MICROCONTROLLED SMART RELAY

Embedded system for environments refrigerated by air-conditioner unit(s)

Preliminary project by ***Brenda Fernandes Ribeiro***

v0.1 Prototype *version updated on April 22, 2021*

Overview

This documentation describes the development process of an embedded system for application in environments refrigerated by air-conditioner unit(s). Its main purpose is energy saving, since, in these environments, open doors and/or windows can be a potential source of energy waste.

The system is composed by a microcontrolled relay for performing, by the click of a button, the energization of an AC unit, conditioned by the response of magnet sensors installed on doors and windows.

Requirements

Functional

- Main input: push button and sensors response.
- Other input: solicitation for usage information download to PC and/or smartphone. Usage information with event and timestamp must be stored in a linked list data structure. The list must be emptied to a desktop file and/or a smartphone app if demanded.
- Main output: relay activation (air-conditioner energization).
- Other output: Usage information file (to desktop and/or smartphone).

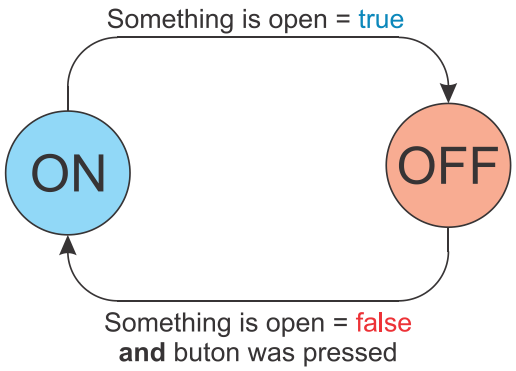
Non-functional

- The hardware and physical structure, after installed, must support room temperature range (22±10°C), and normal indoor weathering conditions (IP20, at least).
- Manufacturing costs must not exceed R$100/unit.
- Response time between main input and main output must not exceed 1s.
- Power autonomy is not necessary. The input must be connected to the 220Vac power grid. A battery can be optionally added to allow the system to support short-term power outages without losing track of time.
- The device must be portable (maximum dimensions being: 15 x 10 x 5cm).

Specifications

Firmware

The main function must be implemented based on the finite state machine (FSM) presented below. The code for the microcontroller must be object oriented (C++). Its design ought to provide easy comprehension and allow adaptation for many architectures, encapsulation is key.



Hardware

The hardware for the inputs should be:

- 1-position dip switch (for time enabling serial communication).
- 2 Push buttons (for starting the system and start the UART sending procedure).
- Magnet sensors.

As for the outputs:

- 3 LEDs (to indicate: if something is open, if the air-conditioner is energized, if the sending mode is enabled)
- Relay (to connect the air-conditioner to its power source).

Software

Desktop

The desktop software for downloading usage information into a file must be developed in C++ language. Its input should be the data sent from the microcontroller via USB serial communication. As for the output, a file containing the received data should be created. Other data processing functionalities may be added in future versions.

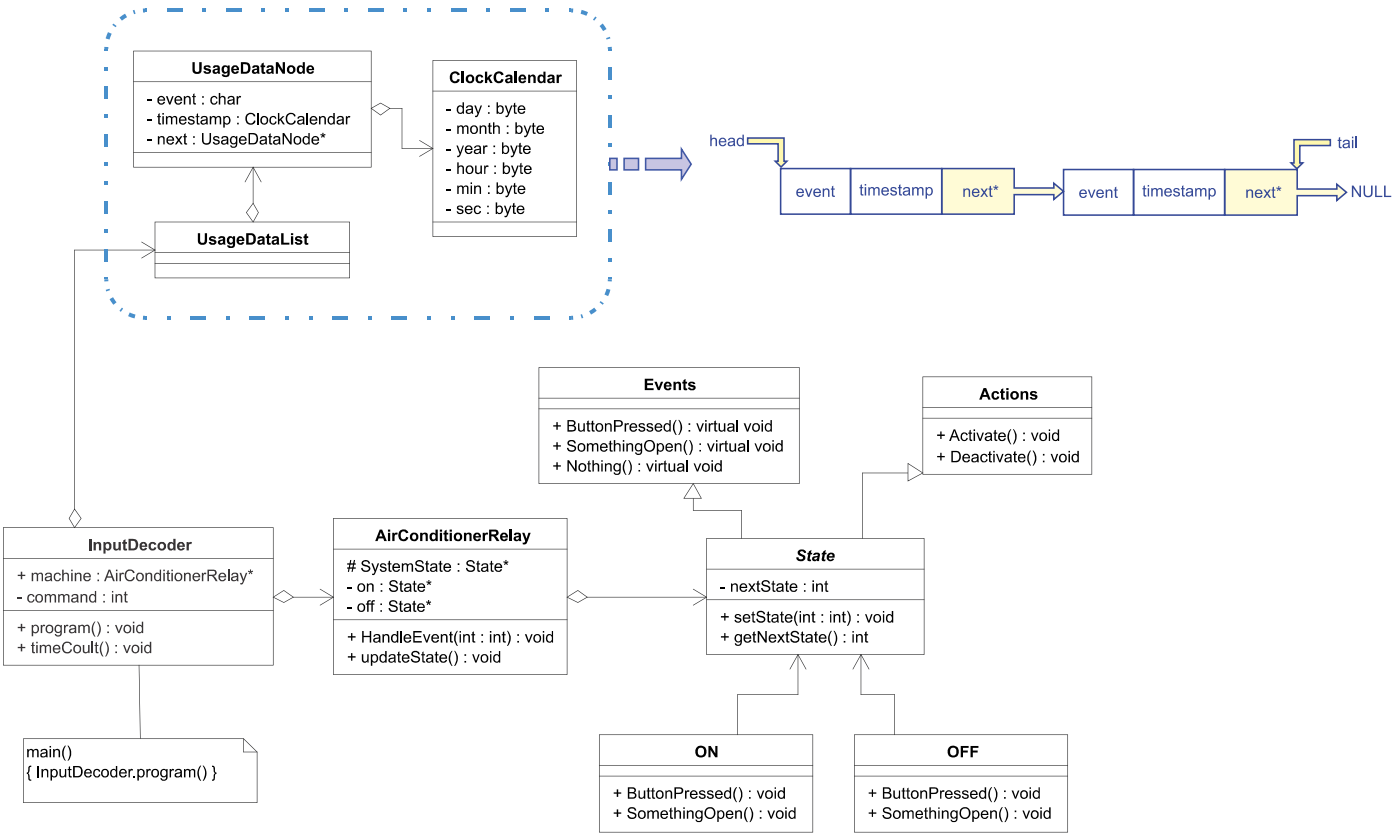
Smartphone

The smartphone app must also be developed in C++ language. It may allow the user to access, through wireless communication, the usage data information stored in the microcontrolled board.

Architecture

Firmware

A sketch of the class diagram is proposed below. The architecture specific commands should be included in the "Action" class (outputs) and in the "InputDecoder" class (inputs), in which the entries must be read and decoded. Not all attributes and methods are represented – these will be detailed in version update notes.



Hardware

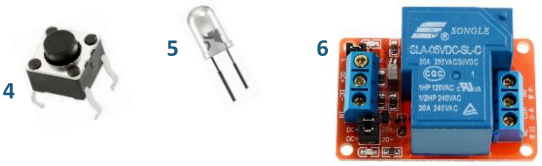
Microcontroller

The Arduino Nano architecture should attend the descripted requirements, for it allows C++ programming, is open source, has low-cost (around R$14/unit, if imported directly from China), has portable dimensions (4.5 x 1.8cm) and supports normal weathering conditions.

Peripherals

1. 1-position dip switch (generic).
2. Push button (plastic, robust).
3. Magnetic sensor (reed-switch).

4. Push button (generic, small).
5. LEDs (small, generic).
6. Relay module (30 A / 240 Vac).



Power

1st Option: No battery

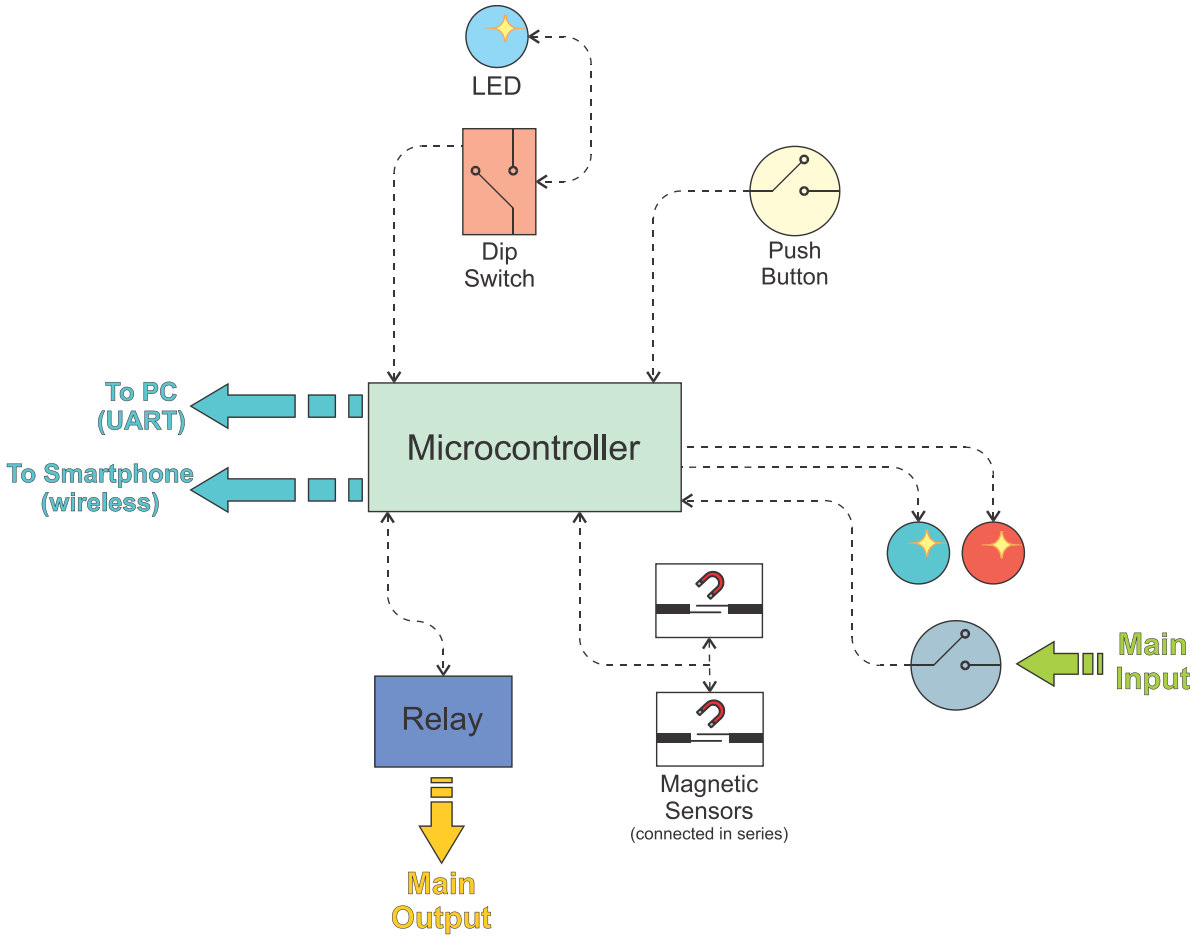
In this case, a 12Vcc/1A power supply is connected to the power grid and to the system through a Jack P4 connector. One step-down module, adjusted to 5Vcc, is needed (suggestion: LM2596-based module, or similar). The 5Vcc output will attend to the microcontroller and the peripherals.

2nd Option: With battery

For this option, the power circuit must also have a charge controller module to interchange between batteries and the power supply connected to the grid. Suggestion: LM317T-based circuit (ATTENTION: this component may require heat dissipator and cooling system).

Block diagram

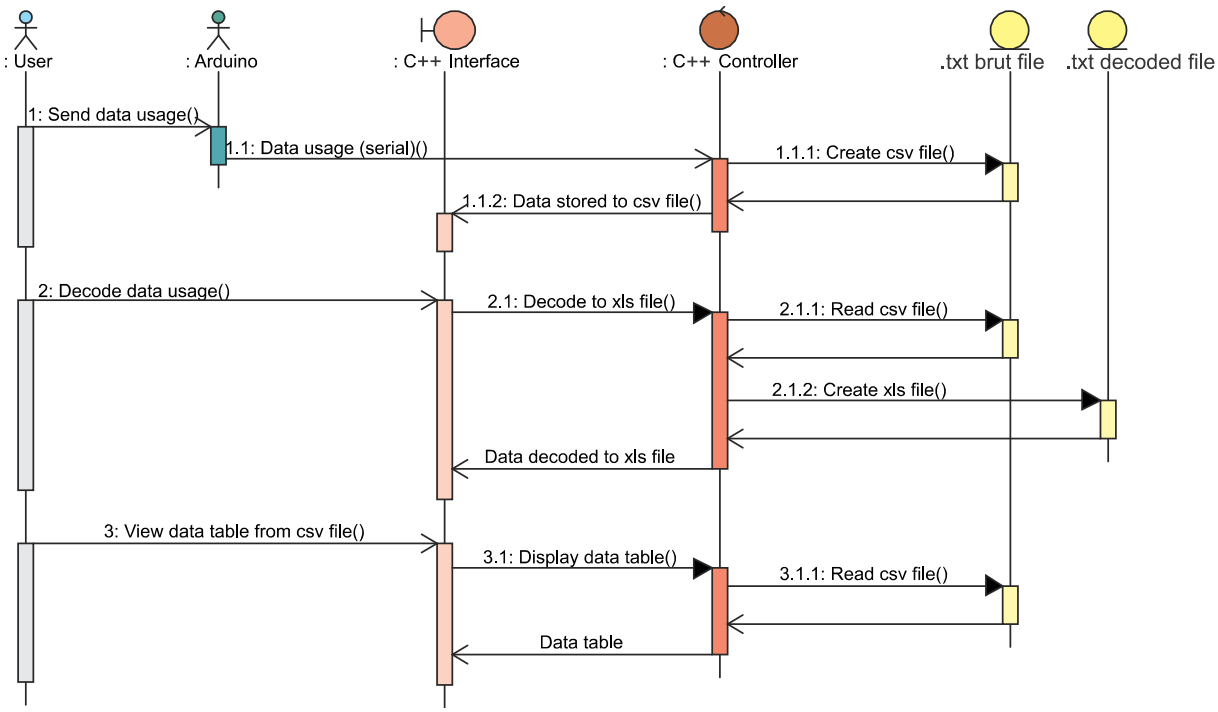
The block diagram bellow provides a preliminary view of the hardware schematics.



Software

Desktop

The desktop software must receive data from the microcontrolled board through serial communication via USB. The code and interface should interact with the user and board as represented in the preliminary sequence diagram displayed ahead. The program must also interact with a csv file generated by itself.



Smartphone

[Em construção] Não consegui especificar a arquitetura desta aplicação ainda, pois ainda não estou certa sobre alguns aspectos de desenvolvimento de aplicativo e não tenho certeza do tipo de comunicação que utilizarei (wifi/bluetooth).

Costs

Prototype v0.1

Description	***Price (R$)***
Microcontroller (Arduino Nano V3)	29,90
4-position dip switch	2,40
Push-buttons (small) x2	0,98
LEDs x3	2,40
Resistors (diverse)	2,90
TOTAL	***38,58***

Product [still not updated]

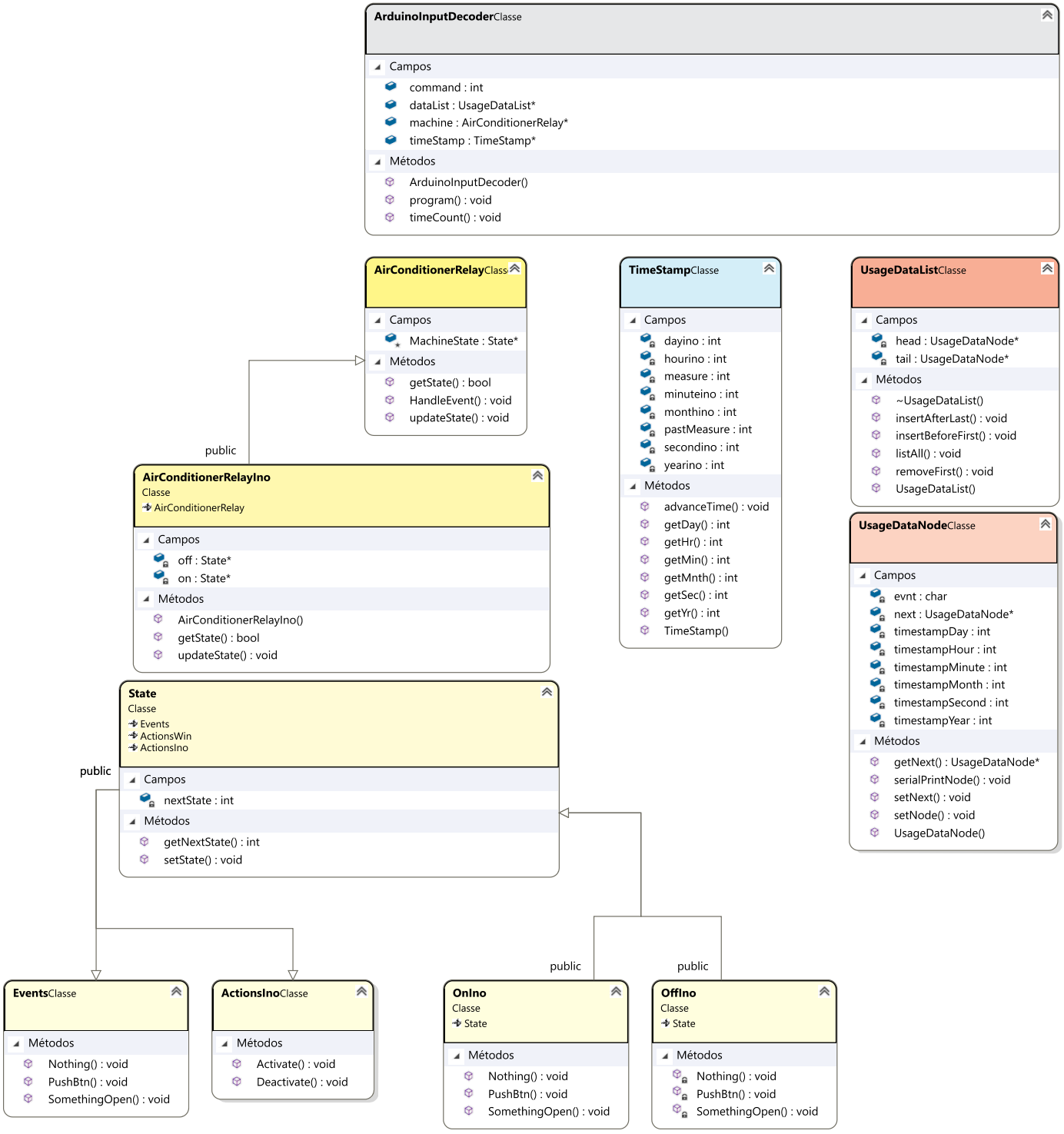
The estimated cost value for the product is detailed below (considering medium scale production).

Description	***Price (R$)***
Microcontroller (Arduino Nano V3)	10,00
4-position dip switch	0,55
Push-button	7,00
Relay 10A	4,50
Relay 50A	15,00
Float switch	8,00
Resistors, buzzer	1,00
Display 7 segments (4 digits)	4,00
Power supply	10,00
Plastic enclosure	5,00
Other costs (manufacture, fees, etc.)	5,50
TOTAL	***70,55***

v0.1: Prototype

Firmware

The code of the first prototype version (v0.1) is organized as shown in the class diagram below.



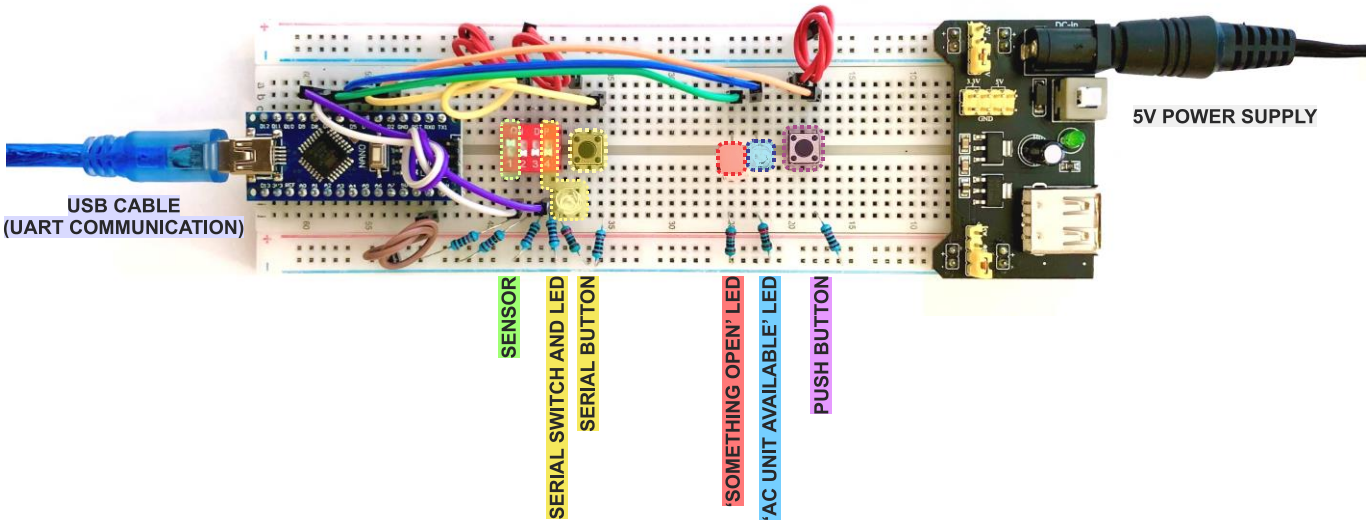
The main() program consists of system initialization routines and a loop containing ArduinoInputDecoder::program() method.

Hardware

The prototype was built using the following components:

- 5Vcc power supply
- Push button tactile micro switch (02 un)
- High bright led (02 un as outputs; 01 un in the circuit)
- 4-way dip switch
- Arduino nano microcontroller
- 10kΩ resistor (06 un)
- 220Ω resistor (03 un)
- Protoboard
- Jumpers (male-male)
- USB to USB mini cable (for UART communication with PC)

The circuit is shown below.



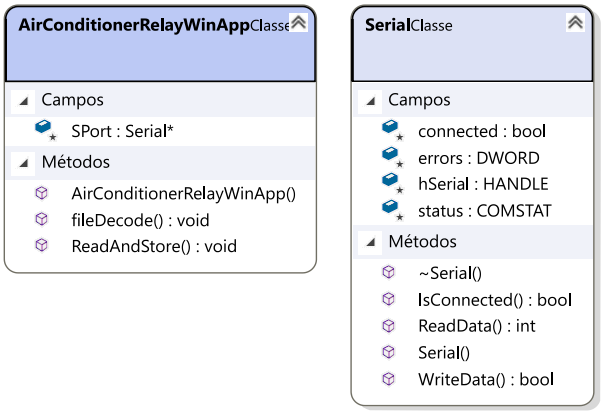
An USB cable is used to stablish serial communication (via UART protocol) to a computer. Also, a dip switch is used to allow data reading (serial switch). The data is sent by the hardware once this switch is ON and the serial button is pressed. This switch is connected to a LED, to visually inform the user about its state. This visual alert is particularly important, because, once the data is sent, it is deleted from the Arduino board.

In this prototype: a switch represents the set of magnetic sensors in the circuit; two LEDs indicate whether there is something open or the AC unit is available; a simple push button represents the robust button accessible to the final user.

Software

Desktop

The software was developed for running in Windows 10 environment. Its class diagram is shown below.



The “main.cpp” consists of a global “AirConditionerRelayWinApp” object and a void main() function containing an infinite loop for the console app menu. The menu has three options chosen accordingly to user’s input:

[1] Read and store: Reads data sent from the microcontrolled board via UART and stores to a *.txt* file.

[2] Decode file: Decodes data from a *.txt* file, generating a new file or appending the information to an existing one. This process transforms the brut data received from the board (*csv* alike) to a better comprehensible format.

[3] Exit: Leaves the console program.

Smartphone

Smartphone software is not contemplated in this version.

Tests

Methodology

[will be updated until April 30]

Results

[will be updated until April 30]