Contents

[Abstract…………. 2](#_Toc381273776)

[1. Project specifications 3](#_Toc381273777)

[1.1 Introducing reactive system 3](#_Toc381273778)

[1.2 Calculus of communications systems (CCS) 4](#_Toc381273779)

[1.3 Specifications for the system on Aduino UNO 5](#_Toc381273780)

[1.4 Specifications for the system on PC 6](#_Toc381273781)

[1.5 Project specifications 7](#_Toc381273782)

[2. Designing 8](#_Toc381273783)

[2.1 project architecture 8](#_Toc381273784)

[2.2 System workflow 9](#_Toc381273785)

[2.3 Data file structure 11](#_Toc381273786)

[3. Project components 12](#_Toc381273787)

[3.1 C#.............................. ………………..12](#_Toc381273788)

[3.2 Arduino UNO 14](#_Toc381273789)

[3.2.1 Input and output 15](#_Toc381273790)

[3.2.1 Programming on Arduino UNO 16](#_Toc381273791)

[3.2.3 Communications 18](#_Toc381273792)

[3.3 LM 35 thermo sensor 18](#_Toc381273793)

[3.4 Breadboard 20](#_Toc381273794)

[4. Screenshots 21](#_Toc381273795)

[5. Conclusions 26](#_Toc381273796)

# Abstract

Fire Alarm System is a system with the responsibility of protecting and preserving an institution's buildings, collections, operations and occupants. Constant attention is required to minimize adverse impact due to climate, pollution, theft, vandalism, insects, mold and fire. Because of the speed and totality of the destructive forces of fire, it constitutes one of the more serious threats. Vandalized or environmentally damaged structures can be repaired and stolen objects recovered. Items destroyed by fire, however, are gone forever. An uncontrolled fire can obliterate an entire room's contents within a few minutes and completely burn out a building in a couple hours.

The first step toward halting a fire is to properly identify the incident, raise the occupant alarm, and then notify emergency response professionals. This is often the function of the fire detection and alarm system.

The following text presents an overview of fire detection and alarm systems including system types, components and operations. The Fire Alarm System operates as a fire detector and alarm. This system has three components which are heat detector, mini-controller and alarming system. Mini-controller operates the entire detecting component while the alarming system supervises heat.

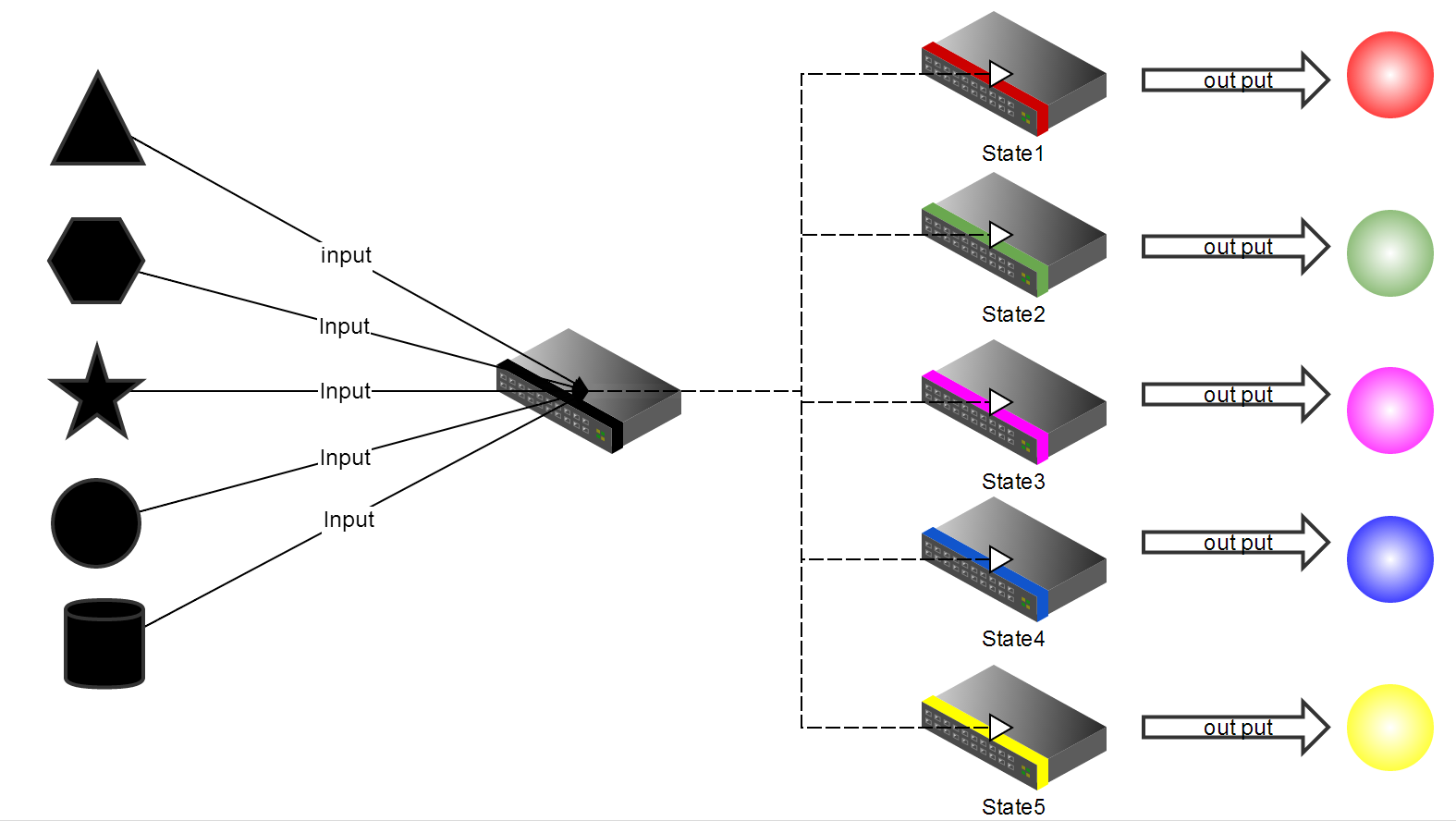
# Project specifications

## Introducing reactive system

The ‘standard’ view of computing systems is that, at a high level of abstraction,

these may be considered as black boxes that take inputs and provide appropriate outputs. This view agrees with the description of algorithmic problems.

An algorithmic problem is specified by a collection of legal inputs, and, for each legal input, its expected output. In an imperative setting, an abstract view of a computing system may therefore be given by describing how it transformsan initial state – i.e. a function from variables to their values – to a final state.



According this we can define reactive system.

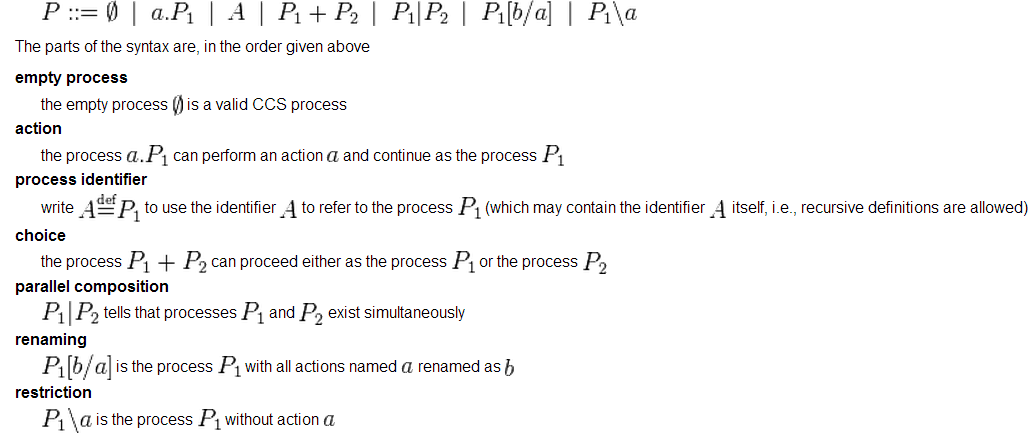
A reactive system is a system that responds (reacts) to external events. Typically, biological systems are reactive, because they react to certain events. However, the term is used primarily for describing human-made systems. For example, a light consisting of a bulb and a switch is a reactive system, reacting to the user changing the switch position.

A most important type of reactive systems is of interactive systems. These systems may react to the events by providing output to its users and operators. The output may be feedback to the event or indication about the system condition. Typically, biological systems are interactive, providing feedback to certain event, such as feeding. However, the term is used primarily to describe machines, designed to perform certain functions to their operators or users. Examples of interactive systems include information systems, workflow management systems, systems for e-commerce, production control systems, and embedded software.

## Calculus of communications systems (CCS)

The Calculus of Communicating Systems (CCS) is a process calculus introduced by Robin Milner around 1980 and the title of a book describing the calculus. Its actions model indivisible communications between exactly two participants. The formal language includes primitives for describing parallel composition, choice between actions and scope restriction. CCS is useful for evaluating the qualitative correctness of properties of a system such as deadlock or livelock. According to Milner, "There is nothing canonical about the choice of the basic combinators, even though they were chosen with great attention to economy. What characterises our calculus is not the exact choice of combinators, but rather the choice of interpretation and of mathematical framework". The expressions of the language are interpreted as a labelled transition system. Between these models, bisimilarity is used as a semantic equivalence.

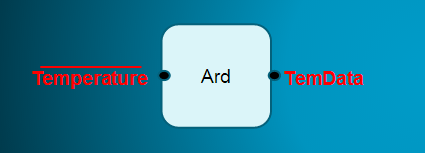
**Reactive system Syntax**

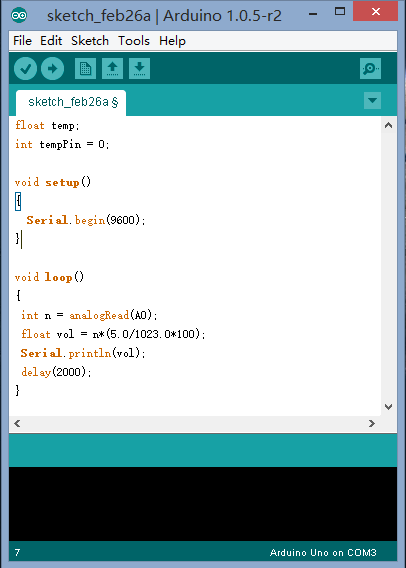
****

## Specifications for the system on Aduino UNO

Program for Arduino is written in C or C ++, we wrote a simple program which enables Arduino to sent data to computer every two seconds. To specify this simple system we have use CCS specification language. In the picture below we made informal process specification.

Since we have only one thermo temperature that attached to Arduino, It can only one input which is the real time environment temperature, data is received by sensors and Arduino sends date to computer as an output of the system.



****

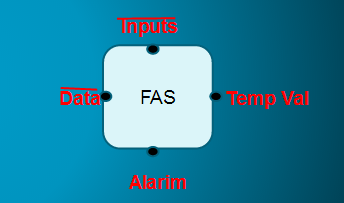
## Specifications for the system on PC

System on computer receives data from Arduino trough the serial port name COM3, together with system tries to find another data which is already been stored in data file, this data will is used for compare with data that received from arduino. To sum up the running system on computer receives two different kind of data from two different sources.

On the getting of two data the algorithm that we build compares them, based on the comparison result the system performs two kinds of actions. One of them if the data that received from Arduino is bigger than the other one that stored, the system performs two actions concurrently, one is show the actual temperature on the computer screen, the other one is making the alarm system.

Another is if the data is less then stored data the system performs only one action which is showing the temperature on the computer screen every 5 seconds.

This process will continues one by one through both all the data received from serial port and stored data.



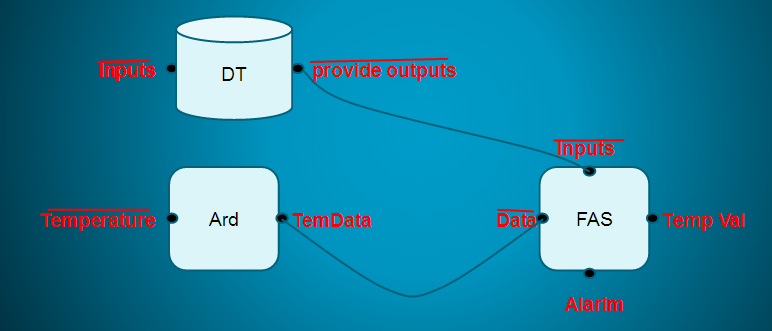
**FAS = temDa.inputs.Alarm.tempVal.FAS**

In this informal specification of process, Data represents temperature value that sent by Aduino, the inputs stand for data that stored on the data file. They both input for the FAS(Fire alarm system).

On the contrary Temp Val and Alarm are outputs of the system, they are triggers of the actions.

## Project specifications

Considering the whole project, It is divided into two separated system, these systems are performing in parallel way. They have connected interfaces. For instance, as shown in the diagram below, the TemData as an output for Arduino while FAS takes it as an input.



Another informal way to specify the process of whle project

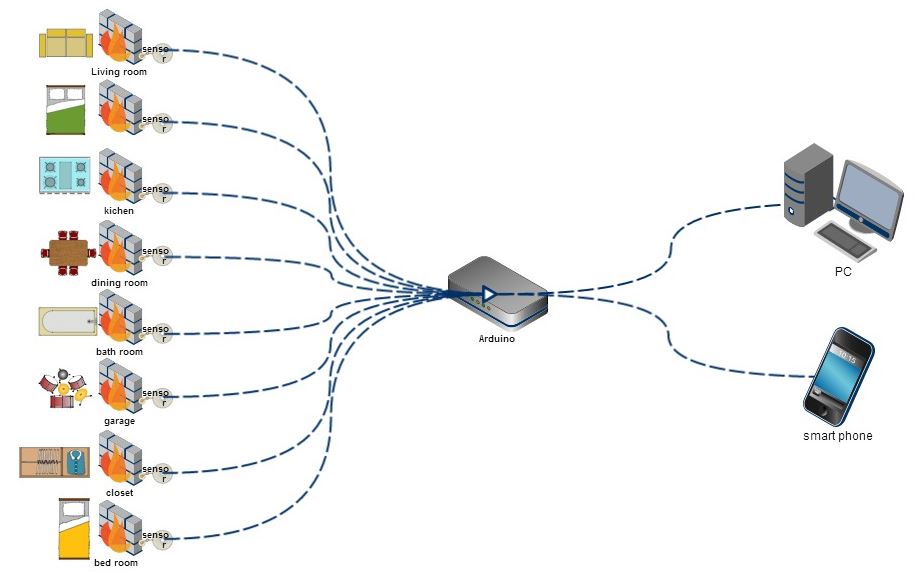


WS represents for Whole system or the project, it consists of two system which we mentioned above, one is the system on Arduino another is on the PC (FAS|Ar).

FAS|Ar is the meaning of parallel system, they work on their own way. If we replace FAS and Ar with associated ones we will get detailed specification of the project.

# 2. Designing

## 2.1 project architecture



**Project architecture diagram**

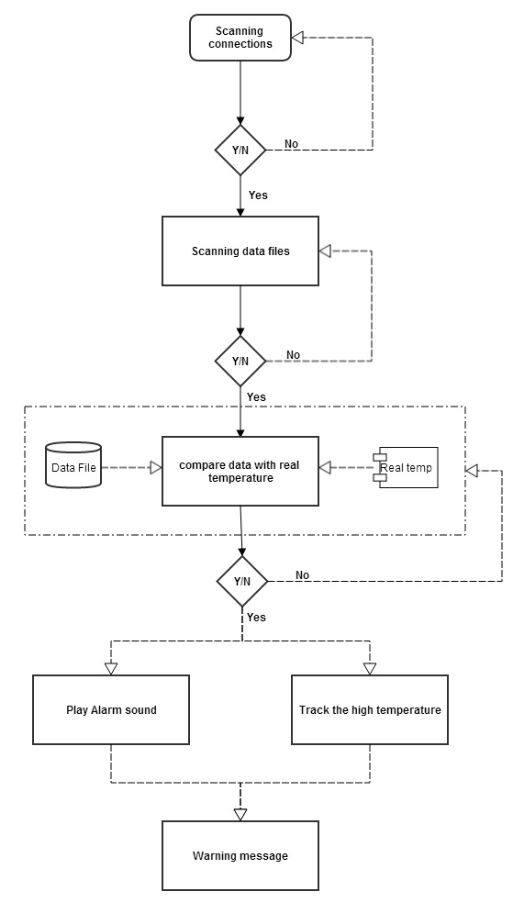
The architecture has three different parts, first is installing the sensors in different rooms each sensor is connected to the Arduino and the measures temperature they involved. The operating temperature range of the sensor is from -55°C to 150°C .

Second part is Arduino, it receives all the temperature values and then immediately sends them to computer and smart phones via cable or wireless.

The third part this computer and mart phone, the algorithm that have been written compares data with certain value on the data file, whenever if considers data from Arduino, computer or smart phone perform actions.

## 2.2 System workflow

At the initial running time, the system scans all connected serial ports to prepare receive data from Arduino, the Arduino will provide the alarm system with real time thermo temperature data that measured by connecting sensors every 5 seconds, if the computer (system) does not have any Arduino connections or if it lost connection during normal working it will alarm the connection lost with sound and message window.



**Alarm system workflow**

After successfully scanning the ports, the system starts to provide the information associate with stored data file and information will be shown on the main window, such as: amount of blue print that have been stored, number of sensors that currently working and connected serial ports.

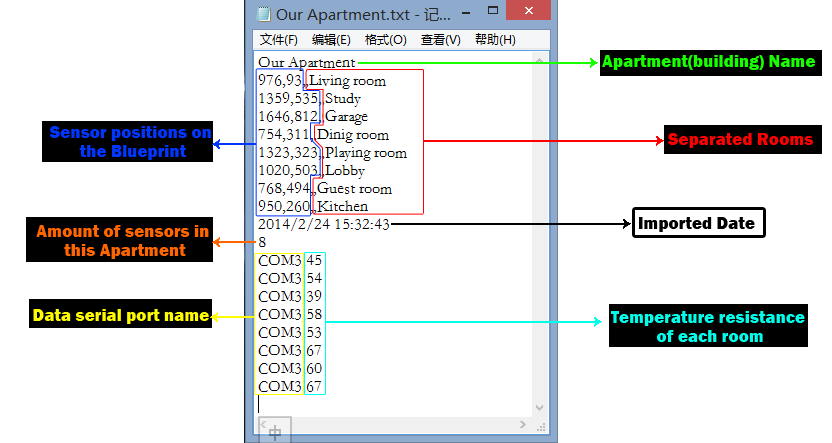
Data file includes maximum centigrade digit numbers for every sensor, which have been input by users. So the system starts working whenever it finds data file and compare maximum resistance with the real time temperature value. whenever the temperature exceeded the maximum resistance, the system tracks, sensors, make fire alarm and then shows sensor location on the blueprint.

If the system does not have any blueprint, user needs to import it, first user imports blueprint that existed on local machine then user will be asked to give some description for the blueprint after this is done the sensors will be activated and positioned in associated locations on the map. Before confirming the sensor’s location it bell be required to input some information related to the sensor and it position.

Once the saving of processes system will begin to work with including of new blueprint.

## 2.3 Data file structure

As shown in the picture below, we divided the data into four big part.

The sensor positions on the blueprint is used for tracking and observing the position of sensors on the blueprint, they are generated during placing the sensors on the blueprint. 

**Overview of the Data base**

Separated room descriptions are also generated together with sensors, they are represent specific room in the real world. When high temperature is detected or checking the realtime temperature the room description will be provided by the system.

Temperature resistance of each sensor is defined by the user by using the controller on the interface. Whenever is the real temperature exceeded this value system considers it a fire instance.

Data serial port names are linked to sensors, this is the tracking point of fire.

User guide of system

# 3. Project components

## 3.1 C#

C# is designed to be a platform-independent language in the tradition of Java (although it is implemented primarily on Windows). It's syntax is similar to C and C++ syntax, and C# is designed to be an object-oriented language. There are, for the most part, minor variations in syntax between C++ and C#. Main has no return type, there are no semicolons after class names, there are some (to C++ programmers) strange decisions regarding capitalization - such as the capitalization of Main. A few other differences, the syntax is often the same. This decision is reasonable, in light of the fact that C syntax has been used with several other languages - notably Java.

Similar to Java, C# does not support multiple inheritance; instead it provides the Java's solution: interfaces. Interfaces implemented by a class specify certain functions that the class is guaranteed to implement. Interfaces avoid the messy dangers of multiple inheritance while maintaining the ability to let several classes implement the same set of methods.

Another helpful feature of C# is garbage collection. Therefore, it is unnecessary to include a destructor for each class unless a class handles unmanaged resources; if so, it's necessary to release control those resources from within the class (The Finalize function is used to clear up these unmanaged resources; it can even be abbreviated with the same syntax as a C++ destructor). Of course, C# also provides direct access to memory through C++ style pointers, but these pointers are not garbage collected until specifically released by the programmer.

C# programs run on the .NET Framework, an integral component of Windows that includes a virtual execution system called the common language runtime (CLR) and a unified set of class libraries. The CLR is the commercial implementation by Microsoft of the common language infrastructure (CLI), an international standard that is the basis for creating execution and development environments in which languages and libraries work together seamlessly. Source code written in C# is compiled into an intermediate language (IL) that conforms to the CLI specification. The IL code and resources, such as bitmaps and strings, are stored on disk in an executable file called an assembly, typically with an extension of .exe or .dll. An assembly contains a manifest that provides information about the assembly's types, version, culture, and security requirements. When the C# program is executed, the assembly is loaded into the CLR, which might take various actions based on the information in the manifest. Then, if the security requirements are met, the CLR performs just in time (JIT) compilation to convert the IL code to native machine instructions. The CLR also provides other services related to automatic garbage collection, exception handling, and resource management. Code that is executed by the CLR is sometimes referred to as "managed code," in contrast to "unmanaged code" which is compiled into native machine language that targets a specific system. The following diagram illustrates the compile-time and run-time relationships of C# source code files, the .NET Framework class libraries, assemblies, and the CLR.



**Dot Net Framework Architecture**

Language interoperability is a key feature of the .NET Framework. Because the IL code produced by the C# compiler conforms to the Common Type Specification (CTS), IL code generated from C# can interact with code that was generated from the .NET versions of Visual Basic, Visual C++, or any of more than 20 other CTS-compliant languages. A single assembly may contain multiple modules written in different .NET languages, and the types can reference each other just as if they were written in the same language. In addition to the run time services, the .NET Framework also includes an extensive library of over 4000 classes organized into namespaces that provide a wide variety of useful functionality for everything from file input and output to string manipulation to XML parsing, to Windows Forms controls. The typical C# application uses the .NET Framework class library extensively to handle common "plumbing" chores.

## 3.2 Arduino UNO

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.



**Picture 2.ArduinoUNO**

### 3.2.1 Input and output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

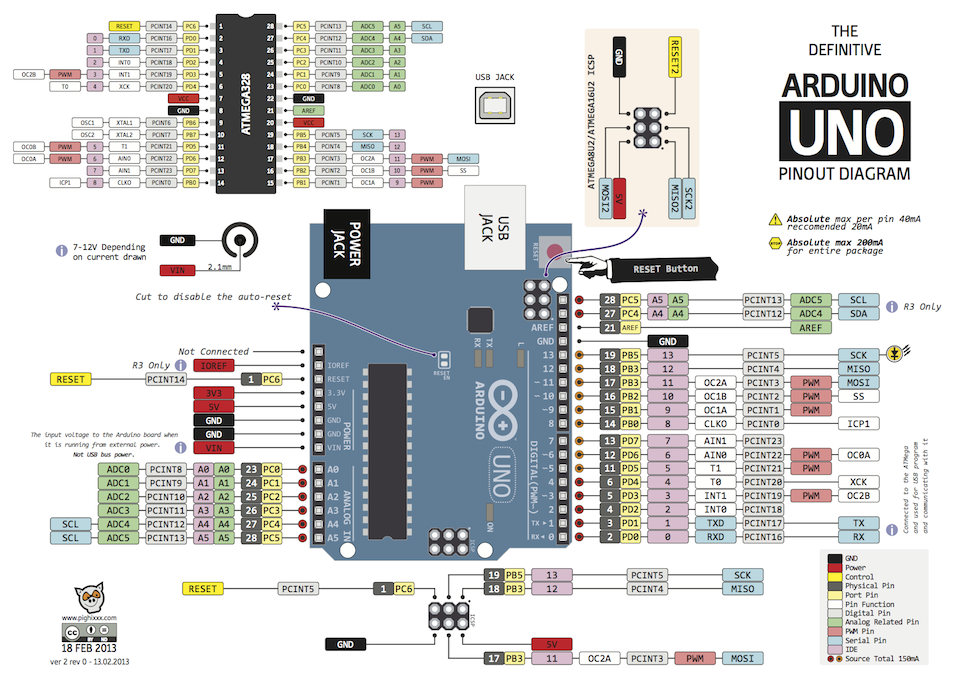
LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analog Reference() function. Additionally, some pins have specialized functionality:

TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library. There are a couple of other pins on the board:

AREF: Reference voltage for the analog inputs. Used with analog Reference().

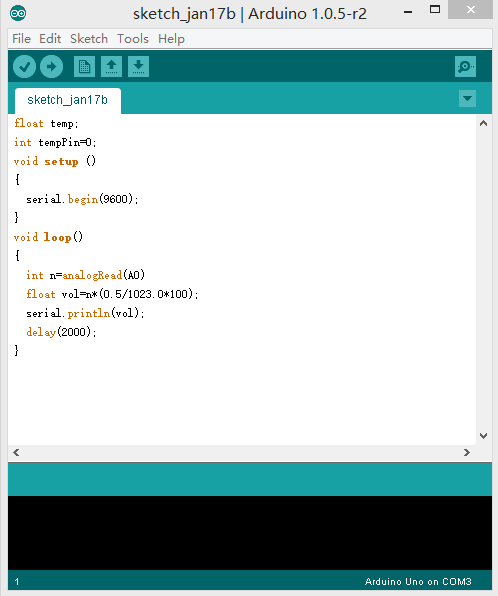
Reset: Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.



**Arduino UNO Pinout Diagram**

### 3.2.1 Programming on Arduino UNO

The Arduino UNO can be programmed with the Arduino software. Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials. The ATmega328 on the Arduino UNO comes pre-burned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files). You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by: On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.



**Our program on Arduino UNO**

### 3.2.3 Communications

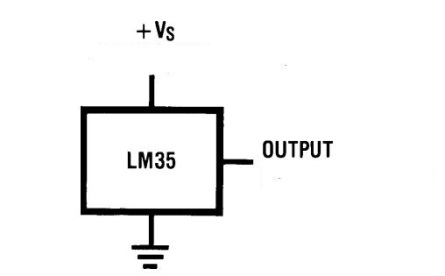
The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

## 3.3 LM 35 thermo sensor

A sensor is a converter that measures a physical quantity and converts it into a signal which can be read by an observer or by an (today mostly electronic) instrument.

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in˚ Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ± 1 ⁄ 4 ˚Cat room temperature and ± 3 ⁄ 4 ˚C over a full −55 to +150˚Ctemperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35’s low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1˚C in still air. The LM35 is rated to operate over a −55˚ to +150˚C temperature range, while the LM35C is rated for a −40˚ to +110˚C range (−10˚with improved accuracy).



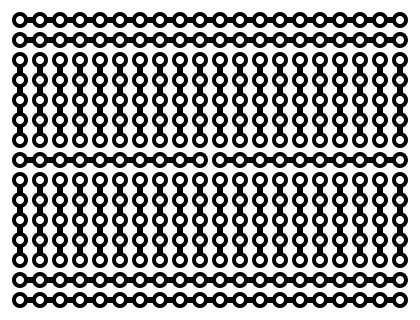
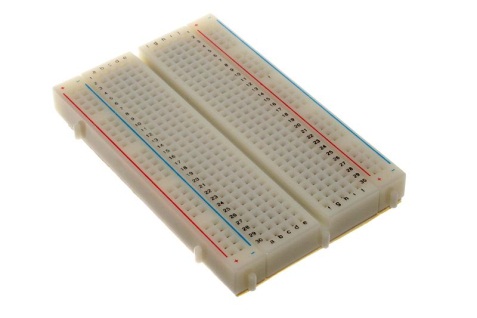
**LM 35 Thermo temperature sensor**

## 3.4 Breadboard

A breadboard (or protoboard) is usually a construction base for prototyping of electronics. The term "breadboard" is commonly used to refer to a solderless breadboard (plug board).

A breadboard originally was a flat wooden [cutting board](http://en.wikipedia.org/wiki/Cutting_board) used to support a loaf of [bread](http://en.wikipedia.org/wiki/Bread) (or other foods) while it was being sliced; this original meaning is still in use, but has a new additional meaning as "a base for prototyping". The concept of "bread boarding" as prototyping is not confined to electronic design; "mechanical breadboards" have been and continue to be used by [mechanical engineers](http://en.wikipedia.org/wiki/Mechanical_engineer).

Because the solderless breadboard for electronics does not require [soldering](http://en.wikipedia.org/wiki/Soldering), it is reusable. This makes it easy to use for creating temporary prototypes and experimenting with circuit design. Older breadboard types did not have this property. A [strip board](http://en.wikipedia.org/wiki/Stripboard) (veroboard) and similar prototyping [printed circuit boards](http://en.wikipedia.org/wiki/Printed_circuit_board), which are used to build semi-permanent soldered prototypes or one-offs, cannot easily be reused A variety of electronic systems may be prototyped by using breadboards, from small analog and digital circuits to complete [central processing units](http://en.wikipedia.org/wiki/Central_processing_unit) (CPUs).

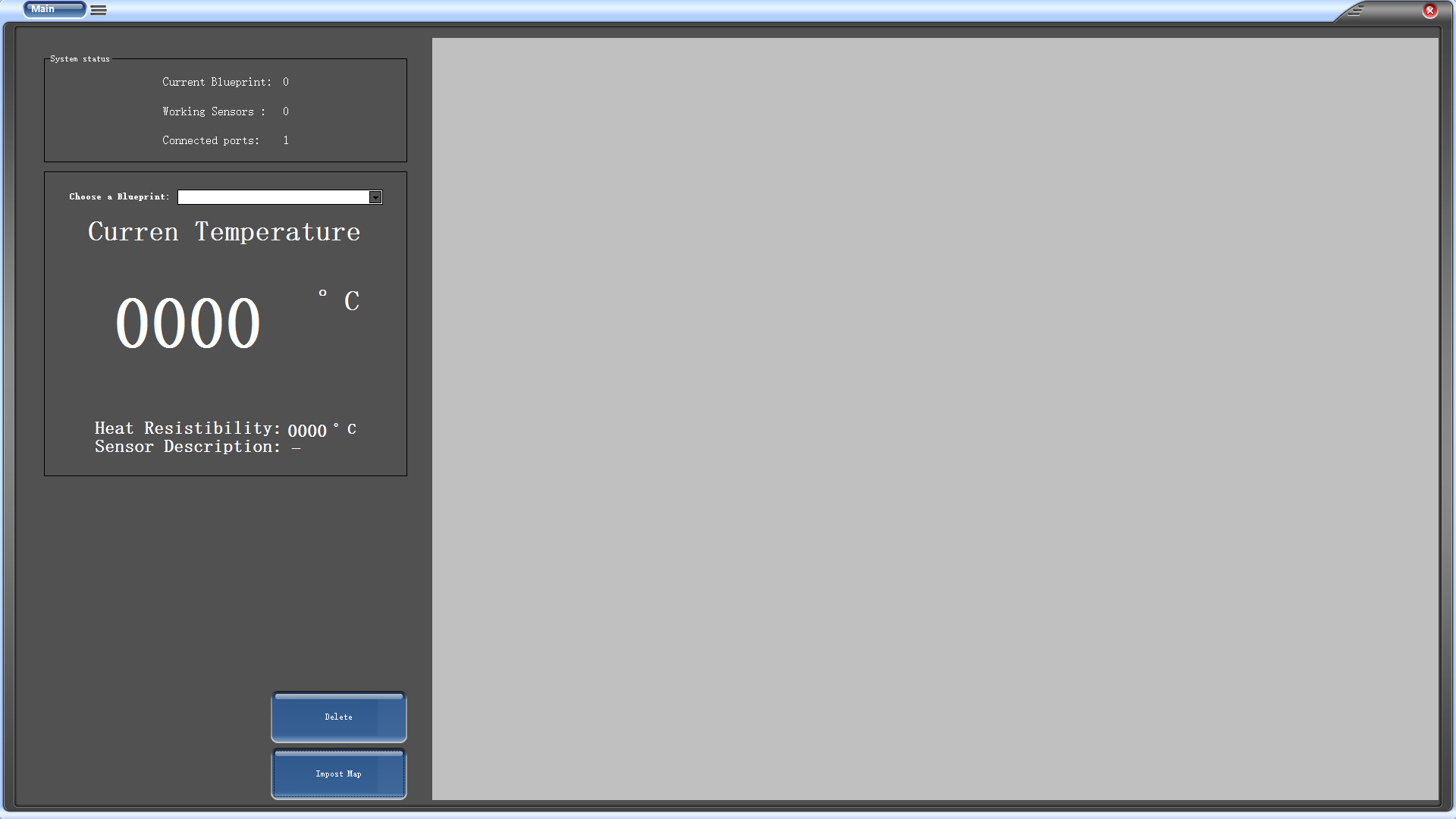


**Bread and its structure**

# Screenshots

**System main interface:**

This is the main window appears when we run the system on the PC, on starting the system it begins to search interfaces for receiving data and calculates amounts of blueprint, virtual sensors and amounts of connections that computer has. If the system could not find any connections it will sound and warn use to check for connections.



**Activating virtual sensors**

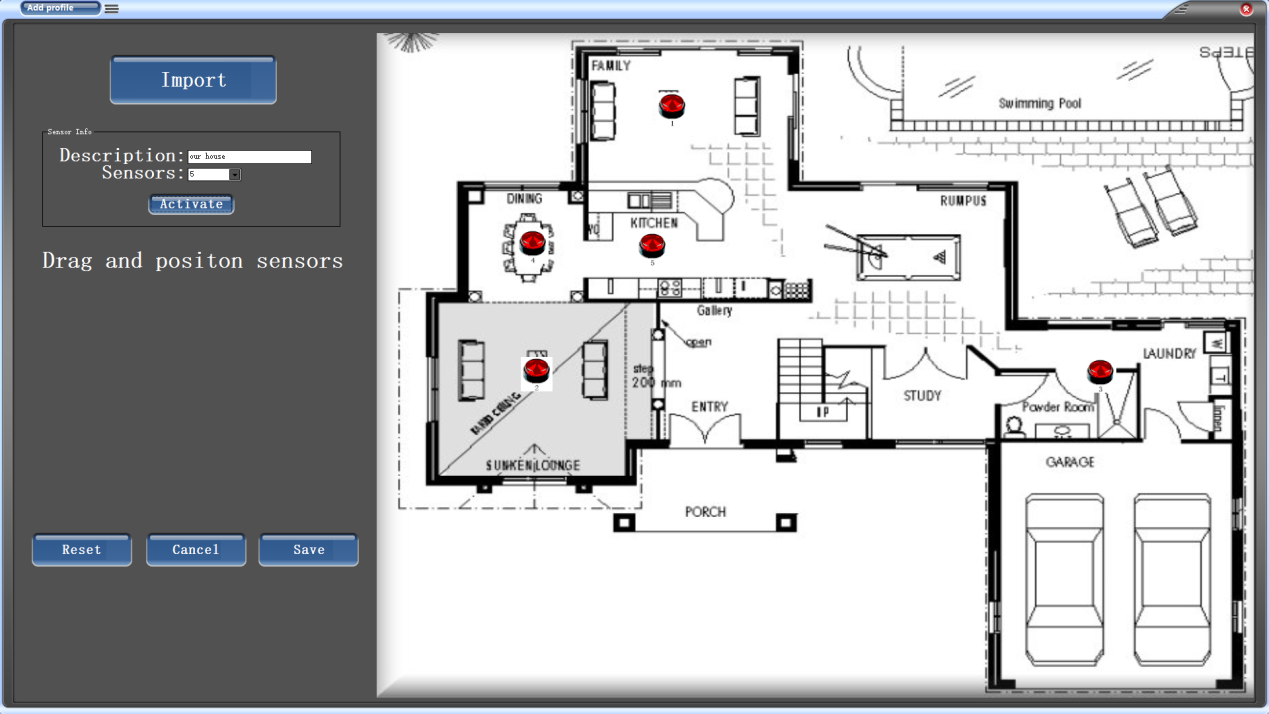
Once user declares the amounts of sensors he needs, the virtual sensors will appear, system user needs to drag and drop the sensor on the blueprint then user is require to input parameters associate to the dropped sensor, ports means data incoming interface for the sensor, for the description it will be needs to input some information about environment that sensor involved.

Most important part is resistibility, use define maximum resistibility for the sensor based on the his demand, for example kitchen is the place that fire always stats so it is relatively sensible place in an house, for this reason we may give it lower value. in additions to this user could observe amounts of alive interface and unique sensor name.



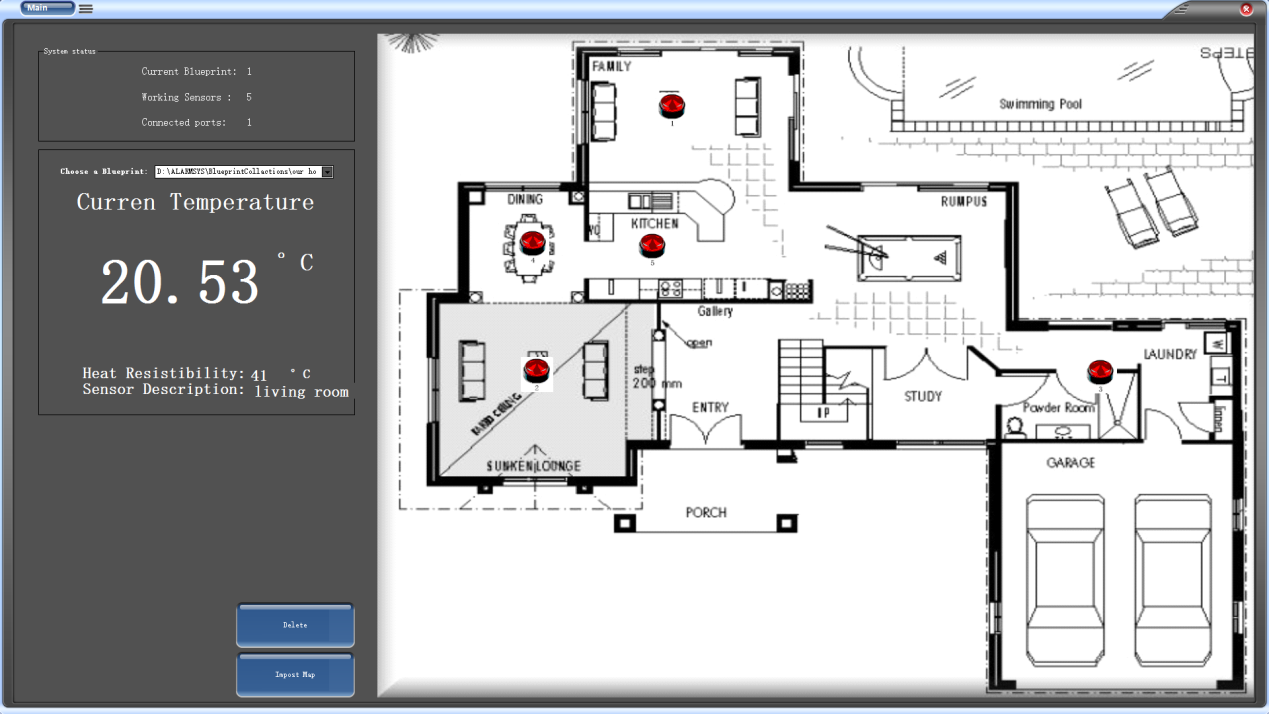
**Importing blueprint**

system accepts JPG formations of blueprint. Once importing is done the blueprint shows up in the this interface.



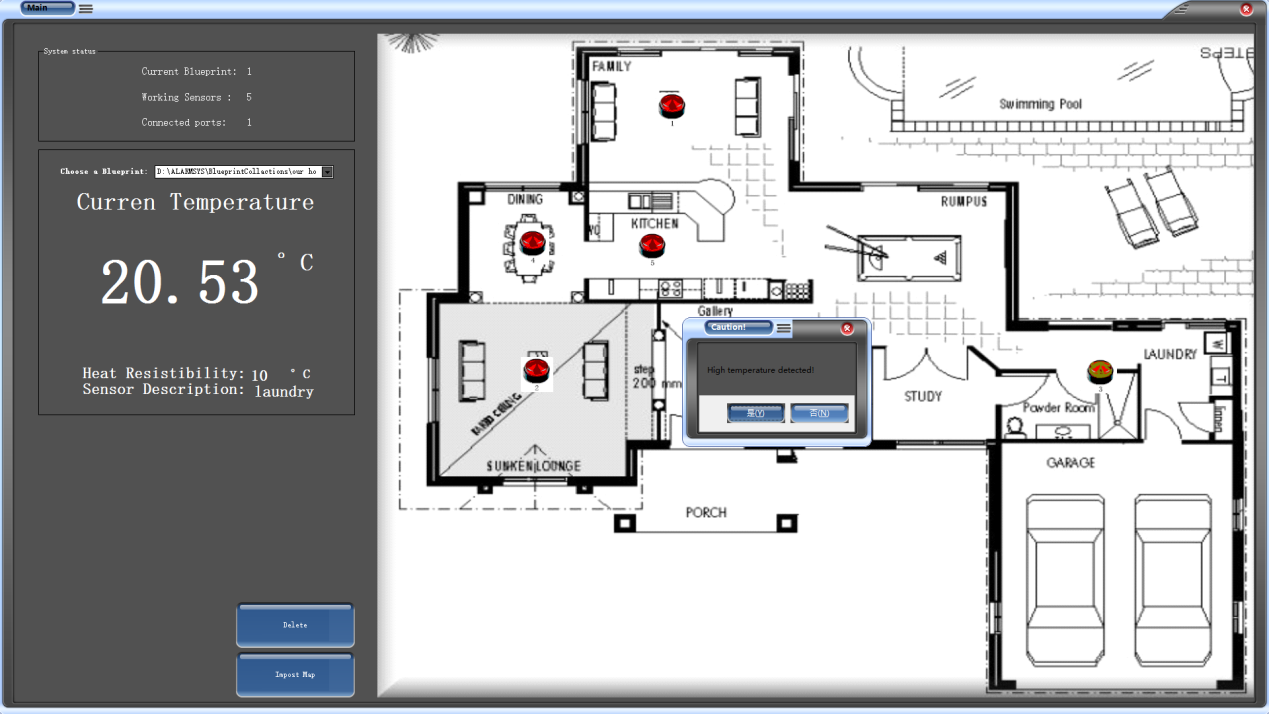
**Temperature observatory**

User can look through all the blueprint and observe actual temperature by pressing the virtual sensors, the temperature is refreshed every 5 seconds, plus system provides resistibility of the sensor and locations information.



**Fire detection**

Fire detection function performs at first starting of the system with collaboration of data base, an algorithm receives the actual data and make comparison with data stored in the data base. If it matches condition the algorithm tracks sensor, makes sound and warning massage.



# Conclusions

Through this entire course we have knowledge of Reactive system, calculus of communication system and specifications. Especially the formal and informal specifications, process specifications and modeling the system. In the project process we became familiar with the open source hardware and its working functions, how to use bread board and use of sensor.

We also learned how to integrate and collaborate with computer and open source hardware devices and it accessories, designing the project was one of the tough process of system stability.

In Coding stage we build small system in C on the hardware device Arduino, the system recognizes the analog inputs and digital connections then receives data and sends it to computer via serial ports. For the system on PC we build a software which manipulates data received from hardware, it receives data then algorithm compares data and make decision to do performance.