

**ISTANBUL TECHNICAL UNIVERSITY**  
**ELECTRICAL – ELECTRONICS FACULTY**

**HARDWARE/SOFTWARE CODESIGN OF A RFID BASED INDOOR  
LOCALIZATION SYSTEM**

**BSc Thesis by**

**Onur Azbar**

**040100518**

**Department: Electronics and Communication Engineering**

**Programme: Electronics and Communication Engineering**

**Supervisor: Assoc. Prof. Dr. Sıddıka Berna Örs Yalçın**

**MAY 2015**

## **ACKNOWLEDGEMENT**

First of all, I would like to express my deepest thanks to my supervisor, Assoc. Prof. Dr. Sıddıka Berna Örs Yalçın for approving me to work on this project, sparing her precious time and sharing her knowledge to complete my thesis.

Also, I would like to thank my family for all their love, support and constant encouragement I have gotten over the years.

Onur Azbar

MAY 2015

## INDEX

<b>ABBREVIATIONS .....</b>	<b>v</b>
<b>FIGURE LIST .....</b>	<b>vi</b>
<b>SUMMARY .....</b>	<b>viii</b>
<b>ÖZET.....</b>	<b>ix</b>
<b>1. INTRODUCTION .....</b>	<b>1</b>
<b>2. BACKGROUND INFORMATION.....</b>	<b>4</b>
2.1 Radio Frequency Identification .....	4
2.2 Received Signal Strength Indicator .....	5
2.3 Realization Platforms.....	7
2.3.1 RFM23B Module .....	7
2.3.2 Arduino Uno R3.....	9
2.3.3 Hardware Description Languages .....	11
2.3.3.1 Verilog.....	11
2.3.3.2 VHDL.....	11
2.3.4 Field Programmable Gate Array .....	11
2.3.5 Xilinx Spartan-6 XC6SLX16.....	13
2.3.6 Microblaze Processor .....	14
2.3.7 Xilinx Integrated Synthesis Environment .....	15
2.3.8 Xilinx Embedded Development Kit.....	16
2.3.8.1 Xilinx Platform Studio .....	17
2.3.8.2 Xilinx Software Development Kit.....	18
<b>3. LOCALIZATION ALGORITHM.....</b>	<b>21</b>
3.1 2D Trilateration Algorithm.....	21
3.2 3D Trilateration Algorithm.....	25
<b>4. RFID AUTHENTICATION PROTOCOL .....</b>	<b>29</b>
4.1 Scalable Encryption Algorithm .....	31
4.1.1 Basic Operations .....	33
4.1.2 The Round and Key Round.....	34

<b>5. IMPLEMENTATION OF LOCALIZATION ALGORITHM.....</b>	<b>35</b>
5.1 Realization on General Purpose Processor .....	36
5.2 Realization on Microblaze .....	39
5.2.1 Software Design .....	39
5.2.2 Hardware/Software Codesign .....	41
5.3 Visualization of localization Results .....	46
<b>6. IMPLEMENTATION OF AUTHENTICATION PROTOCOL .....</b>	<b>50</b>
6.1 Realization of Scalable Encryption Algorithm.....	50
<b>7. CONCLUSION.....</b>	<b>55</b>
<b>REFERENCES.....</b>	<b>57</b>
<b>RESUME.....</b>	<b>59</b>

## **ABBREVIATIONS**

<b>GPS</b>	: Global Positioning System
<b>RFID</b>	: Radio Frequency Identification
<b>RSSI</b>	: Received Signal Strength Indicator
<b>FPGA</b>	: Field Programmable Gate Array
<b>ID</b>	: Identity
<b>SEA</b>	: Scalable Encryption Algorithm
<b>BaaS</b>	: Building as a Service
<b>SPI</b>	: Serial Peripheral Interface
<b>LUT</b>	: Look-up Table
<b>RAM</b>	: Random Access Memory
<b>HDL</b>	: Hardware Description Languages
<b>JTAG</b>	: Joint Action Test Group
<b>USB</b>	: Universal Serial Bus
<b>UART</b>	: Universal Asynchronous Receiver/Transmitter
<b>VGA</b>	: Video Graphics Array
<b>RISC</b>	: Reduced Instruction Set Computer
<b>ISE</b>	: Integrated Synthesis Environment
<b>EDK</b>	: Embedded Development Kit
<b>XPS</b>	: Xilinx Platform Studio
<b>SDK</b>	: Software Development Kit
<b>EXOR</b>	: Exclusive Or

## FIGURE LIST

<b>Figure 1.1</b>	: Basic sections in the project .....	2
<b>Figure 1.2</b>	: Localization System Structure.....	3
<b>Figure 2.1</b>	: PRX versus distance to the tansmitter .....	6
<b>Figure 2.2</b>	: RSSI as quality identifier of the $P_{RX}$ .....	6
<b>Figure 2.3</b>	: RFM23B .....	7
<b>Figure 2.4</b>	: Soldered RFM23B on board.....	8
<b>Figure 2.5</b>	: RFM23B-Microcontroller structure .....	8
<b>Figure 2.6</b>	: Maximum power delivered at quarter wavelength .....	9
<b>Figure 2.7</b>	: Arduino Uno R3 board .....	9
<b>Figure 2.8</b>	: SPI pin connections between Arduino and RFM23B.....	10
<b>Figure 2.9</b>	: Logic Cell structure .....	12
<b>Figure 2.10</b>	: Internal structure of FPGA .....	12
<b>Figure 2.11</b>	: Spartan-6 on Nexys 3 board .....	13
<b>Figure 2.12</b>	: Architecture of Microblaze.....	14
<b>Figure 2.13</b>	: A view of Xilinx ISE tool.....	16
<b>Figure 2.14</b>	: A hardware/software design with Microblaze.....	17
<b>Figure 2.15</b>	: A view of Xilinx XPS tool .....	18
<b>Figure 2.16</b>	: A view of Xilinx SDK tool.....	19
<b>Figure 2.17</b>	: Implementation of hardware/software design .....	20
<b>Figure 3.1</b>	: Infinite solutions with one tag .....	22
<b>Figure 3.2</b>	: Two possible solutions with two tag .....	22
<b>Figure 3.3</b>	: The unique solution with three tag .....	23
<b>Figure 3.4</b>	: Geometry of a circle .....	24
<b>Figure 3.5</b>	: Infinite solutions with one tag .....	25
<b>Figure 3.6</b>	: Infinite solutions with two tags .....	26
<b>Figure 3.7</b>	: Two possible solutions with three tags.....	26
<b>Figure 3.8</b>	: Geometry of a sphere.....	27
<b>Figure 4.1</b>	: Principle of authentication protocol .....	29
<b>Figure 4.2</b>	: Authentication protocol operation.....	30
<b>Figure 4.3</b>	: Encryption/Decryption round .....	32

<b>Figure 4.4</b>	: Key Scheduling round .....	33
<b>Figure 5.1</b>	: Basic operations of localization algorithm .....	35
<b>Figure 5.2</b>	: Coefficient array initialization.....	36
<b>Figure 5.3</b>	: Localization results on general purpose processor .....	38
<b>Figure 5.4</b>	: Localization results on both Microblaze and computer .....	40
<b>Figure 5.5</b>	: Floating number print code for Microblaze.....	40
<b>Figure 5.6</b>	: Microblaze structure with user defined hardware module .....	42
<b>Figure 5.7</b>	: coef_calculator module .....	42
<b>Figure 5.8</b>	: adder module .....	43
<b>Figure 5.9</b>	: subtractor module .....	43
<b>Figure 5.10</b>	: multiplier module .....	44
<b>Figure 5.11</b>	: Internal structure of coef_calculator module.....	45
<b>Figure 5.12</b>	: Visualized localization results .....	48
<b>Figure 6.1</b>	: Key Schedule.....	52
<b>Figure 6.2</b>	: Implementation of authentication protocol .....	53
<b>Figure 6.3</b>	: The role of entity authentication.....	54

# **HARDWARE/SOFTWARE CODESIGN OF A RFID BASED INDOOR LOCALIZATION SYSTEM**

## **SUMMARY**

Although the Global Positioning System (GPS) is very accurate for outdoor positioning operations, it is generally unsteady for indoor operations. To handle this situation, variety of systems have been developed and Radio Frequency Identification (RFID) technology is generally preferred in these implementations. In this project, it is aimed to obtain localization information of people inside a building by using RFID tags. For this purpose, Received Signal Strength Indicator (RSSI) values are used to have a distance estimate. To implement localization algorithms, some mathematical operations need to be performed and Field Programmable Gate Arrays (FPGA) have been used to realize these operations.

The term reader indicates the person who is wanted to be located and the term tag indicates the RFID tags which are placed to the walls in the building. Same RFID transceiver devices have been used as both readers and tags. Transceiver means that the device can operate both in transmitter mode and receiver mode. RFM23B module has been selected as RFID transceiver and this device has been used in control of a microcontroller. A microcontroller is necessary to drive and modify the operation modes of the transceiver and Arduino modules are used for this purpose.

In localization process, the tags transmit their identity (ID) value to the reader. When the reader obtains the radio signal, RSSI value is calculated by related analog equipments of the RFID module and this value converted to a 8-bit binary format to be processed by digital components. This RSSI value is used for a distance estimation and stored with related tag ID. Using the distance values from different tags the localization algorithm calculates the position of the reader. Finally, the results are sent to main frame and printed out to a screen.



## **RFID TABANLI BİR BİNA İÇİ KONUM BELİRLEME SİSTEMİNİN YAZILIM/DONANIM TASARIMI**

### **ÖZET**

Küresel Konumlama Sistemi (Global Positioning System, GPS) bina dışı konum belirleme uygulamalarında oldukça başarılı olmasına rağmen, bina içlerinde veya kapalı alanlarda aynı başarıyı gösterememektedir. Bu sorunu çözmek için çok sayıda sistem geliştirilmiştir ve bu uygulamalarda genellikle Radyo Frekansı ile Tanımlama (Radio Frequency Identification, RFID) teknolojisi tercih edilmektedir. Bu projede, RFID etiketleri kullanılarak bina içindeki insanların konum bilgileri öğrenilmek istenmektedir. Bu amaçla Gelen Sinyalin Güç Değeri (Received Signal Strength Indicator, RSSI) mesafe tahmininde bulunmak için kullanılmaktadır. Konum belirleme algoritmasının gerçekleştirilmesi için bazı matematiksel işlemlerin gerçekleştirilmesi gerekmektedir ve bu amaçla Alanda Programlanabilir Kapı Dizileri (Field Programmable Gate Array, FPGA) kullanılmıştır.

Okuyucu (reader) terimi konumu belirlenecek kişiyi ve etiket (tag) terimi bina duvarlarına yerleştirilen RFID etiketleri ifade etmektedir. Aynı RFID alıcı-verici (transceiver) cihazları hem okuyucular hem de etiketler için kullanılmıştır. Alıcı-verici aynı cihazın hem alıcı modunda hem de verici modunda çalışabildiğini ifade etmektedir. RFM23B modülü RFID alıcı-verici olarak seçilmiştir ve bu cihaz bir mikrokontrolörün kontrolörün güdümünde çalışmaktadır. Mikrokontrolör cihazı sürmek ve çalışma modlarına karar vermek için gereklidir ve bu amaçla Arduino modülleri kullanılmıştır.

Konum belirleme sürecinde, etiketler kimlik bilgilerini (identity, ID) okuyucuya gönderirler. Okuyucu radyo sinyalini aldığı zaman, RSSI değeri ilgili analog parçalar tarafından hesaplanır ve sayısal olarak işlenebilmesi için bu değer ikilik tabanda 8 bitlik bir sayıya dönüştürülür. Bu RSSI değeri mesafe tahmini yapmakta kullanılır ve kimlik bilgisiyle (ID) değeriyle birlikte saklanır. Değişik etiketlerden elde edilen mesafe bilgileriyle konum belirleme algoritması okuyucunun yerini saptar. Son olarak hesaplanan konum bilgisi merkezi bir bilgisayara gönderilir ve ekrana basılır.

## 1. INTRODUCTION

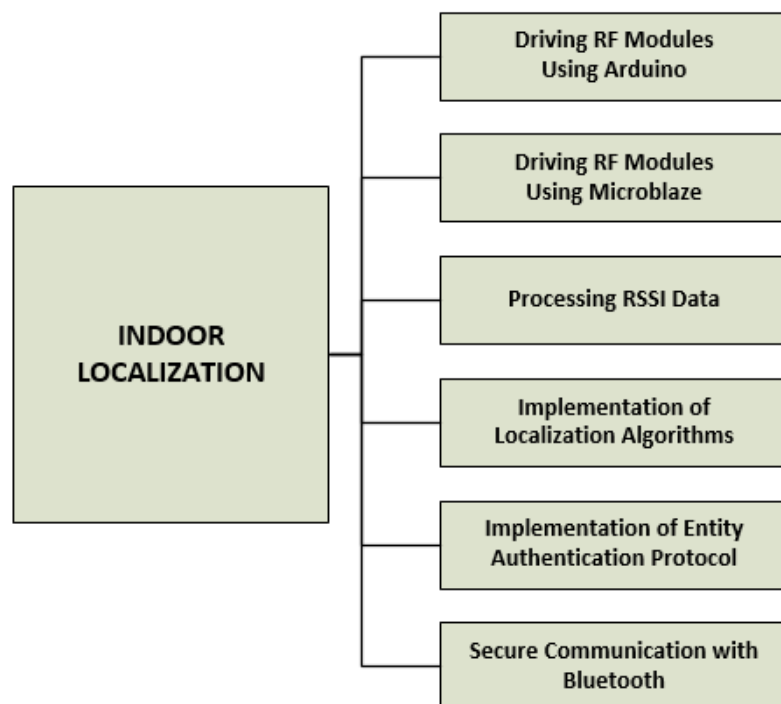
Localization of objects and people is one of the key aspects for security, safety and service applications in today's technology. Although this field of study emerged a while ago, it is still actively researched [1]. For outdoor localization applications, GPS technology is quite sufficient to locate people or objects. However it is not able to work out for indoor localization applications. The main reason is that GPS technology requires direct "line of sight" which means that between the satellite and the target there should not be any obstacles such as trees, walls, buildings etc [2]. Also for localization with GPS, it is required that least 4 different satellites should participate in the operation and this condition increases the cost. Therefore, lots of new methods have been proposed for indoor localization and they all require a wireless technology for implementation. Radio Frequency Identification (RFID) and Bluetooth are the most widely used wireless technologies through low cost and accurate operation performances. Considering all the factors mentioned above, RFID wireless technology has been chosen to realize this indoor localization system. Basically, a RFID system has two types of components which are tags and reader. The reader operates in receiver mode and the tags operate in transmitter mode. Through its transceiver operation property, RFM23B modules have been selected for both the reader and the tag. Transceiver means the device can both operate as a transmitter or a receiver by some software modifications. The tags used in this project can be categorized as active RFID tags which means the devices take their energy directly from power supplies.

Trilateration has been selected as localization algorithm and it has been preferred to realize two dimensional (2D) localization in this study. With similar concepts this operation can be modified to three dimensional (3D) localization. Geometrical properties of circles are used for 2D localization and geometrical properties of spheres are used for 3D localization. Both these methods require a distance information and it is obtained through Received Signal Strength Indicator (RSSI) value. First, the algorithm has been implemented on C++ via Visual Studio and then it is implemented on Field Programmable Gate Array (FPGA) using Microblaze. Microblaze is virtual

microprocessor that can realized on FPGA device and the programs written with C or C++ can be run on the device. Finally, some parts of the algorithm has been implemented as hardware using Verilog HDL language. The designed hardware module has been then operated under the control of Microblaze processor.

Security is also another important issue which needs to be consider. For example, if the localization information of people is revealed by an unauthorized person or group, this situation can cause some privacy violations. Therefore both the reader and the tag need to be sure of communicating with the correct destination. For this purpose, a cryptology protocol which provides entity authentication has been proposed. For encryption and decryption operations Scalable Encryption Algorithm (SEA) is used and the system is realized with C++.

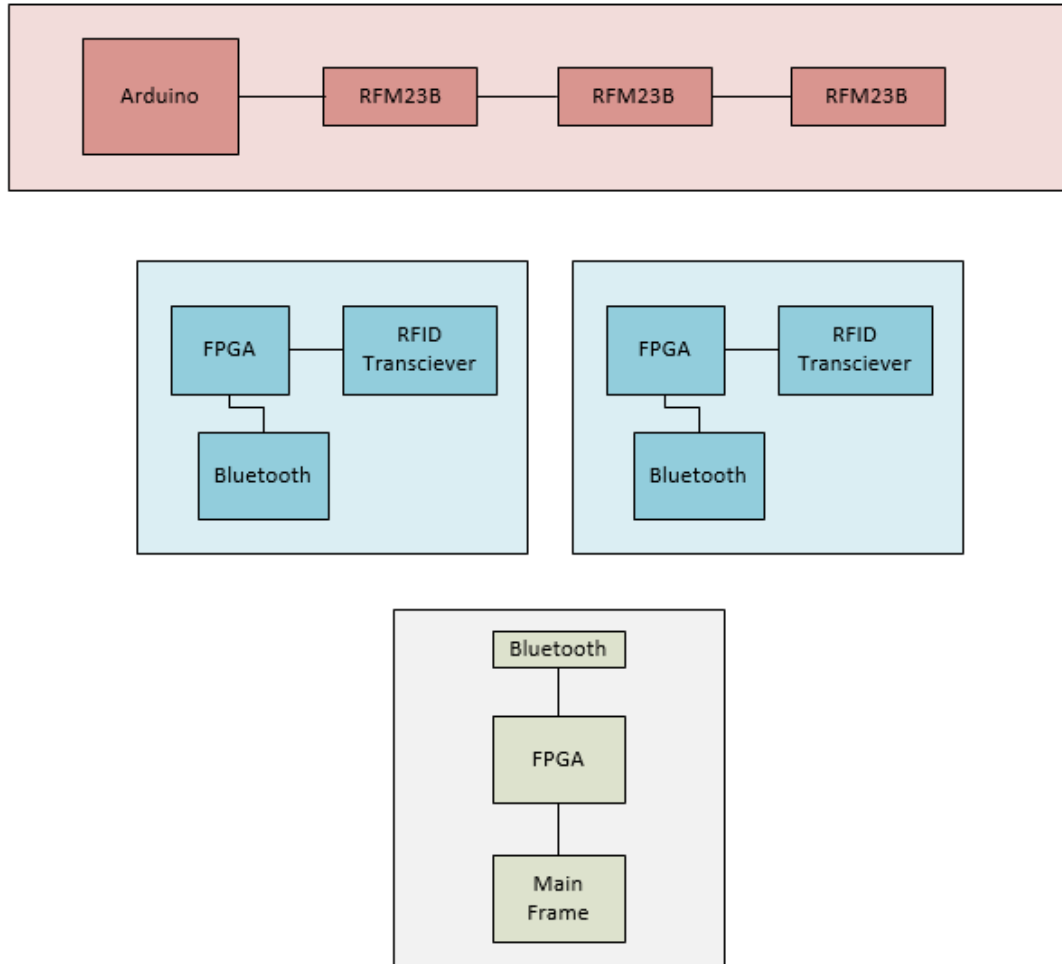
Five students have taken part in this project and everybody has focused on different issues. Main sections of this project are described in the figure 1.1.



**Figure 1.1:** Basic sections in the project.

In the project, driving RF modules using arduino and processing RSSI data have been mainly performed by Ramazan Aktaş [3]. Driving RF modules using Microblaze has been performed by Oğuzhan Çık [4]. Implementation of a localization algorithm and

an entity authentication protocol have been performed by me and Alp Oran [5]. However for both localization and authentication, different algorithms have been realized. Finally, secure communication with Bluetooth has been performed by Bahadır Gün [6]. The whole system structure has been planned to be as in figure 1.2.



**Figure 1.2:** Localization System Structure.

The European Union carries on a project that is named as Building as a Service (BaaS). Main objective of this project is creating technologically equipped buildings. According to BaaS, buildings will be controlled, monitored and managed by distinctive systems. The project has some parts like fire detection alarm, lighting control, and secure access control. Another important property is the evacuation of buildings fast and safely in case of an emergency. Detection of location, determining the number of people in building, and secure communication are basis of this part. Indoor localization is crucial for evacuation of buildings because access to people in emergency is easy if it is known where the people is [7]. This graduation project is planned to be designed in compatible with BaaS project.

## **2. BACKGROUND INFORMATION**

### **2.1 Radio Frequency Identification**

RFID is a rapidly growing technology that uses wireless antennas to identify objects from a distance without requiring line of sight or physical contact. The success rates in capturing real time information accurately from both moving and stationary objects has contributed to be today's widely-used wireless technology [8].

Development of RFID technology primarily depends on the advancements in radio frequency (RF) technology. History of RFID technology over decades is indicated below [9].

- Radar is defined and started to be used. During World War II huge budget and time was devoted to develop this system. RFID was invented in about 1948. (1940-1950)
- Early explorations of RFID technology were made and laboratory experiments were performed. (1950-1960)
- RFID theory was developed and field performance tests were started. (1960-1970)
- RFID development and tests were accelerated considerably and first RFID adopter implementation was performed. (1970-1980)
- Commercial RFID applications entered the mainstream. (1980-1990)
- RFID systems were widely deployed and necessity for standardization emerged. (1990-2000)
- Innovative applications were conducted and RFID systems started to be substantial part of daily life. (2000-2010)

A typical RFID system is mainly composed of two parts that tag and receiver. The tags radiates radio waves with their antennas and the reader receives these radio waves. Depending on the application, different types of tags can be used such as active, semi-active and passive. Active tags are connected a power supplies and can operate continually. On the other hand, passive and semi-active tags are normally in sleep

mode and started to operate with the coming wave energy from reader. In this project, active RFID tags have been used and same RFID transceiver module has been used for both the reader and the tag. Also communication between the tags and the reader has been performed at 433 MHz. The main reasons of active tag usage are constant operation property and higher range. Ranges of these devices depend on both operation frequency and transmitter power. More clearly, passive tags are generally in the range of a few meters and active tags are able to reach 150 m depending on the frequency.

## 2.2 Received Signal Strength Indicator

RSSI is the strength of power which is measured at the receiver antenna. RSSI value depends on transmitter power and the distance between transmitter and receiver antennas. There is an exponential dependency between RSSI and distance instead of linear one. Thus, calculation of the distance value from related RSSI value is a rather complex process. The formula that explains this relationship is shown below [10].

$$P_{RX} = P_{TX} * G_{TX} * G_{RX} (\lambda/4\pi d)^2 \quad (2.1)$$

Where,

$P_{TX}$  = Transmission power of sender

$P_{RX}$  = Remaining power of wave at receiver

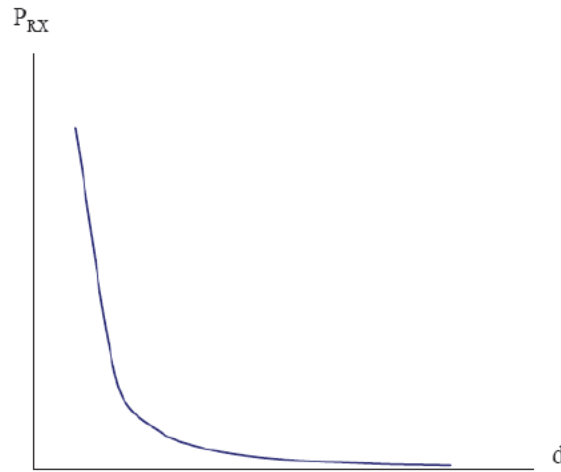
$G_{TX}$  = Gain of transmitter

$G_{RX}$  = Gain of receiver

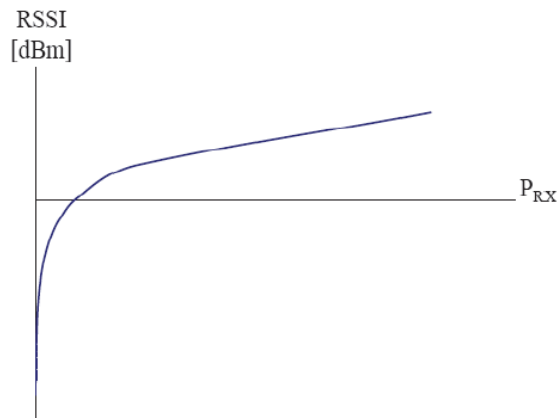
$\lambda$  = Wavelength

$d$  = Distance between sender and receiver

The graphical representation this dependency is also shown in figure 2.1 and figure 2.2.



**Figure 2.1:**  $P_{RX}$  versus distance to the transmitter [10].



**Figure 2.2:** RSSI as quality identifier of the  $P_{RX}$  [10].

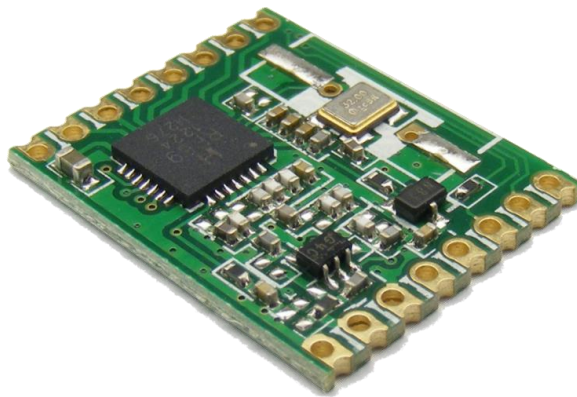
However these results can only be obtained under ideal conditions which means no losses in antennas and transmission path. Considering the all the losses, the results start to deviate from ideal ones. First, transmitting radio waves could be affected by electromagnetic radiations in air and this situation results in a different RSSI value. These electromagnetic radiations generally occur with the increasing number of electronic devices on transmission path. Moreover, feeder losses in both transmitter and receiver antennas also add additional losses. Finally, metallic objects also cause some deviations in signal power and result in different RSSI values.

In this project, corresponding distance and RSSI sample values have been saved and made a table. When a RSSI value is obtained related distance value is obtained from this table. The reason of usage of tables is making the system simpler and considering the losses mentioned above.

## 2.3 Realization Platforms

### 2.3.1 RFM23B Module

To realize the communication between the tags and the reader, RFM23B module generated by HOPERF Electronics company has been used. Through its transceiver property, same module has been used for both RFID tag and RFID reader. A view of used RFM23B module is shown figure 2.3.



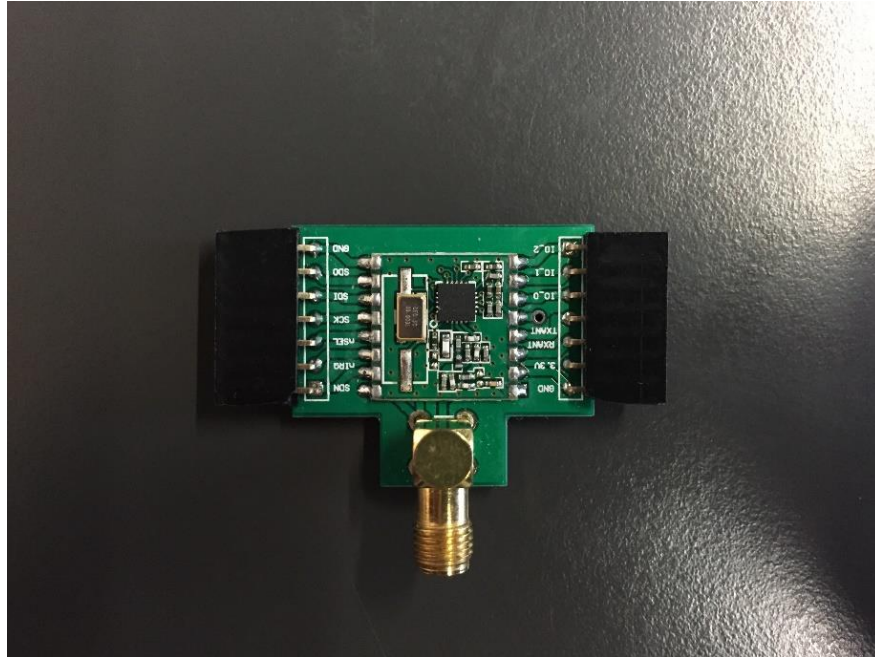
**Figure 2.3:** RFM23B [11].

Important features of the module is also shown below.

- Frequency Range = 240-930 MHz
- +13 dBm Maximum Output Power
- Power Supply = 1.8 to 3.6 V
- Low Power Consumption
  - 18.5 mA receive
  - 28 mA @ +13 dBm transmit
- Data Rate = 1 to 128 kbps
- Digital RSSI

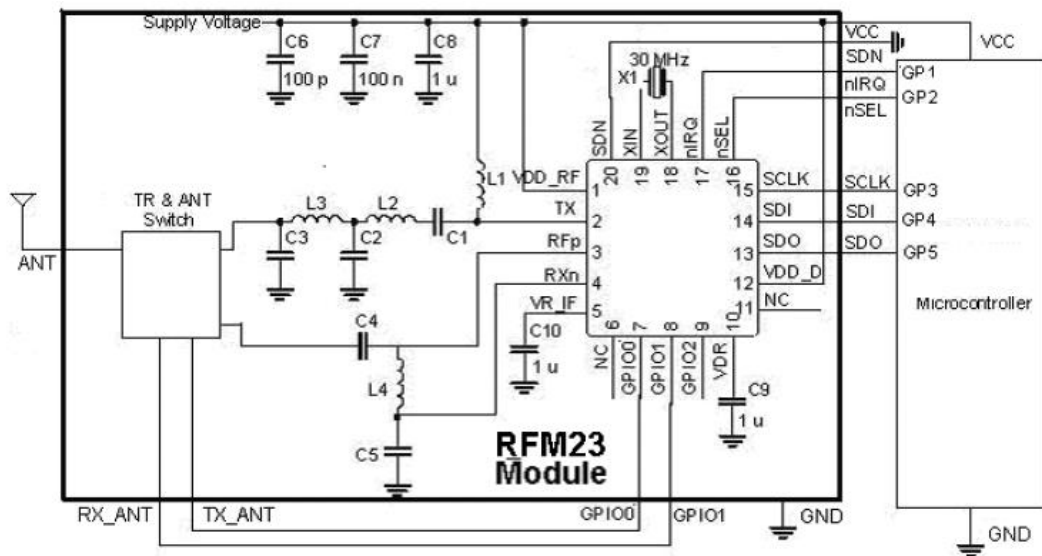
For accurate connection with a microcontroller and an antenna, this module is printed on a board and necessary pin connections are attached as shown in figure 2.4.





**Figure 2.4:** Soldered RFM23B on board.

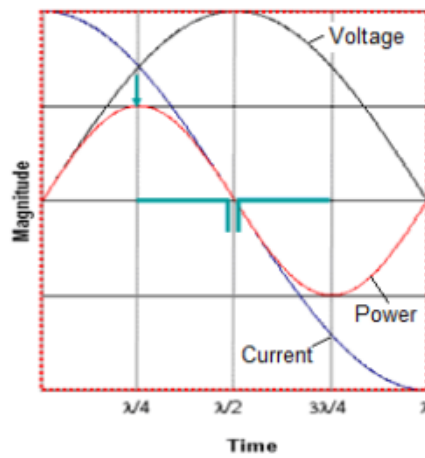
This module is designed to be work with a microcontroller and Arduino module has been used in this project. An example RFM23B-Microcontroller structure with pin connections is described in figure 2.5.



**Figure 2.5:** RFM23B-Microcontroller structure [11].

Frequency has been selected as 433 MHz and transmitter power has been set to maximum in our project. Also isotropic wire antennas have been used for transmitting and receiving operations. To deliver maximum power, the antenna length should be

quarter wavelength and it is 17.3 cm in this study [12]. Related graph which explain this condition is shown in figure 2.6.



**Figure 2.6:** Maximum power delivered at quarter wavelength [12].

### 2.3.2 Arduino Uno R3

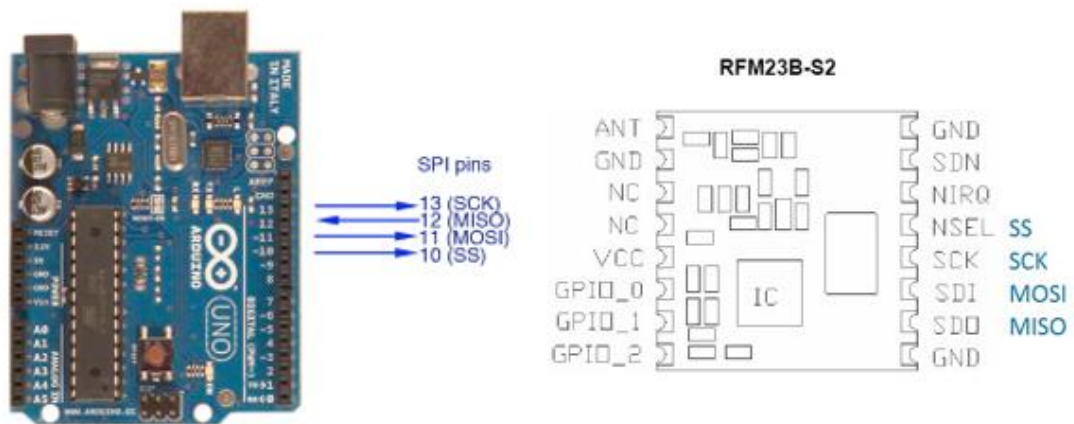
To drive and control operation modes of RFM23B module a microcontroller is necessary. Arduino Uno R3 microcontroller board has been used for this purpose and its structure is shown in figure 2.7.



**Figure 2.7:** Arduino Uno R3 board [13].

ATmega328 microcontroller is located on the board and Arduino Uno R3 makes easier the communication with peripheral units. Main functions of microcontroller arranging transmitted radio wave frequency and power, operation mode, data format that is transferred from one transceiver to another one. Arduino drives RFM23B via Serial

Peripheral Interface (SPI) protocol that related SPI pins of Arduino and RFM23B module are shown in figure 2.8.



**Figure 2.8:** SPI pin connections between Arduino [13] and RFM23B [11].

Arduino has operated as master and RFM23B has operated as slave in this study. Also pin descriptions of modules are shown below [11].

ANT: Antenna is connected to this pin

GND: Represents ground pin

NC: Not connected which means that these are empty pins

VCC: Power supply is connected to this pin

GPIO\_0/1/2: Represents the general input-output pins

SDN: Shutdown pin and grounded in this project

NIRQ: General microcontroller interrupt status control pin and grounded in this project

NSEL: Serial interface select input and connected to slave select pin of Arduino

SCK: Serial clock input and connected to serial clock output of Arduino

SDI: Serial data input and connected to master (Arduino) output

SDO: Serial data output and connected to master (Arduino) input

### **2.3.3 Hardware Description Languages**

#### **2.3.3.1 Verilog**

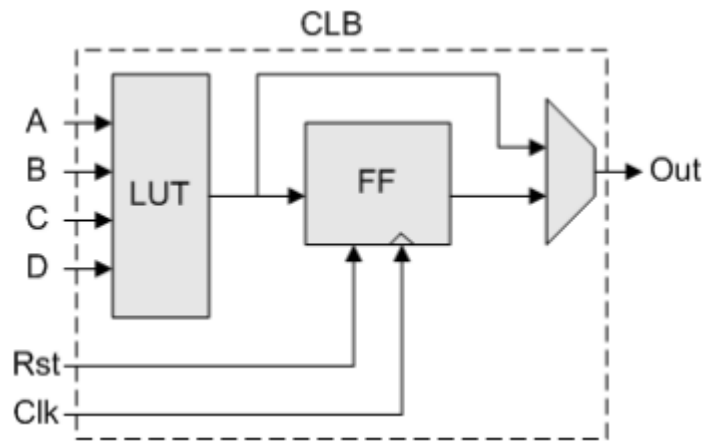
Verilog is a hardware description language (HDL) that is mostly used for design and verification of digital hardware systems. It is created by Prabhu Goel and Phil Moorby during the period of 1984-1985. Both combinational and sequential circuits can be realized with Verilog with related code blocks. Also Verilog shows similar characteristics with C language in some ways and all these make Verilog very attractive for digital hardware designers. In this project, Verilog has been used to realize hardware components of localization algorithm.

#### **2.3.3.2 VHDL**

VHDL is another hardware description language that is used in digital system designs. It can also be used as a general purpose parallel programming language. VHDL is an abbreviation of first letters of VHSIC, Hardware, Description, Language. VHSIC means very high speed integrated circuit. VHDL was developed to meet requirements of U.S Department of Defense about behavior of ASICs documentation. With progressive improvements in the language, it was modified to have a similar characteristics to C language and took place in IEEE standards. Today it is one of the most preferred hardware description language with Verilog by embedded system designers.

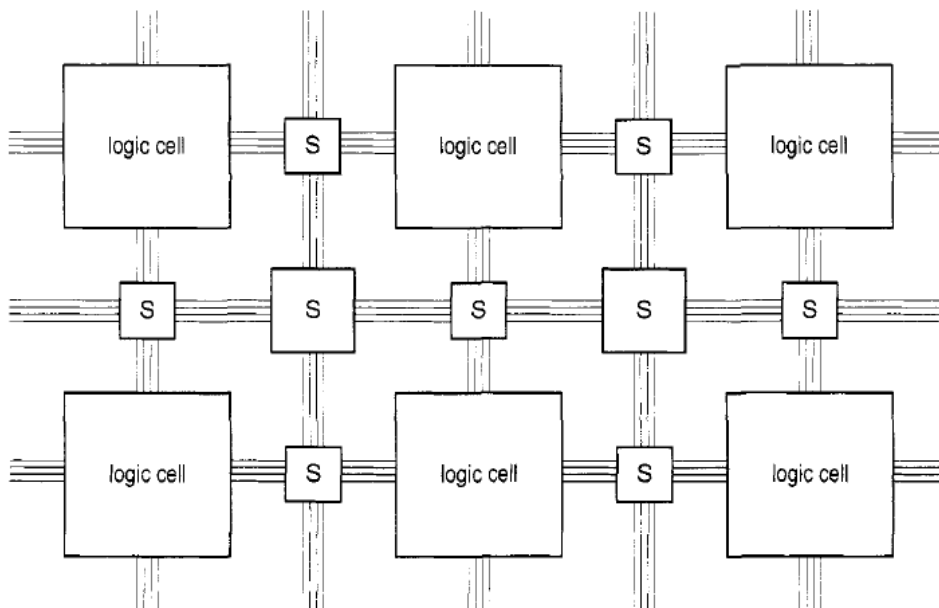
#### **2.3.4 Field Programmable Gate Array**

FPGA is an integrated logic device that can realize digital circuits via hardware description languages such as Verilog and VHDL. FPGA is a two dimensional array that is composed of logic cells and programmable switches. Through programmability property, logic cells can be modified to realize simple logical functions and switches are used to provide necessary interconnections to these logic functions. With this method huge digital circuit structures are able to be implemented on FPGA device. After the circuit is designed with hardware description languages, required logic cell and switch structure information are transferred onto device via a cable [14].



**Figure 2.9:** Logic Cell structure [14].

As seen in figure 2.9, Look-up Table (LUT) based logic cell is a small configurable device that contains a LUT and a D-type flip flop. LUT is used to realize combinational circuit parts and flip flop is generally used when a state machine is necessary in the system. An n-input LUT can be considered as a  $2^n$  memory unit and with the proper code modifications required n-input combinational functions can be realized [14].



**Figure 2.10:** Internal structure of FPGA [14].

Furthermore, FPGA gives designers the opportunity to reconfigure the digital circuit that is implemented on the device. This is rather substantial advantage, because it decreases the digital design costs considerably. For example, when there is problem in the operation of a system, it is able to be fixed by simply software modifications. Also

same device can be used for different operations simultaneously. Microprocessors are another important element of today's electronic systems and a microprocessor systems can be implemented on FPGA device as well.

### 2.3.5 Xilinx Spartan-6 XC6SLX16

Nexys 3 board that manufactured by Digilent company has been used in this project. A Spartan-6 XC6SLX16 type FPGA is located on the center of Nexys 3 board. FPGA on the board is also an integrated digital device and has connections with peripheral devices. Spartan-6 XC6SLX16 is contain of 14579 logic cells, 2278 slices, 18224 flip-flops, 136 maximum distributed random access memory (RAM) (Kb) and 32 block RAM (18 Kb). In addition, logic cells have 6-input LUT structure and FPGA slices are composed of four LUTs and eight flip-flops [15]. The board used in the project is shown in figure 2.11.

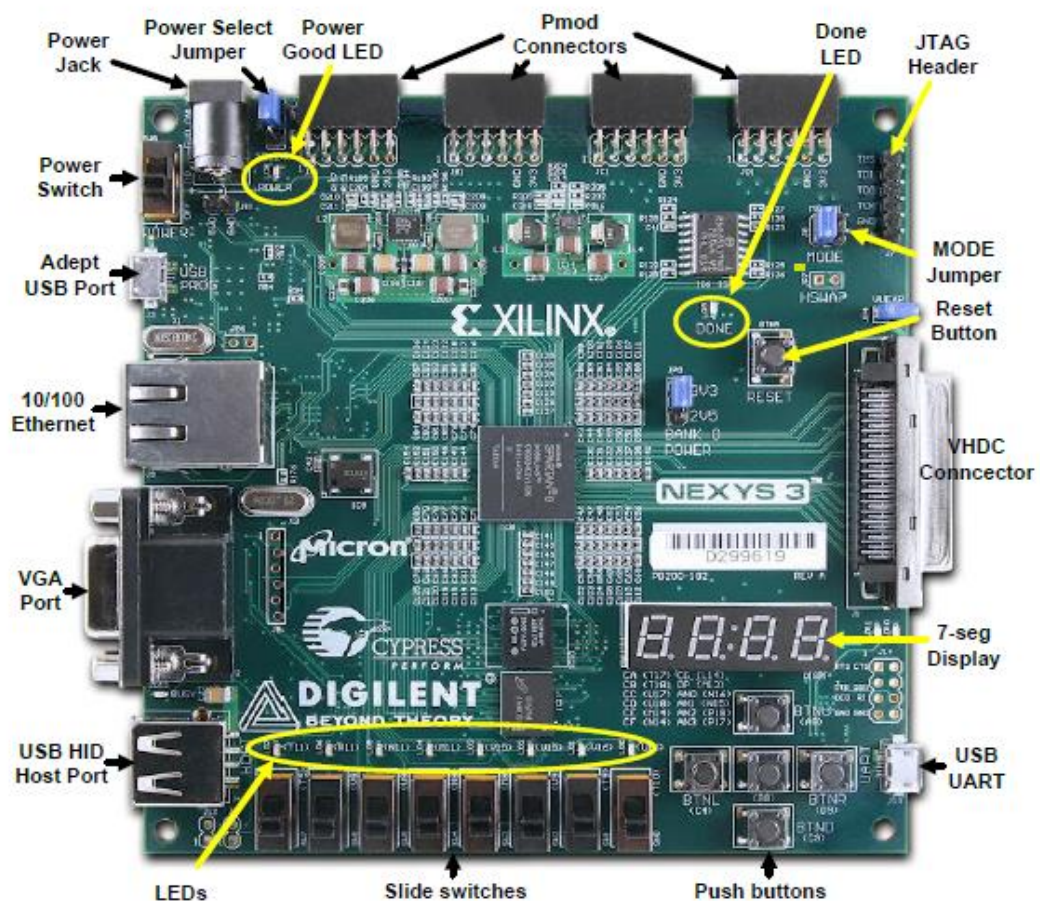


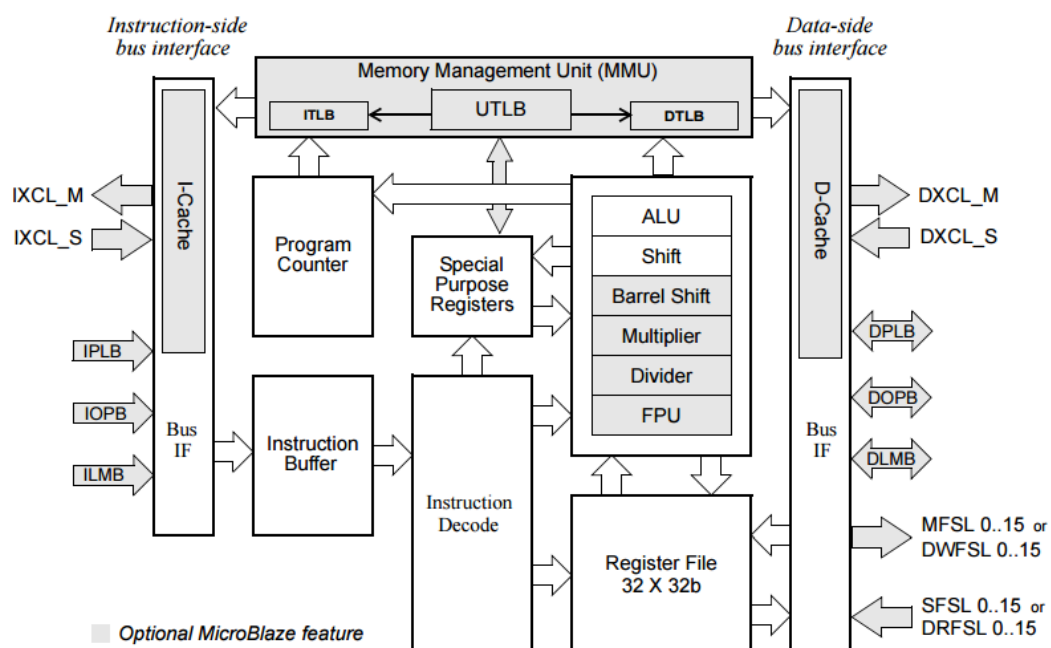
Figure 2.11: Spartan-6 on Nexys 3 board [16].

Spartan-6 operates at 100 MHz clock frequency and can be programmed by both Joint Test Action Group (JTAG) cable and Universal Serial Bus (USB) cable. Also Nexys 3 board involves Universal Asynchronous Receiver/Transmitter (UART) port which is used many time to communicate FPGA device and computer. Switches, buttons, seven segment display and Video Graphics Array (VGA) port are also important elements but not used in this study.

### 2.3.6 Microblaze Processor

Microblaze is a virtual microprocessor and it is generally used to create embedded systems with both software and hardware design. This microprocessor is realized with a certain number of logic cells and related switch assignments. Programming languages such as C and C++ could be used to implement required algorithms. Another important advantage of Microblaze is that some peripheral devices, memory units are able to be used easily through the availability of related protocols. For example, UART port has been used to send localization results to computer in this project. If Microblaze is not used, UART protocol should also be realized on FPGA device.

Microblaze embedded processor is a reduced instruction set computer (RISC) and it is optimized for implementation in Xilinx FPGAs [17]. Architecture of Microblaze processor is also shown in figure 2.12.



**Figure 2.12:** Architecture of Microblaze [17].

Microblaze embedded processor is highly configurable which means that the designers are allowed to select specific features required in their projects. Essential fixed properties of this processor are stated below [17].

- 32-bit general purpose registers
- 32-bit instruction set word with three operands and two addressing modes
- 32-bit address bus
- Single issue pipeline

Microblaze has a floating point unit (FPU) which means it is able to process floating numbers. In this project floating numbers need to be utilized and after activating floating point unit, Microblaze is automatically modified to process with floating numbers.

### **2.3.7 Xilinx Integrated Synthesis Environment**

Integrated Software Environment (ISE) is a software interface which is developed by Xilinx company to implement digital designs on FPGAs. ISE software is compatible with both Verilog and VHDL languages. After digital systems are coded with preferred HDL, designers can synthesize, generate circuit schematics and test their designs. Testing can be performed with either ideal conditions or real conditions. More clearly, if the real time delays are taken into account, real conditions are considered. Availability of this test mechanism allow designers to observe operation performances of their designs. Moreover, after Translate, Map and Place&Route operations are performed, settlement of the design on FPGA can also be observed. In other words, measurement of occupied area, number of LUTs, slices are used on FPGA can observed. When the design is ready for implementation, it can be loaded into FPGA via a cable. After programming file (.bit) of the design is generated, this file is loaded into FPGA through either Adept tool or Impact tool.



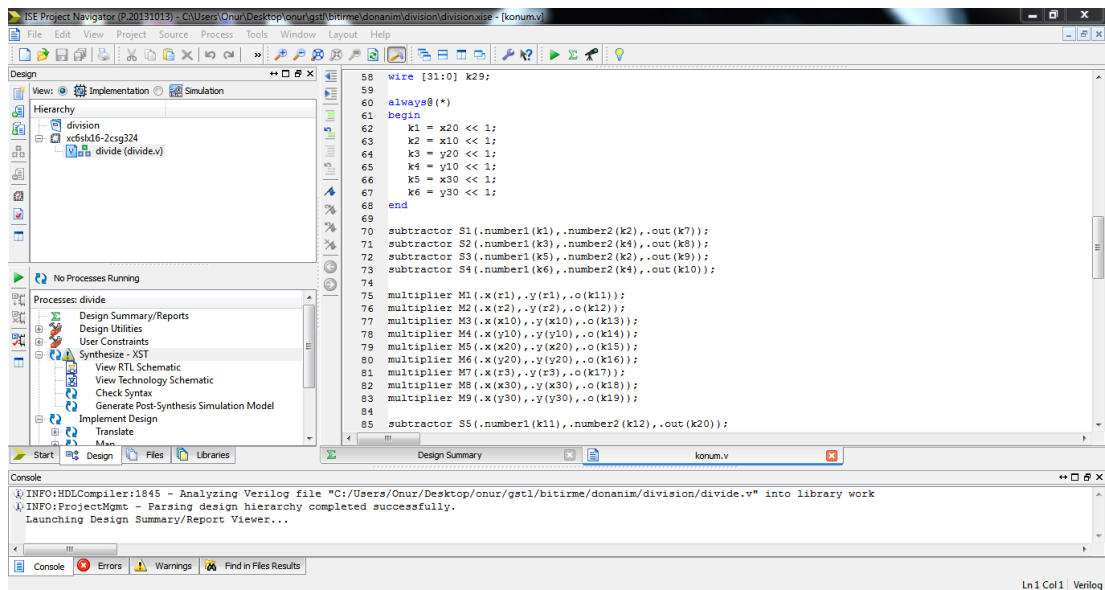
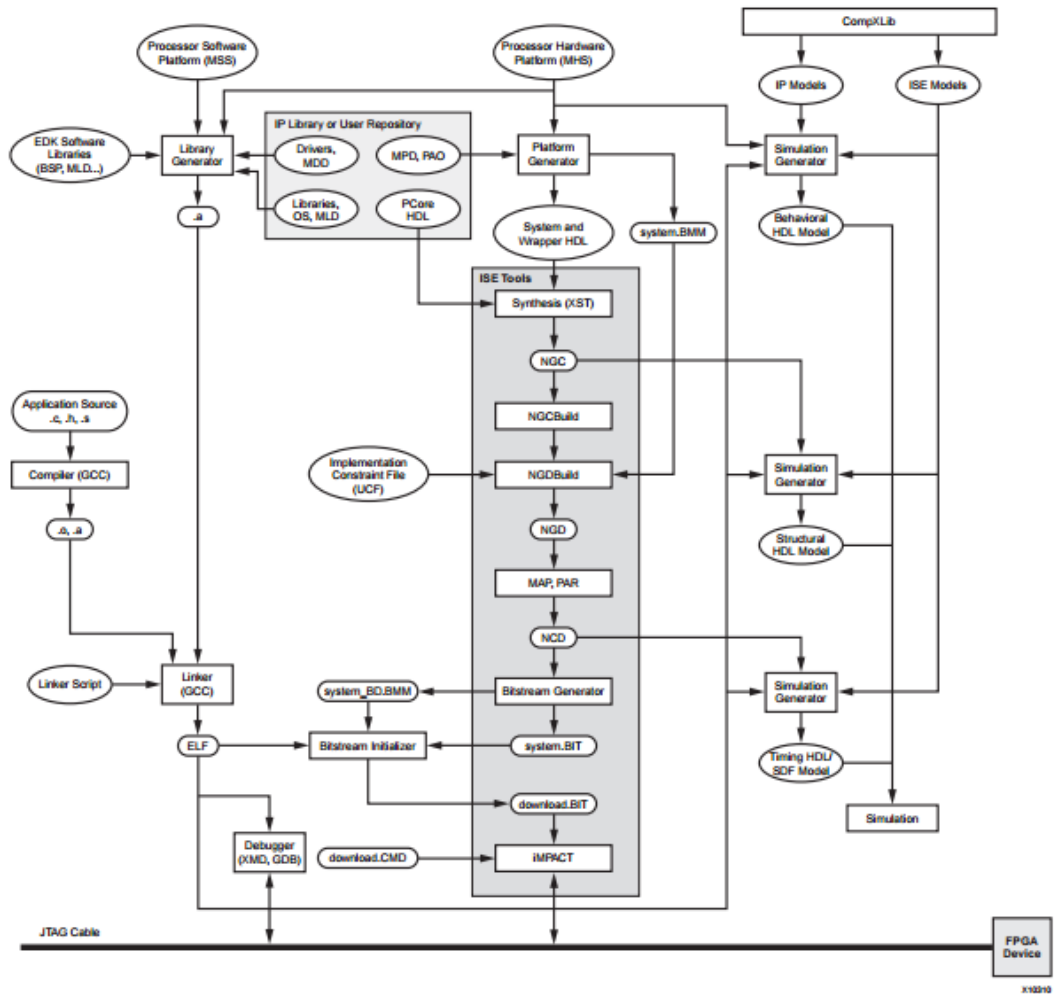


Figure 2.13: A view of Xilinx ISE tool.

### 2.3.8 Xilinx Embedded Development Kit

Embedded Development Kit (EDK) is developed by Xilinx to provide an opportunity to designers for building complex digital system including microprocessors. Depending on the preferences, a single processor system or a processor having connection with hardware parts could be designed. Processor properties such as clock frequency, required number of registers are easily adjusted in EDK. Also there are available peripheral protocols and which are provided by Xilinx to decrease the design effort and duration. For example, UART interface, buttons and leds can easily be used in the design, because EDK provide necessary libraries and functions of these protocols to be controlled with software [18]. An example system design which uses Microblaze is shown in figure 2.14.



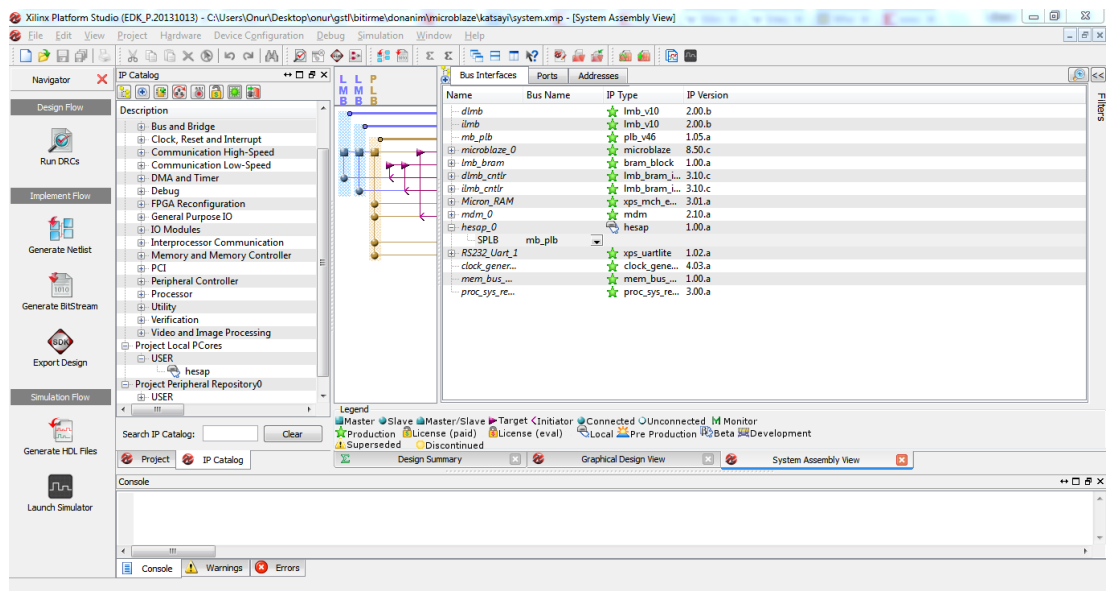
**Figure 2.14:** EDK architecture to create an embedded system [18].

Xilinx EDK software is composed of two software tools that are Xilinx Platform Studio (XPS) and Software Development Kit (SDK). Basically, XPS is used to determine the features of processor and SDK is used to realize software implementation with C or C++.

### 2.3.8.1 Xilinx Platform Studio

As stated in previous section, XPS enables designers to determine the characteristics of the processor which is implemented on FPGA. Base System Builder is a tool in XPS where the properties of the processor are determined. After project folder is determined, type of FPGA board is entered. For example, Digilent Nexys 3 board is added to project with its related library files. Then clock frequency and floating point options are determined. Also dual processor feature is available, but single processor is met the demands of this project. Finally, necessary peripherals are added to the

design. For instance, UART peripheral and Micron\_RAM have been added to our system. According to these decisions, XPS creates related processor system. Hardware modules which operates under the control of Microblaze have also been added to the system. Adding all hardware HDL files to the design is necessary for EDK to be able construct the whole system. Additional hardware modules appears as user defined local pcores in the bus interfaces section. A view of XPS tool is shown in figure 2.15.



**Figure 2.15:** A view of Xilinx XPS tool.

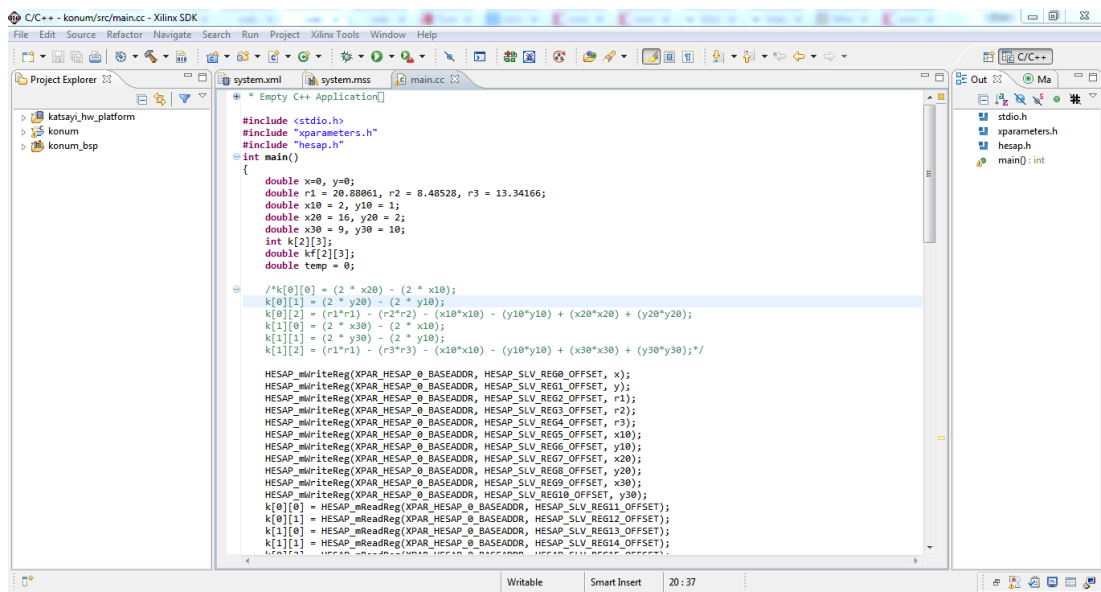
Before implement and export operations, the unaddressed cores need to be addressed. This can be performed with generate address button at the top-right corner of the program. After that, additional peripheral devices are addressed and addressing map is generated. Then the design becomes ready for implement and export operation.

### 2.3.8.2 Xilinx Software Development Kit

SDK is provided by Xilinx to perform software applications on the processor that is implemented in XPS. Certain important features of SDK platform are listed below [18].

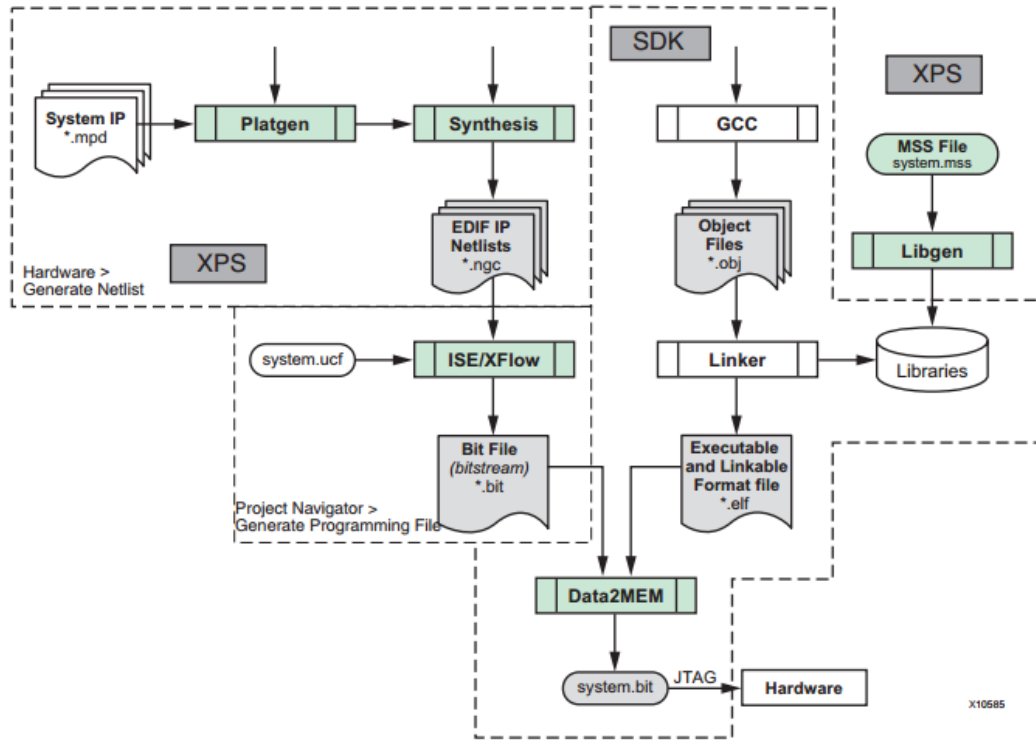
- Feature-rich C/C++ code editor and compilation environment
- Compatibility with single- or multi-processor systems
- Support of error navigation
- Modification of design structure and automatic make file generation
- Control of source code version

After the design is exported and SDK tool is launched, a new application project is created to construct the software part of the system with either C or C++. User defined hardware modules are converted to libraries that can be used in these programming languages. For example, to perform data exchange between the processor and the hardware related write and read functions need to be used. Declarations and descriptions of these functions are defined in a C/C++ library with same name of user defined hardware core. Obviously, this library should be included in program before these write and read functions are performed. A view of SDK platform is shown in figure 2.16.



**Figure 2.16:** A view Xilinx SDK tool.

To run the whole system on FPGA, a file which includes hardware information and another file which includes software information of the design should be loaded into FPGA. The file that holds the hardware data has .bit file extension and the file that holds software data has .elf file extension. Before these files are sent to FPGA, they are combined together. The schematic in figure 2.17 explains the whole implementation operation.



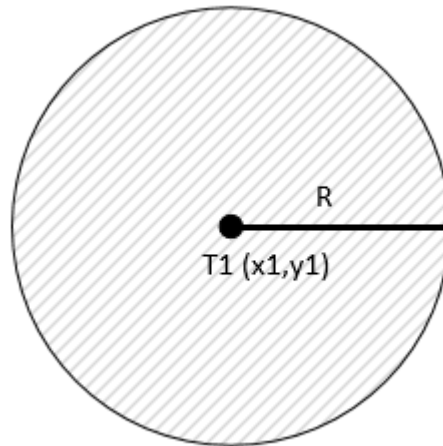
**Figure 2.17:** Implementation hardware/software design [19].

### **3. LOCALIZATION ALGORITHM**

Localization is a set of mathematical operations to obtain the position information of an object or a person. For this project, it has been decided to realize trilateration algorithm for indoor localization. The term trilateration refers to process of determining locations by measurements of distance using some geometrical figures such as circles, spheres and triangles. According to the requirements of projects, necessary geometrical figures are decided. For instance, circles are preferred in two dimensional localization applications while spheres are required in three dimensional localization. Generally 2D algorithms is sufficient for indoor localization and a 2D trilateration algorithm has been realized in this study. 3D localization is mostly preferred in space localizations like satellites. However 3D localization can also be required for indoor localization in some applications. For example, if building has a large ceiling height and there are some balconies in the building, 3D localization become necessary. Both 2D and 3D localization algorithms are explained in the following sections but only 2D localization algorithm has been implemented.

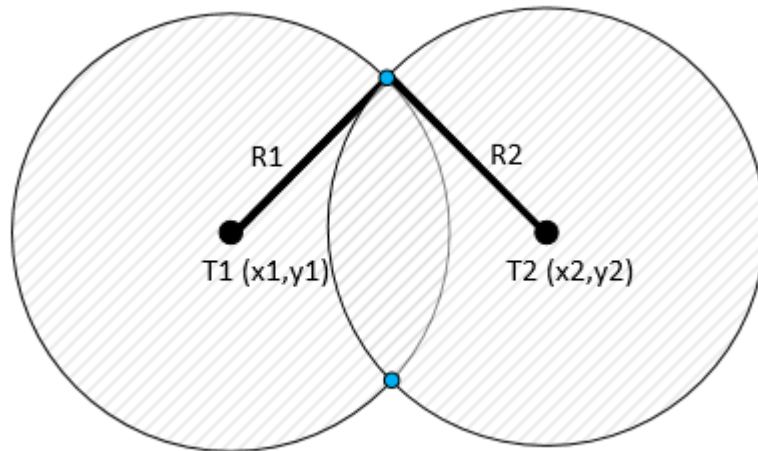
#### **3.1 2D Trilateration Algorithm**

Two dimensional trilateration algorithm is a localization method which uses circle geometry in a plane. Therefore, the algorithm performs through the geometrical formulas of circles. There are at least three reference points and corresponding distance values are required for 2D localization [20]. Reference points indicate the tags on the wall and distance values are obtained from related RSSI values. Number of required circles is determined according to resulting possible solutions. When only one distance value of a tag is considered, this corresponds to infinite possible solutions around the tag with same distances. In other words, possible solutions corresponds a circle as shown in figure 3.1.



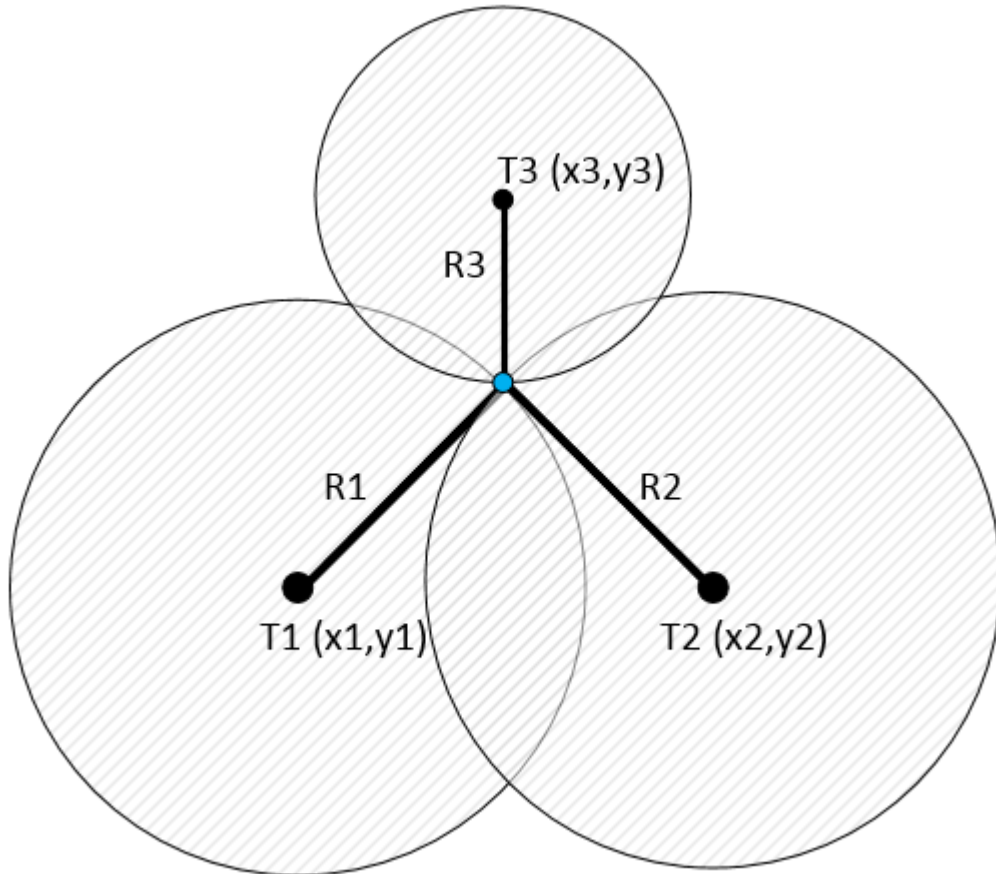
**Figure 3.1:** Infinite solutions with one tag.

$x_1$  and  $y_1$  are coordinates of tags in  $x$  axis and  $y$  axis respectively. Taking second tag into account, decreases the number of possible locations from infinite to two. Obviously the reason is that two circles intersect each other at two points as shown in figure 3.2.



**Figure 3.2:** Two possible solutions with two tag.

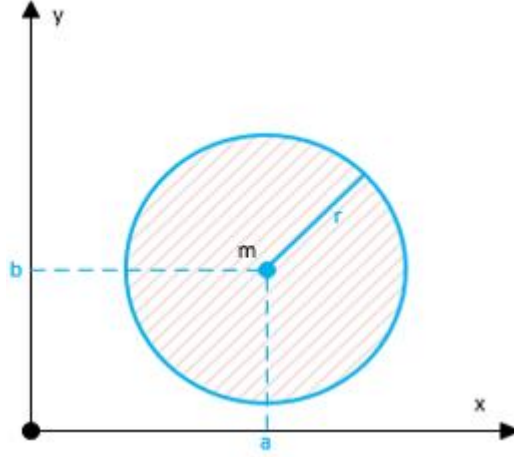
There is not a unique solution obtained yet and that is why the third tag is necessary. After the third tag is participate in the operation the unique solution is obtain as shown in figure 3.3.



**Figure 3.3:** The unique solution with three tags [20].

There is a constraint about how these locations of tags that all three tags should not settled on the same line. Because if this situation occurs, the tags can not be construct a plane and the number of possible solutions can not be reduced to one. Therefore when localization system is implemented a building this situation should be taken into consideration. For example, the tags could be located on different walls to construct a plane. Localization operation proceeds on the circle equations according to Cartesian coordinates which is shown in figure 3.4.





**Figure 3.4:** Geometry of a circle.

$$(x - a)^2 + (y - b)^2 = r^2 \quad (3.1)$$

When considering all three tags with respect to readers position following equations obtained.

$$\begin{aligned} (x - x_1)^2 + (y - y_1)^2 &= r_1^2 \\ (x - x_2)^2 + (y - y_2)^2 &= r_2^2 \\ (x - x_3)^2 + (y - y_3)^2 &= r_3^2 \end{aligned} \quad (3.2)$$

In the equations,  $x_1, x_2, x_3, y_1, y_2, y_3$  indicate coordinates of tags and  $r_1, r_2, r_3$  indicate distance values between the reader and related tags.  $x$  and  $y$  are the coordinates of the reader which are wanted be determined. After proceeding exponential operations of parenthesis, the equations are obtained as below.

$$\begin{aligned} x^2 - 2x(x_1) + x_1^2 + y^2 - 2y(y_1) + y_1^2 &= r_1^2 \\ x^2 - 2x(x_2) + x_2^2 + y^2 - 2y(y_2) + y_2^2 &= r_2^2 \\ x^2 - 2x(x_3) + x_3^2 + y^2 - 2y(y_3) + y_3^2 &= r_3^2 \end{aligned} \quad (3.3)$$

The problem in these equations is squares of  $x$  and  $y$ , because it prevents calculation of  $x$  and  $y$  with linear analysis such as Gaussian elimination. It is serious problem in terms of realization conditions and it is simpler to realize with linear equation solvers. In this project, Gaussian elimination method is used to solve the linear equations. Therefore, to get rid of squares of  $x$  and  $y$ , the equations are subtracted from each other sequentially and results are obtained as below.

$$x(-2(x_1) + 2(x_2)) + y(-2(y_1) + 2(y_2)) + x_1^2 + y_1^2 - x_2^2 - y_2^2 = r_1^2 - r_2^2$$

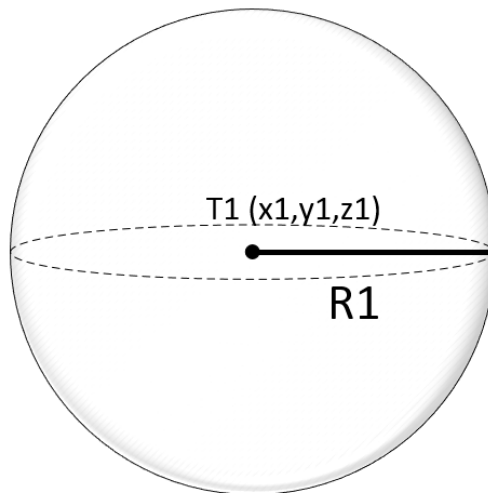
$$x(-2(x_1) + 2(x_3)) + y(-2(y_1) + 2(y_3)) + x_1^2 + y_1^2 - x_3^2 - y_3^2 = r_1^2 - r_3^2$$

$$x(-2(x_2) + 2(x_3)) + y(-2(y_2) + 2(y_3)) + x_2^2 + y_2^2 - x_3^2 - y_3^2 = r_2^2 - r_3^2 \quad (3.4)$$

As seen, three linear equations with unknown  $x$  and  $y$  values are obtained while all other parameters are known. By using linear equation solver  $x$  and  $y$  values are easily calculated and Gaussian elimination method has realized and used in this study.

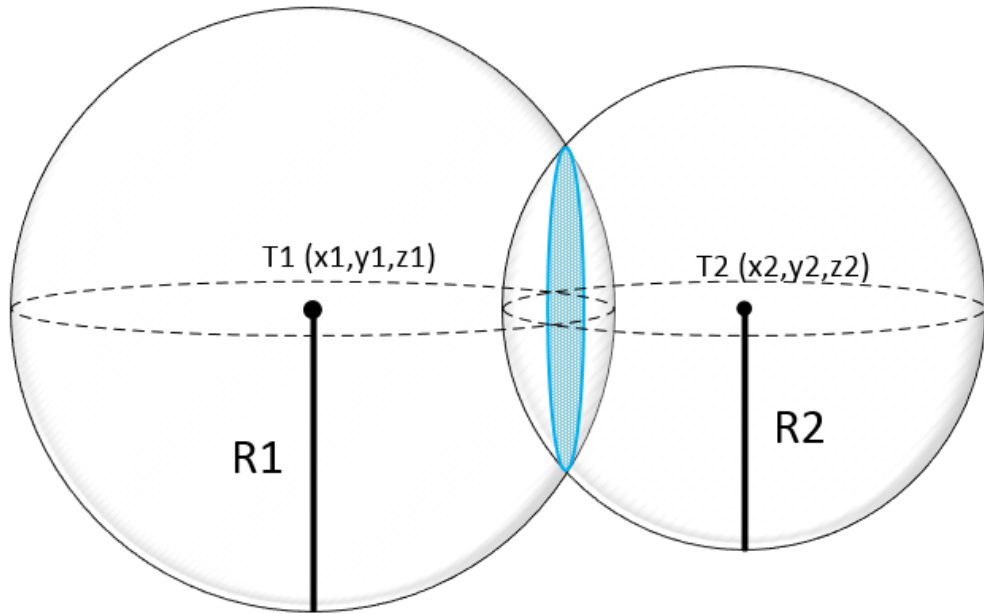
### 3.2 3D Trilateration Algorithm

Three dimensional trilateration algorithm is a localization method which uses sphere geometry in a space. This method is generally preferred for space and outdoor applications. The concept is very similar to 2D trilateration and additional  $z$  coordinate is also taken into account. Just like in 2D trilateration, number of required spheres is determined according to resulting possible solutions. When only one distance value of a tag is considered, this corresponds to infinite possible solutions around the tag with same distances. In other words, possible solutions corresponds a sphere as shown below in figure 3.5.



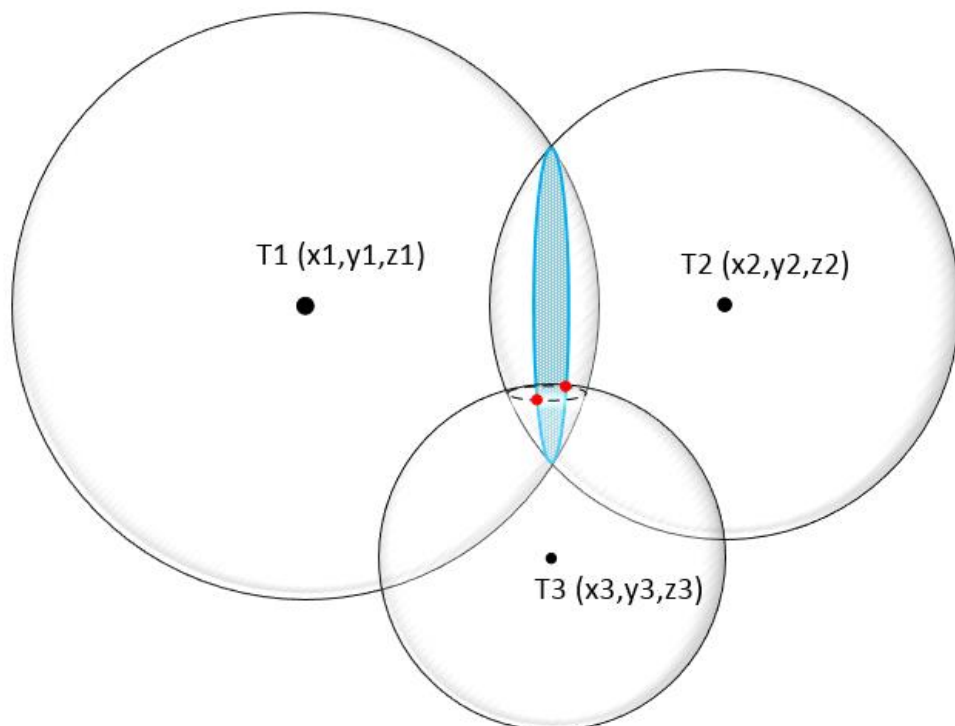
**Figure 3.5:** Infinite solutions with one tag [21].

Taking second tag into account, number of possible locations stays still infinite. However, with one tag, possible solutions are all the points on sphere surface and with participation of two tags, possible solutions become points on a circle. The circle is constructed of two spheres as shown in figure 3.6.



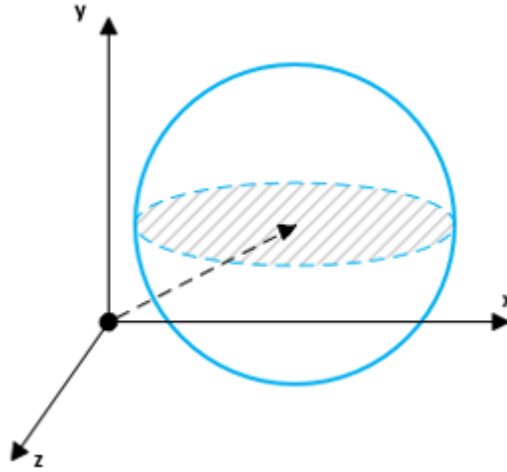
**Figure 3.6:** Infinite solutions with two tags [21].

To decrease number of solutions, additional third tag is necessary. Also third tag should not be located in the same line with other two tags and should be a plane as shown below.



**Figure 3.7:** Two possible solutions with three tags [21].

Red dots represents the two possible solutions and to obtain the unique localization value fourth tag is necessary. While adding the fourth tag, it is necessary to locate it in a position that it should not be in a plane with other tags. Because the fourth tag should be in a position that whole system can construct a space and the unique solution is able to be obtained. Localization operation proceeds on the sphere equations according to Cartesian coordinates which is shown in figure 3.8.



**Figure 3.8:** Geometry of a sphere.

$$(x - a)^2 + (y - b)^2 + (z - c)^2 = r^2 \quad (3.5)$$

With the participation of all four tags construct the following equation set.

$$\begin{aligned} (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 &= r_1^2 \\ (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 &= r_2^2 \\ (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 &= r_3^2 \\ (x - x_4)^2 + (y - y_4)^2 + (z - z_4)^2 &= r_4^2 \end{aligned} \quad (3.6)$$

In the equations,  $x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4$  indicate coordinates of tags and  $r_1, r_2, r_3, r_4$  indicate distance values between the reader and related tags.  $x, y$  and  $z$  are the coordinates of the reader which are wanted be determined. After proceeding exponential operations of parenthesis, the equations are obtained as below.

$$\begin{aligned} x^2 - 2x(x_1) + x_1^2 + y^2 - 2y(y_1) + y_1^2 + z^2 - 2z(z_1) + z_1^2 &= r_1^2 \\ x^2 - 2x(x_2) + x_2^2 + y^2 - 2y(y_2) + y_2^2 + z^2 - 2z(z_2) + z_2^2 &= r_2^2 \\ x^2 - 2x(x_3) + x_3^2 + y^2 - 2y(y_3) + y_3^2 + z^2 - 2z(z_3) + z_3^2 &= r_3^2 \end{aligned}$$

$$x^2 - 2x(x_4) + x_4^2 + y^2 - 2y(y_4) + y_4^2 + z^2 - 2z(z_4) + z_4^2 = r_4^2 \quad (3.7)$$

Just like in 2D localization, it necessary to get rid of squares of x, y and z to be able to implement linear equation solution method. For this reason the equations are subtracted from each other sequentially and results are obtained as below.

$$x(-2(x_1) + 2(x_2)) + y(-2(y_1) + 2(y_2)) + z(-2(z_1) + 2(z_2)) + x_1^2 + y_1^2 + z_1^2 - x_2^2 - y_2^2 - z_2^2 = r_1^2 - r_2^2$$

$$x(-2(x_1) + 2(x_3)) + y(-2(y_1) + 2(y_3)) + z(-2(z_1) + 2(z_3)) + x_1^2 + y_1^2 + z_1^2 - x_3^2 - y_3^2 - z_3^2 = r_1^2 - r_3^2$$

$$x(-2(x_2) + 2(x_3)) + y(-2(y_2) + 2(y_3)) + z(-2(z_2) + 2(z_3)) + x_2^2 + y_2^2 + z_2^2 - x_3^2 - y_3^2 - z_3^2 = r_2^2 - r_3^2$$

$$x(-2(x_1) + 2(x_4)) + y(-2(y_1) + 2(y_4)) + z(-2(z_1) + 2(z_4)) + x_1^2 + y_1^2 + z_1^2 - x_4^2 - y_4^2 - z_4^2 = r_1^2 - r_4^2$$

(3.8)

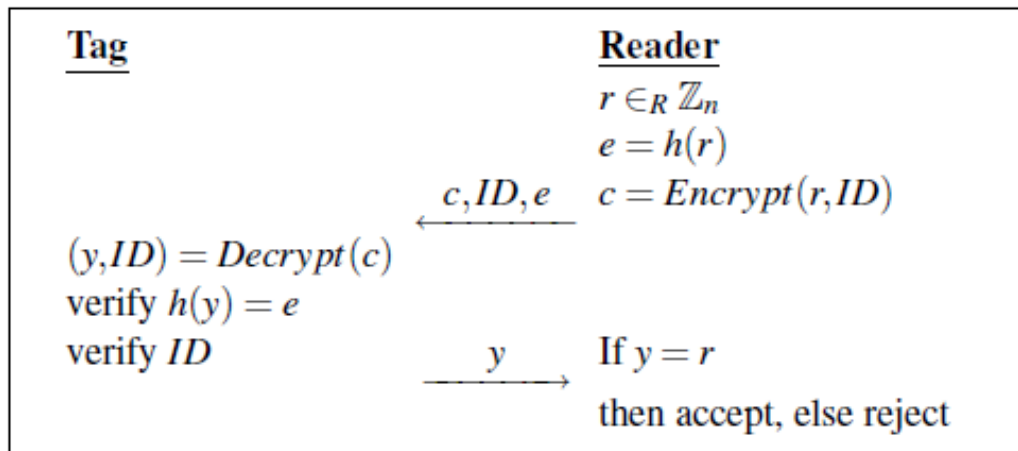
Now linear equation solution method such as Gaussian elimination method could be implemented to obtain x, y and z coordinates of reader, because only linear equations are left.

#### 4. RFID AUTHENTICATION PROTOCOL

In a RFID system the tags send related information to the reader by transmitting electromagnetic waves into air. This operation has some security gaps if necessary precautions are not taken. For example, localization information of people can be seriously abused by unauthorized receivers. To not face such situations, the tags need to ensure that they send their information only to the authorized reader. For this reason, a cryptology protocol which provides entity authentication based on encryption scheme is proposed.

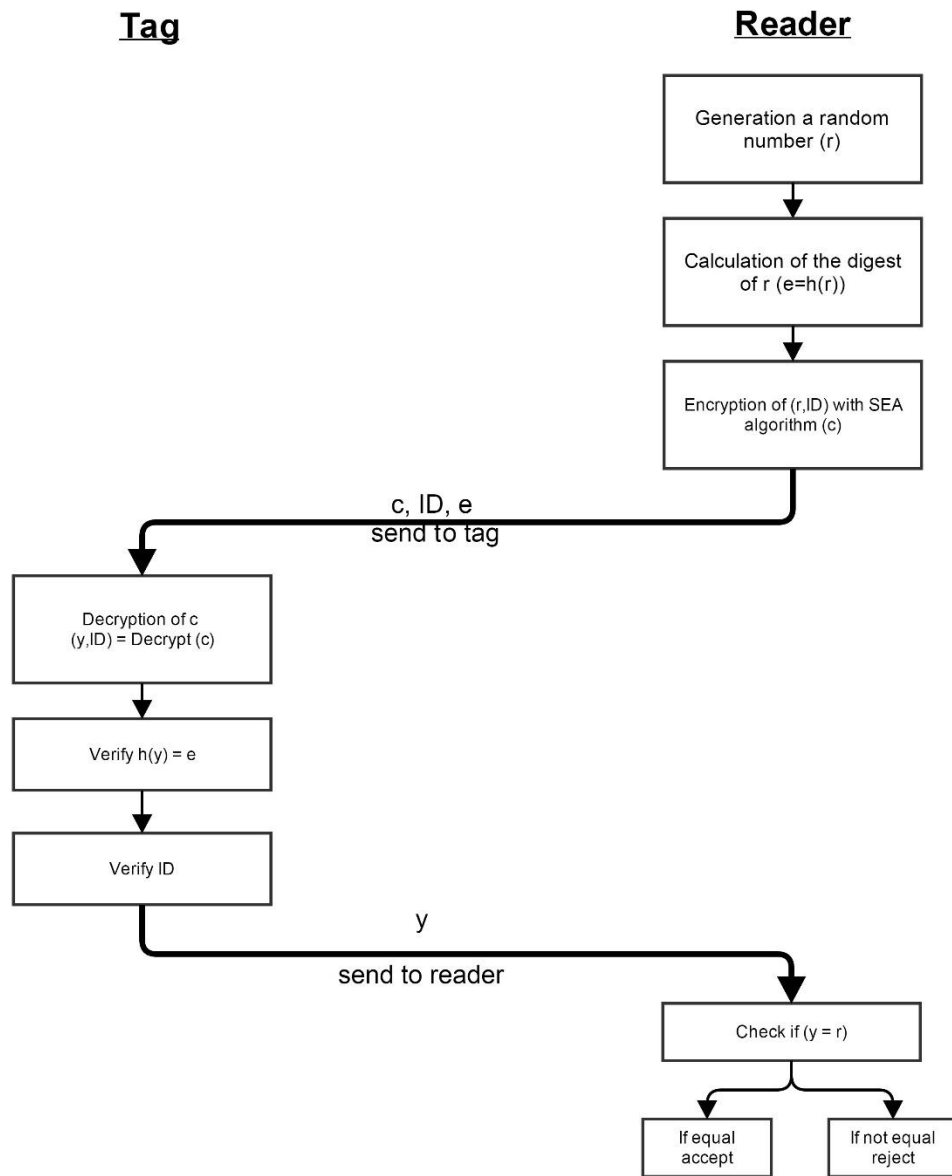
Authentication protocols are mainly used to prevent impersonation with passive and active attacks. Passive attacks obtain the information by passively monitoring (eavesdropping) the communication channel. On the other hand, an unauthorized reader show itself as verifier and obtain the information by interacting with the prover [22].

The proposed protocol in this study is at least provably secure against passive attacks. Also public key method is used and it is presumed that the reader is informed about the ID and the public key of the related tag. Principle of entity authentication protocol is shown in figure 4.1.



**Figure 4.1:** Principle of authentication protocol [22].

The schematic of whole system between tag and reader is also shown in figure 4.2.



**Figure 4.2:** Authentication protocol operation.

First, the reader generates a random number  $r$  and digest of this number is calculated with an one-way hash function. Then the random number ( $r$ ) and ID are encrypted with an encryption algorithm. Scalable Encryption Algorithm (SEA) is chosen as the encryption algorithm in this project. After encryption process, the encrypted value ( $c$ ), output of hash function ( $e$ ) and ID are sent to tag. When the tag receives this information, decryption operation is performed first. Then the random number is obtained from output of decryption function (SEA) and this value is pushed into hash function to check if same digest ( $e$ ) is obtained with reader. Also ID value is checked and decrypted random number value ( $y$ ) is sent back to the reader. Finally reader

checks if the  $y$  is equal to  $r$  or not and according to the result, decides what is going to be performed [22].

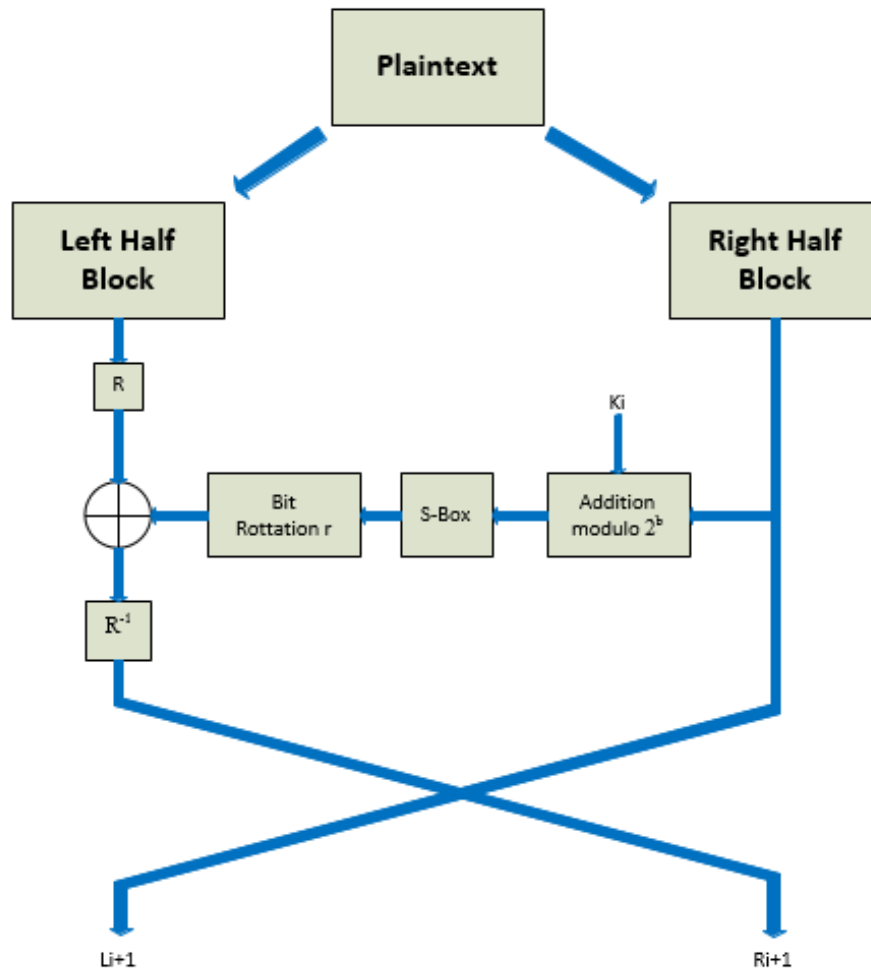
#### 4.1 Scalable Encryption Algorithm

In this project, SEA is selected to perform encryption/decryption operations in the protocol.  $SEA_{n,b}$  operates on various text, key and word sizes. It is based on a Feistel structure with a variable number of rounds, and is defined with respect to the following parameters [23]:

- $n$ : plaintext size, key size.
- $b$ : processor (or word) size.
- $n_b = n/2b$  : number of words per Feistel branch.
- $n_r$ : number of block cipher rounds.

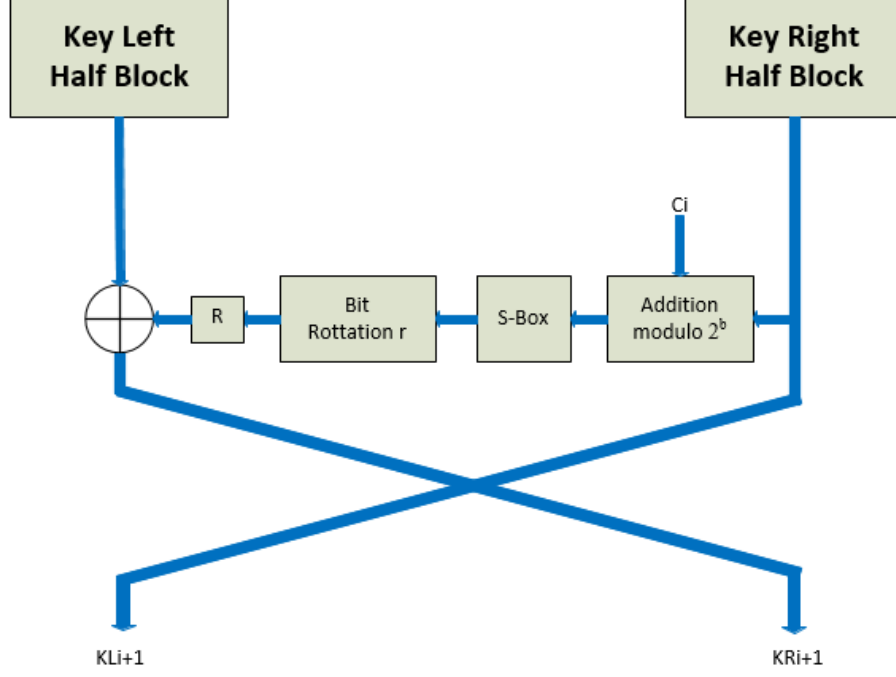
As only constraint, it is required that  $n$  is a multiple of  $6b$ . For example, using an 8-bit processor, we can derive 48, 96, 144, . . . bit block ciphers, respectively denoted as  $SEA_{48,8}$ ,  $SEA_{96,8}$ ,  $SEA_{144,8}$ , ... Block diagram of encryption and decryption round which will be implemented is shown figure 4.3.





**Figure 4.3:** Encryption/Decryption round [23].

As seen, this is first round for encryption and the same process will be performed according to number of rounds. For decryption operation, structure is exactly same, but key schedule is used in reversed order. Block diagram of key round which will be implemented to obtain key schedule is also shown in figure 4.4.



**Figure 4.4:** Key Scheduling round [23].

#### 4.1.1 Basic Operations

In this part, basic operations or sub-blocks that need to be implemented are explained shortly. There are five elementary operations that can be easily implemented to small embedded systems.

The bitwise exclusive or (EXOR) is defined on  $n/2$  bit vector as shown in formula (4.1) and the lower case letter  $i$  is used as bit index. Therefore all bits of  $x$  and  $y$  are inserted into EXOR operation.

$$\oplus : \mathbb{Z}_2^{\frac{n}{2}} \times \mathbb{Z}_2^{\frac{n}{2}} \rightarrow \mathbb{Z}_2^{\frac{n}{2}} : x, y \rightarrow z = x \oplus y \Leftrightarrow z(i) = x(i) \oplus y(i), 0 \leq i \leq \frac{n}{2} - 1 \quad (4.1)$$

$SEA_{n,b}$  uses a 3-bit substitution box (S) that  $S_T = \{0,5,6,7,4,3,1,2\}$  and evaluated according to expressions in (4.2). Also the lower case  $i$  is used as word index this time.

$$S : \mathbb{Z}_2^{n_b} \rightarrow \mathbb{Z}_2^{n_b} : x \rightarrow x = S(x) \Leftrightarrow$$

$$x_{3i} = (x_{3i+2} \wedge x_{3i+1}) \oplus x_{3i},$$

$$x_{3i+1} = (x_{3i+2} \wedge x_{3i}) \oplus x_{3i+1},$$

$$x_{3i+2} = (x_{3i} \vee x_{3i+1}) \oplus x_{3i+2}, \quad 0 \leq i \leq \frac{n_b}{3} - 1, \quad (4.2)$$

where  $\wedge$  and  $\vee$  respectively represent the bitwise AND and OR.

The word rotation is defined on  $n_b$ -word vectors as shown in equations (4.3).

$$R : \mathbb{Z}_{2^b}^{n_b} \rightarrow \mathbb{Z}_{2^b}^{n_b} : x \rightarrow y = R(x) \Leftrightarrow y_{i+1} = x_i, \quad 0 \leq n_b - 2,$$

$$y_0 = x_{n_b - 1} \quad (4.3)$$

Also the bit rotation is defined on  $n_b$ -word vectors as shown in equations (4.4).

$$r : \mathbb{Z}_{2^b}^{n_b} \rightarrow \mathbb{Z}_{2^b}^{n_b} : x \rightarrow y = r(x) \Leftrightarrow y_{3i} = x_{3i} \ggg 1,$$

$$y_{3i+1} = x_{3i+1},$$

$$y_{3i+2} = x_{3i+2} \lll 1, \quad 0 \leq i \leq \frac{n_b}{3} - 1 \quad (4.4)$$

where  $\ggg$  and  $\lll$  represent the cyclic right and left shifts inside a word.

Finally, the mod  $2b$  addition is defined on  $n_b$ -word vectors as shown in equation (4.5).

$$\boxplus : \mathbb{Z}_{2^b}^{n_b} \times \mathbb{Z}_{2^b}^{n_b} \rightarrow \mathbb{Z}_{2^b}^{n_b} : x, y \rightarrow z = x \boxplus y \Leftrightarrow z_i = x_i \boxplus y_i,$$

$$0 \leq i \leq n_b - 1 \quad (4.5)$$

#### 4.1.2 The Round and Key Round

The mathematical representation of encryption, decryption and key rounds is also shown in equations (4.6).

$$[L_{i+1}, R_{i+1}] = F_E(L_i, R_i, K_i) \quad \Leftrightarrow \quad R_{i+1} = R(L_i) \oplus r(S(R_i \boxplus K_i))$$

$$L_{i+1} = R_i$$

$$[L_{i+1}, R_{i+1}] = F_D(L_i, R_i, K_i) \quad \Leftrightarrow \quad R_{i+1} = R^{-1}(L_i \oplus r(S(R_i \boxplus K_i)))$$

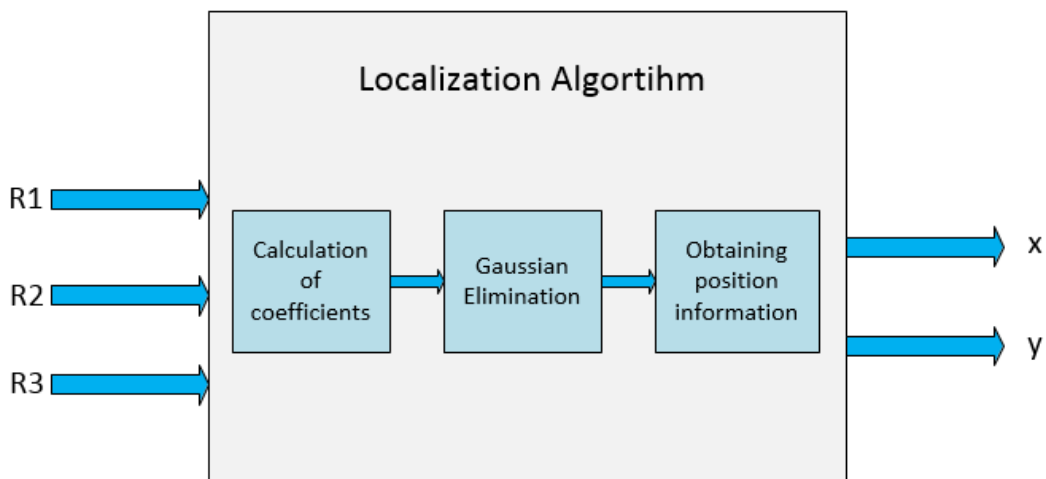
$$L_{i+1} = R_i$$

$$[KL_{i+1}, KR_{i+1}] = F_K(KL_i, KR_i, C_i) \quad \Leftrightarrow \quad KR_{i+1} = KL_{i+1} \oplus R(r(S(KR_i \boxplus C_i)))$$

$$KL_{i+1} = KR_{i+1} \quad (4.6)$$

## 5. IMPLEMENTATION OF LOCALIZATION ALGORITHM

As stated before, two dimensional trilateration algorithm has been realized to obtain localization information of people in a building. At least three tags are necessary to participate in localization operation to obtain the unique solution in two dimension. It is presumed that the locations of tags are known. When the reader enters transmission range of a tag it, entity authentication protocol is performed at first. After the tag is ensured of authorization of the reader, transmit radio waves to the reader. The reader makes a distance estimation according to the RSSI. It is 8-bit unsigned number and a distance-RSSI table is used to obtain the distance between the tag and the reader. Same operation is performed between the reader and different three tags at three time. When three different distance information is acquired the localization algorithm starts to obtain position information. This algorithm is composed of three parts which are shown in figure 5.1.



**Figure 5.1:** Basic operations of localization algorithm.

First, the algorithm has been realized and tested on Visual Studio. After ensuring that the algorithm operates properly, it has been implemented on Microblaze to perform localization operation on FPGA device. Finally, first part of the algorithm which is calculation of coefficients has been designed as hardware. Microblaze processor

operates with this hardware module properly and hardware-software design has been checked.

## 5.1 Realization on General Purpose Processor

In this part, the algorithm has been realized with C++ on Visual Studio. This is purely software design for ensuring that the 2D trilateration algorithm operates properly. First, related three circle equations are formed according to obtained three distance values. Then to get rid of squares of x and y, the equations are subtracted from each other as explained in Section 3. After that, three first order equations are left and x and y values could be obtained using a linear equation solver. Gaussian Elimination method has been preferred to be used and the coefficients in the equations have been assigned to a two dimensional array to perform Gaussian Elimination as shown in figure 5.2.

$$\begin{array}{r}
 x(-2*x1 + 2*x2) + y(-2*y1 + 2*y2) + x1^2 + y1^2 - x2^2 - y2^2 = r1^2 - r2^2 \\
 \hline
 k[0][0] \qquad k[0][1] \qquad k[0][2] \\
 \\
 x(-2*x1 + 2*x3) + y(-2*y1 + 2*y3) + x1^2 + y1^2 - x3^2 - y3^2 = r1^2 - r3^2 \\
 \hline
 k[1][0] \qquad k[1][1] \qquad k[1][2] \\
 \\
 x(-2*x2 + 2*x3) + y(-2*y2 + 2*y3) + x2^2 + y2^2 - x3^2 - y3^2 = r2^2 - r3^2 \\
 \hline
 k[2][0] \qquad k[2][1] \qquad k[2][2]
 \end{array}$$

**Figure 5.2:** Coefficient array initialization.

Gaussian Elimination is based on matrix operations and it is required that columns in the matrix should not be zero to obtain the solution. Although all possible combinations after subtraction requires a 3x3 two dimensional array, working with 2x3 size matrix is enough to obtain the solution. Therefore, just first two equations have been used in the algorithm. As stated in Section 3, x1, x2, x3, y1, y2, y3 indicate coordinates of tags and r1, r2, r3 indicate distance values between the reader and related tags. After the Gaussian Elimination is performed x and y values are obtained which means that the location of the reader is determined. The pseudocode of this algorithm is shown below.

**Initialize** r1, r2, r3 to distance values

**Initialize** x1, y1 to position of tag1

**Initialize** x2, y2 to position of tag2

**Initialize** x3, y3 to position of tag3

**Declare** a 2D double array k

**Set** k[0][0] to  $(2 * x2) - (2 * x1)$

**Set** k[0][1] to  $(2 * y2) - (2 * y1)$

**Set** k[0][2] to  $(r1)^2 - (r2)^2 - (x1)^2 - (y1)^2 + (x2)^2 + (y2)^2$

**Set** k[1][0] to  $(2 * x3) - (2 * x1)$

**Set** k[1][1] to  $(2 * y3) - (2 * y1)$

**Set** k[1][2] to  $(r1)^2 - (r3)^2 - (x1)^2 - (y1)^2 + (x3)^2 + (y3)^2$

**If** k[0][0] is 0 and k[1][0] is not 0

**For** i=0 to i=2 **do**

Temp = k[1][i]

k[1][i] = k[0][i]

k[0][i] = Temp

**end for**

**end if**

**Else if** k[0][0] is 0 and k[1][0] is 0

**Print** "First column is 0 and the unique solution can not be obtained"

**end if**

**For** i=0 to i=1

Temp =  $(k[i][0]) / (k[0][0])$

**For** t=0 to t=2

$k[i][t] = (k[i][t]) - temp * (k[0][t])$

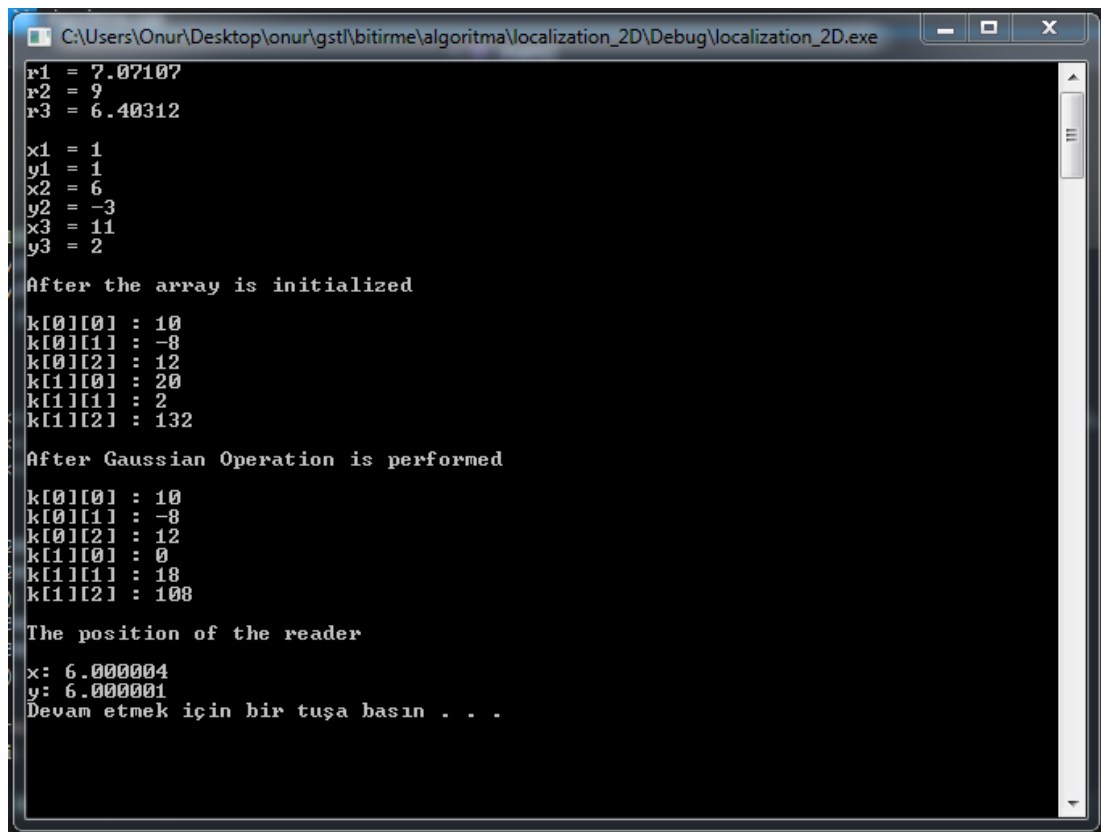
**end for**

**end for**

$$y = (k[1][2]) / (k[1][1])$$

$$x = ((k[0][2]) - (k[0][1]) * y) / (k[0][0])$$

After the program has been written via C++, it has been tested with some virtual position and distance values. For example, according to an origin point x and y values of the reader and the tags are considered as (6,6), (1,1), (11,2), (6,-3) respectively. According to these positions, the distance values between the reader and the tags should be measured as 7.07107, 9 and 6.40312 in meters. As stated before it is presumed that the tag positions are known and using these distance values the reader is located as shown in figure 5.3.



```
C:\Users\Onur\Desktop\onur\gsl\bitirme\algoritma\localization_2D\Debug\localization_2D.exe
r1 = 7.07107
r2 = 9
r3 = 6.40312

x1 = 1
y1 = 1
x2 = 6
y2 = -3
x3 = 11
y3 = 2

After the array is initialized
k[0][0] : 10
k[0][1] : -8
k[0][2] : 12
k[1][0] : 20
k[1][1] : 2
k[1][2] : 132

After Gaussian Operation is performed
k[0][0] : 10
k[0][1] : -8
k[0][2] : 12
k[1][0] : 0
k[1][1] : 18
k[1][2] : 108

The position of the reader
x: 6.000004
y: 6.000001
Devam etmek için bir tuşa basın . . .
```

**Figure 5.3:** Localization results on general purpose processor.

As seen, the position of the reader has been obtained as equal to its real position. The precision of result could be increased with using more precise distance values and this precision is enough in terms of meters.

## **5.2 Realization on Microblaze**

General-purpose processors might be inadequate for speed performance or project costs. Also some drawbacks are able to emerge during implementation period. Considering these reasons, the localization algorithm has been realized on Microblaze to operate on FPGA device. First, it has been implemented purely with software and performed on Microblaze which is a virtual processor realized on FPGA. Then first part of the localization algorithm has been designed as hardware and an embedded system composed of a processor and a hardware module is implemented on FPGA.

### **5.2.1 Software Design**

Xilinx EDK tool which is composed of XPS and SDK parts has been utilized to realize the design. First, characteristics of the processor are determined in XPS tool. Floating point unit is enabled, UART and Micron\_RAM are added to design. UART has been used to print the localization results to the screen via UART USB cable. UART is a hardware unit that is used for transmission of information. It operates with communication standards and mostly RS-232 serial communication standard. On the other hand, Micron\_RAM is required in the system, because distributed RAM in FPGA device is not enough to perform the algorithm. This is caused form usage of floating numbers and size of the algorithm. Therefore the program is run over Micron\_RAM instead of local distributed RAM.

After the design of processor has been completed, hardware design is exported to SDK for implementation of software design. In SDK, a new application project which is compatible with C++ is created and the localization algorithm is implemented with a similar concept as in Section 5.1. As stated before, the program runs over Micron\_RAM which has 16 Mbyte memory capacity. After the program is completed, FPGA device is programmed and localization algorithm is launched on hardware. When position of the reader is determined, the results are sent to computer via UART and printed on screen. To print out the results, a UART console program which is called Putty is used. By the way, UART transmits the results to computer on COM3 port. To test the design same virtual position and distance values in section 5.1 have been used the results are obtained as shown in figure 5.4.



```

C:\Users\Onur\Desktop\onur\gsl\bitirme\algoritma\localization_2D\Debug\localization_2D.exe
r1 = 7.07107
r2 = 9
r3 = 6.40312

x1 = 1
y1 = 1
x2 = 6
y2 = -3
x3 = 11
y3 = 2

After the array is initialized
k[0][0] : 10
k[0][1] : -8
k[0][2] : 12
k[1][0] : 20
k[1][1] : 2
k[1][2] : 132

After Gaussian Operation is performed
k[0][0] : 10
k[0][1] : -8
k[0][2] : 12
k[1][0] : 0
k[1][1] : 18
k[1][2] : 108

The position of the reader
x: 6.000004
y: 6.000001
Devam etmek için bir tuşa basın . . .

COM3 - PuTTY
x = 6.0000041309
y = 6.0000012955

```

**Figure 5.4:** Localization results on both Microblaze and computer.

The obtained results from Microblaze is shown in the window at the bottom-right corner in figure 5.4. As seen localization information of the reader is obtained accurately. In SDK, it is not allowed the print floating numbers directly, because it is just able to print integer values. Therefore fraction parts of x and y values are printed on screen through additional integers. The written code part which performs printing out of floating numbers is shown in figure 5.5.

```

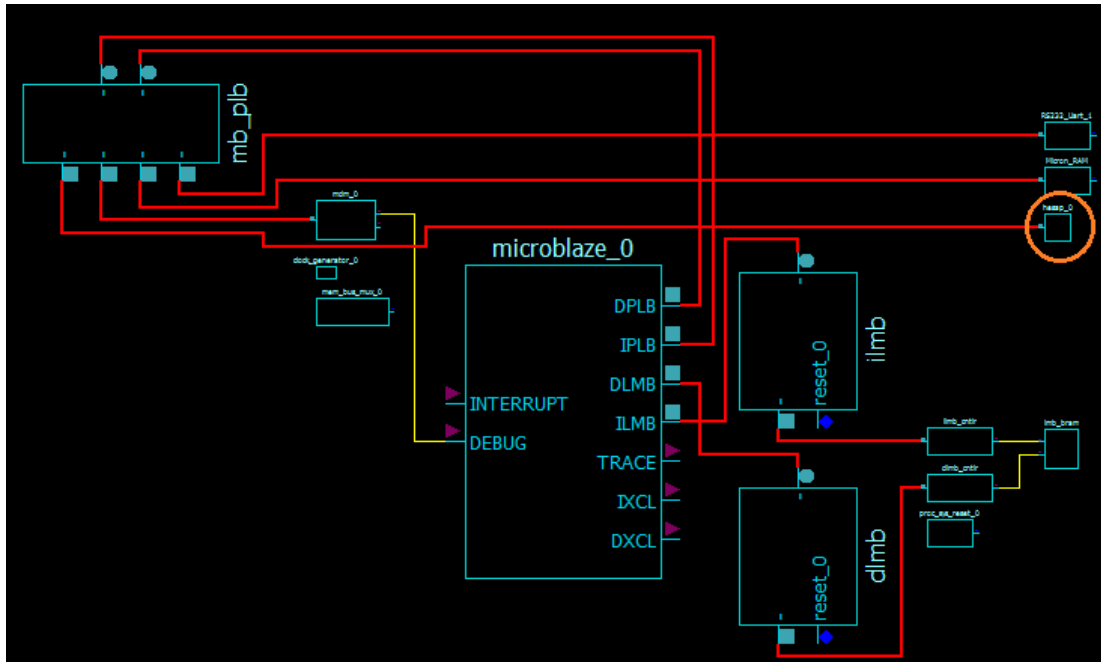
int whole1,whole2,thousands1,thousands2;
whole1=x;
whole2=y;
thousands1=(x-whole1)*1000000000;
thousands2=(y-whole2)*1000000000;
xil_printf("x = %d.%010d\n",whole1,thousands1);
xil_printf("y = %d.%010d\n",whole2,thousands2);

```

**Figure 5.5:** Floating number print code for Microblaze.

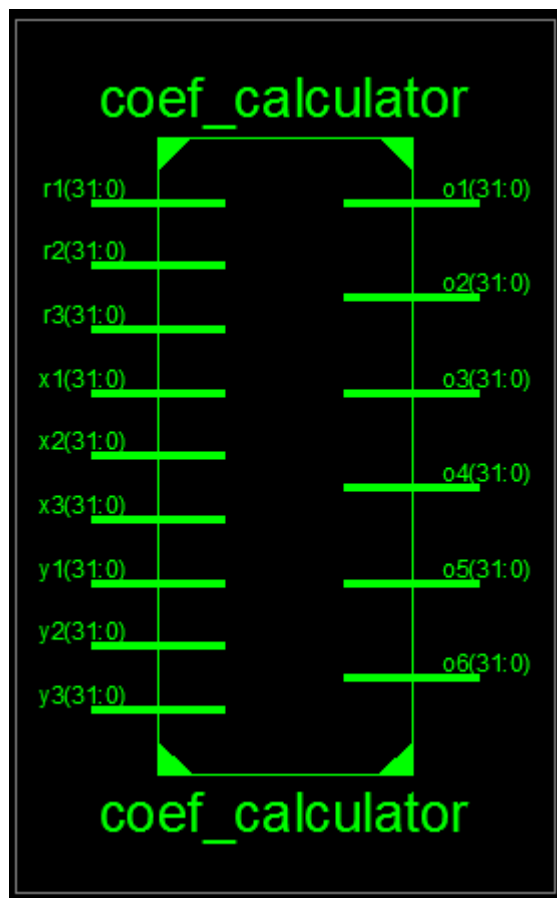
### 5.2.2 Hardware/Software Codesign

In this part, it is aimed to implement the localization algorithm completely in hardware using Verilog. To determine the exact position information of the reader, it is necessary to make mathematical calculations on floating numbers. Although Microblaze has floating point unit which allows to operate on floating numbers, transmission of floating numbers between Microblaze processor and user defined hardware modules are not allowed in EDK. This is caused from the write and read functions which are defined in the default libraries supplied by Xilinx. The functions are designed to work with 32 bit unsigned integers. By making some modifications in these libraries transmission of floating numbers are able to be performed. However, after certain parts of the algorithm has been designed as hardware, the communication between processor and additional hardware module could not perform properly. Although the hardware operates properly and calculate related coefficients accurately, after the results are transmitted to processor, the values obtained incorrectly. Due to this failure, first part of the algorithm which calculates coefficients to initialize the two dimensional array has been designed as hardware. After these coefficients are calculated on hardware module, they are sent back to processor and rest of the algorithm is performed in Microblaze and localization operation is completed. Because of floating error, hardware module takes position information of tags and distance information between the reader and the tags as integer values. Taking as integer eliminates the fraction parts and cause small differences in localization results. However these differences do not result in remarkable errors. The small deviations between real and obtained results are shown in the following parts. The graphical view of the system is shown in figure 5.6 and the added hardware module is indicated in an orange circle.



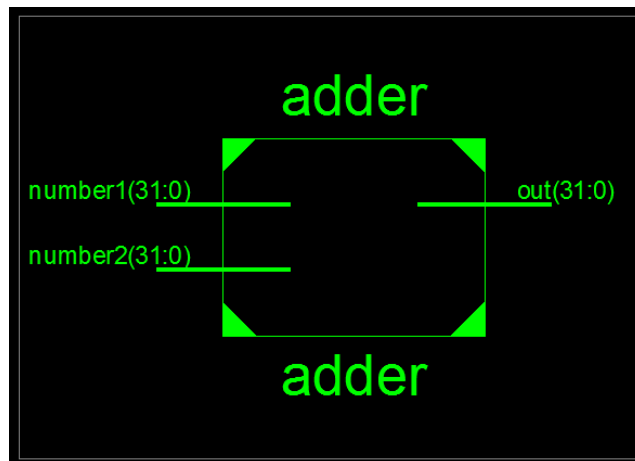
**Figure 5.6:** Microblaze structure with user defined hardware module.

Hardware module has been designed in ISE tool via Verilog. The block schematic of this module which shows its inputs and outputs is shown in figure 5.7.

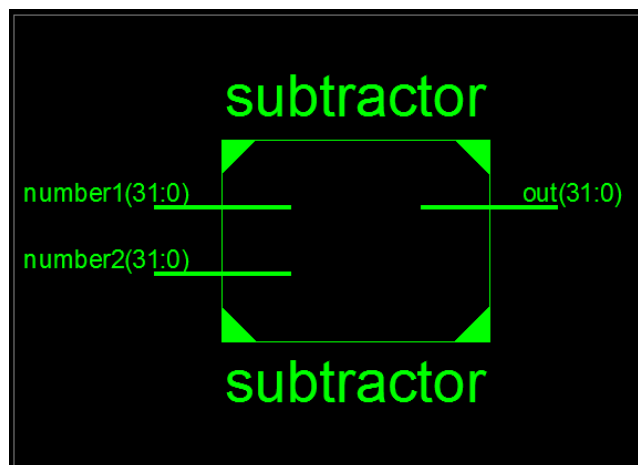


**Figure 5.7:** coef\_calculator module.

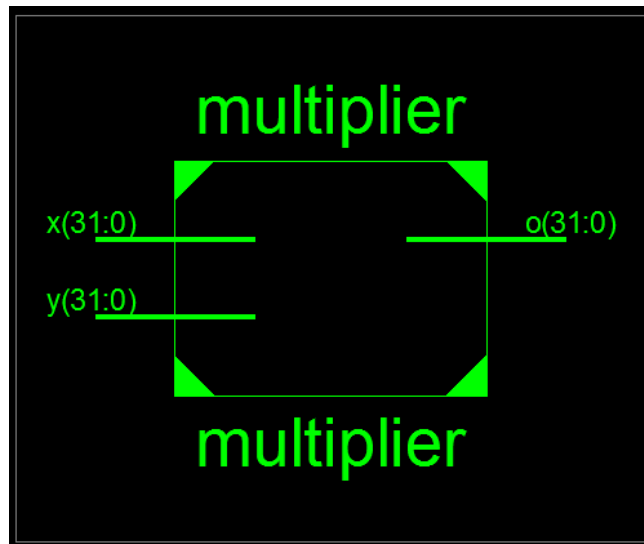
This is a 9-input 6-output hardware module that the inputs are connected to 32-bit registers which holds distance and position values. Similarly six outputs are connected to 32-bit registers to initialize two dimensional array. coef\_calculator block is mainly composed of three components which are 32-bit adder, subtractor and multiplier modules. These subblocks are also shown in figure 5.8, figure 5.9 and figure 5.10 respectively.



**Figure 5.8:** adder module.

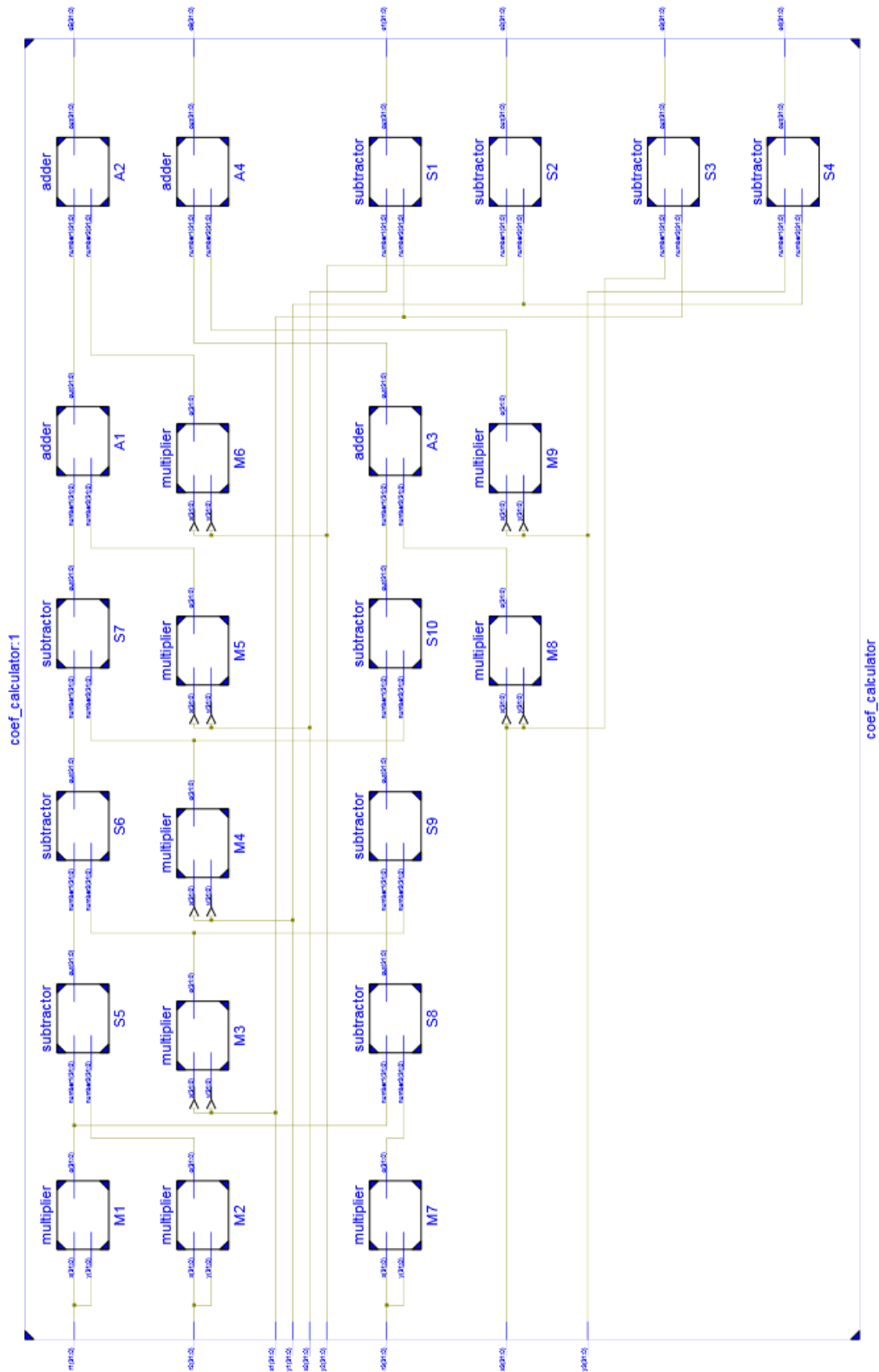


**Figure 5.9:** subtractor module.



**Figure 5.10:** multiplier module.

All three blocks have two 32-bit input and one 32-bit output. Also whole hardware system is designed as a combinational circuit and internal structure of coef\_calculator module is shown figure 5.11.



**Figure 5.11:** Internal structure of `coef_calculator` module.

### 5.3 Visualization of localization Results

After the position of the reader is determined, the results are sent to computer through UART and they are printed out as floating numbers. However, a graphical representation is always more appealing and easier to understand for the users. Therefore, a simple graphical interface program has been written through C# in Visual Studio. As explained in the previous section, there are small errors when some parts of the algorithm is designed as hardware. Localization algorithm and graphical interface program are tested with some virtual numbers both on pure software design and hardware-software design with several virtual distance inputs. It is presumed that, a reader is moving in an indoor area with lengths 10 meters in y axis and 26 meters in x axis. Also (x, y) coordinate pairs of three tags are considered as (2, 1), (16, 1), (9, 10). Some certain positions of the reader, the distance values between the reader and the tags and localization algorithm outputs for each design are shown below.

1) The reader position:  $(x, y) = (5, 5)$

The distance values:  $r_1 = 5, r_2 = 11.7047, r_3 = 6.40312$

Software based design outputs:  $(x, y) = (4.99999, 5.00002)$

Hardware-software based design outputs:  $(x, y) = (5.21475, 5.02941)$

2) The reader position:  $(x, y) = (6, 3)$

The distance values:  $r_1 = 4.47214, r_2 = 10.19804, r_3 = 7.61577$

Software based design outputs:  $(x, y) = (6.00000, 3.00000)$

Hardware-software based design outputs:  $(x, y) = (5.21475, 3.21475)$

3) The reader position:  $(x, y) = (8, 3)$

The distance values:  $r_1 = 6.32456, r_2 = 8.24621, r_3 = 7.07107$

Software based design outputs:  $(x, y) = (8.00000, 2.99999)$

Hardware-software based design outputs:  $(x, y) = (7.21475, 2.21475)$

4) The reader position:  $(x, y) = (8, 5)$

The distance values:  $r_1 = 7.2111, r_2 = 8.94427, r_3 = 5.09902$

Software based design outputs:  $(x, y) = (8.00000, 5.00000)$

Hardware-software based design outputs:  $(x, y) = (8.21475, 4.21475)$

**5)** The reader position:  $(x, y) = (8, 7)$

The distance values:  $r_1 = 8.48528, r_2 = 10, r_3 = 3.16228$

Software based design outputs:  $(x, y) = (7.99999, 6.99999)$

Hardware-software based design outputs:  $(x, y) = (7.21475, 7.14706)$

**6)** The reader position:  $(x, y) = (10, 8)$

The distance values:  $r_1 = 10.63014, r_2 = 9.21954, r_3 = 2.23607$

Software based design outputs:  $(x, y) = (10.00001, 8.00000)$

Hardware-software based design outputs:  $(x, y) = (9.21475, 7.21475)$

**7)** The reader position:  $(x, y) = (13, 6)$

The distance values:  $r_1 = 23.08305, r_2 = 5.83095, r_3 = 5.65685$

Software based design outputs:  $(x, y) = (13.00000, 6.00005)$

Hardware-software based design outputs:  $(x, y) = (12.21475, 6.21475)$

**8)** The reader position:  $(x, y) = (14, 4)$

The distance values:  $r_1 = 12.36932, r_2 = 3.60555, r_3 = 7.81025$

Software based design outputs:  $(x, y) = (14.00000, 4.00000)$

Hardware-software based design outputs:  $(x, y) = (13.21475, 4.21475)$

**9)** The reader position:  $(x, y) = (16, 3)$

The distance values:  $r_1 = 14.14214, r_2 = 2, r_3 = 9.89949$

Software based design outputs:  $(x, y) = (16.00000, 3.00001)$

Hardware-software based design outputs:  $(x, y) = (15.21475, 3.21475)$

**10)** The reader position:  $(x, y) = (18, 3)$

The distance values:  $r_1 = 16.12452, r_2 = 2.82843, r_3 = 11.40175$

Software based design outputs:  $(x, y) = (18.00000, 3.00001)$

Hardware-software based design outputs:  $(x, y) = (17.21475, 3.21475)$

**11)** The reader position:  $(x, y) = (18, 5)$

The distance values:  $r_1 = 16.49242, r_2 = 4.47214, r_3 = 10.29563$



Software based design outputs:  $(x, y) = (18.00000, 4.99999)$

Hardware-software based design outputs:  $(x, y) = (17.21475, 4.21475)$

**12)** The reader position:  $(x, y) = (18, 7)$

The distance values:  $r_1 = 17.08801, r_2 = 6.32456, r_3 = 9.48683$

Software based design outputs:  $(x, y) = (18.00000, 7.00001)$

Hardware-software based design outputs:  $(x, y) = (17.21475, 7.21475)$

**13)** The reader position:  $(x, y) = (20, 7)$

The distance values:  $r_1 = 18.97367, r_2 = 7.2111, r_3 = 11.40175$

Software based design outputs:  $(x, y) = (20.00001, 7.00001)$

Hardware-software based design outputs:  $(x, y) = (18.21475, 6.21475)$

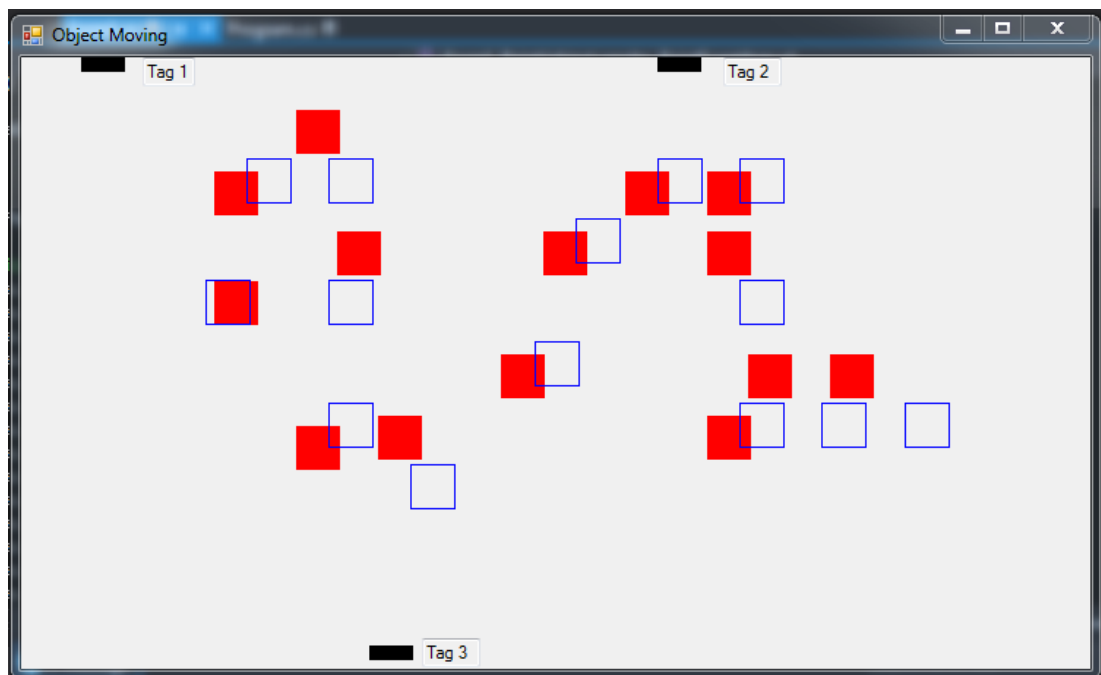
**14)** The reader position:  $(x, y) = (22, 7)$

The distance values:  $r_1 = 20.88061, r_2 = 8.48528, r_3 = 13.34166$

Software based design outputs:  $(x, y) = (22.00000, 7.00000)$

Hardware-software based design outputs:  $(x, y) = (20.21475, 6.21475)$

The graphical representation of these results through c# is shown in figure 5.12.



**Figure 5.12:** Visualized localization results.

In figure 5.12, blue empty squares indicate the several positions of the reader in motion. Purely software designs which are implemented both computer and Microblaze through C# are able to obtain exact location of the reader. Therefore software based results also correspond to blue empty squares. On the other hand, the red squares in the figure indicate the localization results obtained from hardware-software design. As seen there are some differences in the results, but these errors are less than 2 meters and mostly vary between 0.25 and 1.2 meters. This errors might be very critical or not depending on aim of localization system usage. This example operation is considered as in a building with the lengths 10 meters in y axis and 26 meters in x axis which means this error deviation does not make a big deal. Also to figure results in a window, pixel logic is used. The is window has 731 X 418 pixels and related scale operations are performed to correspond the position values to the window.

## 6. IMPLEMENTATION OF AUTHENTICATION PROTOCOL

As mentioned in section 4, an entity authentication protocol based on encryption scheme has been proposed for this project. The protocol has been only implemented and tested on computer through C++. If a localization system such as this study is wanted to be used in daily life, the authentication protocol needs to be implemented on Microblaze and the processors which drives the tags. Implementation process of the protocol is carried out according to figure 4.2 which explains the whole system. In this protocol, public key model is presumed. Also it is assumed that the reader is informed about the ID and the public key which are shared by the tag.

First, the reader generates a 32-bit random number  $r$  and  $\text{rand}()$  functions have been used for this operation. There have been three  $\text{rand}()$  used, because this function generates random numbers between 0 and  $\text{RAND\_MAX}$  which is 32767. This number corresponds to 15-bit number and to obtain 32-bit random number 3  $\text{rand}()$  functions are multiplied. Also  $\text{rand}()$  is not a perfect random number generator, but it meets our requirements in this study. After the random number is generated is inserted into a hash function and related digest ( $e$ ) is calculated. To perform this operation, template hash function of C++ has been used. Then encryption operation is performed as explained in Section 6.1.

### 6.1 Realization of Scalable Encryption Algorithm

In this study, 32-bit random number ( $r$ ), and 8-bit tag ID are concatenated and encrypted by using Scalable Encryption Algorithm (SEA). Necessary parameter lengths are decided as below.

- $n$  (plaintext size / key size) = 48-bit
- $b$  (processor (word) size) = 8-bit
- $n_r$  (number of rounds) = 36

There are two main constraints on deciding these parameters. First,  $n$  should be a multiple of  $6*b$  that for a 8-bit processor the designer could use 48, 96, 144... and 48

bit is sufficient for this study. Other restriction is about number of rounds which should be at least  $3 \cdot (n/4)$  to be resistant against linear and differential cryptanalysis and 36 is considered enough. The pseudo code of SEA is shown below [23].

```

C=SEAn; b(P; K)
{
%Initialization
L0 & R0 = P;
KL0 & KR0 = K;
%Keyscheduling
For i = 1 to  $n_r/2$ 
    [KLi; KRi] = FK(KLi-1, KRi-1, C(i));
    Switch KL[ $n_r/2$ ], KR[ $n_r/2$ ];
For i =  $n_r/2+1$  to  $n_r/2-1$ 
    [KLi; KRi] = FK(KLi-1, KRi-1, C(i-1));
%Encryption
For i = 1 to  $n_r/2$ 
    [KLi; KRi] = FE(Li-1, Ri-1, KR(i-1));
For i =  $n_r/2+1$  to  $n_r/2$ 
    [KLi; KRi] = FK(Li-1, Ri-1, KL(i-1));
%Final
C = R $n_r$  & L $n_r$ ;

```

Where & is the concatenation operator,  $KR_{[n/2]}$  is taken before the switch and  $C(i)$  is a  $nb$ -word vector of which all the words have value 0 excepted the least significant word that equals to  $i$ . Also decryption operation is exactly same which uses  $FD$  instead of  $FE$  [23].

The only difference between encryption and decryption is the order of key schedule that it is reversed order for decryption operation according to encryption operation. Also the master key or input of key generation process is the public key of the tag.

First, key schedule is generated and then encryption and decryption processes are performed.

Encryption operation is performed in the reader through SEA. The input of SEA is  $0^8||r||ID$  which means that first 8-bit indicates ID of the tag, following 32-bit indicates random number (r) generated by reader and last 8-bit are filled with 0 to obtained 48 bit sequence. After these values are encrypted and the output of SEA (c) is obtained, the reader sends ciphertext (c), the tag ID, and digest (e) to the tag.

When the tag receives these information, it decrypts the ciphertext c and obtains values of random number (r) and ID. ID is checked with a single compare operation and r is checked by using has function again. To prevent a misunderstanding, IDt means obtained ID value and y means obtained r value after decryption. Therefore, IDt value needs to be equal to ID and digest of y (h(y)) needs to be e. If these requirements are met, the tag ensure the authorization of the reader. Finally, the tag sends y value to the reader and if y is equal to random number (r), the reader ensures the authorization of the tag. The key schedule for encryption operation is shown in figure 6.1 and for decryption the reversed order is used.

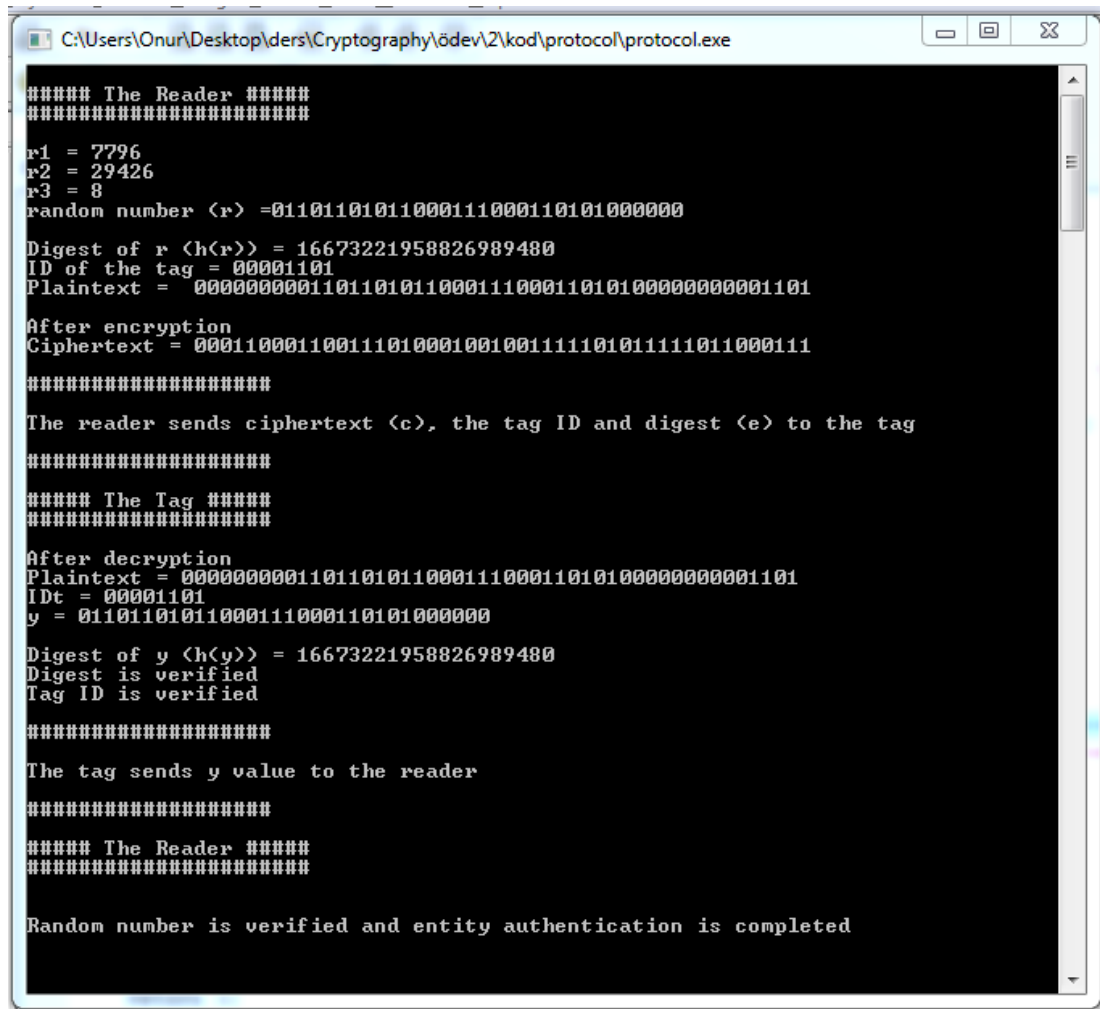
```

C:\Users\Onur\Desktop\ders\Cryptography\ödev\2\kod\protocol\protocol.exe
K[0] = 110001000110111001001000110001000110111001001000
K[1] = 110001000110111001001000100100001011101001010001
K[2] = 100100001011101001010001000100000001101111001000
K[3] = 000100000001101110010000000110101010111100100
K[4] = 00000110101011011100100110000110100010000001111
K[5] = 110000110100010000001111000111101010001101110011
K[6] = 0001111010100011011100011000100000010101011100110
K[7] = 000100000010101011100110111010101111011010110010
K[8] = 111010101111011010110010001010110100001001001000
K[9] = 001010110100001001001000001110000111011000110110
K[10] = 0011100001101100011011010100101101111001010110
K[11] = 10100101101111001010110100011100100101111000110
K[12] = 100011100100101111000110001001100010010111011000
K[13] = 001001100010010111011000001010000100000000100111
K[14] = 001010000100000000100111101000101111010110001100
K[15] = 10100010111101011000110010000011110111001100011
K[16] = 100000111110111001100011011110000010110001010000
K[17] = 011110000010110001010000101010110110011010110011
K[18] = 101010110110011010110011111000111110000101010000
K[19] = 1110001111000010101000010101011110001111110101
K[20] = 1010101111000111110101011110100010010111011010
K[21] = 0111101000100101110110100000010111110000101000
K[22] = 10000001011111000010100010000010110111110100001
K[23] = 10000010110111110100001011110101100001001000010
K[24] = 01111010110000100100001011011001011111110001110
K[25] = 1101100101111111000111000110111101011111011100
K[26] = 001101110101111101110011010000110010001001011
K[27] = 01101000011001000100101100001101001001111011110
K[28] = 00001101001001111011110001111000010101100001010
K[29] = 001111000010101100001010000010110000100111000010
K[30] = 000010110000100111000010100101000011110010100111
K[31] = 100101000011110010100111010110000111100010011000
K[32] = 01011000011110001001100010110100111110100011000
K[33] = 11011010011111010001100000010110000001001011001
K[34] = 000101100000001001011001000000100101110111101100

```

Figure 6.1: Key Schedule.

The implementation of whole entity authentication protocol is also shown in figure 6.2.



```
C:\Users\Onur\Desktop\ders\Cryptography\odev\2\kod\protocol\protocol.exe

##### The Reader #####
#####

r1 = 7796
r2 = 29426
r3 = 8
random number <r> =01101101011000111000110101000000

Digest of r <h(r)> = 16673221958826989480
ID of the tag = 00001101
Plaintext = 00000000011011010110001110001101010000000001101

After encryption
Ciphertext = 000110001100111010001001001111101011111011000111

#####

The reader sends ciphertext <c>, the tag ID and digest <e> to the tag
#####

##### The Tag #####
#####

After decryption
Plaintext = 00000000011011010110001110001101010000000001101
IDt = 00001101
y = 01101101011000111000110101000000

Digest of y <h(y)> = 16673221958826989480
Digest is verified
Tag ID is verified

#####

The tag sends y value to the reader

#####

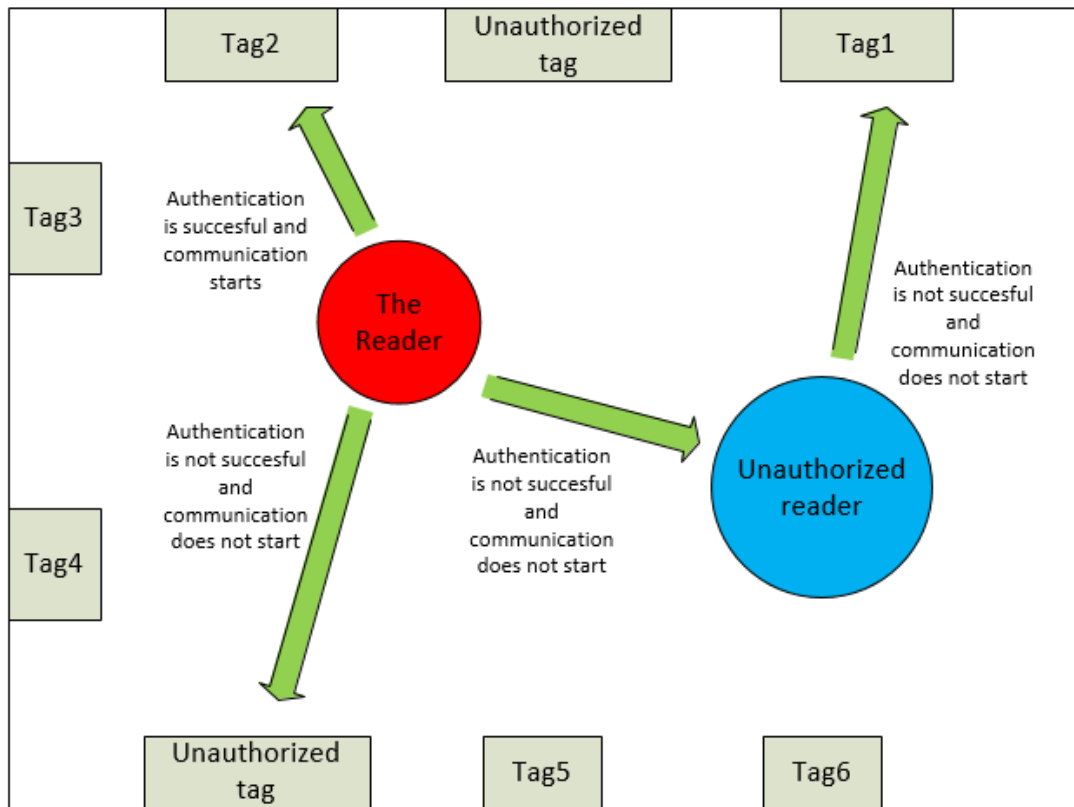
##### The Reader #####
#####

Random number is verified and entity authentication is completed
```

**Figure 6.2:** Implementation of authentication protocol.

As seen the implementation is performed on C++ properly and entity authentication is completed.

This entity protocol does not assure of secure communication between the reader and the tag, but prevents both reader and tag to communicate with an unauthorized device. Therefore, a possible information leak is blocked considerably. Possible scenarios among the reader, the tags and unauthorized devices in a building are explained in figure 6.3.



**Figure 6.3:** The role of entity authentication.

After entity authentication is provided, communication between the reader and the tag is able to start and the localization operation is performed.

## 7. CONCLUSION

In this graduation project, a RFID based localization system for indoor applications is aimed to be designed and implemented on FPGA. After a literature research, trilateration algorithm was selected for this study. High accuracy in positioning performance is the main reason selection of this algorithm. Also compatibility with both two dimensional and three dimensional applications is another significant feature. On the other hand, requirement of considerable sources during the implementation on FPGA device is the main drawback of this method.

In this project, 2D trilateration algorithm is considered sufficient for indoor localization. First, the algorithm was implemented on general purpose computer through C++. Visual Studio was used to design and test the localization algorithm. Tests were performed with virtual distance inputs and positioning results were obtained properly. After ensuring that the algorithm operated properly, localization algorithm was realized on FPGA device through Microblaze. After Microblaze processor was implemented on FPGA device, the algorithm was realized on this processor as pure software. In other words, Microblaze was implemented as hardware on FPGA and the localization algorithm was implemented on this virtual processor. The design was tested in Xilinx SDK tool using virtual distance inputs and system operated properly.

The algorithm is aimed to be designed as completely hardware and operates in conjunction with Microblaze processor. However this system can not be implemented due to some problems of data transmission between Microblaze processor and user-defined hardware module. For transmission data between Microblaze and user-defined hardware module, there are some write and read functions that are supplied by Xilinx. These functions are defined to operate with integer values and do not allow user to work with floating numbers. Performing mathematical operations with floating numbers is necessary to obtain exact position information of the reader. Due to this problem, floating numbers could not be sent to hardware module, because only exponential part of floating numbers are able to be transmitted. Although some



modifications were made in library functions to transmit floating numbers to hardware module implemented systems did not operate properly and generated false positioning results. Therefore only first of the algorithm which calculates necessary coefficients was designed as hardware and only integer values are inserted into this module. After hardware module finishes its operation, sends the related coefficients to Microblaze and localization is completed with C++. This situation cause some deviations in localization results but the position of the reader can still be obtained very similar to real one. Finally, results were printed on a screen by designing a simple graphical interface program through C# and performance of each design observed.

Security of the system is also important as performance and if such a project in wanted to be utilized, some issues need to be considered. There are security gaps in this system because of wireless transmission operations. For example, if the readers and the tags sends their information without checking the authorization of the receiver, an unauthorized device or attacker can abuse the system. Therefore the tags and the reader should ensure of authorizations of each other. For this reason, a entity authentication protocol was proposed and implemented as software. This protocol provides authorization of both the reader and the tags. The system was implemented and tested on general purpose computer through C++.

If the floating number transmission problem can be resolved, the whole trilateration algorithm can easily implemented on hardware and operate with Microblaze which means exact position information of the reader can be obtained with hardware module.

## REFERENCES

- [1] **Loeffler A., Gerhaeuser H.**, 2011. Localizing with Passive UHF RFID Tags Using Wideband Signals, *Microwaves, Communications, Antennas and Electronic Systems (COMCAS)*, pp.1-6.
- [2] **Sanpechua T., Kovavisaruch L.**, 2008. A Review of RFID Localization: Applications and Techniques, *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology Conference (ECTI-CON)*, vol.2, pp.769-772.
- [3] **Aktaş R.**, Distance Estimation Using Received Signal Strength for RFID Tracking System, *BSc Thesis*, I.T.U. Faculty of Electrics and Electronics, Istanbul.
- [4] **Çık O.**, Bir RFID Takip Etme Sistemi için Okuyucuları Sürececek Bir Alt Sistemin FPGA Üzerinde Gerçeklenmesi, *BSc Thesis*, I.T.U. Faculty of Electrics and Electronics, Istanbul.
- [5] **Oran A.**, Implementation of A RFID Based Indoor Localization System on FPGA, *BSc Thesis*, I.T.U. Faculty of Electrics and Electronics, Istanbul.
- [6] **Bahadır G.**, Design and Implementation of A Secure Bluetooth Low Energy Communication, I.T.U. Faculty of Electrics and Electronics, Istanbul.
- [7] **Baas**, Building as a Service, <http://baas-itea2.eu/cms/project-menu-item>.
- [8] **L. Sandip**, RFID Sourcebook, IBM Press, USA, 2005 ISBN: 0-13-185137-3.
- [9] **Landt, J.**, 2005. The history of RFID, *Potentials, IEEE*, vol.24, no.4, pp.8-11.
- [10] **Halder S. J., Park J., Kim W.**, 2011, Adaptive Filtering for Indoor Localization using ZIGBEE RSSI and LQI Measurement, ISBN: 978-953-307-306-4.
- [11] **HopeRF Electronics**, RFM22B/23B ISM Transceiver Module, V1.1.
- [12] **Texas Instruments**, 2010, Antenna Selection Guide, Application Note 058.
- [13] **Arduino**, <http://www.arduino.cc/en/Main/ArduinoBoardUno>.
- [14] **Chu, Pong P.**, 2008, FPGA Prototyping by VHDL Examples. Wiley-Interscience, New Jersey.
- [15] **Xilinx**, Spartan-6 Family Overview.
- [16] **Digilent**, Nexys3™ Board Reference Manual.
- [17] **Xilinx**, 2008, MicroBlaze Processor Reference Guide.
- [18] **Xilinx**, 2011, Embedded System Tools Reference Manual.
- [19] **Xilinx**, EDK Concepts, Tools and Techniques.
- [20] **Oguejiofor O.S., Aniedu A.N., Ejiofor H.C., Okolibe A.U.**, 2013, Trilateration Based Localization Algorithm for Wireless Sensor Network, *International Journal of Science and Modern Engineering (IJISME)*, ISSN: 2319-6386, vol.1.

- [21] **Singh A. P.**, 2013, Hiding a Mobile Phone to Prevent Vehicle Theft, <https://hawktrack.wordpress.com/2013/07/26>.
- [22] **Hutter M.**, 2009, RFID Authentication Protocols Based on Elliptic Curves: A top-down evaluation survey, International Conference on Security and Cryptography (SECRYPT09), pp.101-110.
- [23] **Standaert F. X., Piret G., Gershenfeld N., Quisquater J. J.**, 2006, SEA a Scalable Encryption Algorithm for Small Embedded Applications, Proc. CARDIS, pp.222-236.

## **RESUME**

**Name Surname:** Onur Azbar

**Birth Place and Date:** Adana, 1992

**Highschool:** Bornova Anatolian Highschool, 2006-2010

**BSc:** Istanbul Technical University, Electronics and Communication Engineering, 2010-2015