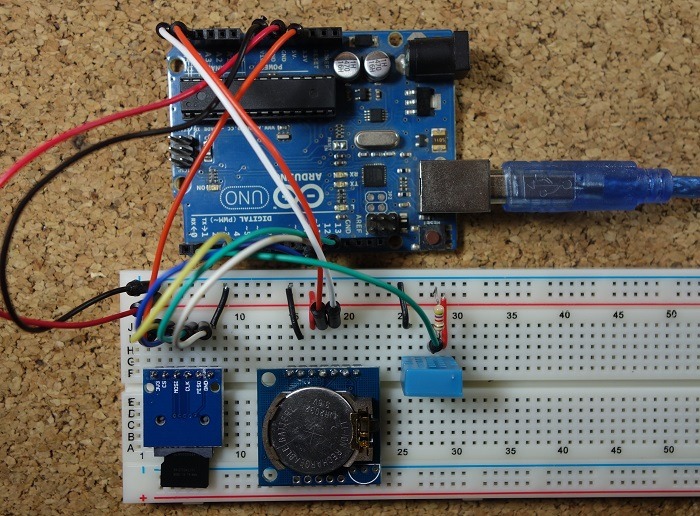
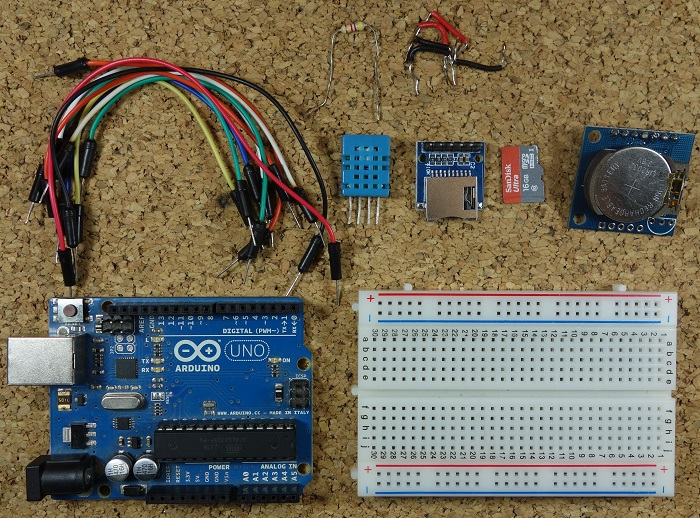
**Arduino Temperature Data Logger with SD Card Module**

This post shows you how to create a temperature Arduino data logger. We’ll use the DHT11 to measure temperature, the real time clock (RTC) module to take time stamps and the SD card module to save the data on the SD card. 

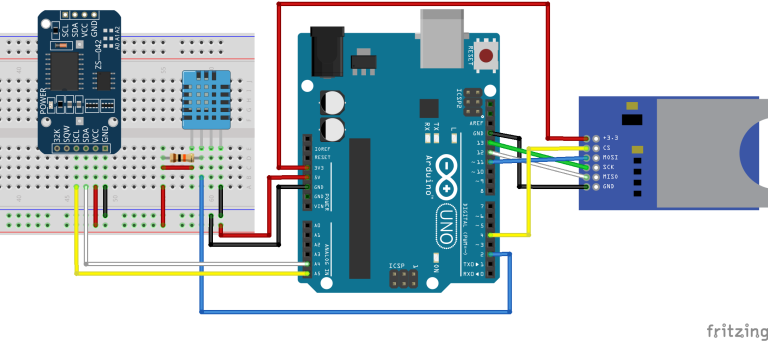
## **Parts required**

Here’s a complete list of the parts required for this project: 

* [Arduino UNO](https://makeradvisor.com/tools/compatible-arduino-uno-r3-board/)
* [SD card module](https://makeradvisor.com/tools/sd-card-module/)
* [Micro SD card](https://makeradvisor.com/tools/microsd-card-raspberry-pi-16gb-class-10/)
* [DHT11 temperature and humidity sensor](https://makeradvisor.com/tools/dht11-temperature-humidity-sensor/)
* [RTC module](https://makeradvisor.com/tools/real-time-clock-module-ds1307/)
* [Breadboard](https://makeradvisor.com/tools/mb-102-solderless-breadboard-830-points/)
* [Jumper wires](https://makeradvisor.com/tools/jumper-wires-kit-120-pieces/)

Note: alternatively to the SD card module, you can use a [data logging shield](https://makeradvisor.com/tools/data-logging-shield/). The data logging shield comes with built-in RTC and a prototyping area for soldering connections, sensors, etc..

## **Schematics**

The following figure shows the circuit’s schematics for this project.

**Note:** make sure your SD card is formatted and working properly. Read

#### **Installing the DHT sensor library**

For this project you need to install the DHT library to read from the DHT11 sensor.

1. [Click here to download the DHT-sensor-library](https://github.com/adafruit/DHT-sensor-library/archive/master.zip). You should have a .zip folder in your Downloads folder
2. Unzip the .zip folder and you should get DHT-sensor-library-master folder
3. Rename your folder from to **DHT**
4. Move the DHT folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE

**Code**

(raw code - <https://raw.githubusercontent.com/RuiSantosdotme/Random-Nerd-Tutorials/master/Projects/Arduino_Temperature_Data_Logger.ino>)

/\*

\* Rui Santos

\* Complete Project Details https://randomnerdtutorials.com

\*/

#include <SPI.h> //for the SD card module

#include <SD.h> // for the SD card

#include <DHT.h> // for the DHT sensor

#include <RTClib.h> // for the RTC

//define DHT pin

#define DHTPIN 2 // what pin we're connected to

// uncomment whatever type you're using

#define DHTTYPE DHT11 // DHT 11

//#define DHTTYPE DHT22 // DHT 22 (AM2302)

//#define DHTTYPE DHT21 // DHT 21 (AM2301)

// initialize DHT sensor for normal 16mhz Arduino

DHT dht(DHTPIN, DHTTYPE);

// change this to match your SD shield or module;

// Arduino Ethernet shield and modules: pin 4

// Data loggin SD shields and modules: pin 10

// Sparkfun SD shield: pin 8

const int chipSelect = 4;

// Create a file to store the data

File myFile;

// RTC

RTC\_DS1307 rtc;

void setup() {

//initializing the DHT sensor

dht.begin();

//initializing Serial monitor

Serial.begin(9600);

// setup for the RTC

while(!Serial); // for Leonardo/Micro/Zero

if(! rtc.begin()) {

Serial.println("Couldn't find RTC");

while (1);

}

else {

// following line sets the RTC to the date & time this sketch was compiled

rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

}

if(! rtc.isrunning()) {

Serial.println("RTC is NOT running!");

}

// setup for the SD card

Serial.print("Initializing SD card...");

if(!SD.begin(chipSelect)) {

Serial.println("initialization failed!");

return;

}

Serial.println("initialization done.");

//open file

myFile=SD.open("DATA.txt", FILE\_WRITE);

// if the file opened ok, write to it:

if (myFile) {

Serial.println("File opened ok");

// print the headings for our data

myFile.println("Date,Time,Temperature ºC");

}

myFile.close();

}

void loggingTime() {

DateTime now = rtc.now();

myFile = SD.open("DATA.txt", FILE\_WRITE);

if (myFile) {

myFile.print(now.year(), DEC);

myFile.print('/');

myFile.print(now.month(), DEC);

myFile.print('/');

myFile.print(now.day(), DEC);

myFile.print(',');

myFile.print(now.hour(), DEC);

myFile.print(':');

myFile.print(now.minute(), DEC);

myFile.print(':');

myFile.print(now.second(), DEC);

myFile.print(",");

}

Serial.print(now.year(), DEC);

Serial.print('/');

Serial.print(now.month(), DEC);

Serial.print('/');

Serial.println(now.day(), DEC);

Serial.print(now.hour(), DEC);

Serial.print(':');

Serial.print(now.minute(), DEC);

Serial.print(':');

Serial.println(now.second(), DEC);

myFile.close();

delay(1000);

}

void loggingTemperature() {

// Reading temperature or humidity takes about 250 milliseconds!

// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

// Read temperature as Celsius

float t = dht.readTemperature();

// Read temperature as Fahrenheit

//float f = dht.readTemperature(true);

// Check if any reads failed and exit early (to try again).

if (isnan(t) /\*|| isnan(f)\*/) {

Serial.println("Failed to read from DHT sensor!");

return;

}

//debugging purposes

Serial.print("Temperature: ");

Serial.print(t);

Serial.println(" \*C");

//Serial.print(f);

//Serial.println(" \*F\t");

myFile = SD.open("DATA.txt", FILE\_WRITE);

if (myFile) {

Serial.println("open with success");

myFile.print(t);

myFile.println(",");

}

myFile.close();

}

void loop() {

loggingTime();

loggingTemperature();

delay(5000);

}

n this code we create a **loggingTime()** function and a **loggingTemperature()** function that we call in the **loop()** to log the time and temperature to the **DATA.txt** file in the SD card.

Open the Serial Monitor at a baud rate of 9600 and check if everything is working properly.

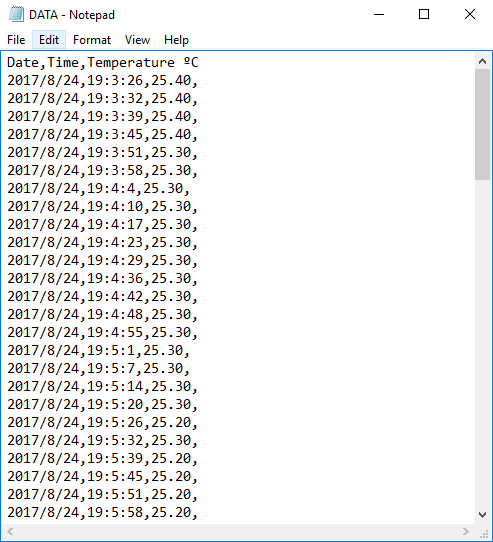
## Getting the data from the SD card

Let this project run for a few hours to gather a decent amount of data, and when you’re happy with the data logging period, shut down the Arduino and remove the SD from the SD card module.

Insert the SD card on your computer, open it, and you should have a **DATA.txt** file with the collected data.



You can open the data with a text editor, or use a spreadsheet to analyse and process your data.



The data is separated by commas, and each reading is in a new line. In this format, you can easily import data to Excel or other data processing software.