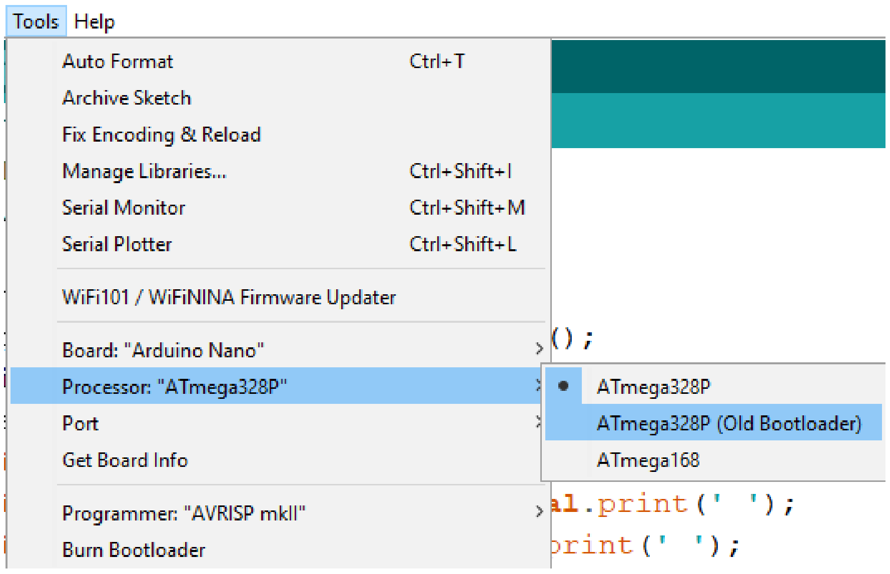
A Short Note: Using Arduino Nano as a substitute for Arduino UNO

David R. Brooks, © 2018, 2020

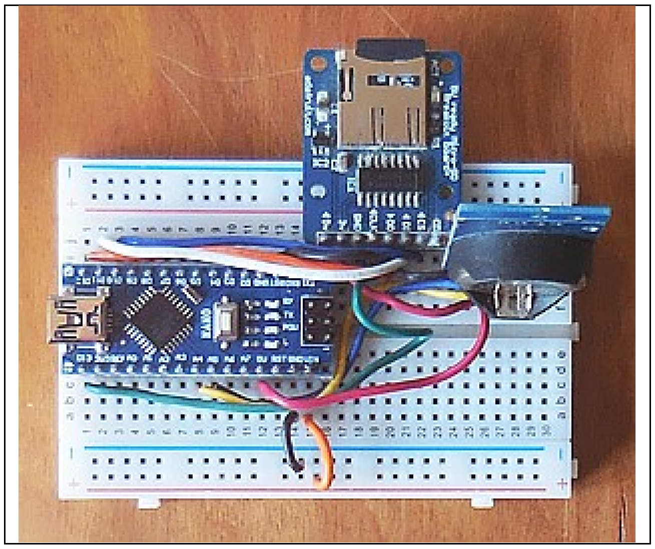
For first-time and casual Arduino users, the most popular board option is the Arduino UNO. This board can be powered with a USB A-B cable when communicating between an Arduino board and the integrated development environment (IDE) that allows you to create and upload sketches, or once sketches are uploaded, by 7-12 V from some other DC voltage source, through an on-board 2.1 mm power jack to an on-board voltage regulator, when a sketch is already in place. There are *many* shields that fit on the UNO headers to add additional capabilities to the board – for example the Adafruit datalogging shield that includes both a date/time clock (set from your computer clock) and a microSD card module for logging data from sensors.

There are alternatives to the Arduino UNO. The Arduino Nano is a small board that has the capabilities of a UNO in a much smaller and less expensive package, with a mini-B USB connector. These are available from allelectronics.com (ARD-20, $8.95) and other several other online sources, sometimes for less than $4 each. Arduino UNOs and compatibles are available at various prices from a variety of sources, some under $15. The only Nano I have tried, which is not an "official" Arduino product, is the one from allelectronics.com, so I can't guarantee performance of any of the others.

UPDATE: Because the NANO from allelectronics.com, and I assume other inexpensive online versions, is not an official version, it may be an older version that will not work with the default ATmega328P choice under the "Processor" menu. Choose the "Old Bootloader" as shown in this screen shot.


Tools
Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Nano"
Processor: "ATmega328P"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
ATmega328P
ATmega328P (Old Bootloader)
ATmega168

Nanos don't support the many Arduino UNO shields, but Arduino Nano is one of the boards supported by the Arduino IDE and it performs just like an UNO without any code modifications. Here is a breadboard layout using an allelectronics.com Nano board. The mini-B USB connector is on the left edge. The Adafruit microSD card (ID 254, $7.50) is at the top center and a real time clock, larger than but compatible with the Adafruit DS1307 module (ID 3296, $7.50) is to the right. There is a little layout room for sensors in the lower right-hand corner of the breadboard.



For external power, 7-12 V would be applied to the on-board 5-V regulator through the VIN pin (the orange and black wires at the bottom of the breadboard). Power for the clock and microSD card modules is taken from the +5 V pin. For all the other connections, refer to the documentation for the Adafruit microSD and clock modules. On the Nano, as on the UNO, pins A4 and A5 are used for the clock's SDA and SCL connections. (It's an I2C device.) Note that the 3.3 V pin is not active when the board is powered by an external source, rather than from the USB connection.

Is the Nano an attractive alternative to a UNO? That depends on your needs. For some projects it will be a less expensive alternative to the UNO. Because of the built-in USB connector, it's more convenient to use than the SparkFun Pro Mini, for example. For projects with access to an AC outlet to power a DC supply, the added convenience of using an UNO with shields probably outweighs the Nano advantages. For weight- and power-sensitive applications, it's a good choice.

Here's some code to test this layout. It sends date and time information to the serial port and the microSD card. The `Wire.h` file is for I2C communications with the clock and the `SPI.h` file is for communicating with the SD card module.

```
/* NanoLogTest.ino, D. Brooks, March 2018
  Tests operation of Nano with RTC and microSD modules for logging data.
*/
#include <Wire.h>
#include <SD.h>
#include <RTClib.h>
#include <SPI.h>
#define SDpin 10
RTC_DS1307 rtc; // real time clock module
File logfile;
char filename[] = "NANOTEST.CSV";
void setup() {
  Serial.begin(9600);
  Wire.begin(); rtc.begin();
  if (!rtc.isrunning()) {
    Serial.println("RTC not running."); exit;
  }
  if (!SD.begin(SDpin)) {Serial.println("Card failed.");
  delay(50);exit(0);}
  Serial.println("card initialized.");
  logfile=SD.open(filename,FILE_WRITE);
  rtc.adjust(DateTime(__DATE__,__TIME__));
}
void loop() {
  DateTime now=rtc.now();
  Serial.print(now.year()); Serial.print('/');
  Serial.print(now.month()); Serial.print('/');
  Serial.print(now.day()); Serial.print(',');
  Serial.print(now.hour()); Serial.print(':');
  Serial.print(now.minute()); Serial.print(':');
  Serial.print(now.second());  Serial.println();
  delay(10);
  logfile.print(now.year()); logfile.print(',');
  logfile.print(now.month()); logfile.print(',');
  logfile.print(now.day()); logfile.print(',');
  logfile.print(now.hour()); logfile.print(',');
  logfile.print(now.minute()); logfile.print(',');
  logfile.print(now.second()); logfile.print(',');
  logfile.println(); logfile.flush();
  delay(1000);
}
```