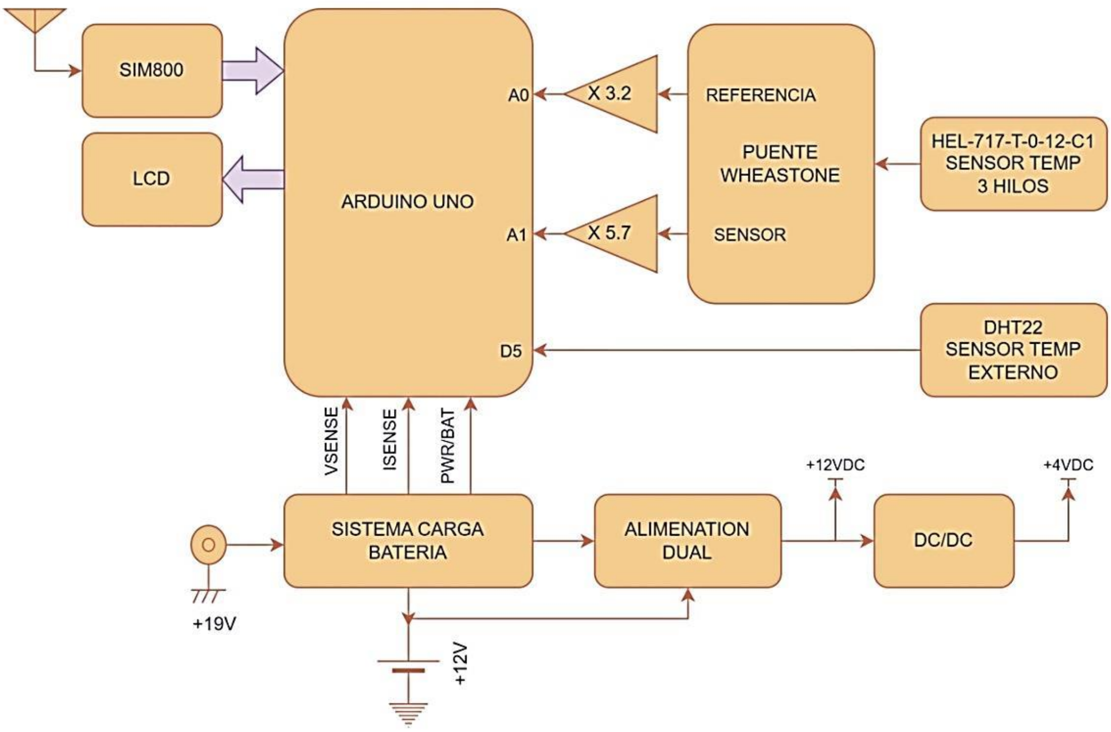
| Responsable<br>Dr. Esteban Valverde | IFIBIO “Houssay” UBA-CONICET<br>Arduino-based SMS alarm system for an ultrafreezer | Ticket #<br>20230323 |
|---|---|---|
| Start date: 23/03/2023 | | End date: 02/09/2023 |

The development of a temperature sensor for the UltraFreezer (UF) and alarm system is a necessity of the entire IFIBIO, but it was presented by the Biomembranes laboratory.

The idea of this development is to be able to measure the temperature of a UF and the external temperature in the environment where it is located and to inform, through SMS text messages, about changes that are considered important in order to take the measures that are considered necessary. These changes could be:

- Increase in UF temperature
- Increased outdoor temperature
- Cutting of the 220VAC line feeding the UF
- Reconnecting the 220VAC line after an outage
- Weekly/bi-weekly/monthly system status reminder to let you know everything is working normally

The general block diagram of the temperature and alarm system is as follows




The main blocks are as follows:

1. CPU with Arduino One: Performs all temperature reading algorithms, controls battery charge status, dialogues with the SIM800 cell phone board and informs the user through LCD display.

2. Internal temperature measurement circuit. It is based on a three-wire HEL-717 sensor for very low temperatures. It is connected to a branch of a Wheastone bridge, in order to differentially measure the voltage change of the sensor branch with respect to the reference branch (we are talking about the order of mV per degree Celsius)

3. Each branch has a different gain amplifier in order to maximize the voltage read by both ADC inputs of the Arduino. Internally, the voltage difference in both branches is calculated and the temperature is obtained.

4. DHT22 sensor to measure the external temperature, that of the room where the UF is located.

5. SIM800 cell phone system with Movistar chip. This device allows, through the "AT" protocol, to send SMS messages from the Arduino to a cell phone line that is defined and that will be the one that receives the messages and alerts sent by the device.

6. 12V sealed Acid Gel battery charging system by charging control by Arduino itself. In addition, the output is connected to a dual ignition system (from power or from battery), which ensures that the entire system works even during a power outage (for as long as the battery is charged).

## Internal Temperature Sensor

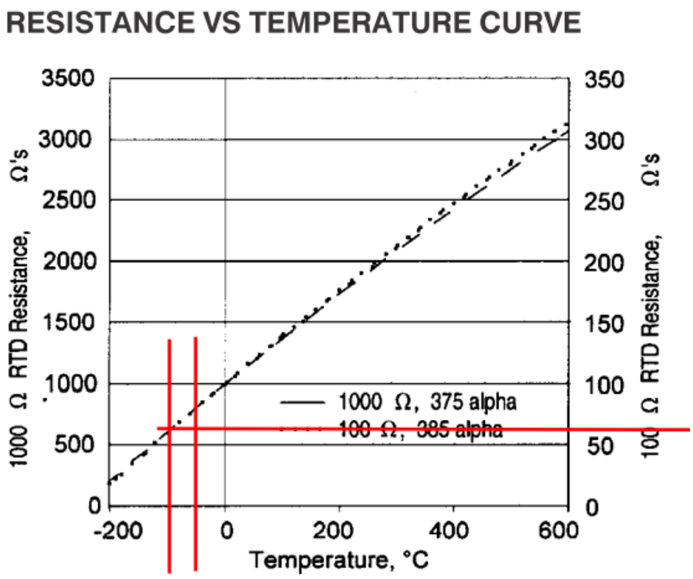
The sensor of choice for measuring temperatures in the range of -60 to -80°C is the HEL-717-T-0-12-C1. This sensor has the following characteristics: Fiberglass sensor, 3-wire and 100E at 0°C, and a temperature coefficient of $\alpha$ = 0.00385 Ω/Ω/°C and $\delta$ = 1.4999°C.

$$R_T = R_0(1+AT+BT^2-100CT^3+CT^4)$$

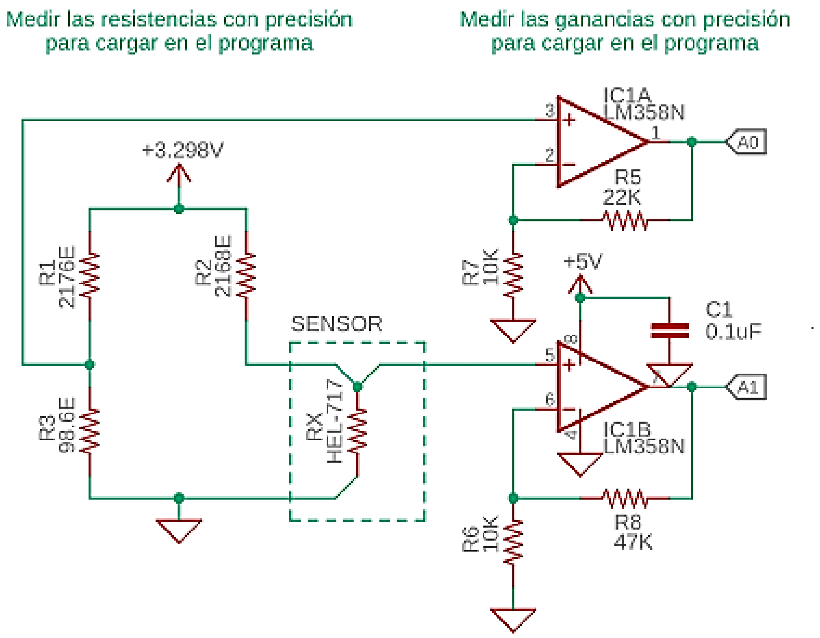
$$A = \alpha + \frac{\alpha\,\delta}{100}$$

Where only the term linear is taken, since I found that the others do not contribute at temperatures between -70 and -50°C.

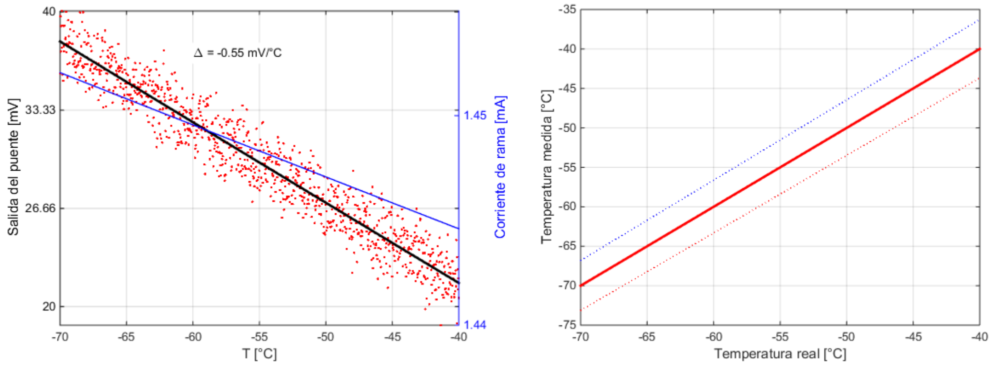
A linear behavior of the sensor resistance with temperature can be observed. Consequently, the bridge branches, reference, and gain of the amplifiers were calculated in order to maximize this value, since the temperature calculation is done internally in the Arduino Uno. As the values read from amp branches of the bridge are continuous voltage values, it is important that they are as high as possible (close to the reference voltage of the Arduino ADC converter) in order to have the highest possible sensitivity.




After several tests, the circuit is finally




The reference values of both branches are 2K2 and 100E, and a voltage of 3.3V offered by the Arduino on one of its pins. The HEL-717 sensor datasheet specifies that the maximum current that can flow through the sensor without generating self-heating is 1mA and an error of 1°C for 2mA. For the 100E nominal value of the sensor and the reference voltage of 3.3V, it means that the value of the sensor branch resistance must be greater than 3.2K. We use the standard value of 2.2K with the consideration of making an error of approximately 0.5°C. With these values, the bridge resolution is 0.55mV/°C. After being amplified, the resolution is around 3.11mV/°C.




It can be seen in the figure that the values were measured with a precision instrument in order to load this data onto the Arduino board and facilitate the calculations.

The figure above shows the error in the temperature measurement on the bridge when the resistors involved are at 1%. In case of using generic values of resistors, it is necessary to draw a correction line that will be loaded on each Arduino board if several of these devices are manufactured.

The following equations are obtained by doing the inverse mathematics of the temperature calculation, from what is obtained in each branch of the Wheastone bridge:

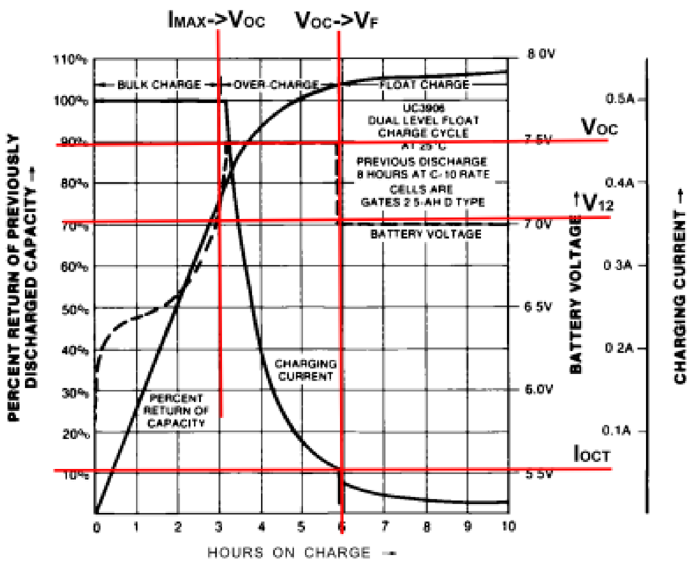
```
VA = A0 * (ADCvRef * 1000.0 / 1023.0); // in [mV]
VA = (VA – VIOA) / GA; // I consider the offset of the amplifier
VB = A1 * (ADCvRef * 1000.0 / 1023.0); // in [mV]
VB = (VB – VIOB) / GB; // I consider the offset of the amplifier
M = R3/(R1+R3) - (VA-VB)/VR; // M has the difference between the divisors of both branches

Rx = R2 * M / (1.0 - M); // from Ma I clear the calculated value of the sensor in ohms
Tx = (Rx/R0 - 1.0) / ALPHA; // I apply the sensor equation to calculate the temperature
```

## External Temperature Sensor

It uses the DHT22, which communicates via serial data with the Arduino, and which has a whole library of functions available, so it does not need further explanation. This sensor reaches up to -40°C, so it is not suitable for measuring inside the UF, but it is suitable for measuring the external temperature of the enclosure where the UF is located.

## Battery Charging System & Dual Power

The battery charging system circuit is a constant current source that is maintained depending on the current measured in the battery and the battery voltage. The system was developed for sealed lead/acid batteries. In this particular case, a 12V-1.5Ah battery is used.



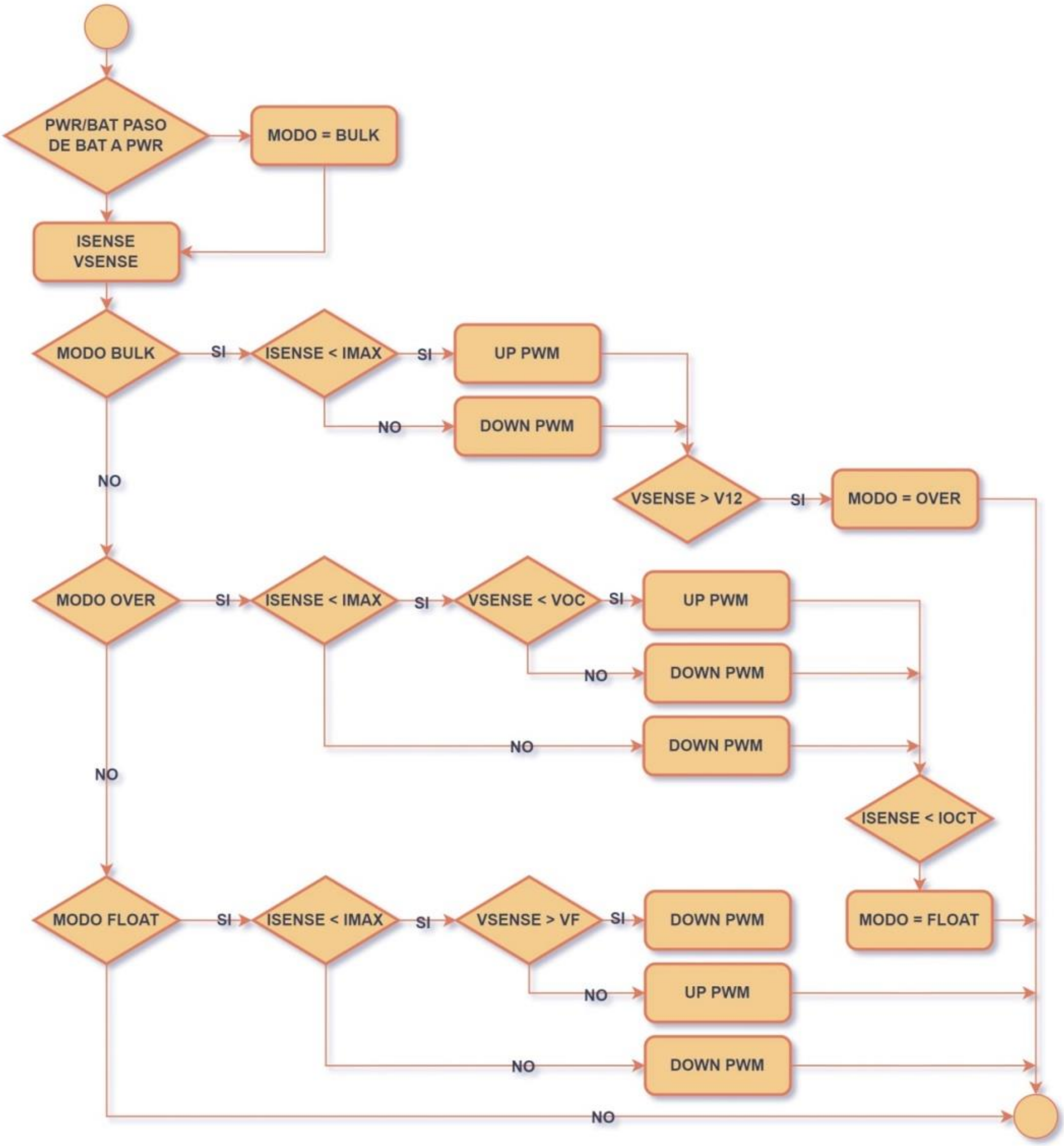

This type of battery has a charge cycle as shown in the figure, which is explained in detail in the UC3906 datasheets and Unitrode application note U-104.

The figure above shows the battery charge cycle. It basically has 3 stages namely: BULK charge (charging at maximum constant current), OVER charge (charging at maximum voltage) and FLOAT charge (keeping the charge at a lower voltage).

These three stages, in that order, ensure the maximization of the battery's charge and the maintenance of that charge once the charging process is finished.

To do this, the charging current (ISENSE) and battery voltage (VSENSE) must be constantly measured. The first should not exceed the designated maximum, e.g. for a 2Ah battery, using a charge of C/5 = 400mA would be a reasonable value of charging current. In the second case, it must be taken into account not to exceed the maximum value indicated in the data sheets, as it can produce a chemical reaction inside the battery and be destroyed. This value is called the cyclic load voltage and is above the nominal value. For example, in a 12V battery, like the one in the figure below, an OVER charge of 14.4 to 15V is recommended, a FLOAT load of 13.4 to 13.8V and a maximum charging current of IMAX = 2.1Amp, this is approx. C/3. V12 voltage is also defined, which is the voltage that is considered to have completed the BULK stage. Finally, the IOCT current is defined as the minimum current to consider that the battery is charged and switch to FLOAT mode. These V12 and IOCT values are either user-defined or taken from the application notes.

The main diagram of the battery charge flow is as follows




In the current design the values that were defined are: IMAX = 500mA, IOCT = 50mA, VOC = 13.56V, V12 = 95% VOC, VT = 65% VOC, VF = 13.4V, Vnom = 12V.

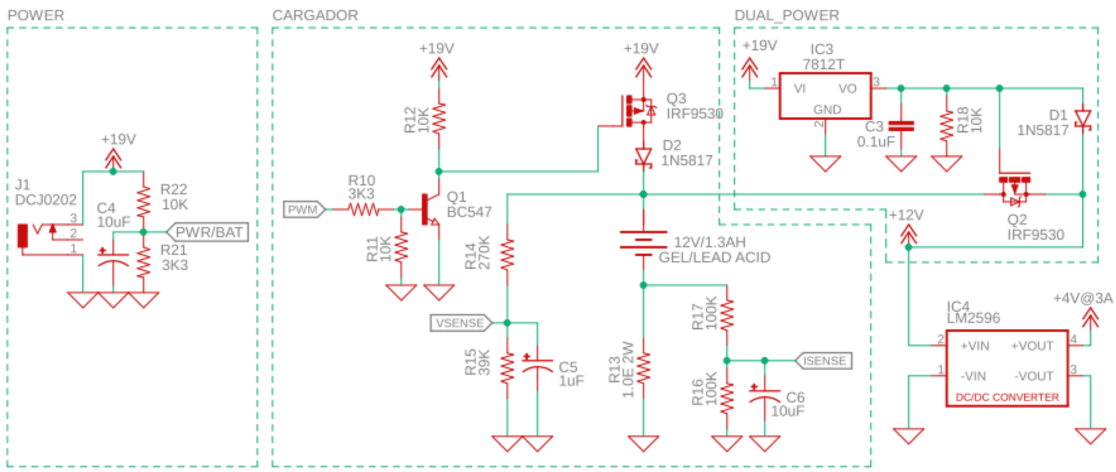
To maintain a constant load current that ensures either the charging current or the charging voltage, a specialized IC such as the UC3906N can be used. In our case we use the same Arduino Uno board for load control.

The central idea is to use a PWM output from the Arduino to control the charging current of a MOSFET. By measuring ISENSE and VSENSE using two ADC inputs of the Arduino, and depending on which stage of battery charge the battery is in, the width of the PWM pulse can be controlled to ensure either a constant current or a constant charging voltage.

It can be observed that depending on the charging mode it is in, and the ISENSE and VSENSE values, the width of the PWM pulse increases or decreases in order to control the charging current or voltage. The diagram also avoids exceeding the maximum charging current and the maximum charging voltage to protect the battery. The complete circuit of the charging system is as follows.

In addition to the external power input, the circuit also includes the dual power supply block. That is, regardless of whether there is external power (when there is a voltage of 220VAC in the mains line) or the battery is present, the device must always be energized. The Q2 MOSFET ensures this.

When there is external power, it is regulated to 12V and the Q2 gate opens the MOSFET, disconnecting the battery from the rest of the device. At this stage, the PWR/BAT pin is at HIGH and the main program is going to be charging the battery.




In the event of a power outage, for example, the voltage in the Q2 gate disappears, so the MOSFET is closed and the battery voltage is present at the output of the MOSFET, powering the device. At this point, the PWR/BAT pin is set to LOW and the main program interrupts the loading process. When the power is restored, the change from LOW to HIGH is detected on this pin and the charging mode is forced to BULK in order to begin the charging process.

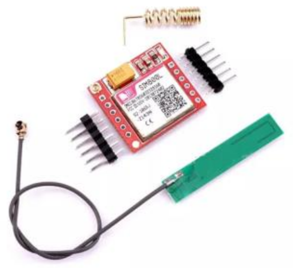
In the event that the battery is completely discharged, or a cell is exhausted or short-circuited, there is a first charging step, in which the charging current is set at very low values until a VT voltage is exceeded (also proposed in the application note). Once this voltage has been passed (first charge at low current due to battery protection) it can be switched to BULK mode. This is done to avoid charging a fully discharged battery at full current, and it is better that it has some charge in the first place.

As stated above, the charging system reads two variables, VSENSE and ISENSE to control the charging current or voltage in the battery.

VSENSE is read by a voltage divider. At a voltage VOC = 13.56V, the voltage VSENSE = 39K / (39K+270K) x 13.56 = 1.71V. ISENSE is measured by a voltage divider over the voltage on a 1E resistor in series with the battery. For IMAX = 0.5A, the voltage at the IMAX resistor is 0.5V, and the divider, which has two 100K resistors, divides by two, with the voltage measured at ISENSE = 0.25V. For a 5V ADC reference, the bit resolution is bit = REF/2^10 = 4.88mV. For ISENSE, 250mV/4.88mV = 51 ADC counts. You could take the VSENSE and ISENSE splitter to higher output values and close to VREF (5V) in order to maximize the reading in both cases, or use an amplifier on each variable to be read.

## SMS Mobile Phone System

The device features a module called the SIM800. Basically, this device is a mini cell phone that, among other things, allows you to send SMS messages to another phone. It uses AT commands via RS232 to communicate with the microcontroller, and only requires a loaded chip.



It should be said that it uses 2G networks for messaging and that these networks are still active in Argentina, but from what I was reading, as of 2024 they would be out of service, so this module could no longer continue sending messages.

In the SIM800 library there is the protocol which is very simple and has a series of AT commands that are sent to the SIM800 module. A command is then sent with the cell phone number to which the SMS message is sent, and finally the message itself. Since RS232 is by AT commands, everything is in text mode.

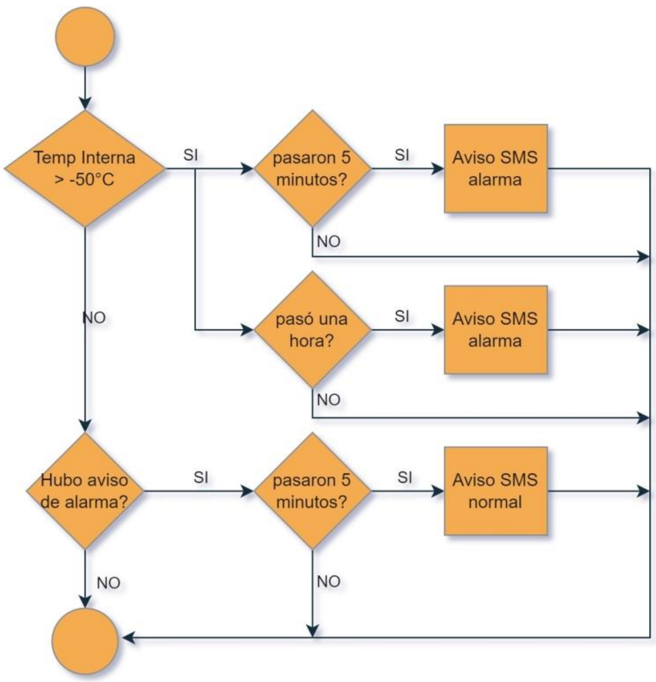
The SIM800 module is powered at 4V, so it is necessary to use an LM2596 DC/DC to regulate its output from 12V to 4V. It can also have consumption peaks of approx 3Amp during transmission bursts, but it's something I haven't noticed even looking at the oscilloscope.

## Alarm and warning protocol

The next section explains the protocol for alarms and warnings, i.e. what the device should do and when it should report any anomalies in temperature measurements. Two temperatures are measured

- inside the freezer (around -65 to -70°C)
- outdoors (higher than 20°C, depending on the season)

The following alarm protocol is proposed




- send alarm if the indoor temperature is higher than -50°C.
- send alarm if the outside temperature is higher than +40°C.
- Send alarm if there is a power outage.
- Send notice when power is restored.
- Send a "system in normal operation" notice once a fortnight or a month (to know that the device is working).

The alarm can be

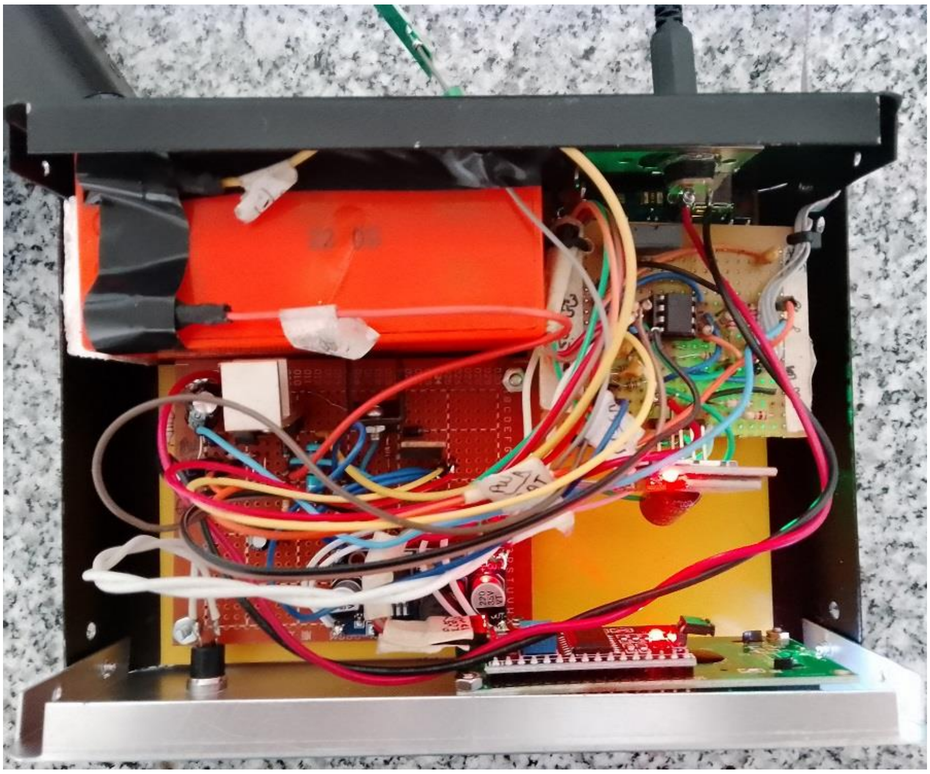
- give a warning by SMS (to several cell phone numbers) 5 minutes after the alarm occurs (to verify that it is a true alarm), and another after one hour, when any of the alarms from 1 to 3 of the previous list occur.
- give a single SMS notification (to multiple cell phone numbers) in condition 4 or 5 of the above list.

The flowchart above applies for both internal and external temperature reading. The diagram below shows the main loop of the program.

## Complete Alarm System Circuit

The complete circuit of the alarm system, which integrates all functionalities, is attached with this document.

## Finished and working model



| **Responsable** Dr. Esteban Valverde | **IFIBIO "Houssay" UBA-CONICET** Arduino-based SMS alarm system for an ultrafreezer | **Ticket #** 20230323 |
|---|---|---|
| **Start date: 23/03/2023** | | **End date: 02/09/2023** |