```
#include <SD.h>
#include <Wire.h>
#include "RTClib.h"
#include <SPI.h>

// A simple data logger for the Arduino analog pins

// how many milliseconds between grabbing data and logging it. 1000 ms is once a second
#define LOG_INTERVAL  1000 // mills between entries (reduce to take more/faster data)

// how many milliseconds before writing the logged data permanently to disk
// set it to the LOG_INTERVAL to write each time (safest)
// set it to 10*LOG_INTERVAL to write all data every 10 datareads, you could lose up to
// the last 10 reads if power is lost but it uses less power and is much faster!
#define SYNC_INTERVAL 10000 // mills between calls to flush() - to write data to the card
uint32_t syncTime = 0; // time of last sync()

#define ECHO_TO_SERIAL   0 // Echo data to serial monitor (Setting it to 0 will turn it off)
#define WAIT_TO_START    0 // Wait for serial input in setup() (Setting it to 1 you have to send a character to zhe Arduino´s Serial port to start the logging)

// the digital pins that connect to the LEDs (Solder pads L1 (green) and L2 (red) on the datalogger shield)
#define redLEDpin 2
#define greenLEDpin 3

// The analog pin connected to the PV cell voltage divider
#define sensorPin A0


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);
}

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

  // use debugging LEDs
  pinMode(redLEDpin, OUTPUT);
  pinMode(greenLEDpin, OUTPUT);

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

```
#endif //WAIT_TO_START

  // 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)) {
    error("Card failed, or not present");
  }
  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);

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


  logfile.println("millis; stamp; datetime; voltage");
#if ECHO_TO_SERIAL
  Serial.print("millis");
  Serial.print("\t");
  Serial.print("stamp");
  Serial.print("\t");
  Serial.print("\t");
  Serial.print("datetime");
  Serial.print("\t");
  Serial.print("\t");
  Serial.println("voltage");
#endif //ECHO_TO_SERIAL
  }

void loop(void)
```

```
{
  // stores the value read on analog input A0
  int sensorVal = analogRead(sensorPin);
  // set the range of values to be considered for datalogging
  if(sensorVal>=3.25) // 3.25 corresponds roughly to 12V on the PV cell
 {
  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("\t");
#endif

  // fetch the time
  now = RTC.now();
  // log time
  logfile.print(now.unixtime()); // seconds since 1/1/1970
  logfile.print("; ");
  // logfile.print();
  logfile.print(now.day(), DEC);
```

```
  logfile.print(".");
  logfile.print(now.month(), DEC);
  logfile.print(".");
  logfile.print(now.year(), DEC);
  logfile.print(" ");
  logfile.print(now.hour(), DEC);
  logfile.print(":");
  logfile.print(now.minute(), DEC);
  logfile.print(":");
  logfile.print(now.second(), DEC);
  // logfile.print('"');
#if ECHO_TO_SERIAL
  Serial.print(now.unixtime()); // seconds since 1/1/1970
  Serial.print("\t");
  Serial.print(now.day(), DEC);
  Serial.print(".");
  Serial.print(now.month(), DEC);
  Serial.print(".");
  Serial.print(now.year(), DEC);
  Serial.print(" ");
  Serial.print(now.hour(), DEC);
  Serial.print(":");
  Serial.print(now.minute(), DEC);
  Serial.print(":");
  Serial.print(now.second(), DEC);
  Serial.print("\t");
#endif //ECHO_TO_SERIAL

  // converting the value of sensorVal to voltage
```

```
  float voltage = (sensorVal/1024.0)*5.0;

  logfile.print("; ");
  logfile.println(voltage);

#if ECHO_TO_SERIAL
  Serial.println(voltage);
#endif //ECHO_TO_SERIAL

// At the end of data logging the green LED is turned off
digitalWrite(greenLEDpin, LOW);

  // Now we write data to disk! Don't sync too often - requires 2048 bytes of I/O to SD card
  // which uses a bunch of power and takes time
  if ((millis() - syncTime) < SYNC_INTERVAL) return;
  syncTime = millis();

  // blink LED to show we are syncing data to the card & updating FAT!
 digitalWrite(redLEDpin, HIGH);
  logfile.flush();
 digitalWrite(redLEDpin, LOW);
 }
 else {
  #if ECHO_TO_SERIAL
  Serial.print("sensorVal is out of range: ");
  Serial.println(sensorVal);
  #endif // ECHO_TO_SERIAL
  }
}
```