```
#include <SPI.h>
#include <Arduino.h>
#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>
#include <SD.h>
#include <avr/sleep.h>
#include <Wire.h>         // this #include still required because the RTClib depends on it
#include "RTClib.h"

int voltage;
int Temperatura;
int TempSenzor = A0;

#define aref_voltage 3.3
float temp[10] = { 0 };

float i = 0;
int n = 0;
float median = 0;

SoftwareSerial mySerial(8, 7);
Adafruit_GPS GPS(&mySerial);

// Set GPSECHO to 'false' to turn off echoing the GPS data to the Serial console
// Set to 'true' if you want to debug and listen to the raw GPS sentences
#define GPSECHO  false
/* set to true to only log to SD when GPS has a fix, for debugging, keep it false */
#define LOG_FIXONLY true

// this keeps track of whether we're using the interrupt
// off by default!
#ifndef ESP8266 // Sadly not on ESP8266
boolean usingInterrupt = false;
#endif

// Set the pins used
#define chipSelect 10
#define ledPin 13

File logfile;

RTC_DS1307 RTC;  // define the Real Time Clock object
RTC_Millis rtc;

 char timestamp[30];
// call back for file timestamps
void dateTime(uint16_t* date, uint16_t* time) {
 DateTime now = RTC.now();
 sprintf(timestamp, "%02d:%02d:%02d %2d/%2d/%2d \n", now.hour(),now.minute(),now.second(),now.month(),now.day(),now.year()-2000);
 Serial.println("yy");
 Serial.println(timestamp);
 // return date using FAT_DATE macro to format fields
```

```
 53   *date = FAT_DATE(now.year(), now.month(), now.day());
 54
 55   // return time using FAT_TIME macro to format fields
 56   *time = FAT_TIME(now.hour(), now.minute(), now.second());
 57  }
 58
 59  // read a Hex value and return the decimal equivalent
 60  uint8_t parseHex(char c) {
 61    if (c < '0')
 62      return 0;
 63    if (c <= '9')
 64      return c - '0';
 65    if (c < 'A')
 66      return 0;
 67    if (c <= 'F')
 68      return (c - 'A')+10;
 69  }
 70
 71  // blink out an error code
 72  void error(uint8_t errno) {
 73    /*
 74    if (SD.errorCode()) {
 75     putstring("SD error: ");
 76     Serial.print(card.errorCode(), HEX);
 77     Serial.print(',');
 78     Serial.println(card.errorData(), HEX);
 79     }
 80     */
 81    while(1) {
 82      uint8_t i;
 83      for (i=0; i<errno; i++) {
 84        digitalWrite(ledPin, HIGH);
 85        delay(100);
 86        digitalWrite(ledPin, LOW);
 87        delay(100);
 88      }
 89      for (i=errno; i<10; i++) {
 90        delay(200);
 91      }
 92    }
 93  }
 94
 95  void setup() {
 96    Wire.begin();
 97   if (!RTC.begin()) {
 98     Serial.println("RTC failed");
 99     while(1);
100   };
101    // connect at 115200 so we can read the GPS fast enough and echo without
       dropping chars
102    // also spit it out
103    Serial.begin(115200);
104    Serial.println("\r\nUltimate GPSlogger Shield");
105    pinMode(ledPin, OUTPUT);
106    pinMode(TempSenzor, INPUT); //postavi izvod TempSenzor (A0) kao ulazni
107    analogReference(EXTERNAL);  // Koristim 3.3 Vref
```

```
108
109   // make sure that the default chip select pin is set to
110   // output, even if you don't use it:
111   pinMode(10, OUTPUT);
112
113   if (!SD.begin(chipSelect)) {
114     Serial.println("Card init. failed!");
115     error(2);
116   }
117   char filename[15];
118   strcpy(filename, "GPSLOG00.csv");
119   for (uint8_t i = 0; i < 100; i++) {
120     filename[6] = '0' + i/10;
121     filename[7] = '0' + i%10;
122     // create if does not exist, do not open existing, write, sync after write
123     if (! SD.exists(filename)) {
124       break;
125     }
126   }
127
128   logfile = SD.open(filename, FILE_WRITE);
129   if( ! logfile ) {
130     Serial.print("Couldnt create ");
131     Serial.println(filename);
132     error(3);
133   }
134   Serial.print("Writing to ");
135   Serial.println(filename);
136
137   // connect to the GPS at the desired rate
138   GPS.begin(9600);
139
140   // uncomment this line to turn on RMC (recommended minimum) and GGA (fix
        data) including altitude
141   //GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
142   // uncomment this line to turn on only the "minimum recommended" data
143   GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
144   // Set the update rate
145   GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);   // 100 millihertz (once every
        10 seconds), 1Hz or 5Hz update rate
146   // Turn off updates on antenna status, if the firmware permits it
147   GPS.sendCommand(PGCMD_NOANTENNA);
148   // the nice thing about this code is you can have a timer0 interrupt go off
149   // every 1 millisecond, and read data from the GPS for you. that makes the
150   // loop code a heck of a lot easier!
151 #ifndef ESP8266 // Not on ESP8266
152   useInterrupt(true);
153 #endif
154   Serial.println("Ready!");
155 }
156
157 // Interrupt is called once a millisecond, looks for any new GPS data, and
      stores it
158 #ifndef ESP8266 // Not on ESP8266
159 ISR(TIMER0_COMPA_vect) {
160   char c = GPS.read();
```

```
161   // if you want to debug, this is a good time to do it!
162   #ifdef UDR0
163       if (GPSECHO)
164         if (c) UDR0 = c;
165       // writing direct to UDR0 is much much faster than Serial.print
166       // but only one character can be written at a time.
167   #endif
168 }
169
170 void useInterrupt(boolean v) {
171   if (v) {
172     // Timer0 is already used for millis() - we'll just interrupt somewhere
173     // in the middle and call the "Compare A" function above
174     OCR0A = 0xAF;
175     TIMSK0 |= _BV(OCIE0A);
176     usingInterrupt = true;
177   }
178   else {
179     // do not call the interrupt function COMPA anymore
180     TIMSK0 &= ~_BV(OCIE0A);
181     usingInterrupt = false;
182   }
183 }
184 #endif // ESP8266
185
186 // function to sort the array in ascending order
187 void Array_sort(float *array, int n)
188 {
189   // declare some local variables
190   int i = 0, j = 0, temp = 0;
191   for (i = 0; i<n; i++)
192   {
193     for (j = 0; j<n - 1; j++)
194     {
195       if (array[j]>array[j + 1])
196       {
197         temp = array[j];
198         array[j] = array[j + 1];
199         array[j + 1] = temp;
200       }
201     }
202   }
203 }
204
205 float Find_median(float array[], int n)
206 {
207   float median = 0;
208   // if number of elements are even
209   if (n % 2 == 0)
210     median = (array[(n - 1) / 2] + array[n / 2]) / 2.0;
211   // if number of elements are odd
212   else
213     median = array[n / 2];
214   return median;
215 }
216
```

```
217 void loop(){
218   DateTime now = rtc.now();
219   if (! usingInterrupt) {
220     // read data from the GPS in the 'main loop'
221     char c = GPS.read();
222     // if you want to debug, this is a good time to do it!
223     if (GPSECHO)
224       if (c) Serial.print(c);
225   }
226
227   // if a sentence is received, we can check the checksum, parse it...
228   if (GPS.newNMEAreceived()) {
229     char *stringptr = GPS.lastNMEA();
230
231     if (!GPS.parse(stringptr))   // this also sets the newNMEAreceived() flag ↵
        to false
232       return;  // we can fail to parse a sentence in which case we should just ↵
           wait for another
233
234     // Sentence parsed!
235     Serial.println("OK");
236     if (LOG_FIXONLY && !GPS.fix) {
237       Serial.print("No Fix");
238       return;
239     }
240
241     float voltage = analogRead(TempSenzor) * 3.3;  //ocitava vrijednosti      ↵
        izvoda (A0)
242     voltage /= 1024.0; //10bit ADC
243     float Temperatura = (voltage - 0.5) * 100;
244     Serial.print("Trenutno: ");
245     Serial.println(Temperatura);
246
247     // Rad. lets log it!
248     Serial.println("Log");
249
250     char tempBuff[5];
251     dtostrf(Temperatura,0,2,tempBuff);
252     uint8_t tempSize = strlen(tempBuff);
253
254     //logfile.flush();
255
256  // ovaj blok kao dela pa pomalo s tim :-)
257         uint8_t stringsize = strlen(stringptr);// + tempSize;
258         if (stringsize != logfile.write((uint8_t *)stringptr,                 ↵
             stringsize))    //write the string to the SD file
259         error(4);
260     if (strstr(stringptr, "RMC") || strstr(stringptr, "GGA") )   logfile.flush ↵
        ();
261     logfile.write(tempBuff);
262     Serial.println();
263
264   }
265 }
266
```