Atentto Manual for Controlling and Data-logging

Department of Applied Mathematics
School of Aeronautical and Space Engineering
Technical University of Madrid (UPM)

Chapter 1

Hardware specifications

1.1 Introduction

The purpose of this chapter is to provide a complete user guide to implement AtenTTo as a electronic control for monitoring and to controlling energy efficiency buildings. AtenTTo is a hardware and software platform based on the Atmel SAM3X8E ARM Cortex-M3 CPU and it is Arduino DUE compatible. AtenTTo comprises two PCB: *(i)* AtenTTo core which is the brain of the system and *(ii)* AtenTTo multipurpose in which the brain is plugged allowing to connect specific sensors and actuators. To facilitate AtenTTo integration, AtenTTo multipurpose is designed to be placed in a standard DIN rail box as it is shown in Figure 1.1.

1.2 Platform description: hardware and software

Once AtenTTo core is plugged in AtenTTo multipurpose, the platform allows to manage 10 relays, 10 mini–jack connectors with sensors and actuators. The system by means a SD card is able to store the collected data and is also able to upload data to the cloud by means of an RJ45 Internet connector. This platform runs an embedded software written in C++ and programmed through a micro USB connector. Besides, AtenTTo core is Arduino DUE compatible. Hence, a large number of software codes, written for Arduino DUE, could be uploaded to this hardware platform without any modification. AtenTTo is a software embedded in a hardware platform able to perform the following tasks:

1. Control an installation by means of sensor readings and acting on actuator relays to switch ON/OFF a hydraulic pump or to command a signal digitally (ON/OFF).
2. Monitor sensors actuators with a prefixed sampling rate.
3. Serve a web page which allows to see on line the automatic controlling and to command manually the installation.
4. Upload data to the cloud.

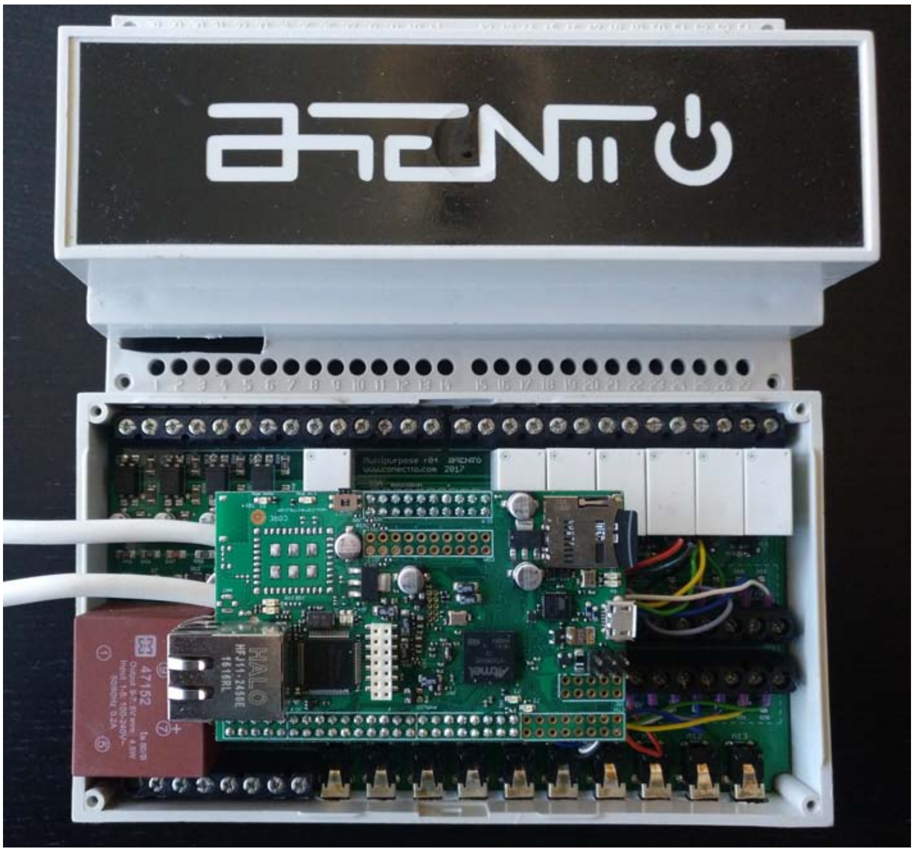


Figure 1.1: AtenTTo Core and AtenTTo Multipurpose in a DIN rail box

1.3 Hardware specifications

AtenTTo Multipurpose is a printed circuit board of $153x85mm$ integrated in a DIN rail CA905 module and powered by 220V-AC voltage supply (Figures 1.2 and 1.3). It incorporates a transformer-rectifier (AC-DC) that delivers +5V to power AtenTTo Core (which is plugged in Multipurpose) and some components inside itself. It also has voltage regulator (+3.3V) for other components and external needs. This board has at the bottom part two connectors for the power source and the analog channels for sensors. At the upper part there are relays with two wire connector for relay and optocouplers with other two. The hardware specifications of this platform are:

- 10 Analog channels (ADC) with a extent range between 0 and 3.3V. These channels are attached to a female mini-jack connector (Tip-Ring-Sleeve, TRS) (Figure 1.4).
- 10 Relays to close AC or DC voltages (Max. current of 8A) (Figure 1.5).
- 4 Optocouplers to sense external signal voltage (Figure 1.6).
- 10 One-Wire channels with up to 10 One-Wire temperature probes per channel (Figure 1.7).
- 1 Analog channel (ADC) to read a variable signal between 0 and 3.3V.
- 1 Analog channel (ADC) to read a variable signal between 0 and 0.1V.
- 4 Digital signals. The digital general purpose input/output signals (Digital GPIO) can sense external signals or implement an specific protocol.
- 4 PWM Digital signals for pump controlling (Figure 1.7).
- 1 Buzzer to provide acoustic feedback tones (Figure 1.7).
- Audio connector with 2 DAC signals, 1 ADC, +5V and +3.3V.

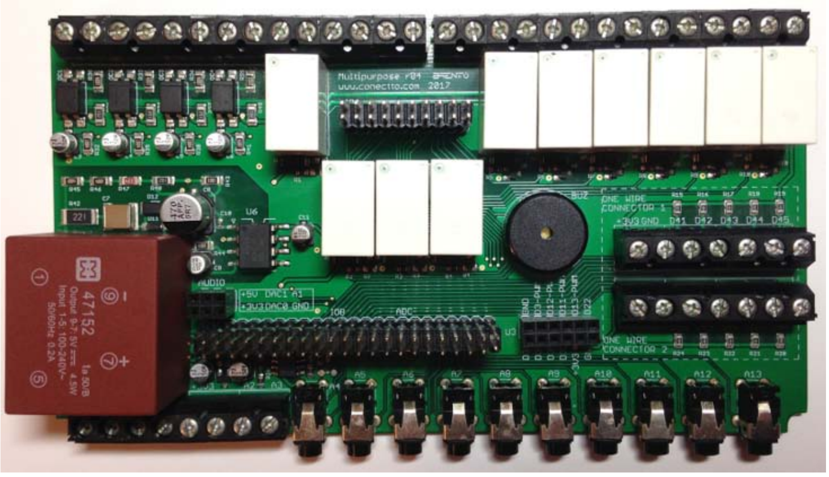


Figure 1.2: Top view of AtenTTo Multipurpose.

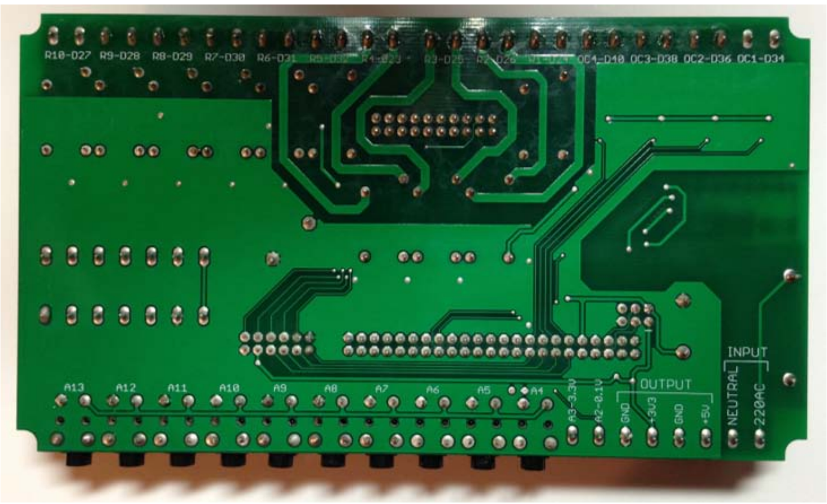


Figure 1.3: Bottom view of AtenTTo Multipurpose.

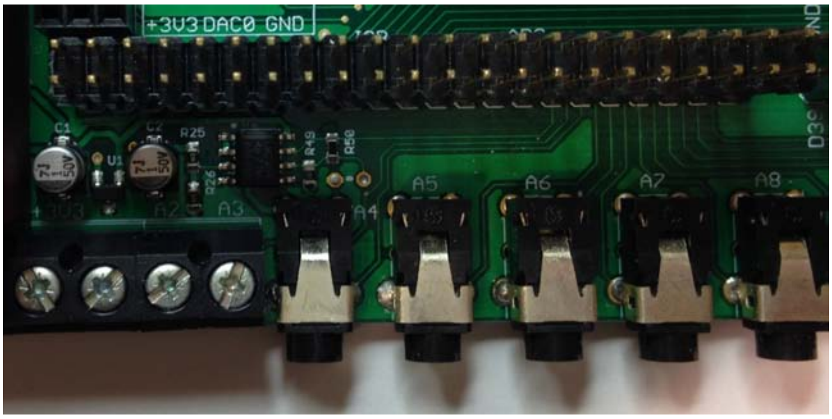


Figure 1.4: Detail of the first five mini-jack connectors where analogical signals are routed; they are specially used for amperometric clamps.

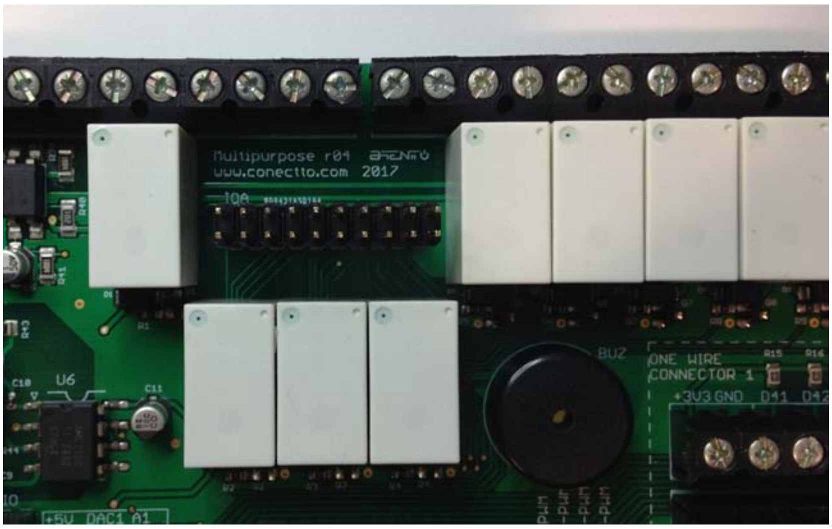


Figure 1.5: Foreground of some relays surrounding IOA connector and screw connectors in the top part of the board.

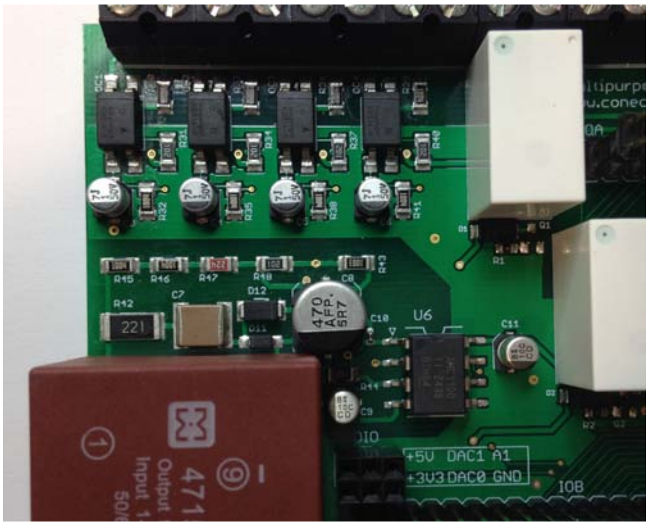


Figure 1.6: Four optocouplers in the top part and all the circuit necessary for measuring the mains voltage (*AMC1100*) in the bottom part.

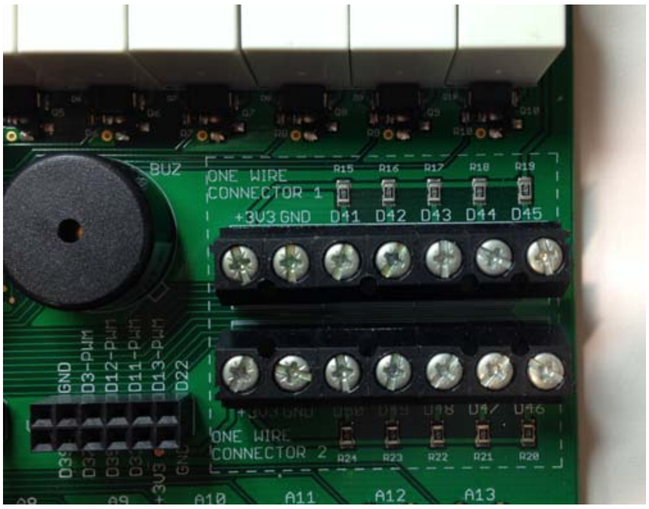


Figure 1.7: Screw connectors for One Wire probes (temperature measurement), Buzzer for acoustic tones and PWM signals connector. One Wire connectors can also be used as general pull-up input/output digital signals.

Chapter 2

Atentto user guide

2.1 How to start running AtenTTo

AtenTTo is prepared to run with a default installation. To start running AtenTTo, only two connections are needed:

1. Power AtenTTo by means of two screw terminals: phase of 230 AC volts and neutral.

2. Plug AtenTTo to Internet by means of a RJ45 wire.

AtenTTo comes with a preinstalled SD card which contains default configuration files. Both the embedded software and the configuration files can be modified at any time.

2.2 Configuring SD AtenTTo card

The configuration of the AtenTTo platform is done by the following JSON files which are saved in the SD AtenTTo card:

Configuration files

atentto.ini, sensors.ini, actors.ini, macros.ini, thermos.ini, timers.ini, wired.ini, alerts.ini.

These files are JSON files and they allow to configure a complete energy management system for a building. The content of these files is the following:

1. `atentto.ini`. This file comprises the name, IP direction, MAC address, email subscriber and controlling and data-logging parameters of the equipment.

2. `sensors.ini`. This file comprises the analog, digital, and radio frequency sensors. For every sensor, its name, address and magnitude are given.

3. `actors.ini`. This file comprises the digital and radio frequency sensors. For every actuator, its name, address and device are given.
4. `macros.ini`. This file has predefined macros. A macro is a set of different actuators.
5. `thermos.ini`. This file comprises simple and differential thermostats linked to a specific macro. Every thermostat is linked to a macro and defines its set point temperature and its mode of functioning: heat, cool or off.
6. `timers.ini`. This file comprises different timers linked to a specific macro. For every timer, the starting hour and finishing hour, days of the week and mode of functioning are given.
7. `wired.ini`. This file comprises logical link between radio frequency sensors and actuators.
8. `alerts.ini`. This file comprises different conditions that are continuously checked to assure the flawlessly functioning of the system.

The user could modify this files by the following three procedures:

1. Extract the SD card and modify manually the content of the above files.
2. Configuration WEB page. Once Atentto is running, a basic configuration WEB page is served in the default IP direction: **192.168.1.200:8082/configuration.html** From this WEB page, files can be edited and modified manually.
3. AtenTTo WEB page. If AtenTTo is running without problems, AtenTTo publishes a complete web page that allows to modify its configuration in a very easy and reliable way. The step by step configuration process is described in the WEB page chapter.

2.3 Detailed setup process

Once AtenTTo is powered a complete setup process is carried out. In order to know if AtenTTo is working flawlessly, the complete initialization process (step by step) is described in the following list.

1. First of all, AtenTTo configures a few parameters needed for sampling (he does the “sampling setup”), like its automatic gain. Then it initializes its serial port. After doing that, AtenTTo writes the message “Reset AtenTTo” through the serial port.
2. The next thing AtenTTo does is searching for an SD card and also checks if it has any errors. If there is not an SD card or it is corrupt, AtenTTo will emit the “Alert error” alarm.
3. Then, AtenTTo reads its configuration file, from where it takes the name of the current facility, its IP direction, its MAC and the email to where it sends all its alarms, and, with that IP, AtenTTo configures its ethernet connection.

4. After that, AtenTTo tries to read an income from its serial prompt, and if the user has introduced anything in it, AtenTTo enters in configuration mode.
5. Then AtenTTo starts the main part of its setup, it reads its last plt files and sets the hour and date. Right after that is when it initializes the buzzer and emits its start sound.
6. It also reads now the other configuration files and configures sensors, actors and macros. This is the part of the configuration where most errors occur usually.
7. It starts with the configuration of the sensors. First of all it tries to configure the radio, if it is not any radio connected to AtenTTo it will emit its "radio error". Then AtenTTo counts all the sensors that are already written in the "sensors.ini" file and configures the motes. After that it configures the analogical, the digital and the OneWire sensors, in that order. Finally, it initializes all the sensors saved.
8. When it has finished initializing all the sensors saved, it reads its memory and shows (via serial prompt) the total memory and the memory occupied by the sensors.
9. The actuators setup works almost identically: AtenTTo reads the "actors.ini" file, counts the written actuators and configures them.
10. Finally, AtenTTo configures its macros, reading the "macros.ini" file and using the previously configured sensors and actuators.
11. After that, AtenTTo sends an email saying that it has reset.
12. Finally, AtenTTo configures and serves its web page.

2.4 What is the use of AtenTTo beeps ?

AtenTTo uses some discrete beeps to give us feedback of its processes: setup configuration and main loop tasks. These beeps can be classified into two groups : normal beeps and malfunction beeps. If AtenTTo detects some configuration problem, hardware problem or logical preconceived alarms, it beeps to alert of the problem.

Normal beeps:

- Start sound: This sound consists in one long beep followed by another short beep. This will sound every time that AtenTTo starts, after it has finished its setup process.
- Datalogging sound: This sound is a very short and low beep that sounds every time that AtenTTo registers the data from its sensors.
- Post sound: This sound is a low and relatively long beep that sounds every time that AtenTTo posts data to the cloud.

Malfunctions or error sounds:

- SD card malfunction. If the SD card is not present, file are corrupted or files are missing, AtenTTo beeps insistently until a new SD card is inserted. This is the most annoying alarm, four loud and short beeps will sound continuously, and the two central LEDs will blink repeatedly. If this happens, it means that there are missing files in the SD card or that the SD card is not introduced. In order to solve this all the files must be copied again into the SD card from Dropbox\Software\AtenTTo\SD card, copying always the latest version (the last folder) into the SD. Then the SD card must be introduced into the AtenTTo and it will reset automatically and restart the setup normally.

- Configuration alert. This alarm is not intended to be especially annoying, one short and low beep will sound every seven seconds and one of the central LEDs will blink at the same time, indicating that an error message has appeared in the configuration page. This happens when there is any semantic error in the .ini files, so in order to solve it you must go to the configuration page, read the error message (which will indicate what the problem is) and change whatever is wrong inside the configuration files. After doing that AtenTTo shall be reset.

- Ethernet malfunction: If the Ethernet wire is disconnected or the connection between the router and AtenTTo does not work, AtenTTo will emit two beeps periodically until the problem is solved. This alarm will consist on two short beeps and two short blinks of the central LEDs that will be continuous. Check all the Ethernet connection between AtenTTo and the router to solve this problem.

2.5 Troubleshooting

The figure 2.1 shows a brief description the Atentto initializing process. every division standing for every step where an error can appear. In order to simplify the identification of possible errors, alerts based on repetitive beeps or blinking led signals are implemented. The correct system initialization is verified by following a detailed check list. Besides this check list, Atentto gives extra information by writing messages through its serial port. In the next section it is explained how to open the serial port to identify this extra information.

2.6 Connection serial protocol.

In this section it will be described the complete process to connect AtenTTo to a serial port from the PC.

1. Connect AtenTTo to the PC via micro USB wire.

2. Windows operating system will make a sound informing that AtenTTo has been recognized.

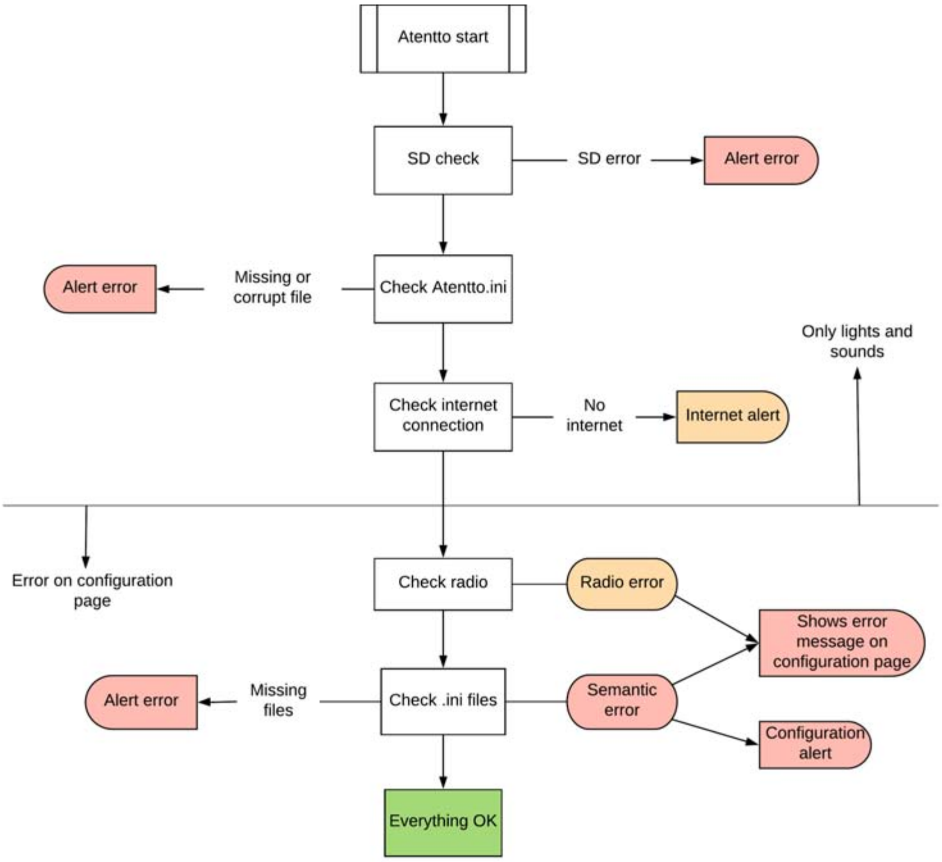


Figure 2.1: Scheme of the possible errors and their responses

3. In order to look for the port in which AtenTTo is connected, go to Control panel/System and security/System/Device administrator/COM and LPT ports.
4. Identify the port AtenTTo is connected.
5. Open any serial monitor (putty, Arduino IDE).
6. Select the specific port (Tools/Port) in which AtenTTo is connected.
7. Open the serial.
8. Set transmission speed to 115200 bauds.
9. AtenTTo information must appear through the serial console.

2.7 Checklist of AtenTTo connection

The following check list is intended to facilitate the correct initializing process. By following step by step this check list, the user can identify possible errors or configuration problems. While the first group of check points assure the right configuration associated to initialization files saved in the SD card, the seconf group of check points assure the right configuration of internet communications.

- □ Hardware connections.
  - □ Plug AtenTTo to the socket.
  - □ Ensure the SD card is inserted, if not, when you plug the AtenTTo, four loud and short beeps will sound continuously.
  - □ A blue and red leds must switch on, which means AtenTTo and Ethernet connection have power.
  - □ Two green leds close to the Ethernet connector must switch on, which means that the Ethernet connection is working.
  - □ If the RJ45 wire is not connected or if the router is not working, the green leds will not turn on, two short beeps you will be heard periodically and and orange led near the RJ45 connector will blink continuously.
  - □ The two led located in the middle of the PCB must switch on, which means that AtenTTo is reading the value of the sensors and writing data on the SD card.
  - □ One of the two orange leds near the SD card (TX led) must blink. This means AtenTTo is sending data via USB.
  - □ The two orange leds located near the RJ45 connector must blink, which means AtenTTo is receiving and sending data to the web page.
  - □ Two beeps (the first one longer than the second one) must be heard, which means that AtenTTo is executing its setup process.
  - □ One of the leds located in the middle of the PCB (led 72) must switch off and start blinking periodically. This means AtenTTo has already read the value of every sensor in the setup process and starts reading them periodically.

  - □ One low and relatively long beep will must sound, which means AtenTTo has posted data to the cloud and the web page is running.
  - □ One low and short beep must sound, which means AtenTTo has register data from its sensors. At the same time the orange led that is still turned on in the middle on the PCB (led 73) must turned off.
  - □ From now on, AtenTTo must produce low and short beeps periodically, which means AtenTTo has register data from its sensors. At the time this sound is emitted, one of the leds located in the middle of the PCB (led 73) must blink.
  - □ From time to time AtenTTo must produce one low and relatively long beep will must sound, which means AtenTTo has posted data to the cloud.

- □ Web page.
  - □ Go in Attento web page introducing in the browser http://54.93.223.235.
  - □ Introduce user and password.
  - □ Select the facility used.
  - □ Icons of sensors and devices connected to Attento must appear in the main page.

2.8 Watchdog

The watchdog is a timer used to detect and recover electronic systems from crashes by running a reset. A timer is an interruption based on the clock frequency, this is, a counter which is decreased each certain number of clock pulses and when it reaches zero an interruption is executed. In the case of watchdogs, this interruption will reset the system. During a normal operation, the watchdog timer is reset periodically in order to prevent it form elapsing. If due to a hardware fault or program error the system crashes, the watchdog timer elapse provoking a reset of the system.
In AtenTTo, if the system keeps blocked more than 16 seconds the Watchdog will reset it.

Chapter 3

System configuration

As it was mentioned, the configuration of the AtenTTo platform is done by the following JSON files which are saved in the SD AtenTTo card:

Configuration files

atentto.ini, sensors.ini, actors.ini, macros.ini, thermos.ini, timers.ini, wired.ini, alerts.ini.

This configuration can be done by modifying with a text editor these files or user friendly WEB page.

3.1 AtenTTo configuration

The first configuration file that AtenTTo reads from its SD card is called: `atentto.ini`. The following example is a typical JSON `atentto.ini` file:

```
{
    "Facility": "DMAIA",
    "IP": "192.168.1.227",
    "MAC": "90-A2-DA-91-A3-07",
    "user_email": "juanantonio.hernandez@upm.es",
    "user_IP": "192.168.1.133",
    "DT_sampling": "60000000",
    "DT_refresh": "6000000",
    "DT_alerts": "60000",
    "DT_control":  "60000",
    "DT_post":     "360000",
    "DT_ethernet": "360000"
}
```

There are eleven different fields containing the following information:

1. **Facility**: This field indicates the name of the AtenTTo or the name of the facility. This is the name that will appear in the web page and in the emails that are sent by AtenTTo.

2. **IP**: This is the IP direction that this AtenTTo will use to serve its configuration web page.

3. **MAC**: This address is used by motes to communicate with AtenTTo. This communication is done by radio frequency in the band of 868 MHz.

4. **user_email**: This email direction is used by AtenTTo to inform with messages or reports of its state of its functioning.

5. **user_IP**: IP direction of the user of the facility. AtenTTo uses this direction to acknowledge the presence of the user close to the facility.

6. **DT_sampling**: Sampling time interval that sensors are read and saved in the SD card. It is measured in milliseconds.

7. **DT_refresh**: Time interval in microseconds to refresh the web page.

8. **DT_alerts**: Time interval to check different alerts described in `alerts.ini`. This interval is measured in milliseconds.

9. **DT_control**: Time interval to control AtentTTo actuators. Every DT_control milliseconds, AtenTTo reads its sensors and depending on the the different thermostats and timers act by commanding the system actuators. It is also expressed in milliseconds.

10. **DT_post**: Time interval in milliseconds that AtenTTo uses to communicate with an external server to inform of its actual state.

11. **DT_ethernet**: Time interval in milliseconds that AtenTTo checks ethernet connection, internet connection, MQTT server and HTTP server.

3.2 Sensors

A sensor is an element of the system capable of measuring a physical quantity by means of a physical effect and transforming it into an electrical signal. The control system in which the linked sensors are located, analyzes the type of sensor and converts the electrical signal into a physical quantity.

The physical magnitudes that measure these sensors and that are considered in this work are: temperature, relative humidity, solar radiation, voltage and electric current and electrical consumption. From the practical point of view, the sensors are implemented by transducers that are devices capable of transforming or measuring an energy associated with a physical phenomenon to an electrical signal. In this way, to define a sensor, the phenomenon of electromagnetic induction can be used, by means of a transformer, to measure an electric current. On the other hand, there are manufacturers that encapsulate different physical phenomena in devices or sensors. Through the data-sheet supplied by the manufacturer, the electrical signal is converted to the physical quantity. Similarly, there are sensors in the market that measure analog quantities such as temperature and through an internal process convert it into a digital signal with a specific protocol. Any of the sensors listed above are considered in this work.

It is important to note that the same electrical signal does not have to be directly associated with the magnitude of a sensor. That is, depending on the magnitude associated with a sensor, it can take values from different transducers to give a magnitude derived from two transducers. For example, if we want to calculate the electrical power in the consumption of a line we must measure two transducers separately: the voltage of the network and the intensity. These two waves are usually out of phase and the actual consumption value depends on the phase difference of these two waves. Thus, the intensity and electrical current are measured by two different transducers and defined an electrical power sensor through its mathematical expression in relation to the intensity and electric current. While transducers are physical elements, sensors are abstractions that are defined through the measurements of certain transducers. It is for this reason that it is important to define the characteristics of a sensor in the following way:

1. **Label**. The label is the name of the sensor that allows us to identify the measured quantity. It is important to provide semantic content to the labels to write control of the system in a more simple way.

2. **Magnitude**. The physical magnitude of the sensor considered. These magnitudes can be analog on infinitude of values or digital with only two possible values. Generally, digital sensors are associated with switches or alarms. The analog sensors considered in this work can measure: temperature, relative humidity, solar radiation, intensity and electric current, electrical consumption.

3. **Type**. The type refers to the associated physical transducer and the type of linkage with the system. Thus, there are the following types:

   (a) **RF22**. Wireless sensors that are linked automatically by radio frequency to the concentrator or control system.

   (b) **OneWire**. Sensors that send their information through a cable with OneWire protocol. The OneWire protocol allows different OneWire sensors to be hooked up on the same cable, which allows for the taking of different data with a single cable. These sensors are linked automatically to the concentrator or control system.

   (c) **Analog**. Sensors that supply an analog electrical signal. These transducers are connected to the hub through analog read ports. The concentrator, depending on its magnitude, will process the electrical signal to determine the value of the measurement.

   (d) **Digital**. Sensors that supply a digital electrical signal. These transducers are connected to the concentrator through digital read ports. In this case, the concentrator will transform a high voltage level of the digital signal to 1 and a low level to 0.

4. **Address**. The direction of a sensor refers to its hardware location. As we have seen previously, the sensors can be wired to without cable. For those wired analog sensors the possible addresses are: A0 - A13 which are the 14 analogue reading channels that the concentrator allows. However, if the magnitude of a sensor is derived from two wired transducers in channels A0 and A7, the syntax for the direction of this sensor would be: A0: A7. Both the electrical power and the electrical consumption are magnitudes derived from the intensity and the electrical voltage. These types of sensors support both an address associated with a channel (e.g. A7) and an address associated with two channels (e.g. A0: A7). In the first case it is assumed that the electrical voltage is in phase with the electrical current and is worth 230 v. For wired digital sensors, the possible addresses are: D30 - D32 which are three digital read channels that the hub allows. For the OneWire temperature sensors, digital ports D43 - D50 are reserved. In this case, the direction of the sensor is determined automatically once the sensors have been wired to the specified ports. For wireless sensors, the addresses are determined automatically as long as the wireless sensors have been linked by software to the hub.

3.3 Actuators

An actuator is an element of the system giving an on or off signal for an element of the system such as a circulation pump or a light bulb. The control system determines the actuator value ON (1) or OFF (0) and converts the digital signal into a high or low voltage signal. This high or low voltage signal attacks a coil of a relay and closes a circuit for a pump to circulate or a light to switch ON/OFF.

The characteristics that define the actuators are:

1. **Label**. The label is the name of the actuator that allows us to identify the device on which it acts. As with the sensors, It is important to provide labels with semantic content to write the control through logical conditions expressed between the values of the sensors that determine the values of the actuators.

2. **Device**. This characteristic defines the physical device on which the electrical signal acts. The devices can be: a hydraulic pump, a light, a heater, the heat / cold mode of a heat pump. Any of these actuators associated with these devices has the same behavior. When the level of electrical voltage determined by the control system is high, a coil that closes a circuit to operate this device is attacked. The presence of this feature allows the system to associate different icons with semantic content that allow programming the system in non-computer language in a simple and graphical way.

3. **Type**. The type refers to the physical signal associated with the actuator. There are the following types:

   (a) **RF22**. Wireless actuators that are linked automatically by radio frequency to the concentrator or control system.

   (b) **X10**. Actuators that are operated through the electrical network with X10 protocol. The X10 protocol allows the existing electrical network to be used to hook different actuators distributed in the system.

   (c) **Digital**. Actuators that operate an electric signal all / nothing. There are 8 relays whose coil is linked to digital write ports. The concentrator depending on the control logic will attack the coils of these relays by closing or opening a circuit. That is, these actuators are relays that can close on the high side any type of direct or alternating voltage and with loads of up to 16 amps.

4. **Address**. The direction of an actuator refers to its hardware location. Actuators can be wired without cable. For those wired digital actuators the possible addresses are: D22 - D29 which are the 8 relays that allows to control the concentrator. For wireless actuators, the addresses are determined automatically as long as the wireless actuators have been linked by software to the hub.

3.4 Macros

The macros are characterized by a name and represent the association of one or more actuators. The definition of the macro allows to act with a single command to turn on or off a set of different actuators. Thus, if the order of a thermostat must start two pumps at the same time, a macro can be created which, by its name, associates the two corresponding actuators of the pumps. Subsequently, a thermostat is created with the name of the macro and when the thermostat is activated, the two pumps start operating at the same time.

The operating modes of the macros can be:

1. **AUTO**. The macro is governed by the thermostat to which it is linked.
2. **MANUAL**. The macro is governed by the user from the WEB application.
3. **OFF**. All the actuators associated with the macro are off.
4. **DISABLE**. The macro is deactivated and does not allow acting either manually or automatically on the associated actuators.

3.5 Thermostats

A thermostat is a component of a control system that opens or closes an electrical circuit based on a temperature. Thermostats can be associated with one or two temperature sensors and can be in different operating modes: cold mode, heat mode or inactive mode. If a thermostat in heat mode has a single temperature sensor associated with it, the thermostat switches ON when the temperature is lower than a certain set-point minus ΔT (hysteresis band) and turns OFF when the sensor temperature exceeds the set-point plus ΔT. If the thermostat is in cold mode, the control logic is the inverse. In the event that there is a second temperature sensor linked to the thermostat, the thermostat is considered differential. That is, the comparison temperature is the difference of the temperatures of both sensors.

The names of the thermostats can be either previously defined acting on macros or scenes labels.

3.6 Timers

Timers allow to program in a certain time any system such as the air conditioning of the building. Timers are defined by the start time, the end or stop time, and by the days of the week to which it is applied. Timers are associated to macros.

Start time or stop time can be set to a specific hour or can have the following descriptors:

1. `"HH:MM"`. Hour and minute to start or to end the timer.
2. `"Sunset"`. The timer starts or ends at sunset. This time is calculated depending on the day of the year and sun trajectory.
3. `"Sunrise"`. The timer starts or ends at sunrise.
4. `"Simul"`. The timer stats or ends randomly. This value is set every time AtenTTo is reset.

If the stop time is smaller than the start time (e.g. timer from 21:00 to 02:00), then the timer commands ON from start time to 24:00 hours and from 00:00 hours to stop time.

In addition, of the above parameters, timers can hve the following operating modes:

1. `"On"`. If actual time is in between starting time and ending time, then the macros is ON, else the macro os OFF.
2. `"Off"`. If actual time is in between starting time and ending time, then the macros is OFF.
3. `"Up"`. Blinds UP command is sent during one minute beginning with the starting time.
4. `"Middle_Up"`. Blinds UP command is sent during 5 seconds beginning with the starting time.
5. `"Down"`. Blinds DOWN command is sent during one minute beginning with the starting time.
6. `"Middle_Down"`. Blinds DOWN command is sent during 5 seconds beginning with the starting time.
7. `"Disable"`. Timer is disabled.

The following example sets a timer to switch ON the the macro named `"Cooling"` from `07:00` to `20:00` hours through Monday to Friday.

```
[
      [
      "Cooling",
      "07:00",
      "20:00",
      "Mon-Fri",
      "On"
      ]
]
```

3.7 Softwired

AtenTTo allows to link digital sensors with different actuators by means of a JSON file named: `wired.ini`. This file has the following columns which allow to connect actuators with digital sensors by a software link.

1. **Digital sensor name**. This column is the name of the digital sensors that is link virtually to different actuators.
2. **Actuator name(1-4)**. These columns content the name of the different actuators which are linked to the digital sensor.

The digital sensor is generally a push button. If the push button is pressed, AtenTTo sends an order to every actuator which has been linked to this digital sensor. Up to four actuator can be commanded at a time.

The following example is a typical JSON `wired.ini` file:

```
[
   [
   "S5",
   "Light1",
   "Light2",
   "",
   ""
   ]
]
```

In this case, the push button **S5** of a Themtto mote is link with two actuators **Light1** and **Light2** of two ATToPlug motes. Once the push button is pressed, the state of the actuators is changed.

3.8 Alerts

AtenTTo alerts can be configured by means of a JSON file named: `alerts.ini`. This file has the following columns which allow to preset different alerts:

1. **NAME**. This column is the name of the alert.

   It can be the name of a sensor or an actuator or `"ANY"`. If it is a sensors or actuator, the alert checks for the variable the rest of the columns and take the requiered actions. If it is `"ANY"`, then the column 5 (reference group) should be a magnitude or a device. If the reference group is a magnitude, then the alert checks every sensor with this magnitude. If the reference group is a device, the alert checks every actuator which is such a device.

2. **MIN**. This column represents the minimum value of range alert.

   This field can only be used in alerts associated with sensors that measure analogical values, such as temperature or flow meter sensors. If the sensor value is out of the range between the minimum and the maximum values, the alert will send a warning.

3. **MAX**. This column represents the maximum value of range alert.

   This field can only be used in alerts associated with sensors that measure analogical values, such as temperature or flow meter sensors. If the sensor value is out of the range between the minimum and the maximum values, the alert will send a warning.

4. **VALUE**. This column is linked with digital sensors, such as the door or button ones, and with all kind of actuators.

   Value can only take seven different values: `"ON"`, `"OFF"`, `"UP"`, `"DOWN"`, `"OPEN"`, `"CLOSE"` and `"NOTNORMAL"`; but depending on the device with which the alert is linked it can only take one, two or three values. A warning will be sent if the value written in this field agree with the one that the sensor or actuator has. We must be aware of the state in which we want to program the alert, for instance, for sensors, when a door is closed its value is `"CLOSE"` and when it is open, `"OPEN"`, or when a button is pushed its state is `"ON"` and when not, `"OFF"`. In case of actuators the functioning is easier, since when they are working their value is `"ON"`, and when not, it is `"OFF"`. The only exception for actuators are the blind ones, which takes `"UP"` value when the blind is going up, `"DOWN"` when it is going down and `"OFF"` when it is stopped. If `"NOTNORMAL"` is chosen, AtenTTo compares the value of the device linked with the alert with the values of all the devices that have alerts with the same reference group by calculating its typical deviation. For analogical sensors, this value is the current value of the sensor, and for actuators it is the time that the device has been working.`"NOTNORMAL"` cannot be used with alerts connected with digital sensors. Finally, if the device value is out of the normal distribution, AtenTTo will send a warning. It is important to mention that if this gap is filled, the two previous gaps (maximum and minimum value) will leave blank.

5. **REFERENCE GROUP**. This column is the label of a reference group.

   The user must fill with a label in order to link different alerts. Devices with alerts with the same reference group will be compared when `"NOTNORMAL"` is selected as a value or at the time of checking or sending a report. If the alert name is `"ANY"`, the reference group of this alert must be a sensor magnitude or an actuator device. In other cases, the reference group name does not play any role.

6. **CONDITION GROUP**. This column is the label a condition group.

   Every alert with the same condition label will be checked. If every alert with this condition is ON, a warning will be sent. This column must always be filled, even if an alert does not share a condition with anothers; in this case, the condition will be attached to a single alert.

7. **TIME BEGIN**.

   This column is the start time in which the alert should be checked (e.g. 10:20). If start time column is empty, the alert is checked 24 hours a day.

8. **TIME END**.

   This column is the end time in which the alert should be checked (e.g. 20:20). If end time column is empty, the alert is checked 24 hours a day.

9. **DAYS**. This colunmn shows the days in which an alert is checked.

   It can take the following different values: `"Daily"`, `"Mon-Fri"`, `"Sat-Sun"`, `"Mon"`, `"Tue"`, `"Wed"`, `"Thu"`, `"Fri"`, `"Sat"`, `"Sun"`. According to these values and depending on the dat of the week, the alert is checked.

10. **MODE**. This column represents the alert mode.

    It can be: `"AUTO"`, `"ARMED"`, `"MANUAL"` or `"ANY"`. If AtenTTo mode coincides with the alert mode, then the alert is checked. If AtenTTo is working in `"OFF"` mode, alerts will be disabled. If the alert mode is `"ANY"`, the alert is checked if the AtenTTo mode is `"AUTO"`, `"ARMED"`,`"MANUAL"`.

11. **ACTIONS**. This column is the specific action that AtenTTo will perform if the alert is ON.

    It can be filled with the words `"sound"`, `"email"`, `"all"` or `"NONE"` in order to AtenTTo produces a certain sound, sends an email, makes both actions or makes nothing, respectively. If sound is selected and an alert is ON, AtenTTo will beep. It is significant to mention that exists different levels of importance for each alert depending on the sensor or actuator they are linked with. If more than one alert is ON, the most important alert will beep. In case of email warnings, the level of importance does not affect, since if more than one alert is enabled, it just will send an email in which will appear the name of those alerts, and therefore, the ones of the sensors or actuators they are connected with. Emails are sent for the first time in the moment that the alert is ON, but if it continuous enabled, they will be sent every day at midnight with the same content.

12. **REPORT**. This last column is related to reports.

It might be said that reports are different from all the other fields mentioned before, as they accomplish a different commitment. This field can be filled with the words `"NONE"`, `"DAILY"`, `"WEEKLY"`, `"MONTHLY"` or `"YEARLY"`, depending on the frequency the user wants to receive them. Reports are sent as an email in which it is written the characteristic values of the sensors or actuators which are linked with the alerts empowered to send them. In case of analogical sensor alerts, the report will send the maximum and the minimum values that the sensor has reach, the average of its values and the relative difference between average values of other sensor alerts of the same reference group and its average value. In case of actuator and digital sensor alerts, the report will send the time the actuator has been turned on and the relative difference between this time and the time that other actuator alerts of the same reference group have been turned on.

3.8.1 Algorithm to check alerts

Every time interval `DT_alerts` measured in milliseconds, AtenTTo checks with the following algorithm is any warning is reached:

1. For each reference group, set its average value and its typical deviation.
2. For each alert:
   (a) Check if the current mode of AtenTTo is the alert mode. If not, exit.
   (b) Check the alert day. If the day of the week coincides with the days the alert should be checked, then set the enforceable flag to true. If not, set this flag to false.
   (c) If enforceable flag is true, check if the actual time is in the time interval the alert should be check. If so, set enforceable flag to true. If not, set this flag to false.
3. For each enforceable alert and if the ACTIONS are different from `NONE`:
   (a) If `MAX` and `MIN` values are distinct of null, check if the actual value of the alert is out of range.
   (b) If `VALUE` is `NOTNORMAL`, compare the device current value with the average value of its reference group.
   (c) If `VALUE` has another content, compare the device current value with `VALUE`.
4. Carry out the corresponding `ACTION`: sound, email or both and taking into account the `CONDITION GROUP`.

3.8.2 Reports based on alerts

The algorithm to generate reports associated to alerts is the following:

1. For each alert, if the current time agrees with the time chosen in the report field, write the report values:

    (a) Maximum, minimum and average values taken and its relative difference with the alerts of its reference group, if the alert is linked with an analogical sensor.

    (b) Time turned on and relative difference with the rest of the alerts of its reference group, if the alert is linked with a digital sensor or an actuator.

3.8.3 Example: Temperature out of range

The following example sets an alert to check if `T_int` is below 18 Celsius degrees or above 29 Celsius degrees. It belongs to `Indoor` temperature group. In this case, the reference group does not play any role. The condition group is `C1`. If every temperature with condition `C1` is out of its range, then a message is sent to alert of the malfunction. Hence, `C1` condition group works as an `"AND"` condition for complex alerts based on two or more conditions. Besides, this alert will be checked from 8:00 hours to 20:00 hours from Monday to Friday when the AtenTTo mode is `"AUTO"`, `"OFF"`, `"ARMED"` or `"MANUAL"`.

If an alert based on `T_int` is reached, a small beep will sound alerting of temperature value. Finally, an every day report is sent informing of the resulting alerts during the day.

```
[
        [
        "T_int",
        "18.0",
        "29.0",
        "",
        "Indoor",
        "C1",
        "08:00",
        "20:00",
        "Mon-Fri",
        "AUTO",
        "sound",
        "DAILY"
        ]
]
```

3.8.4 Example: ANY temperature out of range

The following example sets an alert to check if any temperature is below 0 Celsius degrees or above 40 Celsius degrees. The reference group in this case is the magnitude `Temperature` which means that this condition is checked for every temperature sensor. The condition group is this alert is `C2`. If all other alerts with condition `C2` are out of range, then a message is sent to alert of the malfunction. Besides, this alert will be checked daily when the AtenTTo mode is `"AUTO"`.

If the conditions of this alert are reached, an email is sent to the AtenTTo user. Finally, no reports are sent.

```
[
        [
        "ANY",
        "0.0",
        "40.0",
        "",
        "Temperature",
        "C2",
        "00:00",
        "24:00",
        "Daily",
        "AUTO",
        "email",
        "NONE"
        ]
]
```

3.8.5 Example: ANY blind UP at night

The following example sets an alert to check if there is any blind up at night. The reference group in this case is the magnitude `Temperature` which means that this condition is checked for every temperature sensor. The condition group is this alert is `C1`. If all other alerts with condition `C1` are out of range, then a message is sent to alert of the malfunction. Besides, this alert is checked from 00:00 am to 08:00 am when the AtenTTo mode is `"AUTO"`.

If the conditions of this alert are reached, AtenTTo begins to beep periodically and an email is sent to the AtenTTo user. Finally, no reports are sent.

```
[
        [
        "ANY",
        "",
        "",
        "UP",
        "Blind",
        "C4",
        "00:00",
        "08:00",
        "Daily",
        "AUTO",
        "all",
        "NONE"
        ]
]
```

3.8.6 Example: NOTNORMAL behavior of any temperature

The following example sets an alert to check if any temperature is far from the normal behavior of other selected temperatures. The reference group in this case is the magnitude `Temperature` which means that this condition is checked for every temperature sensor. The condition group is this alert is `C3`. If all other alerts with condition `C3` are out of range, then a message is sent to alert of the malfunction. Besides, this alert will be checked daily when the AtenTTo mode is `"AUTO"`.

If the conditions of this alert are reached, an email is sent to the AtenTTo user. Finally, no reports are sent.

```
[
        [
        "ANY",
        "",
        "",
        "NOTNORMAL",
        "Temperature",
        "C3",
        "00:00",
        "24:00",
        "Daily",
        "AUTO",
        "email",
        "NONE"
        ]
]
```

3.8.7 Example: Door OPEN at night

The following example sets an alert to check if the door is open at night. The reference group in this case is the magnitude `Temperature` which means that this condition is checked for every temperature sensor. The condition group is this alert is `C1`. If all other alerts with condition `C1` are out of range, then a message is sent to alert of the malfunction. Besides, this alert is checked from 00:00 am to 08:00 am when the AtenTTo mode is `"AUTO"`.

If the conditions of this alert are reached, AtenTTo begins to beep periodically and an email is sent to the AtenTTo user. Finally, no reports are sent.

```
[
        [
        "Door",
        "",
        "",
        "OPEN",
        "Indoor",
        "C4",
        "00:00",
        "08:00",
        "Daily",
        "AUTO",
        "all",
        "NONE"
        ]
]
```

3.8.8 Example: Report on power consumption

The following example sets an alert to check if the door is open at night. The reference group in this case is the magnitude `Temperature`which means that this condition is checked for every temperature sensor. The condition group is this alert is `C1`. If all other alerts with condition `C1` are out of range, then a message is sent to alert of the malfunction. Besides, this alert is checked from 00:00 am to 08:00 am when the AtenTTo mode is `"AUTO"`.

If the conditions of this alert are reached, AtenTTo begins to beep periodically and an email is sent to the AtenTTo user. Finally, no reports are sent.

```
[
        [
        "Door",
        "",
        "",
        "OPEN",
        "Indoor",
        "C4",
        "00:00",
        "08:00",
        "Daily",
        "AUTO",
        "all",
        "NONE"
        ]
]
```

Chapter 4

Controlling

4.1 Main loop of the embedded software

Once sensors, actuators, macros, thermostats and timers are configured properly, the system is controlled by means of the different tasks of the main loop. Sensors readings fed thermostats and timers to switch ON and OFF circulation pumps, lights, valves or other actuators.

The control loop performs sequentially the following tasks:

1. Reading all sensors: temperature, solar radiation and power consumption.
2. Controlling actuators by means of thermostats and timers.
3. Writing actuator values.
4. Saving data on an SD memory card, Depending on the selected sampling rate, it stores the value of sensors and actuators .
5. Posting data to the cloud.

4.2 Data monitoring

To evaluate the efficiency of the system and its behavior, sensors and actuators must be monitored over time. The monitoring process can be done in real time or in deferred time. Generally, when it comes to monitoring the air conditioning installations to measure their efficiency or carry out their maintenance, the measurement of data in deferred time is more than enough.

This section aims to explain the different monitoring strategies as well as the services derived from the monitoring data. With respect to the monitoring of the system, we define the following monitoring strategies:

1. Local. The data from sensors and actuators are stored on an SD memory card in the microcontroller.
2. Remote. The local data of sensors and actuators are sent daily to a remote server that, later, uploads them to the cloud (e.g. Dropbox). Depending

on the needs of the system, data can be sent instantly to a remote server that processes them in real time to provide a service.

If sensor data and actuator data is uploaded to the cloud, a predictive maintenance service can be defined allowing the following reports:

1. Reports on system efficiency.
2. Alerts and alarms.
3. Reports on the actions and possible improvements based on system performances.

4.3 Codeless programming

To carry out the energy strategy requirements defined above, the energy management software allows to configure the system with code-less programming. This characteristic allows the user to program the system without the need of knowing any programming language. By creating timers and thermostats, complex conditions can be built. Since actuators are linked to macros or scenes, the state of an actuator is obtained by determining the state those macros. Each scene or macro can have at least one timer or thermostat. Hence, the state of a macro M_k is determined by AND conditions of the state of thermostats or timers associated to M_k. Namely, once the state of timers and thermostats are known, the state of a macro is

$$M_k = T_1 \wedge T_2 \ldots T_m,$$

where $\wedge$ stands for an AND condition and T_m are timers or thermostats associated to a specific macro M_k. Once the state of every macro is known, the state of an actuator is

$$A_i = M_1 \vee M_2 \ldots M_k,$$

where $\vee$ stands for an OR condition and M_k are macros associated to a specific actiator A_i. This is explained graphically in Figure 4.1. From the practical point of view, this code–less programming is accomplished with two different steps: *(i)* define different scenes and *(ii)* define thermostats and timers for those scenes. Once a scene is created, different thermosts and timers can be configured for this scene. These thermostats and timers will decide the scene or macro state by AND conditions. The process can be repated for every macro or scene. Finally, the state of the actuator is decided by OR conditions of the different macros associated to this actuator.

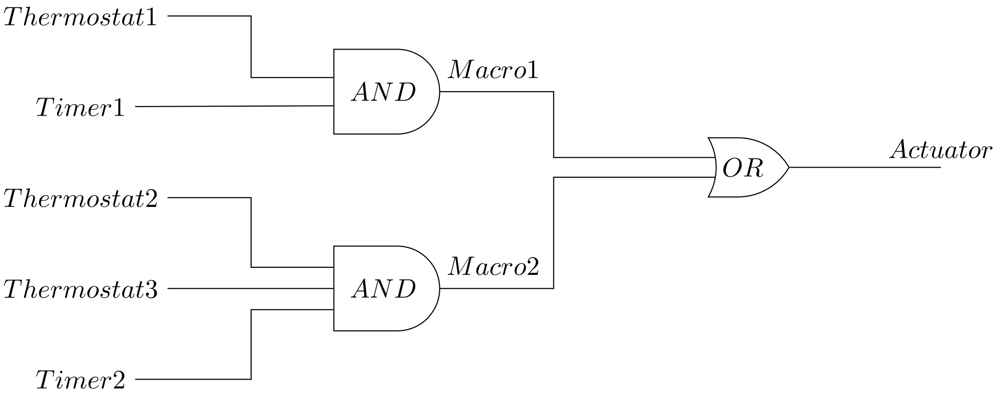


Figure 4.1: Actuator state as a function of different timers and thermostats. State of any macro is determined by AND conditions of all timers and thermostats linked to the macro. Actuator state is determined by OR conditions of all macro linked to the actuator.

Chapter 5

Web page user manual

5.1 AtenTTo main page

Atentto web page has many different functionalities, and most of them are accessible through the main page, which uses the following route: SERVER:PORT/atentto.html (for example: 192.168.1.222:8082/atentto.html). The port indicates the facility, to know in which facility you are, at the top of the web page is written the name of the facility and the actual date and time. While surfing on Atentto web page it is important to know that by clicking on the Atentto logo, on the upper left side, it will redirect us to the main page.

As it is shown in this image, the main page acts as a hub from where almost all the other functionalities can be reached, but first of all the main page functions are going to be described

5.2 Main page functionalities

The main page functionalities are the following:

- The icons in the main page represent all the sensors and actors that are connected to Atentto, and all the macros which are programmed in it. Each icon indicates the state of each sensor/actor/macro: in the case of the sensors it is displayed the numeric value that is being measured at the actual time; in the case of actuators it is displayed whether if they are switched on or off; and in the case of macros it shows if their actors are activated or not.

- By clicking on the icons you can change the state of actuators and macros, switching actuators on and off, or disabling and enabling macros.

- Graphs: In the main page, the sensors icons show the magnitude (if it is a temperature the icon would be a thermometer), and also the value that it has at that moment. If you want more information make double click on the icon. One click won't do anything.

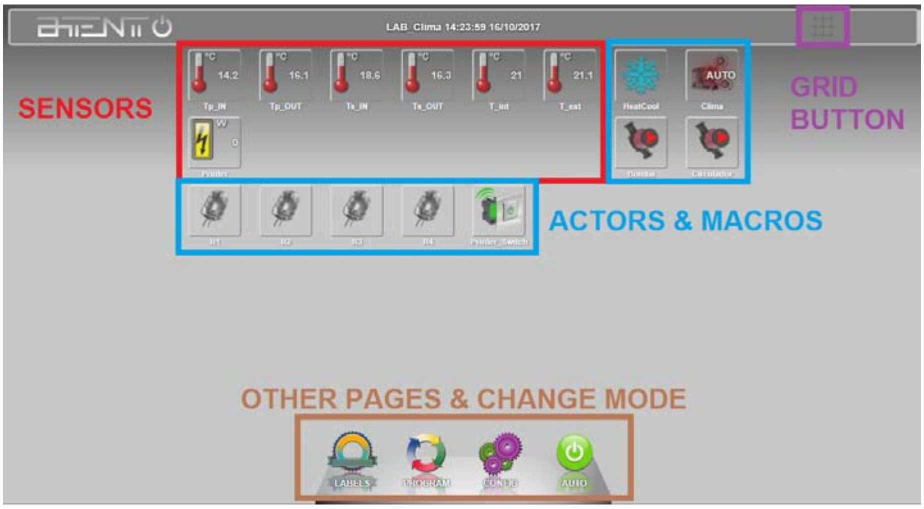
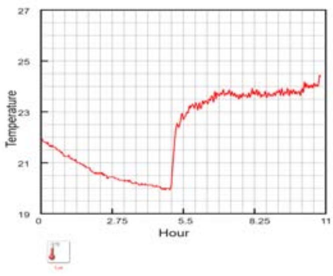
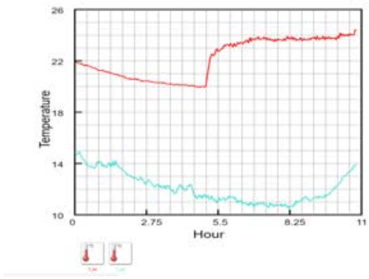


Figure 5.1: Atentto main page



(a) Plot of the first variable



(b) Plot of the first and second variables

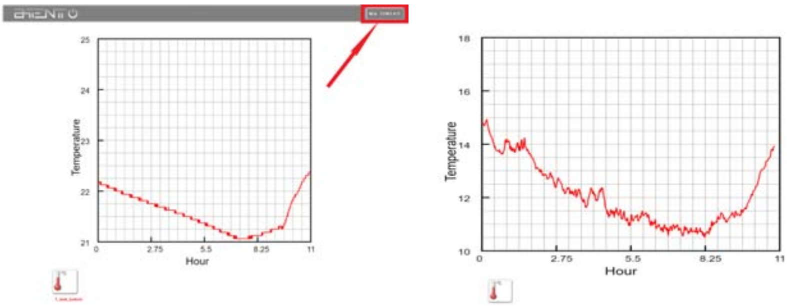
If you make double click, the web page will make a plot with the variable selected. This simple plot is made with the values, of the variable, that were recorded through the day.

In order to show two variables on the same plot, the user will need to return to the main page, and then, click on one variable and then double click on the other one. If, on the contrary, you just want to see one variable, before seeing the other one, on the upper right corner there is a “New Template” button. This clears all the variables, and then just double click on the second variable that you Want to see.

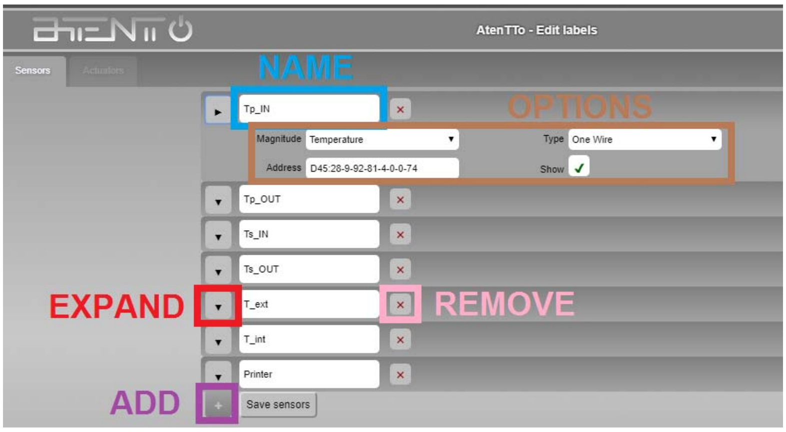
Example:

- To personalize the position of the icons, there is an option on the right side of the header (⌗). This option enables all the possible positions where an icon can be. To position the icon where you want it to be, the user will just have to click and drag it with the mouse.

- The last button on the menu, which looks like an ON/OFF button, changes the way that the scenes(macros) on the page will work. It can be MANUAL (the programs are disabled), AUTO (the programs are enabled) and



(a) New template button (b) New plot with other variable



AtenTTo - Edit labels

Tp_IN

Magnitude: Temperature; Type: One Wire

Address: D45:28-9-92-81-4-0-0-74; Show: ✓

Tp_OUT

Ts_IN

Ts_OUT

T_ext

T_int

Printer

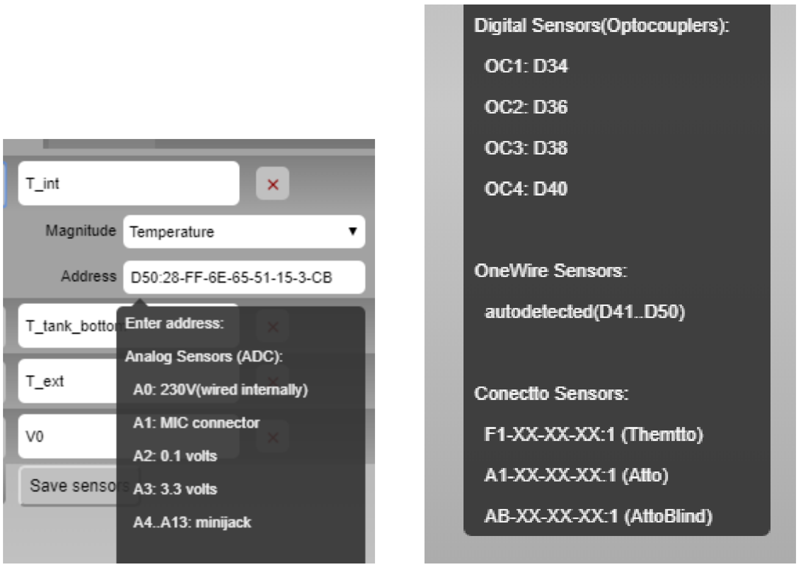
OFF (all devices disabled).

- Finally, on the main page, at the bottom, there is a menu that allows us to navigate trough other pages, Labels, Program and Config, and now these are going to be explained separately.

5.3 Modifying or adding sensors or actuators

Clicking on the first button of the menu, named labels, leads to the page where you can **add or modify sensors and actors**, choosing between different options, such as the sensor's magnitude.It is also possible to select if the users wants them to be shown in the main page, or not. In labels we have two tabs, one for sensors and other for actuators. If we want to modify or add a sensor or an actuator we have to click on its correspondent tab, where we will see all the sensors/actuators.
To modify one of these you must click on the arrow button next to it and a menu with different options will unfold. If what you want to do is removing a sensor,there is a cross on the right side of the sensor, clicking on it will remove the sensor. To add a sensor or an actuator there is an add sign on the left side of the "save" button. It will appear an other row, we must open the options so we can put all the information about the new sensor we are adding.

Here there are going to be explained the possible options that can be chosen in the labels page, starting with sensors:



T_int

Magnitude: Temperature

Address: D50:28-FF-6E-65-51-15-3-CB

T_tank_botto

T_ext

V0

Save senso

Enter address:

Analog Sensors (ADC):

A0: 230V(wired internally)

A1: MIC connector

A2: 0.1 volts

A3: 3.3 volts

A4..A13: minijack

Digital Sensors(Optocouplers):

OC1: D34

OC2: D36

OC3: D38

OC4: D40

OneWire Sensors:

autodetected(D41..D50)

Conectto Sensors:

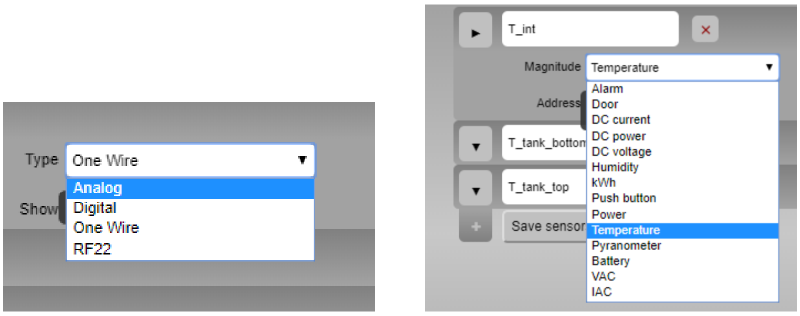
F1-XX-XX-XX:1 (Themtto)

A1-XX-XX-XX:1 (Atto)

AB-XX-XX-XX:1 (AttoBlind)

(a) (b)

Figure 5.4: Addresses for the different sensors



Type: One Wire

T_int

Magnitude: Temperature

(a) Different types (b) Different magnitudes

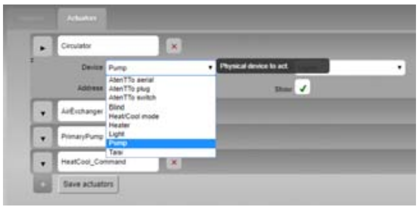
Figure 5.5: Addresses for the different sensors

Sensors options

- Magnitude: It represents the physical variable measured by the sensor. Changing this parameter will change the icon of the sensor which is displayed on the main page.
- Address: It shows the physical port where the sensor is connected to Atentto.
- Type: It indicates what kind of sensor is actually connected.

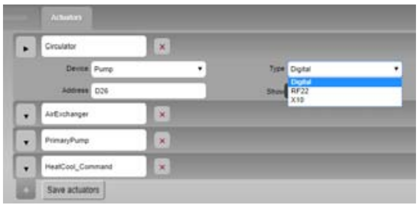
Actuators work more or less the same as sensors do. It is possible to modify the device in order to change its icon, the type of device and the port where it is connected, as it is shown in the next images.

In the "address" field it makes no difference typing R1 or D24, referring



Device: Pump

(a) Different devices

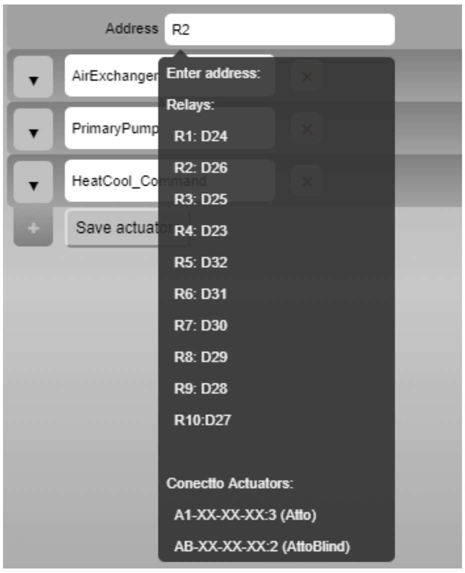


Device: Pump

Address: D26

Type: Digital

(b) Different types


Address: R2

Enter address:

Relays:

R1: D24

R2: D26

R3: D25

R4: D23

R5: D32

R6: D31

R7: D30

R8: D29

R9: D28

R10:D27

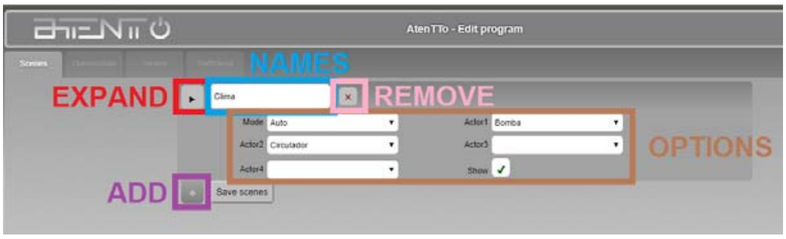
Conectto Actuators:

A1-XX-XX-XX:3 (Atto)

AB-XX-XX-XX:2 (AttoBlind)

Figure 5.7: Possible addresses

directly to the relay in use or specifying the digital port which Atentto multipurpose uses.



Clima

Mode: Auto

Actor1: Bomba

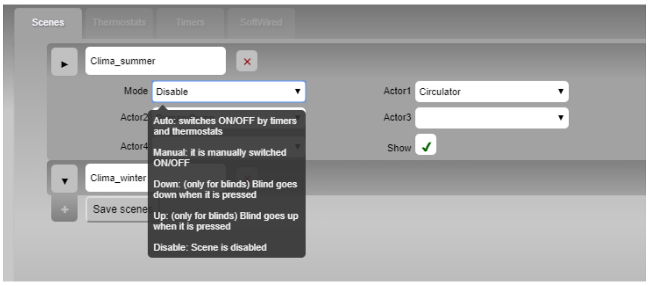
Actor2: Circulador

Actor3:

Actor4:

Show ✓

Figure 5.8: Program page



Clima_summer

Mode: Disable

Actor1: Circulator

Actor2

Actor3:

Actor4

Show ✓

Clima_winter

Auto: switches ON/OFF by timers and thermostats

Manual: it is manually switched ON/OFF

Down: (only for blinds) Blind goes down when it is pressed

Up: (only for blinds) Blind goes up when it is pressed

Disable: Scene is disabled

Figure 5.9: Options in Scenes tab

5.4 Modifying program configuration

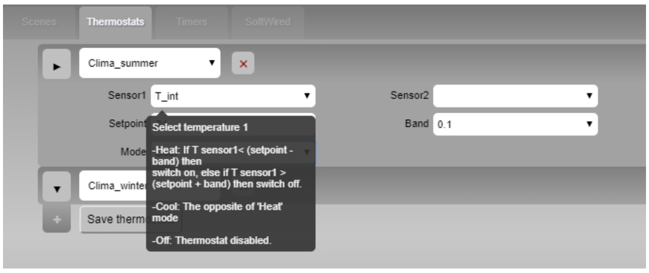
Clicking on the program icon at the bottom of the main page leads to the page where you can **add or modify macros, thermostats, timers, softwired and alerts**. This is very useful when you want to have an automatized installation. It is also possible to select if the macros are shown in the main page, or not (the others don't need an icon).

The program page works almost identically to the labels page, we have here buttons for adding, modifying or erasing a scene, a thermostat or a timer.

Scenes are used to link actuators to thermostats and timers. Thermostats control if actuators are on or off depending on the value of the selected sensors, and timers control whether or not scenes are active.

In the scenes tab it is possible to change the mode in which a concrete scene works and the actuators that are linked to it

In the Thermostats tab it is possible to configure which sensors' values determine the actuators state. This can be done choosing from all the sensors installed in Atentto and their setpoint. We can select one or two sensors, the thermostat will work differently depending on if we selected one sensor or two.



Sensor1 T_int

Sensor2

Band 0.1

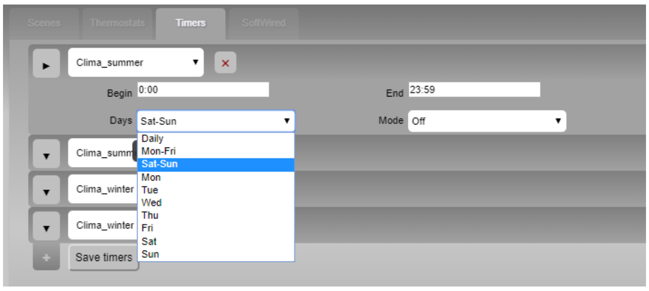
Select temperature 1

-Heat: If T sensor1< (setpoint - band) then
switch on, else if T sensor1 > (setpoint + band) then switch off.

-Cool: The opposite of 'Heat' mode

-Off: Thermostat disabled.

Figure 5.10: Working with one sensor



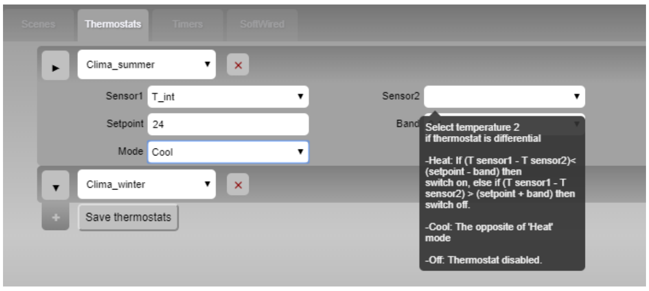
Begin 0:00

End 23:59

Days Sat-Sun

Mode Off

Figure 5.12: Timers options



Sensor1 T_int

Setpoint 24

Mode Cool

Sensor2

Band

Select temperature 2
if thermostat is differential

-Heat: If (T sensor1 - T sensor2)< (setpoint - band) then
switch on, else if (T sensor1 - T sensor2) > (setpoint + band) then switch off.

-Cool: The opposite of 'Heat' mode

-Off: Thermostat disabled.

Figure 5.11: Working with two sensors

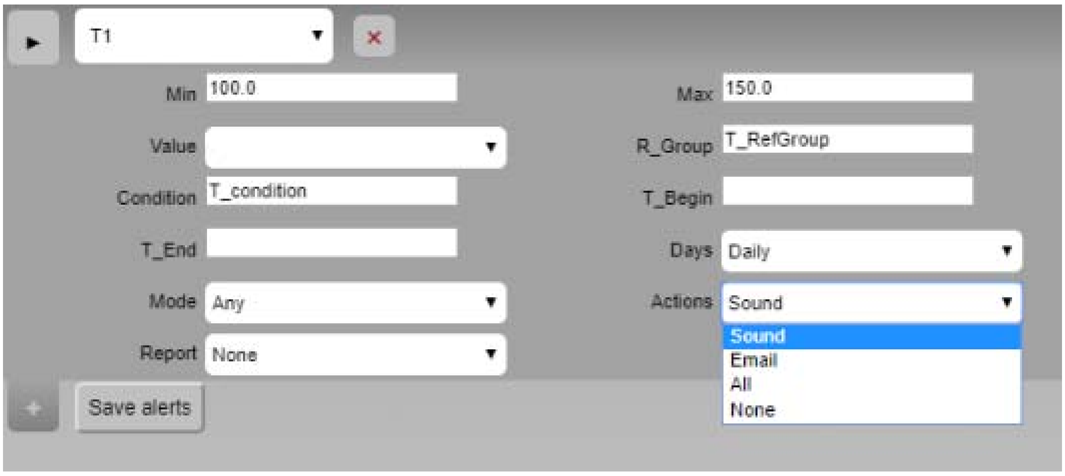
Finally, in timers tab we can configure when a determined scene starts or stops working by defining the starting and ending hours and the days on which the timer is active

5.5 Modifying alerts

In order to verify the correct functioning of the alert software specification it is going to be tested a set of cases in which alerts must behave as expected. These cases will be divided in different sections depending on the specification they are referred. The sections are:

1. Verifying actions specifications.

2. Verifying mode specification.



T1

Min 100.0

Max 150.0

Value

R_Group T_RefGroup

Condition T_condition

T_Begin

T_End

Days Daily

Mode Any

Actions Sound

Report None

Figure 5.13: Example of alert with the all the possible actions.

3. Verifying days specification.
4. Verifying time specifications.
5. Verifying values specifications.
   5.1. Verifying analogical sensor values specifications.
   5.2. Verifying digital sensor values specifications.
   5.3. Verifying actuator values specifications.
6. Verifying conditions specifications.

In the majority of the sections the test cases are only going to be realized by analogical temperature sensors, since the specifications related to these sections work in the same way for actuators, digital sensors and analogical sensors.

5.5.1 Verifying actions specifications.

In this section it will be analysed the correct performance of the action specifications in the following order:

1. Sound action.
2. Email action.
3. All actions.
4. None action.

The alert used in this section, which must have the structure of the one shown in the Figure 5.13, is linked with an analogical temperature sensor called *T1*.

Next,the cases that verify the action specifications are described.

- □ Create an alert with the structure of the shown in the Figure 5.13. Fill the actions field with ”Sound” and reset AtenTTo.
- □ AtenTTo must start beeping, since the minimum value of the alert is too high.
- □ Change the actions field into ”Email” and reset AtenTTo.
- □ AtenTTo will send an email to its associated email address.
- □ Change again the action field into ”All” and reset AtenTTo.
- □ AtenTTo must start to beep and send an email to its associated email address.
- □ Finally, change the action field into ”NONE” and reset AtenTTo.
- □ AtenTTo will not perform any action.

5.5.2 Verifying mode specification.

In this section it will be checked every mode specification in the following order:

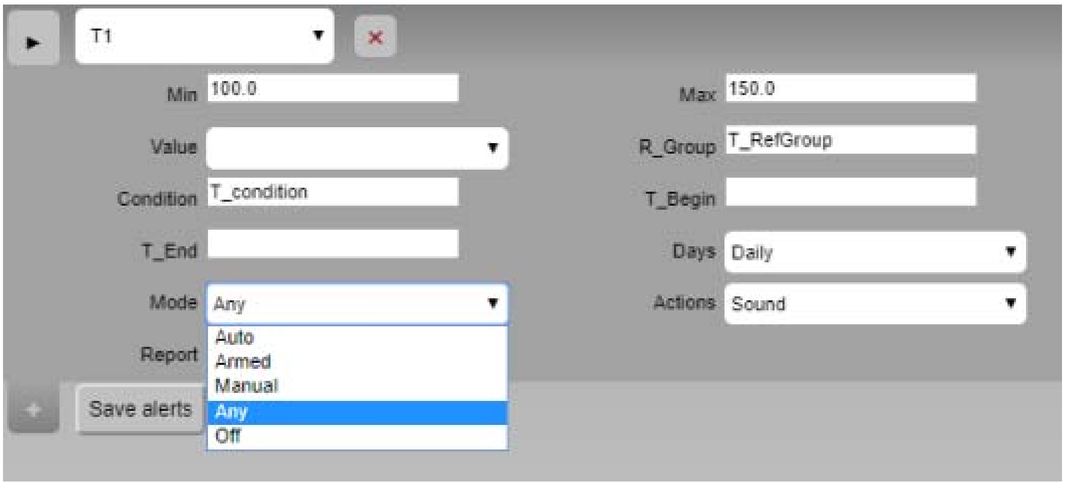
1. Any mode.
2. Auto mode.
3. Armed mode.
4. Manual mode.
5. Off mode.

In order to get it, it is going to be used the sensor *T1*, which have been already used in the previous section. It might also be used the prior alert, but in this case the action gap is going to be set as ”Sound” and it is only going to be changed the mode field, as it is shown in the Figure 5.14.

It is important to mention that from now on, every alert will have the action ”Sound” in order to ease the rest of the benchmark process.

Next,the cases that verify the mode specifications are described .

- □ Create an alert like the described in the Figure 5.14, set the field mode as ”ANY” and reset AtenTTo.
- □ AtenTTo must start beeping, since the current temperature is under 100 degrees and the action is set as ”Sound”.
- □ Switch the field mode to ”Auto” and reset AtenTTo.
- □ If the current AtenTTo mode is ”Auto”, it will start beeping. If not, It will happen nothing. For example, in the case of the Figure 5.15 AtenTTo will beep.



T1

Min 100.0

Max 150.0

Value

R_Group T_RefGroup

Condition T_condition

T_Begin

T_End

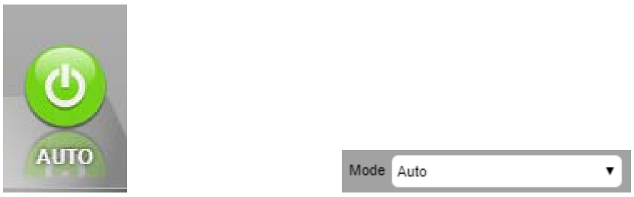
Days Daily

Mode Any

Actions Sound

Report

Figure 5.14: Example of alert with all the possible modes.



Mode Auto

(a)

(b)

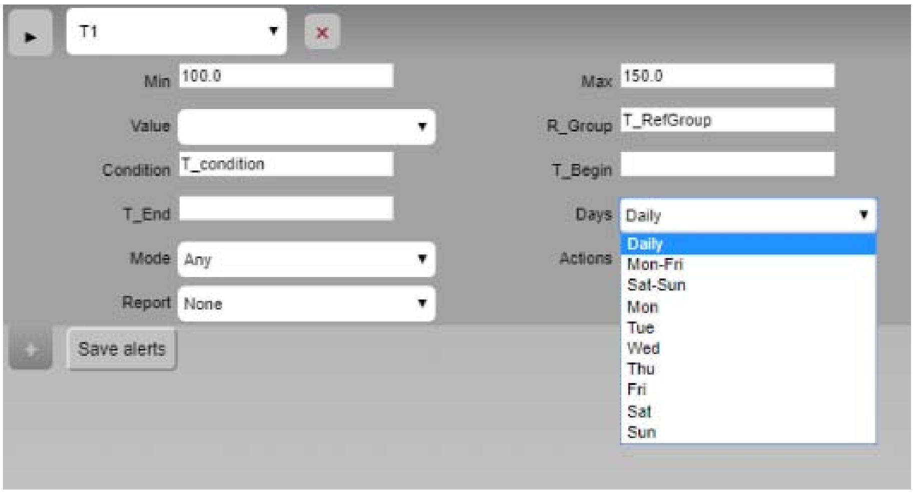
Figure 5.15: If AtenTTo is working on ”Auto” mode (Figure 5.15a) and the alert mode field is set as ”Auto” (Figure 5.15b), AtenTTo will beep.

- □ Switch the alert mode to ”Armed” and reset AtenTTo.
- □ If the current AtenTTo mode is ”Armed”, it will start beeping. If not, it will happen nothing.
- □ Change the alert mode into ”Manual” and reset AtenTTo.
- □ If the current AtenTTo mode is ”Manual”, it will start beeping. If not, it will happen nothing.
- □ Switch the alert mode to ”Off” and reset AtenTTo.
- □ AtenTTo will not do anything, even if it is working in ”OFF” mode, as alerts are disabled.

5.5.3 Verifying days specification.

In this section it is going to be analysed AtenTTo alerts operation depending on the current day. In order to do it, an alert is going to be tested varying its day field with the words shown next:

1. Using ”Daily”.



T1

Min 100.0

Max 150.0

Value

R_Group T_RefGroup

Condition T_condition

T_Begin

T_End

Days Daily

Mode Any

Actions

Report None

Figure 5.16: Example of alert with all possible days.

2. Using ”Mon-Fri”.
3. Using ”Sat-Sun”.
4. Using ”Mon”, ”Tue”, ”Wed”, ”Thu”, ”Fri”, ”Sat” and ”Sun”.

At this moment, it is important to say that every alert tested below is going to have its mode field set as ”Any” so that other specifications can be checked more easily.

In this section it is also going to work with the sensor *T1*, and it can be used the same alert of the previous sections setting its mode as ”Any”, as it is done in the figure 5.16.

The following lines describe the cases that verify the day specifications.

□ Create an alert such as the one shown in the Figure 5.16, set its days field as ”Daily” and reset AtenTTo.

□ AtenTTo must start beeping, since the current temperature is out of the range of the minimum and maximum values and the alert is checked every day.

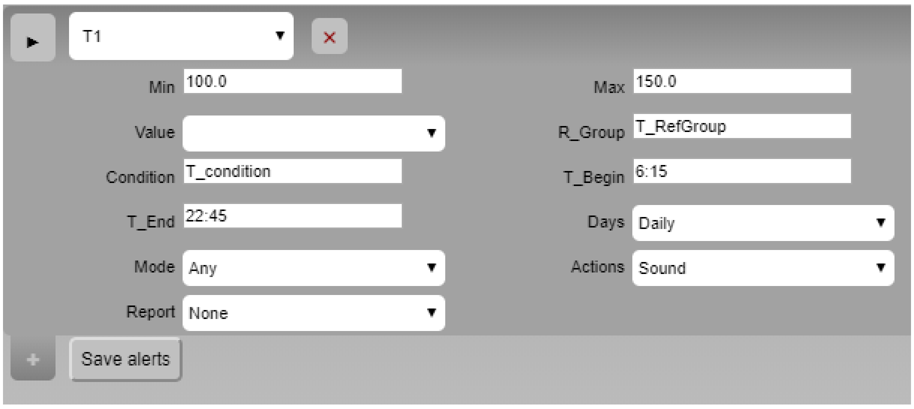
□ Change the alert day field into ”Mon-Fri” and reset AtenTTo.

□ If the current day is between Monday and Friday AtenTTo will beep, otherwise, it will not. For example, if the current day is Saturday, AtenTTo will not beep.

□ Switch the alert day into ”Sat-Sun” and reset AtenTTo.

□ If the current day is Saturday or Sunday AtenTTo will beep, otherwise, it will not. Using the previous example, if the current day is Saturday, AtenTTo will beep.

□ Change the alert day into ”Mon” and reset AtenTTo.



T1			
Min	100.0	Max	150.0
Value		R_Group	T_RefGroup
Condition	T_condition	T_Begin	6:15
T_End	22:45	Days	Daily
Mode	Any	Actions	Sound
Report	None		

Figure 5.17: Example of alert with ”time begin” and ”time end” fields defined.

□ If the current day is Monday AtenTTo will beep, otherwise, it will not. For instance, if the current day is Saturday, AtenTTo will not beep.

□ Repeat the last step using ”Tue”, ”Wed”, ”Thu”, ”Fri”, ”Sat” and ”Sun” and reset AtenTTo each time.

□ AtenTTo will only beep when the day written in the alert corresponds to the current day. Using the prior example, if the current day is Saturday, AtenTTo will only beep when ”Sat” is written.

5.5.4 Verifying time specifications.

In this section it will be checked the behaviour of alerts time specifications. To achieve this, it is going to be performed two different tests:

1. A test leaving the ”time begin” and ”time end” fields as blanks.
2. A test giving values to ”time begin” and ”time end” fields.

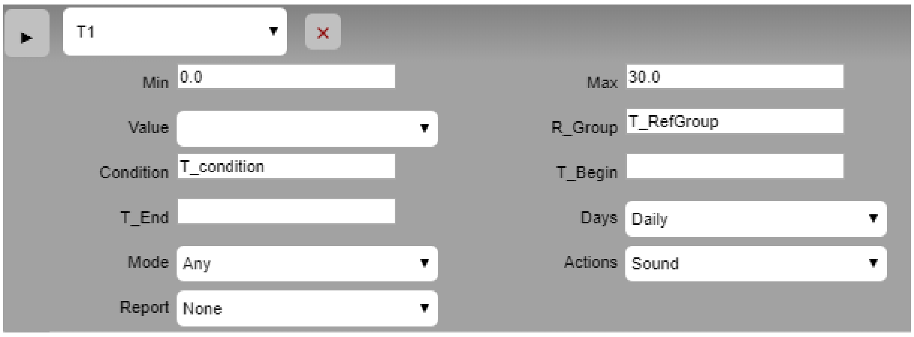
From now on, every alert checked will have its day field set as ”Daily”, which will make the testing process more general. Therefore, it is possible to to perform the tests with the sensor used before (*T1*) by setting its day field as ”Daily”. In this case the alert must have the structure shown in the Figure 5.17.

Next,the cases that verify the time specifications are described .

□ Create an alert such as the one shown in the Figure 5.17, stablish ”time begin” and ”time end” fields as blank gaps and reset AtenTTo.

□ AtenTTo must start beeping, since the current value of the sensor *T1* is out of the range of the minimum and maximum values and the alert is checked every time.

□ Set real times in ”time begin” and ”time end” fields following the proper structure (”_ _ : _ _”). For instance, 6:15 and 22:45, as it is done in the Figure 5.17.



T1

Min	0.0	Max	30.0
Value		R_Group	T_RefGroup
Condition	T_condition	T_Begin	
T_End		Days	Daily
Mode	Any	Actions	Sound
Report	None		

Figure 5.18: Example of a most likely general alert.

□ If the current time is between the range defined by ”time begin” and ”time end”, AtenTTo will beep, otherwise, it will not. For instance, if the current time is 17:34 and the ”time begin” and ”time end” fields are the ones set in the Figure 5.17, AtenTTo will beep.

5.5.5 Verifying values specifications.

In this section will be tested the operation of alerts depending on its value. This value can be defined by a range numbers if the alert is connected with an analogical sensor, or by a boolean if the alert is linked with a digital sensor or an actuator. Hence, their testing process will be done separately, distinguishing between:

1. Analogical sensors.

2. Digital sensors.

3. Actuators.

From now, ”time begin” and ”time end” fields are going to be completed as blank gaps so as to use the most likely general alert. Thus, alerts in this section and the followings are going to be tested with the most general configuration (Figure 5.18), this is:

1. Action set as ”Sound”.

2. Mode set as ”Any”.

3. Days set as ”Daily”.

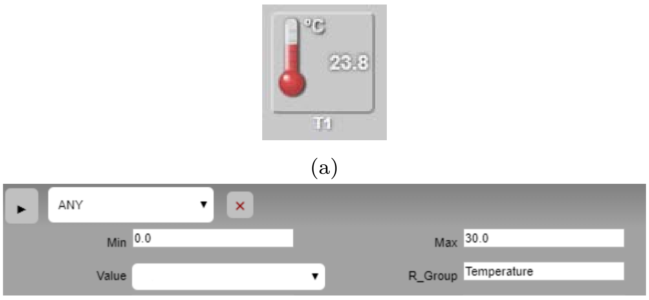
4. ”Time begin” and ”time end” set as blanks.

Besides, in this section, in order to check the correct functioning of every value specification there are going to be used more than one alert in some test cases.

Verifying analogical sensor values specifications.

This subsection is going to be divided in three main test cases:

1. Verify ”min value” and max value” specification.



(a)

ANY

Min 0.0

Max 30.0

Value

R_Group Temperature

(b)

Figure 5.19: If the value of the sensor *T1* is 23 (Figure 5.19a) and the alert ”min value” and ”max value” are 0.0 and 30.0 (Figure 5.19b), AtenTTo will not beep.

2. Verify ”ANY” specification for analogical sensors.
3. Verify ”Notnotmal” specification for analogical sensors.

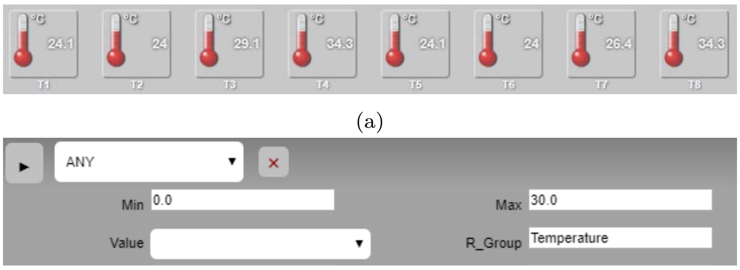
In order to perform the testing process of ”ANY” and ”Notnormal” specifications there are going to be used eight different temperature sensors.

Next,the cases that verify the analogical sensor value specifications are described .

- □ Create an alert with the structure of the one shown in the Figure 5.18 and reset AtenTTo.
- □ If the current value of the sensor *T1* is between the range defined by ”min value” and ”max value”, AtenTTo will not beep. Otherwise, it will beep. For instance, in the case of the Figure 5.19 AtenTTo will not beep.
- □ Using more than one temperature sensor, create an alert whose name is ”ANY” and reset AtenTTo. This alert must have as reference group field the magnitude of the sensors used, this is, in case of using temperature sensors, the reference group must be ”Temperature”.
- □ If any of the temperature sensors used is out of the range definded by ”min value” and ”max value”, AtenTTo will beep. Otherwise, it will not. For example, in the case of the Figure 5.20, AtenTTo will beep.
- □ Using at least four different sensors, create an alert for each of them. All these alerts must have the same reference group and different conditions. Set the value of any of them as ”Notnormal”, as it is done in the Figure 5.21, and reset AtenTTo.
- □ In the ”Notnormal” case, AtenTTo will only beep if the sensor linked with the alert suffers an error or have a strange functioning. In order to check it, draw up a flame to the sensor with the ”Notnormal” alert, as it is done in the Figure 5.22. When the sensor is hot enough, AtenTTo will beep.

Verifying digital sensor values specifications.

In this subsection it is only going to be tested the case of the value specifications for digital sensors, since these kind of sensors do not work with the ”Notnormal”



(a)

ANY

Min 0.0

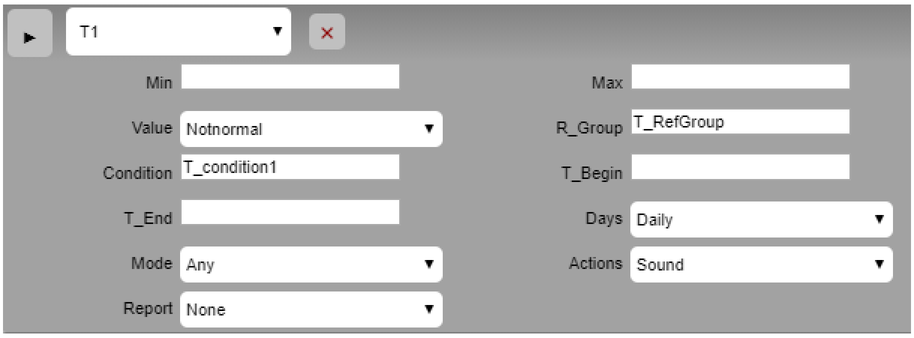
Max 30.0

Value

R_Group Temperature

(b)

Figure 5.20: As the value of the sensors *T4* and *T8* are 34.3 (Figure 5.20a) and the alert ”min value” and ”max value” are 0.0 and 30.0 (Figure 5.20b), AtenTTo will beep.



T1

Min

Max

Value Notnormal

R_Group T_RefGroup

Condition T_condition1

T_Begin

T_End

Days Daily

Mode Any

Actions Sound

Report None

Figure 5.21: Alert set as ”Notnormal” that have the same R_Group as others and different condition.

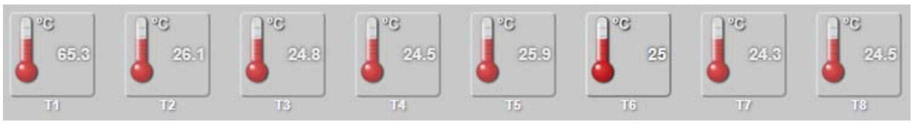
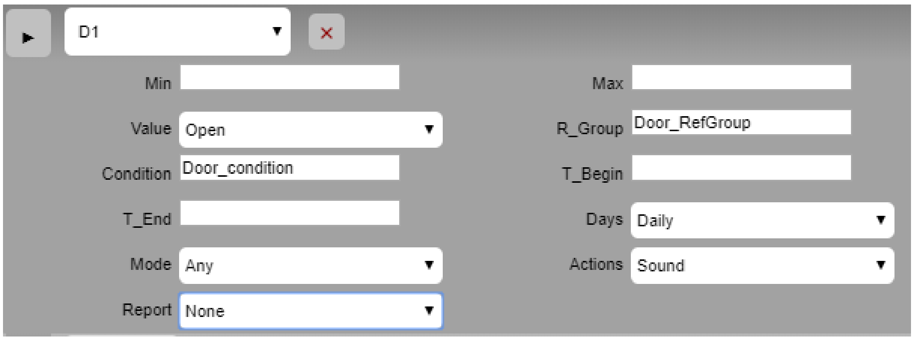


Figure 5.22: Bunch of sensors (linked with alerts with the same reference group and different conditions) in wich *T1* has a value much higher than the others. Hence, the ”Notnormal” alert linked with this sensor must be activated and AtenTTo must beep.



D1

Min		Max	
Value	Open	R_Group	Door_RefGroup
Condition	Door_condition	T_Begin	
T_End		Days	Daily
Mode	Any	Actions	Sound
Report	None		

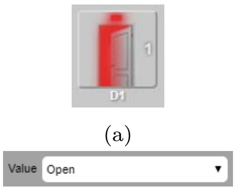
Figure 5.23: Example of alert linked with a door sensor, whose value is open.

specification. Furthermore, as the ”ANY” specification works in the same way for actuators and digital sensors (in both it is compared the current state of the sensor or actuator with the value of the alert), it will only be checked in the Verifying actuators specifications subsection.

In order to verify the value specification for digital sensors it is going to be used magnetic door sensor with the structure shown in the Figure 5.23.

Next, the case that verify the digital value specification is described.

- ☐ Create a new alert as it is described in the Figure 5.23, set its value as ”OPEN” and reset AtenTTo.
- ☐ If the door with the sensor *D1* is opened, AtenTTo will beep. Otherwise, it will not. For instance, in the case of the Figure 5.24, AtenTTo will beep.



(a)

Value: Open

(b)

Figure 5.24: As the value of the sensor *D1* is ”OPEN” (Figure 5.24a) and the alert value is ”Open” (Figure 5.24b), AtenTTo will beep.

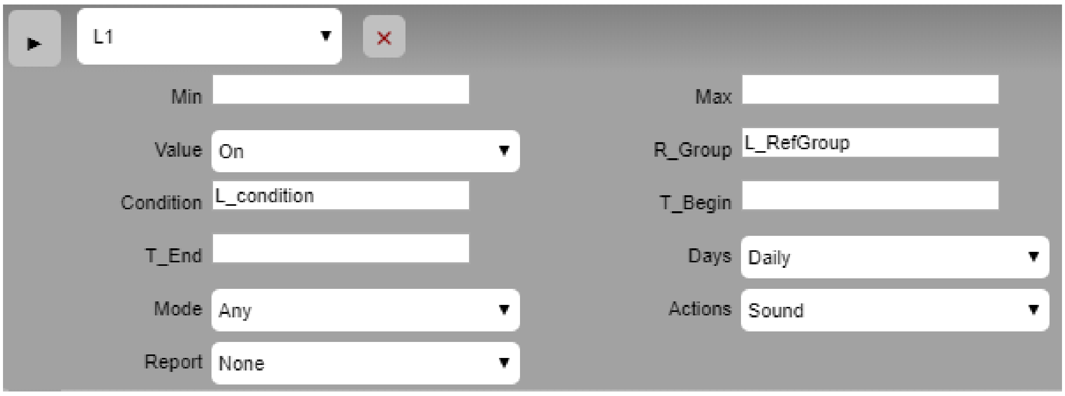
Verifying actuator values specifications.

This subsection is going to be divided in three different test cases:

1. Verify value specifications for actuators.
2. Verify ”ANY” specification for actuators (and digital sensors).
3. Verify ”Notnormal” specification for actuators.

In order to realize these tests it will be used one light device for the value specification and eight for the ”ANY” and ”Notnormal”.

The following lines describe the cases that verify the day specifications.



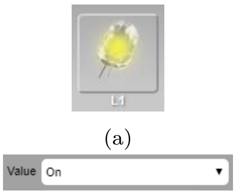
▶ L1 ▼ ✕

Min		Max	
Value	On ▼	R_Group	L_RefGroup
Condition	L_condition	T_Begin	
T_End		Days	Daily ▼
Mode	Any ▼	Actions	Sound ▼
Report	None ▼		

Figure 5.25: Example of alert linked with a light actuator, whose value is ”On”.

- □ Create a new alert as it is shown in the Figure 5.25, set its value as ”On” and reset AtenTTo. This alert must be linked with an existing light device.

- □ Enter the AtenTTo website and switch on the light device in the main menu as it has been done in the Figure 5.26a.



(a)

Value On ▼

(b)

Figure 5.26: As the value of the actuator *L1* is ”On” (Figure 5.26a) and the alert value is ”On” (Figure 5.26b), AtenTTo will beep.

- □ AtenTTo must start beeping.

- □ Turn off the light from AtenTTo web page.

- □ AtenTTo must stop to beep.

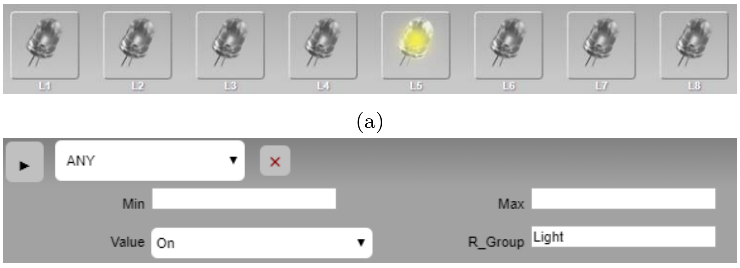
- □ Having different light devices connected to AtenTTo, create a new alert called ”ANY” and reset AtenTTo. It is important that the reference group of this alert is *Light*, as it is shown in the Figure 5.27b.

- □ Turn on any light form AtenTTo web page.

- □ AtenTTo must begin beeping.

- □ Create a list of alerts, in which at least one of the alerts has ”NOTNORMAL” as value and reset AtenTTo. It is important that all of them have the same reference group and different conditions.

- □ Turn on all lights around 2 minutes and then switch off all of them.



(a)

ANY

Min

Max

Value On

R_Group Light

(b)

Figure 5.27: As the value of the actuator *L5* is ”On” (Figure 5.27a) and the ”ANY” alert value is ”On” (Figure 5.27b), AtenTTo will beep.

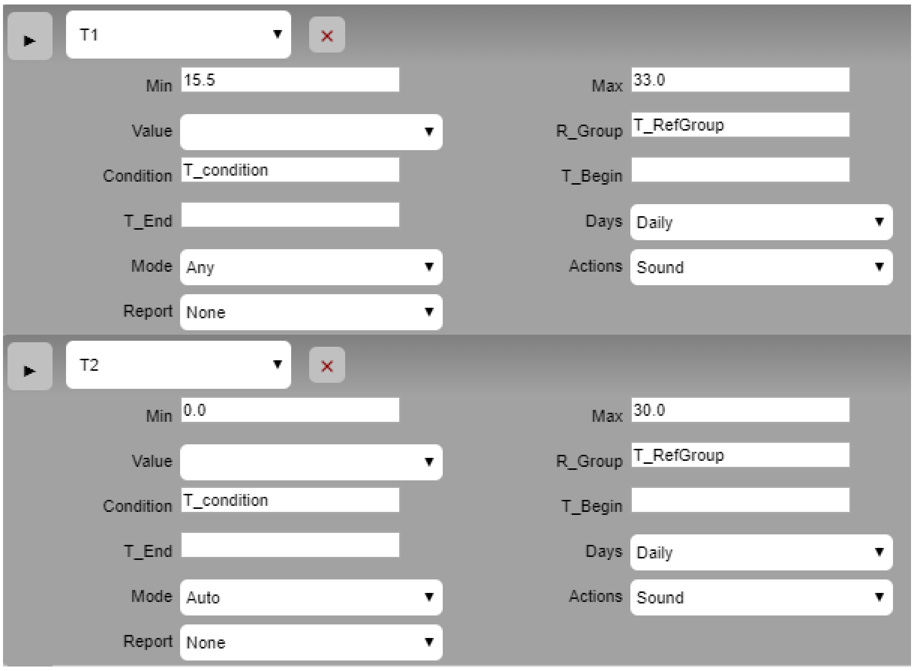
- □ Turn on only the light whose alert has ”NOTNORMAL” as value.
- □ When enough time has elapse, AtenTTo will start beeping.

5.5.6 Verifying conditions specifications.

In this section is going to be checked the condition specifications. In order to get it, two alerts linked with temeperature sensors are going to be tested by checking all the possible combinations between them. It is important to know that the behaviour of these two alerts can be extrapolated to a more amount of alerts, and that alerts linked with actuators or digital sensors work in the same way.

Next,the cases that verify the analogical sensor value specifications are described .

- □ Create two alerts which have the same condition field as it is shown in the Figure 5.28 and reset AtenTTo.
- □ If the current temperature measured by each sensor is between the range of its respective min and max values, AtenTTo will not beep. Otherwise, it will. For instance, in the case shown in the Figure 5.29 AtenTTo will not beep. Nevertheless, if the temperature of *T1* was out of the range of 20.0 and 35.0, AtenTTo would beep.
- □ Change action field of the alert *T2* into ”NONE” and reset AtenTTo.
- □ Conditions will behave in the same way as before. However, if AtenTTo takes an action, it will be the one of the alert *T1*.



T1

Min 15.5

Max 33.0

Value

R_Group T_RefGroup

Condition T_condition

T_Begin

T_End

Days Daily

Mode Any

Actions Sound

Report None

T2

Min 0.0

Max 30.0

Value

R_Group T_RefGroup

Condition T_condition

T_Begin

T_End

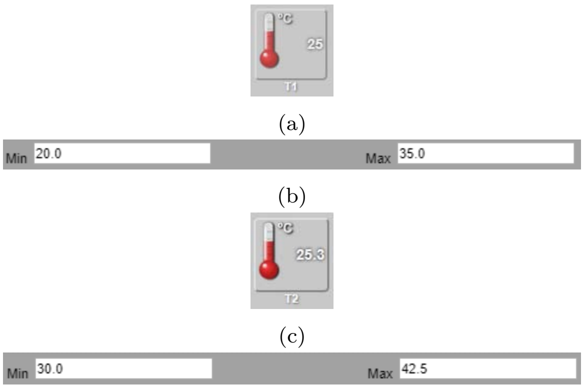
Days Daily

Mode Auto

Actions Sound

Report None

Figure 5.28: Example of alerts with the same condition.



(a)

Min 20.0

Max 35.0

(b)

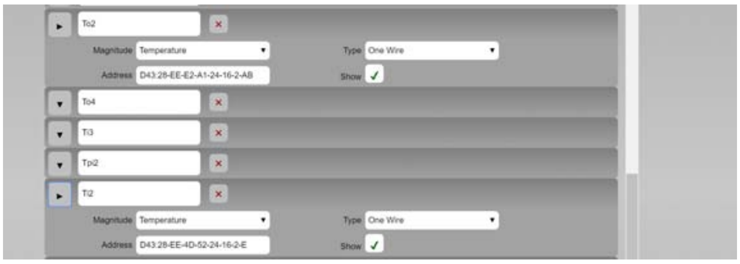
(c)

Min 30.0

Max 42.5

(d)

Figure 5.29: Temperature of *T2* (5.29c) is out of the alert values (5.29d), thus, alert *T2* is activated. However the temperature of *T1* (5.29a) is into the range of values of its alert (5.29b), so alert *T1* is not activated. Therefore, the condition is disabled and AtenTTo will not beep.



To2
Magnitude: Temperature
Type: One Wire
Address: D43:28-EE-E2-A1-24-16-2-AB
Show: ✓

To4

Ti3

Tpi2

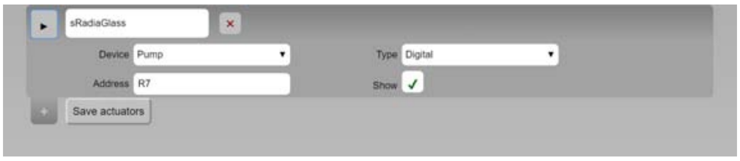
Ti2
Magnitude: Temperature
Type: One Wire
Address: D43:28-EE-4D-52-24-16-2-E
Show: ✓

Figure 5.30: Configuration of inlet and outlet temperature (T_{i2}, T_{o2}) of an InDeWaG module

5.6 Application for the WFG facade

Every InDeWag module module has two sensor probes: inlet and outlet temperature. These OneWire sensors have a hexadecimal unique code which allows to identify every module. In the configuration process, these identification codes are labeled with a more friendly name which allows to identify where this module is located in a specific facade. This configuration process is shown in Figure 5.30. Module number 2 is identified by its inlet temperature T_{i2} and outlet temperature T_{o2} which are linked to unique hexadecimal OneWire codes (28-EE-E2-A1-24-16-2-AB and 28-EE-4D-52-24-16-02-0E addresses of Figure 5.30. Every module has its own solar pump. This hydraulic pump is switch ON and OFF by an actuator. In the system configuration process this actuator is chosen to be a relay R7 as it is shown in Figure 5.31. By this way, the secondary circuit is controlled by relay R7. The hydraulic pump of the primary circuit should be linked to other relay. This configuration process is shown in Figure 5.32. Relay R8 is associated to the pump **pRadiaGlass** which governs the flow rate of the primary circuit of the water flow glazing facade. Once these two actuators **pRadiaGlass** and **sRadiaGlass** are created, a scene or macro is configured to switch ON or OFF these pumps at the same time. The scene **RadiaGlass_winter** is created in Figure 5.33 to govern at the same time **pRadiaGlass** and **sRadiaGlass**. Finally, a thermostat will activate the scene **RadiaGlass_winter** based on the interior temperature. This thermostat is shown in Figure 5.34.

The correct functioning of this module is shown in Figure 5.35. The water flow glazing module number 2 has the inlet temperature T_{i2} and the outlet temperature T_{o2}. Its hydraulic pump is named **sRadiaGlass** which is attached a relay R7. By activating this relay, the hydraulic pump will govern the flow in the secondary circuit. The flow of the primary circuit will be governed by relay R8 or the hydraulic pump **pRadiaglass**. Since the interior temperature is higher than the setpoint of the thermostat (21 Celsius degrees), the scene **RadiaGlass_winter** is not activated and the primary and the secondary pumps are OFF.



sRadiaGlass

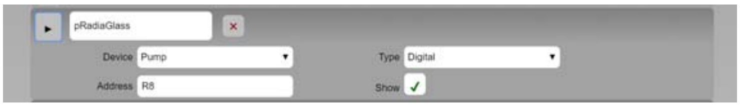
Device Pump

Type Digital

Address R7

Show ✓

Figure 5.31: Configuration of relay $R7$ to control the hydraulic pump of a InDeWaG module named **sRadiaGlass**



pRadiaGlass

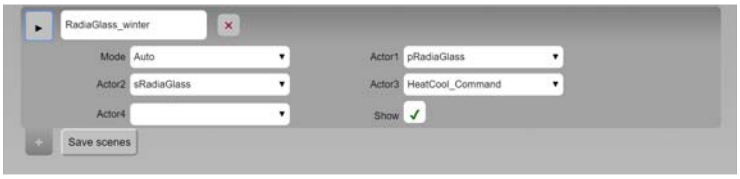
Device Pump

Type Digital

Address R8

Show ✓

Figure 5.32: Configuration of relay $R8$ to control the hydraulic pump ofthe primary circuit named **pRadiaGlass**



RadiaGlass_winter

Mode Auto

Actor1 pRadiaGlass

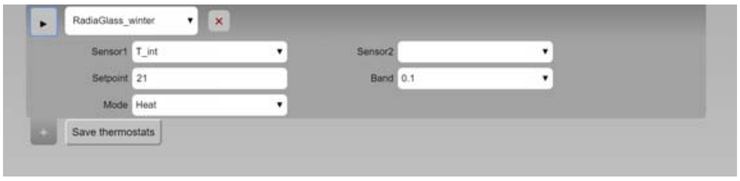
Actor2 sRadiaGlass

Actor3 HeatCool_Command

Actor4

Show ✓

Figure 5.33: Final configuration and normal operation of a complete WFG facade.



RadiaGlass_winter

Sensor1 T_int

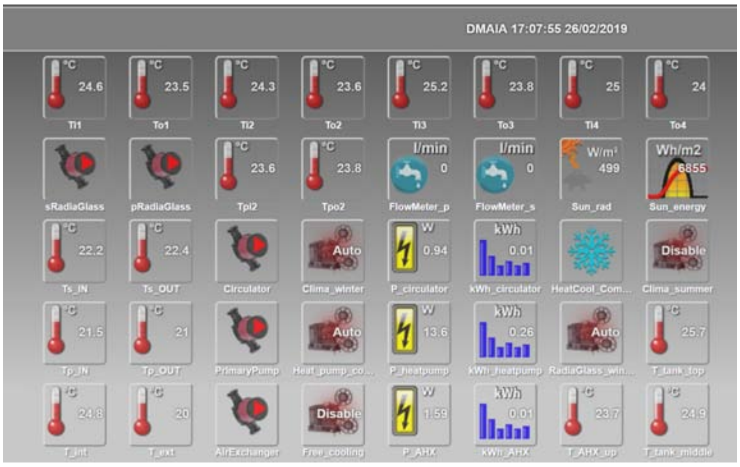
Sensor2

Setpoint 21

Band 0.1

Mode Heat

Figure 5.34: Final configuration and normal operation of a complete WFG facade.



DMAIA 17:07:55 26/02/2019

Ti1 °C 24.6
To1 °C 23.5
Ti2 °C 24.3
To2 °C 23.6
Ti3 °C 25.2
To3 °C 23.8
Ti4 °C 25
To4 °C 24

sRadiaGlass
pRadiaGlass
Tpi2 °C 23.6
Tpo2 °C 23.8
FlowMeter_p l/min 0
FlowMeter_s l/min 0
Sun_rad W/m² 499
Sun_energy Wh/m2 6855

Ts_IN °C 22.2
Ts_OUT °C 22.4
Circulator
Clima_winter Auto
P_circulator W 0.94
kWh_circulator kWh 0.01
HeatCool_Com...
Clima_summer Disable

Tp_IN °C 21.5
Tp_OUT °C 21
PrimaryPump
Heat_pump_co... Auto
P_heatpump W 13.6
kWh_heatpump kWh 0.26
RadiaGlass_win... Auto
T_tank_top °C 25.7

T_int °C 24.8
T_ext °C 20
AirExchanger
Free_cooling Disable
P_AHX W 1.59
kWh_AHX kWh 0.01
T_AHX_up °C 23.7
T_tank_middle °C 24.9

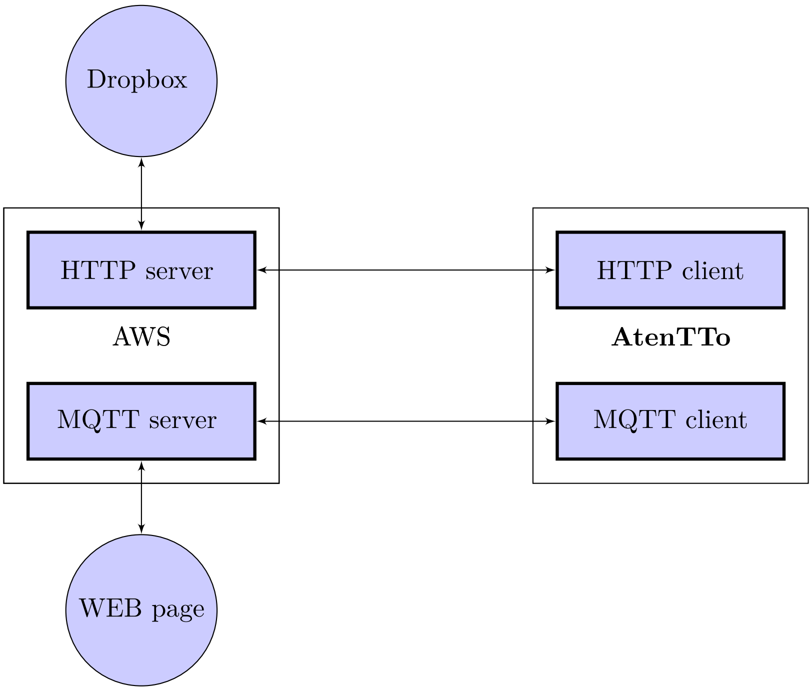
Figure 5.35: Final configuration and normal operation of a complete WFG facade.

Chapter 6

AtenTTo communication

6.1 Introduction

AtenTTo can communicate with the exterior world by means of two protocols: HTTP and MQTT. This communication is used to monitor sensor data or to command AtenTTo externally. The two channels of communication are redundant. AtenTTo acts as a client of a HTTP server or a mosquitto MQTT server. These servers are hosted in the AWS platform (IP:3.123.102.155:8088). The MQTT channel is used to interact with AtenTTo by menas of a web page and the HTTP channel is used to upload monitoring data to the Dropbox cloud. The communication diagram is shown in the following schematic.



6.2 HTTP communication

6.2.1 HTTP server

As it was said, the HTTP server is hosted in the AWS platform (IP:3.123.102.155:8088). The following Javascript code runs in AWS by means of node.js:

Listing 6.1: `HTTP_server.js`

The HTTP server responds to GET and POST requests.

- **GET requests:**

  1. Atentto client requests something like:
     `GET /DMAIA/requests/requests.txt HTTP/1.0`
  2. The HTTP server responds by downloading from Dropbox the file:
     `/Facilities_Monitoring/DMAIA/requests/requests.txt`.
  3. Once this file is downloaded, the content is sent to the Atentto client.
  4. Finally, the content of this file is erased and this empty file is uploaded to Dropbox.

- **POST requests**

  1. Atentto client requests something like:
     ```
     POST /DMAIA HTTP/1.0
     Content-Length:12345
     config/sensors.ini = File content
     ```
  2. The HTTP server responds by uploading to Dropbox the file:
     `/Facilities_Monitoring/DMAIA/config/sensors.ini`.

Hence, `GET` request can be understood as a request of downloading some information from the Dropbox cloud and `POST` request can be understood as a request of uploading information to the Dropbox cloud. Atentto could request (POST or GET) any path and any files to the `HTTP_server`. Namely, configuration files: `*.ini`, monitoring data files: `*.plt` and pending orders: `requests.txt`

6.3 HTTP client

Atentto HTTP client is written is C++ and running in Atentto platform. Every five minutes, Atentto POST its state (state.out) and GET the requests.txt file for pending orders. This frequency can be modified by means of the configuration file: atentto.ini. Besides, every night Atentto POST its gathering monitoring data of the day before. Atentto also posts every night the configuration files in order to have updated configuration files of every facility.

6.3.1 Atentto commanded from outside

Atentto can be commanded from outside by means of HTTP protocol. External orders can be written in requests.txt file and executed by Atentto every five minutes. The syntax of this file is explained in the following code:

```
command1=value1
command2=value2
...
commandN=valueN
```

The following list describes different commands or actions that are implemented:

- ACTUATOR_NAME=VALUE: where ACTUATOR_NAME is the name of some actuator of the configuration file actors.ini and VALUE is the desired commanded value to be set.

- main_switch=mode: where main_switch stands for the functioning Atentto MODE (AUTO, MANUAL,OFF and ARMED).

- reset=1. This command resets the Atentto facility.

- update_RTC=HH:MM:SS dd/mm/yyyy. It updates the time and date of the internal AtenTTo clock.

- upload=FILE. This command will force AtenTTo to perform a GET request to download from Dropbox the configuration file FILE. Once this file is served, AtenTTo will write its content to its SD card. This command is usually related to configuration files such as:

  upload=/config/sensors.ini.

- download=FILE. This command will force AtenTTo to perform a POST request to send the file FILE. Once this file is successfully sent, HTTP remote server will upload its content to the Dropbox folder. This command is usually related to monitoring data files such as:

  download=/datalog/2019/10/20191025.plt

  or configuration files such as:

  download=/config/atentto.ini.

6.4 MQTT communication

As it was shown, MQTT communication protocol is used to show AtenTTo online data on a specific web page. AtenTTo initiates the MQTT communication by opening a socket or channel with the name of the facility. This is called the subscription topic. The MQTT protocol consists in two parts: the topic and the message. MQTT topics are a form of addressing that allows MQTT clients to share information. A topic only exists if a client has subscribed to it.

On the other hand, the web page is built with a MQQT client that subscribes the desired topic or facility. AtenTTo and the web page are now MQTT clients that communicate with the mosquitto MQTT server hosted in the AWS. Once the web page is loaded, some initialization messages are sent to the mosquitto MQTT server which redirects the message to all clients that are subscribed to that specific topic.

The communication between the web page and the AtenTTo platform is done through the mosquitto MQTT server in the following way:

- The web page is subscribed to one topic to receive messages from the server and uses another topic to publish (send) messages to the server.
- Three topics are created when connected:
  1. to publish messages to AtenTT.o
  2. to receive messages from AtenTTo.
  3. to receive only the initial files needed to load the full web page.
- When the user tries to connect to one facility, the web page send an initial message to AtenTTo and it responds with the configuration files (`/config/file.ini` and the *state.out* file. Once the web page is fully loaded, it sends a message at a constant time interval asking for the AtenTTo state. Once this file content is received, the web page is updated.

6.4.1 MQTT mosquitto server

The mosquitto MQTT server is used to redirect different messages between all clients linked to some specific topic.

6.4.2 Web page MQTT message and AtenTTo action

The web page sends to AtenTTo via the mosquitto server different messages of the form:

```
message_label=message_value
```

If `message_value` is not present in the message, `message_label` can be the following list:

- `sendinifiles`. When AtenTTo receives this message, it sends the following JSON files:
  1. { “atentto”: atentto.ini content }
  2. { “sensors”: sensors.ini content }
  3. { “actors”: actors.ini content }
  4. { “macros”: macros.ini content }
  5. { “thermo”: thermo.ini content }
  6. { “timers”: timers.ini content }
  7. { “wired”: wired.ini content }
  8. { “alerts”: alerts.ini content }
- `state.out`. AtenTTo sends the folowing JSON file:
  1. { “state”: state.out content }
- `file.ini`: AtenTTo sends with JSON format the initialization file.

If `message_value` is present, the following different messages are processed:

- `sendplot=path_file_name`. When AtenTTo receives this message, it sends the following JSON file:
  1. { “plot”: path_file_name content }
- `file.ini=file_content`. AtenTTo writes the `file.ini` content in the `/config/` folder of the AtenTTo SD card.
- `state.out`. AtenTTo sends the folowing JSON file:
  1. { “state”: state.out content }
- `command=value`. This kind of message is processed in the same manner as if the message were received by a HTTP protocol.

Chapter 7

Predictive maintenance

7.1 Introduction

Predictive maintenance techniques allow to determine the condition of a WFG facade in order to estimate when maintenance should be performed. This approach promises cost savings over routine or time-based preventive maintenance, because tasks are performed only when warranted.

The main promise of predictive maintenance is to prevent unexpected equipment failures. Predictive maintenance differs from preventive maintenance because it relies on the actual condition of InDeWaG modules, rather than average or expected life statistics, to predict when maintenance will be required. Some of the main components that are necessary for implementing predictive maintenance are data collection and preprocessing and early fault detection. Predictive maintenance evaluates the condition of water flow glazing modules by performing periodic (off-line) or continuous (online) condition monitoring. One goal is to upload data to the cloud in order to trigger maintenance planning, work order execution, and reporting. Ideally, predictive maintenance allows the maintenance frequency to be as low as possible to prevent unplanned reactive maintenance, without incurring costs associated with doing too much preventive maintenance. A key element in this process is the Internet of Things (IoT). IoT relies on predictive maintenance sensors to capture thermal performances of a WFG facade. To check periodically thermal performances, generic alerts are defined to cover all possible malfunctions of the system.

The maintenance services comprises two main services:

1. Alerts detected by AtenTTo platform. AtenTTo by means of file named: `alerts.ini` has configured alerts that are checked every time step `(DT_alerts)`. When some anomaly or warn is reached, AtenTTo will alert by beeping or by sending an email.

2. Malfunction conditions detected from an external server. Since AtenTTo state is uploaded every time step `(DT_post)`, an external server will check its condition. If some malfunction is detected, the server will inform of the anomaly with an email.

7.2 Malfunctions detected by an external server

1. The server monitors the time of each facility. If AtenTTo is delayed, it sends an email to inform
2. Possible malfunctions based on normal behavior.

Algorithm

1. Read dropbox facilities monitoring
2. Read state.out file
3. Extract time and compare with actual time.
4. send an emial if delayed.

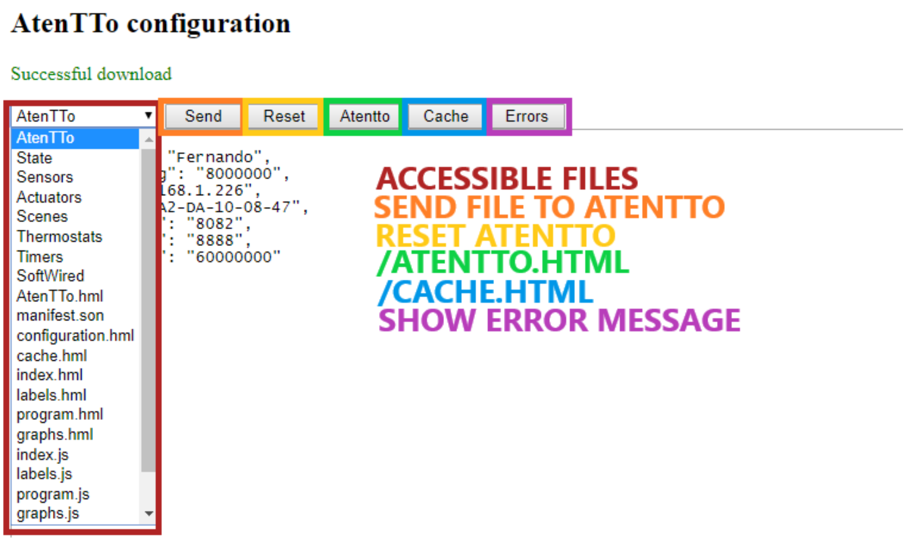
Chapter 8

Advanced configuration

If Atentto is not working properly, or if we desire to configure it manually, we can browse the following default direction: 192.168.1.200:8082/configuration.html

From there it is possible to read and modify most of the files inside the SD card, such as the configuration files or the files related to the web page.

8.1 main page



AtenTTo configuration

Successful download

```
 "Fernando",
g": "8000000",
168.1.226",
A2-DA-10-08-47",
': "8082",
': "8888",
': "60000000"
```

Figure 8.1: Configuration page

As it is shown in the image, pushing the ”Send” button uploads the current file to the SD card, to make any changes effective you should click on the ”Reset” button, that will reset Atentto. The button labelled ”Atentto” redirects to the

main page, and the button named ”Cache” redirects us to ”SERVER:PORT/-cache.html” which serves all the resources written in the CACHE section of the manifest. It is also important to notice that clicking on the cache button is also necessary for the web page to recharge any changes on the .ini files.
It is worth to make a separate comment about the ”Errors” button. This button is implemented for solving the problems that Atentto can handle during the setup process. If Atentto finds a semantic error inside the .ini files, then the configuration alert will start to sound (one low beep every 7 seconds) and clicking the errors button will show a concrete error message. In order to solve this, you should modify the .ini file that is wrong, send it and finally reset Atentto.

There are some functions that are not accessible through the main page, although they can be accessed typing their directions in the command bar.

8.2 SD structure and files

The SD Card used by Atentto contains most of the resources needed to build the Atentto web page, besides .plt files with data provided by sensors. These resources are located inside the folders named **“appweb”** and **“config”**, while the .plt files are inside **“datalog”**, shorted by date with different subfolders. Furthermore, it can also contains another folder named **“sampling”**, which has the Fast Fourier Transform (FFT) of some the data provided by the analogical sensors.

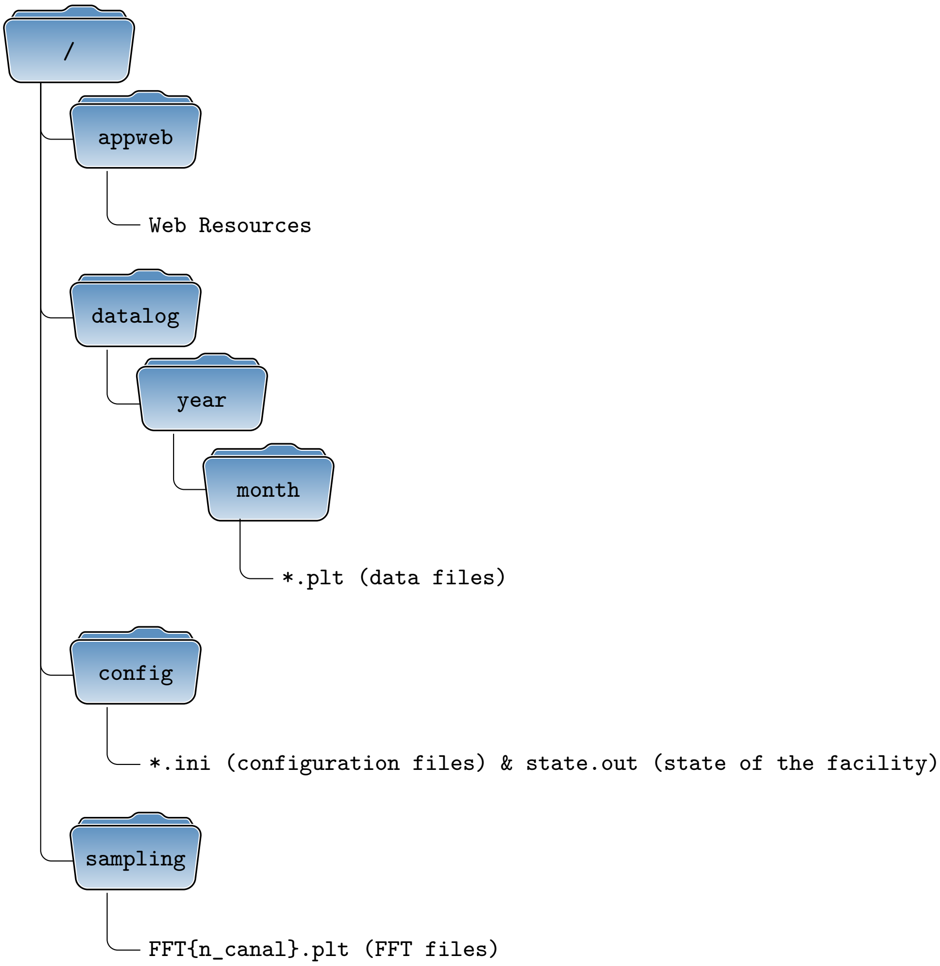

/
appweb
Web Resources
datalog
year
month
*.plt (data files)
config
*.ini (configuration files) & state.out (state of the facility)
sampling
FFT{n_canal}.plt (FFT files)

Figure 8.2: SD Card folder & file structure

The configuration files, such as the list of sensors or the list of timers, and the **"state.out"** (it contains the current state of the facility) are in the **"config"** folder. These are also necessary to build the web page, although they don't have any information about its structure or functionality.

With regard to the web page, there are **4 subfolders** and **4 files** inside the "appweb" folder, as you can see in the image located in the following page. The "css", "html" and "js" subfolders, as their names suggest, contain the .css, .html and .js files, respectively. These compose the source code of the Atentto's web page. On the other hand, the "img" subfolder contains most of the images used.

It is important to say that the .html files are not exactly the final code used in the web page, meaning that they are **templates** which are going to be modified before being used by the browser. The Mustache library (javascript) is in charge of doing so. It is located inside the subfolder "lib" (in "js"), along with another external libraries.

Finally, inside "appweb" there are also 3 images ("favicon.ico", "favicon.png" and "touch.png") which serve as a favicon, and the **manifest** file. The favicon is the image, icon, which represents the web page. It is located right before its name, at the top of the browser. The manifest shows the resources which must be cached by the browser.

It is important to remember that **the manifest must be modified every time you want to reload the resources** (usually because some of them have been modified). Currently, all the files inside "appweb", except for the manifest file itself, are written in the CACHE section of the manifest. In order to know more about the cache manifest, you can go to "**??**" on page ??.

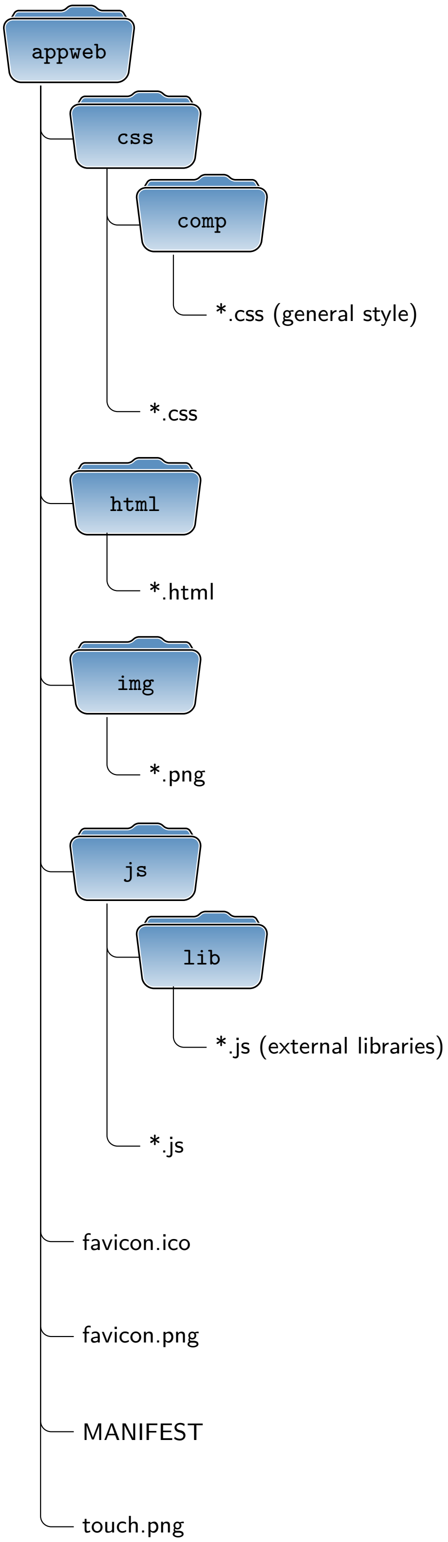
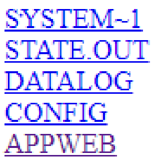


Figure 8.3: Appweb folder & file structure

Introducing the direction SERVER:PORT/ (for example: 192.168.1.222:8082/) shows the **content of the SD card**. You can also navigate through all the folders that are inside of the USB and open the files (by clicking on them). It will show its content in plain html. It is quite useful to see the SD structure.

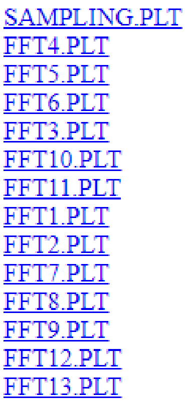


SYSTEM~1
STATE.OUT
DATALOG
CONFIG
APPWEB

Figure 8.4: First functionality of the Atentto Web Page

If it is required, any file direction can be typed in the command bar and the web page will show its content, for example http://192.168.1.226:8082/config/atentto.ini

There is another way to obtain the data from the analog sensors, accessible through SERVER:PORT/sampling. Going there will show a list of links which will look similar to this:



SAMPLING.PLT
FFT4.PLT
FFT5.PLT
FFT6.PLT
FFT3.PLT
FFT10.PLT
FFT11.PLT
FFT1.PLT
FFT2.PLT
FFT7.PLT
FFT8.PLT
FFT9.PLT
FFT12.PLT
FFT13.PLT

Figure 8.5: Sampling page

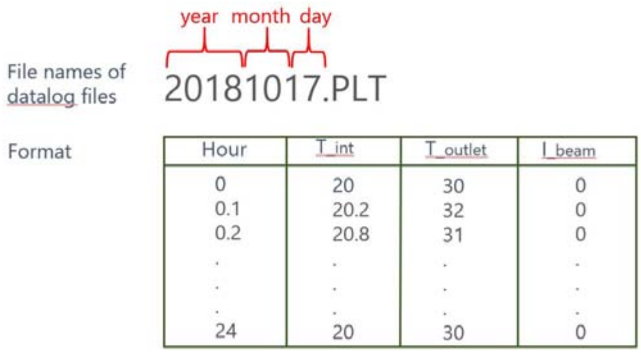
These are all the data from the analog sensors in the instant when you open this direction. The first file contains the data from all the analog sensors connected, there it can be seen the shape of the current or voltage wave and its peak values.
The following links contains the Fast Fourier Transform of each one of the analog sensors. These files are structured in a symmetric way: each file is a column of values, each value representing a harmonic of the current wave function. These harmonics start to repeat themselves from the half to the end of the document, therefore the symmetric structure.
In order to see this data plotted in a graphic the links should be downloaded to the computer by saving them as a link, and then opening them with a program

capable of reading plt files.
If there aren't any analog sensors saved in Atentto, the address SERVER:PORT/sampling will show a blank page.
It is also interesting to notice that if it is required to monitor the voltage of the electric network where Atentto is plugged, an analog sensor on the "A0" channel should be saved, monitoring alternate voltage.

8.3 Errors.out

Typing SERVER:PORT/errors.out in the command bar will redirect to a page where there are shown any error messages that Atentto sends. This could be seen as a direct way to obtain the same error massages that are shown in the configuration page clicking on the "errors" button.

8.4 Format data logging file


year
month
day

File names of datalog files 20181017.PLT

Format

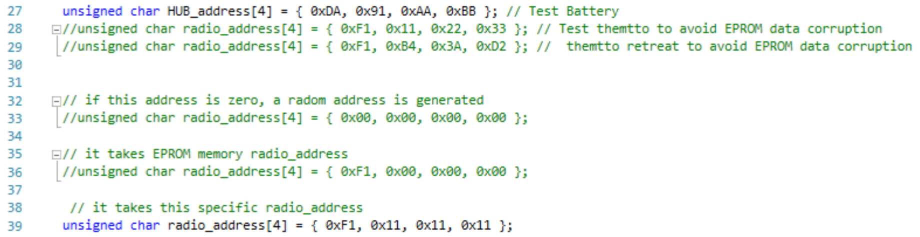
Hour	T_int	T_outlet	I_beam
0	20	30	0
0.1	20.2	32	0
0.2	20.8	31	0
.	.	.	.
.	.	.	.
.	.	.	.
24	20	30	0

Figure 8.6: Format of data logging file

8.5 List of questions

8.5.1 How to connect and configure Themtto?

Themtto is a mote with embedded software that monitors the ambient conditions (temperature and humidity) and sends them to a hub (in this case it will be Atentto) using RF22 transmission. It also allows some other functionalities which will be described later.
In order to set up a working Themtto the first step is to burn in it its embedded software: you should go to D:\Dropbox\Software\Conectto_products\Themtto\Embedded software and open the folder with the latest version. Then you should open the file "Themtto.sln" with Microsoft Visual Studio and burn it into Themtto (is the exact same process that is followed with Atentto). There are a couple of lines of the code that have to be noticed:

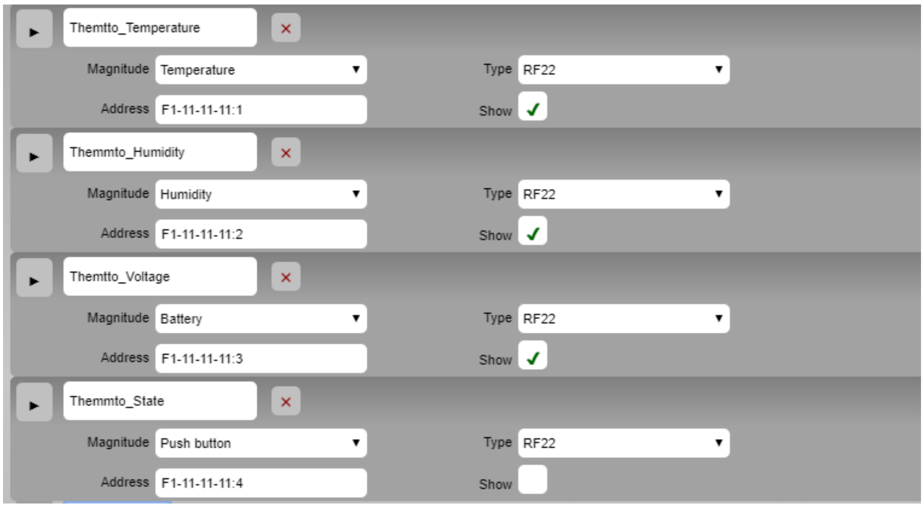


```
27    unsigned char HUB_address[4] = { 0xDA, 0x91, 0xAA, 0xBB }; // Test Battery
28   //unsigned char radio_address[4] = { 0xF1, 0x11, 0x22, 0x33 }; // Test themtto to avoid EPROM data corruption
29   //unsigned char radio_address[4] = { 0xF1, 0xB4, 0x3A, 0xD2 }; //  themtto retreat to avoid EPROM data corruption
30
31
32   // if this address is zero, a radom address is generated
33   //unsigned char radio_address[4] = { 0x00, 0x00, 0x00, 0x00 };
34
35   // it takes EPROM memory radio_address
36   //unsigned char radio_address[4] = { 0xF1, 0x00, 0x00, 0x00 };
37
38     // it takes this specific radio_address
39    unsigned char radio_address[4] = { 0xF1, 0x11, 0x11, 0x11 };
```

Figure 8.7: Configuring Themmto addresses

- In line 27 you can modify the hub address, set this to match the MAC of your Atentto. To avoid problems, set a specific MAC in your Atentto modifying the file Atentto.ini.
- Leaving line 33 uncommented assigns a random direction to Themtto.
- Leaving line 36 uncommented assigns the direction read from EPROM memory.
- Leaving line 39 uncommented assigns a determined direction to Themtto.

After configuring and uploading Themtto code the next step is going to Atentto web page and configure it in the Labels tab following this template:



Themtto_Temperature
Magnitude Temperature
Type RF22
Address F1-11-11-11:1
Show ✓

Themmto_Humidity
Magnitude Humidity
Type RF22
Address F1-11-11-11:2
Show ✓

Themtto_Voltage
Magnitude Battery
Type RF22
Address F1-11-11-11:3
Show ✓

Themmto_State
Magnitude Push button
Type RF22
Address F1-11-11-11:4
Show □

Figure 8.8: Configuring Themmto in Labels tab

Those are the four variables that Themtto sends every 8 seconds, and it is necessary to save all four if you want to link a Themtto with Atentto, if you save only one, two or three variables, Atentto will have an error. It is important to notice that the address is the one that you set before in the Themtto code. Temperature has to be set set in port 1, Humidity in port 2, and Voltage in port 3.
Port 4 allows different functionalities:

- Push button: This links the fourth port with the button in Themtto. It will show 32 as its value when a short push is done and it will show 64 as its value when a long push is done.

- Door: If there is a magnetic sensor connected to Themtto, it can work as an alarm which alerts when a door is open or closed. Selecting door as the magnitude for the fourth port will display an image of a closed door an a 0 when the switch is closed, and it will display the image of an open door and a 1 as its value when the switch is open.

8.5.2 How to connect and configure Optocouplers?

Optocouplers are digital sensors which are already implemented inside Atentto multipurpose, which are able to detect if there is or isn't an electric potential connected to them, this is used to monitor an electrical current of interest. In order to configure these the only thing that is needed is to save a digital sensor in the "Labels" page and configure its address from OC1 to OC4. Then, the optocoupler value will be zero when there isn't any potential in that line and it will be equal to one when there is some potential. This may be useful to monitor some devices, such as heatpumps which send 220 V voltage through their alarm wire when they have any problem.

8.6 Troubleshooting

8.6.1 The main page cannot be accessed

If you try to access Atentto's main page and nothing appears here are some things you can try:

- Maybe the direction that you are trying to access is wrong or leads to a different Atentto. If this happens you can try scanning the local network with your mobile phone (using applications such as Network scanner) and search for the direction of the Atentto that you want to access to. If your Atentto doesn't appear there, then the problem may be with its internet connection.

- Check the internet connection of the Atentto. If Atentto is connected to the internet then it should have two green LEDs switched on. If those are off then the Atentto hasn't any working internet connection. Also, if there is no connection to the internet, an alert will continuously sound when Atentto starts.

8.6.2 The main page seems to load but the icons aren't shown

If the web page does't load this can be due to several reasons:

- First of all, some times the web page spends a little time loading the icons, but it doesn't mean that there is an error. Also, if the web page doesn't load at first, try reloading it and wait until the icons are shown.

- If this problem persist, you can try resetting Atentto. If you have added a new sensor/actuator, then the problem may be resolved going to sensors/actuators tab and clicking on save sensors/actuators.

- The web page may have crashed for several reasons, one consistent way to solve that problem is to open the browser console clicking F12. There will be shown all the actions made by the web page, and the errors will appear in red. Clicking on one of those errors redirects to the line of the files which is causing the web page to crush.

8.6.3 The main page loads but it doesn't apply the last changes

Sometimes there had been any changes to the SD card files or the Atentto embedded software but the web page remains the same. It can be solved with the next tips:

- Any changes made to the .ini file must be recharged through the cache page, which is accessible through the configuration page or in ”SERVER:PORT/-cache.html”.

- Sometimes, the browser saves the web page files in its cache memory and it overrides the SD card files. Then, the web page will be loaded using the older files stored in the browser cache memory. This can be solved erasing the browser cache memory for that specific web page, or erasing all the browser memory.

Chapter 9

New sensors or actuators

9.1 How to use temperature OneWire probes

Atentto is able to manage more than one hundred of digital OneWire temperature probes by means of 10 digital ports(D41-D50). Atentto detects automatically Dallas Temperature sensorswhen it is reset. These new probes are stored in sensors.ini and saved into the SD card. Hence, configuration of these sensors is mada automatically. and Onewire probes are connected properly to these digital ports. Later, the user from the web page can change the label of these temperature probes.

The DS18B20 OneWire probe has the following specifications:

- they use only one 1-Wire port pin in Arduino
- Measure temperatures from -55°C to 125°C
- ±0.5°C accuracy from -10°C to +85°C
- Thermometer resolution is programmable from 9 to 12 bits
- Each sensor contain a unique silicon serial number

The information is transmitted to the digital port in the Atentto Multipurpose. With the Atentto, information can be read, written and converted. The communication is carried out by the OneWire protocol. This protocol allows to share the digital information with probes. the maximum number of probes in the same line depends on the topology of the connection of probes. Generally, a maximum of ten probes per line do not yield protocol problems. Atentto identifies OneWire sensors by using their unique hexadecimal serial number.

Symbol	Description
GND	Ground
DQ	Data Input/Output pin
V_{dd}	Optional V_{dd} pin

Table 9.1: Connections

9.1.1 How to build a OneWire temperature sensor

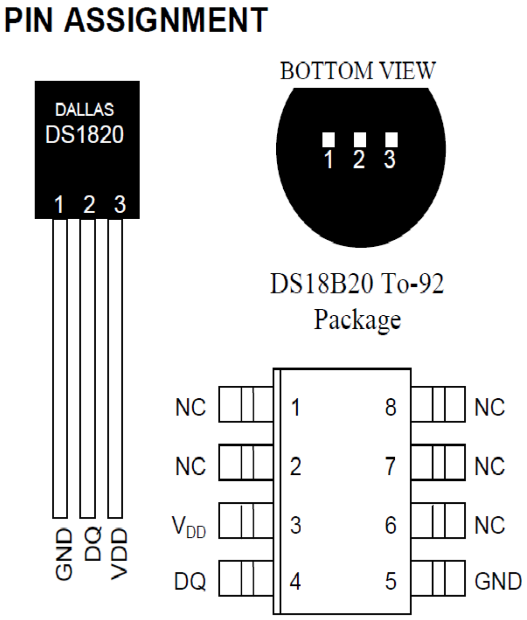


Figure 9.1: DS1820

The probe has three different connectors. The ground connector, the V_{dd} connector and the data connector. In order to connect Atentto with a sensor, it is going to be used the RJ11 cable, which is used because the connection between different RJ11s can be easily done using a splitter.

1. Select a entry of the RJ11 wire and cut it with the appropriate tool.

2. There are going to be four cables of different colors. Assign one of them for the V_{dd} other for the input/output data and finally other for the ground. Last one is going to be free. It is recommended to use the brown/black for the Ground connector, red for the V_{dd} cable and green for the data signal.

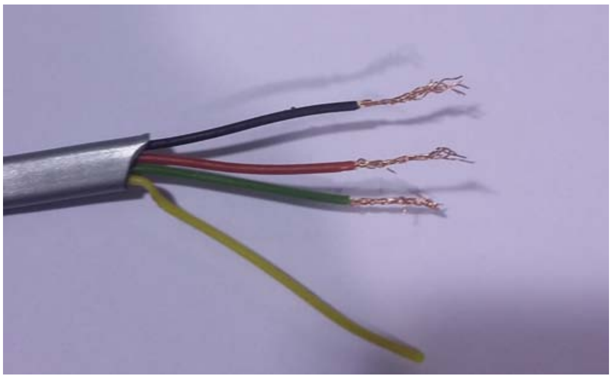


Figure 9.2: RJ11 wire

3. The connection between a sensor and each cable should be done using tin soldier. It has to be welded carefully, connecting each cable with the appropriate pin of the sensor.

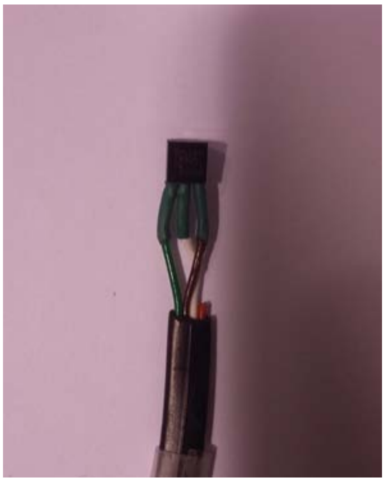


Figure 9.3: Welded probe

4. The welded part should be protected.

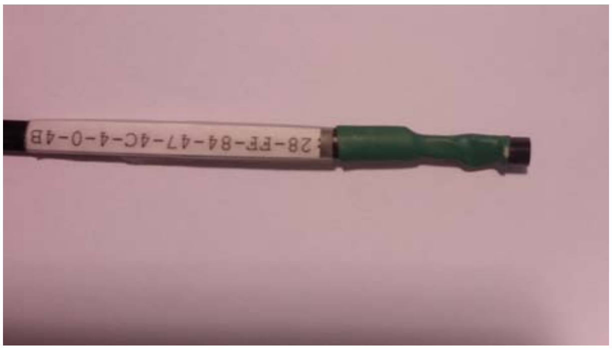


Figure 9.4: Finished probe

5. Finally, the other end of the wire should be linked with a RJ11 connector

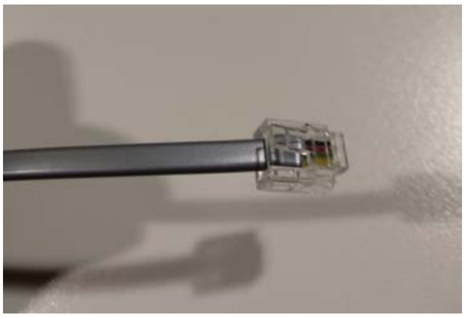


Figure 9.5: RJ11 connector

Many RJ11 cables can be joined using a splitter. Many DS1820s can be connected to the same splitter and Atentto can detect all the sensors separately using their serial number.

9.1.2 Atentto connection of an isolated OneWire probe

Each digital channel from D41 to D50 is able to read Onewire temperature probes. As it was described in section 9.1.1, a OneWire probe has three wires: 3.3 volts, ground and data line.

With the cables already selected and having clear the colors and position criteria in the RJ11, the 1-Wire bus has to be connected with the controller. It is recommended having the same color criteria in all the devices:

Ground	Black
V$_{cc}$	Red
Data	Green (or other color)

Table 9.2: Colors code

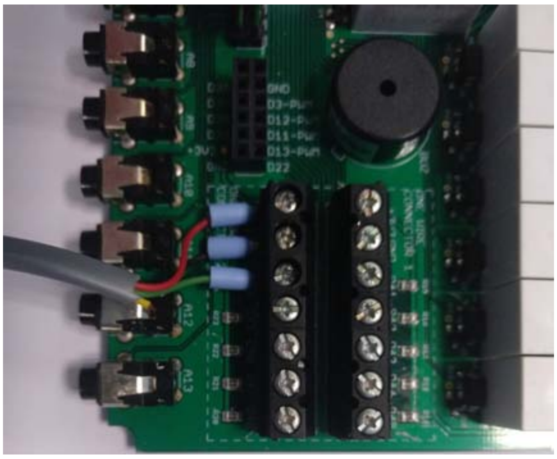


Figure 9.6: Onewire connection to Atentto multipurpose

Then, finally, the current cable is connected with the port of 3,3 volts; the ground cable is connected with the GND pin; and the data cable is connected to one of the digital ports of Atentto multipurpose.

9.1.3 Splitter connection of 5 channels

It is also possible to connect five channels to the same splitter.

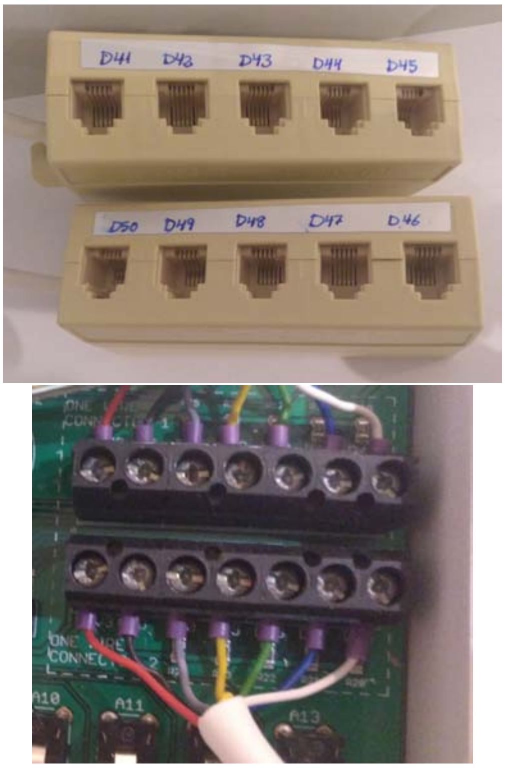


Figure 9.7: Onewire connection to Atentto multipurpose

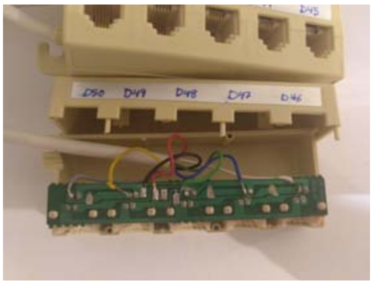


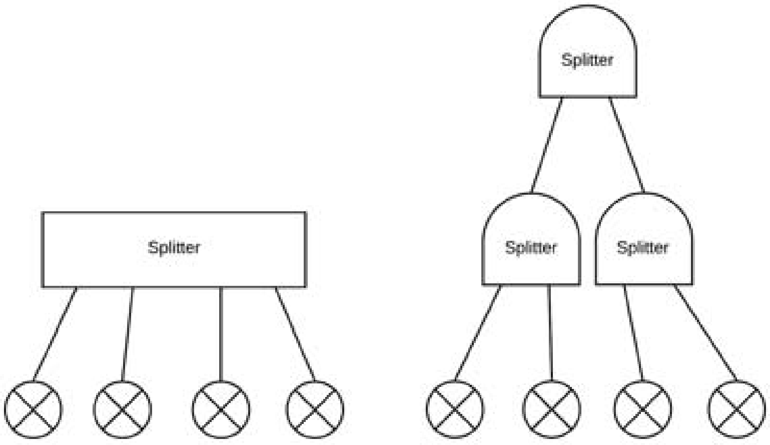
Figure 9.8: Onewire connection to Atentto multipurpose

9.1.4 Troubleshooting

It is relatively frequent to have issues with the OneWire probes, and this section should be useful in order to solve them. The actions taken will be different depending on the kind of failure that is being produced:

1. If the probe appears totally disconnected (it will show -127 as its value) for almost all the time then it might be a problem with the wiring or the connections. In order to check those connections some “OneWire wands” have been made. Those wands consist on a LED connected to an RJ11 wire. Connecting it to a suspicious RJ11 connector will check if the connection is all right: if the LED lights up, then there is signal; if it doesn’t light up, then there is not signal.

2. If the failures are random and distributed through the day, then there might be a protocol problem. These problems are harder to solve, but it is useful to use a “tree” configuration instead of a “hub & branches” configuration for the probes to correctly work. Also, if the OneWire wires run near other wires with strong current, this might cause an interference with the OneWire signals.



(a) Worse configuration

(b) Better configuration

3. The OneWire probes work with a 12 bits resolution, but sometimes they seem to be working with less resolution (they will only give values with 0’5°C resolution). This is caused when a probe is disconnected and connected again without resetting Atentto. There are some probes which have lesser resolution predetermined, but when Atentto starts it sets the resolution of all OneWire probes to 12 bits. Then, if a probe is disconnected and connected again it will start working with its default resolution, so this problem solves easily resetting Atentto every time a probe is disconnected.

9.2 Usage of the current clamps for Atentto r04 2017

In order to measure currents and power there are being used analogical current clamps. They are easy to use, fitting one of the clamps around a wire measures the amperes that are going through it. They are going to be connected to Atentto analogical ports. The only problem is that the mini-jack connectors that the clamps have integrated don't use the same configuration that Atentto multipurpose r04 2017 uses.

9.2.1 Current clamps

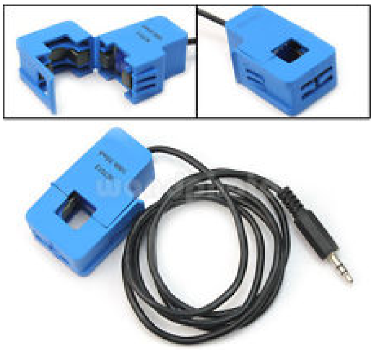


Figure 9.10: Current clamps

9.2.2 Making it possible to use the clamps

The integrated mini-jack connector is configured in the following way:

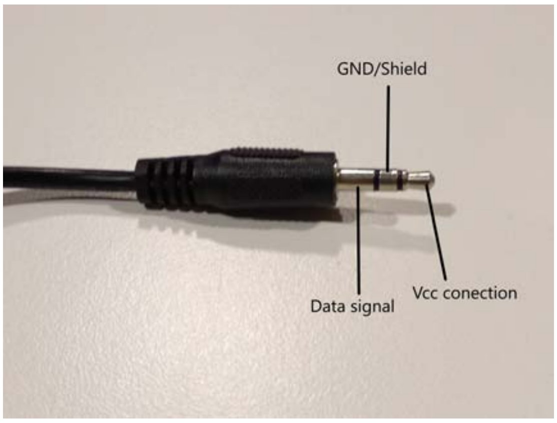


Figure 9.11: Mini-jack connector

Then, Atentto r04 2017 uses this configuration instead:

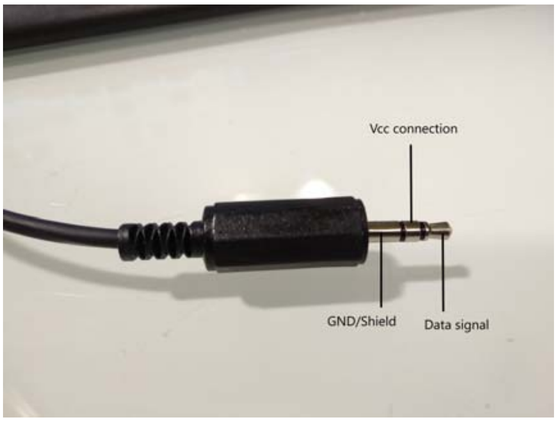


Figure 9.12: Mini-jack connector

So, in order to be able to use the clamps, we are going to use a female mini-jack connector and connect it to another male mini-jack connector changing the order of the wires. Then, to use one clamp there will be needed one female and one male mini-jack and also a mini-jack wire. The first step will be welding the wire to the female connector.

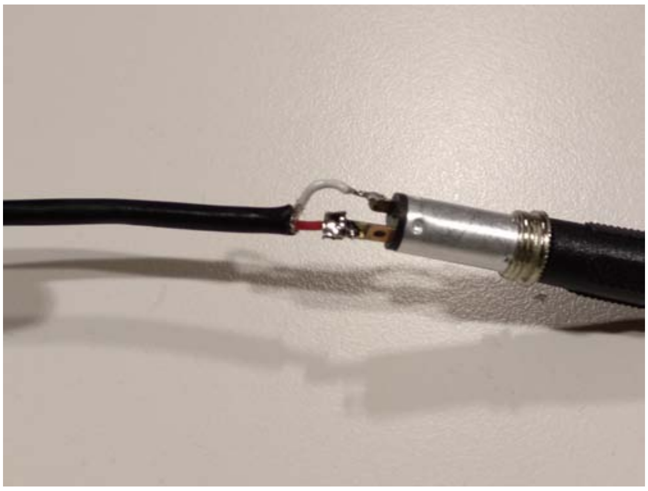


Figure 9.13: Welded connector

As it can be seen in the image, the red wire is going to be connected to the biggest flange (this wire will be carrying the data signal) and the white wire is connected to the little and round flange (this wire will be carrying the Vcc signal). It will be optional to connect the metallic mail to the copper flange.

On the other side of the wire, it must be welded a male connector following the next image:

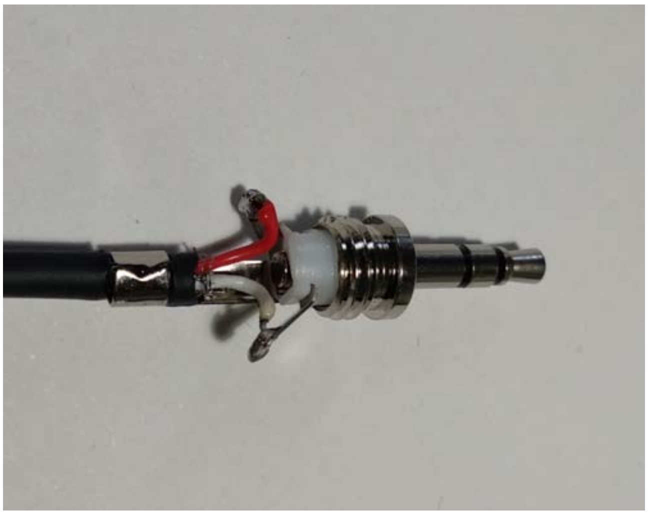


Figure 9.14: Welded connector

Then, the metallic mail should be connected to the exterior flange, the red wire to the interior flange and the white wire to the middle flange.

After this process, the clamp has to be connected to the female connector and the male connector must be connected to the multipurpose board.

9.2.3 Troubleshooting

To know if the clamps are correctly welded, a multimeter will be needed. There must be checked if there is some voltage between the data signal and the current signal. Also is interesting to check is some of the signals are or not welded together, in other words, it is interesting to check that the signals aren't crossed between them. It is also interesting to notice that the core of the clamps is quite fragile and it will break upon receiving an impact, so they should be treated carefully. Finally, it should be also noticed that depending on how the clamps are connected, negative measures can appear when monitoring power. This problem is solved by simply flipping the clamp, therefore inverting the polarity of its magnetic field in relation to the current it is monitoring.

9.3 Usage of the current clamps for Atentto r04.5 2018

The clamps work the same way with the new multipurpose, the only difference is that the new multipurpose board is already prepared for the clamps mini-jack configuration, so in this case the clamps do not need to be modified, they are already prepared to connect and use.

9.4 Usage of the magnetic sensors

There had also been implemented magnetic sensors for Atentto. These are useful, for example, to know when a determined door opens or closes.
The magnetic sensors which are going to be used with Atentto are “normally closed”, this means that when the magnets are near each other the circuit closes, and when the magnets are far from each other the circuit is open (there is not current circulating). Translating this to the web page language, when the circuit is opened the door is opened and when the circuit is closed the door is closed In order to connect these sensors with Atentto, there are going to be used the digital ports (D41 to D50) and the RJ11 protocol, working similarly to the OneWire probes, with the difference that, in this case, only one sensor can be connected to each of the ports.

9.4.1 Making a sensor

The magnetic sensors only have two wires, and the RJ11 has four wires, so we are going to cut the yellow and red cables and leave the other two connected.

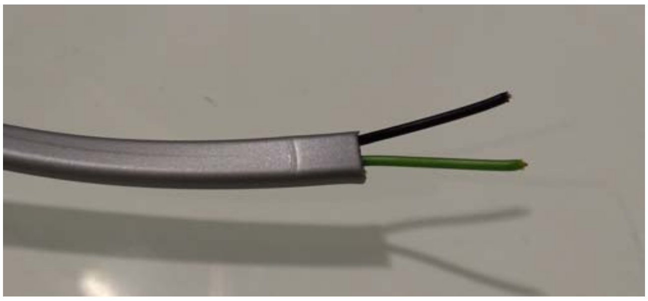


Figure 9.15: RJ11 Wire

Then, we must weld the GND wire (the black one) with one of the sensor wires and the data signal cable (the green one) with the other wire. It is indistinct which one of the wires is connected to the data or the GND. After welding the wires both of them must be protected.

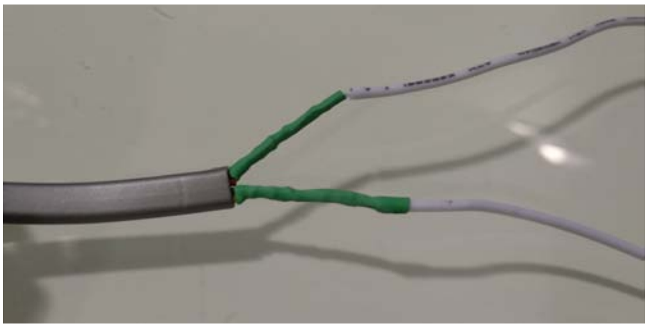


Figure 9.16: Welded sensor

9.5 How to use flow meters & water counters

In order to measure water flow and there are going to be used some flow meters. Those meters work the following way: the water (or fluid in general) passes through the meter, which has an helix inside. Then, the helix rotates when the water passes through it, and, every time that the helix rotates a certain angle, the sensor sends and interruption through the digital port. Finally, Atentto sums these interruptions over a certain time and obtains the water flow. The flow meters are going to be connected in the digital ports with an RJ11 wire, using the same protocol as the one wire probes. The flow meters will only be able to be connected from the port D46 to the port D50.

9.5.1 Making a sensor

First of all, the flow meters have their own connector but it is needed to connect them via an RJ11 wire, so their connector must be cut off. Then, the sensor has three wires, so there are going to used only three wires from the RJ11 wire: the green, the black and the red one.

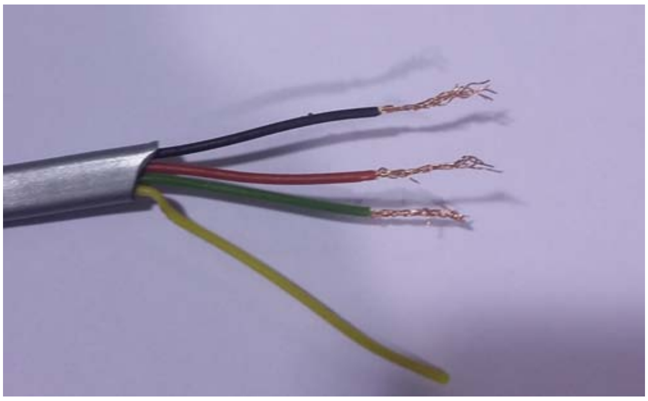


Figure 9.17: RJ11 Wire

Then, the sensor yellow wire must be welded to the green RJ11 wire (for the data signal), the sensor black wire must be welded to the black RJ11 wire (for the GND connection) and the sensor red wire must be welded to the red RJ11 wire (for the Vcc connection). After welding each wire must be protected.

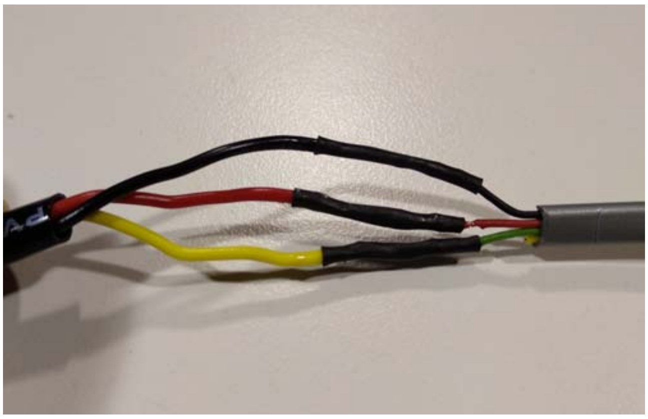


Figure 9.18: Welded sensor

Finally, there must be place an overall protection in the welding zone.

9.5.2 Water counters

The water counters work exactly the same as the flow meters, the only difference is that they should be saved as “water counters” in the web page and they measure the total amount of water which has gone through the meter in a day. It is interesting to notice that a flow meter and a water counter can be linked to the same physical sensor in the web page, obtaining the water flow and the total amount of water at the same time.

9.6 How to burn the code in Atentto

If you want to settle a new Atentto, or modified the software inside an older one, you will need to have installed **Visual Studio** (versions above 2013 are valid), with **Visual Micro**. That is an arduino tool for Visual Studio, and it can be downloaded freely from their web page. It is also necessary to have the two main folders explained in the “HTTP client” section, on page 68, **“Conectto-1.5.8”** and the one which contains **“Atentto.ino”**, in your personal folder.

First of all, you have to open **“Atentto.sln”**, located in the same folder that contains “Atentto.ino”, with Visual Studio. After that, the Visual Studio Solution, with all the Atentto’s software, should appear in the VS interface, as shown in the next image.

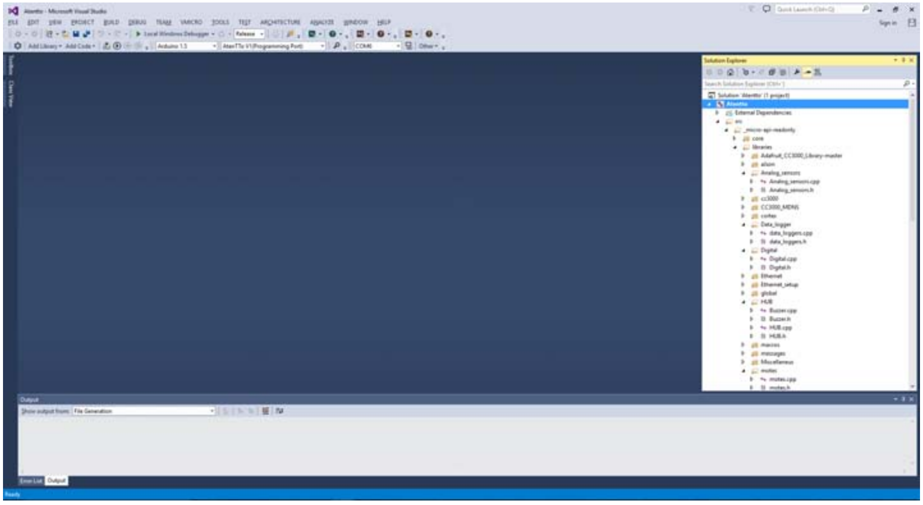


Figure 9.19: Interface of Visual Studio 2013 with the Atentto's solution loaded

Then, the first to do is configure Visual Micro to use the libraries from your Conectto folder and not the one located in Software. In order to do this, you should open the scrollable menu where you can select the different IDEs and then click in "Configure IDE locations". A new window will appear, there you should copy the path of your Conectto folder in the first dialogue box.

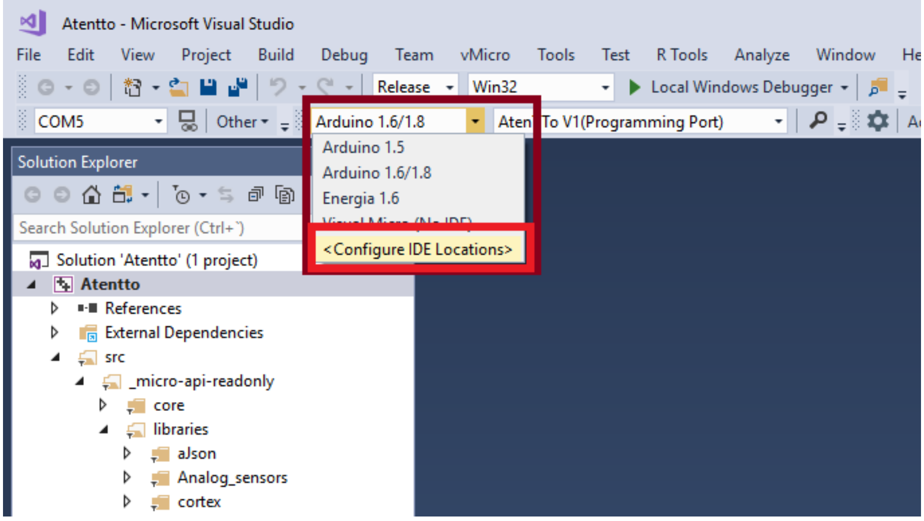


Figure 9.20: Configuring IDE locations

From there, you can navigate through all the files, and modify them, but as have been said previously, you only should modify "webpage_server.cpp" and "Atentto.ino". When you want to test a new version, you must connect the Atentto via USB, with an **USB to Micro-USB cable**, allowing serial communication between computer and Atentto. If you are debugging, it is also quite recommendable to connect an Ethernet cable to the Atentto, in order to see the web page functioning with every change, without having to reconnect the Atentto to the multipurpose board.

Now you can see what Atentto is sending via serial communication (useful also for debugging), and burn new code in the Atentto. For this, you need to have installed the **Arduino IDE** (it is essential). This IDE will add a new menu to the VS interface, just below the main tool bar at the top of the program. After configuring it as shown in the next image, you will be able to burn the code in any Atentto.

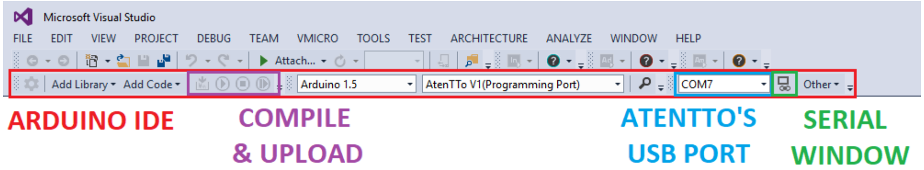


Figure 9.21: Arduino IDE with the configuration used to burn the code in the Atentto

Chapter 10

Hardware Components

This section aims to give a **practical description of the new components** included in Multipurpose and those that has suffered modifications and improvements. Since we count with a great deal of information about some of the components, others like AMC1100, the transformer or the new risers used for plugging AtenTTo Core are new and need some explanations.

10.1 AMC1100

The circuit around the component AMC1100 is included in the board in order to **sense the input AC mains voltage** that supplies power to the board. It have to accomplish with requirements of **galvanic isolation** to ensure the user safety and the signal integrity (it is an alternative to transformers). If we have connected our Multipurpose to a electric network where machines are also connected we can know the power that the specific line, machine or system is consuming using amperometric clamps connected to the line involved and monitoring the voltage network with the circuit mounted here, using just one analog signal (A0). The sampling of this sensor give us the instant values of the voltage and the data can be used to calculate power, Fourier transforms of the curves or study the quality of the network used.

The operation and use are explained in the Application Note "*AMC1100: Replacement of Input Main Sensing Transformer in Inverters with Isolated Amplifier*" created by *Texas Instruments* and included in the Appendix **??** of this project. We are interested in the AC Voltage Sensing through AMC1100 part of the App. Note, which describes the circuit and design guidelines for mounting the sensor in a board. AMC1100 [?] is an isolation amplifier which can provide galvanic isolation for 4000 VPEAK voltages between the input and the output using a silicon dioxide (SiO2) barrier highly resistant to magnetic interference. It is essential to power both sides of the IC using also isolated power sources and that is accomplished in our design thanks to the transformer selected for the powering of the system (10.2) and the distances reserved between both sides of the board (high and low voltage).

Figure 10.1 shows the **schematic of the circuit** used in Multipurpose which is similar to the proposed system in the Application Note. From the AC mains input, there is a capacitive power supply (or capacitive dropper) of around $5.1V$ (in VDD) formed by R42, C7, D11 and D12. It means that the reactance of the capacitor (opposition of the component to a change in voltage) is used to reduce the mains voltage to a lower value (capacitor must be very resistant to voltage and have as much capacitance as current needed, due to that it is used in low power applications). With the help of zener diode D11 and D12, the power supply of $5V$ is achieved. D12 diode is a model M7, an SMD version of the 1N4007 general purpose diodes used in AC adapters for example. It is characterised by admitting a high voltage (1000V) and 1A current (in DO-214AC encapsulation).

The following voltage divider formed by R43 and R44 generates a voltage level of $2.5V = \frac{VDD}{2}$ in the pin VINN (inverting analog input) of the AMC1100 IC. At the same time there is a voltage divider formed by R45, R46, R47 and R48 that adapts the mains voltage for VINP and VINN. AMC110 admits between these pins a maximum of $\pm 250mV$ so we have to select the resistors values considering the input we want to measure and the limit, not exceeding them. We have used the recommended values of the Application Note since they measure a similar range as the one we are interested in (between 90VAC and 300VAC). The Peak voltage at positive pin (VINP), solving the voltage divider and assuming we have an input of 220VAC with a peak value of $\sqrt{2} \cdot 220V = 311V$, is:

$$V_{VINP} = \frac{1k\Omega}{2M\Omega + 220k\Omega} \cdot (\pm 311V) = \pm 140mV$$

which is under the maximum allowed. In addition, C10 is used as a decoupler capacitor in order to reduce noise in the power supply of the integrated circuit.

On the low voltage side of the AMC1100 the output is measured in VOUTP pin with the reference in GND. Since it is powered with $+5V$ the output is centred in $2.55V$ and oscillates in a range of $\pm 1V$ if the maximum range is applied in the input ($\pm 250mV$). If $3.3V$ or $3V$ are used, the signal is automatically centred in $1.29V$. We are using $\pm 140mV$ in the input which involves an output of around $\pm 0.56V$ centred in $2.55V$. We will have the maximum $3.11V$ and the minimum in $1.99V$ so we are loosing part of the available range to measure in the ADC (from $0V$ to $3.3V$). As we will see, the test developed gave better results when using $5V$ powering than $3V$ since this case deliver a distorted wave when a sine is expected.

A clarification must be done here, the amplifier we are using has a nominal gain of 8, this means that the output voltage is eight times the input voltage but we have said that we expect an output of $\pm 1V$ when $\pm 250mV$ is applied, four times higher. This happens because we are measuring between the VOUTP and GND while the gain is achieved between VOUTN and VOUTP, since VOUTN is showing the inverted signal, the voltage difference between both is the double of the one we are measuring. Figure 10.2 helps to understand this, the green line is the VOUTP while the red one is VOUTN, non-inverting and inverting analog outputs. Since we only measure the non-inverting output with the reference in GND and we power the device with $5V$ we can expect a gain 4 with a signal centred in $2.55V$.

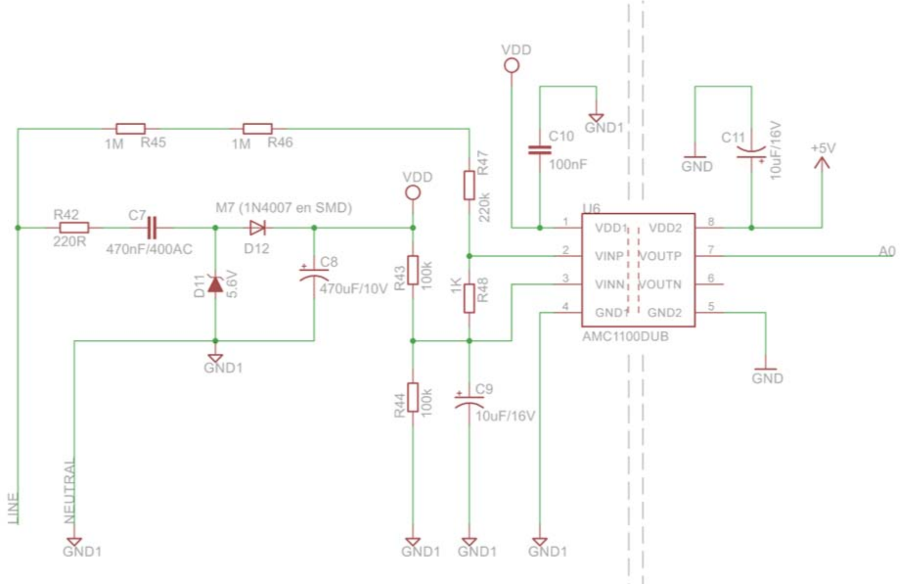


Figure 10.1: Schematic of the isolated sensor used in Multipurpose board in order to measure the voltage of the electric network.

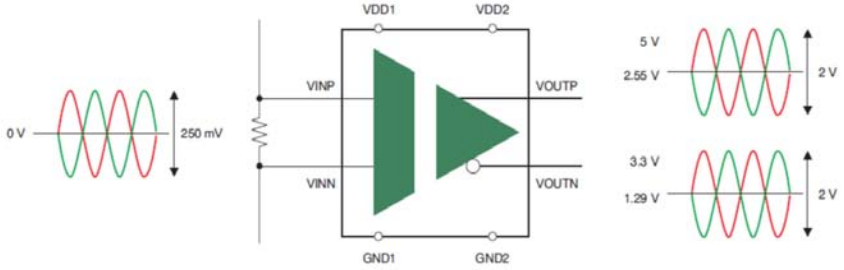


Figure 10.2: Scheme of the input and output of the amplifier used (*AMC1100*).

First test done was mounting the circuit in a protoboard, connect it and read voltage in the output. The oscilloscope shown a curve centred in $2.57V$ and with a maximum of $3.10V$ and a minimum of $2.02V$ really near the expected value. At the same time the voltage in the voltage divider is measured with a multimeter Agilent and the result is $92mVAC$ which means $130mV$ of peak value while we predicted $140mV$. The same test was repeated with $3.3V$ in the low voltage size and the output was plotted with the oscilloscope, the curve obtained had a lot of interferences so we take the decision of powering AMC1100 with $5V$ in the board.

The previous circuit was then connected to an AtenTTo Core using the analog signal A0 (54 software label), sharing the reference (GND) in the low voltage part of the AMC and powering the device with the same $5V$ source. In that situation we just need to upload a program to AtenTTo for reading the specific analog signal at a high speed, store the values and print them in the Serial. The sampling speed is important since we try to measure a wave of 50Hz. The code uploaded was the shown in the code 10.1. The results were plotted and a *Sine* curve was overlapped in order to compare the measurement with the expected values for a network voltage. The results were really satisfactory since both curves were similar as can be seen in the Figure 10.3. The Figure 10.4 shows the same experiment done with a board already manufactured, in this case just uploading a program to an AtenTTo Core which is connected to Multipurpose and power both devices with the network of the office. While the test in protoboard used a measurement converted to voltage, in the case of the board manufactured we only plotted the sampling values of the ADC which are proportional to the voltage.

```
int analogPin = 54;     // potentiometer wiper (middle terminal)
    connected to analog pin 3
// outside leads to ground and +5V
int val = 0;           // variable to store the value read


void setup()
{

Serial.begin(115200);          //  setup serial

}


void loop()
{

val = analogRead(analogPin);    // read the input pin
Serial.println(val);             // debug value

}
```

Listing 10.1: Program uploaded in AtenTTo Core in order to measure the A0 analog pin and capture the form of the mains voltage wave.

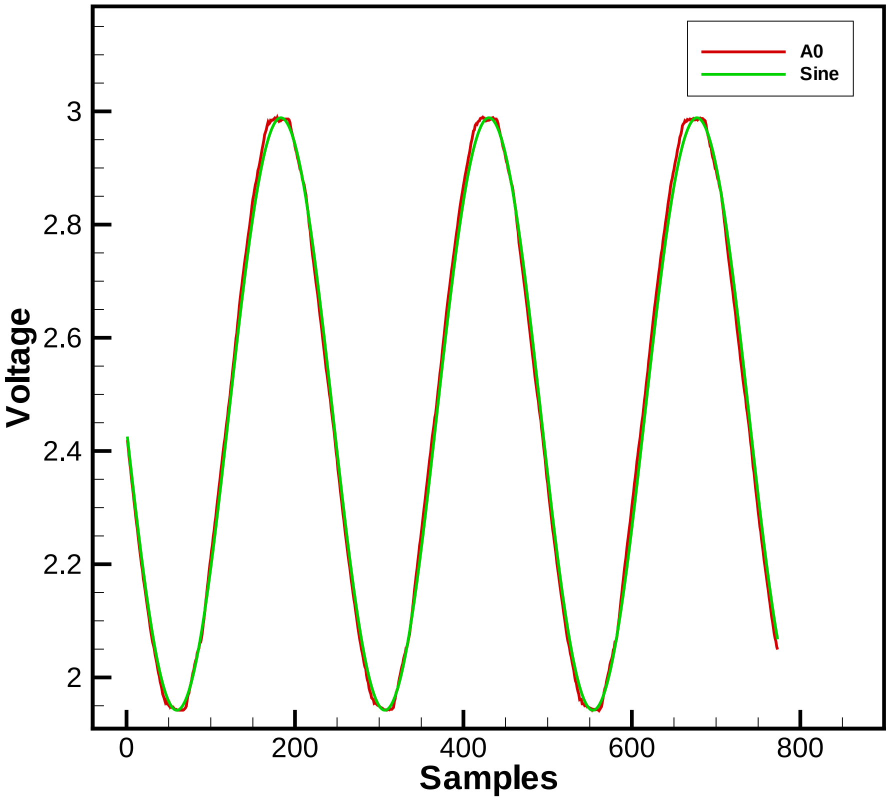


Figure 10.3: Voltage measurement obtained with the ADC0 (in red) compared with a sine curve overlapped (in green) depending on the number of samples acquired; this proof of concept of the AMC1100 circuit made in a protoboard demonstrate the advantages of using this kind of system.

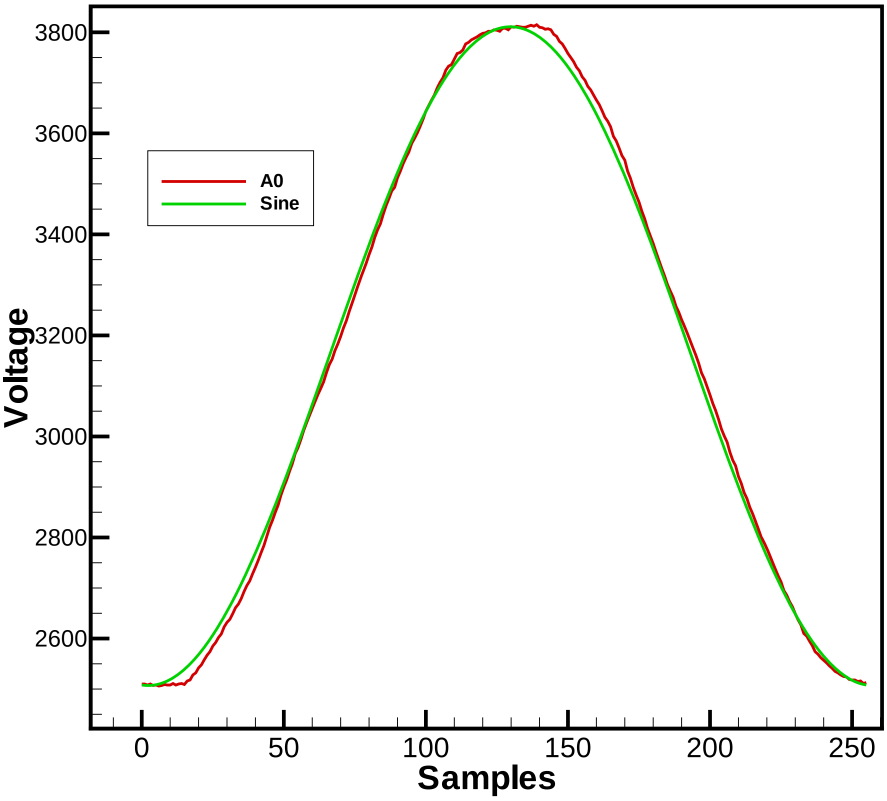


Figure 10.4: Voltage measured with A0 (expressed with discrete levels instead of volts) and compared with a sine curve overlapped (in green); this test is done with the board already manufactured (the centre is in the value 3200 which in a 12bits-ADC and using $3.3V$ is equal to $2.57V$).

A possible future improvement to be made could be choosing different resistors for the voltage divider and select it for measuring the peak value of the $220VAC$ in $3.3V$ in order to benefit from all the range of the ADC. A better idea could be to solve the malfunctioning when using $3.3V$ for powering the AMC1100 because then we would have the output signal centred in $1.29V$ and we could use the maximum output range of $\pm 1V$ or even study how to include in the measurement the inverting pin and have a gain of 8. For that purpose we could need to change the reference of the ADC temporarily.

Regarding the components chosen, the only problem appear when selecting the capacitor C7 since the voltage that supports is limited in all the models found. At the end the chosen one was *2220Y5000474KXT* from *Syfer Technology*, a ceramic multilayer capacitor (MLCC) SMD of 470nF that supports 500V with a size of $5.7x5x4.2mm$. In relation to the routing it is only necessary to remark that the needed change in the position of the transformer in the release r04.5 forces this circuit to be also in a higher position and a little bit comprised but maintaining the distances between low and high voltages and the clearance between lines of high voltage.

10.2 Power Supply

The problematic of the power source in AtenTTo Multipurpose made necessary to study some different options, from **tiny transformers** like those included in the phone chargers to **transformers-rectifiers** from different manufacturers (Myrra, Block, Traco Power). Multipurpose needs at the same time a power source for the whole system and a way of measuring the voltage of the network where it is connected.

The most important condition is that the **consumption of the system must not change or influence the measurement since we would not analyse the real voltage wave of the network**. This condition made difficult to use the same transformer for both functions, the load applied to the low voltage part would modify the behaviour of the transformer. At the same time: **price, size** and **isolation condition** have to be checked. First option, including a transformer and a power source like a transformer-rectifier in the same board is not possible because a lack of space since we could not find very tiny transformers. Second option, using a voltage divider for the measuring process is not permitted because the board would not be isolated from the high voltage part with the associated danger for the user of the board. A third option was using a transformer with two isolated outputs (*Block* for example provides a $220V - AC \quad to \quad 24V - AC/25V - AC$ or even $220V - AC \quad to \quad 24V - AC/49V - AC$), however, it forces to include the circuit for rectifying the current and again there is not enough space. Thus, we decided to use AMC1100 (see 10.1) for the measuring objective so we totally separate both functions and reduce the influence of one on the other.

The **transformer-rectifier** is the best option for powering the device since it includes in the same circuit the reduction of voltage and the AC-DC voltage conversion. The next step is deciding the available voltage levels in the board, Multipurpose can be connected to a lot of different devices and all of them work with a different voltage. The essential voltages are **+5V** for the relays and Core (AtenTTo Core will generate its own $+3.3V$), **+3.3V** for optocouplers and

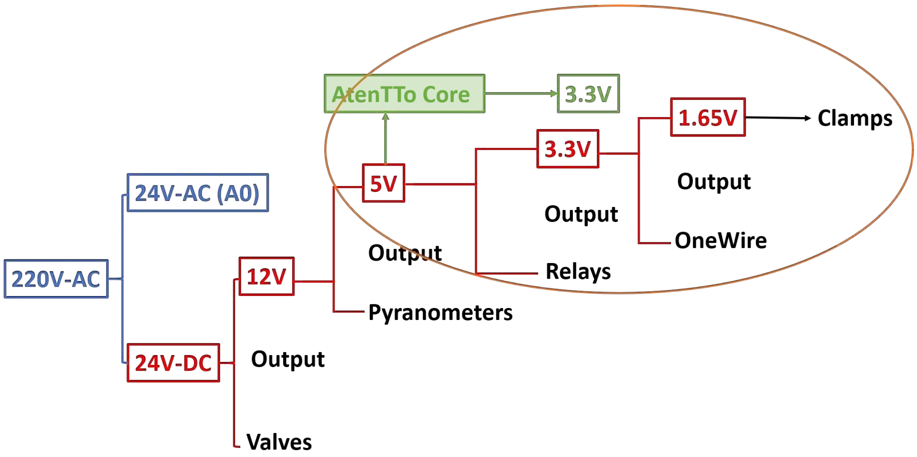


Figure 10.5: Scheme of the different voltage levels that can be useful in Multipurpose board, in orange the available options in current release. "Output" involves that a connector is situated for providing the voltage from outside the board.

OneWire probes and **+1.65V** for the amperometric clamps. The rest of the voltages are not mandatory but including them can be useful for the peripheral devices (solenoid valves, pyranometers, etc). The Figure 10.5 shows the scheme with the voltage levels that initially was decided to include in the board and in **orange the final decision**. $24V-DC$ and $12V-DC$ were discarded because the necessity of successive voltage regulators (space associated) and because the peripherals that need that voltage are only implemented in some cases and not in all the Multipurpose boards to be installed (solenoid valves for irrigation systems and pyranometers when lighting conditions monitoring is needed). An external power source can be installed besides Multipurpose (same DIN rail) in those cases.

In order to decide the specific model of transformer-rectifier we should have an **estimation of the current demanded** by the different components inside Multipurpose, the Core board and the external peripherals directly dependent from this power source. A quick estimation was made since it is not real that all the components are working and demanding the maximum current at the same time. However, the values shown in table 10.1 help to have an order of magnitude of the total power demanded in an extreme case. It is necessary to separate the currents that flow at $+5V$ and those that are delivered at $+3.3V$ because the power that involves is different.

That current provided with the correspondent voltage involves $\approx 1372.8 + 2000 = 3372.8mW$. Hence, the transformer chosen was "47152" from *Myrra* (see Figure 1.2). The essential properties of our power source are shown in table 10.2. It has to be noticed that the maximum current that can be provided is not only **limited by this transformer**, it is also limited in the voltage $3.3V$ for the **three voltage regulator** that the system has, one in Multipurpose and other in Core since they do not share the $3.3V$. The component in Multipurpose is *AP7333-33SAG-7* from the company *DiodesZetex* and admits a maximum output current of **300mA**. AtenTTo Core has at the same time two regulators (IC8 and IC9) of the same model *NCP1117ST33T3G* from *ON Semiconductor*

Component	**Current demanded at** $+5V$
x10 Relays	400mA
TOTAL	400mA
Component	**Current demanded at** $+3.3V$
Core Microcontroller	80mA
Radio	60mA
x100 OneWire	$100x(1 - 1.5mA) \sim 130mA$
Ethernet	140mA
x10 Clamps	No Consumption!
x4 Optocouplers	No Consumption!
x3 Pyranometers	$\sim$ 6mA
Buzzer	1mA
x4 PWM lines	$\sim$nA
TOTAL	$\approx$ 416mA

Table 10.1: Estimated consumption for the different components and devices dependent from the Multipurpose power source and the different voltage levels.

and they can provide each **1A**. It is interesting to remember when the current is provided by the Core and when by the Multipurpose since the current in $3.3V$ in the Multipurpose is more limited.

Component	**Current demanded**
Main Input Voltage	100 up to 240VAC
Input Voltage Limits	85 up to 265VAC
Frequency	47 up to 440Hz
Output Voltage	5VDC
Power	4.5W
Efficiency	68%
Output Voltage Accuracy (100% load)	±2%
Line Output Voltage Variation	±3%
Load Output Voltage Variation	±0.5%
No Load Input Power	$< 200mW$
Size	$31.9x26.9x21.8mm$

Table 10.2: Main properties of the Power source chosen for Multipurpose, "47152" from *Myrra*.

In order to check that the power source is enough and no problems appear due to transient peaks in the relays a **proof of concept is made**. A simulation of Multipurpose is connected to the source chosen. The $+5V$ output is connected to an AtenTTo Core totally operative that is controlling by software one relay, switching between open and close state. At the same time more relays are simulated with a constant charge of $322mA$ and a voltage regulator from $5V$ to $3.3V$ supplies current for a LED and an extra charge of $50mA$ (with a resistor of 62Ω). The power source is totally capable of feeding the system and no transient peaks appear in the rest of the components when the relay is switching. At the same time, the transformer is not warm up drastically so this power source is finally chosen for Multipurpose.

10.3 Optocouplers

While the relays can open and close a line where a voltage is applied (like a switch) and let the current flow, the optocouplers are used for checking if **two terminals have a voltage difference or not**, this can be really useful to monitor if the circuits of a house or office have suffered a power outage. In a house there are at least four or five independent circuits: the lighting circuit, internal circuit for plugs, internal distribution circuit for oven and rest of the kitchen, the circuit for bathrooms, etc. We decided to implement four optocouplers in Multipurpose (see Figure 1.6) in order to connect each of these circuits to the board and measure the voltage condition all the time, also because it is said that we can only have five circuit breakers for each differential switch (considering we just have one in our home we can connect four of the circuits in our house). Using four digital signals we can find a power outage in one line and warn the user via Smartphone or email. It is really useful for example when really important systems for the house are connected to those lines.

We are not going to deepen in this topic since the optocouplers have already been implemented in previous releases of Multipurpose and its functioning is controlled. However, it is interesting to make a quick review of how it works. Figure 10.6 shows the schematic of the circuit used for an optocoupler. Our optocoupler (Kingbright KB184-B) is composed by two LEDs and a phototransistor inside an encapsulation, when one LED is on, the light changes the state of the transistor and a signal is transmitted. The essential property of this component is insulation between the two parts of the circuit (which is added to the electrical insulation applied to the ground planes and the rest of the lines). We need to insulate both parts since we do not want that the high voltage interferes with the digital circuit of the board. With this device the only communication between high and low voltage is the light of the LED.

The need of two LEDs answer to the input signal applied, it is AC voltage so the change in the voltage have to be considered and light have to appear when current flows in both directions. The device admits 1.2V and $\pm 50mA$ in the input so in order to adequate 220V-AC to this input we include the resistor R13 of the figure. If the AC voltage is applied at the input, the transistor is switching between on and off state and the capacitor is charged through R11 resistor because the transistor lets the current flow (R12 must be higher than R11 in order to let the capacitor to be charged). A high voltage level (lower than $3.3V$) is generated in the digital signal. When the input disappear, the transistor goes to off state and the capacitor is discharged through R12 resistor. The digital signal reads then a low level and an alarm can be programmed in order to warn the user of a malfunctioning in that line.

There is no differences in the circuit implemented in both releases (r04 and r04.5). However, due to the need of moving the transformer it was necessary to compress the design. It was done with a change in the orientation of the capacitors but maintaining the mandatory distances between the high voltage part and the digital circuitry.

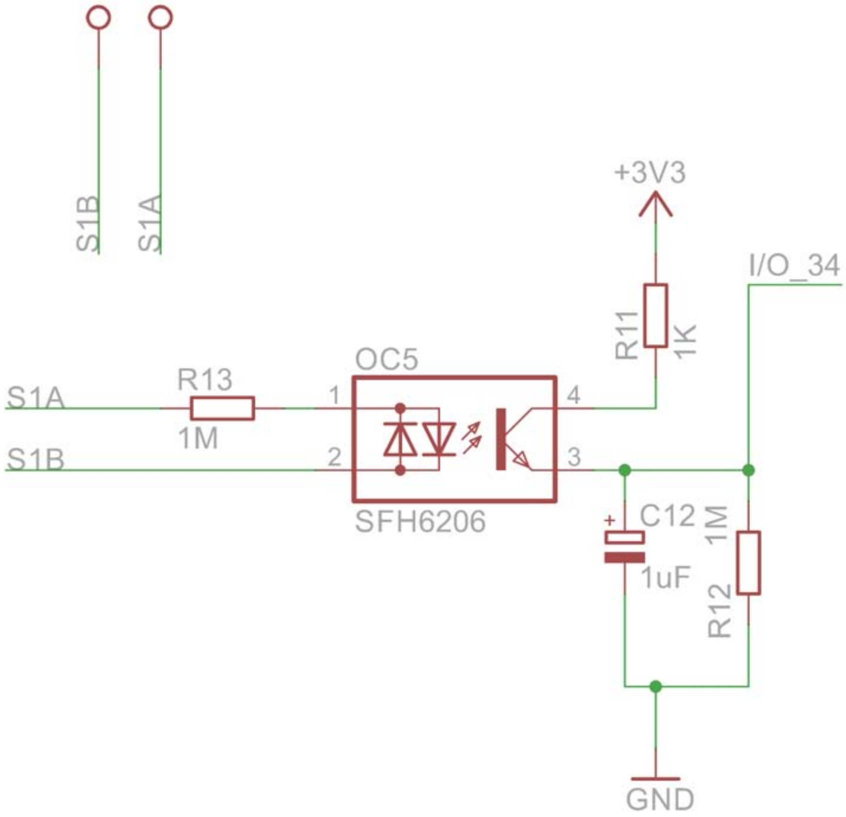


Figure 10.6: Schematic of the circuit used for the four Optocouplers in Multipurpose board.

Important Notice

This circuit can only be used for AC measuring as it is said in [**?**].

10.4 Analog Signals, mini-jacks

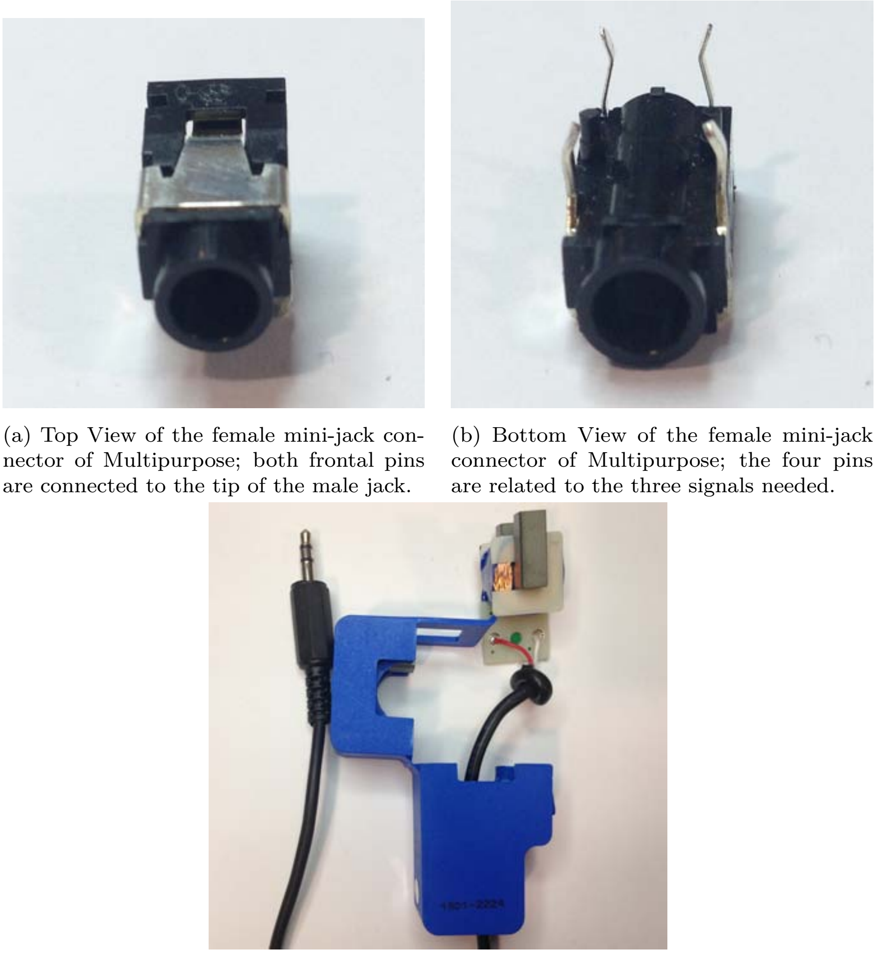
Ten of the analog signals available in the MCU of AtenTTo Core are connected in Multipurpose to ten female mini-jack connectors. These plugs are mounted in order to connect mainly amperometric clamps but we can manufacture mini-jack connector (also called audio jack) for different sensors. There are more companies in the industry that mount amperometric clamps with this standard connexion.

The **amperometric clamps** that we use are *SCT-013-030* manufactured by *YHDC*. The split-core transformer generates a voltage difference (called V_{out}) in the two terminals of a resistor, this voltage is proportional to the current that is passing through the core so we need to obtain and transform that voltage signal using analog pins of our microcontroller. One of the two terminals provided in the clamps is the reference while the other one is the variable voltage to measure, those two are connected to our board in the female mini-jack connectors (see Figure 10.7). The third connexion is used sometimes for the jacket of the wire that connects the mini-jack with the clamp, this jacket (or net) is connected to the real GND reference of the board and helps reducing the noise in the signal. The real reference for the measurement is connected to the voltage $+1.65V$ of Multipurpose board, so the wave generated in the second terminal (with positive and negative values compared with the reference) is translated into positive values for Multipurpose (if are compared with GND, the real reference of the board).

In conclusion, AtenTTo Core receives a variable voltage signal to be measured with an ADC and this information will be translated into current. The three pins of the male mini-jack connector are connected to the two needed terminals of the clamp and optionally to the jacket of the cable. Regarding the female connector, it has four pins (two of them connected) and three terminals again. One will be connected to GND (for the jacket), other to the reference voltage ($+1.65V$) and the last one to an specific analog signal.

In the Figure 10.8 we can see the relation between the footprint of the female mini-jack (top view) and the specific sector in the male mini-jack (tip, ring and sleeve, TRS), this relation does not change while we are using the same connector in all the releases of Multipurpose. At the same time we can see the relation between the sector and the pin to be soldered in the male jack, this is useful for joining the clamps with the connectors (this connexion is already done by the manufacturer). There are three different versions of the connexions made in each of the pins, depending on the different releases of Multipurpose, we here summarise all of them and explain the changes.

The change between r03 and r04 (Figures 10.9a and 10.9b) was made because of the idea that the sleeve of the mini-jack connector should have GND while the tip is the signal (see board plane of release r04 **??**). This was thought trying to copy the configuration of common audio jacks used for phones and computers. Since then, some clamps found in the industry has shown that another protocol



(a) Top View of the female mini-jack connector of Multipurpose; both frontal pins are connected to the tip of the male jack.

(b) Bottom View of the female mini-jack connector of Multipurpose; the four pins are related to the three signals needed.

(c) Clamp used for current measuring, the standard mini-jack connector and the red an white wires are already connected.

Figure 10.7: System needed for current measuring, female and male connectors and the amperometric clamp.

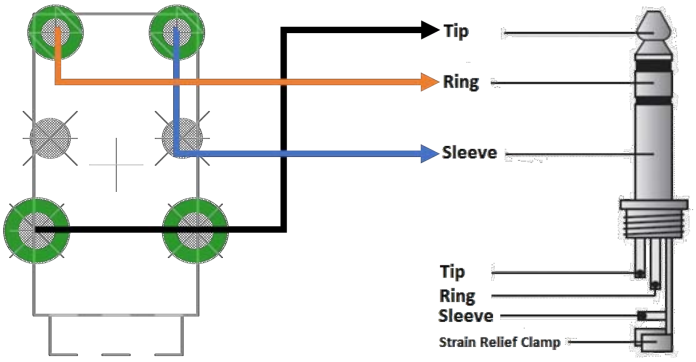
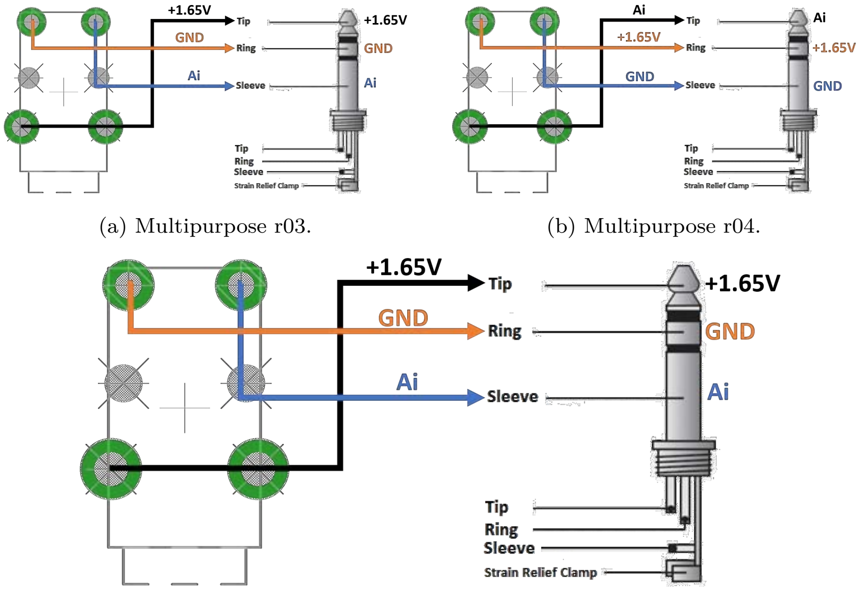


Figure 10.8: Relation between the female connector used in Multipurpose and the male mini-jack.



(a) Multipurpose r03.

(b) Multipurpose r04.

(c) Multipurpose r04.5.

Figure 10.9: Relation between the pins of the female mini-jack connector with the male jack and the signals connected, for three releases of Multipurpose.

is an standard, the sleeve with the signal and the tip with the reference of the clamp is used by different companies. The configuration mounted in the release r04.5 is the final alternative and copies the rest of the devices in the industry (see board plane **??** and Figure 10.9c).

A **problem** appeared when release r04 was manufactured. The female connectors, due to a curve surface in the bottom (part that is supported in the board) does not stay horizontal when mounting it, some of them turns to one or the other side. All the problematic is explained in section **??**, the solution applied to release r04.5 was reducing the size of the holes where the connector pins are soldered. This solution (recommended by the manufacturers) is made in order to grab firmly the component before soldering and avoid the inclination. The size of the holes is reduced from **0.8mm to 0.7mm** in the tip pins and from **0.7mm to 0.55mm** in the other two pins (ring and sleeve).

Regarding the clamps, the two mandatory cables to connect are a red one and a white one. The GND reference can be connected to the jacket of the wire if we want but the other two wires are essential. The red one carries the voltage signal to measure so it is connected to the bigger flange of the male jack, the white one is the reference of the clamp and must be connected to $+1.65V$ in order to rise the reference to that value. This means that the white cable will be connected to the little flange. It happened that lots of clamps were already soldered to be used for release r04 but the solution is using a female-male connexion in between in order to avoid opening all that clamps and change the connexion. However, the new clamps have the configuration used in last release so no changes have to be done if have recently obtain them.

10.5 Analog Amplifier

Multipurpose design includes an **operational amplifier** (U2) in order to generate the voltage reference of **+1.65V**. This reference is an essential function in the board since it is in charge of rising the signal acquired by the amperometric clamps in order to obtain always positive values in the electrical signals measured (the current measured in the clamps will give sine waves with negative values if we use our GND reference). The component mounted is **LM2904DR**, a precise SMD amplifier with encapsulated SOIC-8 which admits 12V, 15V, 18V, 24V, 5V or 9V as voltage source.

We found that this component has two independent amplifiers that can be used at the same time so we decided to benefit from the other one and use it as a **signal amplifier** for one of the analogical signals of AtenTTo Core. This means that the signal connected there (A2) will be amplified and the measuring range will have lower values.

The **above circuit** of Figure 10.10 is the voltage reference, it is constructed using a voltage divider and the operational amplifier in voltage follower mode. While the voltage divider formed by two equal resistors of $100K\Omega$ achieves $\frac{3.3V}{2} = 1.65V$, the voltage follower is used as an isolation amplifier. The operational amplifiers that works as voltage follower (also called unity gain amplifier) achieves that the output voltage is the same as the input with the property that has a very high input impedance so they do not extracts much current in the inputs. At the same time they have very low output impedance. If we use only a voltage divider we will find that the V_{OUT} would vary when the charge applied to the voltage vary a little, this would be a problem because our reference would change. Adding the op.amp. we achieve to isolate both parts of the circuit so the V_{OUT} changes with the resistors of the divider (fixed in the circuit) but not with the charged applied. In conclusion, the circuit maintains the voltage level independently of the demanded current in the circuit associated to $+1.65V$.

The **circuit below** in the figure, second amplifier, is configured as a closed-loop amplifier. The negative feedback is used, we apply a part of the output voltage to the inverting input. In the non-inverting amplifier the presence of a negative feedback via the voltage divider with resistors R50 and R49 ($31.6K\Omega$ and $1K\Omega$ respectively) determines the closed-loop gain:

$$A_{CL} = \frac{V_{out}}{V_{in}} = 1 + \frac{31.6}{1} = 32.6$$

This means that the **signal applied to the input of the amplifier will be multiplied by 32.6**. Since our ADC can measure in a range from **0V** (GND) to **+3.3V**, we can use an input from **0V** (GND) to **+0.1V** and we will measure with the resolution available in the ADC module of the MCU a signal in that range.

It has been included a parallel resistor with R49 which is not mounted in the manufacturing. If we solder a TH resistor (see Figure 10.11), we can reduce the value of the equivalent resistor situated there, reduce the denominator of the expression above and increase the value of the gain, achieving a higher amplification of the signal introduced. This is limited by the noise that appears in the amplifier and the signal, if the noise is in the same range we want to measure we could not amplify for those values. As we can check in **??**, there are two extra analogical signals that can be used connecting wires in the screw connectors besides the power input of the board. One of them (A3) is not connected to any amplifier and admits the standard range imposed by the MCU, the other one (A2) has the range mentioned before.

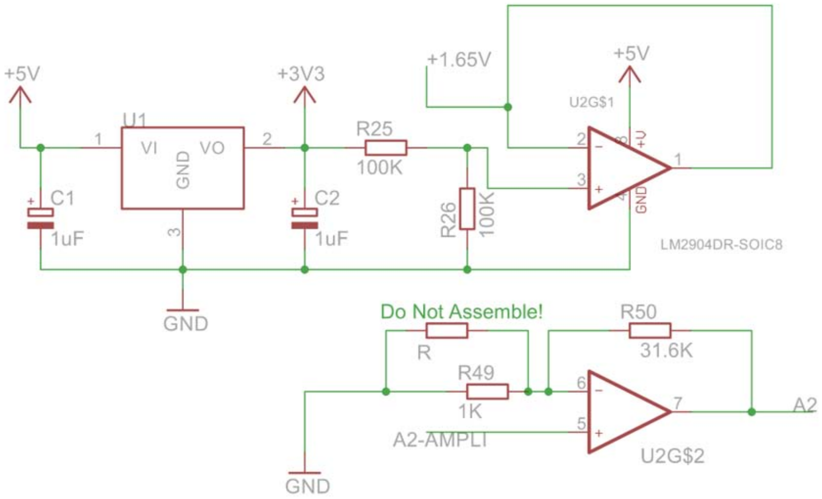


Figure 10.10: Schematic of the circuit that complements *LM2904DR* integrated circuit, two operational amplifiers are exploited.

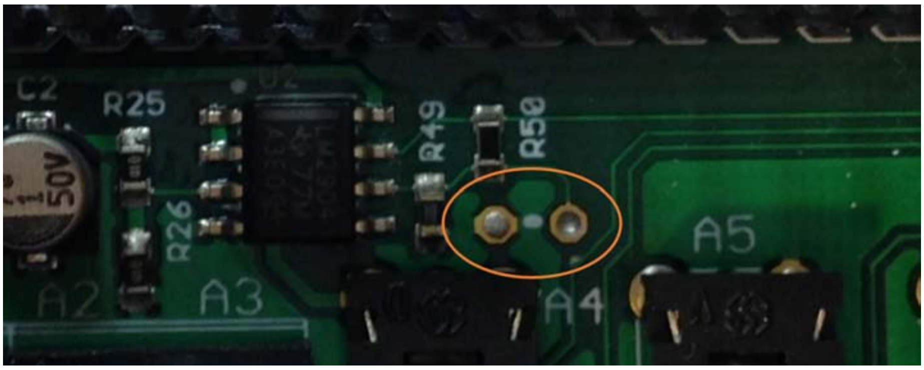


Figure 10.11: Holes reserved for a TH resistor in order to increase the gain of the amplifier and measure lower voltages in A2 analog signal.

10.6 PWM Buses and Digital Outputs

Previous releases of Multipurpose have included a **connector with extra General Input/Output signals**, they can be connected to a lot of different devices and sensors so this connector can be really useful. In release r03 for example the connector was situated in the top left part of the board. Now, with all circuit of AMC1100 it is impossible to include anything there so it has been moved (in both r04 and r04.5) to the right down part, under AtenTTo Core. The only difference in this connector between the two boards presented here is the pin $+5V$ that substitutes a doubled GND pin (in r04 release). This is done in order to benefit from protocol X10, that needs this voltage level.

After re-designing the board and the pins used in the different components (from release r03 to r04), this connector was completed with the signals not used in the rest of the board. Hence: **D22**, **D33**, **D35**, **D37** and **D39** are digital lines included here while **D3**, **D11**, **D12** and **D13** are PWM digital lines included. Finally, $+5V$, $+3.3V$ and GND are the power lines available in the last release (see Figure 10.12). In order to implement a practical use of these functions we want to mount standard male connectors with the wires soldered making a hose so it can be connected directly under the AtenTTo Core board and extend the digital lines (being careful with the space available).

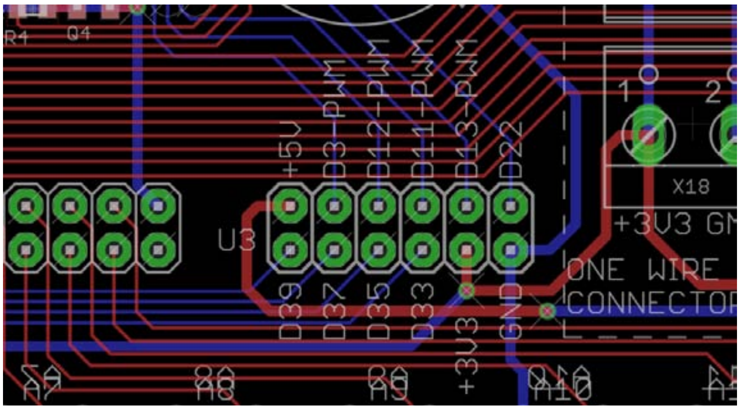


Figure 10.12: EAGLE design screenshot of connector U3 with digital (GPIO) and PWM signals.

PWM signals (Pulse Width Modulation) are the way of generating analog signals with a digital source. The average value of voltage (and current) given to the load is controlled by turning the supply condition between the load and the power source on and off really fast. The longer the switch is active compared to the deactivate periods, the higher the power is supplied to the load. It consists on two components that define how it works: a **duty cycle** and a **frequency**. The duty cycle is the time that the signal is in a high state (as a percentage of the total time). The frequency indicates how fast the PWM completes each cycle or what is the same, how fast it switches between the high and the low (0) state. The result of this will seem to behave like a constant voltage but with a different level than the available digital values (0 or $3.3V$). In conclusion, an analog signal is provided in that line [**?**].

PWM signals are used for a a lot of applications related to **controlling devices**, for example **controlling DC motors**. It can also be used for **valves, pumps or hydraulics and mechanical parts**, actually the idea of reserving this PWM lines in the board came under the necessity of controlling hydraulic pumps in the context of Water Flow Glazing (project where this boards were born). The specific use of the PWM signal (the response time of the application) will decide the frequency to be programmed. In the context of **pump controlling** for example we would have to connect in the socket a hose that extend the lines to a secondary board. In that board a transistor with the correspondent circuit would be mounted so the PWM line changes the state of the transistor and this responds delivering (in the needed ranges) power to the pump.

In the other side of the socket we have the digital lines that can be used (between other functions) for **X10 protocol**. We are not going deeper in this protocol but a brief definition is interesting in order to clearly understand the main functions that Multipurpose can offer. It is a communication protocol for **remote controlling of electric devices through the electric installation** of our house. It uses the pre-existing 220AC circuit and transmit pulses bursts of radio frequency with digital information. The pulses are synchronised with the passage through value zero of the voltage (50Hz or 60Hz), with the presence of a pulse in one cycle and the absence in the next one we have a logical one (1), the opposite represents an logical zero (0).

10.7 Risers

We call **Risers** to the connectors between AtenTTo Core and Multipurpose, they are in charge of transmitting the **power to the Core** while **distributing the signals and lines** in the Multipurpose. The footprint of the components can be seen in **??**, they are named IOA, MISC, IOB and ADC, giving an idea of the signals that are transmitting (General Input/Output, Miscellanea or Analog to Digital Converter).

The connectors have two sides with **male pins** and an intermediate part that rise the board to be connected (see Figure 10.13). Then, two similar rows of female connectors are soldered in AtenTTo Core, where this risers are introduced. It is mandatory to rise the board since there are some obstacles to avoid in Multipurpose board (relays, a buzzer and the transformer mainly), this is why the intermediate part of the component force the Core to stay at an specific heigh (see Figure 10.14). At the same time we can not rise the board the distance we want, the whole system is embedded in an standard box (Figure **??**) with a cover so the upper board must not touch that cover. At the end, a really limited margin is available for situating AtenTTo Core and that is why this component has been difficult to find. Regarding the dimensions in the plane of the board, it is an standard in the industry of PCBs, so there are hundreds of connectors with same sizes. The distances are 2.54mm (0.1inch) between pins in the same row and 2.54mm between both rows.

The needed length of the risers is **14.54mm** between the side that is supported in Multipurpose board and the part where the female connectors of Core are supported (see Figure 10.15), this has been calculated considering the previous conditions. While there are few options for the length of the male pins to connect Core (standard) the length of the male pins soldered in the board is also an standard distance. Hence, the variables to consider when looking for the component are the length of the intermediate part and the number of pins and rows.

In order to find the exact components lots of distributors were consulted (RS, Mouser, Digikey, Farnell, etc). However, the family "AMPMODU headers" of *TE-Connectivity* (the best option found) were not available in any of the distributors. After checking many different manufacturers, the component were found in **Electrónica Embajadores**, a physical store in Madrid, distributor of *Conexcon* components. The final riser selected was the **series 2666** of this manufacturer whose datasheet can be found in **??**. As can be checked in the appendix, the reference number that we ask for when buying the components (each time a new Multipurpose is sent to manufacture) is **2666-3462** and **2666-3202**. The first one, with 46 pins in two rows is soldered in MISC, IOB and ADC connectors while the second one with 20 pins is situated in IOA. While the number $T = 3$ in the reference is not essential (surface finish), the final number $C = 2$ it is the other main parameter because it indicates the lengths of the pins and the intermediate part, for example $C = 2$ means that $D = 14.50mm$ and $H = 8mm$, really near to the needed sizes. If these specific number of pins is not available, we can also ask for the next model available (they are cut to the needed length by the manufacturers).

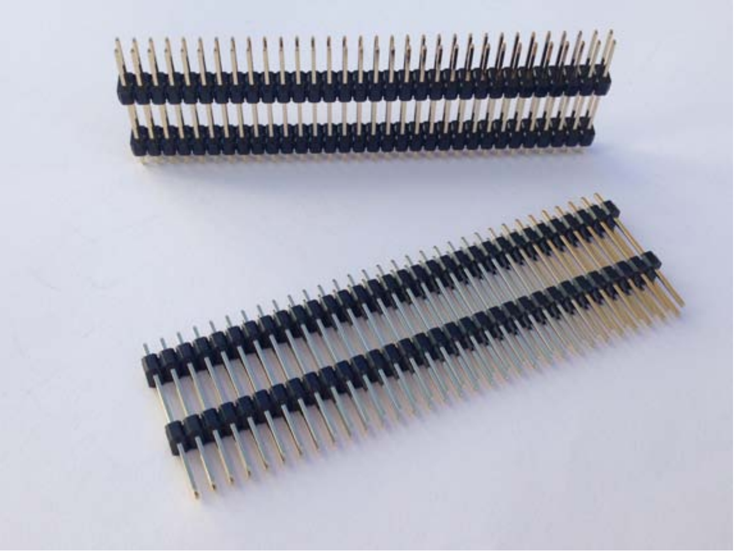


Figure 10.13: *Conexcon* connector 2666-3642; alternative component selected for rising the board AtenTTo Core in order to avoid contact with the components soldered in Multipurpose board.

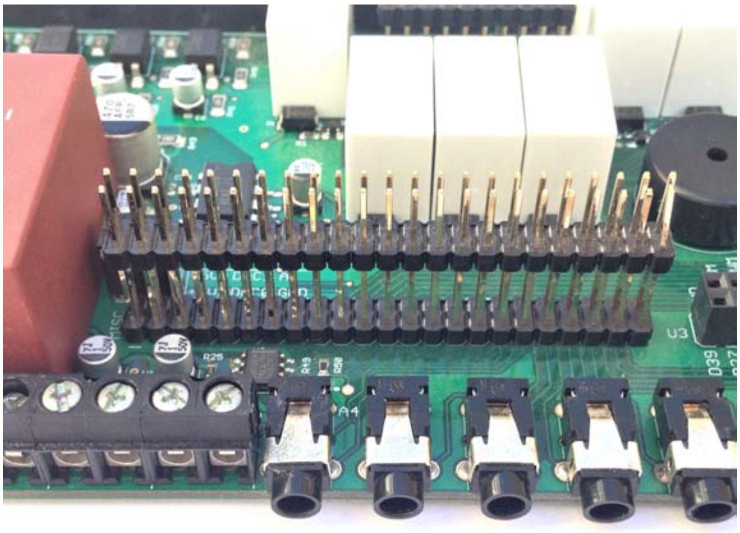
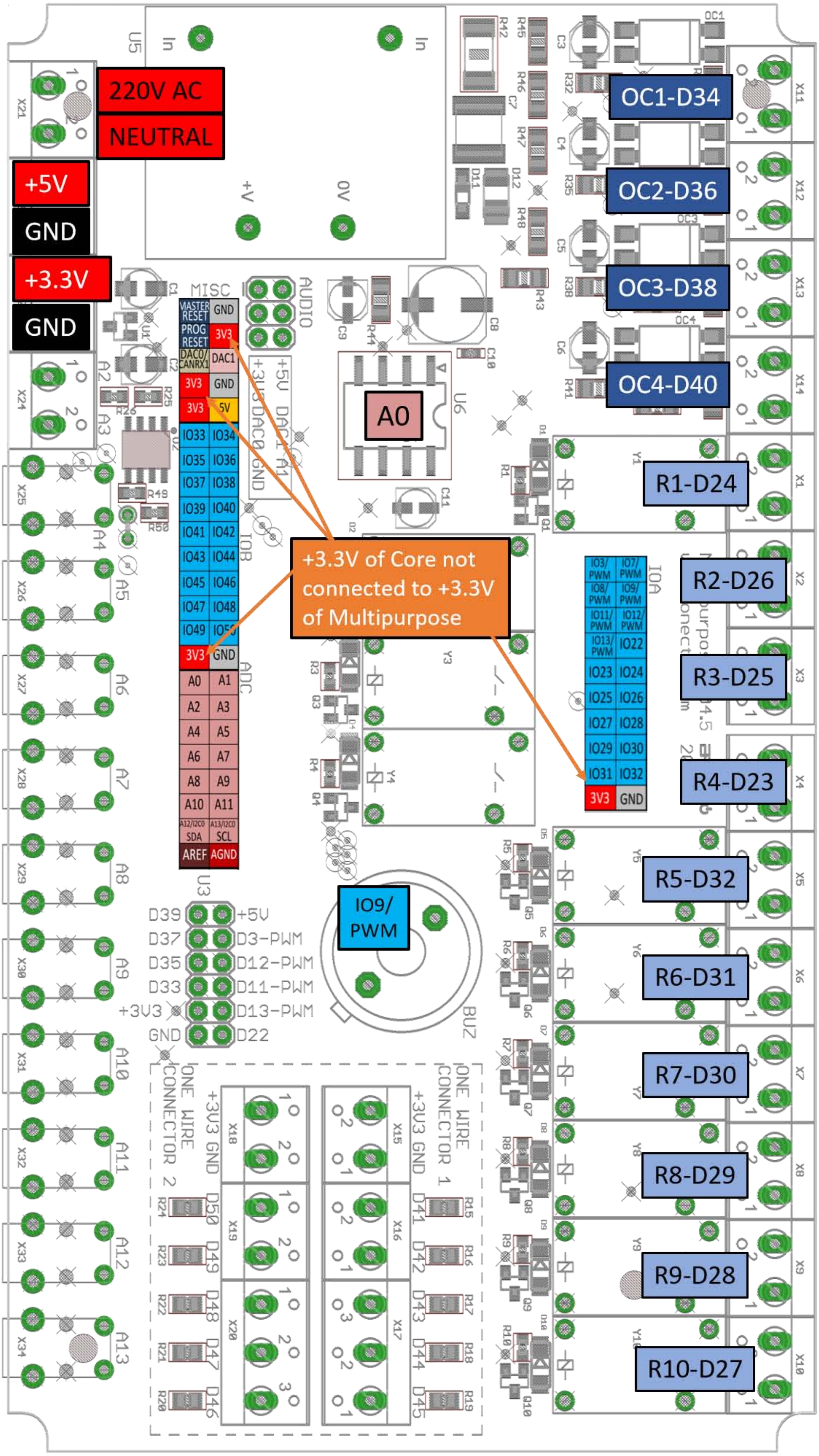


Figure 10.14: Risers already mounted in Multipurpose board.

10.8 Pins and Signals

The main microcontroller in the system is the SAM3X8C and all its lines and signals are distributed in the different layers of AtenTTo Core. A big amount of them are connected to the Multipurpose board through the connectors IOA, IOB, MISC and ADC and those are the important signals for this project. In order to have a quick guide of the important information about Multipurpose it is included a **map of the signals and connectors** (Figure 10.8). It is also included the piece of code that **define all this signals in AtenTTo software** (Code 10.2) so the specific function can be accessed in the program using different naming, being very comfortable for the user since the Multipurpose names are used in the silkscreen of the board.



+3.3V of Core not connected to +3.3V of Multipurpose

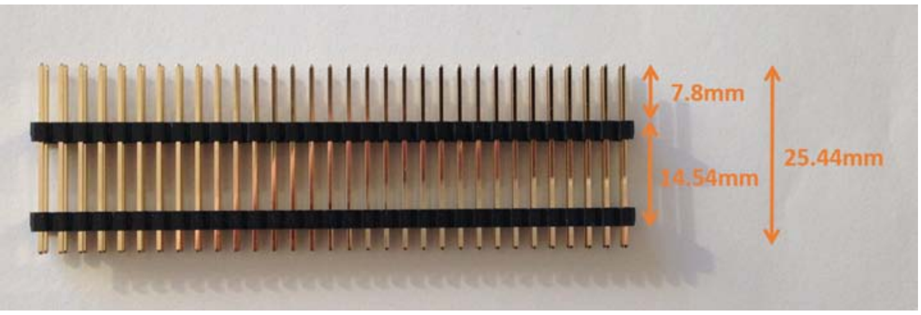


Figure 10.15: Main dimensions of the Risers.

```
// Relays
#define R1  24
#define R2  26
#define R3  25
#define R4  23
#define R5  32
#define R6  31
#define R7  30
#define R8  29
#define R9  28
#define R10 27

// Opto couplers
#define OC1  34
#define OC2  36
#define OC3  38
#define OC4  40

// OneWire or input pullups
#define OW1  41
#define OW2  42
#define OW3  43
#define OW4  44
#define OW5  45
#define OW6  46
#define OW7  47
#define OW8  48
#define OW9  49
#define OW10 50

// X10 protocol
#define X10DATA 33
#define X10ZEROCROSS 35
#define X10RX 37
#define X10TX 39

// PWM
#define PWM1 3
#define PWM2 12
#define PWM3 11
#define PWM4 13
```

Listing 10.2: Definition of the signals in AtenTTo software in order to use the same naming in the board and the software.