International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

Real Time Control of Robotic Arm Using Bluetooth Low Energy & Wi-Fi with the Module Board ESP32 and Android Application

Ing. José Eduardo Tzompantzi Netzahual[1], M.D.S. Higinio Nava Bautista[2], M.C.C. María Janai Sánchez Hernández[3]

[1, 2, 3] Tecnológico Nacional de México, Instituto Tecnológico de Apizaco, Fco. I Madero s/n, Barrio de San José, 90300, Tlaxcala Mexico

Abstract: ***Currently many companies around the world strive to design and produce new hardware platforms that can be used more easily in the construction of prototypes for the development of technologies such as IoT. This paper presents an alternative to make prototypes of hardware and software adapted to the educational and industrial environment, making use of wireless technologies Bluetooth and Wi-Fi, with a module based on the SoC ESP32 system and a native application in Android. To demonstrate the functionality and accessibility of the platform, the design of an electronic circuit was implemented to a robotic arm that was used for didactic purposes for the learning of topics related to Electronics and the Programming of Microcontrollers for IoT. In addition the ESP32 module is configured to broadcast a SSID similar to a wireless router so any user with access credentials can connect to the robotic arm module via Wi-Fi. Using Bluetooth only one user can establish a point-to-point connection from an Android device with the ESP32. In the same way the control of the robotic arm is done manually and by means of the elements of the graphical interface, with this one manages to have a mobile application of easy use and access that allows the control of electronic objects.***

Keywords: Robotic arm, ESP32, Android Application, Bluetooth Low Energy, Wi-Fi, Microcontrollers, IoT.

1. Introduction

IoT promises to revolutionize the world in the near future, its objective is to bring simplification to different field, we cite robotics, electronics, programming as the most popular ones. [1]. Due to the importance of these fields in the world of today, where almost everything is regulated by technology [2], the IoT applications begin to develop and in some cases are in experimental stages. In order to create similar products, the first step is to make a prototype of work, however it requires certain skills and knowledge in the area of electronics and programming where only some are dedicated to experiencing this trend. This is why different products have emerged that aim to simplify these two fields and make them accessible for a larger audience even if they don't have a technical background [3]. A very clear example is Arduino [4], which today is one of the most popular educational platforms among people who just start with electronics, where you can control almost any object using mobile devices as remote control. The Company offers a range of software tools, hardware platforms and documentation that allow to carry out electronic projects, extending its functionalities with other development cards, as is the case of the low consumption ESP32 system [5] used in the construction of the present prototype.

With the massive penetration of smartphone devices in everyday life, mobile app development becomes a popular trends. Many literature studies said that in largely, commercial smartphones which available at market are installed by Android operating system: an open source OS which are profitable to the developers and also bring some benefits to the users [6]. According to the official website of Android [7] there are more than 24.000 unique devices, made by 1.300 different brands and even more accessible prices. The unprecedented offer of Android does not end with devices, Android has the advantage that as it is free software offers everything needed for free development of applications, which according to site statistics [7] there are more than 1 million of applications available only in Google Play Store.

Just as devices, smartphone technologies have evolved enormously. Not only are they using GSM/UMTS mobile telephony networks, but also standard wireless Bluetooth and Wi-Fi technology, to exchange data with other electronic devices. The wireless control is one of the most important basic necessities and as mentioned above, it is intended to show a functional solution, accessible and of good performance, to carry out the control of a robotic arm using the characteristics of the module and the hardware of a mobile device.

2. Proposed System

Microside S. A de C. V is a company that is dedicated to the design, production and commercialization of technological solutions oriented to the Internet of Things (IoT). In recent days they have designed a hardware solution that uses a module of the ESP32 series as a base. In order to be able to experience the functionality and accessibility of this platform, the module was integrated into a manipulatable robotic arm of 4 degrees of freedom. The built in electronic prototype includes a serial port that allows the user to connect the circuit to an external power supply, or to a computer to be able to load the programmed code from the Arduino IDE to the ESP32 module. The ESP32 module is responsible for initializing wireless connectivity in dual mode and executing the instructions programmed in the movement of the robotic arm. Up to this instance the

prototype can be programmed to perform motion sequences independently, but it lacks an interface that allows user interaction with the set of robotic arm servomotors, and facilitates connectivity to the ESP32 module by using a mobile device. Given this situation, and to take full advantage of the hardware of the device, developed a native application based on Android, to establish connectivity with the ESP32 and make it send data, as well as receiving them using the HTTP protocol and Bluetooth.

3. Hardware Description

3.1 ESP32

In general the ESP32 is a low-power system on a chip (SoC) series with 2.4 GHz Wi-Fi and Bluetooth capabilities, designed for mobile, wearable electronics, and Internet of Things (IoT) applications. The ESP32 is a highly integrated solution, with around 20 external components. Integrates an antenna switch, RF balun, power amplifier, low noise receive amplifier, filters, and power management modules. As such, the entire solution occupies minimal Printed Circuit Board (PCB) area [8].

Figure 1: Examples of ESP32 Development boards [9]

An important aspect to consider about the ESP32 is that it's capable of functioning reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C [10].

3.2 ESP32-based Module

As shown in Figure 1, currently on the market are commercialized various development kits ESP32, it is worth mentioning that the ESP32 can be found as a standalone module or as a development board with all its functions and features. As for the construction of the prototype was integrated a module of the series ESP-WROOM , in a development plate distributed by Microside S. A de C.V in Figure 2.

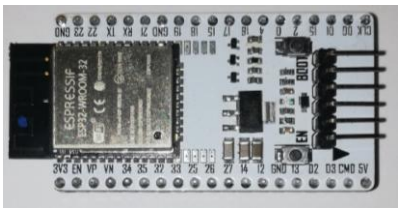

Figure 2: ESP32-based Module developed by Microside S.A de C.V

3.3 Technical Specifications ESP32-based Module

- Dual-core 32-bit Processor.
- 160Mhz Speed (maximum 240 Mhz).
- 520 KB of Memory SRAM.
- Up to 16 MB of external Memory flash.
- Wi-Fi 802.11 b/g/n 2.4GHz.
- Bluetooth v4.2 BR/EDR y BLE.
- UART, SPI, I2S, I2C.

3.4 Robotic Arm

In this case to be able to demonstrate the functionality and wireless connectivity of the electronic module, a kit of a didactic robotic arm was used as shown in Figure 3, which consists of 4 Tower Pro SG90 servo motors: one in the base, one in function of shoulder, one as elbow and the fourth for the opening and closing of the clamp.

Figure 3: Robotic Arm used in the construction of the prototype

The Tower Pro SG90 is a micro servo that works with most electronic control cards with microcontrollers. Usually these small servos operate on 5V and the control is made by a PWM control signal that generates the microcontroller, in which the width of the pulse indicates the angle that we want to adopt the axis in Figure 4 [11]. The servos are controlled by sending a variable-width electric pulse or pulse-width modulation (PWM) through the control cable. Therefore it can be controlled by PWM pins or by any digital pin [12] of microcontrollers like Arduino, PIC, ARM or even a microprocessor like Raspberry Pi.

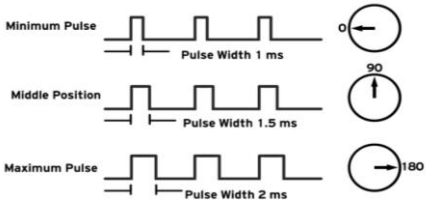

Figure 4: Pulse width control (PWM) for the servo position [12]

4. Software Description

4.1 Arduino IDE

Arduino is an open source platform developed to build electronic projects. Arduino consists of a physical programmable circuit board (called microcontroller) and works in conjunction with the software or IDE (Integrated Development Environment), which runs on the computer, and is used to write, edit or load the code on the board physics or electronic module.

4.2 Arduino Libraries

The Arduino development environment can be extended even further by using the platform's own libraries or external

to it. Libraries provide additional functionality for use in sketches (programs), such as working with non-Arduino hardware or data manipulation.

a) **ESP32 BLE Arduino:** This Library provides a low power Bluetooth implementation (Bluetooth Low Energy) for the ESP32 using the Arduino platform. [13].
b) **ESP32Servo:** This Library can control a many types of servos. It makes use of the ESP32 PWM timers: the library can control up to 16 servos on individual channels. At the moment no attempt has been made to support multiple servos per channel. With the use of this library allows the ESP32 plates to control the servomotors using the semantics of Arduino [14].
c) **ESP32 Web Server:** The webserver library for the ESP32 is not yet listed in an official repository. This Library provides some very useful methods that allow you to configure the server and handle incoming HTTP requests without having to worry about low-level deployment details.

4.3 Android Studio IDE

Android Studio is the official IDE for the development of mobile applications on the Android platform. The Android operating system is completely open source, this means that anyone, even the Android competitors, can choose to download, install, modify and distribute their source code for free. Android developers are free to benefit from the hardware of the device, access location information, run services in the background, and so on. When given the appropriate permissions, Android application can share data among one another and access shared resources on the system securely [15].

4.4 Bluetooth Low Energy (BLE)

In contrast to Classic Bluetooth, Bluetooth Low Energy (BLE) is designed to provide significantly lower power consumption. This allows Android apps to communicate with BLE devices that have stricter power requirements, such as proximity sensors, heart rate monitors, and fitness devices. With Bluetooth Low Energy, there are two types of devices: the server and the client. The ESP32 can act either as a client or as a server. The server advertises its existence, so it can be found by other devices, and contains the data that the client can read. The client scans the nearby devices, and when it finds the server it is looking for, it establishes a connection and listens for incoming data. This is called point-to-point communication. [16]. In Addition to this type of connection, BLE also supports broadcast mode and Mesh network: where all the devices are connected, this is a many to many connection and the Broadcast mode: that unlike the previous, in this mode the server transmits data to many clients that are connected.

5. Wi-Fi and BLE Implementation

In the implementation of Wi-Fi connectivity, the ESP32 system-based module has been configured as an access point (AP mode), this functionality is similar to a traditional wireless router. The ESP32 module broadcasts an SSID (Service Set Identifier) through the air and any nearby device that is successfully authenticate to the wireless network, obtains a unique IP address as shown in Figure 5. In this mode of operation up to 5 users can connect to the wireless interface and send HTTP requests from a device. By default the ESP32 module obtains the first IP address of the 192.168.4.1 range, knowing this information any device connected to the local network can access the Web Server, through a browser.

Figure 5: Device connected to the wireless network of the ESP32 module

To send and receive data from the ESP32, the Web Server listens for incoming HTTP requests through port 80. According to the parameters that are received from the mobile application, automatic will run a function that responds to the request previously established in that path. As shown in Figure 6, such an answer may be a plain text message that will be displayed in the mobile application interface, or failing that, by activating any servomotor connected to one of the digital outputs of the robotic arm through the instruction servoPinza.write (180).

Proyecto_BrazoRobotico

```
/* Control de la Pinza */
server.on("/abrirPinza", []() {
  servoPinza.write(180);
  server.send(200, "text/plain", "Abrir Pinza");
});

server.on("/cerrarPinza", []() {
  servoPinza.write(0);
  server.send(200, "text/plain", "Cerrar Pinza");
});
```

Figure 6: Predefined Instructions for opening and closing the robotic arm clamp

As for the topic of programming in Android as shown in Figure 7, the sending of the instructions is made through the events of the buttons and scrollbars. However, the IP address of the Web Server has previously been assigned to a text field for request-sending by the HTTP protocol. In Addition, an additional variable of the String type has been defined to compare the status of the button when pressed into the application interface.

```
View.OnClickListener abrirCerrarPinzaClickListener = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String estadoPinza;
        if (v == abrirPinza) {
            estadoPinza = "/abrirPinza";
        } else {
            estadoPinza = "/cerrarPinza";
        }

        abrirPinza.setEnabled(false);
        cerrarPinza.setEnabled(false);

        final String serverIP = ingresarIp.getText().toString() + estadoPinza;
        TaskEsp taskEsp = new TaskEsp(serverIP);
        taskEsp.execute();
    }
};
```

Figure 7: Predefined Instructions for opening and closing the robotic arm clamp

On the other hand, in Bluetooth connectivity, the ESP32 module was configured with a specific name to be able to identify the BLE (Bluetooth Low Energy) server. Due to the Bluetooth version, the server will issue its services in Figure 8, which can be filtered by UUID (Unique Universal Identifier). When a client scans nearby devices and establishes a point-to-point connection with the BLE server you are looking for, this is where you can access the features of that specific service and we can read or write values. In this case, the values correspond to the instructions programmed in the events of the buttons of the application in Android. Therefore to realize the Bluetooth connectivity the user must enter the interface of the mobile application and click on the option to connect. In this way the application will request permission to activate the Bluetooth service on the device, and automatically start a scanner showing on screen the list of all devices compatible with BLE. Selecting the correct will change to the main screen in charge of performing the actuators control and a connected/disconnected message as the case may be.

```
// Bluetooth
BLECharacteristic *pCharacteristic;
bool deviceConnected = false;
uint8_t cc = 64;
#define SERVICE_UUID        "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"
```

Figure 8: UUID Service and Characteristic, used in Arduino programming and mobile application

6. Testing and Results

To experience the potential of the Web server created by the ESP32, data from the traffic from the wireless network of the robotic arm were collected, using the free software application Wireshark. As a result of this test it was possible to confirm that the HTTP requests sent from a browser had been recognized by the Web server, obtaining information in detail of the response time, the number of bytes sent and the protocol of Communication of each one of the packages analyzed. Table 1 shows the server's response time in relation to the client's distance.

Table 1: Sending instructions in real time through Wi-Fi

HTTP Instruction	*Response time*	*Distance*
192.168.4.1/bajarPinza	0.049912000 s	1 m
192.168.4.1/subirPinza	0.050899000 s	1 m
192.168.4.1/bajarPinza	0.053902000 s	10 mts
192.168.4.1/subirPinza	0.055742000 s	10 mts
192.168.4.1/bajarPinza	0.054044000 s	20 mts
192.168.4.1/subirPinza	0.051312000 s	20 mts
192.168.4.1/bajarPinza	0.056863000 s	30 mts
192.168.4.1/subirPinza	0.059534000 s	30 mts
192.168.4.1/bajarPinza	0.051645000 s	40 mts
192.168.4.1/subirPinza	0.057363000 s	40 mts

According to the tests carried out, an optimal performance of the ESP32 module is guaranteed in a radius of 40 MTS and even to a few meters more, as long as the user's device remains connected to the ESP32 wireless network. It Should be noted that the instructions cannot be sent simultaneously, if it were the case the connection would be lost approximately 3 seconds and a message would be displayed on the screen of the mobile application.

On the other hand, in order to be able to visualize the intensity of the Bluetooth beacons emitted by the ESP32 module and the stability of the signal, the RSSI values (Received Signal Strength Indicator) were obtained from a third party application that disaggregates in detail the statistics of the beacons BLE. As shown in Figure 9, the Analyser BLE application registers an average of -47.8 dbm a meter away from the ESP32 module in a time span of 60 s. Regardless of the recorded time interval, the signal will always be ideal with transfer rates stable due to the short distance of the mobile phone with the electronic module.

Figure 9: Third-party Application showing the stats of the BLE beacons at 1 meter away.

In comparison with the previous test, in Figure 10 the signal strength can be observed at a distance of approximately 40 meters. The average RSSI that was registered from the BLE Analyzer application was -88.7 dbm, in a maximum range of-95 dbm and a minimum -81 dbm, this indicates that the greater the negative number, there will be greater signal loss and more unstable will be the RSSI. As seen on the left side of the image, even at this distance the application of the robotic arm detected the beacons that transmitted the ESP32 at-91 dbm, in this instance the coverage was still good allowing Bluetooth connectivity and sending instructions from the graphical interface without any disconnection. This test also confirmed the reliability of the application of the robotic arm in the calculation of RSSI compared to other applications.

Figure 10: RSSI Calculation at 40 meters distance by the application of the robotic arm and a third party application

As a final result of each of the tests carried out, it was possible to demonstrate that both technologies guarantee optimum performance in distances less than 40 meters. As such the maximum distance was not factor to process the information in real time and to actuate any actuator of the robotic arm connected to the ESP32 module. It is important to mention that as long as the module remains connected to a power supply, low energy beacons are transmitted and a SSID is diffused that allows to establish connectivity at any time.

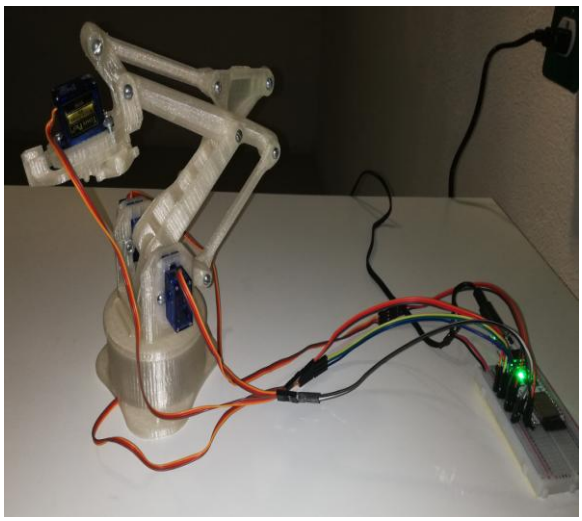

Figure 11: Implementation of the ESP32 module and the robotic arm

It should be noted that for the control of the robotic arm, a single program was created in the Arduino IDE, integrating the libraries and corresponding functions to initialize the Bluetooth wireless connectivity and Wi-Fi simultaneously through the ESP32 module. As for the design of the mobile application is the possibility to choose the wireless technology to use and is only compatible with devices that have installed an Android version superior to the KitKat 4.4.

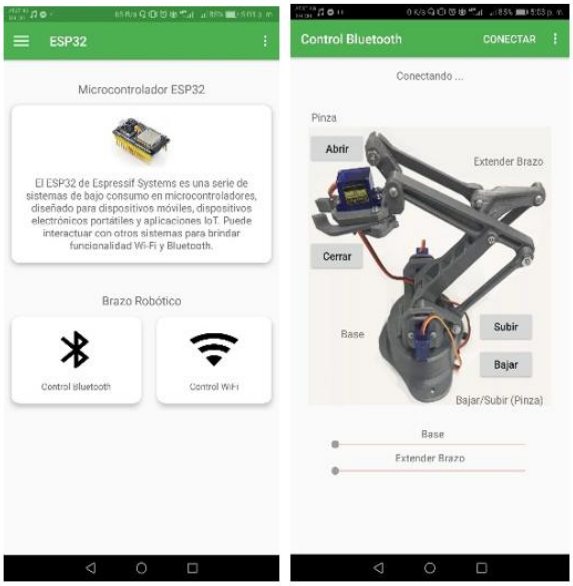

Figure 12: Interface to perform servo motor control of robotic arm

7. Conclusions

As mentioned at the beginning of the document the integration of different areas can be a little complicated, but not impossible. Disciplines such as electronics and programming are considered key skills for the development of IoT, for this same reason hardware platforms have emerged that facilitate the learning of this trend to a wider audience.

This paper presents an accessible hardware solution for the development of electrical systems using the Arduino programming language and a module based on the ESP32. To get the final result that was to implement wireless connectivity and make it send/receive data to a ESP32 module from a mobile device, we started by knowing the Arduino platform, as well as the programming language. It was essential to start with simple practices, such as: Turn off and turn on an led automatically, to develop a native application on Android, with wireless connectivity to the ESP32 module. In general terms it was able to show the good performance that this SoC has as a Web Server and BLE, as well as the wireless scope in open spaces. It is worth mentioning that the environmental factor is a very important variable to consider because it can cause loss of signal for both technologies, so the results can vary depending on the environment.

With regard to the programming of the mobile application, it is necessary to know the communication protocol and the existing methods of the Android platform, which facilitate the interaction with any electronic device. No doubt this application will serve as a support for the implementation of systems or control projects even more complex, requiring wireless connectivity Wi-Fi and Bluetooth. This allows us to conclude that for this kind of projects, the ESP32 in conjunction with the Arduino IDE are a good option as a control system fulfilling very well with the requested using a compact electronic card. As a future work and based on the good results obtained, there is a need to design a similar

application for IOS devices and experience its operation by connecting the ESP32 to a service in the cloud that integrates the complete system of IoT.

References

[1] S. Chtourou, M. Kharrat, N. Ben Amor, M. Jallouli and M. Abid, "Improvement of IOIOAI service for external sensors communication between app inventor and ioio," 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Sousse, 2016, pp. 181-186.
[2] S. Chtourou, M. Kharrat, N. Ben Amor, M. Jallouli and M. Abid, "Easing the development of android apps to create electronic prototypes: IOIO+App Inventor," in IET Circuits, Devices & Systems, vol. 11, no. 4, pp. 310-320, 7 2017.
[3] S. Chtourou, M. Kharat, N. B. Amor, M. Jallouli and M. Abid, "Development of an Android Service to add IOIO Hardware Features to Android Apps," 2016 IEEE International Conference on Computer and Information Technology (CIT), Nadi, 2016, pp. 100-103.
[4] Arduino. About Us, 2018. [Online]. Available: https://www.arduino.cc/en/Main/AboutUs. [Accessed: Jun. 12, 2018.
[5] Espressif, Dual-core Modules with Wi-Fi & Dual-mode Bluetooth, 2018. [Online]. Available: https://www.espressif.com/en/products/hardware/modules. [Accessed: Jul. 24, 2018].
[6] K. Afifah, S. Fuada, R. V. W. Putra, T. Adiono and M. Y. Fathany, "Design of low power mobile application for Smart Home," 2016 International Symposium on Electronics and Smart Devices (ISESD), Bandung, 2016, pp. 127-131.
[7] Android, Android is for everyone, 2018. [Online]. Available: https://www.android.com/everyone/enabling-opportunity/. [Accessed: Jun. 18, 2018].
[8] Espressif, ESP32 Series Datasheet, 2019. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Accessed: Jan. 24, 2019].
[9] Getting Started with the ESP32 Development Board, 2018. [Online]. Available: https://randomnerdtutorials.com/getting-started-with-esp32/. [Accessed: Jun. 24, 2018].
[10] Espressif, ESP32 A Different IoT Power and Performance, 2018. [Online]. Available: https://www.espressif.com/en/products/hardware/esp32/overview. [Accessed: Sept. 12, 2018].
[11] Prometec, Conociendo los Servos, 2018. [Online]. Available: https://www.prometec.net/servos/#. [Accessed: Jul. 02, 2018].
[12] MIYmakers, Controlando el servomotor con un potenciómetro, 2018. [Online]. Available: https://miymakers.wordpress.com/2017/05/12/controlling-servo-motor-with-a-potentiometer/. [Accessed: Jul. 12, 2018].
[13] Arduino Library List, ESP32 BLE Arduino 2018. [Online]. Available: https://www.arduinolibraries.info/libraries/esp32-ble-arduino. [Accessed: Sept. 12, 2018].
[14] Arduino Library List, ESP32Servo 2018. [Online]. Available: https://www.arduinolibraries.info/libraries/esp32-servo. [Accessed: Sept. 12, 2018].
[15] R. V. Golhar, P. A. Vyawahare, P. H. Borghare and A. Manusmare, "Design and implementation of android base mobile app for an institute," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, 2016, pp. 3660-3663.
[16] Getting Started with ESP32 Bluetooth Low Energy (BLE) on Arduino IDE, 2018. [Online]. Available: https://randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/ [Accessed: Sept. 25, 2018].

Author Profile

José Eduardo Tzompantzi Netzahual has a degree in Engineering in the Information and Communications Technologies from the Instituto Tecnológico de Apizaco, 2017. He is currently studying the Masters in Computer Systems in Software Engineering from the Instituto Tecnológico de Apizaco.

Higinio Nava Bautista has a degree in Computer Engineer, Universidad Autónoma de Tlaxcala, 2007. Master in Software Development, Instituto Universitario en Tecnologías y Humanidades, 2012. Professor in the Engineering Career in Information Technology and Communications. Professor of the Master of Computer Systems of the Instituto Tecnológico de Apizaco.

María Janai Sánchez Hernández has a degree in Informatics bachelor degree, Instituto Tecnológico de Apizaco, 20a01. Master degree in Computation Sciences, Instituto Tecnológico de Apizaco, 2006. Professor in the Systems and Computation department, 2006. Coordinator of the Computational Systems Master of the Instituto Tecnológico de Apizaco, 2015.