Registration of Changes in the Environment During a Total Solar Eclipse Using ARDUINO

György Hudoba
Óbuda University, Alba Regia Technical Faculty, Institute of Engineering, Székesfehérvár, Hungary
Hudoba.gyorgy@amk.uuni-obuda.hu

***Abstract*—In August 21. 2017 a total solar eclipse was visible across the United States. I registered the changes in light, temperature, air pressure and relative humidity changes during the eclipse. In the followings I shortly present the device and the results as well.**

I. INTRODUCTION

On Aug. 21, 2017, skies darkened from Lincoln Beach, Oregon to Charleston, South Carolina. The event was the first total solar eclipse visible from coast to coast across the United States in 99 years. A total solar eclipse occurs when the shadow cast by the moon is sweeps through the surface of the Earth (Figure 1.). Seeing from Earth, the disk of the moon appears to completely cover the disk of the sun in the sky. During a total solar eclipse, the sun's tenuous outer atmosphere, the corona, becomes visible.

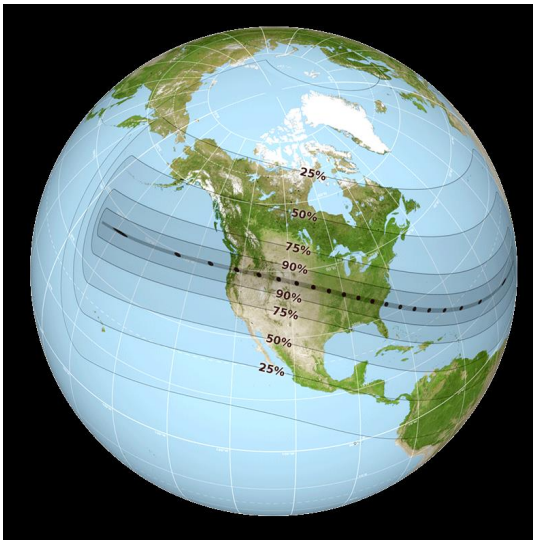

Figure 1. The globe view of the path of totality for the August 21, 2017 total solar eclipse.
[Image Credit: NASA's Scientific Visualization Studio]

My goal was registering and analyzing the changes in the environment conditions, namely the changes of light, temperature, humidity and atmospheric pressure during the eclipse. For this purpose I designed an ARDUINO-based device.

II. THE ARDUINO FAMILY

Arduino is just one small part of the single-board computing (e.g. Raspberry PI among many others) and the world of embedded electronics and DIY (Do It Yourself) technology. Some selected resources: [1] – [22].

Arduino was named after a bar frequented by students at the Interaction Design Institute in the northern Italian city of Ivrea. The bar was named for an Italian king, Arduin of Ivrea, who briefly ruled Italy around 1000 CE. The word "Arduino" roughly translates to "strong friend."

Arduino is:

- an open-source electronic prototyping platform based on flexible easy to use hardware and software, that was designed for artists, designers, hobbyists, hackers, newbies, or even professionals, and anyone interested in creating interactive objects or environments.
- consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on our computer, used to write and upload computer code to the physical board.

There are many varieties of Arduino boards that can be used for different purposes. The term "ARDUINO" refers to a whole family, see TABLE I. below.

The boards differ in sizes and shapes as well as capabilities. Some are credit card sized, others just a tiny stripe, or even round for fashion designers. (The LilyPad line of wearable Arduino boards feature large, sewable tabs for connecting project with conductive thread, and a distinct lack of corners so they don't get caught up in our fabric.)

TABLE I.
ARDUINO MEMBERS

General purpose ARDUINO boards	
	MCU only boards Combined MCU / MPU boards
Special purpose ARDUINO boards	
	ARDUINO Esplora ARDUINO Robot
ARDUINO compatible boards	
	Intel Galileo, Gen 2 Intel Edison
ARDUINO shields	
	Ethernet, WiFi, GSM, Motor, Relay and others

Most Arduino boards built around an ATmega microcontroller unit (MCU), which is like a complete computer. It has CPU, RAM, Flash memory, A/D converter, and input/output pins, all on a single chip. It is designed to attach all kinds of sensors, LEDs, small motors and speakers, servos, etc. directly to these pins, which can read in or output digital or analog voltages between 0 and 5 volts. Its wide voltage tolerance and low power consumption makes it perfect for the Arduino.

Arduino can interact with buttons, LEDs, motors, speakers, GPS units, cameras, the internet, and so on. This flexibility combined with the fact that the Arduino software is free, the hardware boards are pretty cheap, and both the software and hardware are easy to learn has led to a huge variety of Arduino-based projects.

A. *Some special ARDUINO terms*

- shield: Arduino shields are modular circuit boards that piggyback onto Arduino to expand with extra functionality.
- sketch: Arduino programs are called sketches. They are usually written in C++.

B. *ARDUINO programs*

The structure of the sketches is very simple. It consists two parts: *setup* and *loop*.

- setup: It is called only once, when the Arduino is powered on or reset. It is used to initialize variables and pin modes.
- loop: The loop function runs continuously till the device is powered off. The main logic of the code goes here. Similar to while (1) for micro-controller programming.

The Arduino connects to the computer via USB, where we program it in a simple language (C/C++) inside the free Arduino IDE (Integrated Development Environment) by uploading the compiled code to the board. Once programmed, the Arduino can run with the USB link back to our computer, or stand-alone without it — no keyboard or screen needed, just power.

III. THE DEVICE

For logging the data measured during the total solar eclipse I used the following components:

- ARDUINO Uno R3 board (Figure 2.)
- XD-05 Arduino Data Logging Shield Module (Figure 3.)
- YURobot Easy Module Shield v1(Figure 4.)
- BMP-280 Barometer Atmosphere Pressure Sensor module (Figure 5.)
- VK2828U7G5LF GPS Module (Figure 6.)

A. *The ARDUINO Uno R3 board*

The ARDUINO UNO R3 is a General purpose MCU only board. The heart of the board is the Atmel AVR ATmega 328.The ATmega328 on Arduino Uno contains a modified Harvard architecture 8-bit RISC processor as well as a block of flash memory, multiple timers, analog-to-digital converters and PWM generators, all packed into that one little chip. (The R3 refers to the development phase of the basic ARDUINO board.)

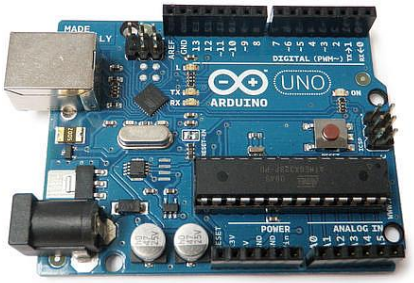

Figure 2. The ARDUINO Uno board used in the project

B. *The XD-05 Arduino Data Logging Shield Module*

This shield has a RTC (Real Tim Clock) module and a SD card module. In addition to, there is an empty space for prototyping. The main features:

- RTC with battery
- Realtime reading
- SD card interface
- Could save data to any FAT16 / FAT32 SD card
- 3.3V level transmit circuit

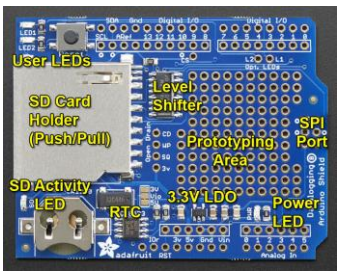

Figure 3. The Data Logging Shield

C. *The YURobot Easy Module Shield v1*

This shield integrates various module functions and we can directly program to complete the experiment without welding and coping jungle of cables.

Board Features:

– Two pushbuttons
– Two channels LED module (a Blue and a Red LED
– Full color LED module (one RGB LED)
– Infrared receiver module
– Brightness sensor module (CdS Photorezistor)
– LM35D temperature sensor module
– Passive buzzer module
– Rotary potentiometer module
– DHT11 temperature and humidity sensor module[1]
– One I2C interface (SDA A5, SCL A4)
– One TTL serial port
– Two channel digital quantity port (D7, D8)
– One channel analog port (A3)
– Reset button

[1] The DHT-11 Digital Temperature and Humidity Sensor is laboratory pre-calibrated, and uses Single-wire communication

Figure 4. The YURobot Easy Module Shield v1

D. The BMP-280 Barometer Atmosphere Pressure Sensor module

This module has two sensors, one for temperature and another for barometric pressure, and can even be used in both I2C and SPI. (SPI = Serial Peripheral Interface is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices.) The sensors are very precise: measures barometric pressure with ±1 hPa absolute accuracy, and temperature with ±1.0°C accuracy. Because pressure changes with altitude, and the pressure measurements are so good, we can also use it as an altimeter with ±1 meter accuracy.

Figure 5. The BMP-280 module

E. The VK2828U7G5LF GPS Module

I used this GPS module for obtaining the precise geographic coordinates (latitude, longitude and altitude) and the time of the observation.

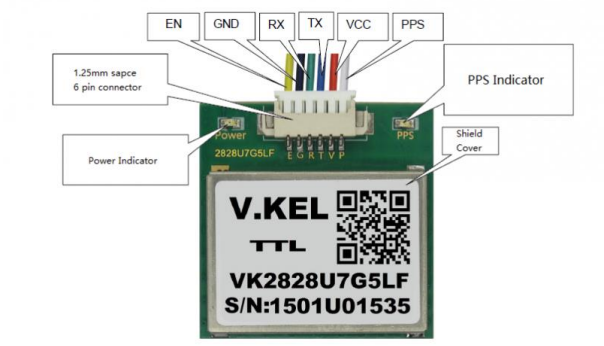

Figure 6. The GPS module

The GPS module communicate with the ARDUINO via serial port.

IV. Completing the Hardware

Although the Data Logging and Easy Shield modules match perfectly to ARDUINO as a piggyback device using pins and headers, completing the device still needed some wiring and other considerations. The BMP-280 module uses I2C, so I connected SDA to the A4, the SCL to the A5 pin (A means analogue pin of the Microcontroller). The GPS module uses serial communication, so the GPS TX went to the D8, and the Rx to the D7 pin of the ARDUINO board (D means digital pin of the Microcontroller).

For the possible most accurate measurements the A/D converter needs a reference voltage. Because the ARDUINO UNO has limitations, as a circumvention I connected the on board 3.3 V to the pin A3, considered as stabilized value. The idea behind this, that if the power of the microcontroller as reference is changing for any reason, measuring a fixed 3.3 Volt gives us an opportunity correcting the digitized values of other analog devices.

V. Other Considerations

As from the previous sections follows, I used many interfaces. Some sensors communicate via I2C, others on one-wire, on serial port or even produced analog signal, and needed free pin for connecting to the A/D converter. Moreover, the A/D converter uses the power level of the microcontroller as a reference.

Powering up the Arduino has two possibilities. One possible way is to use the USB port, the other is using the power jack. Since I want using the ARDUINO as a standalone device, I needed an external power bank. Because the measurement lasted many hours, the input voltage could not be considered as constant. On the other hand, the on board 3.3 V power regulator needs some extra power, so the output power drops about one to two Volts. As a result, if we try to operate the device from 5 Volts, the system become instable, so the external power should be about 9 Volts.

The UNO has some limitations in number of pins and memory. That caused another headache. I could not use the GPS module and the sensors simultaneously. Moreover, as it turned out, the RTC does not keeps the proper time on long run. As a result, the measurement occurred in two steps. First, I had to upload the sketch for the GPS to synchronize the RTC, and get the geographic position of the site, then upload the data collecting and recording sketch.

VI. The Software

The program using the device consists of two sketches: the “GPS2RTC_Sync.ino” and the “EclipseLOG.ino”. (The “.ino” extension is the standard for the ARDUINOs.) The first program reads the standard NMEA sentences from the GPS receiver, and prints to the serial monitor the RTC time, the GPS time and date, and the quality of the reception. If the quality of the reception is good enough (Fix = 1, Quality = 2), then the RTC time will be updated (synchronized) once. The serial monitor will show the Fix, and Quality, the geographic position (latitude, longitude and altitude), and the number of GPS satellites seen in each secons.

The second program at start initializes the SD card, and creates a new .csv file (LOG_xx.CSV), by incrementing the index. After that it creates a record with the following contents:

- # of record
- milliseconds elapsed from the start
- seconds elapsed from 01.01.1970

- the current date and time (UTC) in human readable format
- luminosity (Volts on CdS photoresistor)
- atmospheric pressure (BMP-280)
- temperature in Celsius (BMP-280)
- temperature (Volts on LM35)
- temperature in Celsius (converted from LM35)
- temperature in Celsius from DHT11 sensor
- relative humidity in % from DHT11 sensor
- Ref3 – digitalized value of the stabilized 3.3 V

The registration is started for pushing either K1 or K2 buttons on the Easy Module Shield. (We can stop recording data for the same action.) The data is updated in each second, and is echoed to the serial monitor as well. The records created are stored in a buffer, and after collecting 10 records the content of the buffer will be stored on the SD card.

VII. RESULTS

The measurement was successful. The results (the content of the csv file created) were processed in EXCEL, and present them in graphical form.

On board RTC time:	2017/8/21 15:31:4
GPS time:	15:31:4.0
Date (yyyy.mm.dd):	2017.08.21
Fix:	1
Quality:	2
GPS coordinates:	4351.5336N, 11232.2119W
Coordinates for Google Maps:	43.8589, -112.5369
Elevation:	1463.70 m
# of GPS satellites:	12

Figure 7. The data from GPS module

The Figure 7. shows the data obtained from the GPS module. Figure 8. presents, how the illumination, the temperature and the air pressure changed during the 4 hours observational period, the measurement was taken. The illumination came from the CdS photoresistor, the temperature and the pressure from the BMP-280 detector.

On Figure 9. we can see the temperature obtained from LM35 and the DHT11 sensor, and the relative humidity changes. We have all together three temperature curves. They have the same figure, but different values. This is not an error, but result of different position. The BMP-280 was inside the device, between two shields, the LM35 was exposed to the direct radiation of the Sun, and the third was in a shaded place, under the DHT11 protecting housing.

Zooming into the totality (Figure 10.), we get a nice smooth curve. The bottom of the light curve is not flat, because of the Sun's although dim, but well visible outer atmosphere. the corona. Differentiating the light curve, we can determine the duration of the totality which is 2 minutes 17 seconds, corresponds to the prediction for the site of the observation.

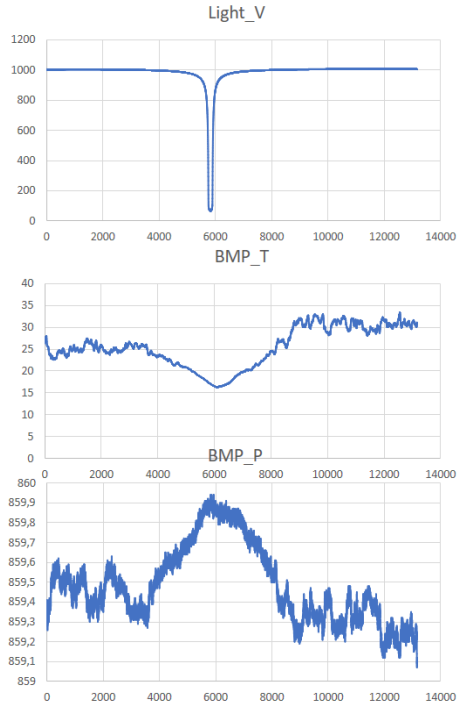

Figure 8. Changes of light, temperature and atmospheric pressure during the 4 hours observational period

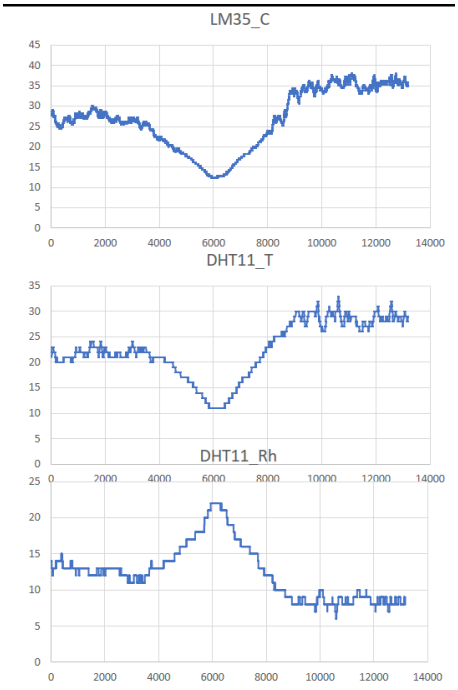

Figure 9. Changes of temperature and relative humidity during the 4 hours observational period

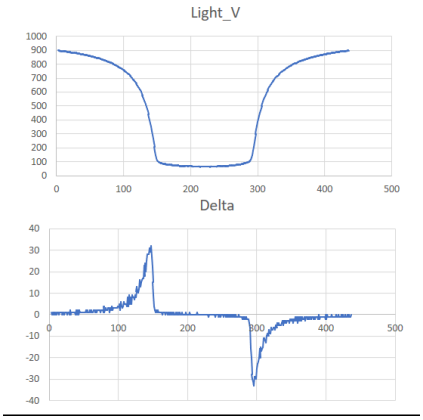

Figure 10. Zooming into the totality. The lower cure is the derivative of the light curve.

RESOURCES

(The internet sources last visited in October, 2017)

[1] adafruit-data-logger-shield.pdf
[2] Arduino-A000066-datasheet.pdf
[3] InstallingAdditionalArduinoLibraries.pdf
[4] tavir_arduino_notebook.pdf
[5] http://arduinolearning.com/code/arduino-easy-module-shield-v1.php
[6] https://cdn.sparkfun.com/assets/home_page_posts/2/0/6/6/arduino_field_guide_copy.pdf
[7] https://eclipse2017.nasa.gov/eclipse-who-what-where-when-and-how
[8] https://en.wikipedia.org/wiki/Solar_eclipse_of_August_21,_2017
[9] http://forefront.io/a/beginners-guide-to-arduino
[10] https://github.com/adafruit/Adafruit_Sensor
[11] https://hci.rwth-aachen.de/tiki-download_wiki_attachment.php?attId=1909
[12] https://learn.sparkfun.com/tutorials/what-is-an-arduino
[13] http://physics.kg.ac.rs/fizika/scopes/ARDUINO.pdf
[14] http://students.iitk.ac.in/eclub/assets/lectures/embedded14/arduino.pdf
[15] https://www.arduino.cc/en/Reference/HomePage
[16] https://www.arduino.cc/en/Reference/Wire
[17] https://www.arduino.cc/en/Tutorial/MasterWriter
[18] https://www.arduino.cc/en/Main/Products
[19] http://www.instructables.com/id/Intro-to-Arduino
[20] http://www.instructables.com/id/BMP280-Barometric-Pressure-Sensor
[21] http://www.instructables.com/id/How-to-Use-the-Adafruit-BMP280-Sensor-Arduino-Tuto
[22] https://www.introtoarduino.com/downloads/IntroArduinoBook.pdf