGGER00.CSV и для следующих файлов увеличивается номер, если файл уже существует, пока не будет создан LOGGER99.csv. Новый файл создается каждый раз, когда включается регистратор.

FILE_WRITE – создать файл и записать в него данные.

Предположим, что файл создан успешно, передадим имя в последовательный порт.

```
Wire.begin();
 if (!RTC.begin()) {
 logfile.println("RTC failed");
#if ECHO_TO_SERIAL
 Serial.println("RTC failed");
#endif //ECHO_TO_SERIAL
 }

 logfile.println("millis,time,light,temp");
#if ECHO_TO_SERIAL
 Serial.println("millis,time,light,temp");
#if ECHO_TO_SERIAL// attempt to write out the header to the file
```

```
 if (logfile.writeError || !logfile.sync()) {
 error("write header");
 }

 pinMode(redLEDpin, OUTPUT);
 pinMode(greenLEDpin, OUTPUT);

 // If you want to set the aref to something other than 5v
 //analogReference(EXTERNAL);
```

Теперь мы подтолкнем RTC путем инициализации библиотеки Wire и будем запускать RTC, чтобы увидеть, работает или нет. Затем выводим заголовок. Заголовок в первой сроке файла помогает открыть электронную таблицу. Данные в файле находятся в CSV формате (значения, разделенные запятыми), поэтому заголовок имеет вид "миллисекунды, время, освещенность, температура". Выполнение logfile.print() записывает данные в файл на SD-карте. Если вы установите ECHO_TO_SERIAL равным 0, то данные в последовательный терминал поступать не будут. Устанавливаем два контакта для светодиодов. Существует закомментированная строка, где мы устанавливаем аналоговое опорное напряжение. Программа устанавливает, что будет использовано по умолчанию для опорного напряжения питание МК, которое составляет 5 В. Можно получить более высокую точность, снижая опорное напряжение.

Главный цикл

Теперь программа переходит к циклически повторяющимся действиям.

1. Выполнить временную задержку между чтениями.
2. Получить дату и время из часов.
3. Записать время и дату на SD карту
4. Получить данные фотоэлемента и датчика температуры
5. Записать информацию на SD карту
6. Синхронизация данных

Установка данных

Давайте посмотрим в первой части:

```
void loop(void)
{
 DateTime now;

 // delay for the amount of time we want between readings
 delay((LOG_INTERVAL -1) - (millis() % LOG_INTERVAL));

 digitalWrite(greenLEDpin, HIGH);

 // log milliseconds since starting
 uint32_t m = millis();
 logfile.print(m); // milliseconds since start
 logfile.print(", ");
#if ECHO_TO_SERIAL
 Serial.print(m); // milliseconds since start
 Serial.print(", ");
#endif
```

```
 // fetch the time
 now = RTC.now();
 // log time
 logfile.print(now.get()); // seconds since 2000
 logfile.print(", ");
 logfile.print(now.year(), DEC);
 logfile.print("/");
 logfile.print(now.month(), DEC);
 logfile.print("/");
 logfile.print(now.day(), DEC);
 logfile.print(" ");
 logfile.print(now.hour(), DEC);
 logfile.print(":");
 logfile.print(now.minute(), DEC);
 logfile.print(":");
 logfile.print(now.second(), DEC);
#if ECHO_TO_SERIAL
 Serial.print(now.get()); // seconds since 2000
 Serial.print(", ");
 Serial.print(now.year(), DEC);
 Serial.print("/");
 Serial.print(now.month(), DEC);
 Serial.print("/");
 Serial.print(now.day(), DEC);
 Serial.print(" ");
 Serial.print(now.hour(), DEC);
 Serial.print(":");
 Serial.print(now.minute(), DEC);
 Serial.print(":");
 Serial.print(now.second(), DEC);
#endif //ECHO_TO_SERIAL
```

Первый важный этап работы программы это вызов задержки. Вспомним, мы с помощью #define установили задержку между показаниями 1000 миллисекунд. Увеличение задержки позволяет экономить энергию и экономить объем памяти карты.
Затем включаем зеленый светодиод, чтобы сообщить о чтении или записи данных.
Происходит вызов RTC.now() чтобы получить данные времени. После пишем метку времени, а также дату в формате YY/MM/DD HH:MM:SS.

Сохранение данных датчиков

Далее идет код регистрации данных датчика.

```
int photocellReading = analogRead(photocellPin);
 delay(10);
 int tempReading = analogRead(tempPin);

 // converting that reading to voltage, for 3.3v arduino use 3.3
 float voltage = tempReading * 5.0 / 1024;
 float temperatureC = (voltage - 0.5) * 100 ;
 float temperatureF = (temperatureC * 9 / 5) + 32;
```

```
 logfile.print(", ");
 logfile.print(photocellReading);
 logfile.print(", ");
 logfile.println(temperatureF);
#if ECHO_TO_SERIAL
 Serial.print(", ");
 Serial.print(photocellReading);
 Serial.print(", ");
 Serial.println(temperatureF);
#endif //ECHO_TO_SERIAL

 digitalWrite(greenLEDpin, LOW);
}
```

Цикл заканчивается выключением зеленого светодиода.

Скачать
https://codeload.github.com/adafruit/SD/zip/master
https://github.com/adafruit/SD

Проверьте GitHub – следует заменить библиотеку SD, если у вас Arduino Mega или Leonardo.
https://codeload.github.com/adafruit/RTClib/zip/master

Проверьте GitHub, посмотрите код RTC библиотеки Adafruit

Для установки Arduino библиотеки

Домашняя страница

Техническая документация

Схема

По материалам сайта www.adafruit.com