

# 8. IOT TECHNOLOGY IN MODERN BUILDINGS

## 8.1. Introduction to IoT

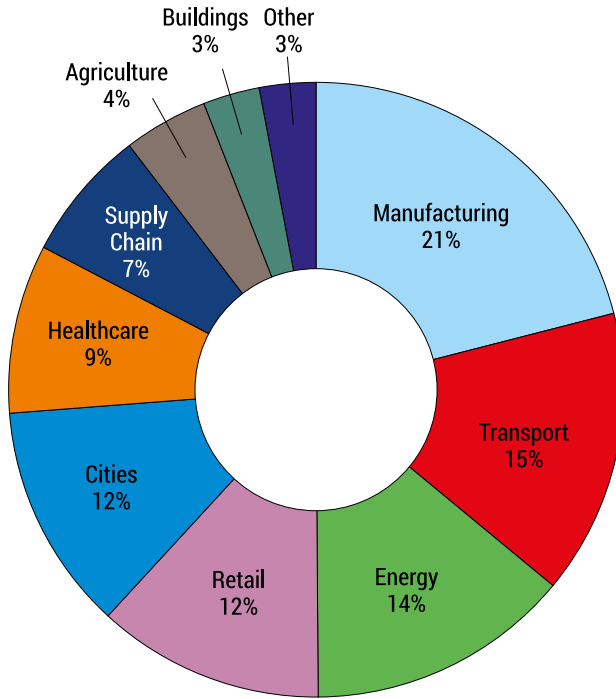
### 8.1.1. History of IoT

The Internet of Things consists of any device with an on/off switch that is connected to the Internet. The Internet of Things (IoT) involves machines communicating information over the internet, and has not been around for very long.

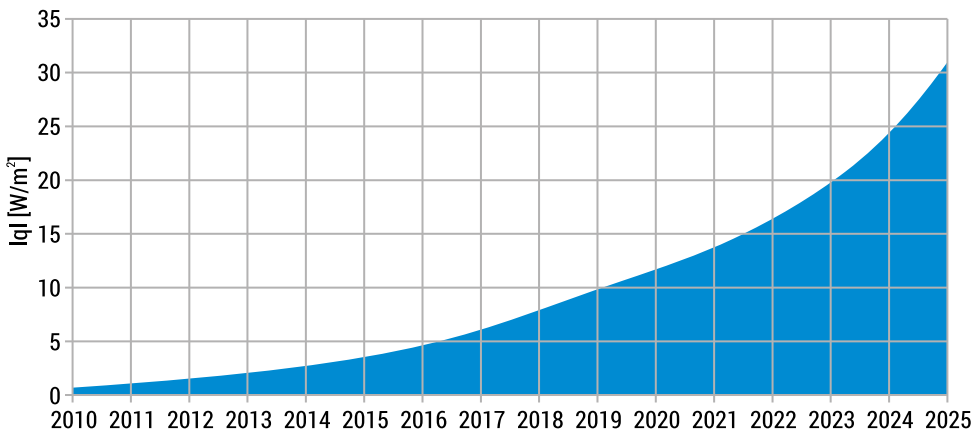
The term “Internet of Things” seems to owe its appearance to Kevin Ashton, who in 1997, working for Proctor and Gamble, for supply chain management has applied radio frequency identification (RFID) technology (Rayes & Salam, 2019). Thanks to this work, in 1999 he was invited to the Massachusetts Institute of Technology, where he and a group of like-minded people organized the research consortium Auto-ID Center. Mr. Ashton stated: „Today computers, and, therefore, the Internet, are almost wholly dependent on human beings for information. Nearly all of the roughly 50 petabytes of data available on the Internet were first captured and created by human beings by typing, pressing a record button, taking a digital picture or scanning a barcode. The problem is, people have limited time, attention, and accuracy. All of which means they are not very good at capturing data about things in the real world. If we had computers that knew everything there was to know about things, using data they gathered without any help from us, we would be able to track and count everything and greatly reduce waste, loss, and cost. We would know when things needed replacing, repairing, or recalling and whether they were fresh, or past their best.”

Since then, the Internet of Things has made the transition from simple RFID tags to an ecosystem and an industry, including smart buildings, medicine, energetics, agriculture, transportation and many more (Fig. 8.1).

The Internet of Things will take over virtually every segment in industry, business, healthcare, and consumer products. It is important to understand the implications and why these very different industries will be forced to change their approach to producing goods and providing services. The number of connected IoT devices is growing almost exponentially over the recent years (Fig. 8.2).



**FIG. 8.1.** IoT application areas in 2020 (Source: own elaboration based on WEB-1)



**FIG. 8.2.** Total number of IoT device connections (Nov 2020 with future estimates, Source: own elaboration based on WEB-2)

Consumer devices were one of the first categories of items connected to the internet. The consumer Internet of Things began with an internet-connected coffee maker at a university in the 1990s. It flourished with the spread of Bluetooth technology in the early 2000s. Now millions of homes are equipped with Nest thermostats, Hue

light bulbs, Alexa virtual voice assistant and Roku TV boxes. In addition, people use Fitbit bracelets and other portable devices. The consumer market is usually the first to adopt all new technologies. We can also consider these devices as gadgets. They all come neatly packaged and wrapped, and basically all of them are plug and play. Here are some examples of smart devices for the home: irrigation system, garage doors, locks, lights, thermostats, and security system.

### 8.1.2. The Structure of IoT

The IoT system starts with the simplest sensors located in the most remote corners of the world and transforms the analog physical exposure to digital signals. The data then travels through wired and wireless signals, various protocols, through natural interference and the imposition of electromagnetic fields, because of which they enter the Internet. From there, data packets are transmitted through various channels to the cloud or to a large data center. The strength of the Internet of Things is that it is not just a single signal from one sensor, but the sum of all signals from hundreds, thousands, perhaps millions of sensors, points, and devices.

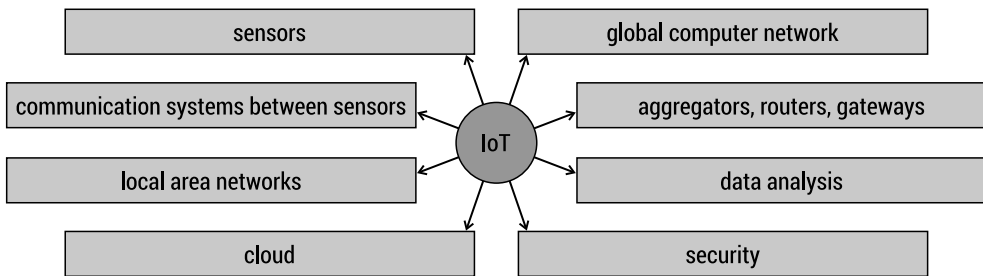
These areas are actively using the multitude of devices, software and services offered by the Internet of things. Almost every large technology company invests or has invested in the Internet of things. New markets and technologies have already emerged (and some of them have failed or been resold).

The IoT consists of (Fig. 8.3):

- **sensors:** embedded systems, real-time operating systems, uninterruptible power supplies, microelectromechanical systems (MEMS);
- **communication systems between sensors:** the coverage area of wireless personal networks is from 0 cm to 100 m. For data exchange between sensors, low-speed low-power information channels are used, which are often not based on the IP protocol;
- **local area networks:** usually these are IP-based communication systems, for example, 802.11 Wi-Fi network for fast radio communication, often these are peer-to-peer or star networks;
- **aggregators, routers, gateways:** embedded system vendors, lowest cost components (processors, DRAM, and storage), module vendors, passive component vendors, thin client vendors, cellular and wireless radio vendors, middleware vendors, fog infrastructure developers computing, edge analytics tools, edge device security, certificate management systems;
- **global computer network:** cellular operators, satellite operators, low-power global network operators (Low-Power Wide-Area Network, LPWAN). Internet transport protocols for IoT and network devices (MQTT, CoAP and even HTTP) are commonly used;
- **cloud:** infrastructure as a service provider, platform as a service provider, database developers, streaming and batch processing service providers, data analysis

tools, software as a service provider, data lake providers, software-defined network operators/program-defined perimeters, machine learning services;

- **data analysis:** huge amounts of information are transferred to the cloud. Working with large amounts of data and getting value from it is a task that requires complex event processing, analytics, and machine learning techniques;
- **security:** when bringing all the elements of the architecture together, security issues arise. Security touches every component, from physical sensors to CPUs and digital hardware, radio systems, and the communication protocols themselves. Security, credibility, and integrity must be ensured at each level. There should be no weak links in this chain, as the Internet of Things will become the main target for hacker attacks in the world.



**FIG. 8.3.** IoT structure (Source: own elaboration)

The IoT architecture, as we have already mentioned, covers many technologies. Each architect must understand what impact the chosen design solution will have on the entire system as a whole and each of its parts separately. The complexity and versatility of the Internet of Things is since this technology is much more complex than traditional technologies: it is distinguished not only by its large scope, but also by a combination of various, often unrelated interrelated types of architecture. The number of possible design solutions is amazing.

For example, at the time of this writing, there are more than 600 IoT platform providers (based on IoT Analytics' latest research) in the world offering cloud storage, SaaS components, IoT management systems, IoT security systems and any kind of data analysis. Add to this a huge number of different protocols for personal, local, and wide area networks, which are constantly changing and adjusting depending on the region.

Selecting the wrong protocol can lead to communication problems and noticeably poor signal quality, which can only be corrected by adding more nodes to the network. The architect must consider the interference in local and global networks: how is data taken from edge devices and transmitted to the Internet? The architect must evaluate the fault tolerance of the system and the cost of possible data loss. Which layer should be responsible for the fault tolerance of the system – the lower layers or the protocol layer? The architect must also choose Internet protocols: MQTT or CoAP and AMQP, and you also need to think about how all this will work if you switch to another cloud service (Perry, 2018).

You also need to decide at what point the data will be processed. At this stage, you can consider fog computing to processing data near the source, which solves the problem of latency and, more importantly, reduces network load and costs when transferring data over global networks and cloud services. Next, we consider all options for analyzing the received data. The wrong analytics tool can clutter up the system with redundant data or force you to use algorithms that require too much computing power to run on edge nodes. And how will requests sent from the cloud to the sensor affect the battery life of the sensor itself?

In addition to all this wide range of options, we must not forget about the security system, as the IoT system we have created becomes the largest target for attacks in the city. As you can see, the choice is huge, and each decision affects the others.

The Internet begins or ends with one event: a simple movement, a change in temperature, or maybe a lever snaps a lock. Unlike many existing IT devices, the Internet of Things is mostly associated with a physical action or event. It gives out a reaction to some factor of the real world. Sometimes a single sensor can generate a huge amount of data, such as an acoustic sensor for preventive maintenance inspections. In other cases, just one bit of data is enough to convey vital information about the patient's health status. Whatever the situation, sensor systems have evolved and, in accordance with Moore's law, have shrunk to sub-nanometer sizes and become substantially cheaper. This is what those who predict that billions of devices will be connected to the Internet of Things are appealing to, and that is why these forecasts will come true.

The Internet of Things would not exist without reliable technologies for transferring data from the most remote and unfavorable areas to the largest data collection centers of Google, Amazon, Microsoft, and IBM. The phrase "internet of things" contains the word "internet", so we must study issues related to networking, data exchange, and even signal theory. The basic pillar of the Internet of Things is not sensors or applications, but the ability to establish a connection. A successful architect understands the intricacies of interfacing a sensor to a WAN and vice versa (WAN to sensor interactions).

To transfer data from sensors to the Internet space, two technologies are needed: a router-gateway and basic Internet protocols that ensure the efficiency of data exchange. The router is especially important in aspects such as security, management, and data routing. Edge routers manage and monitor their respective mesh networks and align and maintain data quality. Data privacy and security are also of great importance. This part explains the role of the router in creating VPNs, VLANs and software-defined wide area networks (Perry, 2018). They can literally contain thousands of nodes served by a single border router, and to some extent, the router serves as an extension for clouds.

The architect must understand what the flow of data is, and in the typical schemes for building cloud services. To learn how to correctly assess how the system will develop and grow, it is necessary to understand all the intricacies and complexities of the architecture of cloud systems. The architect must also understand the impact latency has on the IoT system. Also, not everything needs to be sent to the cloud.

Sending all IoT data is significantly more expensive than processing it at the edge of the network (edge computing) or including an edge router in the area served by the cloud service (fog computing). Data that has been obtained by converting an analog physical stimulus into a digital signal can carry a lot of weight. This is where IoT analytics and rules engines come into play. The degree of complexity of putting an IoT system into operation depends on what solution is being designed. In some situations, everything is quite simple: for example, when you need to install a simple rules engine on an edge router that monitors several sensors that monitors anomalous temperature jumps. Another situation is that a huge amount of structured and unstructured data is transferred in real time to a cloud data lake, which requires high processing speed (for predictive analytics) and long-term forecasting based on high-tech machine learning models, such as a recurrent neural network in a signal analysis package with time correlation.

Many IoT systems will not be limited to the secure space of a home or office. They will be in public places, in very remote areas, in moving vehicles, or even inside person. The Internet of Things is a huge single target for all kinds of hacker attacks. We have witnessed countless mock attacks on IoT devices, well-organized hacks, and even nationwide security breaches.

## 8.2. Programming IoT Devices

### 8.2.1. Selecting Hardware and Software Environment with Arduino Example

The best way to learn is to learn by example. In this chapter we will complete some step-by-step examples using ESP8266 Arduino microcontroller, which is one of the most affordable and simple options for education purpose.

The easiest way to work with the ESP8266 is to send text commands through the serial port, because the chip was originally conceived as elementary data transceiver over Wi-Fi. However, this method is unsatisfactory and is not recommended. The better approach is to use the Arduino IDE, which you will have to install on your computer. It will do the job with the ESP8266 is very comfortable, because the Arduino IDE is familiar to everyone, who has ever dealt with Arduino based projects. This method will go throughout this chapter.

We will also need a suitable power source. This is often forgotten, which leads to many problems. If you try, for example, to power the ESP8266 chip from the +3.3V power supply built into an FTDI USB adapter board or an Arduino board, the device simply won't work properly. Therefore, most ESP8266 modules require a power supply that provides at least 300mA of current to operate reliably. Some boards have a microUSB connector and a built-in power supply, but this does not apply

to the boards discussed in this chapter. Therefore, a breadboard power supply that provides 500 mA on the line +3.3 V could be used here (McEwen & Cassimally, 2014).

After all components are properly connected (we will not go into details here but concentrate on programming) we need to install Arduino IDE. The latest version could be found here: <https://www.arduino.cc/en/software>. After installation of Arduino IDE adding of esp8266 platform is necessary (you can do it in the Settings dialog by installing additional package for this platform).

The next step is to test that the Arduino IDE and ESP8266 are working properly by connecting your module to your home Wi-Fi network. To do this, we will perform the following steps:

Write a program and load it into the module's memory. The program is very simple – we just want to establish a connection to the home Wi-Fi network and display the IP address that our board received in the terminal window. Here is the source code of the program:

```
// import of ESP8266 library
#include <ESP8266WiFi.h>
// your Wi-Fi network parameters
const char* ssid = "your_wifi_name";
const char* password = "your_wifi_password";
void setup(void)
{
  // serial port initialization
  Serial.begin(115200);
  // starting Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  // IP adress out
  Serial.println(WiFi.localIP());
}
void loop() {}
```

Do not forget to substitute the name and password of your Wi-Fi access point in the program source code. Save the program with any file name of your choice. Then enter the correct virtual serial port number to which your USB adapter is connected. Now you need to put the board into firmware download mode. To do this, connect

the GPIO0 pin to a common wire using the same conductor that was connected earlier. Then reboot the board by disconnecting and reconnecting power.

In the Arduino IDE settings set the speed to 115200 and click the button to upload the firmware to the board. Disconnect the wire between the GPIO0 and GND pins. Reboot the board by disconnecting and reconnecting the power. When the connection is established and the board has obtained an IP address, you will see a message like this:

```
WiFi connected 192.168.1.103
```

This message indicates that your card is connected to a Wi-Fi network.

## 8.2.2. Programming Fundamentals

Programming is directly related to the algorithm. Initially, an algorithm or sequence of actions is developed that can be performed by an automatic computing device (computer). Usually, this sequence is written (coded) with a certain system of formal notation called an algorithmic (programming) language, the text of the algorithm is called a program, and the process of creating it is called programming.

Programming – development of programs for a computer. Programming involves detailing the problem-solving algorithm and writing it down in the appropriate programming language, selecting and encoding the data structure, and debugging the created program. Programming language – a formal language used to describe computer programs.

### 8.2.2.1. Short History of Programming

Initially, all algorithms were written using machine language. This approach made the development of algorithms very difficult and very often led to errors that then had to be corrected. Thus, even the development of a simple program was a very complicated and labor-intensive process.

The first step in making it easier for a programmer was to stop using numbers to write commands exactly as they are used on a computer. To this end, the use of mnemonic records instead of numbers was introduced in the development of the program. Identifiers were used to describe some areas of memory, not the coordinates of the corresponding memory cell. These tools helped to create more understandable programs.

Initially, programmers took a mnemonic approach to developing software on paper and then translated it into machine language. However, it soon turns out that the process can be done by the computer itself. As a result, programs called assemblers were developed to translate mnemonic programs into machine language.

The assembler obtained its name because its purpose was to collect machine codes from commands obtained by transforming mnemonic symbols and identifiers. Mnemonic recording systems can now be considered as special programming



languages – assembly languages. At one time, the development of assembly languages was a big step forward in the development of programming languages. These languages are called second generation languages (the first generation was machine languages). Despite their advantages over machine languages, assembly languages could not provide a complete programming environment. In addition, the constructions used in these languages were the same as those used for machine languages. The only difference was in the form of syntax. Assembly languages are also machine-dependent, so to run this program on another computer, you had to start (overwrite) it for the appropriate configuration of the registry and command list. The programmer still had to think of machine language concepts that are not necessarily used to create the program.

There are four independent directions in which programming languages have developed:

- procedural,
- declarative,
- functional,
- object-oriented.

The direction of the procedure is the traditional approach to programming. This approach defines the programming process as a sequential command record, in the execution of which the data will be processed to obtain the required result. In this case, the primary ones are the teams. With this approach, it is best to start learning programming, because it is more understandable to a person, but for creating large programs, the direction of the procedure is practically not valid.

In an object-oriented approach (OOP), a data element is considered an active object as opposed to a traditional approach. An object contains not only data, but also operations that can be performed with this data. OOP offers the ability to use objects multiple times, which facilitates program development, as well as object creation libraries that allow programs to be collected as a constructor.

In the declarative direction, the main question is “what is the task”, not what algorithm will be needed. The problem is to find a common algorithm for solving the whole range of tasks. In this case, the programmer must formulate the task precisely, not look for an algorithm. Initially, declarative languages were designed to solve a very narrow range of specific tasks.

The functional approach considers the program development process as construction from “black boxes”, each of which receives input data and outputs output data. Mathematicians call such boxes functions. Functional languages consist of elementary functions based on which a programmer can build the most complex functions. The programming process is the construction of the necessary functions with other simple functions embedded in each other. The advantage of the functional paradigm is the modular approach to programming. In this case, the program is more organized compared to the procedural approach. This is like building a program from building blocks, not from scratch.

### 8.2.2.2. Variables, Data Types and Arithmetic Operators

A variable is an area of the computer's memory where a value used by a program can be stored. Variable – a character or sequence of characters used to denote a saved value that has its own name and value and can be changed during program execution. All variables must be specified with their data type and name before they can be used in the program. Variables of the same type can be defined in one or separate rows (all examples are given in C++ programming language).

```
int a; //variable definition
```

```
int a, b; //definition of 2 variables with same type
```

The variable name must meet certain requirements:

- letters may use letters, numbers, and underscores;
- the name can only start with a letter or an underscore symbol;
- C ++ reserved words may not be used in names.

In C ++, names of variables of any length are allowed, but shorter names are easier to remember and write. It is recommended that variable names reflect their meaning, such as sum or number1, and so on. The C ++ language distinguishes registers, so variables a1 and A1 are not the same. You can define variables in any row of the main function, but you must use these variables. It is also recommended to insert a blank line before defining variables.

An example let us consider a two-number counting program. The text of the program code is as follows:

```
#include<iostream>
#include<conio.h>
using namespace std;
void main()
{
int sk1, sk2, sum;
cout<<"Input first number\n";
cin>>num1;
cout<<"Input second number\n";
cin>>sk2;
sum=sk1+sk2;
cout<<"Sum = "<<sum<<endl;
getch();
}
```

This program allows the user to enter two numbers, then calculates the sum of these numbers and displays the result on the monitor screen.

### 8.2.2.3. Data Types, Arithmetic Operators, and Flowcharts

The data type of an object is determined during its creation (definition) and denotes the values that can be accepted by the given type of object (integers, logical values, dates, etc.) and the operations that are possible with this type of object. The data type also determines the amount of memory allocated during variable definition.

C ++ supports the following groups of data types:

- base types – are indicated with the help of reserved keywords; the types themselves do not need to be defined;
- user-defined types – structures, indicators, arrays that must be defined before use.

Only base data types will be considered in this collection. In many programming systems, the size of the int data type corresponds to the size of the “machine name”. For example, on a 16-bit computer, the int size is 16 bits and 32 bits are 32 bits. The use of different types of data allows you to use computer resources more efficiently, which in turn speeds up program execution.

Many programs perform arithmetic operations. C ++ uses several symbols that are different from those accepted in algebra. Table 8.1 gives the arithmetic operators of the C ++ language.

**TABLE 8.1.** C++ arithmetic operators (Source: own elaboration based on Deitel, 2018)


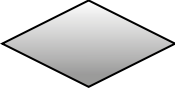



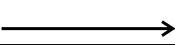

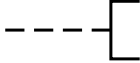

Operation name	Arithmetis operation	Example from algebra	C++ example
Summation	+	$F + 7$	F+7
Subtraction	-	$F - 7$	F-7
Multiplication	*	BM	B*M
Division	/	$X / Y$ vai $X : Y$	X/Y
Residual calculation	%	$R \text{ mod } S$	R%S

Arithmetic expressions in C ++ must be written on one line to be executed. Round brackets are used for the same purposes as in mathematics. The operations in parentheses are performed first, followed by the multiplication, division, and balance residual operations, and finally the addition and subtraction (operation priority).

The graphical implementation of algorithms is more compact and more pseudo-coded. The algorithm is represented as a sequence of interconnected blocks, where each of the blocks corresponds to one or more operators. This type of graphical representation is called flowcharts. Table 8.2 gives the most widely used flowchart elements description.

Later in this chapter, specific examples of algorithm representation using flowcharts are discussed.

**TABLE 8.2.** Basic flowchart elements (Source: own elaboration based on Deitel, 2018)

Name	Element graphical representation	Description
Process		Execution of an operation or group of operations that results in a change in value, display format, or data layout.
Decision		Choice of the direction of execution of the algorithm depending on some variable conditions
Defined process		Use of previously developed and separately described algorithms or programs
Input/Output		Use of previously developed and separately described algorithms or programs
Document		Output data to a printer or other similar device
Connection line		Indication of links between flowchart objects
Connector		Indication of interrupted flow lines (connection of interrupted lines)
Comments		Link between scheme element and explanation
Start/End		Indication of interrupted flow lines (connection of interrupted lines)

### 8.2.2.4. Control structures

Program operators are usually executed one after the other in the order in which they were written. This is called sequential execution. However, some operators allow you to change this order. These are transfer of control operators. In the 1960s, the unrestricted use of goto operators proved to cause many errors and prolong the program execution process. It was believed that the goto operator, which allowed the programmer to hand over control over a very wide range, was to blame.

Again: //label, where the program will return

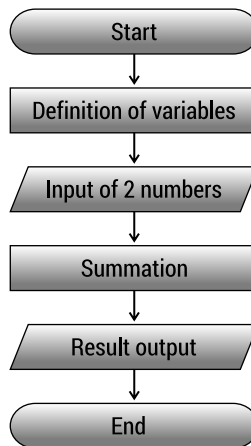
...  
goto Again;

The idea of structuring the program was based on the principle of “Do not use goto”. In the 1970s, it was proved that programs could be written without a goto operator at all (Deitel, 2013). The results of this approach have been impressive: the execution time of the program has been reduced as the programs have become more comprehensible and the number of errors has decreased significantly.

It turns out the program can be written using only three control structures:

- sequence structure;
- selection structure;
- repetition structure.

Sequential execution is built into programming languages. Unless otherwise specified (by default), all operators are executed in the order in which they are listed. The program example discussed above are implemented with sequential execution. The flowchart of the two-number counting program is given in Figure 8.4.



**FIG. 8.4.** The flowchart of the two-number counting program (Source: own elaboration)

The first control structure is conditional one. A simple if structure performs an equality or relationship test. The type of record is as follows:

if (condition) action;

If the condition is true, then the action that follows it (one or more operators) will be executed, otherwise the (false) action will not be executed.

An example let us consider a program in which the user must enter two numbers, but the program compares them and outputs the result of the comparison.

```

#include<iostream>
#include<conio.h>
using namespace std;
  
```

```

void main()
{
int sk1, sk2;
cout<<"Input the first number\n";
cin>>sk1;
cout<<" Input the second number \n";
cin>>sk2;
if (sk1>sk2)
cout<<sk1<<" is greater than "<<sk2;
if (sk1<sk2)
cout<<sk1<<" is less than "<<sk2;
if (sk1==sk2)
cout<<sk1<<" equals to "<<sk2;
getch();
}

```

In this case, the program contains three conditional control operators that consider three comparison cases. The program can output only one comparison result, because in any case only one condition will be true. By placing the semicolon immediately after the condition, the user makes an error in the empty condition test operator (the action that goes immediately after the condition test will be executed in all cases, and the program will output an incorrect result).

The if / else control structure is shown in an example of a quadratic equation solution (code example and a flowchart in Figure 8.5).

```

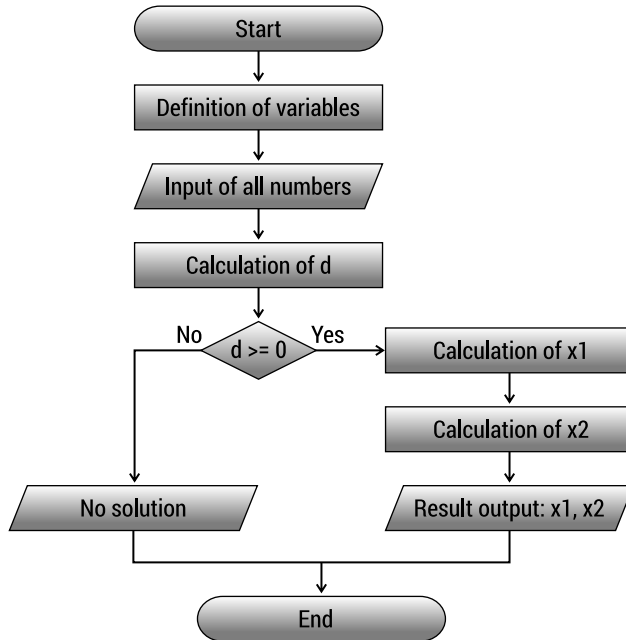
#include<iostream>
#include<conio.h>
#include<math.h>
using namespace std;
void main()
{
float a, b, c, d, x1, x2;
cout<<"Input coefficient a\n";
cin>>a;
cout<<" Input coefficient b\n";
cin>>b;
cout<<" Input coefficient c\n";
cin>>c;
d=b*b-4*a*c;
if (d>=0)
{
x1=(-b+sqrt(d))/(2*a);

```

```

x2=(-b-sqrt(d))/(2*a);
cout<<"The solution is "<<x1<<" and "<<x2;
}
else
  cout<<"No solution";
getch();
}

```



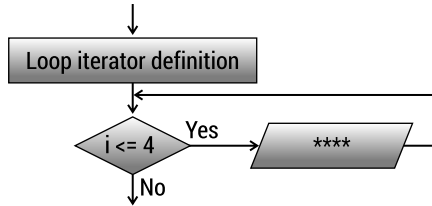
**FIG. 8.5.** The flowchart of quadratic equation program (Source: own elaboration)

Repetition structures (loops) allow a programmer to define an action that must be performed until a condition is met. Example of a loop while is as following (for simplification the only loop part of a program is given, see Figure 8.6):

```

...
int i=1;
while (i<=4)
{
  cout<<"****\n";
  i=i+1;
}
...

```



**FIG. 8.6.** The flowchart of while loop outputting 4 asterisks (Source: own elaboration)

The result of the program will be as follows:

```

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

The program shown in the example loops four times. A special variable called a cycle counter can be used for this purpose. In this case, it is a variable  $i$ , the initial value of which is 1, but with each subsequent iteration of the cycle, it increases by one (the last program line). The condition in while indicates how many times the loop will run (in this example, until  $i$  is equal to 4). As a result, 16 asterisks will be displayed on the monitor screen, four in each line.

The iteration structure for is like the while structure and contains the same elements, the difference being the notation. When this structure begins to execute, a control variable (cycle counter) is defined and assigned a value. The condition for continuing the cycle is then checked and, if true, the operation is executed (iterated). Then increment the counter and check the condition again.

The example above with asterisks for the loop looks like this (the flowchart and result are identical to the while loop):

```

for (int i = 1; i <= 4; i++)
  cout << "**** \n";
  
```

It should be noted that in the example the operation  $i \leq 4$  is used in the cycle conditions. If  $i < 4$  is recorded, the cycle will be executed only three times. This is a typical logical error. If the loop counter is defined in the loop header (in parentheses), then this variable must not be used after the end of the loop (syntactic error) because it will not be known to the program. This constraint is called the scope of the variable.

It is possible to use the break and continue operators in cycles. When the operator break is executed in cycles, the cycle exits immediately, but the program continues to execute subsequent operators after the cycle. Typically, the break operator is used to end a loop, but the continue skips operators that are immediately after it, and the iteration of the next loop is executed.

Depending on the task, the loops can be placed in each other in unlimited quantities (nested loops).



## 8.3. Automation Concept and Implementation in Smart Houses

### 8.3.1. Automation Concept

Home automation or smart house is an extension of building automation and includes the control and automation of lighting, heating, ventilation, air conditioning (HVAC), and security, as well as home appliances such as washer/dryers, ovens or refrigerators that use WiFi for remote monitoring. Modern systems generally consist of switches and sensors connected to a central hub from which the system is controlled with a user interface that is interacted either with a wall-mounted terminal, mobile phone software, tablet computer or a web interface, often via internet cloud services.

The terms “Home Automation” and “Internet of Things” are distinct parts of the Smart Home concept: where a home’s electrical devices are connected to a central system that automates those devices based on user input.

These are electrical devices that are intelligent, courtesy of a connection to the Internet. These devices are capable to assist a user needs. This intelligence comes from device programming, but with time the device can learn and adapt to patterns and interact with its users thanks to artificial intelligence algorithms and methods.

Automated home security systems tend to offer a wider range of features than their predecessors. Systems respond to voice and biometric data, and locks can be upgraded to keypads that are opened with codes or swipe cards. These systems can be turned on or off via remote control, email, or phone, and camera feeds can be sent directly to one’s computer. Audible alarms can be used to alert one of intruders, while silent alarms can be used to alert the authorities.

IoT is going to give the net a more objective way of gathering data, meaning that the conclusions that will be drawn by AI and Machine Learning algorithms are potentially going to be very different from what we expect. It’s also going to mean that the scale and quality of decisions being made by virtual agents in future is going to become better than today.

The smart house is a complex concept which integrates diverse technologies to resolve the new problems in cities and rural area. From one side, new technologies enable new applications such as intelligent transportation and smart building management. From the other side, the new applications raise challenges for the current technology. For example, the increasing number of autonomous systems such as self-driving cars and robots need high accurate real-time location information service and context awareness and anomaly detection for decision making. There is also a strong need of better human physical activity information for the purpose of improving security, understanding the human behavior models and health condition monitoring.

Smart homes utilize sensors and controllers to monitor and automatically trigger services to save valuable time in cases of emergency (e.g., fire, intrusion, or gas leak). With the smart building system, services like video monitoring, light control, air-condition control, and power supply control are often managed from the same control center.

The main aspects of a smart buildings IoT technologies are (Sun et al., 2018):

- Safety monitoring and alerting: Examples include noise level monitoring in urban zones and sounding alarms in real-time, electromagnetic field level monitoring by measuring the energy radiated by cell stations and other devices, chemical leakage detection in rivers by detecting leakages and wastes of factories in rivers, air pollution and control of CO<sub>2</sub> emission factors, pollution emitted by cars and toxic gases generated in farms, as well as earthquake early detection.
- Smart lighting: here IoT is used to minimize energy consumption, to provide weather adaptive lighting in streetlights, and to automate maintenance.
- Flooding, water leakage, and pollution monitoring: monitoring of safe water levels in rivers, lakes, dams, and reservoirs. Detection of the presence of toxic chemical. Monitoring of tanks, pipes, and pressure variations. Real-time control of leakages and waste in the sea.
- Detection of hazardous gases and radiation levels: Detection of gas levels and leakages in and around industrial buildings and chemical factories. Monitoring of ozone levels during the meat drying process in food factories. Distributed measurement of radiation levels in the surroundings of nuclear power stations to generate leakage alerts.
- Other use cases include detection of garbage levels in containers to optimize the trash collection routes, preemptive monitoring of burning gases and fire conditions to define alert zones, snow level measurement to know in real time the quality of ski tracks and alert avalanche prevention security corps, monitoring vibrations and earth density to detect dangerous patterns in land conditions, and monitoring of vibrations and structural conditions in buildings and bridges.

### 8.3.2. IoT in Smart Houses

The operation of a smart home system is not as simple as it seems. Before the advent of smart home technology, each installation in a building had only one specific task and had its own automation with separate controllers and remote controls. The boiler heated the house and heated water for domestic needs, the alarm system ensured the safety of residents and their property.

The problem with various controls came at a time when homeowners began to use more modern installations and equipment, such as ventilation and air conditioning, smart lighting, multimedia, blinds and external awnings, automatic watering. Engineers and electronics have joined forces and developed technologies that combine

various household appliances into one system and provide uniform control. However, easier control is just one of the many benefits of new technologies in human life.

With this technology under your own roof, you get (Perry, 2018):

- Reducing housing maintenance costs. From an economic point of view, a properly tuned installation effectively reduces electricity and gas consumption. It is estimated that single-family homes save up to 30% of these energy sources. A house that works with motion sensors can always turn off the lights when it detects that no one is in the room. If the windows are open, it makes no sense to use ventilation and heat the premises, and when the owners leave and the window is left open, a notification will come about this, and the heating and ventilation mode will change so that they do not incur unnecessary costs.
- Feeling of safety and comfort. From a security point of view, the most important advantage of the „thinking” technology is its efficient operation, due to the ability to exchange information between all related elements. If events are possible in the building or outside that potentially dangerous for residents, they immediately respond to them by initiating alarm, security, and containment procedures. Ease of use lies in the remote performance of several actions to control and manage for users.
- The pleasure of spending time at home.
- Free expansion of the system and its adaptation to constantly changing conditions and lifestyle requirements (provided that the home is equipped with automation that can be expanded in the future).

The principle of operation of a smart home is to control a computer (advanced controller) that collects, and processes signals received by it from various sensors located inside or outside the house. Electrical cables or radio waves are used to send signals. Sensors (motion, humidity, pressure, light intensity, flooding, smoke, carbon monoxide, sleep gas, etc.) register changes inside and outside and transmit them to the central unit. There are also systems without a central unit, in which each element decides for itself which signals from the detectors are relevant to it. Any change that occurs in one subsystem is immediately considered in another. All subsystems take this into account and respond immediately to the change.

High flexibility is one of the most important performance parameters when choosing a home automation system. Thanks to this, you can reprogram it many times and adapt it to the current needs of residents, expand and add new devices and functions in the future.

**Closed systems:** offer products (technologies, programs, devices, etc.) from only one manufacturer. When choosing this type of system, users cannot use products from other companies in their work.

**Open systems:** software, devices, and accessories for them are produced by many programmers and many companies. One popular open system suitable for single-family housing is, for example, the KNX system, formerly known as EIB. All products

used in this system are compatible, although they are manufactured by more than 100 companies (Sun et al., 2018).

Systems don't come cheap, so it's best to make sure they're fully tailored to the individual requirements of each family member and the home they live in. Before embarking on a project, it is worth exploring the possibilities of several systems, open and closed. An intelligent installation based on traditional cables can work in the building under construction. On an already built object or after repair, it is easier to set up wireless radio systems. Although the latter, as a rule, are more expensive than regular ones (cable).

How to lay the cables, where to place the sensors and all other necessary components (e.g., drive motors, servo motors, controllers, buttons) must be specified in detail by the project author. It is best if it contains several detailed designs for specific installations – electrical, signaling, multimedia. Before starting the preparation of the project, an in-depth analysis of the needs and requirements of the residents is necessary.

For users of single-family buildings, a mandatory function of intelligent automation is the remote control of heating and electrical systems. They are also usually willing to include systems responsible for the safety of the home and its inhabitants, as well as ventilation and air conditioning.

**Temperature and humidity control** in a smart building is based on the operation of temperature sensors located in each room, a weather station on the street and humidity sensors. When users leave the house, the system lowers the temperature, closes windows, and manages ventilation and air conditioning more efficiently. After the family returns, the elements begin to work in a different mode, quickly bringing the room to a comfortable temperature and humidity. The devices can be programmed in economy mode for rooms that are rarely used and at night, and rest mode for when traveling.

**Lighting:** in the premises, the lamps are switched on or off remotely, shine brighter or dimmer thanks to motion sensors and light intensity sensors. Outside (in the garden, on the terrace, at the gate and gate), the lighting points are controlled by twilight sensors. In many smart homes, you set up so-called light scenes from selected lighting points. They are used wherever it is worth creating individual lighting, in the living room, kitchen, corridors, halls. However, the so-called light trails are designed to provide efficient passage through the premises, access to the garage and entrance to the house, or when residents need to go to the toilet at night (dim light or soft LED light is usually used then). In addition, with the help of light it is possible to simulate the presence in the building, this function is directly related to the security function.

**Alarm and monitoring:** the smart system integrates with building alarms or monitoring. They are indispensable for protection against accidental events caused by fire, gas leakage, etc. The safety of family members and the building can be protected with the help of numerous motion, smoke, gas, water leakage sensors. The panic button, installed next to the owner's bed, triggers a series of events – it sends information to the security service, it can signal or turn on the light in the building. An interesting security feature is access control. Residents use an individual card or key that

is recognized in their home. A useful solution is a special button, with which, when leaving the house, you can turn off all electrical appliances that you do not need in our absence (of course, except for the refrigerator, boiler, etc.). With one button, we get 100% confidence that the iron, oven, coffee maker, induction cooker will stop working. The driveway to the entrance and garage gates, video intercom and lighting of the path leading to the house and driveway are the main elements of automation in the field of access control.

**Blinds, awnings:** in summer, when the sun is too strong, the remote control allows you to facilitate the operation of ventilation or air conditioning by automatically lowering blinds or awnings over terraces and balconies. In winter, lowering the outer blinds further protects the interior from freezing and maintains heating.

**Multimedia:** a huge number of users include in the intelligent system the so-called multi-room, that is, a multi-zone sound system and a home theater. By integrating a multi-room system with speakers and controllers installed in each room, they can use state-of-the-art audio equipment throughout the building. The sound “wanders” with them from room to room.

**Watering:** automatic watering starts after the information is sent to the controllers using dusk and soil moisture sensors. A motion sensor near the sprinkler protects garden users from unwanted wetness, while a rain sensor blocks irrigation during prolonged rain.

**The septic tank and rainwater tanks:** full level sensors in a septic tank, combined with a smart home, keep dirt out of the area. They inform owners (e.g., via SMS) of the need to call the slurry tanker. They are also good in rainwater tanks.

### 8.3.3. Technological Aspects of Smart Houses

The most popular smart home devices use Wi-Fi, Bluetooth, ZigBee and Z-Wave technologies. Each of the technologies has its pros and cons, and no one forbids using them together, compensating for the shortcomings of each. But different technologies are used for different tasks and different types of smart devices. For example, household appliances (TV, refrigerator, and coffee maker) usually use Wi-Fi or Bluetooth, which are also found in any phone. The reason is that this technique is used even without a full-fledged smart home system. For lighting and climate automation, ZigBee or Z-Wave embedded modules are more suitable, as they are specifically designed to integrate with existing lighting and climate equipment. But for their full-fledged work, a special hub is needed.

**Wi-Fi** is indispensable in IP cameras, TVs, audio / media players and other video signal transmission equipment. Of course, Wi-Fi can also be used in light switches, sensors, thermostats, but the lack of signal relay and high power consumption do not allow making sensors that work for years on it. Each manufacturer for their Wi-Fi device, whether it's a smart light bulb, a kettle, a refrigerator or a robot vacuum cleaner, releases its own application, and there is no single standard to control all appliances

from one application. This does not allow making a Wi-Fi-only smart home truly convenient.

The current version of **Bluetooth** Low Energy 4.2 has low power consumption, thanks to which tiny wireless headphones, speakers and various battery-powered sensors work (Perry, 2018). The problems here are the same as with Wi-Fi: the lack of a common control standard forces each manufacturer to make its own application, which is inconvenient for the user. Mesh technology (mesh network), which is very important for a smart home, appeared only in version 5.0, which is still rarely used anywhere, but perhaps the future of smart homes is in Bluetooth LE 5.

**ZigBee** was originally developed for use in networks of sensors such as electricity, water, gas meters, temperature sensors. The network topology can be different, including cellular (mesh). This means that any sensor sees all other sensors and can transmit a signal through them, i.e. use relaying, which greatly increases the reliability of transmission. In 2007, a command standard for managing a smart home appeared, the so-called „Home Automation” profile (Perry, 2018). With ZigBee, almost all devices for creating home automation are released: relays, dimmers, lamps, thermostats, locks, sensors. But you will not find household appliances such as refrigerators and TVs with ZigBee. Compared to other smart home protocols, ZigBee devices have the most attractive prices, but the lack of 100% compatibility between devices and hubs from different manufacturers does not allow building a smart home only on ZigBee.

**Z-Wave** is a wireless protocol developed since 2001 specifically for home automation. Its main advantage is full compatibility between devices from different manufacturers. So the motion sensor from Fibaro can control the Qubino dimmer, and all the automation is based on the RaZberry controller from Z-Wave.Me. At the moment, more than 3,000 different Z-Wave devices are being sold, which cover all the needs of a smart home. This is the most popular protocol for objects ranging from 10 to 500 m<sup>2</sup>. Z-Wave, like ZigBee, uses a mesh topology with support for signal relaying and automatic finding of the best route. The main disadvantage is the price. On average, the cost of the device is 60-80 euros, which is about twice as high as that of analogues with ZigBee.

Every year, smart home technologies are gaining more and more fans. The range is growing rapidly, and there really is plenty to choose from, it is not even necessary to order abroad. When you start planning the assembly of a smart home, the question usually arises, what manufacturers are on the market? Whose decision to choose? How is one different from the other?

**Xiaomi** is one of the most popular smart device manufacturers. Their assortment includes almost all household appliances connected to a smart home, as well as specialized IP cameras, sockets and light bulbs, various sensors (temperature, humidity, CO<sub>2</sub>) and many other devices. Xiaomi does not use any one wireless technology for its devices but chooses the best one for each type. For example, ZigBee is used to control lighting, sockets, and curtains, and to connect them, you definitely need a hub from Xiaomi with support for this protocol. TVs, vacuum cleaners, and IP cameras are connected via Wi-Fi through a router, because not everyone needs

a full-fledged smart home, and almost everyone has Wi-Fi. Temperature, humidity, air quality sensors and locks work via Bluetooth. Such devices can be connected directly to the phone and only view readings, or they can be connected to a Xiaomi hub with Bluetooth support, then it becomes possible to use the sensor in climate control scenarios.

Smart home from Xiaomi is a great solution, because the company offers many good devices for creating a smart home and convenient automation settings. But the Xiaomi hub does not allow you to set up complex automation and use scripts. Only ZigBee devices can be controlled from a phone without the Internet (via a hub), but Wi-Fi lamps and sockets work only via the Internet. ZigBee hubs from other manufacturers allow you to remove these restrictions.

**Apple** doesn't make smart home devices, but it did create the HomeKit protocol that other manufacturers use to build compatible devices. HomeKit devices work over Bluetooth and Wi-Fi protocols. Locks, thermostats, lighting control modules, RGBW lamps, cameras and many sensors are available with HomeKit support. In addition to devices, there are also gateways that convert commands from ZigBee and Z-Wave devices into HomeKit commands. Xiaomi, Ikea, Philips and many others have such gateways. This expands the range of smart home devices from Apple. So far, Apple's automation capabilities are very modest and do not allow you to create a completely arbitrary scenario. Also HomeKit, for obvious reasons, is not suitable for Android users.

**Fibaro** is the manufacturer of the most popular Z-Wave devices and home automation centers. All Fibaro devices have many settings and additional features. The equipment line includes opening, motion, leakage, smoke sensors, micro-relay modules, dimmers, etc. The Home Center 2 home automation controller has a pleasant and intuitive user interface and allows you to configure scenarios of any complexity. Because since the Z-Wave protocol provides for compatibility between devices from different manufacturers, Fibaro can work with any other Z-Wave devices.

## References

- [1] Ammar, A., Salam Samer (2019). *Internet of Things from Hype to Reality. The Road to Digitization. Second Edition.* Springer: Cham, Switzerland.
- [2] Deitel, P., Deitel, H. (2013). *C++ How to programm. Ninth Edition.* Pearson Publishing.
- [3] *From Internet of Things to Smart Cities (2018).* Edited by Hongjian Sun, Chao Wang, Bashar I. Ahmad. CRC Press: London, New York.
- [4] McEwen, A., Cassimally, H. (2014). *Designing the Internet of Things.* John Wiley and sons: Chichester, West Sussex.
- [5] Perry, L. (2018). *Architecting Internet of Things.* Packt Publishing: Birmingham, Mumbai.
- [6] Rayes, A., Salam, S. (2019). *Internet of Things From Hype to Reality. The Road to Digitization. Second Edition.* Springer: Cham, Switzerland.
- [7] WEB-1: [Online] Available from: <https://iot-analytics.com/top-10-iot-applications-in-2020/>
- [8] WEB-2: [Online] Available from: <https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/>