POLÍGONOS. Revista de Geografía
2016, nº 28, 161-181
e ISSN – 2444–0272

A PROTOTYPE OF AN OPEN HARDWARE-BASED AUTOMATIC PHOTOGRAPHIC CAMERA TO MONITOR SNOW COVER EVOLUTION AND WEATHER PHENOMENA IN THE CONTEXT OF THE FROZEN GROUND MONITORING: PERMARDUINO-CAMERA

M.A. DE PABLO
Depto. de Geología, Geografía y Medio Ambiente. Universidad de Alcalá
C. DE PABLO S.
Especialista en electrónica. Madrid
M. RAMOS
Departamento de Física y Matemáticas. Universidad de Alcalá
M. PRIETO
Departamento de Automática. Universidad de Alcalá

Recibido: 29/12/2015
Aceptado: 25/05/2016

ABSTRACT: The study of permafrost and active layer thermal behavior require continuous monitoring of ground temperatures as well as other additional parameters, including snow cover, because of its isolation effect when the thickness is enough high. Typical monitoring stations from Thermal State of Permafrost (TSP) network includes the air temperature monitoring to derive snow thickness. Moreover, its low snow deep accuracy, this device return data from one place each time. To know and study the snow onset, offset, duration and distribution is sometimes required and the use of automatic digital photographic cameras contribute to have an adequate approach. However, commercial automatic cameras are expensive. For that reason, we developed a robust, simple, low-cost, open hardware-based (Arduino) prototype of an automatic camera to take pictures and store them into an SD card (2 Gb) that allows more than 4 years of hourly continuous image acquisition. The device is powered by a small solar cell that charges a li-po battery. The firmware allows a detailed monitoring of the device and error detection, and configure the camera for its own. We used a TTL serial JPEG camera with CMOS ¼ inch sensor with 480x640 pixels in resolution, with NTCS video capacity to allow real-time camera focus. The camera acquires images on panchromatic and near IR band. The fully operative prototype has been tested in Antarctica, and it was finally installed in Byers Peninsula in order to study the snow cover evolution at the Limnopolar Lake CALM site. In this work we present the prototype and its firmware to allow others to develop their own cameras to monitor snow cover or any other phenomenological parameter.
KEY WORDS*:* Frozen ground; Snow cover; Digital automatic camera; Instrumentation.

PROTOTIPO DE CÁMARA FOTOGRÁFICA AUTOMÁTICA BASADA EN HARDWARE ABIERTO PARA MONITORIZAR LA EVOLUCIÓN DE LA CUBIERTA DE NIEVE Y FENÓMENOS METEOROLÓGICOS EN EL CONTEXTO DE LA VIGILANCIA DEL SUELO CONGELADO: PERMARDUINO-CÁMARA

RESUMEN: El estudio permafrost y la capa activa requiere de la medición continuada de la temperatura del terreno, y de otros parámetros adicionales, incluyendo el espesor de la capa de nieve debido al efecto aislante que puede tener. Las estaciones típicas de estudio térmico del permafrost (TSP en inglés), incluye la medida de la temperatura del aire a diferentes alturas para derivar el espesor aproximado de la capa de nieve. Más allá de la baja resolución de este método, estos termonivómetros sólo facilitan datos de un punto del territorio. Sin embargo, conocer y estudiar la evolución y distribución de la cubierta de nieve requiere en muchos casos del uso de cámara fotográficas automáticas, con el problema del alto coste de las mismas. Por ello, hemos desarrollado un dispositivo robusto, simple, de bajo coste, y basado en el uso de hardware libre (Arduino), capaz de tomar fotografías y almacenarlas en una tarjeta de memoria SD (2 Gb) con capacidad para más de 4 años de actividad continuada. Este dispositivo está alimentado por una placa solar y una batería Li-po. La cámara usada permite obtención de imágenes JPEG mediante el uso de un sensor CMOS de ¼ de pulgada e imágenes de hasta 480x640 píxeles de resolución. Además, dispone de salida NTCS de video que facilita las tareas de enfoque en el campo. La cámara adquiere imágenes en el rango pancromático e infrarrojo cercano. El prototipo funcional del dispositivo fue probado en la Antártida, e instalado finalmente en la península Byers para estudiar la evolución de la capa de nieve en el emplazamiento CALM Limnopolar Lake. En este trabajo se presenta el prototipo y su configuración para permitir a otros desarrollar sus propias cámaras para el seguimiento de la cubierta de nieve u otros fenómenos meteorológicos.
PALABRAS CLAVE: Suelos congelados, Nieve, Cámara digital automática, Instrumentación.

I. INTRODUCTION

The study of permafrost and active layer plays an important role on both periglacial environments knowledge and climate evolution monitoring (e.g., HINKEL, 1997; HARRIS *et al*., 2001; TURNER *et al*., 2007; RAMOS *et al*., 2008, 2009; VIEIRA *et al*., 2010; DE PABLO *et al.*, 2013, 2014a; BOCKHEIM *et al*., 2013). In fact, the monitoring of permafrost and active layer is one of the common research topics on different environments, including high mountain, polar and subpolar (e.g., BROWN *et al*., 2000; MATSUOKA and HUMLUM, 2003; NELSON *et al*., 2004; BOCKHEIM, 2006; MATSUOKA, 2006; NELSON and SHILOMANOV, 2009; VIEIRA *et al*., 2010). Research networks, like Ground Temperature Network –Permafrost (GTN-P), developed different monitoring protocols to allow the ground thermal monitoring and active layer thickness measurement such as the Thermal State of Permafrost (TSP) and Circumpolar Active Layer Monitoring (CALM), respectively (e.g., BROWN *et al*., 2000; MATSUOKA and HUMLUM, 2003; NELSON *et al*., 2004; MATSUOKA, 2006). Monitoring ground temperatures allow both to observe yearly evolution of the temperatures and long-standing trends. The second case requires decadal of continuous monitoring but it is the most interesting in the context of climate warming. However, the first case allows continuous data analysis to observe and establish short-time thermal behaviors, although to really understand the thermal evolution of the ground requires to monitor other parameters what could influence on the ground temperature. Then, air and surface temperatures

and snow cover, are the most common additional variables required to monitor, such as proposed at the TSP protocol. On the other hand, more complete monitoring stations include also radiation flux, precipitation, wind speed and other meteorological variables. In general, the use of digital cameras to monitor weather events are not common in these TSP monitoring stations. High costs and relatively low resolution do not compensate, in general, the qualitative information provided by the pictures. Only when to determine timely meteorological events is required to understand the thermal events or general thermal behavior observed in the data, or when the data from a monitoring station require to be spatially extended, a digital automatic camera is required. Because the TSP stations usually do not monitor snow cover deep, but air temperature at different height in a mast from which data could be derived snow cover deep by different procedures (e.g., DANBY and HIK, 2007; LEWKOWICZ, 2008). Then, the use of cameras also provides a way to check the derived snow deep, that it is really relevant when trying to understand the ground thermal behavior. In fact, the snow cover plays an important role on the ground thermal evolution due to its isolation effect (e.g., GOODRICH, 1982, ZHANG *et al*., 2003; ZHANG, 2005) or its contribution to runoff and groundwater during the snow melting season (e.g., DEWALLE and RANGO, 2008). However, the snow cover deep and duration changes areally, and the snow cover deep derived at a TSP station does only provide punctual information. Then, when trying to understand the local variability of the ground thermal behavior between different nearest TSP stations, or of the active layer thickness in a CALM site, the analysis of the snow onset, offset, duration and thickness is required not punctually, but areally. To solve this, multiple snow deep monitoring devices are required, or one of them together with an automatic digital camera could be used in homogeneous terrains to provide a good approach to these parameters of the snow cover.

However, both, the installation of multiple snow deep monitoring devices and commercial digital cameras result on expensive monitoring stations what which goes against the philosophy of the TSP stations of simplicity and low costs in order to allow the research teams to extend the network to have more monitoring sites at many as possible periglacial environments. To avoid this problem, we develop a robust low-cost and low maintenance device on the use of open-hardware electronics (Arduino, http://arduino.cc/), and a miniature digital camera sensor. This device, that we named PERMARDUINO-CAMERA, could be included or be an independent element of a PERMARDUINO station, a low-cost, open hardware based station to monitor ground, surface and air temperatures and snow cover deep (de Pablo *et al*., 2014b, 2015). PERMARDUINO-CAMERA reduces a 90% the costs of a

standard commercial outdoor scientific camera with the same resolution, contributing to allow to extend the use of digital phenomenological cameras inside the different monitoring research stations (mainly TSP and CALM).

Here we present the first prototype we developed, showing and explaining its characteristics: camera resolution, memory capacity, and maintenance tasks under cold and harsh weather conditions. We show the electronic circuit of the prototype in order to allow other colleagues to produce their own devices as well as to modify by their own our design in order to fit their needs. The necessary firmware is also showed and explained to make possible to others to easily introduce modifications. Finally, after to show the results of the first test of few days of activity in a monitoring site in Byers Peninsula, Livingston Island, Antarctica, in January 2015, we explain the future possible modification could be applied to the device to improve it characteristics.

II. PERMARDUINO CAMERA

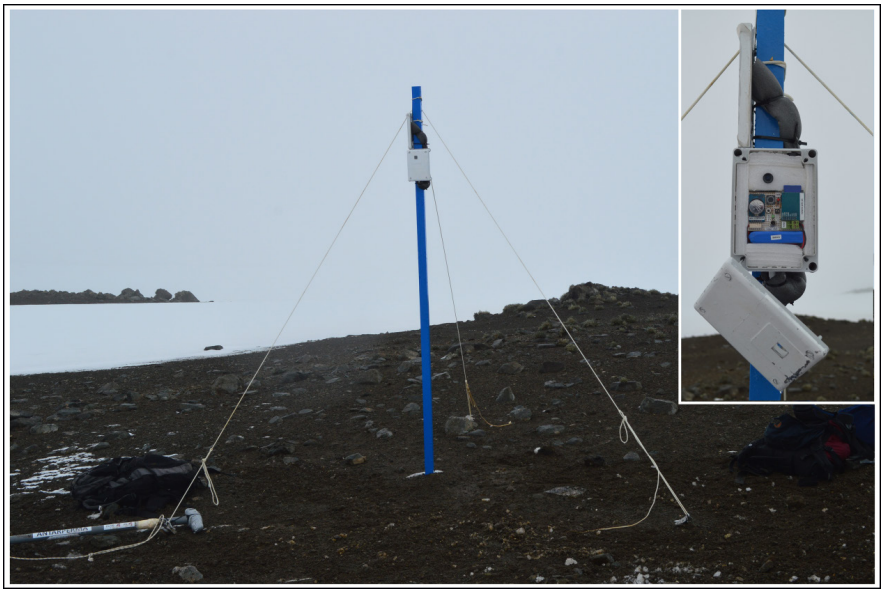
II.1. General design

PERMARDUINO-CAMERA device (FIGURE 1) is a prototype of a phenomenological camera complimentary to both the monitoring devices of a TSP station and a CALM site (e.g., BROWN *et al*., 2000; MATSUOKA and HUMLUM, 2003; NELSON *et al*., 2004; MATSUOKA, 2006). This device results from our efforts on the PERMARDUINO project to develop low-costs, low maintenance and open hardware-based ground thermal monitoring stations (DE PABLO *et al*., 2014b, 2015), based on our experience on the maintenance of this type of monitoring stations in South Shetland Island, Antarctica. The design of the PERMARDUINO-CAMERA tried to provide the next characteristics to the device: (1) low cost, (2) simple design, (3) easy construction, (4) easy use, (5) easy installation, (6) simple maintenance, (7) easily configurable, (8) robust to weather conditions, and (9) small in size and light in weight). Those characteristics are shared with the PERMARDUINO thermal monitoring station (DE PABLO *et al*., 2014b).

The PERMARDUINO-CAMERA device includes a CMOS sensor in order to take pictures automatically that are stored into a SD card, and it is powered by a small battery that is charged with a solar panel. The camera is controlled by a microcontroller and use a real time clock and a battery and its charger. The complete device, except the solar cell, is housed in a 160x110x70 mm waterproof semi-rigid plastic case that has been adapted to be installed in a (wood) mast. A hole has been open in the front of the case and protected with double glass (this one used to protect digital cameras LCD screens) to allow the camera to take the pictures without expose it to the rush environmental

conditions. Except the solar cell and its connection cable, no external elements are required.

Figure 1. *Picture of the PERMARDUINO-CAMERA mounted in a 2 m high mast installed in Byers Peninsula, Antarctica, and detail of the house opened for configuration and SD memory card replacement.*



SOURCE: The authors

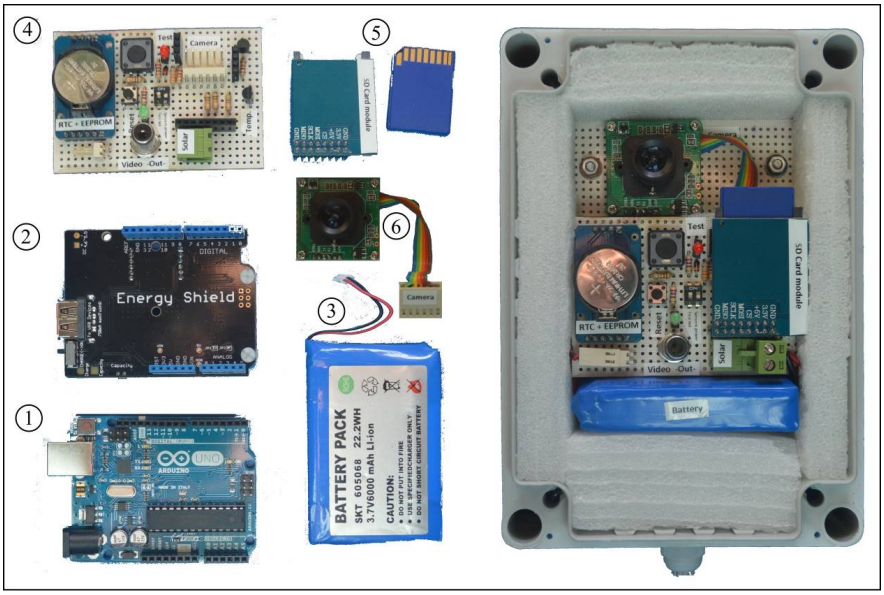
II.2. Core, power and electronics

PERMARDUINO-CAMERA device (FIGURE 2) is based on Arduino: an open-hardware board that uses a microcontroller to control both digital and analogical inputs and outputs (http://arduino.cc/), easily coded in C language, and extended by adding stackable shields or any kind of electronic devices. PERMARDUINO-CAMERA use an Arduino Uno board, equipped with and ATMega328P microcontroller (from ATMEL) with 32 Kb of flash memory, 14 digital inputs/outputs, 6 analog inputs, and provide I2C, UART and ICSP communications. The board is also equipped with a 16 MHz crystal oscillator, a USB connection, a power jack, and an ICSP header (https://www.arduino.cc/en/Main/ArduinoBoardUno). It operated at 5V and could be powered by an external supply of 6 to 20V. Serial communications over USB are able thanks to a FTDI chip on the board, what made possible virtual com port to software on the computer to program it by the use of Arduino Integrated Development Environment (IDE) freeware

(https://www.arduino.cc/en/Main/Software), such as we will see below. Arduino Uno board, is about 50 mm wide and 63 mm long, and it contains 2 arrays of female pins that provide direct connection to the inputs/outputs of the microcontroller.

We used these characteristics to connect a commercial shield to charge a Lithium polymer-based battery: Energy shield (FIGURE 2) by Seed Studio (http://www.seedstudio.com). This shield allows to charge Li-po batteries by the use of solar cells. This shield contains its own arrays of female pins to allow to stack other shield or direct access to inputs/outputs in the Arduino board. PERMARDUINO-CAMERA is powered by a 3.7V and 6,000 mAh Li-po battery combine with a 3W solar cell (5V and 660mA), 180x140x20mm in size.

Figure 2. *(Left) Elements of the PERMARDUINO-CAMERA device: (1) Arduino Uno board, (2) Energy shield, to charge a (3) 3.7V lithium-polymer battery by a 3W solar panel, (4) camera shield that includes a (5) SD memory card holder, and (6) the NTCS jpeg serial camera. (Right) These elements already stacked and adjusted into a waterproof electronics housing box, including vibrations and thermal isolator.*



SOURCE: The authors

The used camera is a JPEG camera with a CMOS ¼ inch sensor that provides a configurable resolution, with a maximum resolution of VGA 480x640

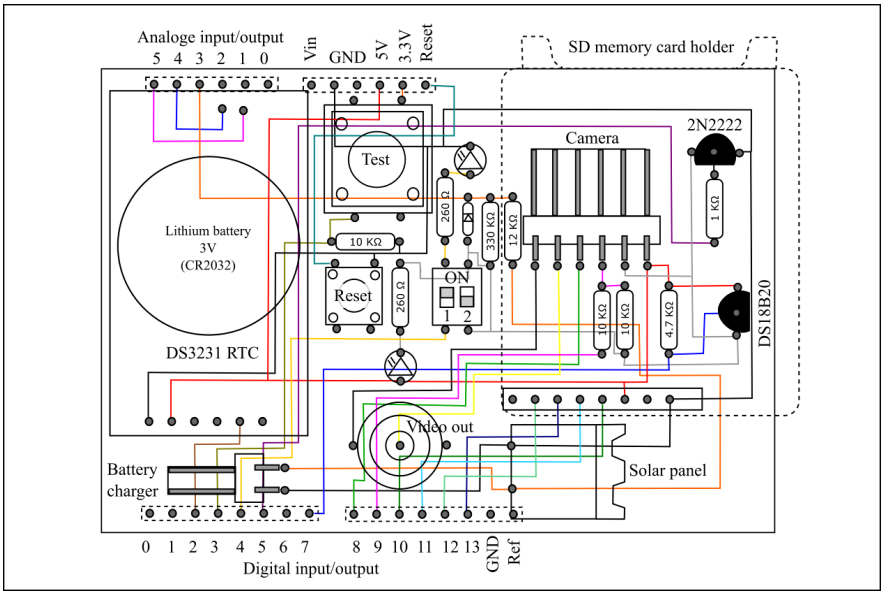
pixels (0.7 Mpixels), with progressive scan mode, automatic white balance and exposure adjustment, and 60º of vision angle. The sensor does not have IR filter, so, it captures data in the visible plus near IR. This camera is mounted in a 32x32 mm module, and it provides NTCS video capacities, that allow to adjust focus in real time by connecting to a device with AV-in ports or to a computer by means of any standard video-to-USB connector. This camera communicates with Arduino board by a configurable 38400 to 115200 baud by TTL serial port.

To acquire the pictures at regular times, a board with a DS1337 Real Time Clock (RTC) chip to remain accurate timing and a 3V backup coin cell battery, was required. This RTC allow to program alarms that advise the Arduino board to activate, take the picture, save it and deactivate again. Pictures, in JPEG format, are stored into a 2 Gb SD memory card (FAT 16 format) slot at a SD card module connected to the Arduino board by ISCP communications protocol. The pictures files names are controlled by firmware and they are named with an 8 characters long code based on the acquirement date including year(Y), month (M), date (D) and hour (H) (YYMMDDHH.jpeg; e.g., 15011513.jpeg is the picture acquired in January 15, 2015 at 13:00h). In this way, images could be easily later organized by filename in a computer. The memory card we use have enough capacity to store about 40,000 pictures (it is about 1,600 days of monitoring with hourly acquirement of pictures of 50 Kb in size), although it additionally stores an ASCII file, excel-compatible ASCII file, CSV formatted, with device control data, such as date, time, inner temperature, device temperature, picture name, detected error, etc.

The, a 2Gb SD ensure about 4 year of continuous monitoring at full resolution and device capacities. In fact, device temperature, a DS18B20 digital temperature sensor has been used, what only required a pull-up 4.7 kohms resistor. For maintenance reasons during working operations, our device includes a 3 mm in diameter red led what is switched by another digital pin of the Arduino board controlled by firmware to show the user when the device is measuring. This led is connected to a micro-switch what could be turned off by the user to save battery during the stand alone period of work of the device, and turned on during testing and maintenance operations. We will see latter that this led is used by firmware to show error detection. A pushbutton was also added in order to allow the user to check the device status (device activity and error on execution time). An additional small pushbutton has been added to the shield for reseat purposes (for example after an error detection, or SD card connection). Other small pushbuttons have been added to reset the device.

All the electronics components (TABLE 1) were soldered into a PCB board by simple circuits (FIGURE 3). To constitute a shield stackable to the Arduino and Energy shields. The size of this shield is the same than the Arduino board, 50x63mm, although it does not include the camera that is connected to this board by a parallel cable due to space requirements inside the case. One all the elements are stacked, the electronics part of the device is a module and small, light, robust and compact block of 50x63x60 mm in size, allowing enough space inside the case to hold the battery and the camera.

Figure 3. *Components and circuit of the camera shield (element '4' in Figure 2), stackable to the Arduino Uno board (and compatible), what constitute the main development of this project, together with the corresponding firmware.*



SOURCE: The authors

Table 1. *Electronics components required for the PERMARDUINO-CAMERA device*

#	*Component*	#	*Component*
1	Arduino Uno board	2	Resistor 260Ω
1	Energy Shield	1	Resistor 1 KΩ
1	3.7V 6000 mAh Li-po battery	1	Resistor 4.7 KΩ
1	Protoboard (2.54 mm spaced)	3	Resistor 10 KΩ
1	SD card holder	1	Resistor 12 KΩ
1	2Gb SD card	1	Resistor 330 KΩ
1	NTCS jpeg serial camera	1	Diode Zener 5V
1	DS3231 RTC	1	Transistor 2N2222

1	DS18B20 temperature sensor	1	Led Red 3mm
1	2 pins connector (male & female)	1	Led Green 3mm
1	6 pins connector (male & female)	1	Micro-switch
1	2 pins connector (male & female)	1	Video connector
1	Array of male pins connector	1	Push button mini
1	Array of female pins connector (40)	1	Push button normal

SOURCE: The authors

II.3. Firmware

The PERMARDUINO-CAMERA device is controlled by a firmware written in C language by the use of Arduino IDE version 1.6.5 (available for Windows, Mac and Linux operative systems). The freeware Arduino IDE allows the firmware compilation and its upload to the microcontroller in the Arduino board by means of an USB standard cable. The firmware contents are limited by the amount of free memory available in the microcontroller (30Kb), used for both firmware and volatile memory during the program running. However, this memory is enough to run a program to control the image acquirement and its save into an SD card.

The firmware (APPENDIX 1) is divided into different sections 4 different sections what run in sequence by the microcontroller: (1) device configuration, that contains information required by the microcontroller to assigns tasks to each digital/analogical pin as well as to read additional procedures: (a) libraries definition, (b) analogue pins definition, (c) digital pins definition, (d) variables definition, (e) constants definition, (f) log and images files definition, and (g) libraries configuration; (2) Device initialization, that (i) power on the camera and sensors, (j) check the RTC, (k) check the SD card, (l) check the data file, (m) write the datafile header is required, (n) check camera and its configuration, and (o) check the temperature sensor and configure; (3) main device operation, that (s) read the present time, (p) check for error -if the pushbutton was pressed, (q) take a picture and read the sensors –if the pushbutton was not pressed and it is the measurement time, (r) save the data, (s) changes to low-power consumption and (t) sleep during the waiting period or the pushbutton is pressed; and (4) Procedures, that contains tasks required in any moment during the device configuration (Section 2) or the main program running (Section 3), like sensor reading, data saving, device checking, etc. Section 1 and 2 run only one time after a startup or a reset (by pushing the small pushbutton in the device shield); Section 3 runs into an infinite loop until an error, a reset or a power down; and Section 4 runs only when called from any of the other sections.

This firmware could be modified by each user in the Arduino IDE and uploaded as many times as required to the board to fit the user requirements. And could be also modified to add other sensors, increase/reduce the measurements frequency, change the datafile data stamp, etc. The content, design, and workflow of the firmware depend on the user requirements, the used sensors and hardware stacked and connected to the Arduino board. Extensive information about how to wire devices to Arduino, and how to write the corresponding firmware could be found in the Arduino forum webpage as well as in many books (e.g., MARGOLIS, 2011; BANZI, 2011; EVANS, 2011; TIMMIS, 2011; MONKS, 2012; GLETZ and DI JUSTO, 2012; DI JUSTO and GERTZ, 2012; BAYLE, 2013; BLUM, 2013). On the other hand, the use of different sensors and electronics elements could require long codes, what are usually already available such as libraries what reduce the lines of code required to be write to the user to prepare the firmware.

III. FIRST TESTS AND PRELIMINARY RESULTS

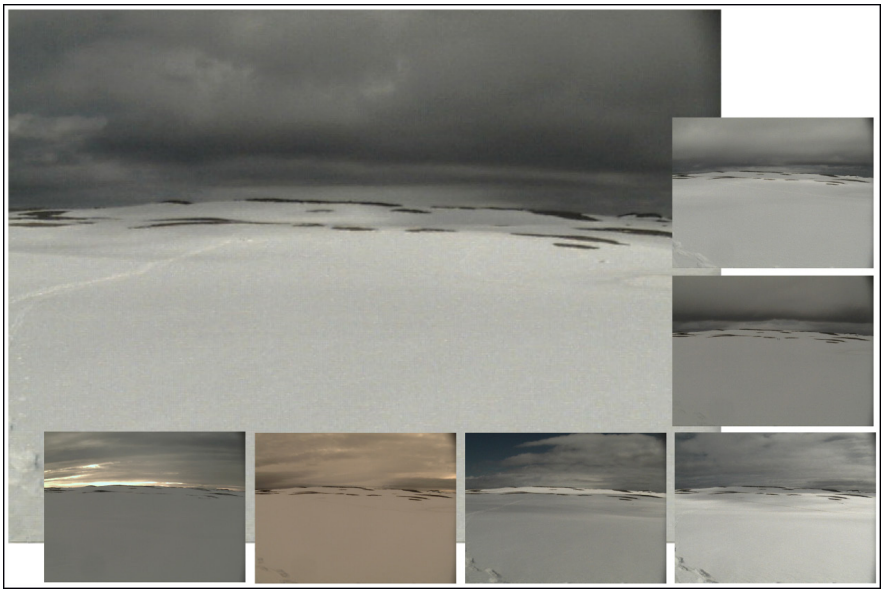
Although a 6 months-long testing period was applied to the device in the laboratory, in order to test the prototype under real cold, rough and harsh weather conditions, the device was installed between December 25th 2014 and January 3rt 2015, at the Spanish Antarctic Station «Gabriel de Castilla» in Deception Island (South Shetland Archipelago), Antarctica (62°58′38″S, 60°40′33″W). During this period, we examined the device behavior and tried to detect any problem in the electronics, firmware and housing. Due to the excellent results in spite of the cold, windy and snowy conditions, in January 2015, finally we installed the first prototype of PERMARDUINO-CAMERA device in its definitive emplacement in Byers Peninsula, Livingston Island, Antarctica, in early 2015, during the Spanish Antarctic Campaign (FIGURE 4).

The device was installed in a low slope terrain at the shore of the Limnopolar Lake (62°38′59.7″S, 61°06′09.5″W). This site was selected because from this site there are a complete view of the Limnopolar Lake CALM site that we monitor since 2009 (DE PABLO *et al*., 2010, 2013, 2014a), and where we observed an increase on the snow cover that is affecting the thermal behavior of the ground (DE PABLO *et al*., 2016, submitted). We already have a commercial automatic digital camera, but, although it has the same resolution (480x640 pixels), it is further from the CALM site since it is shared with other project that want to monitor the Limnopolar Lake water level and snow cover evolution (e.g., CAMACHO *et al*., 2014).

After its installation in its final emplacement in a 2 m height wood mast ensured by 3 cord clamps, we focused it with the help of a laptop and a video recorder software. We leaved the device working for few days meanwhile we

completed different tasks in the TSP and CALM site at the Limnopolar Lake inside the view field of the camera. Few days later, and before to leave the site and the instrumentation for one year, we checked how it was working. The device recorder pictures as expected (FIGURE 4) without any problem in spite of the cold and windy conditions. During the sunny days, no condensation was registered in the pictures due to the glass that isolate the camera from the environment. However, since the camera was placed pointing to the west, some of the images captured the Sun, saturating partially the images, but without disturbing image of the rest of the scene, and the images are perfectly valid. In any case, reflects or flares were not recorded in the images.

Figure 4. *Examples at full resolution (480x640 pixels) and thumbnails of the images acquired by the camera in January 2015 once it was installed in Byers Peninsula, Antarctica, to take pictures (hourly) of the A25 CALM-S site.*



SOURCE: The authors

In spite of the short daytime duration in polar latitudes, we decided to leave the camera configured to take a picture each hour, also during the night, in order to check the consume under real low luminosity conditions to charge the batteries, as well as under the lower temperatures on nighttime, and memory card space.

IV. CONCLUSIONS

The PERMARDUINO-CAMERA is part of the PERMARDUINO project to develop an automatic device to monitor permafrost and active layer following the TSP and CALM international protocols. This project is based on the use of open hardware. A functional prototype of PERMARDUINO-CAMERA device is presented here. It is an automatic digital photographic camera to take pictures in cold and harsh weather conditions to replace expensive commercial ones, or, at least, to allow the installation of multiple of those by lower price t better monitor the snow cover and other meteorological phenomena. In our case, we develop it to help us to monitor the snow cover evolution and its spatial distribution in order to understand the spatial distribution of active layer thickness in a CALM site in Byers Peninsula, Antarctica.

PERMARDUINO-CAMERA use a TTL serial JPEG VGA 480x640 pixel in resolution camera provided by a CMOS ¼ inch sensor to acquired pictures in the panchromatic and near IR band with an angle vision of 60º. This camera is connected to our shield stackable to the Arduino Uno (and compatible) board, and an Energy shield to changer by a solar cell a 3.7V li-po battery. The shield contains all the electronics required to connect the camera to a screen to see video on real time to focus the camera, as well as a SD memory card holder where the images and auxiliary data are stored. The device is programmed, by the use of C language in Arduino IDE, to take pictures hourly, although it could be easily configurable the users by the modification of the firmware we provided here.

The design of PERMARDUINO-CAMERA device tried to provide the next characteristics to the station: (1) low cost, (2) easy design, (3) easy construction, (4) high accuracy in data measurement, (5) easy use, (6) easy installation, (7) simple maintenance, (8) easily configurable, (9) robust to weather conditions, and (10) small in size and light in weight.

The simple design and the low knowledge requirements on electronics and programming make this device a perfect solution for researches who want to develop their own experiments and automatic digital cameras to monitor weather phenomena (among other) on periglacial environments.

ACKNOWLEDGEMENTS

Authors want to thank to the Arduino Team for the development of Arduino open-hardware project, a way to open the technology to the society, and to the Arduino user's community for their help and collaborations to improve PERMARDUINO-CAMERA code. We also want to than to the Spanish Polar Committee, The Spanish Polar Research Program, and the whole crew of the

«Juan Carlos I» Spanish Antarctic Station for their help to develop the fieldtrip in Byers Peninsula, Livingston Island, Antarctica. We also want to thank to Cayetana Recio for her help during the camera installation in Antarctica.

This project was partially supported by PERMASNOW (CTM2014-52021-R), ANTARPERMA (CTM2011-15565-E), PERMAPLANET (CTM2009-10165E), PERMAMODEL (POL2006-01918), and PERMATHERMAL projects from the Ministry of Economy and Competitiveness, and the Ministry of Education and Science, Government of Spain.

REFERENCES

BANZI, M. (2011): *Getting started with Arduino*. O'Reilly Media, Inc.

BAYLE, J. (2013): *C programming for Arduino*. O'Reilly Media, Inc.

BLUM, J. (2013): *Exploring Arduino: Tools and Techniques for Engineering Wizardry*. O'Reilly Media, Inc. BROWN, J.; NELSON, F.E.; and HINKEL, K.M. (2000): «The circumpolar active layer monitoring (CALM) program research designs and initial results». *Polar Geography*, 3, 165–258.

BOCKHEIM, J.G. (2006): «Permafrost distribution in the southern circumpolar region and its relation to the environment: a review and recommendations for further research». *Permafrost and Periglacial Processes*, 6, 27-45.

BOCKHEIM, J.; VIEIRA, G.; RAMOS, M.; LÓPEZ-MARTÍNEZ, J.; SERRANO, E.; GUGLIELMIN, M.; WILHELM, K.; and NIEUWENDAM, A. (2013): «Climate warming and permafrost dynamics in the Antarctic Peninsula region». *Global and Planetary Change*, 100, 215–223. doi: 10.1016/j.gloplacha.2012.10.018.

CAMACHO, A.; VILLAESCUSA, J.A.; ROCHERA, C.; and JØRGENSEN, S.E. (2014): «Modeling the response of the planktonic microbial community to warming effects in maritime antarctic lakes: Ecological implications». *Developments in Environmental Modelling*, 26(9). 231-250.

DANBY, R. and HIK, D. (2007): «Responses of white spruce (Picea glauca) to experimental warming at subsurface alpine treeline». *Global Change Biology*, 13, 437–451.

DE PABLO, M.A.; DE PABLO, C.; and RAMOS, M. (2014b): «A prototype of an open hardware-based device for active layer and frozen ground monitoring: PERMARDUINO». *4th European Conference on Permafrost*. Évora (Portugal). Abstracts, 444.

DE PABLO, M.A.; DE PABLO, C.; and RAMOS, M. (2015): «Improvements on PERMARDUINO prototype device for active layer and permafrost thermal monitoring, and automatic digital camera development». *VI Congreso Ibérico de la International Permafrost Association*. Valladolid, Spain. Abstracts, 23.

DE PABLO, M.A.; RAMOS, M.; VIEIRA, G.; and QUESADA, A. (2010): «A new CALM-S site on Byers Peninsula, Livingston Island, Antarctica». In: *Ambientes Periglaciares, Permafrost y Variabilidad Climática*: II Congreso Ibérico de la International Permafrost Association, edited by: BLANCO, J.J.; DE PABLO, M.A.; and RAMOS, M.; Servicio de Publicaciones de la Universidad de Alcalá, Alcalá de Henares, 153–159.

DE PABLO, M. A.; BLANCO, J. J.; MOLINA, A.; RAMOS, M.; QUESADA, A.; and VIEIRA, G. (2013): «Interannual active layer variability at the Limnopolar Lake CALM site on Byers Peninsula, Livingston Island, Antarctica». *Antarctic Science*, 25, 167–180. doi: 10.1017/S0954102012000818.

DE PABLO, M.A.; RAMOS, M. and MOLINA, A. (2014a): «Thermal characterization of the active layer at the Limnopolar Lake CALM-S site on Byers Peninsula (Livingston Island), Antarctica». *Solid Earth*, 5. 721-739.

DE PABLO, M.A.; RAMOS, M. and MOLINA, A. (2016): «Snow cover evolution at the Limnopolar Lake CALM-S site on Byers Peninsula, Livingston Island, Antarctica, 2009-2014». *Catena*. In press

DEWALLE, D. and RANGO, A. (2008): *Principles of Snow Hydrology*. New York: Cambridge University Press. 428 pp. ISBN 978-0-521-82362-3.

DI JUSTO, P. and GERTZ, E. (2012): *Atmospheric monitoring with Arduino*. O'Reilly Media, Inc.

EVANS, B. (2011): *Beginning Arduino programming*. Apress Inc.

GERTZ, E. and di Justo, P. (2012): *Environmental monitoring with Arduino*. O'Reilly Media, Inc.

GOODRICH, L.E. (1982): «The influence of snow cover on the ground thermal regime». *Canada Geotechnical Journal*, 19, 421 – 432.

HARRIS, C.; HAEBERLI, W.; VONDER MÜHLL, D.; and KING, L. (2001): «Permafrost monitoring in the high mountains of Europe: the PACE project in the global context». *Permafrost and Periglacial Processes,* 12(1), 3–11.

HINKEL, K.M. (1997): «Estimating seasonal values of thermal diffusivity in thawed and frozen soils using temperature time series». *Cold Regions Science and Technology*, 26, 1–15.

LEWKOWICZ, A. G. (2008): «Evaluation of miniature temperature-loggers to monitor snowpack evolution at mountain permafrost sites, northwestern Canada». *Permafrost and Periglacial Processes*, 19, 323–331. doi: 10.1002/ppp.625

MARGOLIS, M. (2011): *Arduino cookbook* (2nd Edition). O'Reilly Media, Inc.

MATSUOKA, N. (2006): «Monitoring periglacial processes: towards construction of a global network». *Geomorphology*, 80, 20–31.

MATSUOKA, N. and HUMLUM, O. (2003): «Monitoring periglacial processes: new methodology and technology». *Permafrost and Periglacial Processes*, 14, 299–303.

Monks, S. (2012): *Programming Arduino: Getting started with sketches*. McGraw-Hill.

NELSON, F.E.; SHIKLOMANOV, N.I.; HINKEL, K. and CHRISTIANSEN, H. (2004): «Introduction: the Circumpolar Active Layer Monitoring Network (CALM) workshop and CALM II program». *Polar Geography*, 28, 253–266.

NELSON, F.E. and SHIKLOMANOV, N.I. (2009): «The Circumpolar Active Layer Monitoring Network–CALM III (2009–2014): long-term observations on the ''Climate-Active Layer-Permafrost System''». En BLANCO, J.J.; DE PABLO, M.A. and RAMOS, M. (eds.) *Ambientes periglaciares, permafrost y variabilidad Climática*: *II Congreso Ibérico de la International Permafrost Association*. Alcalá de Henares: Servicio de Publicaciones de la Universidad de Alcalá, 9–14.

RAMOS, M.; HASLER, A.; VIEIRA, G.; GRUBER, S. and HAUCK, C. (2009): «Setting up boreholes for permafrost thermal monitoring on Livingston Island in the Maritime Antarctic». *Permafrost and Periglacial Processes*, 20, 57–64.

RAMOS, M.; VIEIRA, G.; BLANCO, J.J.; GRUBER, S.; HAUCK, C.; HIDALGO, M.A. and TOME, D. (2008): «Thermal active layer monitoring in two different sites on Livingston Island during the last seven years: a comparative study». In *Proceedings of the Ninth International Conference on Permafrost*, Fairbanks, Alaska. Fairbanks, AK: University of Alaska, Institute of Northern Engineering, 1463–1467.

TIMMIS, H. (2011): *Practical Arduino engineering*. Apres Inc.

TURNER, J.; LACHLAN-COPE, T.A.; COLWELL, S.; MARSHALL, G.J. and CONNERLLEY, W.M. (2007): «Significant warming of the Antarctic winter troposphere». *Science*, 131, 1914–1917.

VIEIRA, G.; BOCKHEIM, J.; GUGLIELMIN, M.; BALKS, M.; ABRAMOV, A.; BOELHOUWERS, J.; CANNONE, N.; GANZERT, L.; GILICHINSKY, D.A.; GOTYACHKIN, S.; LÓPEZ-MARTÍNEZ, J.; MEIEKLEJOHN, I.; RAFFI, R.; RAMOS, M.; SCHAEFER, C.; SERRANO, E.; SIMAS, F.; SLETTEN, R. and WAGNER, D. (2010): «Thermal state of permafrost and active-layer monitoring in the Antarctic: advances during the International Polar Year 2007–2009». *Permafrost and Periglacial Processes*, 21, 182–197.

ZHANG, T. (2005): «Influence of the seasonal snow cover on the ground thermal regime: An overview». *Rev. Geophys*.; 43. RG4002. doi:10.1029/2004RG000157

ZHANG, T.; BARRY, R.G.; KNOWLES, K.; LING, F.; and ARMSTRONG, R.L. (2003): «Distribution of seasonally and perennially frozen ground in the Northern Hemisphere». In: *Proceedings of the 8th International Conference on Permafrost*, 21-25 July 2003, Zurich, Switzerland [Phillips, M.; S.M. Springman, and L.U. Arenson (eds.)]. A.A. Balkema, Lisse, the Netherlands, pp. 1289–1294.

APPENDIX 1: COMMENTED PERMARDUINO-CAMERA (VERSION 1.0) FIRMWARE, READY TO COPY AND PASTE ON ARDUINO IDE.

```
/*
************************************
  PERMARDUINO CAMERA 1.0
  M.A. de Pablo & C. de Pablo, 2015

 Arduino-based phenonenological camera

 ---------------------------------------
 Version 1.0  2015-01-10  v20150110
 ************************************
 */

 // DEVICE CONFIGURATION
// Libraries definition
#include <Wire.h>                       // Library for I2C communications protocol
#include <DS3231.h>                     // Library for DS3131 RTC manage-ment
#include <Adafruit_VC0706.h>            // Library for TTL serial JPEG camera
#include <SoftwareSerial.h>                     // Library for serial communications
#include <OneWire.h>                    // Library for 1-Wire communications protocol
#include <DallasTemperature.h>          // Library for DS18B20 tempera-ture sensor
#include <SD.h>                                 // Library for SD card management
#include <avr/sleep.h>                  // Library for sleep and power save
```

```
// Analogue pins definition
const byte BatteryPin = 0;                    // Battery voltage - Analog pin 0
const byte SolarCellPin = 3;                  // Solar cell voltage - Analog pin 1

// Digital pins definition
//const byte RTCAlarmPin = 2;                 // Not required to be declared
//const byte TestButtonPin = 3;               // Not required to be declared
const byte TestLedPin = 4;                    // Device working led pin –Digital pin 3
const byte PowerPin = 5;                      // Activation pin to power the camera – Digital pin 4
const byte TempPin = 7;                       // Inner temperature sensor - Digital pin 4
const byte chipSelect = 10;                   // SD card configuration pin – Digital pin 10

// Variables definition
int Year;                                     // Year information
byte Month;                                   // Month information
byte Date;                                    // Day information
byte Hour;                                    // Hour information
byte Minute;                                  // Minute information
byte Second;                                  // Second information
boolean WorkNow = true;                       // Cancel sleep cycle
unsigned int n = 0;                           // Measurements counter
volatile byte e = 0;                          // Error type
unsigned int batt;                            // Battery voltage
unsigned int solar;                           // Solar cell voltage
float temp;                                   // Device temperature data
long picfilesize;                             // Pictures filename size
uint8_t imgsize;                              // Pictures file size
int32_t time;                                 // Pictures acquisition time

// Constant definition
// No constants

// Log files configuration
File datafile;                                // File to store the acquired data
char filename1[] = "DataCam.csv";             // Name of the file to store the data
File imgFile;                                 // File to store the acquired picture
char filename2[13] = "YYMMDDHH.jpg";          // Name of the file to store the picture
char symbols[] = ";:/ ";                      // Constant symbols used during data record

// Libraries configuration
DS3231 RTC;                                   // Configure library to read the Real Time Clock
SoftwareSerial cameraconnection = SoftwareSerial(8, 9);   // Configure camera communica-tions
Adafruit_VC0706 cam = Adafruit_VC0706(&cameraconnection);  // Initialize the camera
OneWire oneWire(TempPin);                     // Configure library to read device temperature sensors
DallasTemperature TempBus(&oneWire);          // Initialize the temperature sensor bus
DeviceAddress tempsensoraddress;              // Define the temperature sensor address

// DEVICE INITIALIZATION
void setup() {
 pinMode(PowerPin, OUTPUT);                   // Configure the power pin
 pinMode(TestLedPin, OUTPUT);                 // Configure the working led
 pinMode(chipSelect, OUTPUT);                 // Configure SD card

 digitalWrite(TestLedPin, HIGH);              // Activate the led

 Wire.begin();                                // Initialize I2C communications
 RTC.begin();                                 // Initialize the RTC
 if (! RTC.begin()){                          // Check if RTC is running
  e = 1;                                      // Error #01: RTC is not running
  error();                                    // Blink the led and freeze the device
  return;
 }
 ReadTime();                                  // Read the present time from RTC
 if (Year < 2014) {                           // Check if RTC is about up to date
  e = 2;                                      // Error #02: RTC is out of date
```

```
 error();                                                    // Blink the led and freeze the device
 return;
}
if (! SD.begin(chipSelect) ) {                  // Check if the card is present and can be initialized:
 e = 3;                                  // Error #03: Unable to set SD card
 error();                                // Blink the led and freeze the device
 return;
}
if (! SD.exists(filename1) ) {                  // Check and/or create the logdata file on SD Card
 datafile = SD.open(filename1, FILE_WRITE);      // Create the datafile if it doesn't exist
 datafile.close();
 if (! SD.exists(filename1) ) {                 // Check if datafile was created
  e = 4;                                 // Error #04: Unable to create a file on SD card
  error();                                    // Blink the led and freeze the device
  return;
 }
 datafile = SD.open(filename1, FILE_WRITE);      // Open the datafile
 if (datafile){                                          // Check if datafile could be open
  datafile.println(F("Permarduino_camera"));          // Save a file header (I)
  datafile.println(F("d_t;Num;Vcc;Bat1;Bat2;Sol1;Sol2;DeT;Pic;P_s;P_d"));      // Save a file header (II)
  delay(30);                                  // Give enough time to complete the file writing
  datafile.close();                                   // Close the datafile
 }
 else {
  e = 5;                                  // Error #05: Unable to write on data file
  error();                                // Blink the led and freeze the device
  return;
 }
}

// Sensors initilization, configuration and test
digitalWrite(PowerPin, HIGH);                   // Turn on power for sensors
delay(5000);                                    // Delay to ensure the signal stabilization from sensors
if (cam.begin()){}                              // Initialize the camera
else {
 e = 6;                                 // Error #06: Unable to initialize the camera
 error();                               // Blink the led and freeze the device
 return;
}
cam.setImageSize(VC0706_640x480);               // Set camera resolution to 640x480 pixels
imgsize = cam.getImageSize();                   // Take the camera image resolution
if (imgsize == VC0706_640x480){                 // Check if the resolution is correct
 Serial.println("640x480");
}
else {
 e = 7;                                 // Error #07: Unable to set camera pictures resolution
 error();                               // Blink the led and freeze the device
 return;
}
delay(3000);                                                // Delay to ensure the signal stabilization from sensors
if (! cam.takePicture()){                       // Try to take a picture
 e = 8;                                 // Error #08: Unable to take a picture
 error();                               // Blink the led and freeze the device
 return;
}
TempBus.begin();                                // Initialize the Temperature sensor on OneWire bus
delay(100);                                                 // Delay to ensure the signal stabilization from sensors
uint8_t tempsensoraddress[8] = {};              // Configure temperetature sensor address
TempBus.getAddress(tempsensoraddress, 0);               // Read the temperature sensor address
TempBus.setResolution(tempsensoraddress, 12);   // Set DS18B20 sensor precision to 12 bits

digitalWrite(PowerPin, LOW);                    // Turn off the power for sensors
digitalWrite(TestLedPin, LOW);                  // Turn off test led
delay(500);
CheckDevice();                                  // Check other not fundamental error
RTC.enableInterrupts(EveryHour);                        // Set the working alarm for the device (hourly)
```

```
}

// DEVICE OPERATION
void loop() {
 ReadTime();                          // Read the present time
 if (WorkNow == false){               // Shows device status if test button was pressed
  CheckDevice();                      // Show device status and higher error code
  delay(100);
 }
 if (WorkNow == true){                // Check if it is time to measure
  RecordData();                       // Read the sensors, take a picture and save the data
  delay(100);
  WorkNow = false;                    // Made the device be ready for the next measurement
 }

 attachInterrupt(0, WakeUp, FALLING);          // Activate again the interrupt from the RTC alarm
 attachInterrupt(1, CheckNow, CHANGE);         // Activate again the interrupt from the test button
 RTC.clearINTStatus();                         // Ensure RTC is working

 delay(300);
 sleepNow();                          // Send the arduino to sleep
}

// PROCEDURES
// Sensors reading
void RecordData(){
 digitalWrite(PowerPin, HIGH);                 // Turn on the sensor's power
 delay(1000);                         // Give time to sensors to stabilize their reading
 n = n + 1;                           // Update measurement counter
 SD.begin(chipSelect);                         // Configure SD card
 datafile = SD.open(filename1, FILE_WRITE);    // Open the file to store the data
 if (datafile){                                // Try to open the datafile
  printDigits(Date);                  // Save present day
  datafile.print(symbols[2]);
  printDigits(Month);                 // Save present month
  datafile.print(symbols[2]);
  datafile.print(Year);                        // Save present year
  if (Year < 2015) {                  // Check if RTC is up to date
   e = 2;                             // Error #02: RTC is out of date
  }
  datafile.print(symbols[3]);
  printDigits(Hour);                  // Save present hour
  datafile.print(symbols[1]);
  printDigits(Minute);                // Save present minute
  datafile.print(symbols[1]);
  printDigits(Second);                // Save present second
  datafile.print(symbols[0]);
  datafile.print(n);                  // Save measurement number (counter)
  datafile.print(symbols[0]);
  datafile.print(readVcc());          // Read and save microcontroler volt-age
  datafile.print(symbols[0]);
  datafile.print(readTemp());               // Read and save microcontroler inner temperature
  datafile.print(symbols[0]);
  delay(10);
  batt = analogRead(BatteryPin);            // Read the battery voltage
  datafile.print(batt);                     // Save battery voltage (raw)
  datafile.print(symbols[0]);
  datafile.print(2 * (readVcc() * batt) / 1023);     // Save the battery voltage (in volts)
  datafile.print(symbols[0]);
  delay(10);
  solar = analogRead(SolarCellPin);         // Read the solar cell voltage
  datafile.print(solar);                           // Save solar cell voltage (raw)
  datafile.print(symbols[0]);
  datafile.print(0.00666 * solar);          // Save solar cell voltage (in volts)
  datafile.print(symbols[0]);
```

```
 delay(10);
 TempBus.requestTemperatures();                    // Activate temperature sensor bus
 temp = TempBus.getTempCByIndex(0);                // Request device temperature
 datafile.print(temp);                             // Save device temperature data
 datafile.print(symbols[0]);
 if ((temp == -127)||(temp == 85)){                // Check for errors on the temperature sensor
  e = 9;                                           // Error #09: problems reading sensor
 }
 delay(10);

 TakeImage();                                      // Take a picture with the camera
 delay(10);
 datafile.print(filename2);                        // Save picture file name
 datafile.print(symbols[0]);
 datafile.print(picfilesize);                      // Save picture file size
 datafile.print(symbols[0]);
 datafile.print(time);                             // Save picture acquirement time
 datafile.println();
 delay(100);                                       // Give enough time to save all the data
 datafile.close();                                 // Close data file
}
digitalWrite(PowerPin, LOW);                       // Turn off the sensor's power
}

//Inner voltmeter
long readVcc() {
 long result;
 ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1)       // Read 1.1V reference against AVcc
 delay(2);                                         // Wait for Vref to settle
 ADCSRA |= _BV(ADSC);                              // Convert
 while (bit_is_set(ADCSRA,ADSC));
 result = ADCL;
 result |= ADCH<<8;
 result = 1126400L / result;                       // Back-calculate AVcc in mV
 return result;
}

//Inner temperature
long readTemp() {
 long result;
 ADMUX = _BV(REFS1) | _BV(REFS0) | _BV(MUX3);          // Read temperature sensor against 1.1V reference
 delay(2);                                         // Wait for Vref to settle
 ADCSRA |= _BV(ADSC);                              // Convert
 while (bit_is_set(ADCSRA,ADSC));
 result = ADCL;
 result |= ADCH<<8;
 result = (result - 125) * 1075;                   // Back-calculate temp in ºCx1000
 return result;
}

// Take a picture
void TakeImage(){
 filename2[0] = (Year/10)%10 + '0';                // Extract year number to image filename
 filename2[1] = Year%10 + '0';                     // Extract year number to image filename
 filename2[2] = Month/10 + '0';                    // Extract month number to image filename
 filename2[3] = Month%10 + '0';                    // Extract month number to image filename
 filename2[4] = Date/10 + '0';                     // Extract day number to image filename
 filename2[5] = Date%10 + '0';                     // Extract day number to image filename
 filename2[6] = Hour/10 + '0';                     // Extract hour number to image filename
 filename2[7] = Hour%10 + '0';                     // Extract hour number to image filename

 cam.takePicture();                                // Take a picture

 imgFile = SD.open(filename2, FILE_WRITE);         // Open the file to save the pic-ture in
 uint16_t jpglen = cam.frameLength();              // Get the size of the image (frame) taken
 picfilesize = jpglen;                             // Assign the image file size
```

```
 time = millis();                                    // Initialize a timecounter
 byte wCount = 0;                                    // For counting # of writes
 while (jpglen > 0) {                                // Check if there is image data to be saved
  uint8_t *buffer;                                   // Read 32 bytes at a time;
  uint8_t bytesToRead = min(32, jpglen);             // Readd image bytes
  buffer = cam.readPicture(bytesToRead);             // Buffer image data
  imgFile.write(buffer, bytesToRead);                // Transfer data to the imagefile
  jpglen -= bytesToRead;                             // Calculate remaining image bytes
 }
 imgFile.close();                                    // Close the image file
 time = millis() - time;                             // Stop the time counter
}

// Alarm stops the sleep process
void WakeUp(){
 noInterrupts();                                     // Disable interrupts to not disrupt the sensor readings
 WorkNow = true;                                     // The device is ready to take new sensors readings
}

// User push test button
void CheckNow(){
 noInterrupts();                                     // Disable interrupts to not disrupt the de-vice checking
 WorkNow = false;                                    // The device could be checked now
}

// Read present time and date
void ReadTime(){
 DateTime now = RTC.now();                           // Read date and time from the RTC
 Year = now.year(), DEC;                             // Read year information from the RTC
 Month = now.month(), DEC;                           // Read month information from the RTC
 Date = now.date(), DEC;                             // Read day information from the RTC
 Hour = now.hour(), DEC;                             // Read hour information from the RTC
 Minute = now.minute(), DEC;                         // Read minute information from the RTC
 Second = now.second(), DEC;                         // Read second information from the RTC
}

// Set the arduino to sleep mode
void sleepNow(){
 set_sleep_mode(SLEEP_MODE_PWR_DOWN);                // Configuration of the sleep mode
 sleep_enable();                                     // Enables the sleep bit in the microcontroler register
 sleep_mode();                                       // Put the microcontroller in sleep mode
 // DEVICE IS SLEEPING HERE
 sleep_disable();                                    // Disable sleep when the device is getting up
}

// Blink a led
void blinkLed(byte Pin, int numBlinks, int blinkRate) {
 for (int i=0; i < numBlinks; i++) {
  digitalWrite(Pin, HIGH);
  delay(blinkRate);
  digitalWrite(Pin, LOW);
  delay(blinkRate);
 }
}

// Shows the device status by a blinking led
void CheckDevice(){
 blinkLed(TestLedPin, 3, 500);
 delay(100);
 blinkLed(TestLedPin, e, 200);
}


// Error control
void error() {
 while(1){
```

```
  blinkLed(TestLedPin, e, 200);
  delay(1000);
 }                    }

// Utility function to print numbers leading 0
void printDigits(int digits){
 if(digits < 10){
  datafile.print('0');
 }
 datafile.print(digits);
}

*****************************************
```