  
  
**T.C. MALTEPE UNIVERSITY**

**Department of Computer Engineering**

**Graduation Project II:**

**Communication-assisted Device for Patients and Disabled People**

**PREPARED BY:**

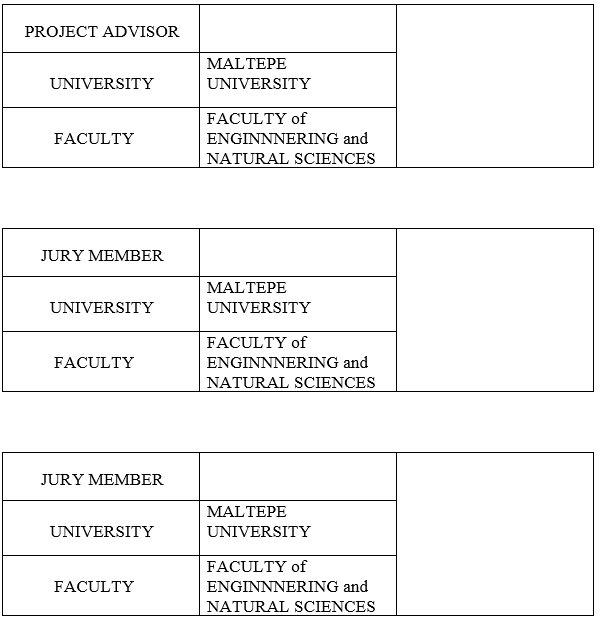
**Ertuğrul BAYRAKTAR**

**ADVISOR: Assist. Prof. Dr. Ali AKMAN**

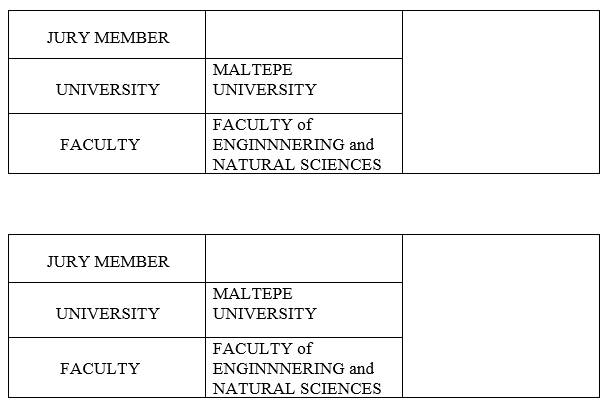
i

This project was accepted as a Graduation Project in the Department of Computer Engineering by the following jury on …/…/20..

Graduation Project Jury



ii



iii

**Contents**

1. INTRODUCTION.…………………………………………………………………………1  
 1.1. Definition of the Project.…………………………...………………………………..1  
 1.2. The Goal of the Project.……………………………………………………………..1  
 1.3. Original Value of the Project ……………………………………………………….2  
 1.4. Subject and Scope of the Project……...…………….……………………………….3  
 *1.4.1. Subject*………………………………………………………………..….3  
 *1.4.2. Arduino Uno*…...…………………………………….………….……….4  
 *1.4.3. IO Expansion Shield*…………………………………………………......5  
 *1.4.4. XBee*……………………………………………………………………..6  
 1.5. Industry / Literature Research ……………………………………………..………..8  
 *1.5.1. Home Automation Projectusing XBee & Arduino*…………………….8  
 *1.5.2.* *Wireless Sensor Network Architect*……………………………….........10  
 *1.5.3. Where to Use Xbee Modules ?*................................................................14  
 1.6. Risk Assessment...…….… ……………………………………………………..…20  
 1.7. Work Plan…..……………………………...………………………………...….....21  
 1.8. Task Management…..………………………………………………………………25

2. PROJECT SCOPE…………………………………………………………………………26  
 2.1. System Architecture……………………………………………………………..…26  
 *2.1.1. System Overview*…………………………………………………….....26  
 2.2. System Development Plan…………………………………………………………27  
 *2.2.1. XCTU*…………………………………………………………………..27  
 *2.2.2. Configuration of Arduino Uno and XBee Modüle*………………..……28  
 *2.2.3. IO Expansion Shield*…………………………………………...……….29  
 *2.2.4. How to use with XBee modüle with IO Expansion Shield*……………...30  
 *2.2.5. Exploring XBee with PC*……………………………………….………32  
 *2.2.6. Range Tes*t……………………………………………………….……..34  
 *2.2.7. Arduino Button Use*…………………………………………………….37  
 2.3. Project Platform…………………………………………………………...……….43  
 *2.3.1. Operating Systems*………………………………………………………44  
 2.4. Work Sharing………………………………………………………………………47  
 2.5. Test Plan…………………………………………………………………….……..48  
  
3. USERGUIDE FOR NURSE…..…………………………………………………………....50  
 3.1. User Guide for Nurse……………………………………………………………….50  
 3.2. User Guide for Patients……………………………………………………………..55

4. REFERENCES…………………………………………………………………………….56  
 4.1. Figures……………………………………………….………………….…………56  
  
 iv

**1. INTRODUCTION**

**1.1. Definition of the project**

Communication-assisted electronic equipment for patients with speech difficulties, elderly and disabled people who cannot express their wishes.

**1.2. The goal of the project**

It is a device developed especially for people with autism and the elderly. This device is planned to be produced in a wide range of areas. It is a device that will be useful for individuals with autism who have speech difficulties in educational institutions focusing on child development through this plan. Similarly, it will be a source of hope for the elderly people who do not say their needs in their homes. It is also a device to help parents, nurses and educators in educational institutions.

For this purpose, the electronic materials to be used in order to make the device more comfortable for everyone should be budget-friendly. So it was named Arduino Uno as physical programming platform. The computer system was considered as a control mechanism. This tool is used as a principle to be extremely useful.

Considering all these, the necessary studies have been initiated for the realization of the project.

1

**1.3. Original Value of the Project**

Numerous projects related to Arduino Uno are at a time when technological inventions have become extremely important. Many articles are written and followed worldwide. It occupies an important place in terms of both adaptation and forward light to the needs of the technology age.

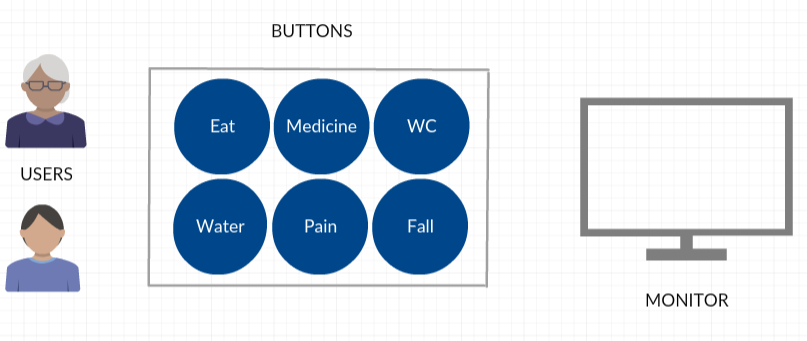
This project, which we have done, is aimed to provide benefits to society and people both in today's conditions by using today's innovations and to be a draft for future products.

Our most important goal is to transfer the needs of the person more easily. Accordingly, it provides low-cost supply to health institutions, nursing homes and institutions supporting child development.

Our project will also be very clear to develop and add new features. Maximum performance and durability are indispensable.

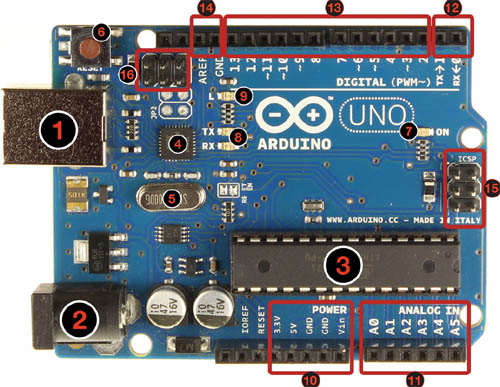
2

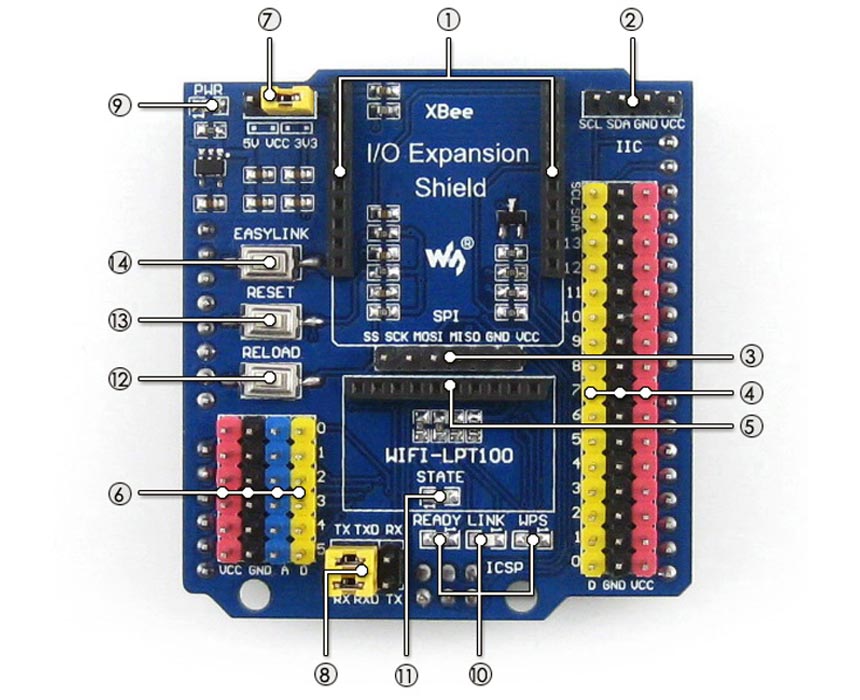
**1.4. Subject and Scope of the Project  
  
1.4.1. Subject**In this project, it was aimed to design an electronic device to help the patients with speech difficulties, disabled people and elderly people who could not express their wishes. 6 different buttons will be created to determine the user's request on the device to be designed. These buttons: eat, water, medicine, pain, wc, fall will be in the form of needs. When the user presses one of these buttons, the arduino will be processed, the information will go to the computer with the communication of the XBee and the interface will appear as output on the screen.

  
 Figure 1.4.1. Buttons  
  
**1.4.2. Arduino Uno**The Arduino Uno is an Arduino card that includes the ATmega328 microcontroller. It can be said that Arduino is the most widely used card. After the first model of Arduino Uno, Arduino Uno R2, Arduino Uno SMD and lastly Arduino Uno R3 came out..

Arduino Uno has 14 digital input / output pins. 6 of them can be used as PWM output. There are also 6 analog inputs, one 16 MHz crystal oscillator, USB connection, power jack (2.1mm), ICSP header and reset button. The Arduino Uno contains all of the components required to support a microcontroller. By connecting the Arduino Uno to a computer, you can operate it with an adapter or battery. The following illustration shows the parts of the Arduino Uno R3.

3

  
 Figure 1.4.2. Arduino Uno  
   
1 : USB jack  
2 : Power jack (7-12 V DC)  
3 : Micro controller ATmega328  
4 : Communication chip  
5 : 16MHz crystal  
6 : Reset button  
7 : Power led  
8 : TX / NX LEDs  
9 : Led  
10 : Power pins   
11 : Analog inputs  
12 : TX / RX pins  
13 : Digital input / output pins (can be used as a PWM output with pins with a ~ mark).  
14 : Ground and AREF pins  
15 : ICSP for ATmega328   
16 : ICSP for USB interface   
 4

**1.4.3. IO Expansion Shield**  
  
   
 Figure 1.4.3. IO Expansion Shield   
  
1. XBee module connector  
2. IIC interface  
3. SPI interface  
4. 3-pin sensor interface  
 VCC : power positive  
 GND : ground  
 D : digital pin, correspond to the Arduino board  
5. WIFI-LPT100 connector  
6. 4-pin sensor interface  
 VCC : power positive  
 GND : ground  
 A : analog pin, correspond to the Arduino board  
 D : digital pin, correspond to the Arduino board  
7. VCC configuration : for selecting sensor interfaces power voltage  
8. Debugging/Communication selection jumper  
 when connecting TXD and TX, RXD and RX respectively, the Arduino board maydebug/config the XBee module or WIFI-LPT100 through serial port  
 when connecting TXD and RX, RXD and TX respectively, the Arduino board may communicate with the XBee module or WIFI-LPT100 through serial port  
9. Power indicator  
10. WIFI-LPT100 state indicator  
11. XBee module state indicator  
12. WIFI-LPT100 RELOAD button : for restoring to factory setting  
13. XBee module and WIFI-LPT100 RESET button  
14. XBee module EASYLINK button  
 5

**1.4.4. XBee (Wireless Kit)**

Xbee module is a 2.4 GHz wireless communication module developed by Digi. Xbee modules are classified according to the protocols they use. Series 1 modules Xbee, series 2 modules are called Zigbee. Let's explain them in order.

**1.4.4.1. What is XBee?**

It uses the IEEE 802.15.4 network protocol. It supports point-to-point or point-to-point communication. For point-to-multi point data transfer, the serial 1 module is ideal.  
  
 

Figure 1.4.4. XBee S2

**1.4.4.2. What is Zigbee?**

Developed in accordance with IEEE 802.15.4 and ZigBee Mesh communication protocol. The most important feature according to IEEE 802.15.4 standard is low power consumption, long range data transmission and network support. Zigbee takes its name from the complex motion structures of the bees. It can read analog and digital values via analog, digital pins on the Xbee modules. UART can communicate with other devices via serial communication pins. It supports communication within point-to-point or multi-point networks such as Point to Point and Point to Multi-Point. The Zigbee module is shown in the picture below.  
  
   
  
  
  
  
  
 6

**1.4.4.3. ZIGBEE ARCHİTECTURE**

Zigbee system structure consists of three different types of devices such as Zigbee coordinator, Router and End device. Every Zigbee network must consist of at least one coordinator which acts as a root and bridge of the network. The coordinator is responsible for handling and storing the information while performing receiving and transmitting data operations. Zigbee routers act as intermediary devices that permit data to pass to and fro through them to other devices. End devices have limited functionality to communicate with the parent nodes such that the battery power is saved as shown in the figure. The number of routers, coordinators and end devices depends on the type of network such as star, tree and mesh networks.

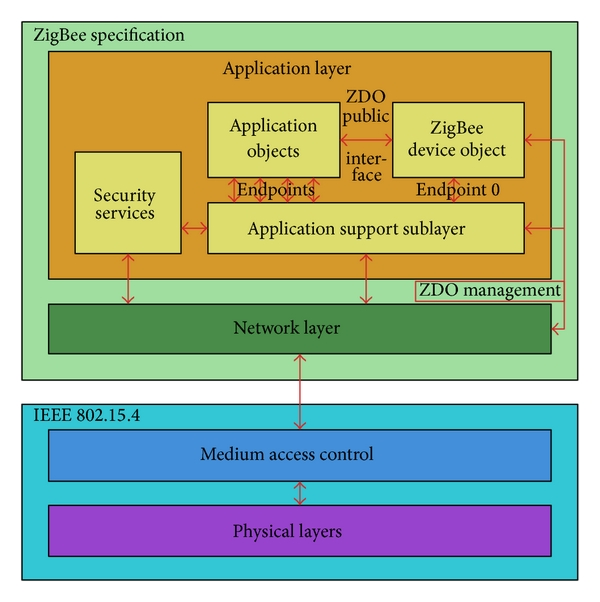


Figure 1.4.5. Zigbee protocol architecture

7

**1.5. Industry / Literature Research**

**1.5.1. Home Automation Project using XBee & Arduino**

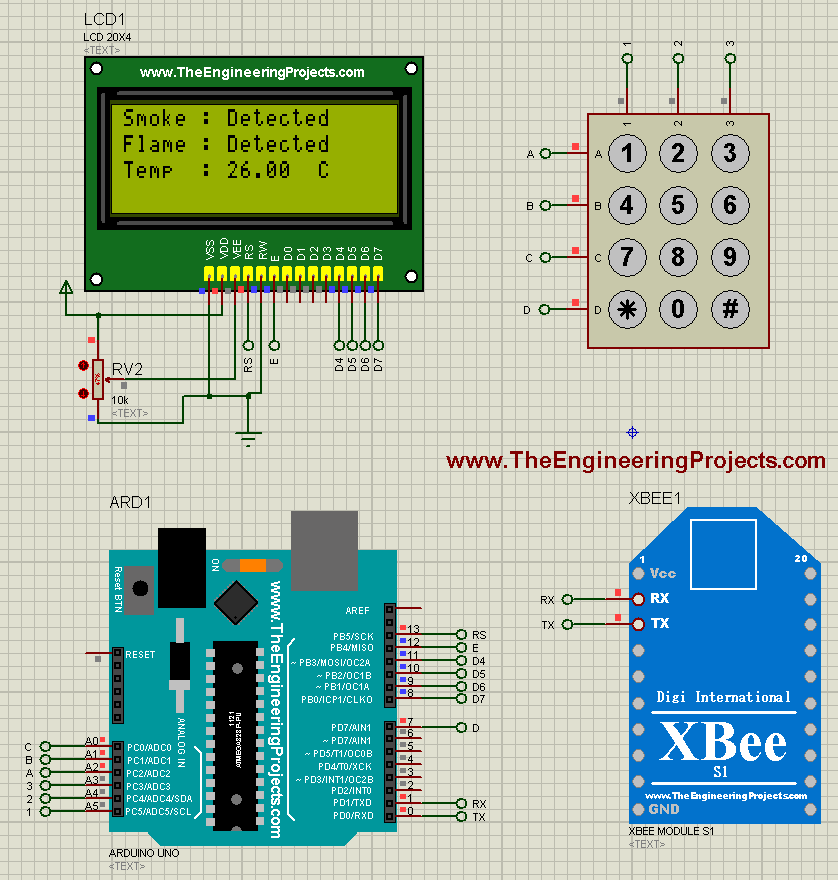


Figure 1.5.1. Home Automation Project

•First of all, let’s have an overview of this Home Automation Project.  
•In this Project, I have designed two simulations, one simulation is for Remote using which we are gonna control our appliances and the second simulation is for the controlling of these appliances.  
•So, when you press buttons from your remote section, a wireless command will be sent to the control board and it will turn ON or OFF the respective load  
•Moreover, there’s an LCD on the Remote on which you will also check the values of the sensors.

8

•So, in simple words, the remote will be in your hand and using this remote you can easily turn ON or OFF your appliances and can also check the status of your different sensors wirelessly.

**Remote Control:**  
  
•In Remote Control Section, I have used the below main modules:  
 Arduino UNO: Microcontroller Board.  
 KeyPad: Commands will be sent by clicking this Keypad’s buttons.  
 LCD (20 x 4): For Displaying Sensor’s Data & Commands.  
 XBee Module: It’s an RF Module used for sending wireless commands.  
 Now when you click any button on your Keypad, a command is sent from •Arduino to XBee Module and the XBee module then forwards that command to other XBee on the Control Unit.  
•Moreover, when the Control Unit sends the Sensors’ data on xbee then Arduino receives that data and then displayed that data on LCD.  
•Here’s the block diagram of Remote control section which will give you a better idea of its working:



Figure 1.5.2. Working Shematic

9

**1.5.2. Wireless Sensor Network Architect**

**1.5.2.1. Bus Topology**

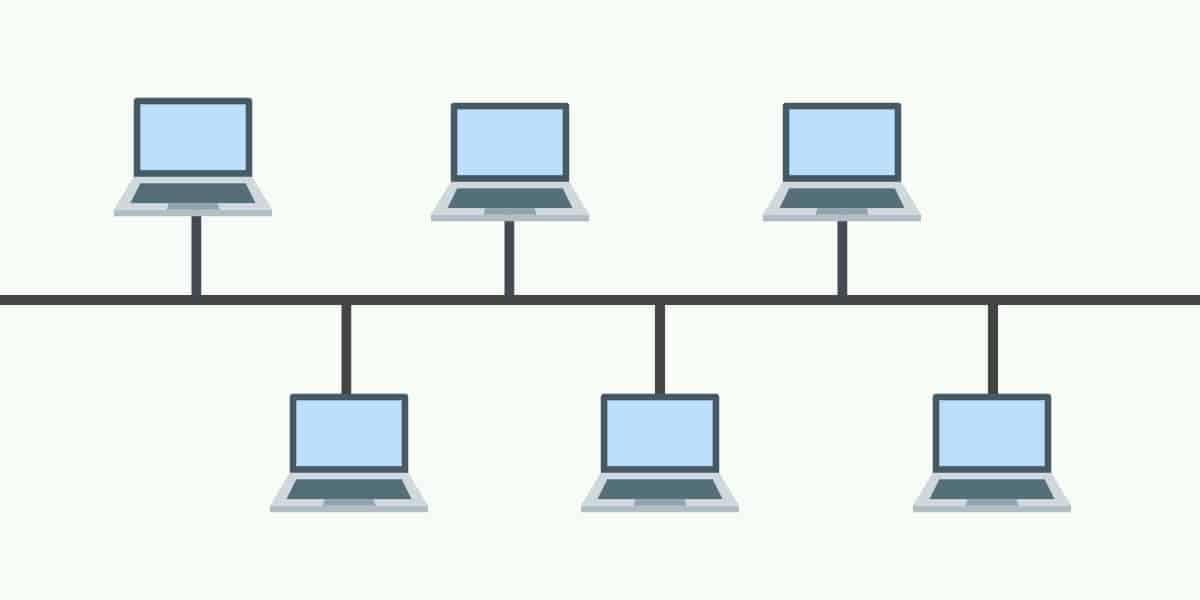


Figure 1.5.3. Bus Topology

Bus topology is a type of network where every device is connected to a single cable which runs from one end of the network to the other. This type of type of topology is often referred to as line topology. In a bus topology, data is transmitted in one direction only. If the bus topology has two endpoints then it is referred to as a linear bus topology. Organizations using this type of topology will generally use an RJ45 cable to link devices together.

**1.5.2.2.** **Ring Topology**

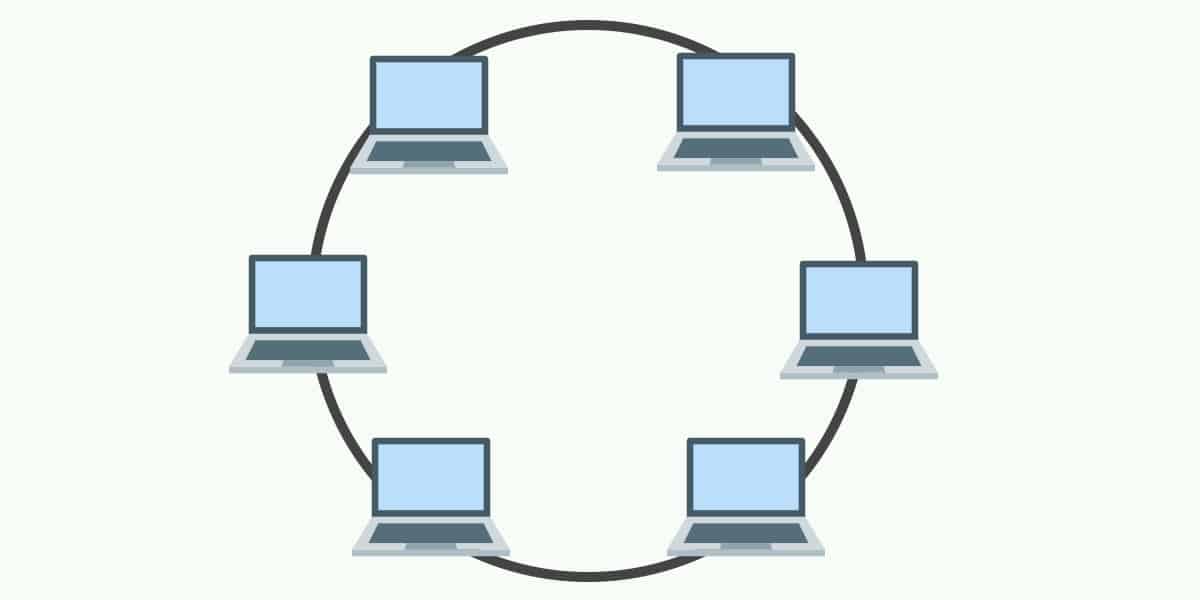


Figure 1.5.4. Ring Topology

10

In networks with a ring topology, computers are connected to each other in a circular format. Every device in the network will have two neighbors and no more or no less. The first node is connected to the last node to link the loop together. As a consequence of being laid out in this format packets need to travel through all nodes on the way to their destination.

**1.5.2.3. Star Topology**

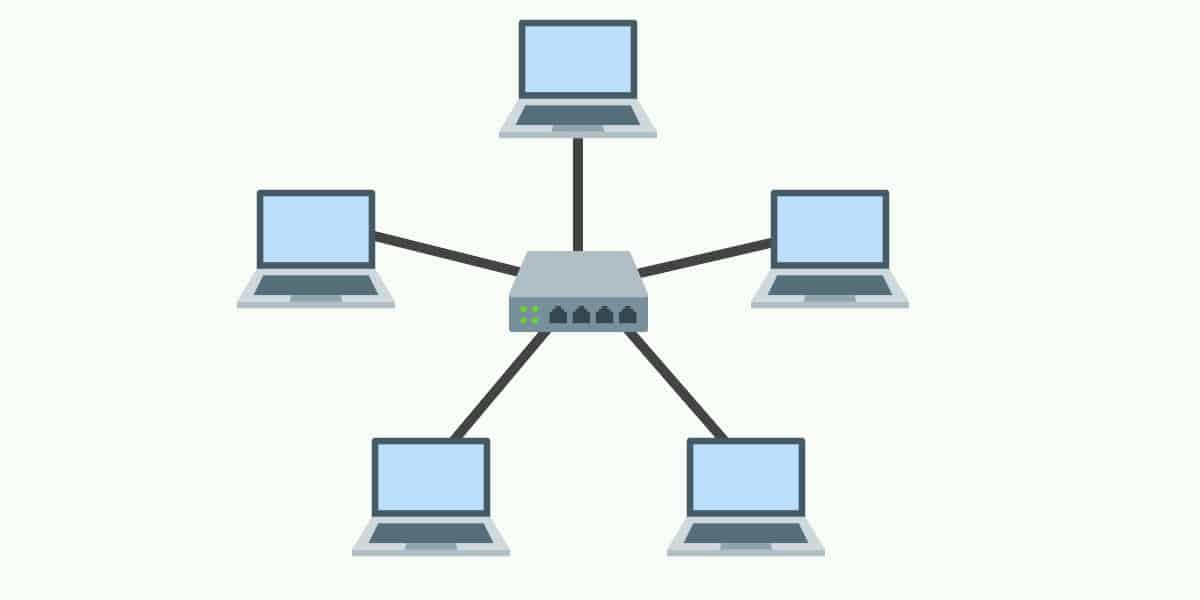


Figure 1.5.5. Star Topology

A star topology is a topology where every node in the network is connected to one central node. Every device in the network is directly connected to the central node and indirectly connected to every other node. The relationship between these elements is that the central network device is a server and other devices are treated as clients. The central node has the responsibility of managing data transmissions across the network. The central node or hub also acts as a repeater. In star topologies, computers are connected with a coaxial cable, twisted pair, or optical fiber cable.

11

**1.5.2.4.** **Tree Topology**

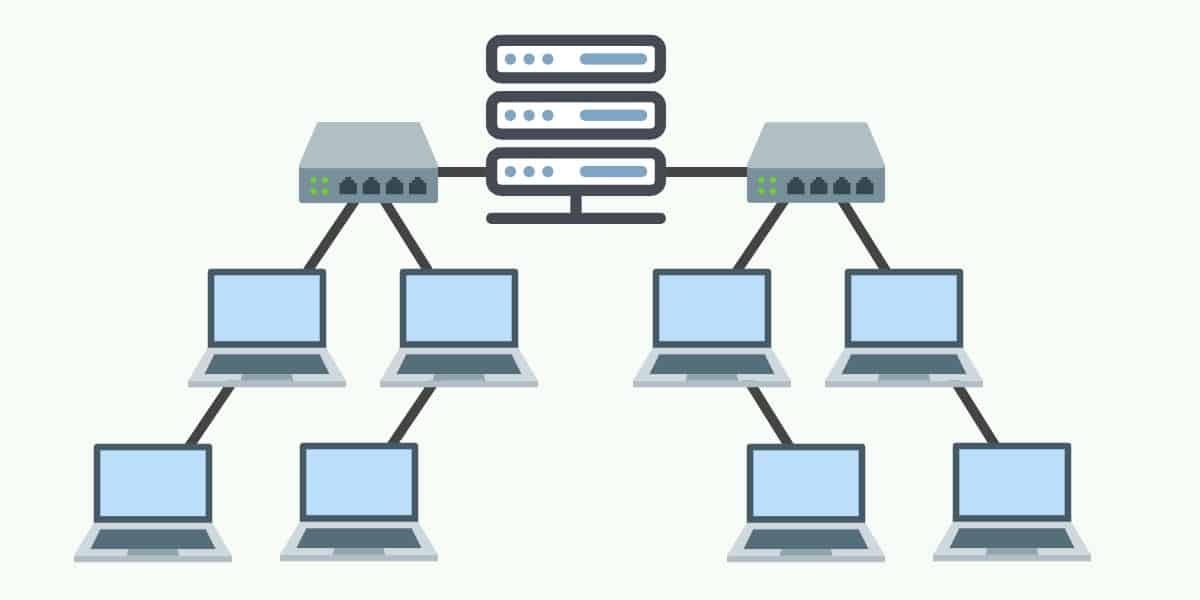


Figure 1.5.6. Tree Topology

As the name suggests, a tree topology is a network structure that is shaped like a tree with its many branches. Tree topologies have a root node which is connected to other node hierarchy. The hierarchy is parent-childwhere there is only one mutual connection between two connected nodes. As a general rule, a tree topology needs to have three levels to the hierarchy in order to be classified this way. This form of topology is used within Wide Area Networks to sustain lots of spread-out devices.

**1.5.2.5.** **Mesh Topology**

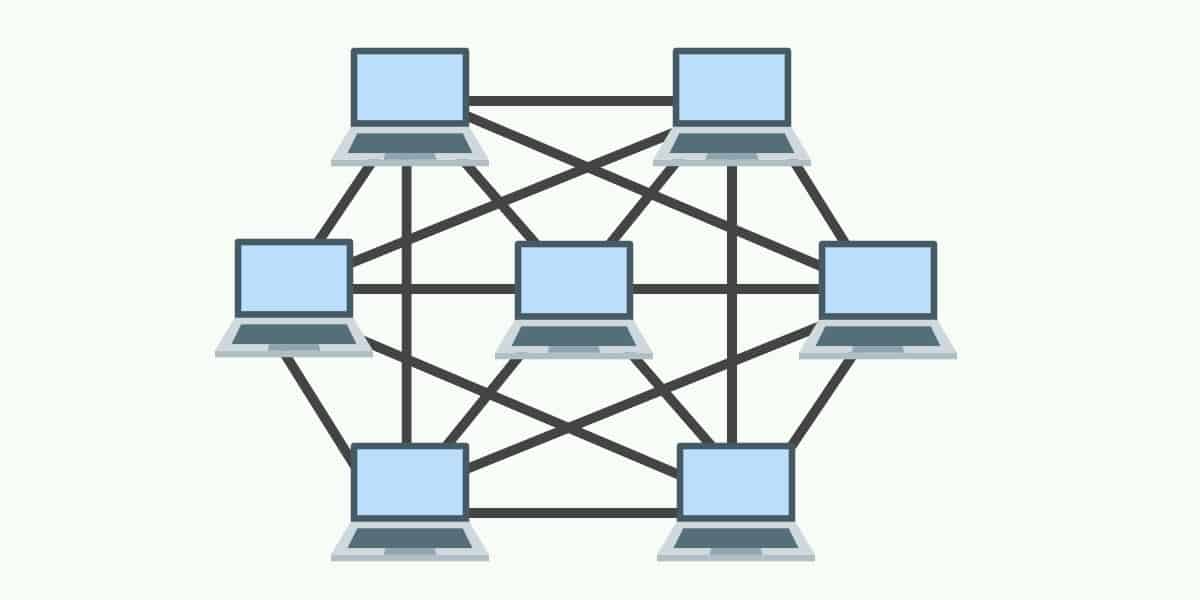


Figure 1.5.7. Mesh Topology

12

A mesh topology is a point-to-point connection where nodes are interconnected. In this form of topology, data is transmitted via two methods: routing and flooding. Routing is where nodes use routing logic to work out the shortest distance to the packet’s destination. In contrast flooding, data is sent to all nodes within the network. Flooding doesn’t require any form of routing logic to work.

**1.5.2.6.** **Hybrid Topology**

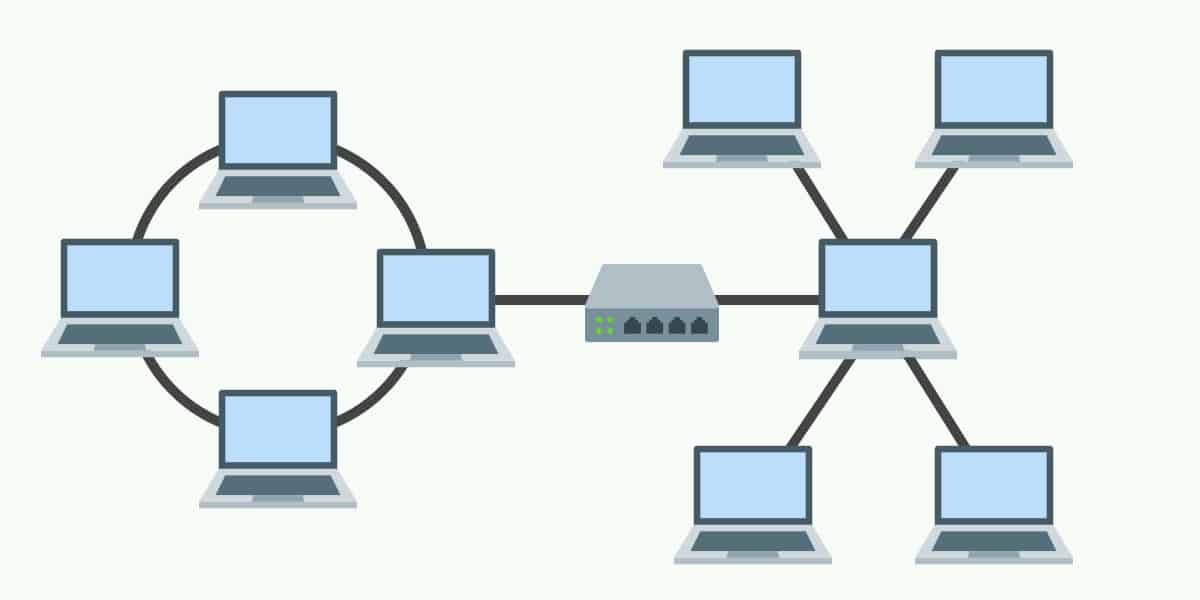


Figure 1.5.8. Hybrid Topology

When a topology is comprised of two or more different topologies it is referred to as a hybrid topology. Hybrid topologies are most-commonly encountered in larger enterprises where individual departments have network topologies that different from another topology in the organization. Connecting these topologies together will result in a hybrid topology. As a consequence, the capabilities and vulnerabilities depend on the types of topology that are tied together.

13

**1.5.3. Where to Use Xbee Modules ?**

Using Xbee modules together with microcontrollers, we can create many projects. To illustrate a few of these;

1. In parking tracking systems with wireless sensor network,

2. In home automation systems,

3. Vehicle control systems,

4. For robotic systems that need to be controlled wirelessly,

5. When the amount of humidity and temperature in greenhouses is transferred to the center where the greenhouses are followed,

6. The amount of energy obtained from the wind turbine and wind speed to the follow-up center

14

**1.5.4. Development of Arduino**

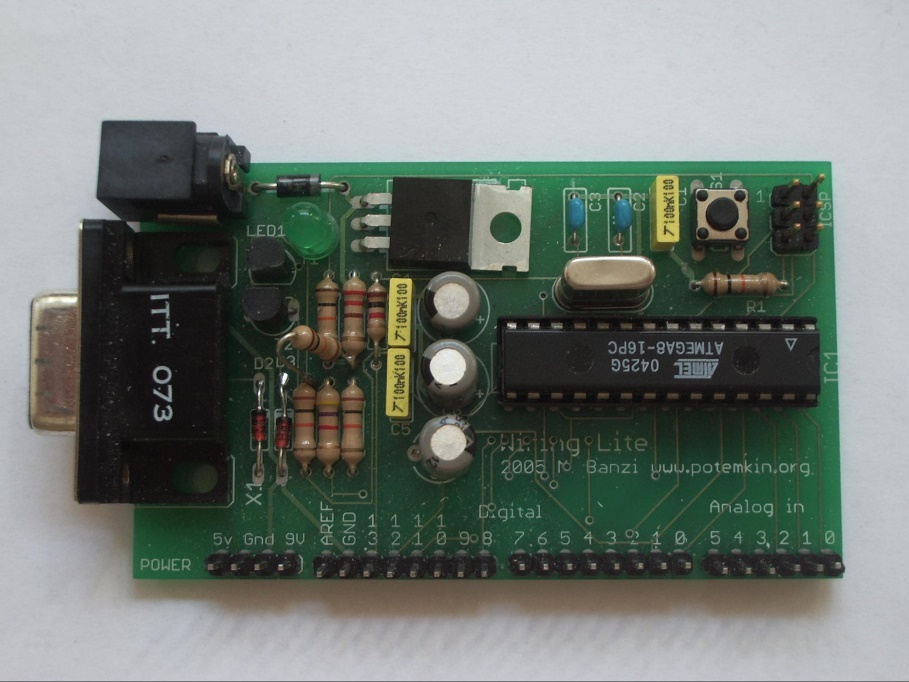
Physical or Embedded Programming is called physical programming to design physical systems that exchange data with the analog external world using software and hardware in the widest sense.

Physical Programming in daily use; with the help of sensors and microcontrollers, it is possible to obtain information from the analogue world by processing this information by means of mechanical, electromechanical, pneumatic systems or monitors, lighting devices etc. design / engineering products and “do it yourself” hobby projects.

The integrated system, which is formed by electronic hardware and software, which makes it a smart system by taking part in any system, is called embedded system. The biggest difference between the software in our computers is that it fulfills a single task and interacts indirectly with the user. It is possible to see this system in almost all of our belongings used in our daily life.

Embedded Systems typically work “embedded” within a larger system. For example, there are many mechanical sections in a washing machine or refrigerator, but one or more microcontroller-based systems that control all of them. These systems are units that enable the lar smart “devices we use. For example: when we open the lid of the refrigerator, it is a process that is caused by the unit which is in charge of this brain.

Microprocessor / Microcontrollers have complex structures, Embedded system designs are very costly and difficult to be programmed, at the same time difficult to program For such reasons, physical programming platforms have begun to be developed and even common in ready-to-use kits for easier programmable and customized designs.

Figure 1.5.9 First Arduino

15

**1.5.5. Embedded System**

An embedded system is a controller programmed and controlled by a real-time operating system (RTOS) with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints.[1][2] It is embedded as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today.[3] Ninety-eight percent of all microprocessors manufactured are used in embedded systems.[4]

Modern embedded systems are often based on microcontrollers (i.e. CPUs with integrated memory or peripheral interfaces),[5] but ordinary microprocessors (using external chips for memory and peripheral interface circuits) are also common, especially in more complex systems. In either case, the processor(s) used may be types ranging from general purpose to those specialized in certain class of computations, or even custom designed for the application at hand. A common standard class of dedicated processors is the digital signal processor (DSP).

Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, and largely complex systems like hybrid vehicles, MRI, and avionics. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

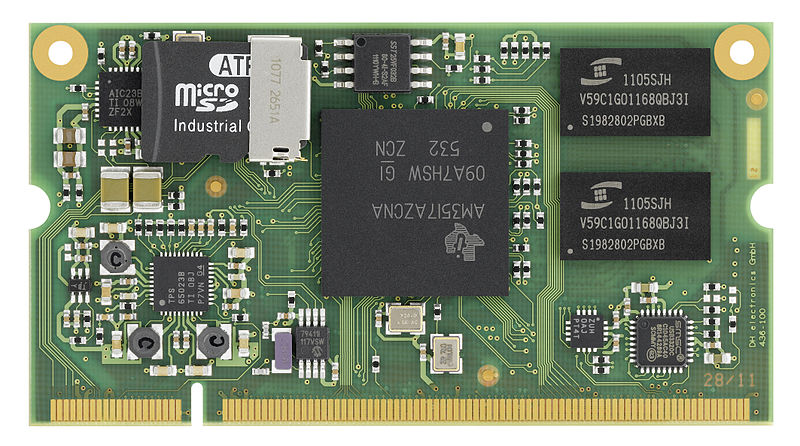


Figure 1.5.10.  
 16

**1.5.5.1. History**

One of the very first recognizably modern embedded systems was the Apollo Guidance Computer,[citation needed] developed ca. 1965 by Charles Stark Draper at the MIT Instrumentation Laboratory. At the project's inception, the Apollo guidance computer was considered the riskiest item in the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An early mass-produced embedded system was the Autonetics D-17 guidance computer for the Minuteman missile, released in 1961. When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits.

Since these early applications in the 1960s, embedded systems have come down in price and there has been a dramatic rise in processing power and functionality. An early microprocessor for example , the Intel 4004 (released in 1971), was designed for calculators and other small systems but still required external memory and support chips. In 1978 National Engineering Manufacturers Association released a "standard" for programmable microcontrollers, including almost any computer-based controllers, such as single board computers, numerical, and event-based controllers.

As the cost of microprocessors and microcontrollers fell it became feasible to replace expensive knob-based analog components such as potentiometers and variable capacitors with up/down buttons or knobs read out by a microprocessor even in consumer products. By the early 1980s, memory, input and output system components had been integrated into the same chip as the processor forming a microcontroller. Microcontrollers find applications where a general-purpose computer would be too costly.

A comparatively low-cost microcontroller may be programmed to fulfill the same role as a large number of separate components. Although in this context an embedded system is usually more complex than a traditional solution, most of the complexity is contained within the microcontroller itself. Very few additional components may be needed and most of the design effort is in the software. Software prototype and test can be quicker compared with the design and construction of a new circuit not using an embedded processor.

17

**1.5.6. Realized Projects**

**1.5.4.1. Panic Button**

Turkey's leading security company Pronet, thanks to the solution Panic Button, users not only against the risk of theft, receiving emergency health care and protection in the case of panic. With the Panic Button, especially the elderly people living alone, people with chronic illness or physical disabilities, feel more comfortable and safe in their daily lives. The Panic Button, which can be moved to all parts of the house, sends a signal to Pronet with a single touch, allowing the ambulance, police or fire department to reach the address in the fastest way. Thus, the eyes of the elderly, sick and disabled relatives do not stay behind.

Family elders, chronic patients and physically disabled people are safe;

Turkey's leading security solutions offered by the company, Panic Button, Pronet, especially the elderly in their families, handicapped or diabetes, is often preferred by those with chronic health problems such as epilepsy. The panic button not only makes the elderly and sick relatives more peaceful when away from them, but also makes them feel safer in their daily life in the home, thereby improving their quality of life.

The Panic Button, which can be used in every point of the house without any size, can be worn as a necklace, as a waistband or as a cola watch. So it's all over the house, at every moment. Pronet Panic Button, according to the user's needs or preferences, can be purchased in the desired number.



Figure 1.5.11. Panic Button

18

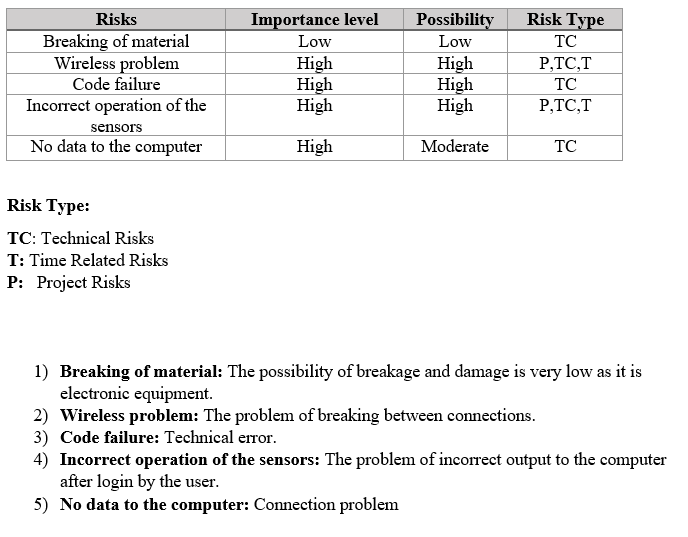
**1.5.6.2. Gloves that Turn Hand Gestures Into Writing and Sound**

The sign language was taken as a model during this project. The sign language is a visual language created by the hearing impaired when communicating among themselves, using hand gestures and facial expressions.  
   
The project uses the Arduino Mega card, a physical programming platform consisting of an I / O card and an application environment that includes an application of Processing / Wiring. The most important reason for Arduino to be preferred in this project is the Arduino libraries which enable everyone to program without requiring detailed information about the microcontroller. Many different arduino libraries are used in the project. The project consists of two parts.  
The first part is the glove we will wear. Flex on the glove has individual Flex sensors for each finger. Flex sensors increase the resistance on the sensor as they flex. The resistance of the Flex sensor changes when the metal pads are out of the fold.   
   
The second part of the project consists of Display and Speaker. According to the information from the sensors in the first part, output operations are performed in this section.  
  

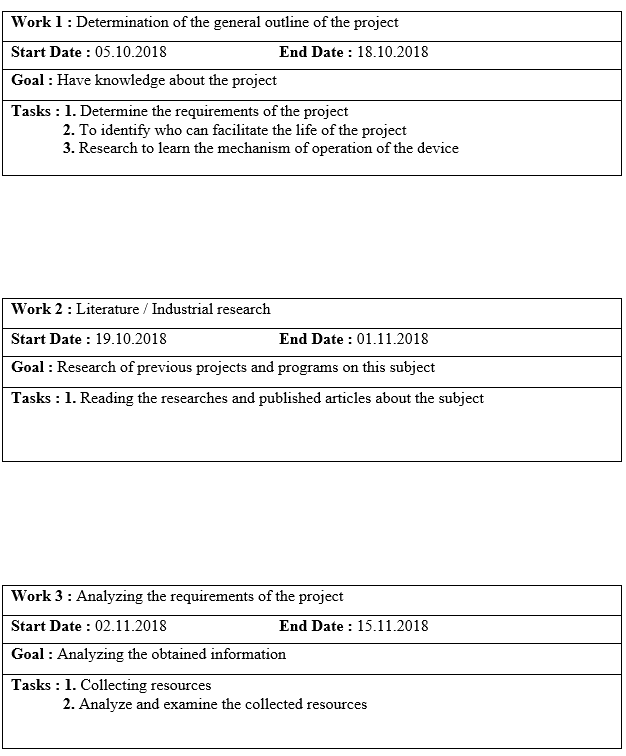

Figure 1.5.12. Gloves

19

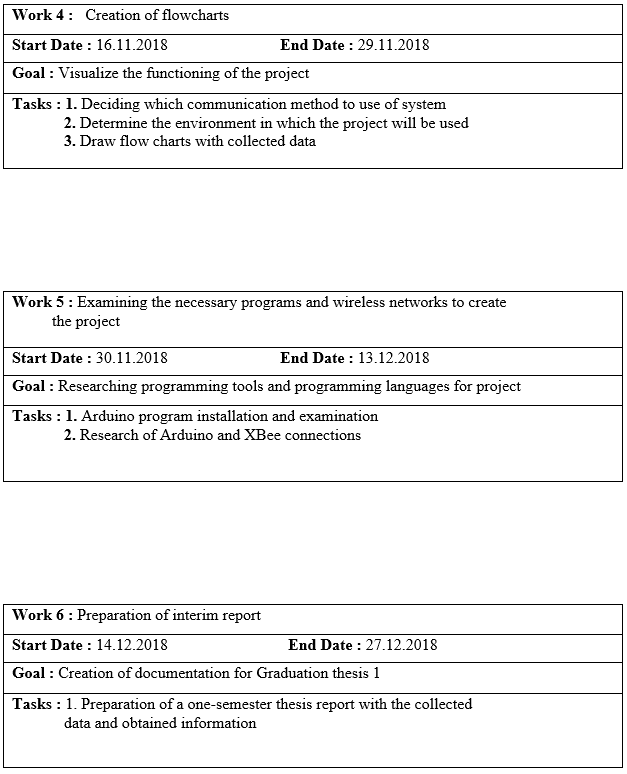
**1.6. Risk Assessment**

  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 20

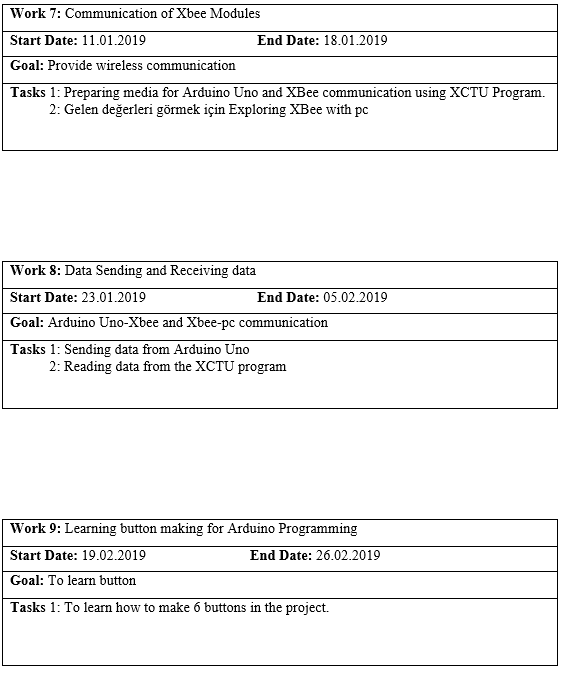
**1.7. Work Plan**



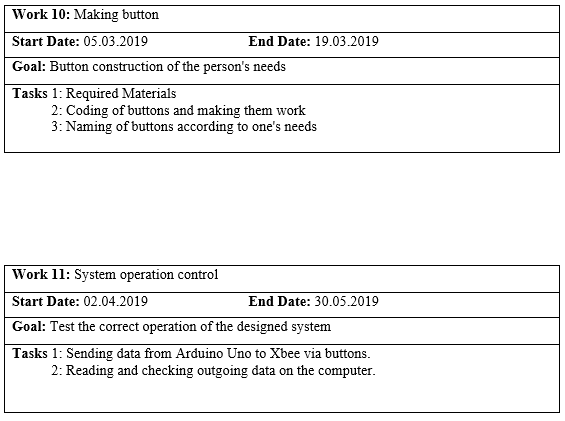
21



22

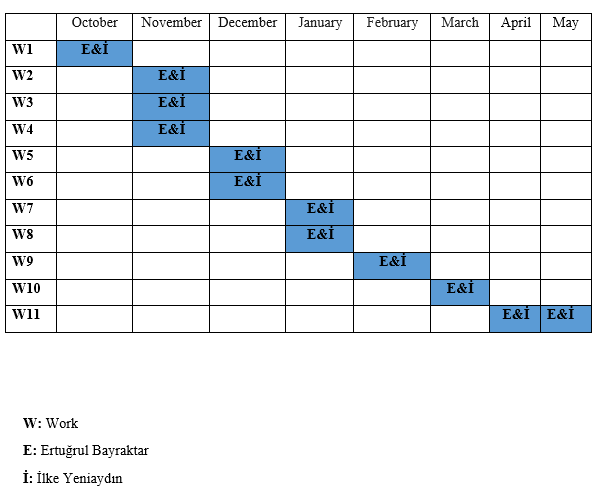


23



24

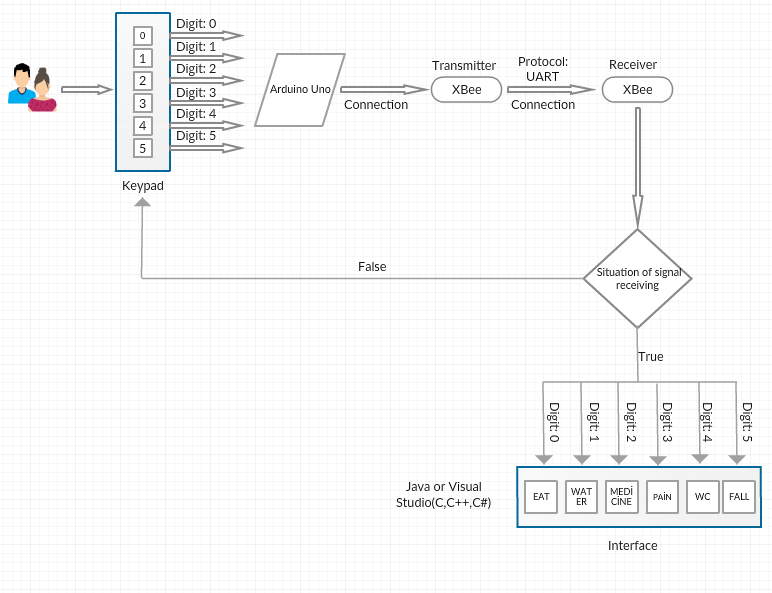
**1.8. Task Management**



25

**2. PROJECT SCOPE  
2.1. System Architecture  
2.1.1. System Overview**

Pressing a button to tell the user the need. The ID of this button reaches XBee by processing Arduino.  
XBee, which is in the transmitter function, sends the data to the receiver using the UART protocol.  
The computer displays the output corresponding to the data(ID) received by the computer.When one of the defined digits is not displayed on the computer, “Wrong” appears on the display.

  
 Figure 2.1. System Architecture  
  
  
  
  
  
  
 26

**2.2. System Development Plan  
2.2.1. XCTU**XCTU is a free multi-platform application designed to enable developers to interact with Digi RF modules via a simple-to-use graphical interface. XBee® RF modules include new tools that make it easier to install, configure, and test.

XCTU includes all the tools a developer needs to get up and work with XBee quickly. Developing unique features such as the XBee API framework builder that intuitively helps in creating and interpreting API frameworks for the XBees that graphically represent the XBee network, along with the signal strength of each connection, and the API mode, is easier than ever on the XBee platform.

Other key features of XCTU include:

• You can manage and configure multiple RF devices remotely (over-the-air) even connected device.  
• Two dedicated APIs and the AT console are designed from the ground up to communicate with your wireless devices.  
• You can now save your console sessions and upload them to a different PC running XCTU.  
• XCTU includes a series of embedded tools that can be run without connecting any RF modules:  
 •Frames generator: Easily create any API framework to maintain value.  
 • Frame interpreter: Decode an API framework and see specific frame values.  
 • Range test: Perform the range test between 2 radio modules of the same network.  
 • Firmware browser: Navigate through the XCTU's firmware library.

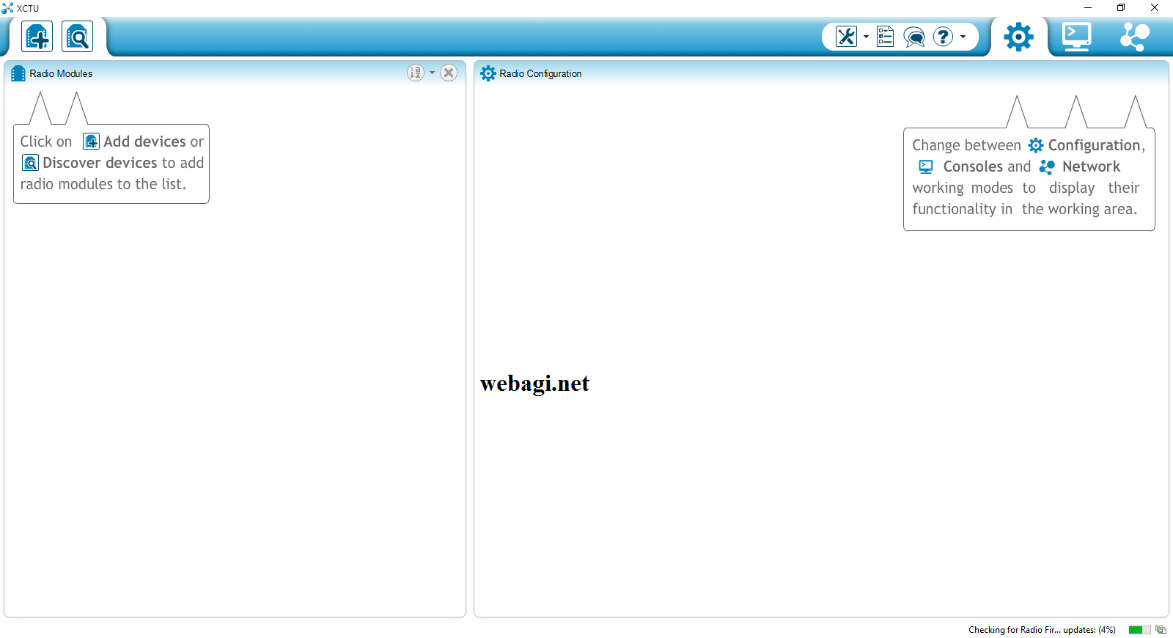


Figure 2.2. XCTU Interface

27

**2.2.2. Configuration of Arduino Uno and XBee Modüle**



Figure 2.3. Arduino Uno and XBee Connection

We see the arduino uno and xbee schematic.  
The circuit diagram is connected as shown in the figure and the system is ready.

28

**2.2.3. IO Expansion Shield**

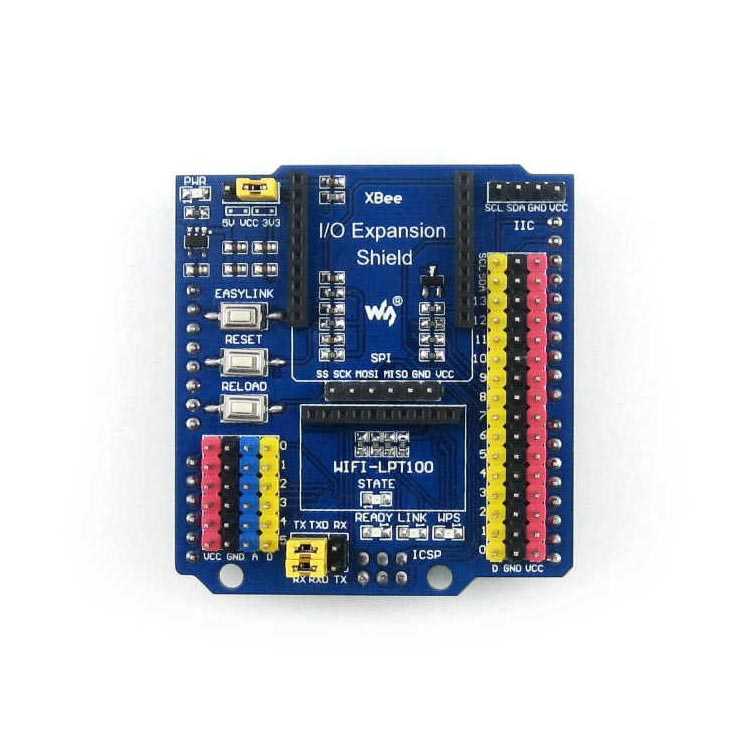


Figure 2.4. IO Expansion Shield

1) We use IO Expansion Shield, we need more pins inputs and outputs because we will use more than one button.

2) IO Expansion Shield is designed for XBee.

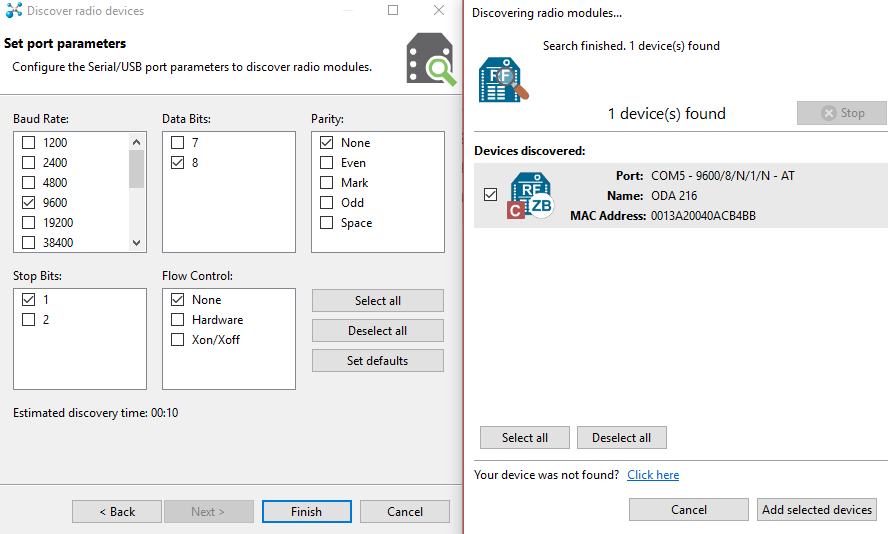
3) It is quite useful and protective against external influences.

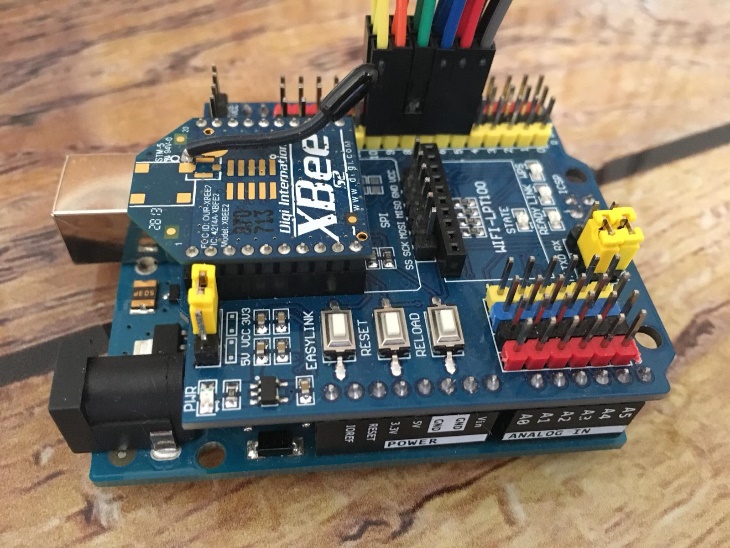
4) At other times, during the project development phase, it allows us to do many things because it has extra pins.

29

**2.2.4. How to use with XBee modüle with IO Expansion Shield**

How to use with XBee module XBee of MaxStream is a wireless communication module based on ZigBee technology. In an easy-to-use design, it can automatically transmit the inputted data to another XBee module by wireless connection.   
1) Prepare two XBee modules, two IO Expansion Shields and two Arduino development boards. In this document, we will divide them into two groups: Group A and Group B, of which Group A includes XBee-A, IP Expansion Shield-A and Arduino development board-A, and Group B includes XBee-B, IP Expansion Shield-B and Arduino development board-B.   
2) Insert the XBee-A into the XBee interfaces of IO Expansion Shield-A, and insert the XBee-B into the XBee interfaces of IO Expansion Shield-B respectively.   
3) Connect the IO Expansion Shield-A to the Arduino development board-A, and connect the IO Expansion Shield-B to the Arduino development board-B respectively.   
4) Set IO Expansion Shield jumpers: IO Expansion Shield User Manual Waveshare 4  Connect TXD to TX  Connect RXD to RX Notices: The USB interface, TX and RX pins of Arduino development board should be used in the processes described below, so please make sure that the projects running on the Arduino development board will not occupy any serial port.   
5) Start X-CTU software to configure XBee module on your PC.   
Select the corresponding COM port in PC Settings bar, and set relative parameters, such as baud rate. The factory default settings of XBee module is as followed:   
Baud rate: 9600  
Data Bits: 8   
Parity: NONE  
Stop Bits: 1

  
Figure 2.5. Xbee Configuration  
 30

  
Figure 2.6. Arduino Uno and IO Expansion Shield

As shown in the figure, the XCTU is ready for xbee Wireless Communication with the IO Expansion Shield attached to the Arduino Uno.

The same PAN ID was configured by the Coordinator via the X-CTU between the routers.

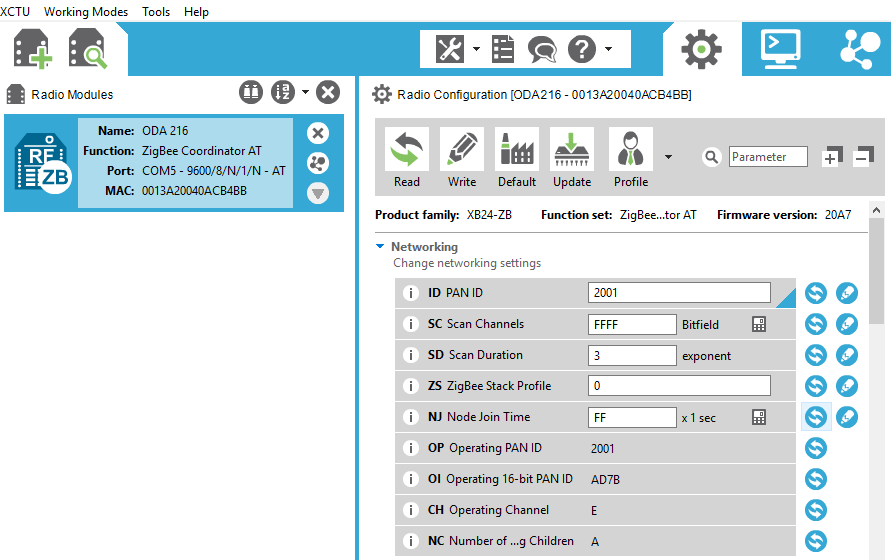


Figure 2.7. PAN ID

Thus, communication between the devices was achieved.

31

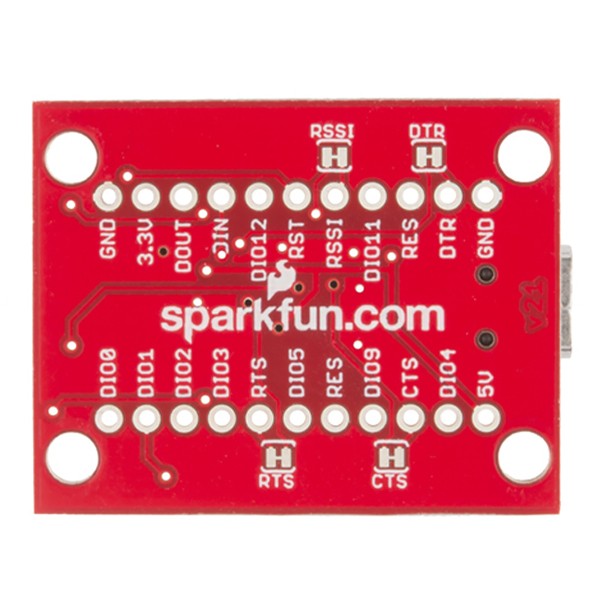
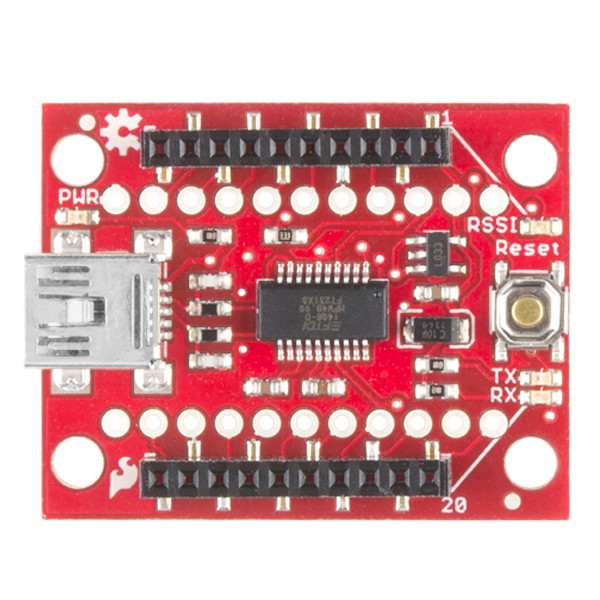
**2.2.5. Exploring XBee with PC**

Figure 2.8. Sparkfun XBee Explorer Figure 2.9. Sparkfun XBee Explorer

We have to configure our other Xbee in order to read the values that come to PC.  
Sparkfun XBee Explorer for this we need to use.  
This device is designed for XBee’s and is very useful.

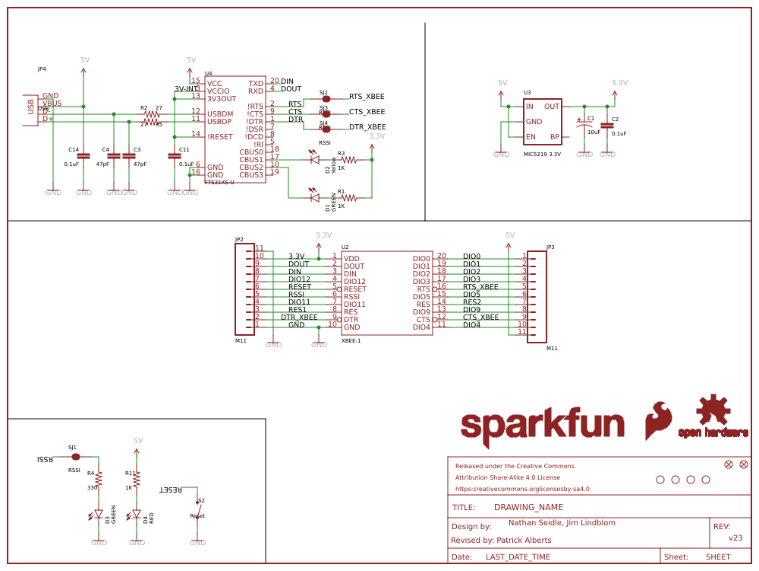
****

Figure 2.10. Sparkfun XBee Explorer Shematic   
  
 32

In the same way, the communication environment was provided by configuring the same PAN IDs in the XCTU program.

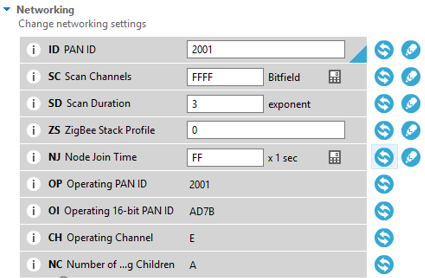


Figure 2.11. PAN ID



Figure 2.12. XBee

33

**2.2.6. Range Test Between XBee S2 Modules**

****

Figure 2.13.

Range test result;

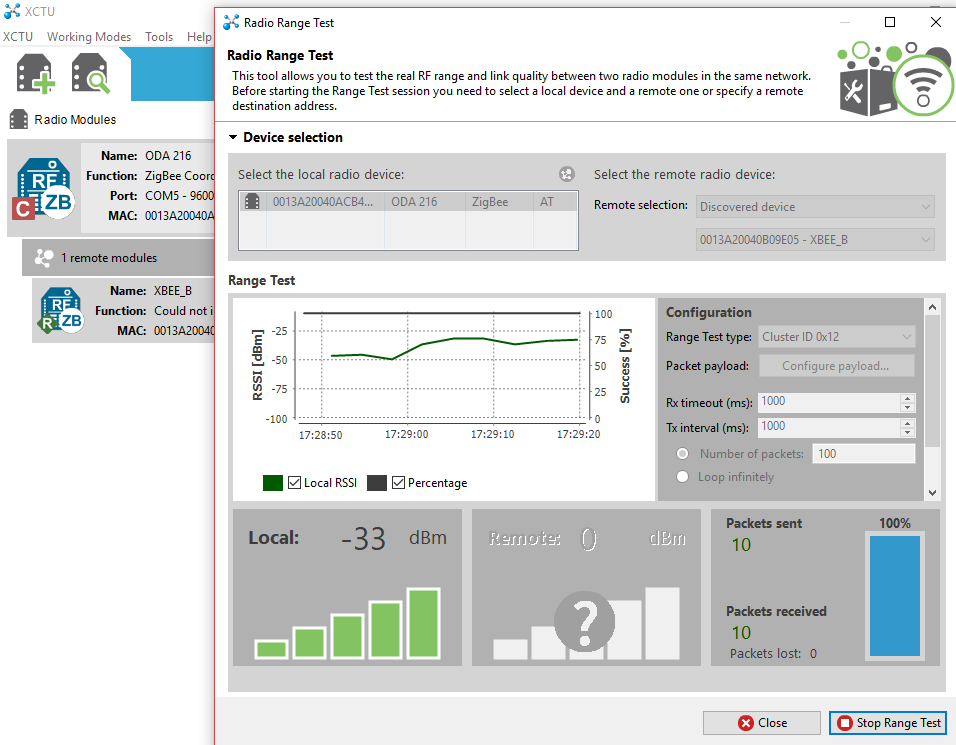
****

Figure 2.14. XBee’s Range Test

34

🡪 In order to test whether our XBee’s are communicating, we try to send data from Arduino Uno via XBee’s.

**Arduino Code:**

void setup() {

Serial.begin(9600);

}

void loop() {

}



Figure 2.15. Sending Data

Here, we opened the Arduino serial port display and sent the data "HELLO WORLD".

35

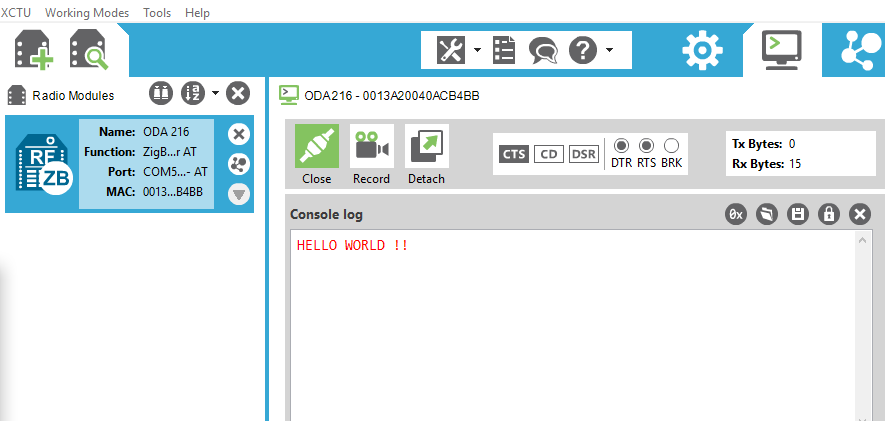


Figure 2.16. Receive Data

Data successfully reached the PC.

36

**2.2.7. Arduino Button Use**

We will do communication-assisted electronic equipment for the patients who have difficulty speaking and the elderly who cannot express their wishes.

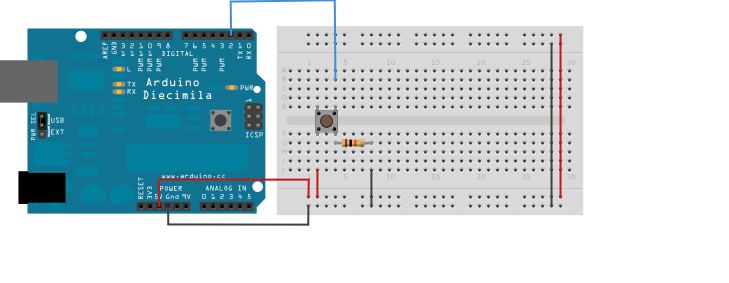
We're gonna do this with the Buttons. If the person needs what he needs to press the buttons and the person who is interested in the computer will appear on the request.

**2.2.7.1 Making Button**

Hardware

•Arduino or Genuino Board  
•Momentary button or Switch  
•10K ohm resistor  
•Hook-up wires  
•Breadboard

Circuit

Figure 2.17. Making Button

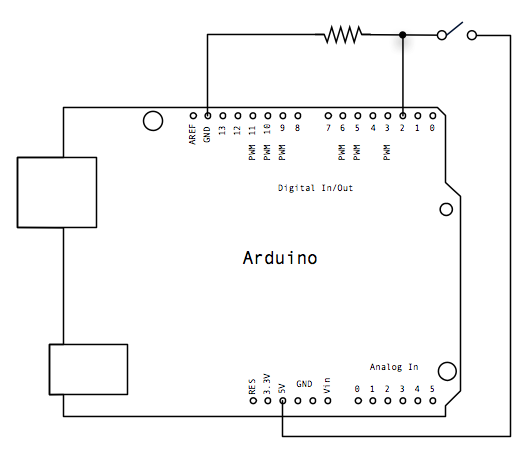


Figure 2.18. Making Button

37

**Arduino Code;**

#define Buton 10

void setup() {

pinMode(Buton, INPUT\_PULLUP); // initialize the pushbutton pin as an input:

Serial.begin(9600); // start serial connection

}

void loop() {

if(digitalRead(Buton)==1) // read the state of the pushbutton value:

Serial.println("EAT"); // write

delay(100);

}

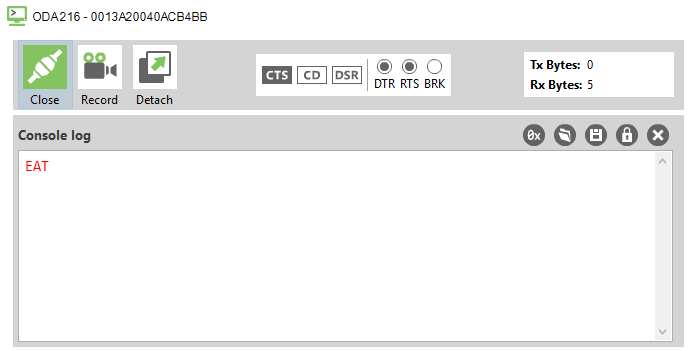


Figure 2.19. Receive Data

Data successfully reached the PC.

38

**2.2.7.2. Code Descriptions**

**2.2.7.2.1. setup()**

The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup() function will only run once, after each powerup or reset of the Arduino board.

**2.2.7.2.2. loop()**

After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

**2.2.7.2.3. pinMode()**

[Digital I/O]

**Description**  
Configures the specified pin to behave either as an input or an output. See the Digital Pins page for details on the functionality of the pins.  
As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode INPUT\_PULLUP. Additionally, the INPUT mode explicitly disables the internal pullups.

**Syntax**  
pinMode(pin, mode)

**Parameters**  
pin: the number of the pin whose mode you wish to set.  
mode: INPUT, OUTPUT, or INPUT\_PULLUP. See the Digital Pins page for a more complete description of the functionality.

**2.2.7.2.4. digitalRead()**

[Digital I/O]

**Description**  
Reads the value from a specified digital pin, either HIGH or LOW.

**Syntax**  
digitalRead(pin)

**Parameters**  
pin: the number of the digital pin you want to read

**Returns**  
HIGH or LOW

39

**2.2.7.2.5. delay()**

[Time]

**Description**Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

**Syntax**delay(ms)

**Parameters**ms: the number of milliseconds to pause. Allowed data types: unsigned long.

**Returns**Nothing

40

**2.2.7.3. 6 Buttons for Arduino Code**

#define Buton 10  
#define Buton2 9  
#define Buton3 8  
#define Buton4 7  
#define Buton5 6  
#define Buton6 5  
void setup()

{

pinMode(Buton, INPUT\_PULLUP);  
 pinMode(Buton2, INPUT\_PULLUP);  
 pinMode(Buton3, INPUT\_PULLUP);   
 pinMode(Buton4, INPUT\_PULLUP);  
 pinMode(Buton5, INPUT\_PULLUP);  
 pinMode(Buton6, INPUT\_PULLUP);  
 Serial.begin(9600);

}

void loop()

{

if(digitalRead(Buton)==1)  
Serial.println("EAT");  
delay(100);  
if(digitalRead(Buton2)==1)  
Serial.println("WATER");  
delay(100);  
if(digitalRead(Buton3)==1)  
Serial.println("MEDICINE");  
delay(100);  
if(digitalRead(Buton4)==1)  
Serial.println("PAIN");  
delay(100);  
if(digitalRead(Buton5)==1)  
Serial.println("WC");  
delay(100);  
if(digitalRead(Buton6)==1)  
Serial.println("FALL");  
delay(100);  
}   
 41

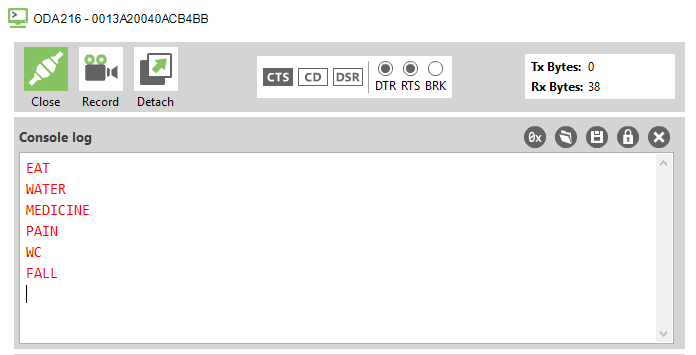


Figure 2.20. Receive Datas

Data successfully reached the PC.

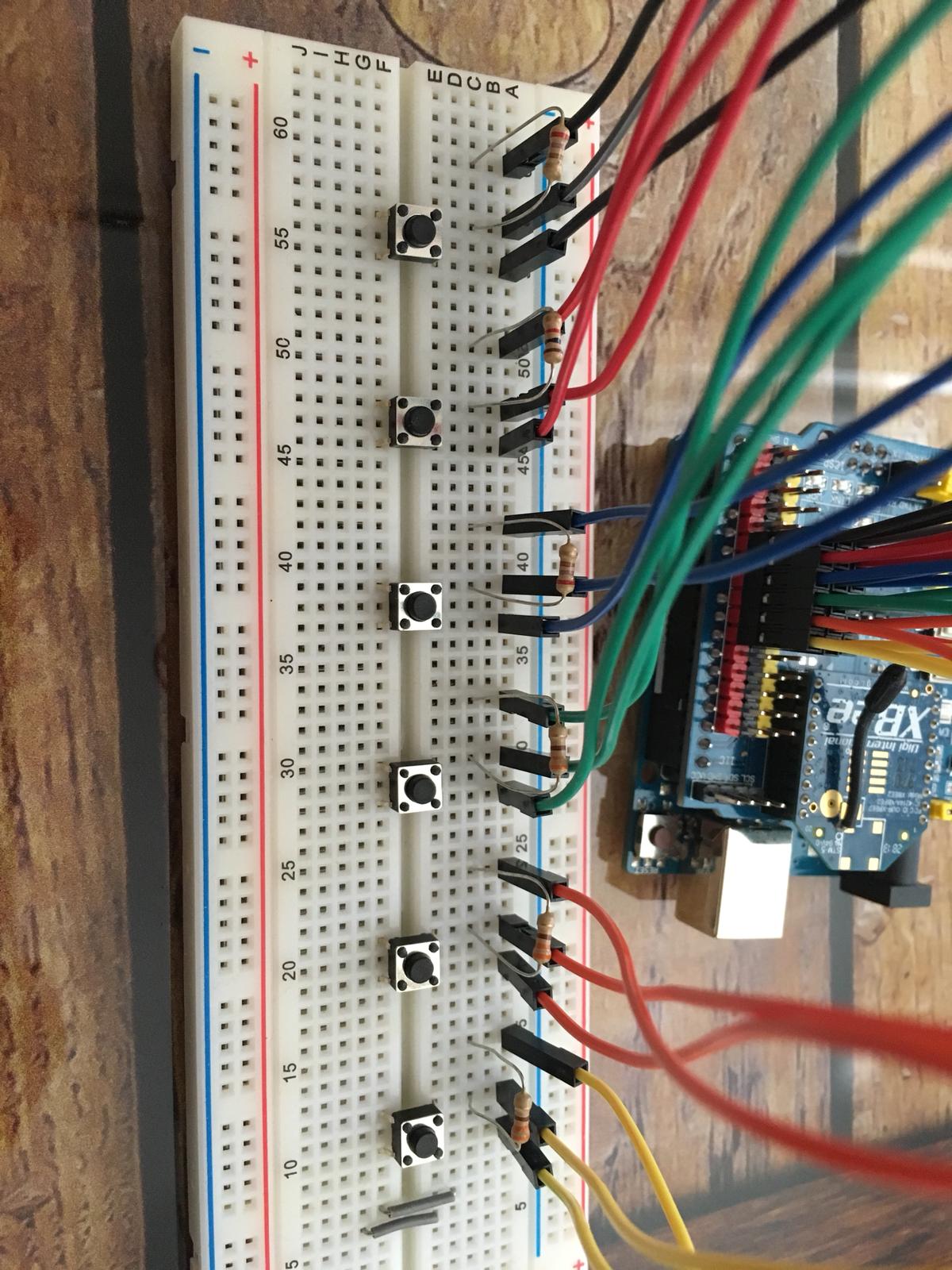
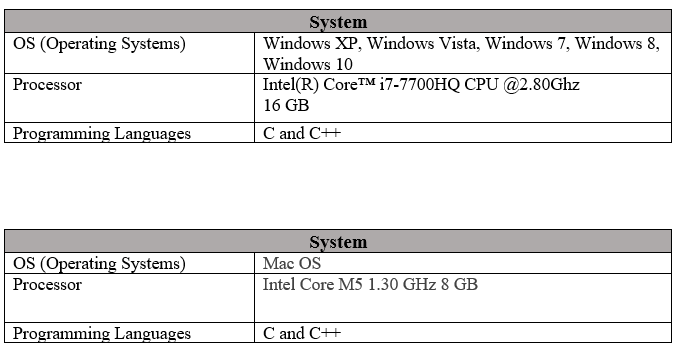


Figure 2.21. Buttons

42

**2.3. Project Platform**



43

**2.3.1. Operating Systems**

**2.3.1.1. Microsoft Windows**

Microsoft is a software and IT company with headquarters in the United States. Founded in 1975 by two university students in Seattle, Washington, USA. One of these students is Paul Allendi and Bill Gates. The main aim and foresight of the learners was that there could be a computer in each house. In the future, the students who are going to do a very large business with the Windows and MS-DOS company founded by moving to the summit of the world have become a dominant institution.

Windows is an operating system developed by Microsoft, which eliminates the need to write input from the keyboard, such as executing programs, giving commands, by approaching the user with graphical interfaces and visual messages. Instead of entering commands from the keyboard to perform operations in Windows, the same operations can be performed using the mouse. Ms-Dos is a DOS system developed by Microsoft. It was one of the most common operating systems used on PC compatible platforms in the 1980s.

The first version of Microsoft Windows is Windows 1.0 version made in 1985. This version is not functional and has not received much attention. Windows 2.0 version was released in 1987 and the interest in this was more than the 1st version. There are many reasons why Excel and Word for Windows are included in the release.

The real success came with the Windows 3.0 version made in 1990. This version offers the user a better task. In 1993, 3.1 was developed due to some shortcomings in version 3.0.

Then, Windows 95 will be developed and will make the most amount of sales until that time. This version will have a serious change to the interface and will include 255 characters of file names as well as features such as the built-in Internet Explorer and the start button that appeals to the user. After 3 years, 98 versions will be developed and marketed. In the course of these developments, respectively, Windows 2000, Windows Xp, Server 2003, Windows Vista, Windows 7 will show.

In 2015, Windows 8 was developed because Windows 8.1 achieved the desired success, and according to the statement made by Microsoft, Windows 10 announced that they would continue to update the new operating system on Windows 10.

44

**2.3.1.2. MAC OS**

The history of macOS, Apple's current Mac operating system originally named Mac OS X until 2012 and then OS X until 2016, began with the company's project to replace its "classic" Mac OS. That system, up to and including its final release Mac OS 9, was a direct descendant of the operating system Apple had used in its Macintosh computers since their introduction in 1984. However, the current macOS is a Unix operating system built on technology that had been developed at NeXT from the 1980s until Apple purchased the company in early 1997.

Although it was originally marketed as simply "version 10" of the Mac OS (indicated by the Roman numeral "X"), it has a completely different codebasefrom Mac OS 9, as well as substantial changes to its user interface. The transition was a technologically and strategically significant one. To ease the transition, versions through 10.4 were able to run Mac OS 9 and its applications in a compatibility layer.

It was first released in 1999 as Mac OS X Server 1.0, with a widely released desktop version—Mac OS X 10.0—following in March 2001. Since then, several more distinct desktop and server editions of macOS have been released. Starting with Mac OS X 10.7 Lion, macOS Server is no longer offered as a separate operating system; instead, server management tools are available for purchase as an add-on. Starting with the Intel build of Mac OS X 10.5 Leopard, most releases have been certified as Unix systems conforming to the Single Unix Specification.[1][2][3][4][5]

MacOS has retained the major version number 10 throughout its development history to date; releases of macOS have also been named after big cats (versions 10.0–10.8) or locations in California (10.9–present).

**2.3.2. Programming Languages**

**2.3.2.1. C Language**

History of C language is interesting to know. Here we are going to discuss a brief history of the c language.  
C programming language was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.  
Dennis Ritchie is known as the founder of the c language.  
It was developed to overcome the problems of previous languages such as B, BCPL, etc.  
Initially, C language was developed to be used in UNIX operating system. It inherits many features of previous languages such as B and BCPL.

45

**2.3.2.1. C++ Language**

The C++ programming language has a history going back to 1979, when Bjarne Stroustrup was doing work for his Ph.D. thesis. One of the languages Stroustrup had the opportunity to work with was a language called Simula, which as the name implies is a language primarily designed for simulations. The Simula 67 language - which was the variant that Stroustrup worked with - is regarded as the first language to support the object-oriented programming paradigm. Stroustrup found that this paradigm was very useful for software development, however the Simula language was far too slow for practical use.

Shortly thereafter, he began work on "C with Classes", which as the name implies was meant to be a superset of the C language. His goal was to add object-oriented programming into the C language, which was and still is a language well-respected for its portability without sacrificing speed or low-level functionality. His language included classes, basic inheritance, inlining, default function arguments, and strong type checking in addition to all the features of the C language.

46

**2.4. Work Sharing**

🡪Investigation of the Project  
🡪Report Preparation  
🡪Making Electronic Connections

- Arduino Uno XBee connection

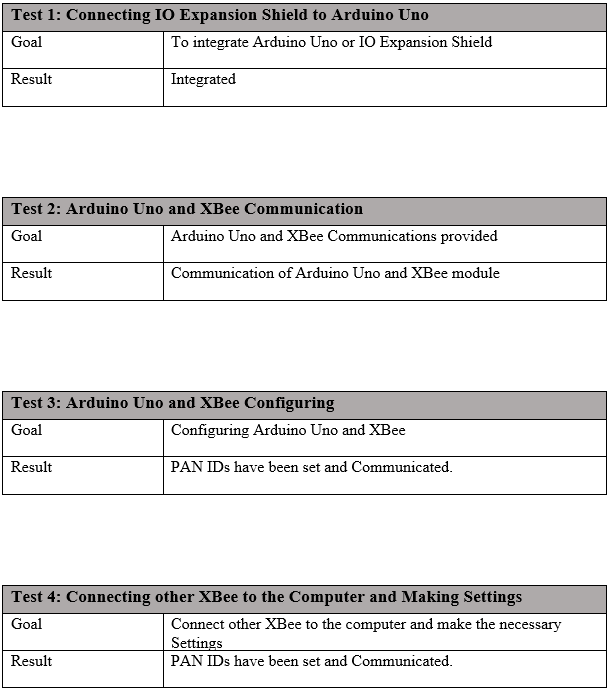
- Ensuring Communication

- Button construction

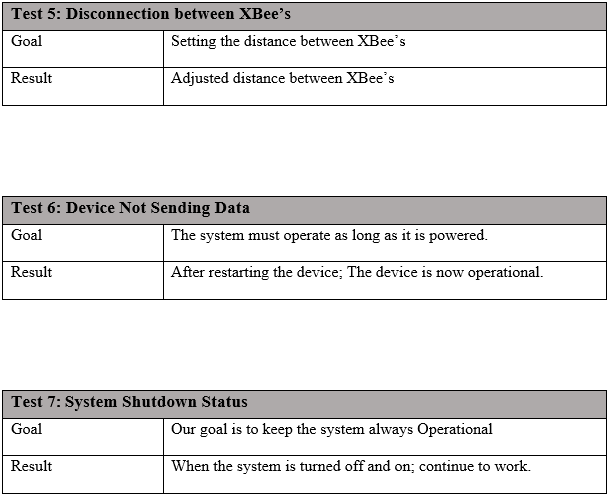
🡪Detection of errors

47

**2.5. Test Plan**



48



49

**3. User Guide**

**3.1. User Guide for Nurse**

🡪 Arduino uno power connection is made. Power Connection Input Can be powered from anywhere with any Usb 3.0.



Figure 3.1. Arduino Uno and USB 3.0 Cable

🡪 The green light above is expected to flash.

  
Figure 3.2. Green Light

🡪 If there is no interruption in the green light, the system has started to operate in a healthy way.

🡪 Arduino Uno has already sent the necessary software for the operation of the system.

50

🡪 We are ready to give you the ready-made IO Expansion Shield in Arduino Uno.

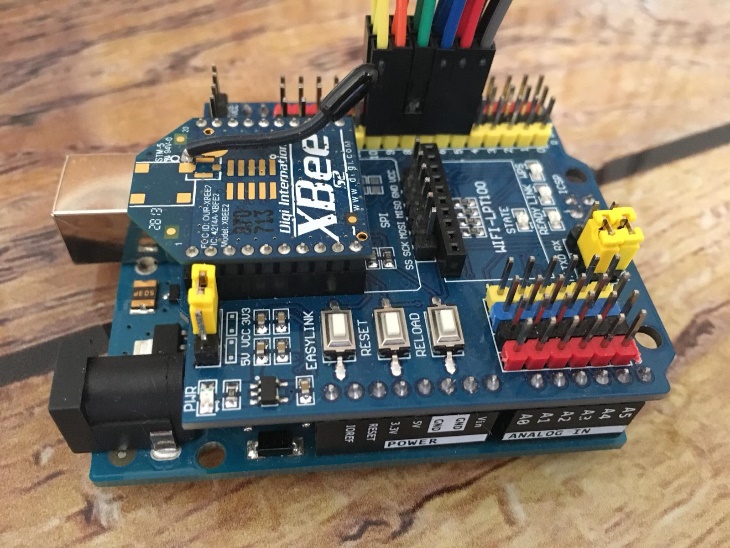


Figure 3.3. Arduino Uno and IO Expansion Shield

🡪 The most important reason is that we do this; IO Expansion Shield is a module designed for XBee.

🡪 We'll give you the bulk of the device so XBee will be connected to the device.

🡪 If there is a shortage of connection between XBee and Arduino Uno, you can follow the steps below;

1)



Figure 3.4. Arduino Uno and XBee Connection

51

2) Then scan the device in the XCTU program and perform the following steps;

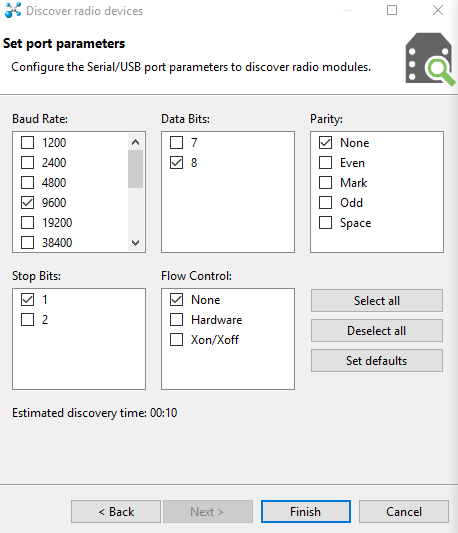


Figure 3.5. XCTU Set Port Parameters

52

🡪 To see the data from Arduino Uno, we need to connect the other XBee module to the computer.



Figure 3.6. XBee Module

🡪 Our other XBee is also powered by USB 3.0. And it is attached to the computer as in the figure above.

🡪 Then the required settings from the XCTU program must be made.

🡪 This XBee acts as a coordinator, the PAN ID must be the same as XBee that we connect to the Arduino Uno.

53

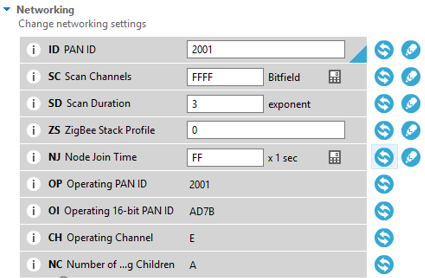


Figure 3.7. PAN ID

🡪 Process completed.

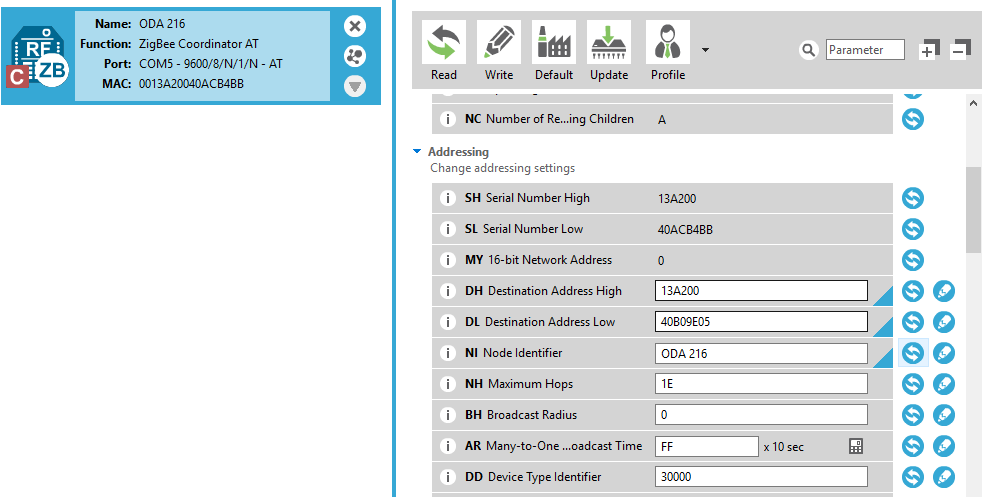


Figure 3.8. XCTU Settings

🡪 In the NI (Node Identifier) section, we can change the name of our XBee as we like and this name becomes a Uniqe value.

54

**3.2. USER GUIDE FOR PATIENTS**

Patients will use this device to inform the nurse of their needs.

When they press the button, the need will go to the server and the need will be met.

The buttons can be printed respectively:

-EAT

-WATER

-MEDİCİNE

-PAİN

-WC

-FALL

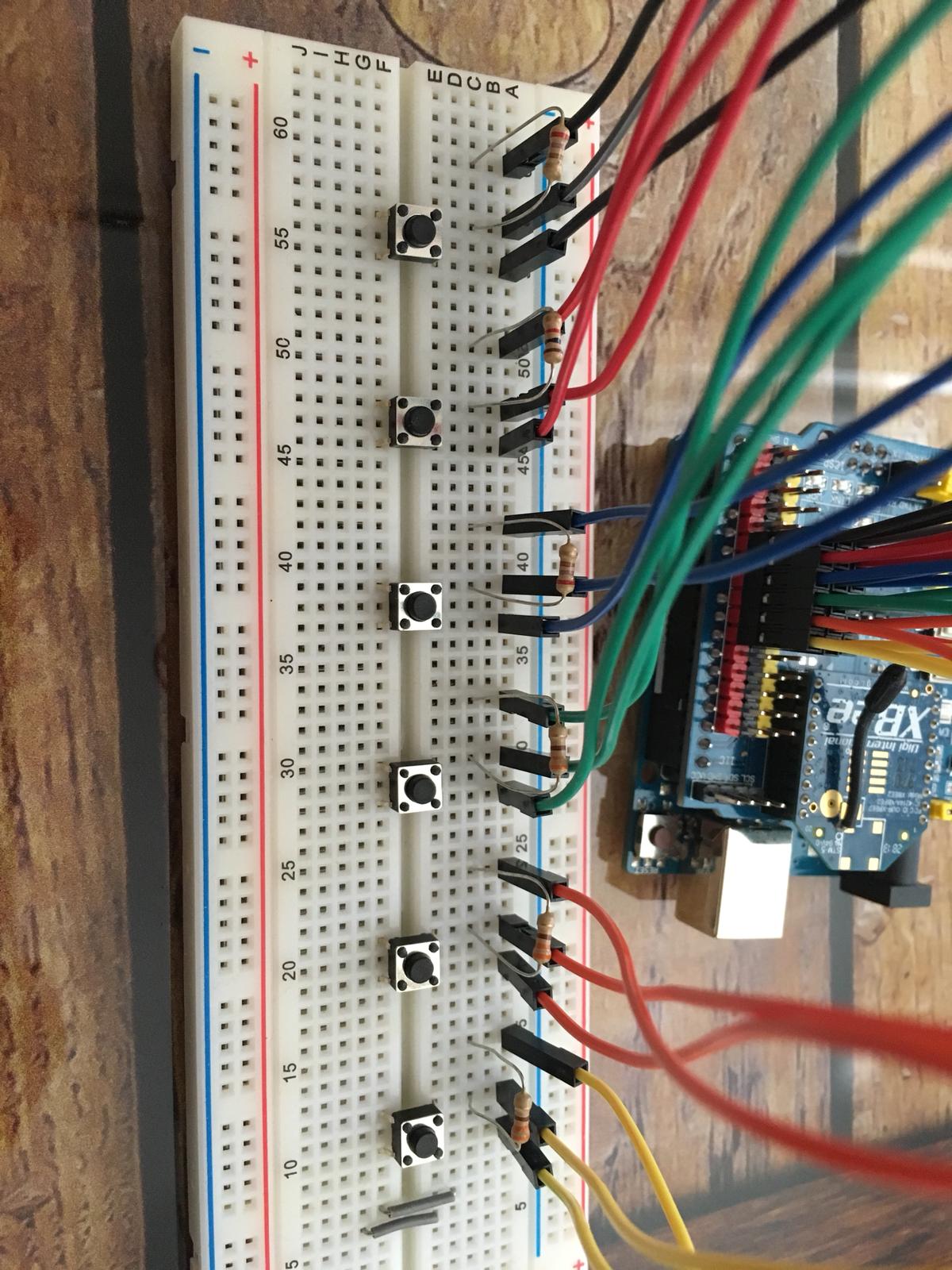


Figure 3.9. Patient Guide

55

**4. REFERENCES**  
  
🡪[www.robotiksistem.com](http://www.robotiksistem.com)

🡪[www.webagi.net](http://www.webagi.net)

🡪 www.alldatasheet.com

🡪www.triple-eee.com

🡪[roboturka.com.](http://roboturka.com.)

🡪www.wikipedia.org  
  
🡪[www.robotistan.com](http://www.robotistan.com)

🡪[www.electronicwings.com](http://www.electronicwings.com)

🡪[www.arduino.cc](http://www.arduino.cc)

🡪circuitdigest.com

**4.1. Figures**

Figure 1.4.1. Buttons……………………………………..…………………………………….3

Figure 1.4.2. Arduino Uno……………………………………………………….…………….4

Figure 1.4.3. IO Expansion Shield…………………………………………………….…….…5

Figure 1.4.4. XBee S2…………………………………………………………………….……6

Figure 1.4.5. Zigbee protocol architecture……………………………………………………..7

Figure 1.5.1. Home Automation Project……………………………………………………….8

Figure 1.5.2. Working Shematic……………………………………………………………….9

Figure 1.5.3. Bus Topology…………………………………………………………..………10

Figure 1.5.4. Ring Topology………………………………………………………………….10

Figure 1.5.5. Star Topology…………………………………………………………………..11

Figure 1.5.6. Tree Topology…………………………………………………………….……12

Figure 1.5.7. Mesh Topology…………………………………………………………………12

Figure 1.5.8. Hybrid Topology………………………………………………………….……13

56

Figure 1.5.9 First Arduino……………………………………………………………………15

Figure 1.5.10. …………………………………………………………………….…………..16

Figure 1.5.11. Panic Button……………………………………………………….………….18

Figure 1.5.12. Gloves………………………………………………………………..………..19

Figure 2.1. System Architecture…………………………………………………..………….26

Figure 2.2. XCTU Interface………………………………………………………..…………27

Figure 2.3. Arduino Uno and XBee Connection…………………………………..……….....28

Figure 2.4. IO Expansion Shield…………………………………………………………..….29

Figure 2.5. Xbee Configuration…………………………………………………..……..……30

Figure 2.6. Arduino Uno and IO Expansion Shield………………………………...….……..31

Figure 2.7. PAN ID……………………………………………………………….…………..31

Figure 2.8. Sparkfun XBee Explorer…………………………………………………………32

Figure 2.9. Sparkfun XBee Explorer…………………………………………………...…….32

Figure 2.10. Sparkfun XBee Explorer Shematic……………………………………………...32

Figure 2.11. PAN ID………………………………………………………………………….33

Figure 2.12. XBee…………………………………………………………………………….33

Figure 2.13……………………………………………………………………………………34

Figure 2.14. XBee’s Range Test………………………………………………………….…..34

Figure 2.15. Sending Data………………………………………………………………….…35

Figure 2.16. Receive Data………………………………………………………………….…36

Figure 2.17. Making Button…………………………………………………………….…….37

Figure 2.18. Making Button…………………………………………………………….…….37

Figure 2.19. Receive Data…………………………………………………………….………38

Figure 2.20. Receive Datas……………………………………………………….…………..42

Figure 2.21. Buttons………………………………………………………………….……….42

Figure 3.1. Arduino Uno and USB 3.0 Cable……………………………………….………..50

57

Figure 3.2. Green Light……………………………………………………………….………50

Figure 3.3. Arduino Uno and IO Expansion Shield……………………………….………….51

Figure 3.4. Arduino Uno and XBee Connection…………………………………..………….51

Figure 3.5. XCTU Set Port Parameters……………………………………………….………52

Figure 3.6. XBee Module………………………………………………………….………….53

Figure 3.7. PAN ID…………………………………………………………………….……..54

Figure 3.8. XCTU Settings…………………………………………………………...………54

Figure 3.9. Patient Guide…………………………………………………………………..…55

58