University “Politehnica” of Bucharest

Faculty of Electronics, Telecommunications and Information Technology

Voice Controlled Robot and Internet of Things

Dissertation Thesis

submitted in partial fulfillment of the requirements for the Degree of Master of Science in the domain *Electronics Engineering* study program *Advanced Microelectronics*

Scientific Advisor

Prof.Univ. Dr. Ing. Corneliu BURILEANU

Student

George-Laurentiu TUDORACHE

Statement of Academic Honesty

I hereby declare that the thesis Voice controlled robot and internet of things, submitted to the Faculty of Electronics, Telecommunications and Information Technology in partial fulfillment of the requirements for the degree of Engineer/Master of Science in the domain *Electronics Engineering*, study program *Advanced Microelectronics ,* is written by myself and was never before submitted to any other faculty or higher learning institution in Romania or any other country.

I declare that all information sources sources I used, including the ones I found on the Internet, are properly cited in the thesis as bibliographical references. Text fragments cited "as is" or translated from other languages are written between quotes and are referenced to the source. Reformulation using different words of a certain text is also properly referenced. I understand plagiarism constitutes an offence punishable by law.

I declare that all the results I present as coming from simulations and measurements I performed, together with the procedures used to obtain them, are real and indeed come from the respective simulations and measurements. I understand that data faking is an offence punishable according to the University regulations.

Bucharest, *date*

George-Laurentiu TUDORACHE

_____

(student's signature)

SUMMARY

1. INTRODUCTION

This paper studies the possibility of human- machine collaboration through direct verbal communication for simplifying and protecting our lives, from concept to prototype, the robot presented being able to reduce casualties caused by gas leaks or fires in the event of industrial plant failures or during natural disasters. By patrolling or being sent to the assigned area and by permanent monitoring of air quality or the presence of a toxic gas, it is able to provide active feedback to users.

Its sensors allow the provision of such information in real time to prevent loss of life during rescue operations where the rescuers are not aware of the "silent killer" hidden in the form toxic gas.

The use of MQ series gas sensors, 8-bit microcontrollers , pocket computeres with ZigBee wireless communication modules and voice commands provide an effective means of monitoring and control.

For this reason, such a system is recommended for use in any field that involves the risk of toxic gas leaks, such as the military , the chemical industry or the oil and gas industry .

People have always been affected by the negative effects of environmental problems. Air or, more precisely, oxygen is crucial for the survival of our race, so toxic air can cause widespread loss of life.

Nuclear tragedy in Japan in 2011 is testimony to this. Unfortunately technology has not yet been developed to reduce adverse effects so the focus was set on prevention. The focus was on the construction of stand-alone devices that need to be adaptable for use in any situation. [1]

Countries like the United States still spend millions of dollars in research to make robots that can help or even replace the role that people play in hostile environmental monitoring and in rescue operations, but developing countries like Romania cannot invest such money in this, but after demonstration of the functionality of the prototype developed, this technology can be used to save lives simply by measuring air quality and assisting rescue operations by providing real-time information that can be used to make well informed decisions before risking the lives of rescuers involved.[1]

The proposed solution involves a vehicle capable of patrolling while monitoring air quality or checking gas content, which is being controlled by voice commands and is also able to communicate wirelessly with the user , so this paper presents a robot voice control from concept to implementation.

1.1 Internet of Things introduction

Overview

Internet of Things (IoT) is a concept and a paradigm that considers active presence in the environment of a variety of things/objects that through wireless and wired connections and unique addressing schemes are able to interact with each other and cooperate with other things/objects to create new applications/services and reach common goals. [2]

The goal of the Internet of Things is to enable things to be connected anytime, anyplace, with anything and anyone ideally using any path/network and any service so it has to use the already existing internet IP infrastructure .
Internet of Things is a new revolution of the Internet. Objects make themselves recognizable and they obtain intelligence by making or enabling context related decisions thanks to the fact that they can communicate information about themselves. They can access information that has been gathered by other things, or they can be components of complex services. [2]
This transformation is concomitant with the emergence of cloud computing capabilities and the transition of the Internet towards IPv6 with an almost unlimited addressing capacity.
New types of applications can involve the electric vehicle and the smart house, in which appliances and services that provide notifications, security, energy-saving, automation, telecommunication, computers and entertainment are integrated into a single ecosystem with a shared user interface.

Enabling technologies for the Internet of Things are sensor networks,
RFID, M2M, 2G/3G/4G mobile Internet, semantic data integration, semantic
search, IPv6 and devices able to connect to the internet or to each other via Wi-FI / Bluetooth / Zigbee etc. are considered in and can be grouped into three categories:
1. technologies that enable “things” to acquire contextual information,
2. technologies that enable “things” to process contextual information, and
3. technologies to improve security and privacy. [2]

Fig 1.1 Applications of IoT[2]

My application is related to the IoT concept because it involves connecting a reconnaissance vehicle capable of monitoring environment parameters to the internet by sending the collected sensor information to an embedded server though an Zigbee gateway.

The information provided by the robot are available everywhere to any computer or smart phone able to connect to the servers IP address .

1.2 Motivation

The idea to build a robot remote controlled reconnaissance vehicle excites every electronics student so I decided to build my own remote controlled robot, but had to be different from what i have seen so far.

So I thought, "why not talk to the robot?", Instead of using a conventional remote control , because talking is humans natural way of communications , this make it reliable and easy to use

Therefore, we came up with the idea to design a utility robot voice control with a remote control that is able to listen, to understand, to act and to provide feedback to commands like a teammate . In this paper I proposed to implement and demonstrate the usefulness of a system that combines the simplicity of voice commands with the range of radio links and the security offered by a utilitarian reconnaissance robot. And by adding an internet connection robots versatility it's

enhanced even further by enabling a command centre to have access to collected data from it's targeted area , robot's status and to take appropriate decisions in case of emergencies .

1.3 Current State of achievements in the field

In terms of concept and as a practical realization both the idea of a robot which humans can talk to and the idea of a reconnaissance robot used for monitoring, particularly of possible gas leak, already exist worldwide.

To achieve genuine replicas of men, the famous Japanese robotics companies robots endowed their own humanoids with the ability to interact verbally with people thinking their creations as a replacement for human in some areas, particularly in services, or as therapists etc. An example is the famous company operating in the automotive industry: Honda which has built its own humanoid robot called Asimo Fig 1.1

Figure 1.2 humanoid robot Asimo [4]

It is able to understand voice commands or questions from those around him, can recognize objects and gestures made by people such as greetings. it can autonomously move in the desired direction by choosing the best route and these are only a few of the capabilities of the human robot designed as a replacement in the future.

These complex systems present currently a prohibitive price, which is why on the market are also available as low-cost alternative voice recognition modules in the form of simple electronic modules to simplify user interaction with automated systems dedicated to building utilitarian robots or commanded and controlled in a smart and easy way, as in this paper presents. An example of such a low-cost system is chosen in this paper and is called Easy VR manufactured by Veear company Figure 1.2, it can recognize up to 32 user-defined commands by recording them and therefore can equip any desired system with the hearing sense but it's not able of learning new words.

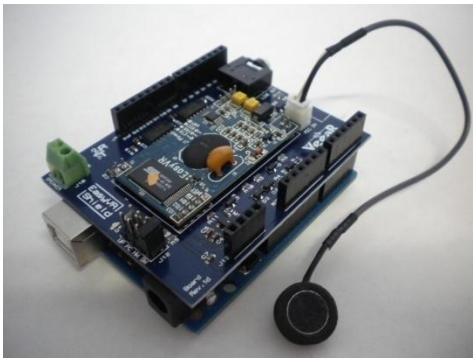


Figure 1.3 Shield Easy VR [18]

Another options involve Voice recognition on Cloud , corporations such as Google developed

Google Cloud Platform and Cloud Speech API which provided speech to text conversion powered by machine learning

This approach benefits from a big speech database hosted by their servers therefore the words which can be recognized are not limited , it depends on the application made to use this words how many commands it can interpret . This is used by Android OS voice command applications such as Ok Google which connected to the Internet to use its companies Voice processing Servers , therefore the Voice recognition is not done on the device but on Cloud .

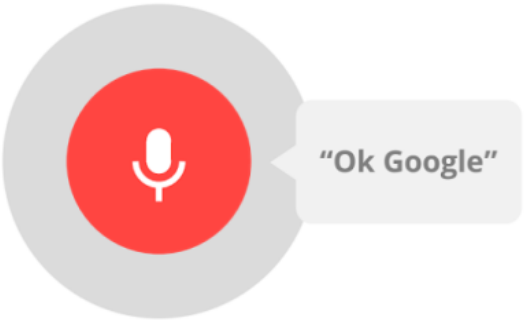


Figure 1.4 Ok Google symbol [6]

Another example is iPhone's Siri which also does its large vocabulary recognition on servers ,the voice recognition task is performed both locally and on the servers , if the word is too difficult to be figured out on the device . Apple also introduced an alternative to the cloud voice recognition

and crated an API named Open Ears which is based on CMU Sphinx open source project which has also been my choice in designing my system

Figure 1.5 iPhone's Siri [7]

The features provided by the two Operating systems can be used to control by Bluetooth any reconnaissance vehicle , but the reliability of the command system depends on the quality of the built -in microphone

An offline solution to the voice command control system is using an embedded pocket computer , the very popular Raspberry Pi and CMU Sphinx an open source speech recognition toolkit which allows the developer to use its own speech database recorded in any language , to train its own acoustic models and transform the voice commands to actions with Pocket Sphinx Continuous .

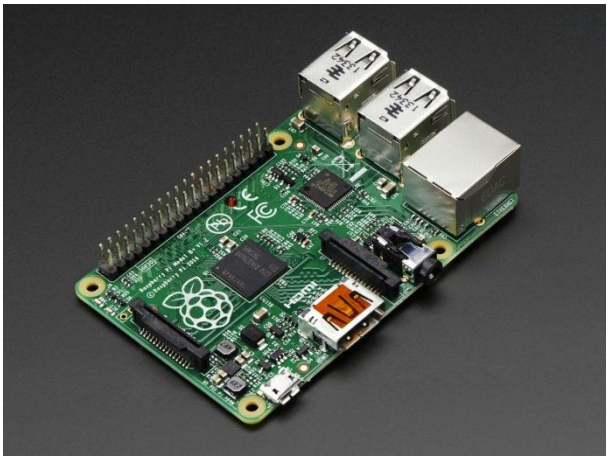


Figure 1.6 Raspberry Pi [8]

The accuracy of this system depends on the size of the recorded speech database and the number of speakers , see table for CMU Sphinx development recommendations

Voice recognition task	Speaker Dependent system	Speaker Independent system
Command and control	1 h of recordings 1 speaker	5 h of recordings 200 speakers

Table 1 Sphinx recommendations [9]

Nowadays states that allocate billion $ (ie USA) for the security of people working in hazardous areas or for the security forces deployed in different theatres of operations already have in the testing stage or have deployed on small-scale reconnaissance robots designed to replace men in high potential risk activities. The most pertinent example is the prototype of the robot which had been tested by the US military used in theatres of operations in the Middle East for inspection and even destruction of objects suspected of being improvised explosive devices, replacing engineers. An example is the robot entered the US Army since 2005, called SWORDS (Special Weapons Observation Reconnaissance Detection System). It is controlled by joystick and features a camera enabling remote controlling by an operator to inspect explosive devices such as the projectile in Fig. 1.7

Figure 1.7 Military SWORDS robot [10]

In terms of command and control systems of existing robots is widely used wireless control system by implementing a type transmitter - receiver both by using a computer or smart phone or by using a classic remote control with buttons and / or joystick.

The intelligent system from concept to prototype presented in this paper is the control with voice commands transmitted wirelessly to a utilitarian robot capable of making recognition and monitoring of environmental parameters. It uses the benefits of other concepts such existing in

the world: reliability and ease of operation system offered by direct verbal communication, coverage range and data transfer rate offered by wireless communication and security offered by a reconnaissance vehicle .

2. Objectives

2.1 Final objectives

My objective in this paper is to increase the versatility of the voice controlled reconnaissance vehicle by connecting it to an embedded server and therefore to the internet through a Zigbee Gateway in order to collect sensor data sent by the robot's sensors and make them accessible everywhere to any device (PC or Smartphone) connected to the internet which can access the IP address of the webpage hosted by the server.

Added components will include the embedded server built with an Arduino microcontroller and an Ethernet Shield which receives robot data through an XBee wireless transceiver which can be considered a gateway between the Ethernet network and the Zigbee network that the vehicle is part of.

In the robot's target patrol area such a server can be placed so a command center and not only the user can have access to the information (data or warnings) provided by the robot therefore enabling real time informing of intervention / emergency teams if something goes wrong .

Also the voice recognition feature of the robot will be improved by replacing the embedded voice recognition board (Easy VR) by a famous pocket computer named Raspberry Pi running a custom voice recognition software

2.2 Features-in-Brief

My project has the following features : - It’s controlled by giving voice commands

- It can recognize user defined number of commands in Romanian

- It’s capable of executing basic motion commands : Forward , Backward etc. and displaying it’s status on the special watch – This is the manual mode operation
- It’s capable of reading environment sensors : Temperature , Smoke , CO Gas and displaying the data
- It can display when asked his battery level
- It can confirm the actions asked by the user by giving a vocal report (It can talk not only listen)
- It’s able of warning the user when a certain environment parameter (e.g Smoke an Gas Level) exceeds an established threshold
- It monitors its own battery level and warns the user when it dropped under 30% to enable him to call back the robot

3 Brief description of project architecture

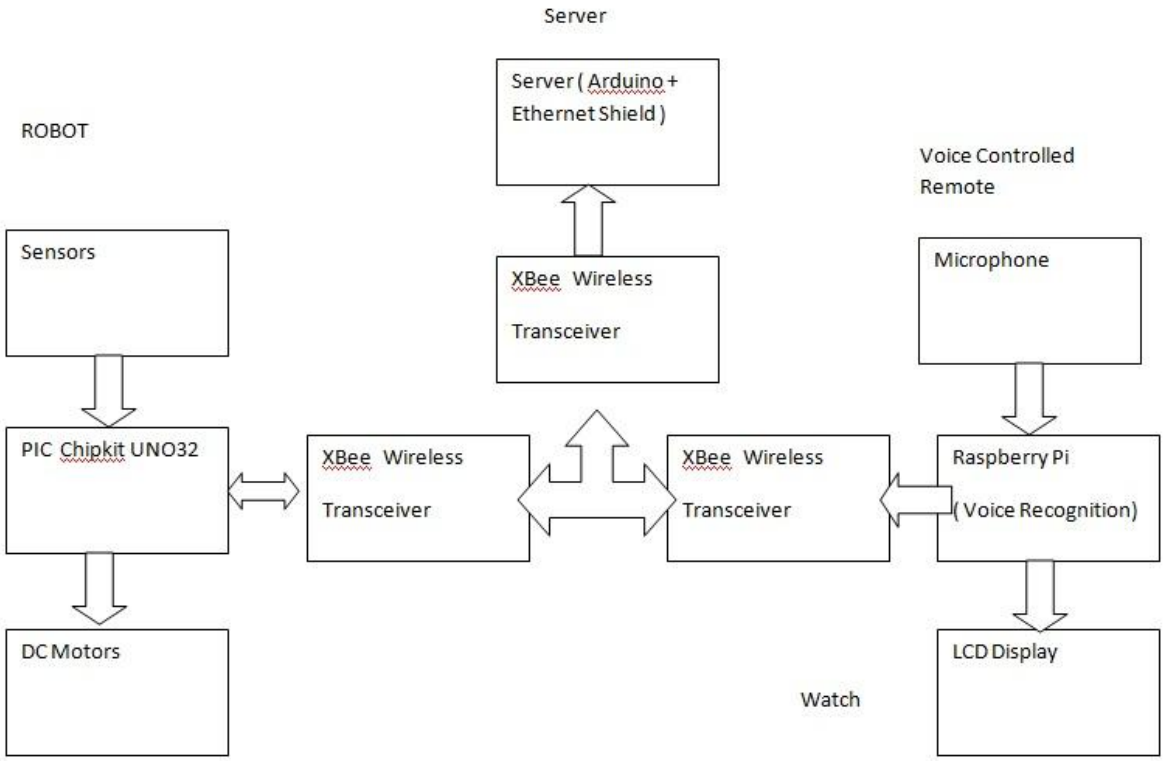


Figure 3.1 Project architecture

4. Detailed Design Description

4.1 Design Part 1: The Robot

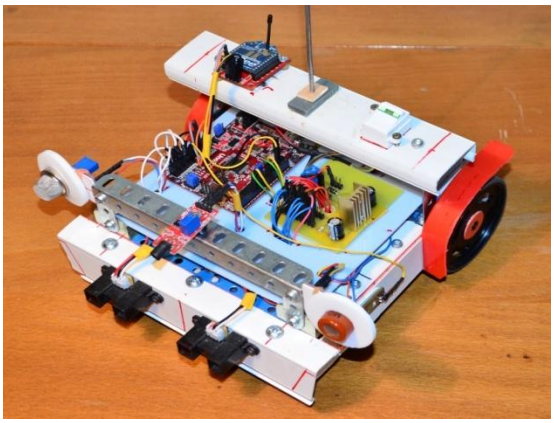


Figure 4.1 Reconnaissance robot

The robot is a two-wheel drive vehicle as you see in the Fig. 4.1, based on the ChipKit Uno32 microcontroller and powered from a 7.4 V to 5V DC to DC Converter

It has 6 sensors (4 environment sensors and 2 proximity sensors) and commands his 2 two motors via his 2 Motor Drivers.

The two-way Wireless communication through which the voice commands are received and the feedback is sent is established by the Xbee module connected to the Serial port of the Chipkit , its block architecture can be studied in Fig. 4.2.

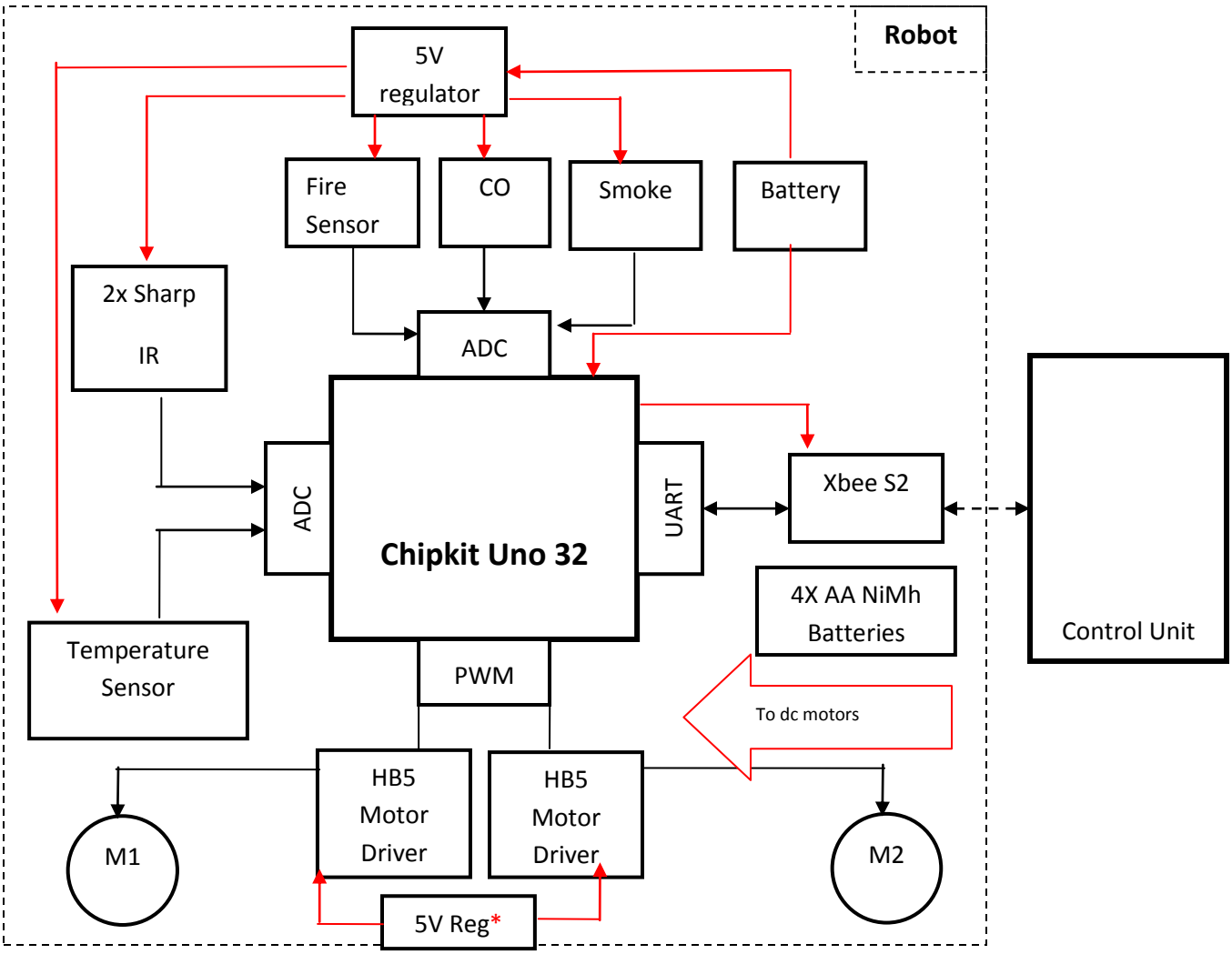


Figure 4.2 Robot block schematic

Comments : Red arrows represent power connections

5V Reg* : It's a single voltage regulator on the robot

There is a single ADC port on the board

The 4 AA NiMh Batteries power the motors 6 V when fully charged

4.1.1 PIC Chipkit UNO 32 Microcontroller

The Chipkit Uno32 development board supplied by Digilent is the heart of the robot. It takes care of driving the motors, reading the sensors and ensures communication with the Control Unit.

I found it appropriate for out project due to its performances and ease of use because of its similarities with the Arduino development board that i'm familiar with .

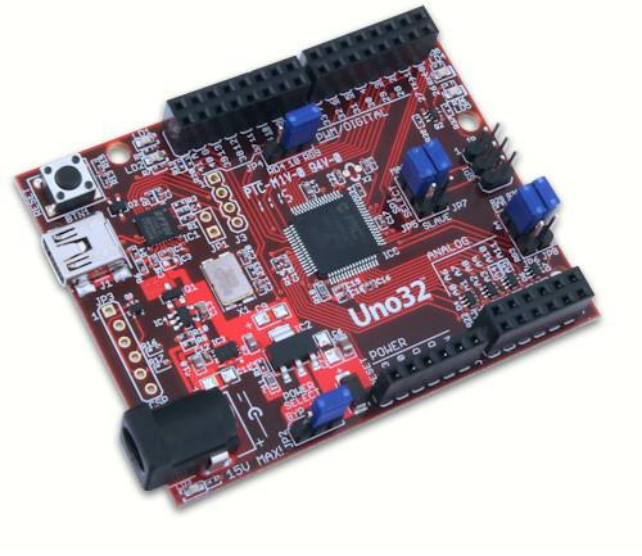


Figure 4.3 Chipkit Uno 32 Microcontroller[11]

- Microchip® PIC32MX320F128 processor
  - 80 Mhz 32-bit MIPS
  - 128K Flash, 16K SRAM

- Programmable using MPIDE IDE environment
- Compatible with many existing Arduino™ code examples, reference materials and other resources
- Can also be programmed using Microchip's MPLAB® IDE
- Compatible with many Arduino™ shields
- 42 available I/O
- 12 available Analog Inputs
- 2 Hardware Serial communication ports

- SPI & I2C Communication
- User LED [11]

4.1.2 DC Motor Control

In an analog circuit, motor speed is controlled by varying the input voltage to a circuit. In a digital circuit, however, only a logic high or logic low signal can be applied to the motor. Therefore, there are only two ways to control a motor digitally: use a variable resistance circuit to control the motor voltage, or, pulse the power to the motor. Since variable resistance circuitry is expensive, complicated, and wastes much energy in the form of heat, the better solution is pulse width modulation (PWM). Pulse width modulation is a digital method of transmitting an analog signal by generating a square wave signal with an adjustable duty cycle. [12]

The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast pace. The longer the switch is on compared to the off periods, the higher the power supplied to the load .

PWM has an important effect on DC motors: Inertial resistance is overcome more easily at startup because short bursts of maximum voltage achieve a greater degree of torque than the equivalent DC voltage.

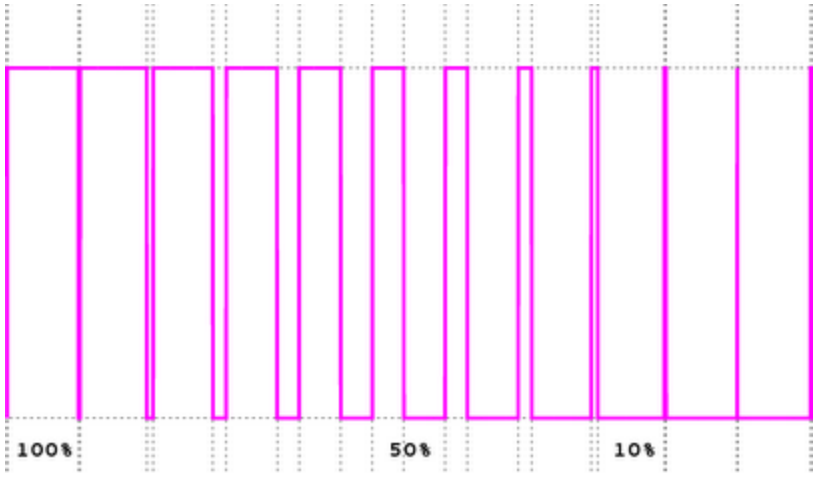


Figure 4.4 PWM Signal [15]

But because by being supplied by an output port of the microcontroller, the PWM signal cannot be applied directly to DC motors , instead it's used to command some power transistors connected in an H Bridge .

For the project I used 2 Pmod HB5 motor drivers which work with power supply voltages ranging from 2.5V to 5V. The HB5 is designed to work with either Digilent programmable logic system boards or embedded control system boards.

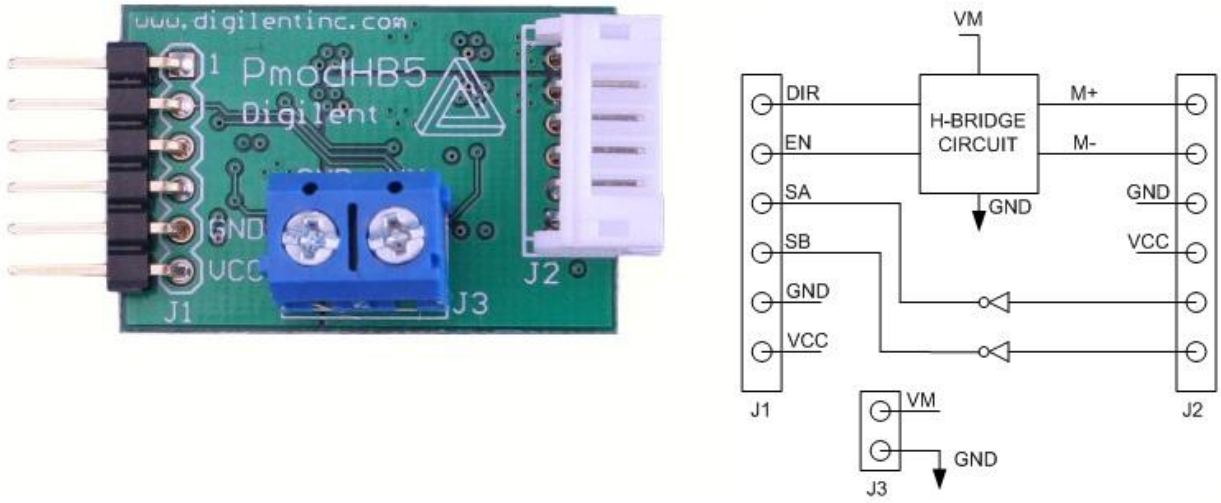


Figure 4.5 HB5 driver [12]

Besides the power pins , the pins needed for controlling the rpm and direction on rotation of the motors are : DIR = Direction and EN = Enable.

The motor rotation direction is determined by the logic level on the Direction pin and Current will flow through the bridge when the Enable pin is brought high. Motor speed is controlled by pulse width modulating the Enable pin.

The other two pins , the 2-channel quadrature encoder can be used in a control loop to precisely control the speed of the motor by constantly adjusting the PWM until the motors reach the desired speed. Both in the Arduino and Chipkit cases the PWM is generated by using the analogWrite ( ) function which can write values from 0 to 255 (0 – 100% duty cycle).

So I decided to create a function for all it's motion routines : forward () , backward() , left () ,

right () , with their parameters , the 2 PWM signals duty cycle's for the 2 motors

Than all he had to do was : to call the functions with different parameters , as shown in the following table , the speed of the wheels can be the same but "under the hood " they can turn in different directions depending the situation

FORWARD	forward (255 , 255)	full speed ahead , the wheels will rotate on the same direction
BACKWARD	backward (255 , 255)	full speed in reverse , the wheels will also rotate on the same direction
LEFT	Left (255 , 255)	Full speed spinning left , the wheels turn in opposite directions
RIGHT	Right(255 , 255)	Full speed spinning right , the wheels turn in opposite directions

Table 4.2 Motion functions table

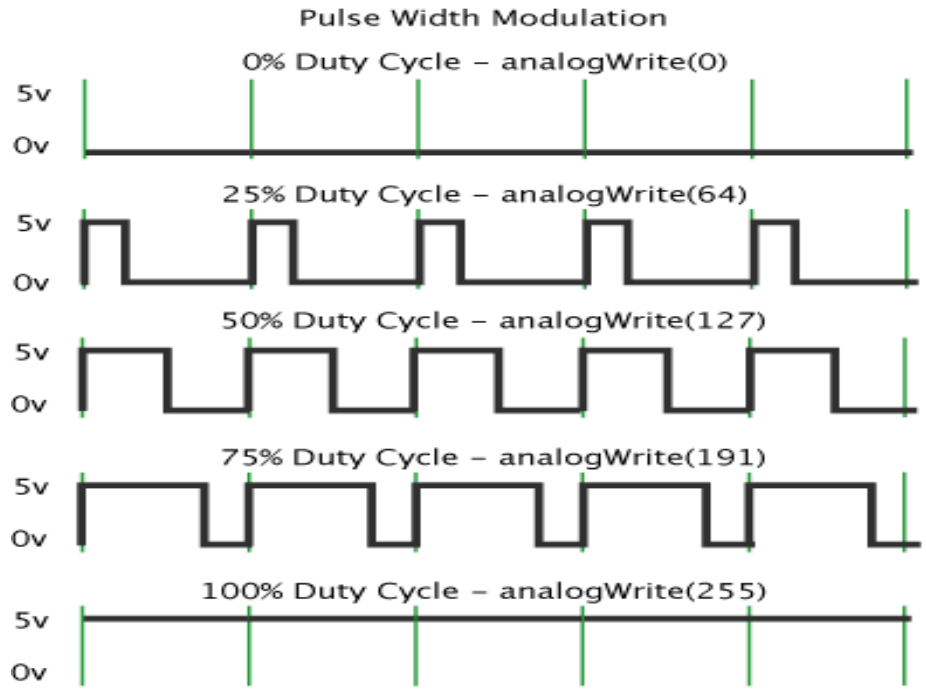


Figure 4.6 Correspondence between analog Write and PWM duty cycle [14]

4.1.3 Sensors

The robot has six analog sensors : 2 x IR Proximity Sensors , 1 Analog Temperature Sensor, 1 Flame Sensor , 1 Smoke Sensor and MQ-7 CO sensor and a Battery level measuring circuit which need to be analyzed by the Robot so we have a total need of 6 ADC Channels – 6 Analog Input Pins. Our need is easily covered by the ChipkitUno 32 which features Up to 12-Channel 10-bit Analog-to-Digital Converter with 1000 ksps conversion rate.

The digital I/O pins on the PIC32 microcontroller are 5V tolerant. The analog capable I/O pins are not 5V tolerant. To provide 5V tolerance on those pins, the Uno32 contains clamp diodes and current limiting resistors to protect them from 5V input voltages.

The ADC

Converts a continuous physical quantity (usually voltage) to a digital number that represents the quantity's amplitude. The conversion requires quantization of the input, and instead of doing a single conversion, the Microcontroller's ADC performs the conversions ("samples" the input) periodically. The result is a sequence of digital values which correspond to the samples taken from the continuous-time and continuous-amplitude analog signal and they represent a discrete-time and discrete-amplitude digital signal.

The resolution of the converter indicates the number of discrete values it can produce over the range of analog values. The values are usually stored electronically in binary form, so the resolution is usually expressed in bits. In consequence, the number of discrete values available, or "levels", is a power of two. [15]

In our case the 10-bit ADC has 1024 "levels " (0-1023).

But we don't have what to do with this values , they don't have measuring unit , they don't have physical meaning , they are simple quantifications of the sensors output voltage so we need to use data processing functions in order to obtain the value of the parameter of interest , this involves calibrations , testing ,and the voltage dependent's parameter characteristic

4.1.3.1 Temperature Sensor

For measuring the temperature I choose Texas Instrument's LM50

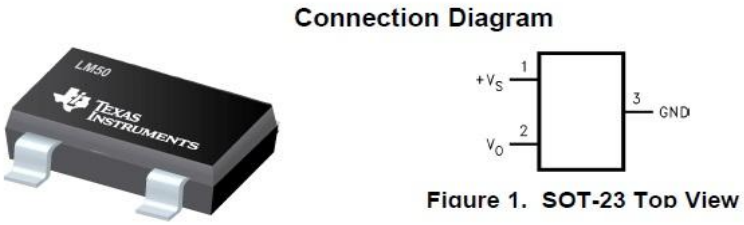


Figure 4.7 LM 50 package [16]

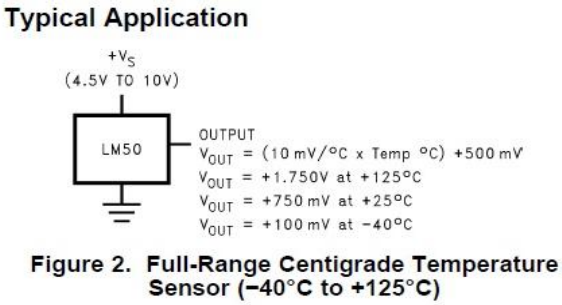


Figure 4.8 Sensor typical application [16]

It's an analog precision , linear. integrated-circuit temperature sensor LM50 , that can sense a −40°C to +125°C temperature range using a single positive supply from 4,5 to 10V and it gives an output voltage that is linearly proportional to Celsius (Centigrade) temperature (+10 mV/°C) with an DC offset of +500 mV (at -40°C output is 100mV and at +125°C output voltage is 1.75V) . [16]

The sensor is connected to the A3 Analog input pin , from which the Microcontroller will read an analog signal , the output voltage of temperature sensor I used a formula based on the specifications of the datasheet and determine the temperature with an maximum error of 1 degree Celsius .

4.1.3.2 CO Gas Sensor

Figure 4.9 MQ-7 Sensor [17]

For detecting gas presence i used MQ7 High Sensitivity Carbon Monoxide Detector Sensor
As it name says it detects common toxic gas like CO .

This type of device is important because there are many gases that can be harmful to humans or animals and CO is one of them a very dangerous which kills without warning – like a "silent killer" .

As shown in Fig 4.10, standard measuring circuit of MQ-7 sensitive components consists of 2 parts. one is heating circuit having time control function (the high voltage and the low voltage work circularly). The second is the signal output circuit, it can accurately respond changes of surface resistance of the sensor.

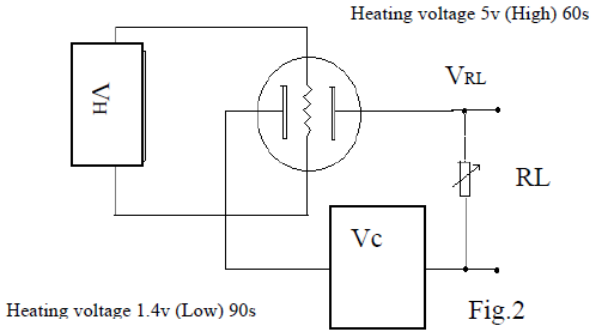


Figure 4.10 MQ-7 Internal circuitry [17]

The CO gas sensor must cycle through alternating voltages on its heater when active

It operates at 5V DC but unlike the MQ-2 sensor, the heater coil requires dual voltage of 1.4V and 5V . The sensor operates in two modes, A. *heating mode* where the heater coil is applied with 5V DC for 60s and

B .*cooling mode*, where the heater coil is applied with 1.4V for a duration of 90s. This mode is used to remove all the residue deposited on sensing element during previous cycle. Thus after 150s we can obtain the level of CO present in the atmosphere [17]. The timing is illustrated in Fig. 4.11

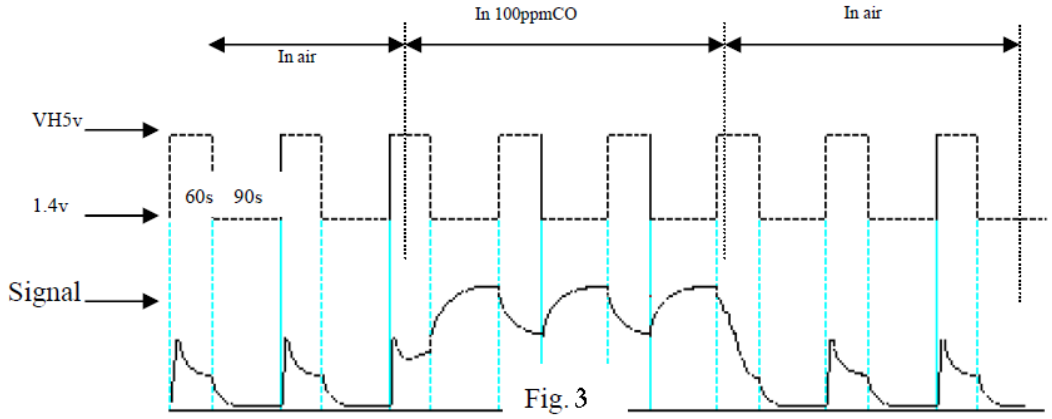


Figure 4.11 Timing of sensor operation [17]

These Gas Sensor Modules are designed to allow a microcontroller to determine when a preset gas level has been reached or exceeded in order to set an alarm limit when the presence becomes excessive .

I used this feature and set 6 levels in our CO Gas Sensors Function so we can know precisely how dangerous is the environment.

4.1.3.3 Smoke Sensor

For detecting gas presence i used MQ2 High Sensitivity Smoke Detector Sensor
Sensitive material of MQ-2 smoke sensor is SnO2 – tin dioxide ,

Figure 4.12 MQ-2 Sensor [18]

The internal circuitry is similar to the one described in the previous subchapter

The sensor can also detect gases like common flammable and combustible gases – Methane , Butane, LPG, smoke , etc .

Usage for notification leakage in home & environment applications. This type of device is important because these gases can produce fire and explosions when ignited with or without intent .

Just for the CO Gas sensor's case I used smoke levels in order to set an alarm limit when the presence becomes excessive .

So in our program we also 6 levels in our Smoke Sensors Function so we can know precisely how dangerous is the environment.
For detailed explanation please consult the datasheet

4.1.3.4 Battery Level

In order to measure the robot's battery level I used a very simple circuit – a voltage divider in order to map the batteries voltage range which cannot be measured by directly connecting it to

the microcontroller , to a 0…3.3 V range with current limitation in order to be read by the ChipKit's ADC.

So in our case the 2 Li-on 3.7 Batteries provide a voltage range from 7 V when discharged to 8.2 V when fully charged . We use a divider with 2.5 dividing ratio in order to bring the interval to a maximum 3.3 V .

Using a formula and a mapping function in the Battery_level() macro we restored the voltage measured from 3.3 V maximum to it's original range and map it from 10 % - 7V to 99 % - 8.2 V

4.1.3.5 Proximity Sensors

For detecting obstacles in the robot's proximity I used 2 analog distance measuring sensors from Sharp **GP2Y0A02YK0F IR Sensors**.

Figure 4.13 Sharp IR sensor [19]

The working principle can be described as follows :it sends and electro-magnetic beam (IR) using an IRED (Infrared emitting diode) that when it hits an object it reflects back to the sensor which is collected by the PSD (Position sensitive detector) which, depending on where the beam hit the detector (depending on the reflected beam's angle – smaller angle involves big distance , meanwhile bigger angle involves the opposite) it gives an inverse proportional voltage with the distance at its analog output based on which we can determine the distance to the object using the Microcontroller . In order to calculate the distance we must consider a non-linear (falling exponential) dependence of the voltage with the distance which is provided by the datasheet. However this was not necessary for our application , we didn't need to know the exact distance, so we used the number of quantums returned by the ADC to estimate a distance.

Distance measuring range is from 20 to 150 cm ,

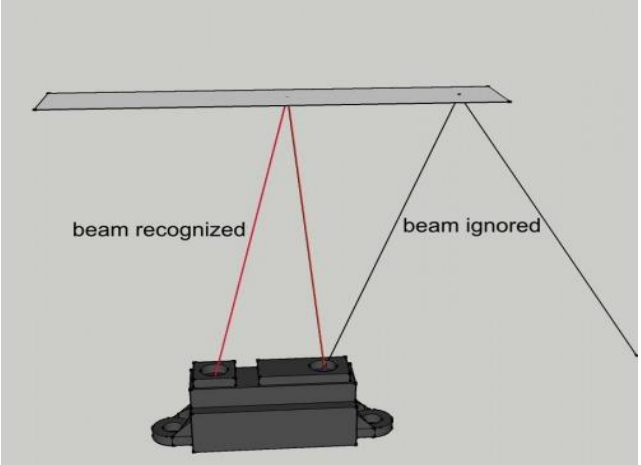


Figure 4.14 Sensor working principle [19]

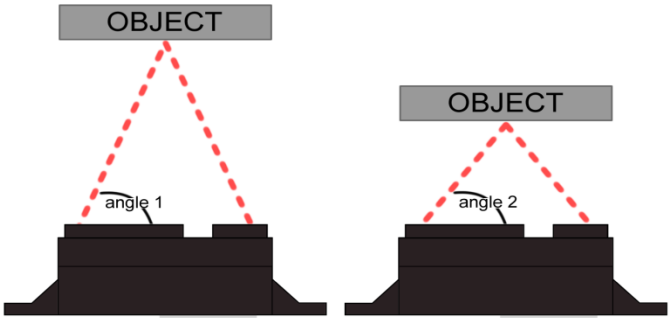


Figure 4.15 Sensor dependence on distance [19]

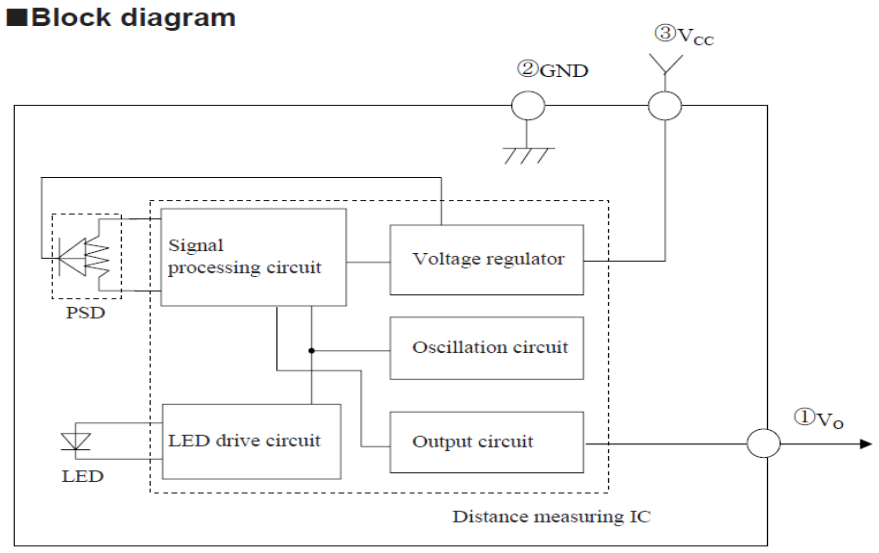


Figure 4.16 Sensor block diagram [19]

4.1.3.6 Warnings

The warning procedure was implemented for The temperature sensor , the Gas and Smoke Sensors and for the battery level indicator.

The first three sensors warn the user when a certain threshold of temperature or Smoke or CO Gas level has been exceeded which may suggest that something has happened in the robot's environment and the user doesn't necessarily have to ask all the time if something is wrong.

I grouped this three warning and described them together because the code for the procedure is the same and only the parameters and values are different.

The microcontroller polls this analog sensors , and when the warning requirements are met , the warning code is sent back to remote .

Trouble is that , as fast as the microcontroller polls the sensors , this fast it sends the warning code to the remote because the time lapse of the program's loop is in the order of µs/ ms (depeding the code) so transmission results to be too often and occupies the channel , disabling other communication.

Not only that ,but something funny happens – if you shut down the robot , the remote still displays incoming warnings for some time, of course there is no magic involved , the explanation is simple : Data fills the Xbee's and Arduino's Serial data buffer and the Remote's program which doesn't loop very fast due to its delays and voice recognition timings keeps reading and displaying the data from the serial buffer until it runs out .

So the solution to this problem is to force the program to send only one warning and no other warnings from that sensor to be issued unless the situation changes . The warning procedure will be re-enabled if the parameter drops or exceeds a certain threshold

We defined functions called Temperature_Warning () , respectively Gas , Smoke_ level where we used a State procedure in order to enable / disable warnings . See Appendix A7 for the sensors warnings code

The battery level circuit also warns the user when the robot's battery has dropped below 30 % , to enable him to call back the machine before it shuts down.

4.1.4 Wireless Communications

Figure 4.17 XBee Module [20]

For establishing a wireless communication between the robot and voice controlled remote , i used a XBee S2 2mW Wire Antenna Zigbee RF Module Fig. 4.17 which is a very small piece of hardware that can connect wirelessly with another using the *Zigbee* communication protocols. There are many different models, including aerial types (Wire Antenna, Chip Antenna , and devices with the possibility of adding an external antenna) as in Fig. 4.18 and power outputs from 1 mW to 100mW which translates to 100 meters to 2 miles outdoor range.
I choose a pair of 2mW Wire Antenna which works on 2.4 Ghz frequency ,due to their antenna's Omni directional characteristic and their acceptable range for our application (120 m)

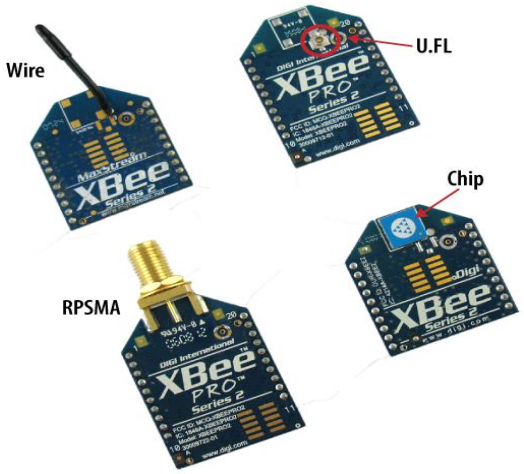


Figure 4.18 Various antenna types XBee modules [3]

ZigBee is a specification for a suite of high level communication protocols using small, low-power digital radios based on the IEEE 802.15.4-2003 standard for wireless home area networks (WHANs), such as, consumer electronics equipment via short-range radio.

The technology defined by the ZigBee specification is intended to be simpler and less expensive than other WPANs, such as Bluetooth. ZigBee is targeted at radio-frequency (RF) applications that require a low data rate, long battery life, and secure networking [21]

XBee network was conceived as a network with multiple nodes. Nodes are clearly defined and can be of three types:

1.Coordinator

2. Router

3. End

In a Xbee network can be a single coordinator node, multiple router type nodes, and the rest can be end devices: sensors, displays, household equipment, etc..

The coordinator is responsible for building the network: scans around the assigned frequencies and occupies a free channel, reserves a network address (PAN ID) and assigns addresses to other nodes, which will be attached to the network

Routers can enter networks using the network address and then deal with routing data received or with attaching End devices .

End device type nodes can hang to the network by connecting to a router.

Here we have the possible networks but for our application which involves also the embedded server gateway it's necessary to configure a cluster tree network composed of Coordinator and two Routers where the robot's transceiver is the coordinator and the remote's Xbee along with servers one are the routers .

But see the potential of using such devices capable of forming a network ! Imagine that networks of robots can be formed that will enable them to share their information like a real team to cover larger areas or to share responsibilities , meanwhile being monitored by a single operator .

Also , a robot can't have on board all the existing sensors , so they can ask data themselves from other sensors mounted on walls or ceilings connected to other Xbee's configured as End-Devices if this system is present in the room / area

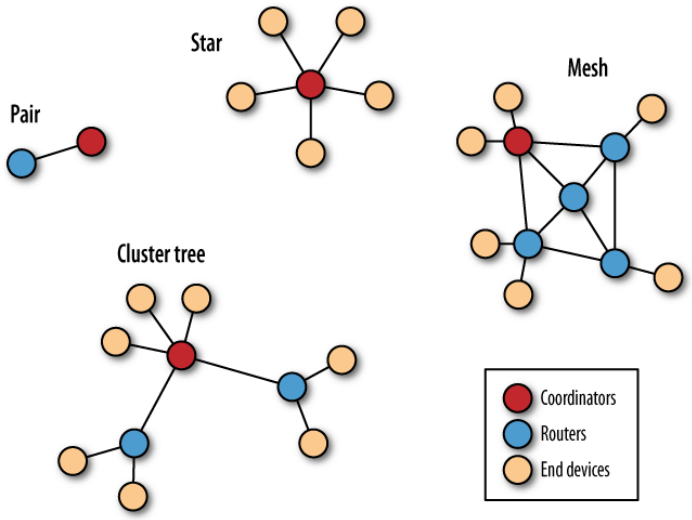


Figure 4.19 Possible Zigbee network topologies [3]

4.1.5 Communication protocol

I will discuss the communication protocol between the robot and the remote because the server behaves the same as the remote , in our case for the two blocks , the Two Xbee's that form a point to point network are a Coordinator and a Router , where the Coordinator is responsible for forming the PAN network , assigning a PAN ID and "occupying " one channel .

Because we have only one channel in our network , both the robot and the remote , must know who , when , and what is transmitting.

The first two are solved very easily :

Who is transmitting is known because , in the first place the user has to give the commands while the robot is listening and than, the remote waits for the feedback while in the same time waiting also for the future spoken commands .

There's no need for us as users to know exactly or to control the moment When one is transmitting , because is the Hardware Serial ports and its built-in interrupt feature job, to flag to the microcontroller when there are Serial data coming on the channel, that's Because in both Arduino and Chipkit IDE we use a High level programming language

But in the third case What is transmitting is up to us as users to establish a Communication Protocol to be sure that commands are not being confused with data , or to be sure that the Remote understands what data did the robot send , because due to it's various number of sensors , the remote must know what to display. (Example : A 24 ° C temperature can be confused with 24 % Battery level if the remote doesn't know what data the robot has sent)

The classic type of communication protocol are strings of several bytes composed of different characters both numbers and letters which have their own signification depending on their position in the string. (E.g. Code = 1A245B341 , where the number one may represent a start / stop bit , and the other characters may signal to the receiver what has been transmitted)

We also decided to make our own encoding of the data , but for the simplicity's sake I came up with the idea to regard the transmitted information as a Modulated Signal (the type is not relevant)

Any Modulated signal has a Carrier signal and the actual information signal Also , it has a limited Bandwidth depending on the Information signal's own bandwidth and on the Bandwidth of the Spectrum itself.

In our case , because the specific function used for Serial communication Serial.write () works by sending one byte and the time , in order to avoid limiting the amount of information which can be sent to the remote i choose to use instead Serial.println() function which sends the ASCII character corresponding to each figure from the number to be sent in a series of bytes followed by a newline character once the number has been sent. So in this way by using a buffer at reception the original number has restored from the ASCII Code corresponding to each of it's figures

We have three types of " information signals " : Commands , which represent a number from 0 to 9 , Warnings from sensors which represent numbers from 10 to 13 and Data from our different kind of sensors , which represent numbers from 0 to 99 maximum

And we considered the " carrier " , a bigger number than the data which is added to the number representing data sent.

If the information sent consist only of digits (0….9) which corresponds to the CO , and Smoke levels sent by this sensors , then the carrier would be in the tens (10…90) , so there is " room " for 9 different carriers in this case , but we used only 2

If the information consist of tens or digits , or both (0….99) , which is the case of the parameters values returned by the sensors – temperature and battery level , than the carrier would be in the hundreds .

For this job we defined a function called Coding (int Carrier , int Value) which has two parameters : the carrier number and the value to be sent , and the code is calculated with the simple formula

Code =Carrier+value;

Sensors / Commands	Value Range	Carrier	Code
Commands & Warnings	0…13	None	The exact values
Smoke_level	0….5	50	50 … 55
CO Gas_level	0…..5	60	60 … 65
Temperature	0.. 99	200	200…299
Battery_level	10…99	100	110 … 199

Table 4.2 Communication protocol codes

4.2 Design Part 2 The Control Unit (Voice Controlled Remote)

For designing a Voice recognition system one must create an acoustic model , a language model for the desired recognition language and a phonetic model. My goal with creating the voice recognition function is replacing the previous Easy VR Shield in the Voice controlled remote for controlling the robot with the Raspberry Pi. So it's goal is to recognize a set of commands for the robot , therefore the voice model has to be created using recordings database containing the commands and their transcripts for the language and phonetic models. The acoustic models for the system described in this paper has been developed on SPEED Laboratories (Speech and Dialogue) BIOSINF Server and the model have be tested with voice commands given by the user on Raspberry Pi

4.2.1 SSH Brief Description

Secure Shell, or **SSH**, is an encrypted network protocol used to remote login in a secure way to other computers in the network , especially with Linux OS

SSH provides a secure channel over an unsecured network in a client-server architecture, connecting an SSH client application with an SSH server

SSH is typically used to log into a remote machine and execute commands, but it also supports tunneling, it can transfer files using the associated SSH file transfer (SFTP) or secure copy (SCP) protocols. SSH uses a client-server model. [41]

SSH connections use the standard TCP port 22 .

4.2.2 SFTP Brief Description

Stands for SSH File Transfer protocol or Secure File Transfer Protocol and is an associated protocol of the previous described one - SSH

Is a network protocol that provides file access, file transfer, and file management over any reliable data stream. This protocol assumes that it is run over a secure channel, such as SSH, that the server has already authenticated the client, and that the identity of the client user is available to the protocol. [41]

SFTP protocol capabilities include besides File transfer a range of operations on remote files which make it more like a remote file system protocol. An SFTP client's extra capabilities include resuming interrupted transfers, directory listings, and remote file removal [41]

4.2.3 BIOSINF Server

In order to connect to UNIX-like systems like the BIOSINF Server or Raspberry Pi , the SSH Protocol is used with a free tool named Putty (Fig. 3.1). The host name or IP address has to entered , in this case the host name is dev.speed.pub.ro , and the SSH port is 12122

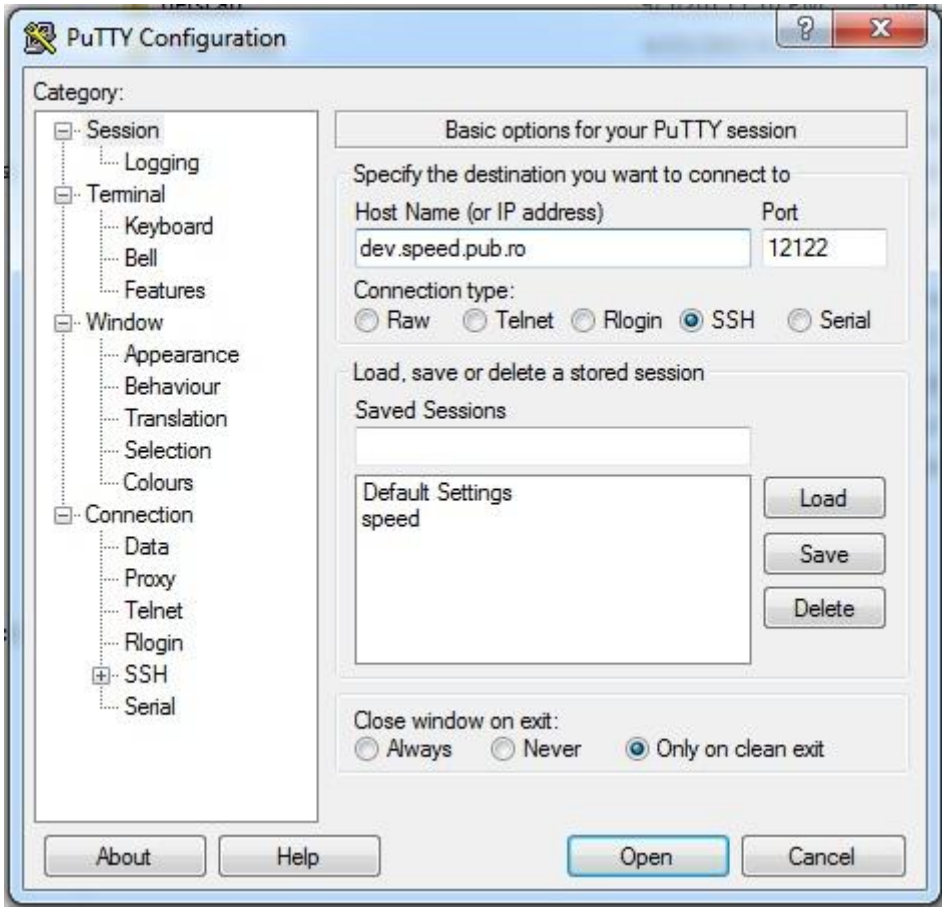


Figure 4.20 Putty configuration for connecting to Server

In figure 4.20 , we have the command line terminal of the BIOSIF Server were the CMU Sphinx project is developed .

In order to copy files to the Server or to Raspberry Pi , a tool named WinSCP (Fig. 4.20) is used to connect to the same address and SSH port using the SFTP protocol

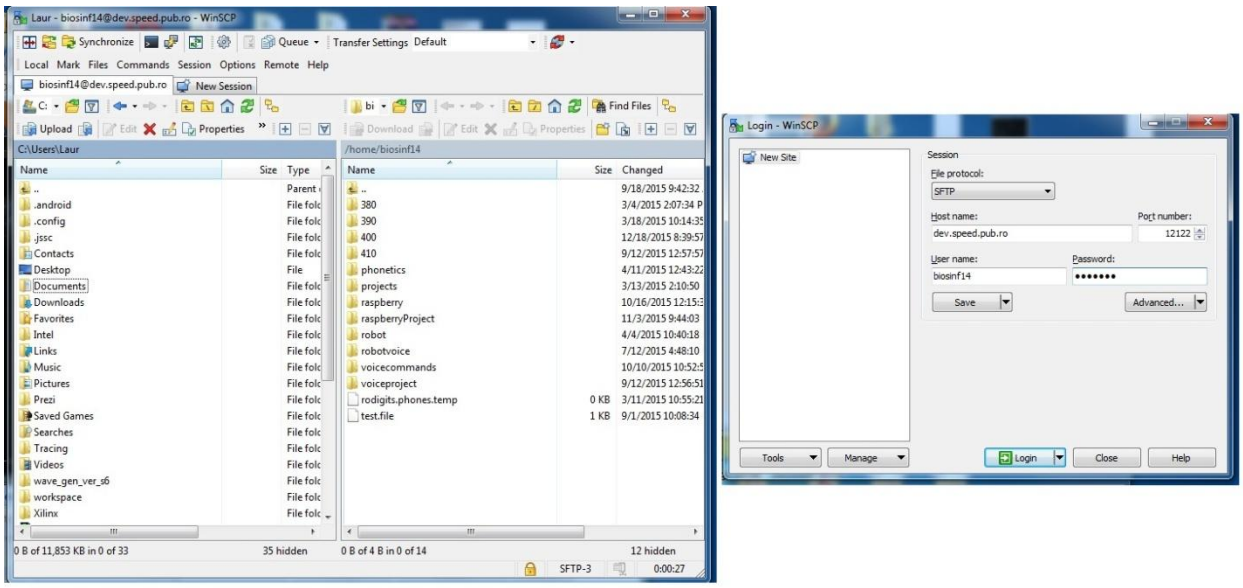


Figure 4.21 WinSCP

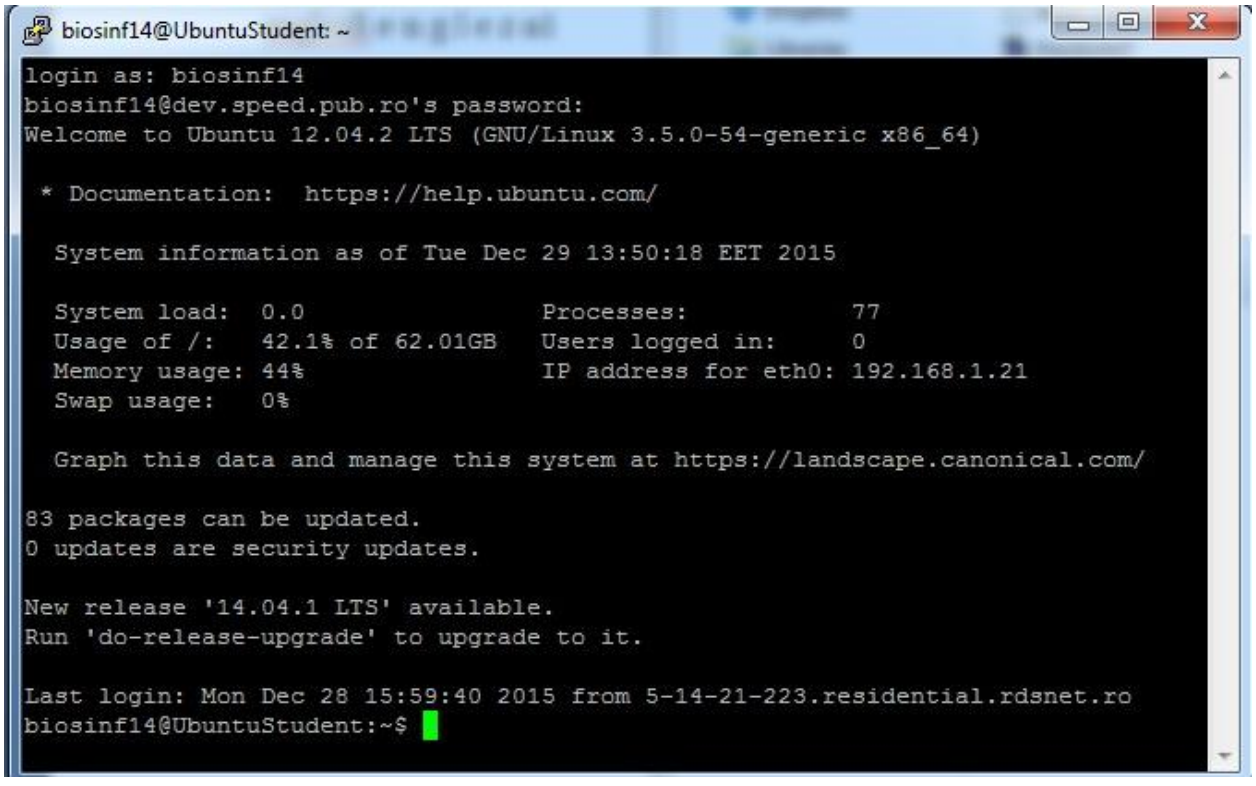
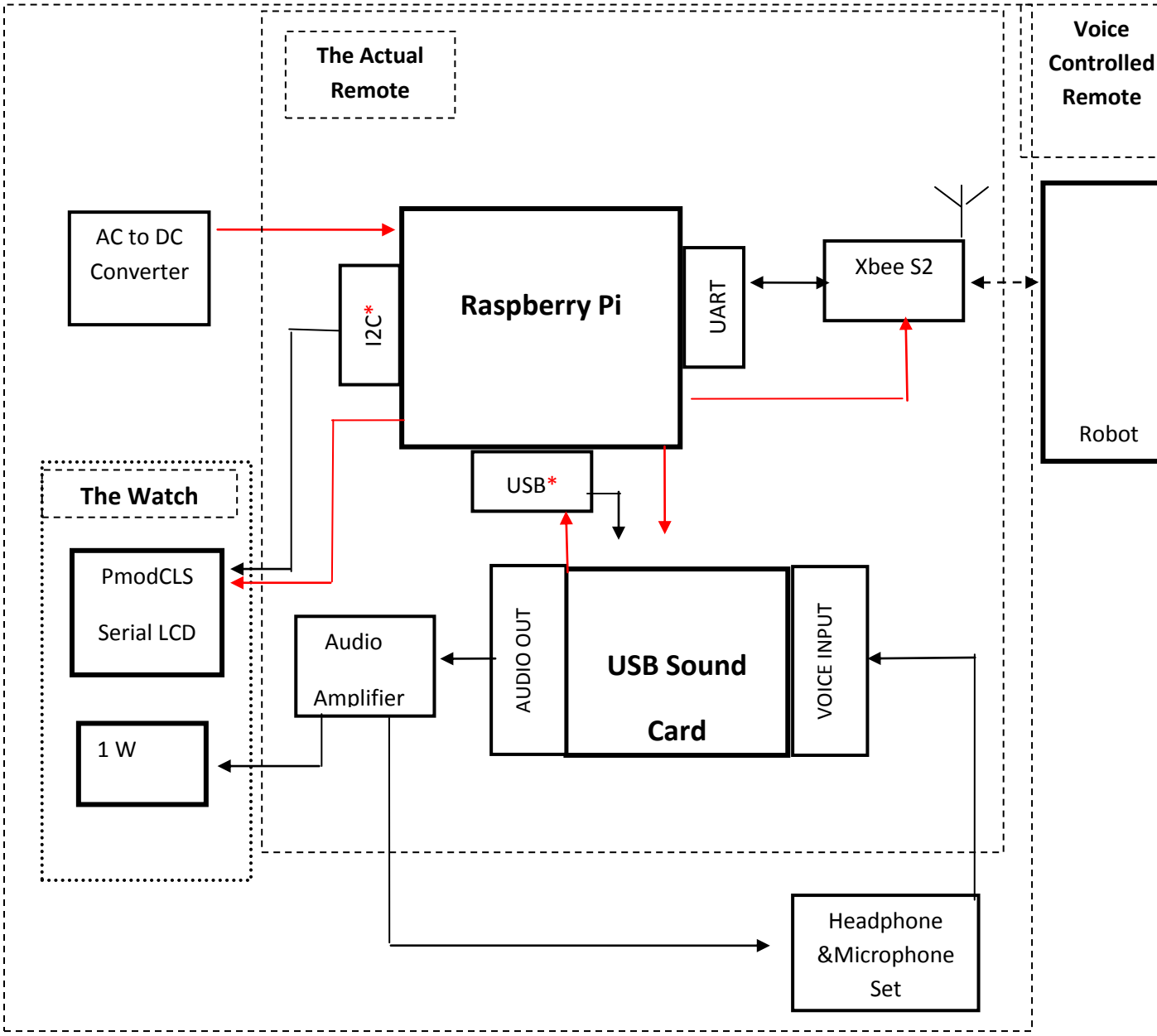


Figure 4.22 BIOSINF Server Command Line

4.2.4 Block Architecture

It's composed of Raspberry Pi pocket computer where voice recognition is performed and has connected to its UART port a Xbee wireless transceiver used for establishing the radio link with the Robot . Has connected to one of its 4 USB ports a Sound Card where the headphone and microphone set is connected. Feedback asked by the user is displayed on The watch which is composed of a Serial LCD connected to Raspberry's I2C port and a Speaker connected to the Audio Amplifier



4.23 Voice controlled remote

4.2.5 Raspberry Pi

Figure 4.24 Raspberry Pi [8]

Is a single board linux computer (Fig 3.4) which i used for running Pocketsphinx continuous which is continuously sampling user voice input for the voice commands defined in the model.

Features :

- ARM Processor, Broadcom SoC running at 700MHz (can be over clocked)
- RAM, 512MB Same power connector, microUSB
- Raspbian linux distribution
- First 40 GPIO pins are the same
- HDMI port
- Ethernet Port
- 4 x USB Ports
- microSD Card Slot where all files are stored [22]

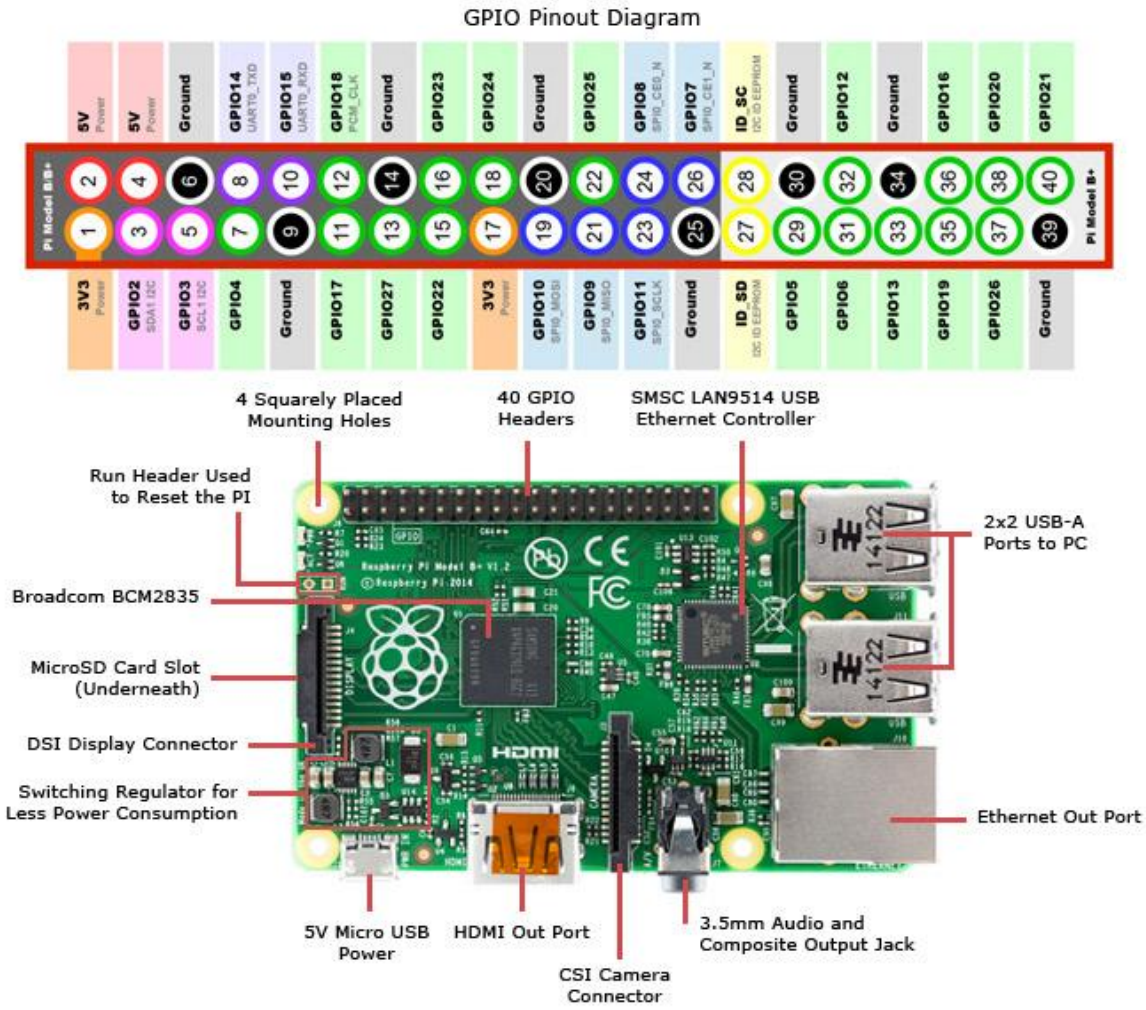


Figure 4.25 Raspberry Pi Pin out diagram and ports [23]

Raspberry Pi pocket computer runs a Linux OS distribution based on Debian , named Raspbian which has a graphical user interface it can be used with any LCD monitor which has a HDMI port (Fig. 4.25), and uses a standard keyboard and mouse just like any other computer or the user can connect to it remotely via the internet by using SSH protocol and use the command line terminal if both the PC and raspberry pi board are connected to the same LAN

In order to use the SSH protocol Putty (Fig. 4.26) is used.

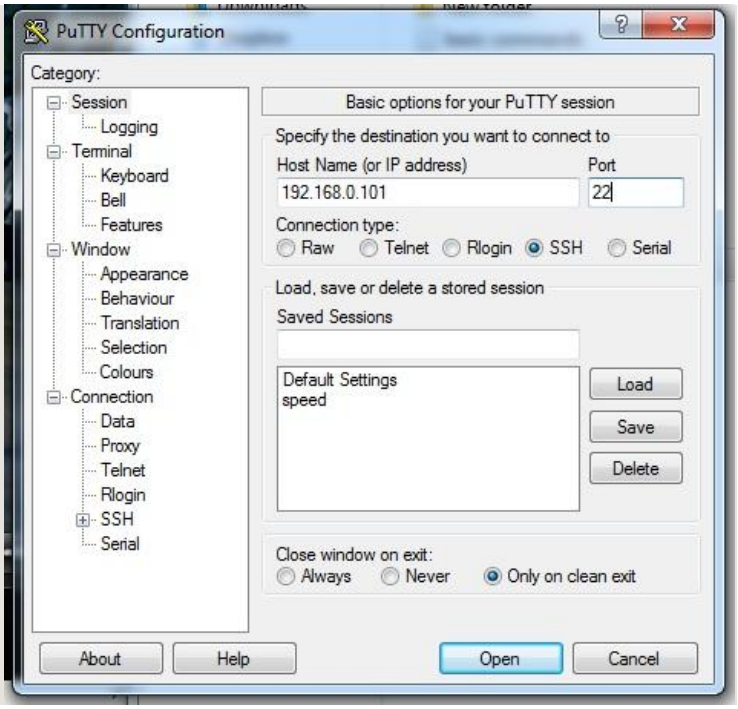


Figure 4.26 Configuration for connecting to Raspberry Pi

Raspberry Pi has to be connected to router by using an Ethernet cable and it will have assigned an IP address available in the LAN by the DHCP server. If we're lucky to use a home network where we have administrator rights , we can connect to the Routers IP address which is 192.168.0.1 and identify Raspberry Pi's assigned address from the DHCP Client List

If we connect to another network , where we don't have access to the gateway's page , we can use a tool called Network scanner (Fig. 4.27) to find IP addresses of all the devices connected in the LAN and connect to the IP address TCP port set as standard for the SSH servers which is 22

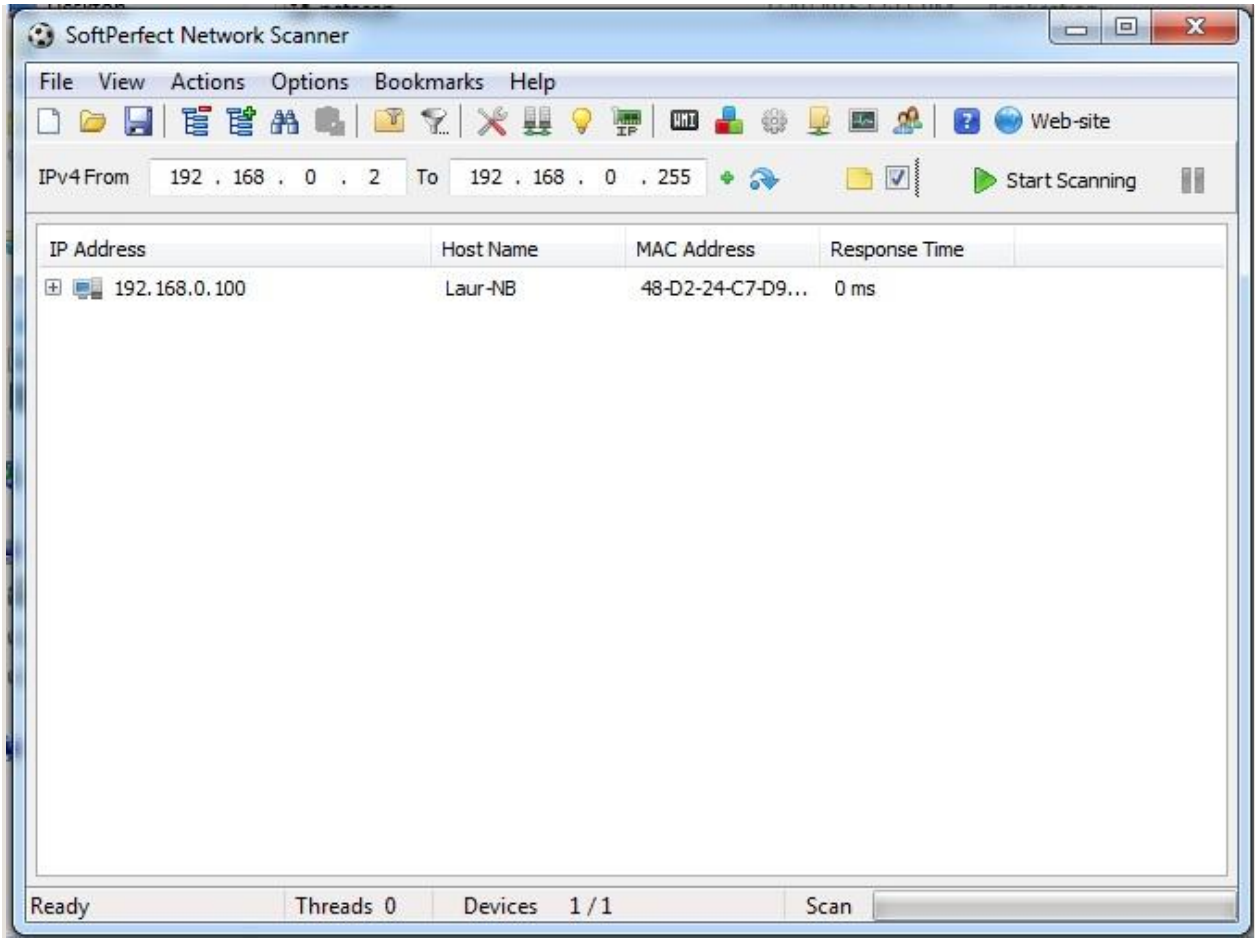


Figure 4.27 Network Scanner

One of the USB ports is used with a Sound Card adaptor where the Headphone and Microphones are connected. and from the GPIO pins (Fig. 4.25) , the UART pins will be used for establishing a radio communication with the robot described in the my previous research activity papers and also the I2C pins for establishing communication with the Special Watch responsible for displaying robot's feedback.

4.2.6 Feedback Display

For mobility considerations the vehicle's user has to have free hands so for receiving the feedback from the robot I choose to design a special kind of watch (Fig. 4.25)

Figure 4.28 The "special" watch

Represents the feedback provider unit , which is responsible for both displaying the status or parameters received from the robot and playing the robot's spoken feedback.

It's composed of two main parts : The LCD Display and the Speaker placed in their own housings connected to each other with a hinge . The whole system is connected to the remote by cable .For displaying the data i choose a serial LCD Display- **PmodCLS - Character LCD w/ serial interface Rev- E**

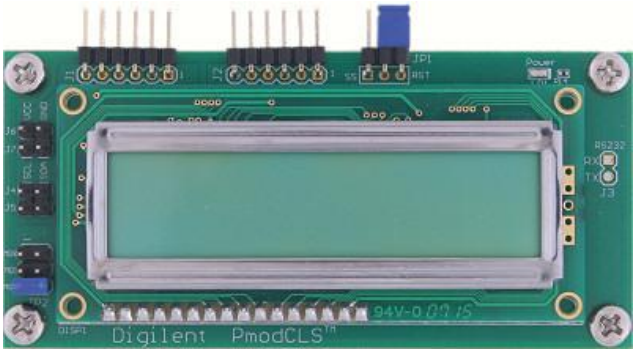


Figure 4.29 Pmod Serial LCD [24]

Serial LCD characteristics include [24] :

- 16x2 Character Display
- Flexible communications using UART, SPI or TWI interface
- Simple terminal-like display interface
- Measures 3.75" x 1.75"

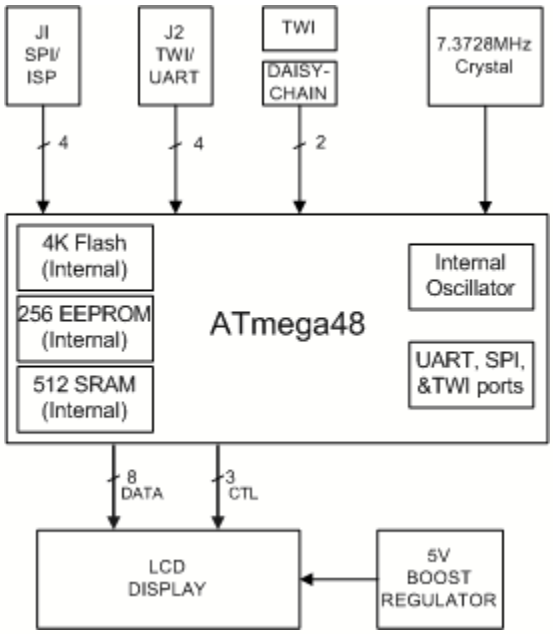


Figure 4.30 Internal LCD Block schematic [24]

First thing to do , is to choose one of the three serial communication options : UART / SPI / TWI (I2C) . In order to do that and also to set the speed of the communication in the UART's case , we have to position some jumpers in the down- left side of the device where we encounter MD2,MD1,MD0 (J2), just like in table 4.6 .

MD2, MD1, MD0	**Protocol**	Details
0,0,0	UART	2400 baud rate
0,0,1	UART	4800 baud rate
0,1,0	UART	9600 baud rate
0,1,1	UART	baud rate in EEPROM
1,0,0	TWI	address: 0x48
1,0,1	TWI	address in EEPROM
1,1,0	SPI	
1,1,1	Specified in EEPROM	Specified in EEPROM

Table 4.3 Serial communication configuration of the LCD

Where the 0 / 1 specified for the three MD jumpers represent a connected jumper , respectively no connection .

I choose the fifth option : I2C communication because the only UART port of the board is already used by XBee. .

4.2.6.1 I2C Introduction. Methods used for I2C communication

I2C stands for Inter-Integrated Circuit and is a synchronous Master-Slave serial bus invented by Philips which solves UART 's one to one communications restrictions and previous agreements over protocol and data transfer rates and SPI's connections required which grows in numbers with any new slave device added. [25]

The serial bus is composed of two line - Serial Clock (SCL)

- Serial Data (SDA) lines with 7-bit addressing.

The bus has two roles for the connected devices: master and slave:

- Master node — node that generates the clock and initiates communication with slaves
- Slave node — node that receives the clock and responds when addressed by the master

The bus is a multi-master bus which means any number of master nodes can be present (Figure 4.28). Additionally, master and slave roles may be changed between messages (after a STOP is sent). [26]

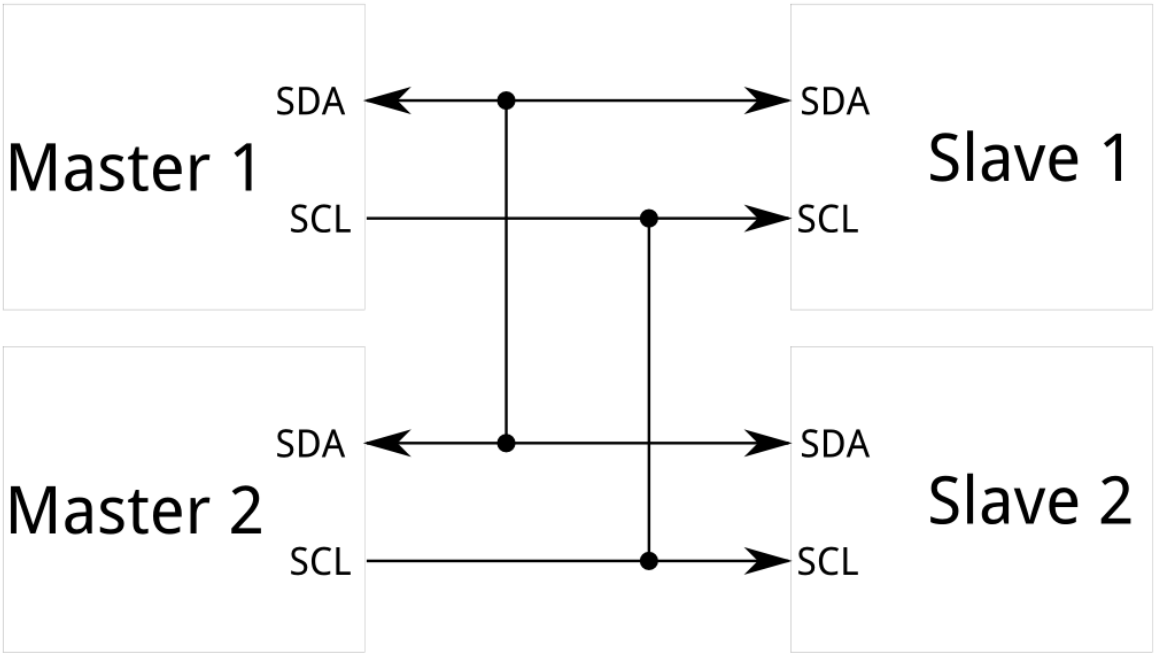


Figure. 4.31 I2C Bus [25]

There may be four potential modes of operation for a given bus device, although most devices only use a single role and its two modes:

- master transmit — master node is sending data to a slave
- master receive — master node is receiving data from a slave
- slave transmit — slave node is sending data to the master
- slave receive — slave node is receiving data from the master

The master is initially in master transmit mode by sending a start bit followed by the 7-bit address of the slave it wishes to communicate with, which is finally followed by a single bit representing whether it wishes to write(0) to or read(1) from the slave.

If the slave exists on the bus then it will respond with an ACK bit (active low for acknowledged) for that address. The master then continues in either transmit or receive mode (according to the read/write bit it sent), and the slave continues in its complementary mode (receive or transmit, respectively). [26]

In our project the Master Device is the RaspberryPi and the Slave Device is the LCD Display and their role doesn't change , so it's only one way communication.

For I2C communications Arduino provides the Wire.h library and their methods are used as well on ArduPi for sending data to slave device are :

Wire.beginTransmission (uint8_t address) - which takes as a parameter the address of the slave device - in our case it's provided in LCD Reference guide - 0x48 [27]

Wire.write(byte data) - to send data byte by byte and Wire.endTransmission

4.2.6.2 Sound reproduction

The Watch has the possibility of playing some feedback recordings when the robot sends the parameter asked by the user or a warning.

I've performed all recordings directly on Raspberry Pi B+ by using a Sennheiser headphone and microphone set with noise-canceling function and a USB Sound card because the pocket computer has only an audio jack for headphones and not its own microphone input .

Sound recording is done with ALSA which stands for Advanced Linux Sound Architecture using the command

arecord -f S16_LE -D plughw:0 -r 16000 test.wav where we need to specify the number of bits of the DAC which converts the sound and format S16_LE stands for Signed 16 bits Little Endian , and the sampling rate - 16 kHz in this case

Sound is played with bash command aplay mySound.wav

The headphone output is connected to the Audio amplifier made with TEA 2025B see schematic in Anexa

4.3 Voice Recognition

4.3.1 Voice recognition basics

The automatic speech recognition (ASR) aims to transform an audio signal containing speech in a sequence of words. The general architecture of a automatic speech recognition system (ASR) is shown in Figure 4.32 . From the figure below two important things regarding the speech recognition can be noticed : a) The voice recognition is done using a number of voice extracted from the voice signal by using the spoken message and b) recognition is based on models : the acoustic , phonetic and linguistic models which have to be developed by the designer. [9]

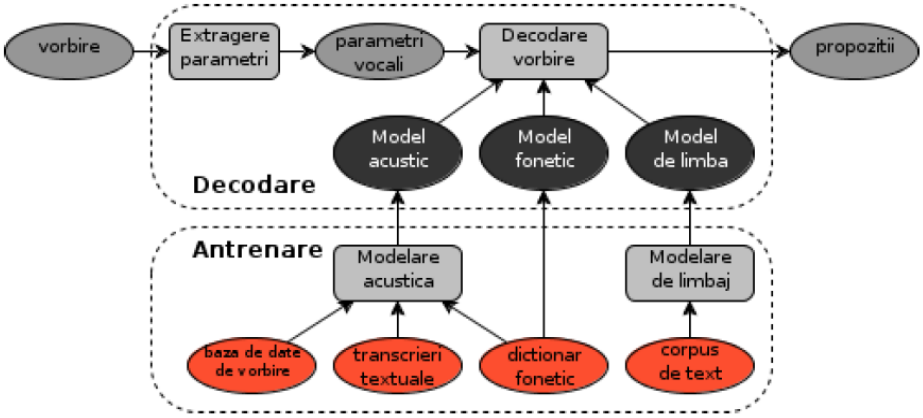


Figure 4.32 ASR Architecture [9]

The acoustic model has the purpose of estimating the probability of a certain spoken message , given a series of words. In voice recognition systems the acoustic model does not use words as base acoustic units. Instead of words, sub-lexical base acoustic units are used which are called phonemes. Therefore , the acoustic model is formed from a set of phonemes connected during the decoding process to form models for words and then models for sequences of words. They are used to estimate the probability that the speaker's spoken message to be made up of a succession of words or another . [9]

The language model is used during decoding process to estimate the probabilities of all sequences of words. In other words , the role of a language model is to estimate the probability that a sequence of words is a valid language sentence . These probabilities help the acoustic model in the decision process . [9]

The phonetic model has the role of connecting the acoustic model (which estimates the acoustic probability of phonemes) with the language model (which estimates the probability of sequences of words) . In other words the phonetic model is a dictionary which makes the association between each word in the vocabulary and it's pronunciation with one or more appropriate phoneme sequences , representing how the word can be pronounced .

Figure 4.32 shows also the processes involved in developing a ASR system of RAV , and also resources needed to create acoustic , language and phonetic models . Acoustic model is built on the basis of a set of recorded audio clips associated with textual transcription of spoken messages and a dictionary that contains all the words in phonetic transcription

4.3.2 ASR System Design

Step 1 : Speech database creation

For speaker dependent voice recognition systems , minimum an 1 h of speech recording is needed to train the acoustic and model . In my pursuit for creating a reliable system for recognizing commands in Romanian , I've recorded many sets of audio files with commands with different microphones , with different command combinations and, both on my laptop using audacity and directly on raspberry pi in order to say the commands in pocketsphinx continuous in conditions similar to those in which they were recorded . For this purpose I've recorded two sets of audio files . In order to copy the audio files to the server , WinSCP Tool is used. Recording conditions , 16 bit samples , 16 kHz

Audio set 1

The audio files include a set of 10 recordings for every command containing only one command repeated multiple times per recording . The command set has 11 commands so this gives a total of 110 recordings.

Audio set 2

This audio files include 100 recordings containing different combinations of the commands in the set because the system should be able to recognize the commands in random order

These recordings were made both on the laptop by using Audacity software and on raspberry Pi ,by using arecord. In linux a sample recording command by using arecord is **arecord -f S16_LE -D plughw:0 -r 16000 test.wav.** The user has to specify directly the parameters

All recordings have to fulfill the following conditions : sample frequency : 16 kHz with 16 bit coded samples and the file has to be stored in a mswav format

Command set

Nr	Command	Description
1	Mergi inainte	Robot motion command
2	Vino inapoi	Robot motion command
3	Roteste stanga	Robot motion command
4	Roteste dreapta	Robot motion command
5	Stop	Robot motion command
6	Masoara Temperatura	Displays temperature
7	Nivel fum	Checks for smoke presence
8	Nivel gaz	Checks for gas presence
9	Nivel Baterie	Measures robot battery level
10	Engleza	Changes command set to english

Table 4.4 Command list

Step2 Creating the phonetic dictionary for the command set

A phonetic dictionary is a language tool that specifies how to pronounce the words of a language. In other words, phonetic dictionary and make correspondence between the written form of a language and the phonetically form . In a continuous speech recognition system phonetic dictionary aims to link the acoustic model (which models the way specific sounds of a language are produced) and the language model (which models the way words succeed in a language) .

mergi m e r g1 i1

înainte i2 n a i n t e

vino v i n o

înapoi i2 n a p o i3

roteşte r o t e s1 t e

stânga s t i2 n g a

dreapta d r e1 a p t a

stop s t o p

măsoară m a1 s o1 a r a1

temperatura t e m p e r a t u r a

nivel n i v e l

fum f u m

gaz g a z

baterie b a t e r i3 e

engleză e n g l e z a1

la l a

revedere r e v e d e r e

Step 3 Training of the acoustic model

Acoustic model training is done in a CMU Sphinx project running on a linux OS on the BIOSINF Server and needs the following resources :

1. The speech database stored in a folder named wav.

2.The transcription of the audio files

3. The phonetic dictionary of both the commands and of other acoustic elements including speech pauses , noise etc.

Steps for creating the project

The project's folder is named **robotcommands** and is configured as sphinxtrain setup directory where the training errors logs and audio files decoding results will be stored by using the command **sphinxtrain_biosinf -t robotcommands setup .** This command add automatically an accoustic model file named **feat.params** , a configuration file named sphinx_train.config and system etc folder which stands for Edit to Configure

where is stored the dictionary , the phones list , a list of the audio files names , and their matching data transcription .

Before transferring the acoustic parameters of the trained model to raspberry pi , it's tested on server with sphinx train's decoder therefore , in etc folder should provide the files used for training and the ones used for testing.

The files used for training are : audio files ids list named **robotcommands.fileids.train** (Ex: 410/410_10_0001 ; 410/410_10_0002 ...) and their matching transcription named

robotcommands.transcription.train which represents the reference text which will be used to generate the language model and that should contain contains the text delimited by <s></s> tags <s> mergi înainte .... mergi înainte </s> (410_10_0001)

The file structure for the database is:

etc :

robotcommands.dic - *Phonetic dictionary*

robotcommands.phones - *Phoneset file*

robotcommands.filler - *List of fillers*

robotcommands.fileids.train - *List of files for training and speaker folder*

robotcommands . transcription .train - *Transcription for training*

robotcommands.fileids.test - *List of files for testing*

robotcommands.transcription.test - *Transcription for testing*

Next Sphinx_train.cfg file (Fig. 4.33) has to be configured in order to set the paths for the dictionary , phones , filler file , fileids , transcriptions and feat parameters

$CFG_DICTIONARY = "$CFG_LIST_DIR/robotcommands.dic";

$CFG_RAWPHONEFILE = "$CFG_LIST_DIR/ robotcommands.phones";

$CFG_FILLERDICT = "$CFG_LIST_DIR/ robotcommands.filler";

$CFG_LISTOFFILES = "$CFG_LIST_DIR/ robotcommands.fileids.train";

$CFG_TRANSCRIPTFILE = "$CFG_LIST_DIR/ robotcommands.transcription.train";

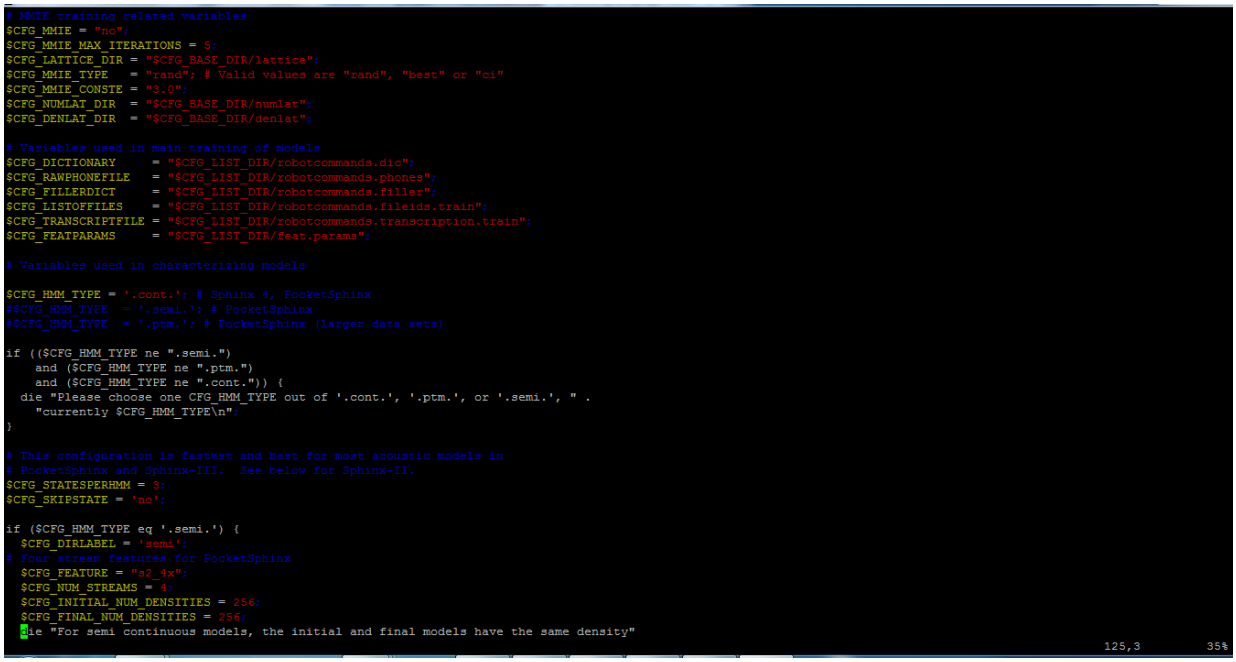


Figure 4.33 Sphinx Train configuration file

Next the files containing the MFCC (Mel Frequency Cepstral Coefficients) corresponding to the audio files have to be created. The Cepstral coefficients characterize the envelope of the audio signal and are used to determine the phoneme succession in the recordings . They are extracted by running the **command usr/local/sphinx/lib/sphinxtrain/scripts/000.comp_feat/slave_feat.pl** on the project folder

Finally , after this intermediate steps the training process of the acoustic model has to be run , by using the command **sphinxtrain_biosinf run.**

Step 4 Creating Language Model (Grammar)

For constructing a Automatic Speech recognition system with a large vocabulary , statistic language models are used because word appearance probability has to be taken into account , some word successions which have a meaning in that language are more probably to appear. On the other hand for controlling a robot only a finite set of commands with equal probability of appearance is used therefore we create a Finite State Grammar.

For this project I used Java Speech Grammar robotcommands . jsgf which contains

#JSGF V1.0;

grammar robotcommands;

public <commands> = (mergi înainte | vino înapoi | roteşte dreapta | roteşte stânga | stop | măsoară temperatura | nivel fum | nivel gaz | nivel baterie | engleză | la revedere) * ;

The first line in the file represents the grammar name and the second line represents a public rule defined , which contains the allowed words , and the unary OR signs indicates that the commands have equal probability of appearance .

Next , by using the tool **sphinx_jsgf2fsg** called by the command **sphinx_jsgf2fsg -jsgf robotcommands.jsgf -fsg robotcommands.fsg** the Java Speech Grammar is transformed in the internal CMU Sphinx Finite State Grammar format

Step 5 Evaluation of the speech recognition system

Becuse the acoustic mode , language model and phonetic model are available , the next stept is to decode the audio files listed in fileids.test file and compare the hypothetical transcription of the audio files resulted after this process with the reference text provided in the transcription.test file

For configuring the decoding process further modifications have to be made in sphinx_train.cfg

$DEC_CFG_SCRIPT = 'psdecode.pl'; has to be modified to look like the following

$DEC_CFG_SCRIPT = 'psdecode_fsg.pl';

Next the lines which specify the name and location of the files used in the decoding process and the name of the directory where the results are saved

$DEC_CFG_DICTIONARY = "$CFG_BASE_DIR/etc/rodigits.dic";
$DEC_CFG_FILLERDICT = "$CFG_BASE_DIR/etc/rodigits.filler";
$DEC_CFG_LISTOFFILES = "$CFG_BASE_DIR/etc/rodigits.fileids.test";
$DEC_CFG_TRANSCRIPTFILE = "$CFG_BASE_DIR/etc/rodigits.transcription.test";
$DEC_CFG_RESULT_DIR = "$CFG_BASE_DIR/result.cd_cont_200_8";

The lines which specifies the language model has to be modified from standard DMP model to FSG language model which we have .

Identify specific application line then they will align with the text resulting from the reference text recognition (to assess the rate of error in the word) :

// pictures decode

$DEC_CFG_ALIGN = "builtin";

$DEC_CFG_ALIGN = "sclite";

Next the Cepstral coefficients for the test audio files have to be created by using the same command

/usr/local/sphinx/lib/sphinxtrain/scripts/000.comp_feat/slave_feat.pl.

De evaluation process is started by running the command
/usr/local/sphinx/lib/sphinxtrain/scripts/decode/slave.pl in the project folder

The goal of running this script are the Sentence Error Rate and Word Error Rate obtained after aligning the two audio files transcriptions by using sclite that we set up in the configuration file . Sclite is a tool for scoring and evaluating the output of speech recognition by comparing the hypothesized text (HYP i.e output of decoder) output by the speech recognizer to the correct, or reference (REF i.e transcription) text. By analyzing robotcomands.align ?? in result_cd_cont_200 folder we can also figure out what commands are systematically not recognizing or mistaken with others

The results are quantified by defining a WER and SER

4.3.3 Viewing and interpreting the decoding results

The evaluation of a speech recognition system is done automatically by comparing textual transcripts of the two audio evaluation files : the reference textual transcription and the hypothetical textual transcription (resulted from the decoding process). Sentence by sentence analysis is done in two stages: a) hypothetical reference phrase and the reference sentence are aligned by an algorithm that aims to minimize the number of transcription errors, then b) count all word recognition errors (inserted words , substituted or deleted words). Distinguished as two standard performance criteria used to evaluate continuous speech recognition systems : Sentence Error Rate (SER) and Word Error Rate (WER). SER is calculated as the ratio between the number of correct sentences (sentences with no word errors) and total sentences. The word error rate is calculated from word level by taking into account the deletions and insertions and substitutions

By following these steps i made 4 experiments by training 4 acoustic models with 4 different audio sets and testing there recognition accuracy by using different test audio files sets .I'll present each ones results (Table 4.5)

Nr	Training database	Language model	Testing database	WER
1	PClow-train	Components grammar	PClow-eval	6.3 %
		Commands grammar	PClow-eval	11,11%
			PClow-eval + PChigh#1-eval	28%
2	PChigh#1-train	Commands grammar	PChigh#1-eval	10.80%
			PClow-eval + PChigh#1-eval	25.3
			PClow-eval + PChigh#1-eval + PChigh#2-eval	33.30%
3	PChigh#1-train + PChigh#2-train	Commands grammar	PChigh#1-eval + PChigh#2-eval	3.80%
			PClow-eval + PChigh#1-eval + PChigh#2-eval	9.50%
4	Rasp-train	Commands grammar	Rasp-eval	13.3 %

Table 4.5 Decoding results for each acoustic model

The description of the audio file sets is given in the table below

Audio set name	Audio Recordings	Training audio set	Test audio files	Recording Conditions
PClow	100 recordings containing each command repeated 7 times . 10 recordings / command	80	20	Medium Quality microphone. Recordings on PC in Audacity . Noisy recordings
PChigh#1	110 recordings containing each command repeated 9 times . 10 recordings / command	90	20	High Quality Sennheiser microphone with noise cancelling function , Recordings on PC in Audacity
PChigh#2	110 Recordings containing combinations of all commands	90	10	PChigh#1 likewise
Rasp	110 recordings containing each command repeated 9 times . 10 recordings / command	90	20	Sennheiser Microphone .Recordings directly on Raspberry Pi , which adds background noise

Table 4.6 Audio set description

Obs : PClow / high - stands for the recordings condtions . PC -the set was recorded on the PC , and low /high - stands for low / high quality recordings.

Grammar

The commands grammar has been introduces at Step 4 Creating the Grammar of the ASR Design

Components Grammar

#JSGF V1.0;

grammar robotcommands;

public <commands> = (mergi | înainte | vino | înapoi | roteşte | dreapta | stânga | stop | măsoară |temperatura | nivel | fum | gaz | baterie | engleză) * ;

It's the grammar that i used for the first experiment it's a less restrictive one and gives equal probability of appearance to the commands components words.

Experiment 1

In my first experiment I used The PClow recordings set and Components grammar , and after decoding the test audio files i obtained a WER = 6.3 % . For every experiment , after obtaining a WER , i moved forward on testing the training acoustic model on Raspberry Pi using Pocketsphinx continuous .

I realized that the components grammar has a major drawback , it allows illegal combinations such as " roteste temperatura " which doesn't make any sense , so i decided to try next with the commands grammar.

The commands grammar gives equal probability of appearance to the commands. If the machine recognizes for example " mergi " , the other commands component : " inainte " has 100% probability of appearance after the first word therefore the commands are recognized more easily .

After decoding the Test audio files i obtained WER = 11.11 %

Next I added to the fileids.test PChigh1 - test audio set. After decoding the Test audio files i obtained WER =28 %

I decided to create a new acoustic model trained with the PChigh1-train database

Experiment 2

Recordings database contains 110 recordings with one command per recording repeated multiple times. The recordings were done on a PC by using Audacity and a noise cancelling microphone.

In figure 4.35 we have the detailed results of the decoding process for each speaker. For this experiment I used was PChigh#1 -test audio file set containing a total of 20 recordings and a **WER = 10.8 %**

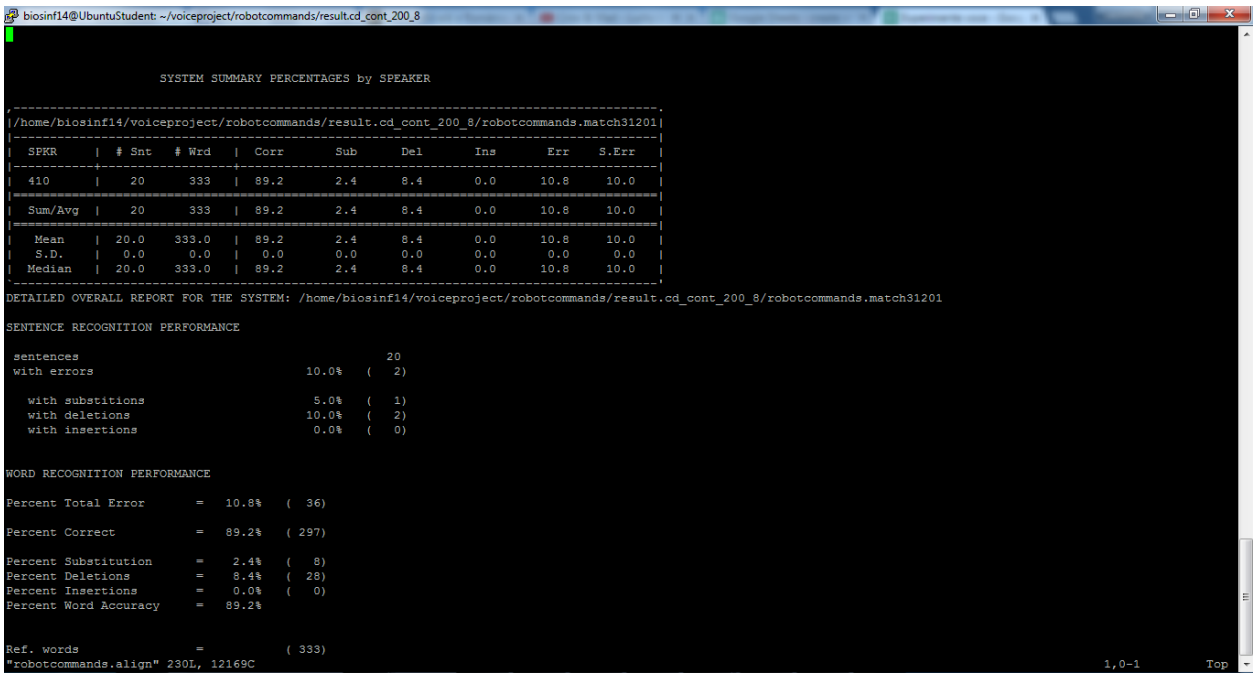


Figure 4.34 Experiment 2 test 1 Results

In figure 4.35 , we have an example of the Hypothesis and reference texts and the commands which were recognized good and those who weren't . In this experiment only one command has systematically not recognized . But as it will be noticed when using the model created with this steps with pocketsphinx continuous on raspberry pi , the accuracy is much more affected and only a couple of commands are recognized

```
              DUMP OF SYSTEM ALIGNMENT STRUCTURE

ystem name:   /home/biosinf14/voiceproject/robotcommands/result.cd_cont_200_8/robotcommands.match3518

peakers:
   0:  410

peaker sentences   0:  410   #utts: 16
d: (410-410_10_0055)
cores: (#C #S #D #I) 0 0 18 0
EF:  LA REVEDERE LA REVEDERE LA REVEDERE LA REVEDERE LA REVEDERE LA REVEDERE LA REVEDERE LA REVEDERE
YP:  ** ******** ** ******** ** ******** ** ******** ** ******** ** ******** ** ******** ** ******** ** ********
val: D  D        D  D        D  D        D  D        D  D        D  D        D  D        D  D        D  D

d: (410-410_10_0056)
cores: (#C #S #D #I) 18 0 0 0
EF:  mergi înainte mergi înainte mergi înainte mergi înainte mergi înainte mergi înainte mergi înainte mergi înainte mergi înainte
YP:  mergi înainte mergi înainte mergi înainte mergi înainte mergi înainte mergi înainte mergi înainte mergi înainte mergi înainte
val:

d: (410-410_10_0057)
cores: (#C #S #D #I) 18 0 0 0
EF:  vino înapoi vino înapoi vino înapoi vino înapoi vino înapoi vino înapoi vino înapoi vino înapoi vino înapoi
YP:  vino înapoi vino înapoi vino înapoi vino înapoi vino înapoi vino înapoi vino înapoi vino înapoi vino înapoi
```

Figure 4.35 Experiment 2 test 1Reference & Hypothesis words aligned

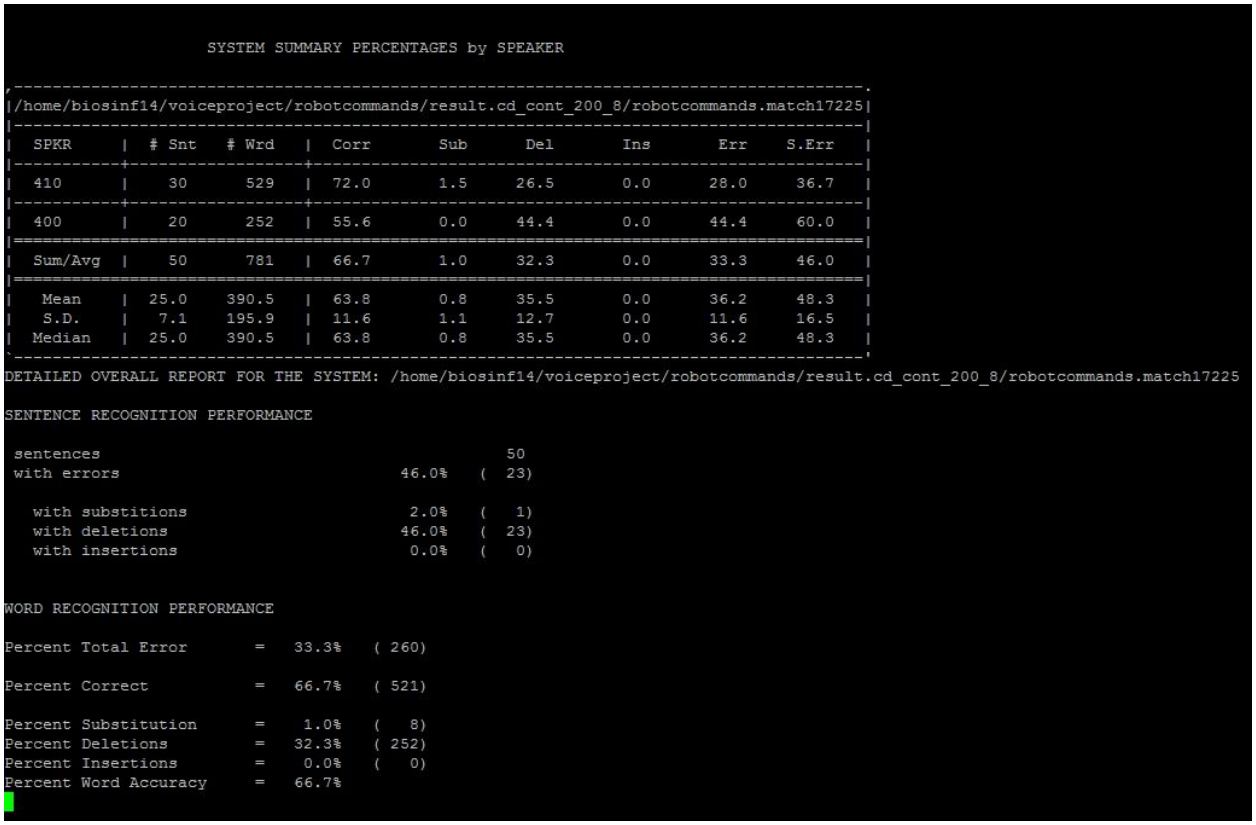
Next I decided to test the system also with noisy recording by adding PClow-test recording set. (Fig. 4.36)

```
biosinf14@UbuntuStudent: ~/voiceproject/robotcommands

                    SYSTEM SUMMARY PERCENTAGES by SPEAKER

,-------------------------------------------------------------------------------.
|/home/biosinf14/voiceproject/robotcommands/result.cd_cont_200_8/robotcommands.match30958|
|-------------------------------------------------------------------------------|
|  SPKR     | # Snt   # Wrd   |  Corr       Sub      Del       Ins       Err    S.Err   |
|-----------+-----------------+-------------------------------------------------|
|  410      |    20      333  |  89.2       2.4      8.4       0.0      10.8     10.0   |
|-----------+-----------------+-------------------------------------------------|
|  400      |    20      252  |  55.6       0.0     44.4       0.0      44.4     60.0   |
|===============================================================================|
|  Sum/Avg  |    40      585  |  74.7       1.4     23.9       0.0      25.3     35.0   |
|===============================================================================|
|   Mean    | 20.0    292.5  |  72.4       1.2     26.4       0.0      27.6     35.0   |
|   S.D.    |  0.0     57.3  |  23.8       1.7     25.5       0.0      23.8     35.4   |
|  Median   | 20.0    292.5  |  72.4       1.2     26.4       0.0      27.6     35.0   |
`-------------------------------------------------------------------------------'
DETAILED OVERALL REPORT FOR THE SYSTEM: /home/biosinf14/voiceproject/robotcommands/result.cd_cont_200_8/robotcommands.match30958

SENTENCE RECOGNITION PERFORMANCE

 sentences                                 40
 with errors                    35.0%   (  14)

   with substitions              2.5%   (   1)
   with deletions               35.0%   (  14)
   with insertions               0.0%   (   0)


WORD RECOGNITION PERFORMANCE

Percent Total Error       =   25.3%   ( 148)

Percent Correct           =   74.7%   ( 437)

Percent Substitution      =    1.4%   (   8)
Percent Deletions         =   23.9%   ( 140)
Percent Insertions        =    0.0%   (   0)
Percent Word Accuracy     =   74.7%

                                                                          1,0-1         Top
```

Figure 4.36 Experiment 2 test 2 Results

Because in a real life scenarios , from the user the system receives command in random order , i decided to test it with PChigh#2-test recording set.

The test audio file set used has 50 audio files and a 780 words. In figure 4.37 , we can see that SER is 46 % , WER = 33,3 % . And if we analyze figure 4.38 we can see that in the recordings corresponding to PChigh#2 test set which contain combinations of commands are not recognized systematically .



```
                    SYSTEM SUMMARY PERCENTAGES by SPEAKER

,-------------------------------------------------------------------------------.
|/home/biosinf14/voiceproject/robotcommands/result.cd_cont_200_8/robotcommands.match17225|
|-------------------------------------------------------------------------------|
|  SPKR     | # Snt   # Wrd  |  Corr      Sub      Del      Ins      Err    S.Err  |
|-----------+----------------+-------------------------------------------------|
|  410      |    30      529 |  72.0      1.5     26.5      0.0     28.0     36.7  |
|-----------+----------------+-------------------------------------------------|
|  400      |    20      252 |  55.6      0.0     44.4      0.0     44.4     60.0  |
|===============================================================================|
|  Sum/Avg  |    50      781 |  66.7      1.0     32.3      0.0     33.3     46.0  |
|===============================================================================|
|   Mean    |  25.0    390.5 |  63.8      0.8     35.5      0.0     36.2     48.3  |
|   S.D.    |   7.1    195.9 |  11.6      1.1     12.7      0.0     11.6     16.5  |
|  Median   |  25.0    390.5 |  63.8      0.8     35.5      0.0     36.2     48.3  |
`-------------------------------------------------------------------------------'
DETAILED OVERALL REPORT FOR THE SYSTEM: /home/biosinf14/voiceproject/robotcommands/result.cd_cont_200_8/robotcommands.match17225

SENTENCE RECOGNITION PERFORMANCE

 sentences                                 50
 with errors                         46.0%   (  23)

   with substitions                   2.0%   (   1)
   with deletions                    46.0%   (  23)
   with insertions                    0.0%   (   0)


WORD RECOGNITION PERFORMANCE

Percent Total Error       =   33.3%   ( 260)

Percent Correct           =   66.7%   ( 521)

Percent Substitution      =    1.0%   (   8)
Percent Deletions         =   32.3%   ( 252)
Percent Insertions        =    0.0%   (   0)
Percent Word Accuracy     =   66.7%
```

Figure 4.37 Experiment 2 test 3 Results

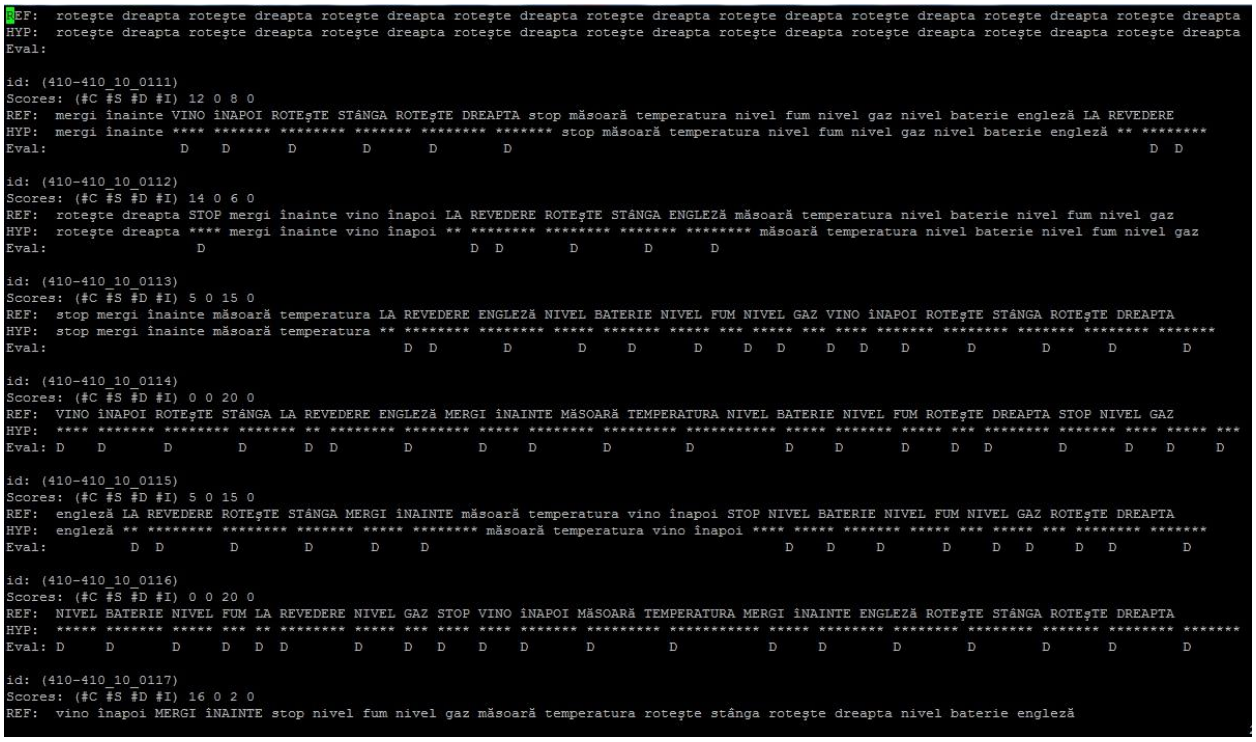


Figure 4.38 Experiment 2 test 3 Reference & Hypothesis words aligned

The Hypothesis of the recordings contains only deletions . So we can conclude that with the acoustic model trained by audio files containing only one command per recording we don't obtain good results with random combinations of commands. This leads us to our next experiment , where I decided to add audio files containing combinations of commands

Experiment 3

In the third experiment , in addition to the previous 110 recordings PChigh#1-train , I've added another 100 recordings containing combinations of all commands in each recording PChigh#2 - train . This fits to a real testing scenarios , because the system should work with commands given by the user in a random order.

For this experiment , i used 30 test sentences containing a total of 529 words and I obtained the best WER of al 3 experiments , WER = 3.8 % (Fig. 4.39)

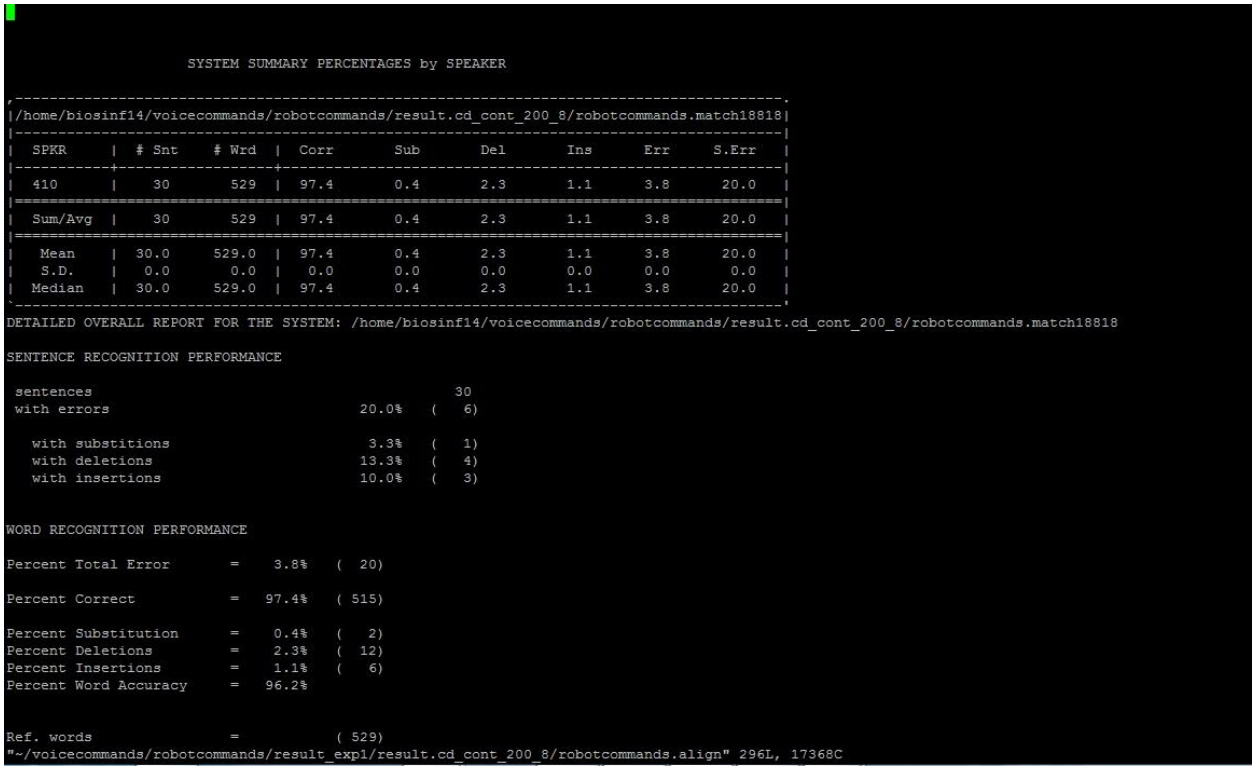


Figure 4.39 Experiment 3 Test 1 Results

Next i added PClow-test to the test audio file set .in order to check the system's recognition accuracy with noisy recordings. The test set has 50 recordings and a total of 781 words

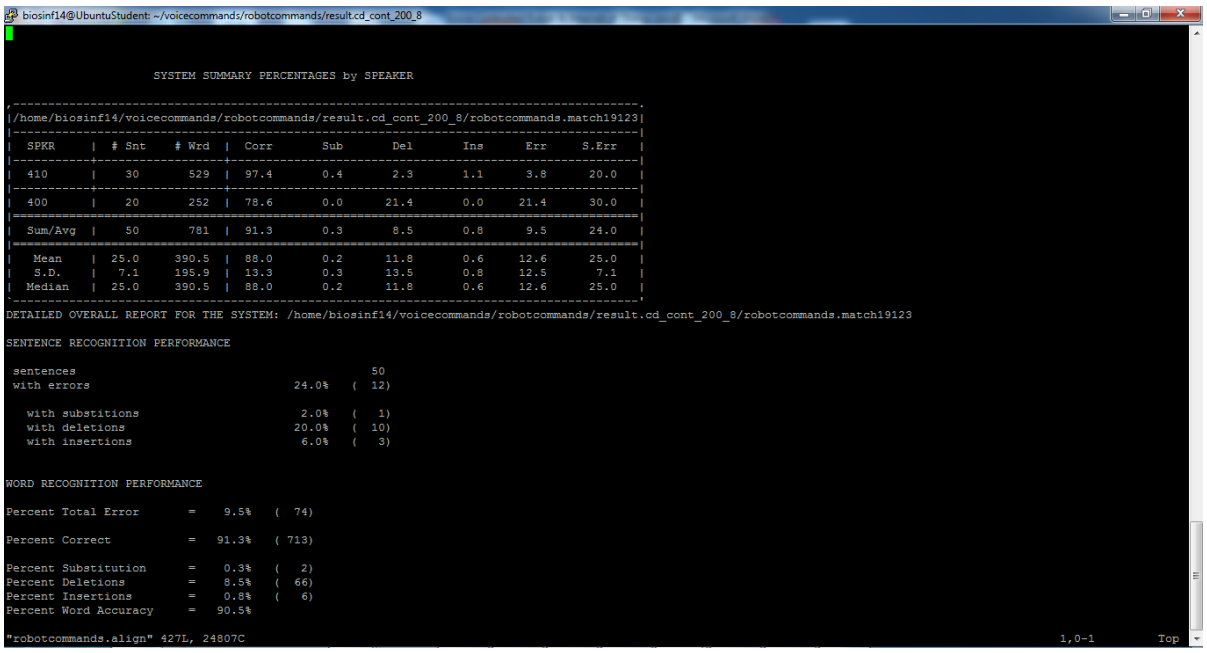


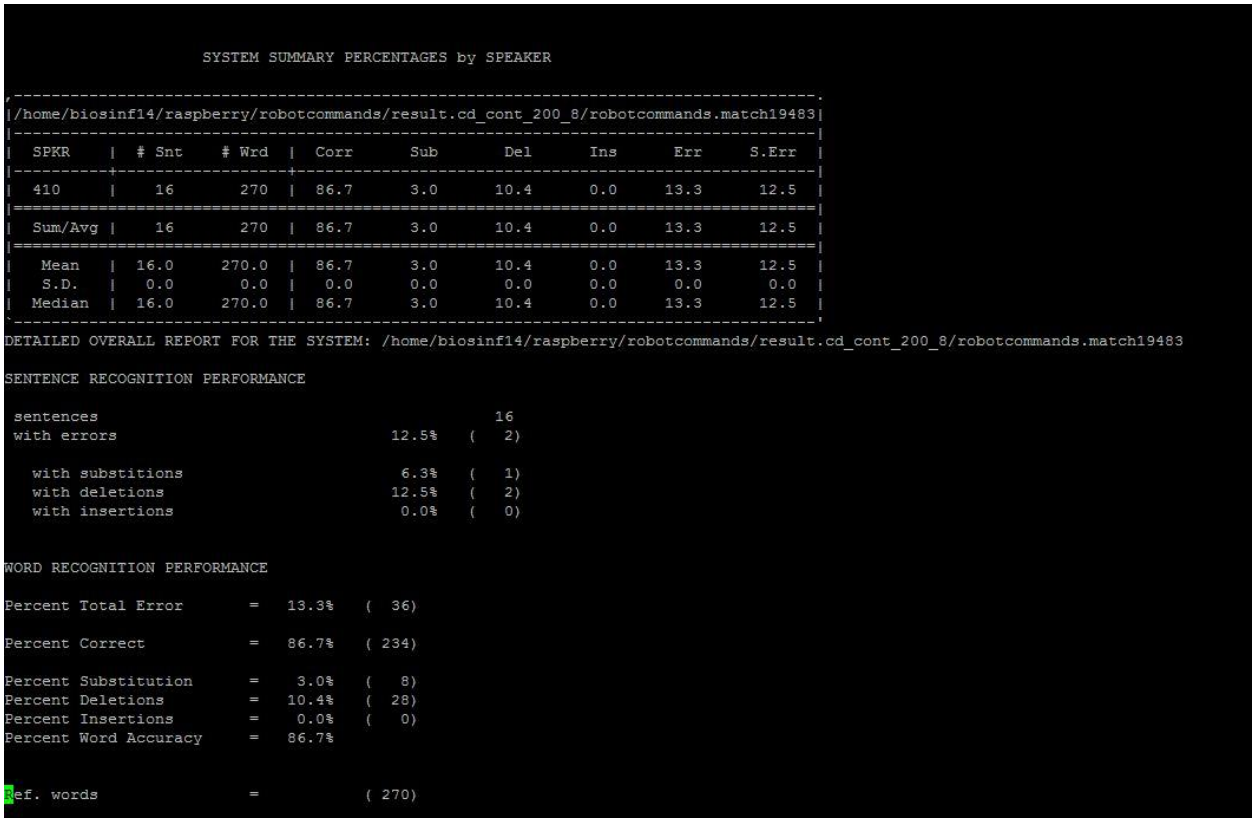
Figure 4.40 Experiement 3 Test 2 results

As we can see in the figure above , we obtained a WER = 9.5 % , a result which can be compared with the results obtained in the previous experiment with the same test set which had a WER = 33,3 %.

By analyzing the two results we can conclude that the best acoustic model for testing on Raspberry PI is the one trained in the third experiment

Experiment 4

Similar to experiment 2 , but the recording had been done directly on raspberry Pi in order to say the commands in conditions similar to those in which they were recorded . The audio files are noisier than those recorded on the PC because Raspberry Pi doesn't have an audio card of its own and a USB Audio Sound Card had to be used . WER =13.3 %



```
                SYSTEM SUMMARY PERCENTAGES by SPEAKER

,-------------------------------------------------------------------------------.
|/home/biosinf14/raspberry/robotcommands/result.cd_cont_200_8/robotcommands.match19483|
|-------------------------------------------------------------------------------|
|  SPKR    |  # Snt    # Wrd  |  Corr      Sub       Del      Ins      Err     S.Err  |
|----------+-----------------+-------------------------------------------------|
|  410     |    16       270  |  86.7      3.0      10.4      0.0     13.3      12.5  |
|===============================================================================|
|  Sum/Avg |    16       270  |  86.7      3.0      10.4      0.0     13.3      12.5  |
|===============================================================================|
|   Mean   |  16.0     270.0  |  86.7      3.0      10.4      0.0     13.3      12.5  |
|   S.D.   |   0.0       0.0  |   0.0      0.0       0.0      0.0      0.0       0.0  |
|  Median  |  16.0     270.0  |  86.7      3.0      10.4      0.0     13.3      12.5  |
`-------------------------------------------------------------------------------'
DETAILED OVERALL REPORT FOR THE SYSTEM: /home/biosinf14/raspberry/robotcommands/result.cd_cont_200_8/robotcommands.match19483

SENTENCE RECOGNITION PERFORMANCE

 sentences                                16
 with errors                        12.5%   (   2)

   with substitions                  6.3%   (   1)
   with deletions                   12.5%   (   2)
   with insertions                   0.0%   (   0)


WORD RECOGNITION PERFORMANCE

Percent Total Error       =   13.3%   (  36)

Percent Correct           =   86.7%   ( 234)

Percent Substitution      =    3.0%   (   8)
Percent Deletions         =   10.4%   (  28)
Percent Insertions        =    0.0%   (   0)
Percent Word Accuracy     =   86.7%


Ref. words                =           ( 270)
```

Figure 4.41 Experiment 4 Results

4.3.4 Testing the model using Pocketsphinx continuous

In order to get the program running , the folder robotcommands.cd_cont200 containing the training results has to be copied to the Raspberry Pi by using the WinSCP tool , the acoustic model has to be copied to the hidden markov models folder (hmm) in a folder named after the language used " ro ". The phonetic dictionary and grammar have to be copied to the language model folder named lm , also in a folder named "ro ".

Pocketsphinx continuous is launched with the following command :

./pocketsphinx-0.8/src/programs/pocketsphinx_continuous -fsg ~/pocketsphinx-0.8/model/robot/robotcommands.fsg -dict ~/pocketsphinx-0.8/model/robot/robotcommands.dic -hmm ~/pocketsphinx-0.8/model/hmm/ro/robotcommands.cd_cont_200 -silprob 0.1 -wip 1e-4 -bestpath 0

With ./ in linux the executables are called , in this case pocketsphinx_continuous preceded by it's folder path , also the type of grammar (finite state grammar or fsg) and the grammar , acoustic model and phonetic dictionary paths are mentioned.

I've tested each one of the three models , and realized that when sampling user input commands the recognition accuracy is seriously affected . In order to evaluate how the recognition system works I made a subjective Word Error Rate evaluation by repeating every command 10 times measured in failures / total

Experiment 2 Acoustic Model

Nr	Command	WER
1	Mergi inainte	2/10
2	Vino inapoi	10/10
3	Roteste stanga	3/10
4	Roteste dreapta	8/10
5	Stop	6/10
6	Masoara Temperatura	10/10
7	Nivel fum	2/10
8	Nivel gaz	1/10
9	Nivel Baterie	10/10
10	Engleza	8/10
11	Mix of commands	7/10

Table 4.7 Experiment 2 Subjective WER

By comparing the decoding results made by sphinx train and the the ones made by pocketsphinx continuous , it can be notices that in the latter case only a couple of commands are recognized properly : " Mergi inaine " , " Nivel fum " , "Nivel gaz " , " Roteste stanga " , some are not recognized at all.

Experiment 3 Acoustic Model

Nr	Command	WER
1	Mergi inainte	10/10
2	Vino inapoi	5/10
3	Roteste stanga	7/10
4	Roteste dreapta	10/10
5	Stop	2/10
6	Masoara Temperatura	8/10
7	Nivel fum	5/10
8	Nivel gaz	0/10
9	Nivel Baterie	10/10
10	Engleza	5/10
11	Mix of commands	6/10

Table 4.8 Experiment 3 Subjective WER

Experiment 4 Acoustic Model

Nr	Command	WER
1	Mergi inainte	5/10
2	Vino inapoi	10/10
3	Roteste stanga	7/10
4	Roteste dreapta	9/10
5	Stop	5/10
6	Masoara Temperatura	7/10
7	Nivel fum	6/10
8	Nivel gaz	0/10
9	Nivel Baterie	10/10
10	Engleza	9/10
11	Mix of commands	7/10

Table 4.9 Experiment 4 Subjective WER

4.3.5 Improvements

Because of the different results obtained after decoding the audio commands using CMU SPhinx Decoder and by Using PocketSphinx Continuous (see table 4.10) i started analyzing a reliable improvement to be made for the system's voice recognition accuracy. I decided to focus on improvements the models created with experiment 2 and 3

Experiment	CMU Sphinx Decode	PocketSphinx Continuous
2	33.3%	70 %
3	9.5 %	60%

Table 4.10 Comparison between Server and Raspberry Pi decoding results

I've compared the decoding parameters of both the decoder and of pockesphinx continuous , and i found that they were not using the same default parameters , The parameters listed in the table below had different default values .

Parametrii	Valori Default Server	Valori Default Raspberry	Valori curente Server	Valori curente Raspberry
beam	1.00E-48	1.00E-48	1.00E-80	1.00E-48
bestpath	yes	yes	yes	no
lw	6.5	6.5	1.00E+01	6.5
wbeam	7.00E-29	7.00E-29	1.00E-40	7.00E-29
silprob	0.05	0.005	5.00E-03	1.00E-01
wip	0.65	0.65	2.00E-02	1.00E-04

Table 4.11 Comparison between Server and Raspberry configuration parameters

Where wip stands for Word Insertion Probability and represents probability of inserting a word that is a close match to the hypothetic word recognized from the audio command , in other words the system decides on choosing a command from the grammar even if it doesn't match 100 % with the one in the audio file

silprob - Silence probe

From table 4.11 we can notice that WIP on the Decoder is 200 times larger than WIP configured on pockesphinx continuous . If a small WIP is configured , the system most likely will conclude that the commands given by the user can't be found in the grammar if it doesn't match 100 % with the one trained , which explained the big WER obtained on pocketsphinx continuos

So as an improvement i decided to use the same parameters on both systems in order to perform a proper comparison between them. On pockesphinx continuous there parameters have to be set manually on the command 's arguments

-silprob 0.1 -wip 2e-1 -beam 1e-80 -bestpath yes -lw 10 -wbeam 1e-40

Experiment 2 Model Tested with new configuration

In the table below we have the results obtained with the new system configuration for the model trained in experiment 2

Nr	Command	WER
1	Mergi inainte	1/10
2	Vino inapoi	5/10
3	Roteste stanga	1/10
4	Roteste dreapta	8/10
5	Stop	2/10
6	Masoara Temperatura	5/10
7	Nivel fum	0/10
8	Nivel gaz	0/10
9	Nivel Baterie	5/10
10	Engleza	4/10
11	Mix of commands	3/10

Table 4.12 Experiment 2 Subjective WER

Experiment 3 Model Tested with new configuration

In the table below we have the results obtained with the new system configuration for the model trained in experiment 3

Nr	Command	WER
1	Mergi inainte	10/10
2	Vino inapoi	3/10
3	Roteste stanga	4/10
4	Roteste dreapta	8/10
5	Stop	2/10
6	Masoara Temperatura	4/10
7	Nivel fum	0/10
8	Nivel gaz	0/10
9	Nivel Baterie	8/10
10	Engleza	2/10
11	Mix of commands	4/10

Table 4.13 Experiment 3 Subjective WER

In the figure 4.42 and 4.43 below we have samples of commands recognized in pockesphinx continuous

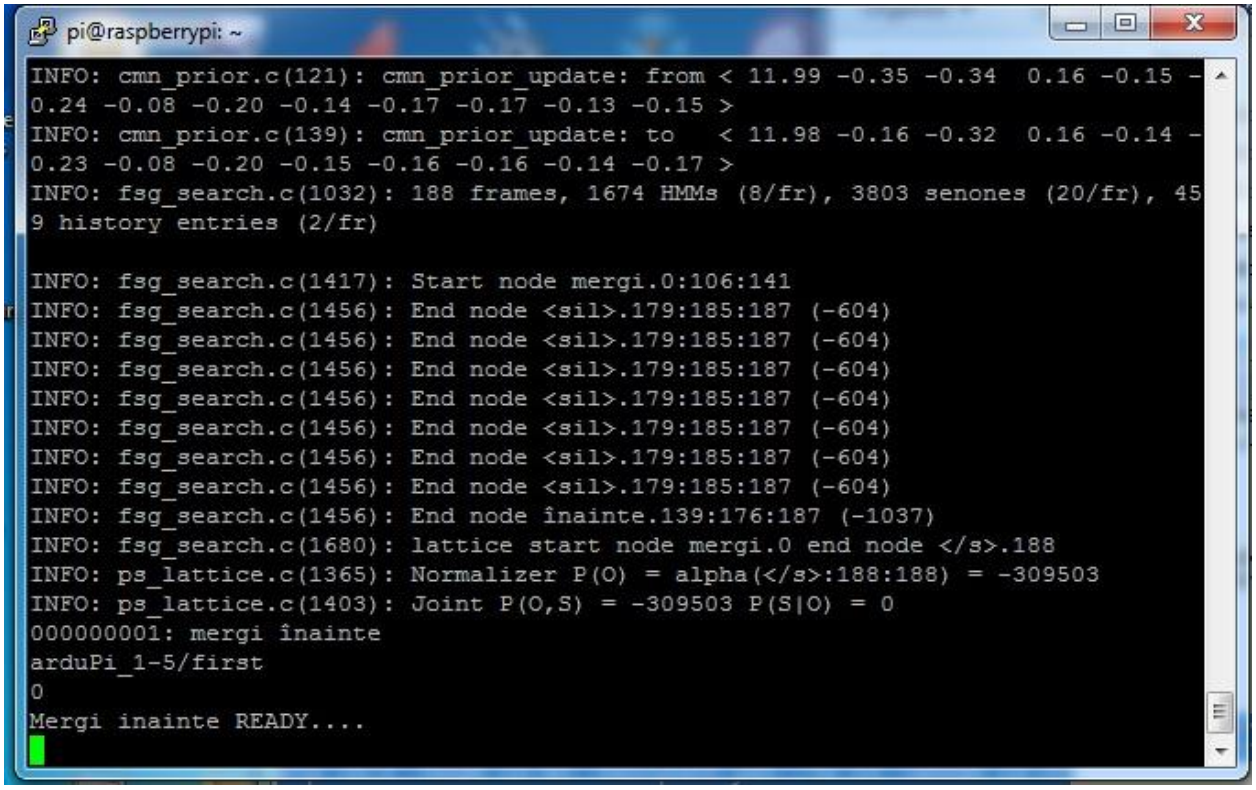


Figure 4.42 Command recognition in Pockesphinx Continuous

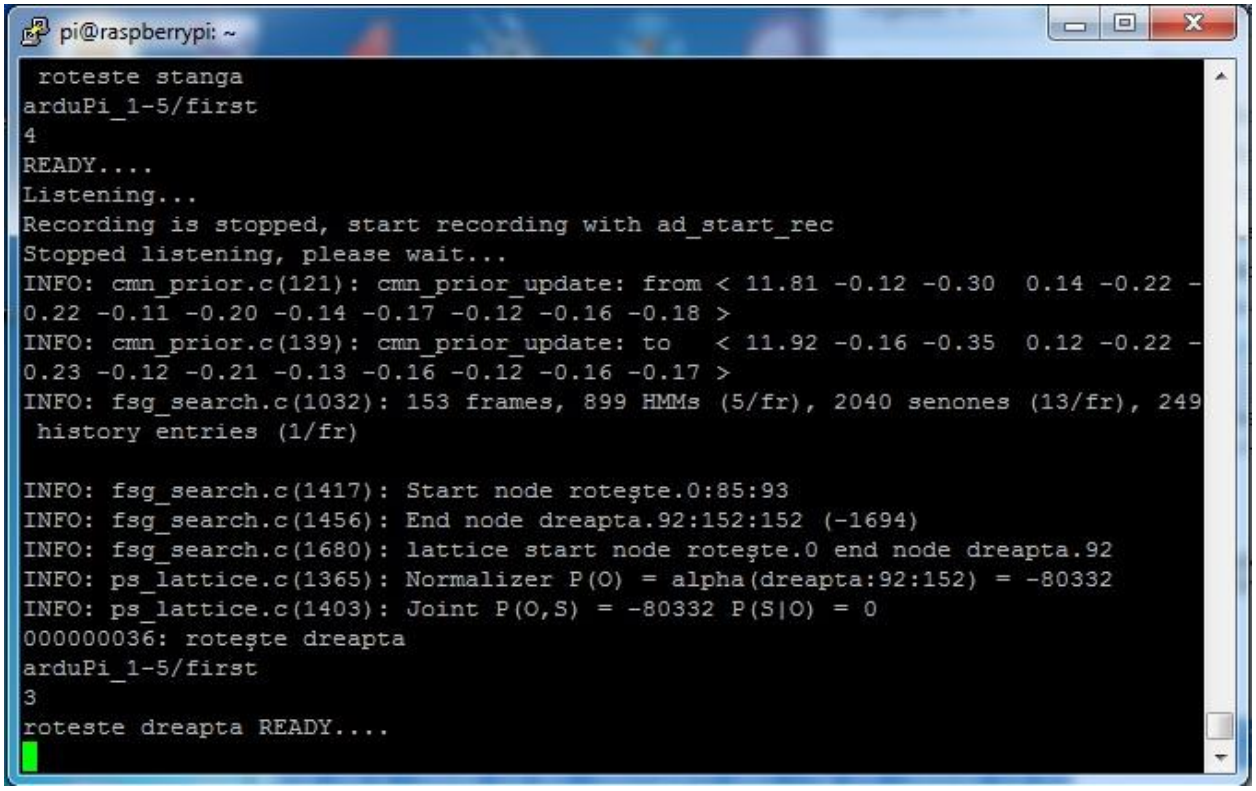


Figure 4.43 Command recognition in Pockesphinx Continuous

4.3.6 Integrating the components. VCR Working principle

Next I'll present how I integrated all the components presented in the block architecture of the Voice Controlled Remote

In order to send user voice commands recognized by Pocketsphinx continuous , receive robot feedback and display it on the watch , we have to use a different approach than that used when the system was composed of a microcontroller and a voice recognition shield. Microcontrollers run only one program at a time over and over again ,but by using Raspberry Pi we benefit from the advantages of a computer which can run multiple processes in parallel and also that in a C program we can start other processes if a we like.

For the sake of simplicity a used a C library written for Raspberry Pi named arduPi which has defines most of Arduino's methods and is compiled with a g++ compiler .

ArduPi template looks like this :

```
int main (){
    setup();
    while(1){
        loop();
    }
    return (0);
}
```

And unlike in Arduino IDE we have to define ourselves the methods setup() and loop ()

In order to use the voice commands recognized in Pocket sphinx continuous we need to handle the situations where the input commands matched a desired string - in this case the commands names in the file named continuous.c which can be found in **/pocketshpinx-0.8/src/programs .**

I made 3 programs - which are called when needed or run in background along with pocket sphinx continuous

Before talking about programs I'll make a short introduction of Processes and Background processes

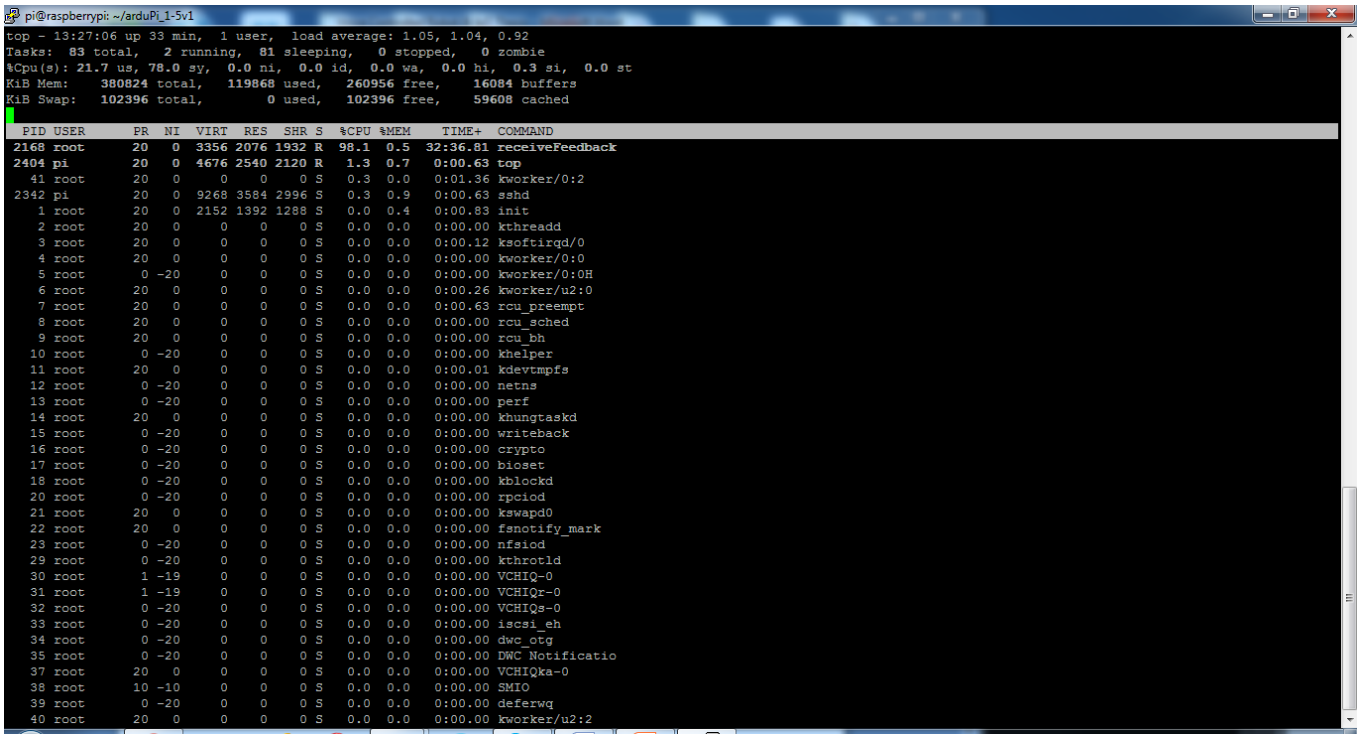
Processes . Background processes

After the compilation of a cpp source code , we obtain either a binary myProgram.o or an executable file myProgram.exe , and a running program it represents a process . A process is only an instance of a running program because the program can be started multiple times and ran in parallel.

The operating system tracks processes through a five digit ID number known as the **pid** or process ID . Each process in the system has a unique pid. [28]

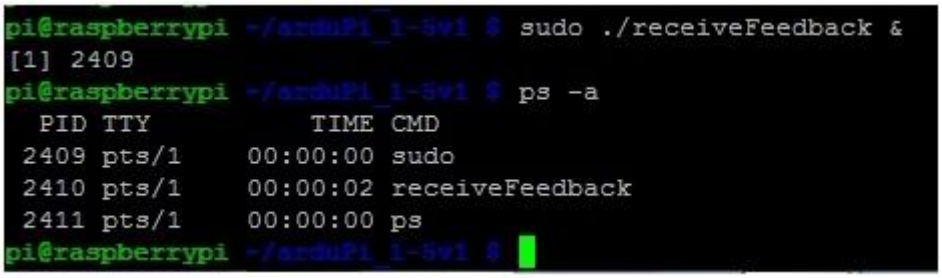
There are foreground processes and background processes also named daemons or jobs , the difference between them being that the foreground processes locks the terminal and the user can't give any more commands while the background process does not . [28]

Processes which have been launched from the current terminal can be seen with the command *ps -a* , and all system processes can be seen with *top* command. To launch a process from a binary file *./myProgram.o* command has to be used



4.44 Output of top command , all processes

Background processes are launched in the same way , but with by using an & symbol as we can see in figure below



4.45 Background process

The programs used are

One for sending a proper radio commands to the robot when a voice commands is recognized named sendCommand .

In order to start a process from a C code if the desired conditions is met , in Linux

system method can be used which can be called in the following way

system(" ./path/ binary ") , in this case

Or you can use any bash command you want , for example for playing an audio message along with the feedback received from the robot , I used aplay command and the path to the wav file as arguments for system () method .

system(aplay /home/pi/recordings/robotRecordings/NivelBaterie.wav) ;

One for displaying the status of the voice recognition program - when a commands was successful recognized or if the process failed named displayCommand.

One which runs as a background process / job named receiveFeedback used for listening to UART port for incoming feedback from the robot and displaying it on the Watch and playing feedback message.

We have to use it as a background process because pockesphinx continuous is already running as a foreground process therefore has already locked the terminal.

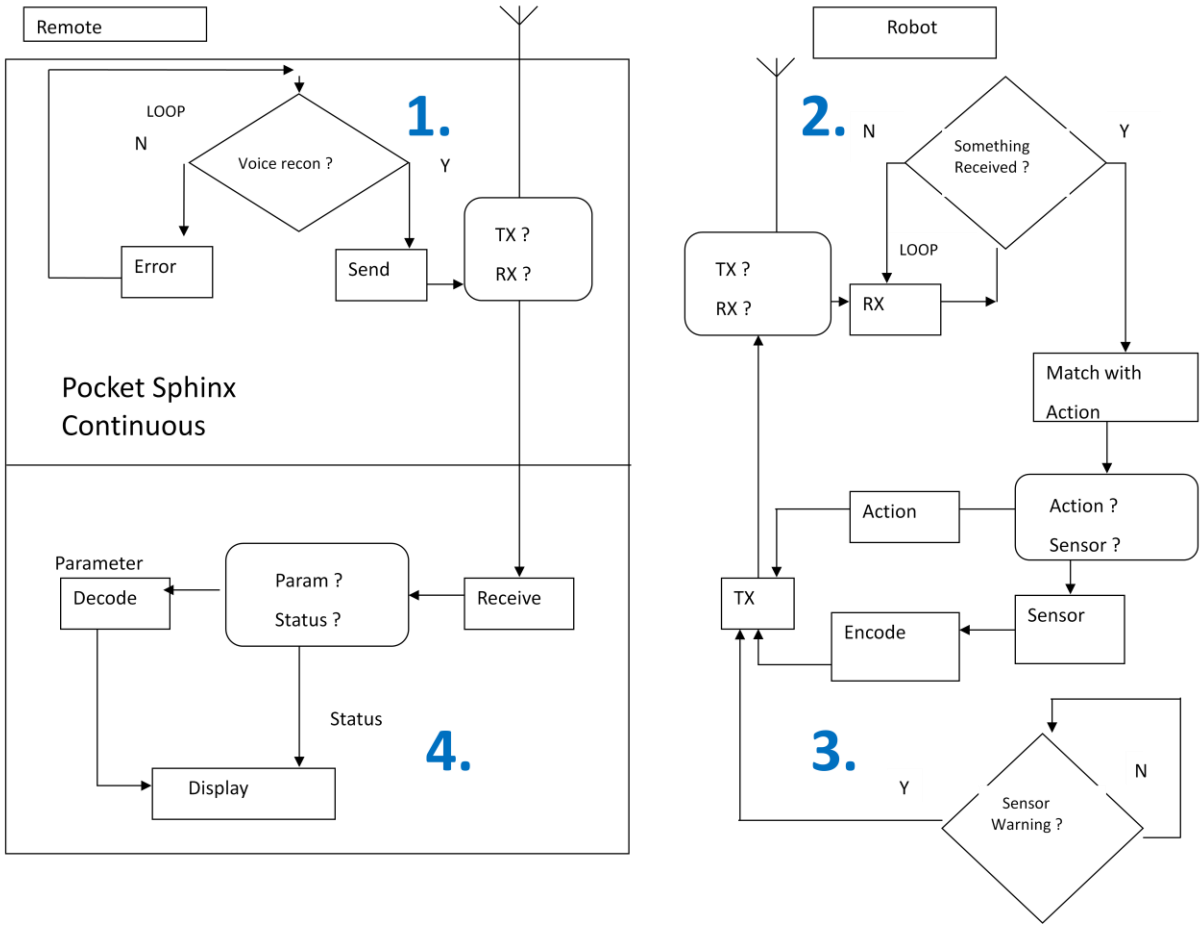
And also if we want to avoid to start manually the programs after connecting to raspberry Pi through Ethernet port we should add the processes to start with Linux at Boot time . This can be accomplished by adding the commands to *rc.local* [29]

rc.local is a file found in /etc and can be edited with command *sudo nano /etc/rc.local*

But we must not forget to add them both as background processes otherwise Linux will not finish booting anymore

Software flowchart

In this subchapter after integrating all the components above described I'll present the working principle , see figure 4.46



4.46 Software flowchart

Step 1 Send commands :

In essence it consist of the voice recognition loop . Pocket Sphinx continuous process running and listening for User voice commands. If a group of words matching the command list is recognized it's index will be sent to the robot

Step 2 Robot executes

First the robot is set on reception , it constantly checks the serial port (The something received loop).

If an order has been received the, Match with action block enters stage - in essence a switch case statement whose job is to match an action or a DAQ request with the order received.

In parallel there is a warning procedure. The robot polls the environment sensors and battery level and issues a warning whenever a threshold is exceeded.

Step 3 Feedback

Consists of the encoding operation and the Code transmission. Encoding is needed because of the different kind of data that can be sent from sensors and because of using only one channel of communication . E.g. Number 10 can be a command or a temperature level or a battery level , so the robot encodes it in order to avoid confusion.

Step 4 Feedback displayed

Feedback is received by the receiveFeedback process running in background which listens to the serial port . A decision has to be made whether a status or a parameter has been received be decoding the received data . Both of them are displayed on the special watch and also a voice report is played.

4.4 Design Part3 TheArduino Server

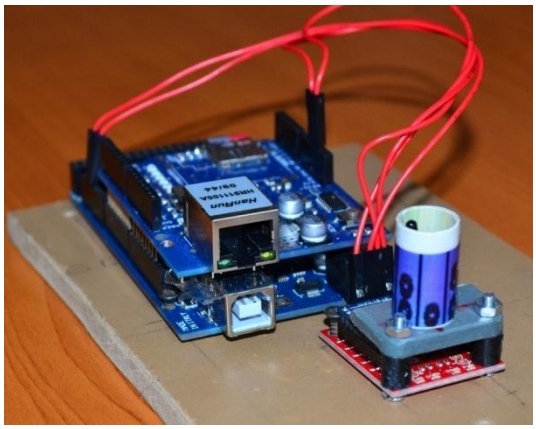


Figure 4.47 Arduino server

I choose to build the embedded server with an ArduinoMega microcontroller board and an Ethernet Shield (Fig. 4.47) which is able to provide the Arduino with internet connectivity by using its SPI bus and also by using a microSD card to store server files , in this case the google maps API . And in order to receive the sensor data sent by the robot I used a Zigbee Network gateway , in short an XBee wireless transceiver which is part of the Zigbee network formed by the robot and voice controlled remote. It's block architecture is represented in Fig. 4.48.

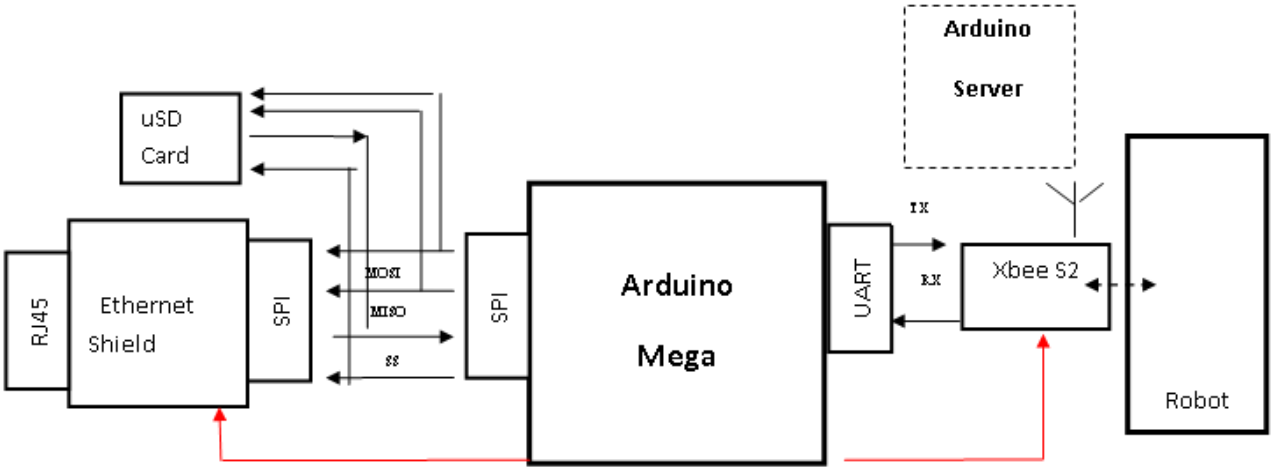


Figure 4.48 Arduino Server Block Schematic

Obviously the internet connection speed will be limited by the SPI's bus maximum data transfer rate , but for this application it's enough since the webpage will only display the robot's position on the map , and once in 5 seconds the sensor parameters or warnings sent by the robot.

Since in order to create web pages,markup languages such as HTML ans CSS are used so, one has two possibilities :

1. Write the code in a notepad text document and save it on the microSD card

2. Send the HTMLtags line by line to the client in the arduino code Arduino Mega

I used both methods for my application , the first one for the Google maps API and the second for the robot data.

4.4.1 Arduino Mega

The Arduino Mega 2560 is a microcontroller board(Fig. 4.49) based on the ATmega2560 . It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

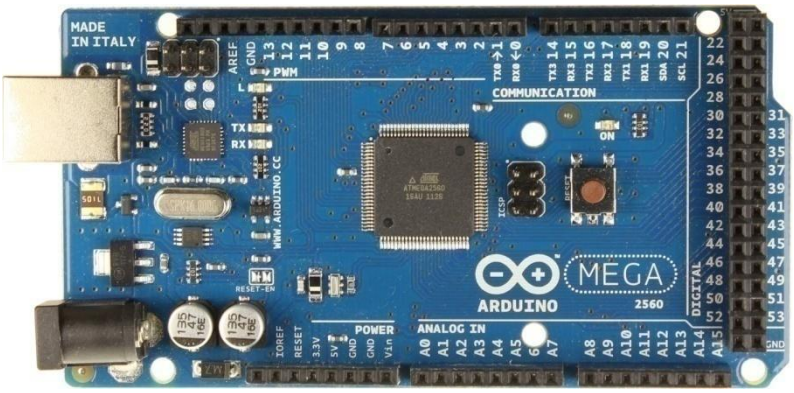


Figure 4.49 Arduino Mega 2560 [30]

Summary [30]

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

- Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data.
- External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).
- PWM: 2 to 13 and 44 to 46. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).

- TWI: 20 (SDA) and 21 (SCL). Support TWI communication using the Wire library. [30]

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values)

For my application I used the SPI bus and one of UART ports. Next I will introduce the SPI bus

4.4.2 SPI Bus

The **Serial Peripheral Interface** (**SPI**) bus is a 4 wire asynchronous serial communication interface specification used for short distance communication.

SPI devices communicate in full duplex mode using a master-slave architecture with a single master. The master device originates the frame for reading and writing. Multiple slave devices are supported through selection with individual slave select (SS) lines. [31]

The SPI bus specifies four logic signals:

- SCLK : Serial Clock (output from master).
- MOSI : Master Output, Slave Input (output from master).
- MISO : Master Input, Slave Output (output from slave).
- SS : Slave Select (active low, output from master).
    - The SPI bus can operate with a single master device and with one or more slave devices.(Fig. 4.50)
    - If a single slave device is used, the SS pin *may* be fixed to logic low if the slave permits it. With multiple slave devices, an independent SS signal is required from the master for each slave device.
    - Most slave devices have tri-state outputs so their MISO signal becomes high impedance (*logically disconnected*) when the device is not selected. Devices without tri-state outputs cannot share SPI bus segments with other devices; only one such slave could talk to the master, and only its chip select could be activated.

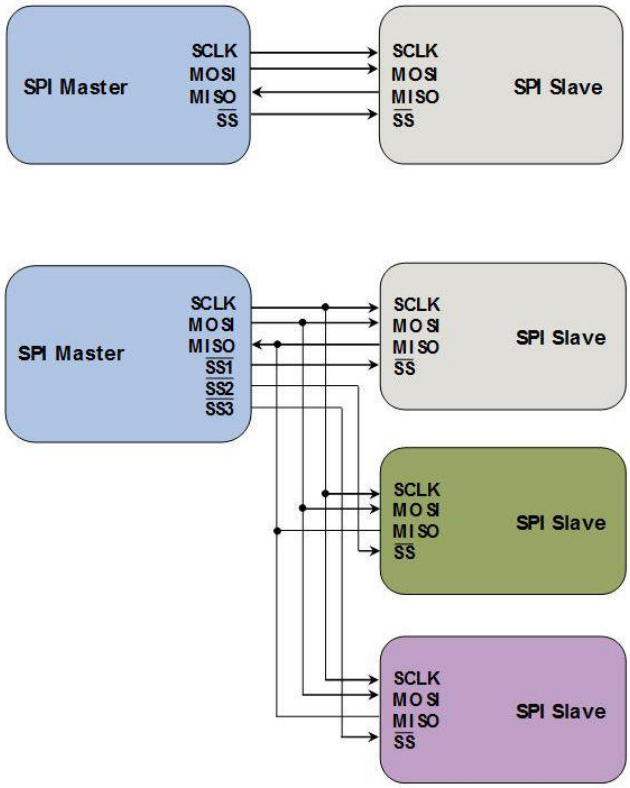


Figure 4.50 SPI communication [32]

To begin the communication, the bus master configures the clock, using a frequency supported by the slave device, typically up to a few MHz.The master then selects the slave device with a logic level 0 on the select line. If a waiting period is required, such as for analog-to-digital conversion, the master must wait for at least that period of time before issuing clock cycles.

During each SPI clock cycle, a full duplex data transmission occurs. The master sends a bit on the MOSI line and the slave reads it, while the slave sends a bit on the MISO line and the master reads it. This sequence is maintained even when only one-directional data transfer is intended.

Transmissions normally involve two shift registers of some given word size, such as eight bits, one in the master and one in the slave; they are connected in a virtual ring topology (Fig. 4.51). Data is usually shifted out with the most-significant bit first, while shifting a new least-significant bit into the same register. After that register has been shifted out, the master and slave have exchanged register values. If more data needs to be exchanged, the shift registers are reloaded and the process repeats. Transmission may continue for any number of clock cycles. When complete, the master stops toggling the clock signal, and typically deselects the slave. [31]

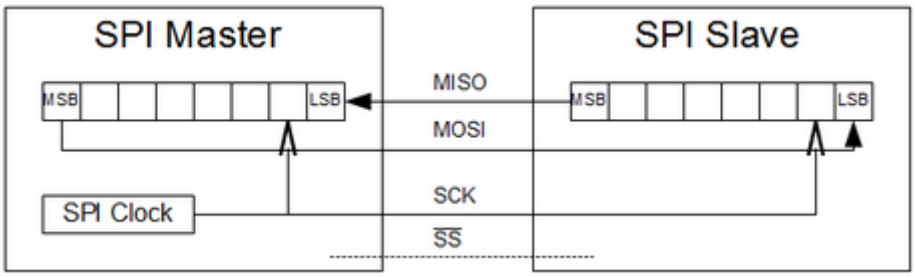


Figure 4.51 Master – slave communication [31]

4.4.3 Xbee Network configuration

As we have seen in subchapter 4.1.4 in Figure 4.19 the possible network connections are : tree , mesh , poin to point, start but for the current application which involves also the embedded server gateway it's necessary to configure a cluster tree network composed of a Coordinator and two Routers where the robot's transceiver is the coordinator and the remote's Xbee along with servers one are the routers . If for a simple pair of transceivers it has enough to configure each of the two transceiver to send data to the other's address , now when using a 3 radio network things are different , so I set up the system to exchange data in the following way :

The VCR sends its commands to the robot , and robot feedback will be sent to both the VCR and the Server at the same time , so in terms of destination addresses , I set up both the server's and the VCR's transceiver to send data to the robot , and the robot's transceiver (the coordinator) to broadcast it's data packets to all active radios on the Zigbee network .

Xbee configuration is done in a software called X-CTU by connecting it to a USB port using and Xbee USB explorer board responsible for converting UART to USB protocols.

As it can be seen in Fig. 4.52 ,Serial Number High and Serial Number Low fields compose the transceivers MAC address , so in order to connect this device with another one , in the Destination Address High and Low fields one has to write the other device's MAC address , and also to make sure they are on the same network (the same PAN ID) . This is usually done to set up pair networks. In order to configure a transceiver to broadcast its data packets , Destination Address High field has to be set to 0 and Destination Address Low has to be set to 0Xffff.

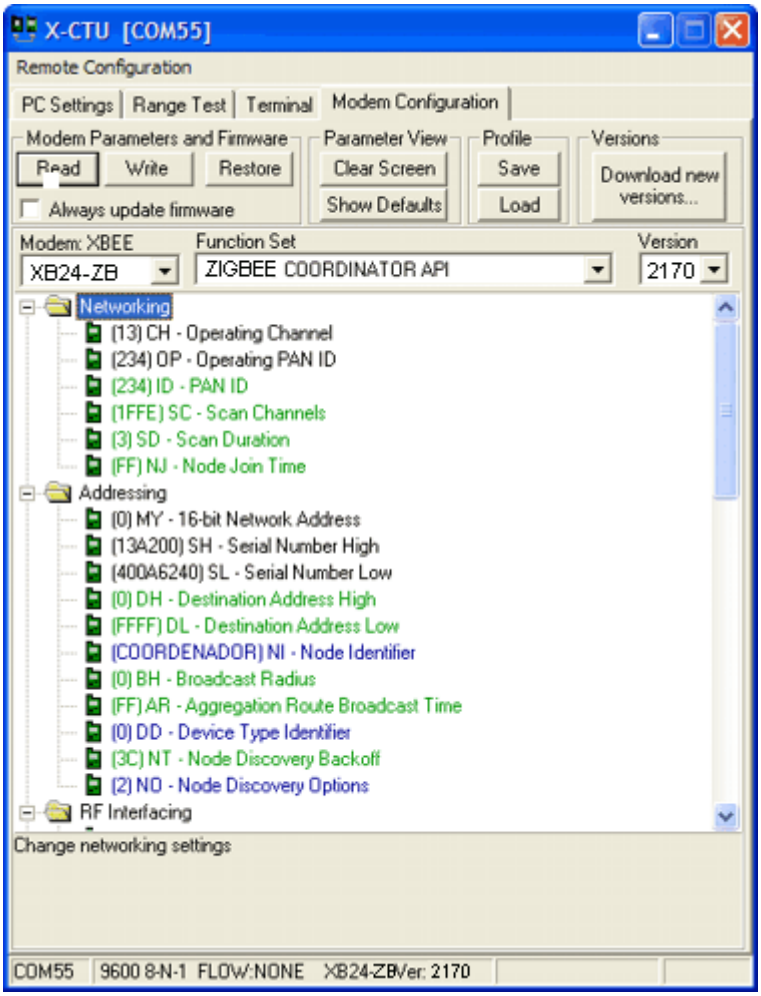


Figure 4.52 Xbee configuration in XCTU

4.4.4 Network communications introduction

In this subchapter I will introduce the necessary theoretical information needed to understand in general how a server-client communication works , how is assign the IP address and in particular how the Arduino based server works

4.4.4.1 Server

A **server** is a running instance of an application (software) capable of accepting requests from the client and giving responses accordingly.

Servers operate within a client-server architecture. Servers are computer programs running to serve the requests of other programs, the clients. Thus, the server performs some tasks on behalf of clients. It facilitates the clients to share data, information or any hardware and software resources. [33]

- Web server, a server that HTTP clients connect to in order to send commands and receive responses along with data contents [33]

  For my application , the server built with the Arduino and Ethernet Shield forms a web server

4.4.4.2 HTTP protocol

By knowing that network communications is performed by using abstraction layers I will make a brief description of them before moving to the HTTP protocol which belongs to one of the superior layers.

According OSI model (Open Systems Interconnection model) which represents a conceptual standard model that applies to all telecommunications regardless of their internal structure of technology in order to achieve interoperability between networks either wired or wireless , every communication system has to be partitioned into 7 abstraction layers as represented in Table 1.[34]

The examples I provided are only those used for my applications.

The first layer usually refers to the physical layer (Ethernet /SPI) , and the 7th layer is closest to the user by being the application layer.Any layer serves the layer above it and is served by the layer below it.

OSI Model			
Layer	Data Unit	Function	Examples
7.Application layer	Data	High-level APIs, including resource sharing, remote file access	HTTP
6. Presentation		Translation of data between a networking service and an application; character encoding, data compression and encryption/decryption	ASCII
5. Session		Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes	
4. Transport	Segments	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing	TCP
3. Network	Packet	Structuring and managing a multi-node network, including addressing, routing	IPv4
2. Data link	Bit/Frame	Reliable transmission of data frames between two nodes connected by a physical layer	

1.Physical	Bit	Transmission and reception of raw bit streams over a physical medium	SPI, Ethernet

Table 4.14 Abstraction layers [34]

Hyper Text Transmission protocol is a request-response protocol placed at the Application layer in the client-server computing model. A web browser, for example, may be the client and an application running on a computer hosting a web site may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body. [35]

In order start the communications between a client and a server a HTTP session has to be openedAn HTTP session is a sequence of network request-response transactions. An HTTP client initiates a request by establishing a Transmission Control Protocol (TCP) connection to a particular port on a server (typically port 80) [35]

An HTTP server listening on that port waits for a client's request message. Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own. The body of this message is typically the requested resource – such as the HTML page , although an error message or other information may also be returned.

The request message consists of the following:

A client request line, for example :

GET /Robot DAQ Server.html HTTP/1.1,

host : 192.168.0.101

Which requests the html with the same name from the server which like in our case can be found at this IP address.

Request header fields, such as Accept-Language: en

An empty line.

An optional message body.

The request line and other header fields must each end with <CR><LF> (that is, a carriage return character followed by a line feed character). The empty line must consist of only <CR><LF> and no other whitespace. [35]

The Server response includes :

HTTP/1.1 200 OK

Content-Type: text/html

Connection: close

<!DOCTYPE HTML>

<html>

…

</html>

This example is the typical response given by my Arduino based server assuming that the connections has been established, but other server messages may include also other information.

Other types of answer codes include HTTP/1.1 408 Request time out or HTTP/1.1 404 Not Found is there isn't a network connection.

4.4.4.3 DHCP

Dynamic Host Configuration Protocol is used by computers for requesting Internet Protocol parameters, such as an IP address from a network server.

Within a local network, DHCP assigns a local IP address to devices connected to the local network.

When a computer or other networked device connects to a network, the DHCP client software in its operating system sends a broadcast query requesting necessary information.

The DHCP server manages a pool of IP addresses and information about client configuration parameters such as default gateway, domain name, .On receiving a request, the server may respond with specific information for each client, as previously configured by an administrator, or with a specific address and any other information valid for the entire network, and the time period for which the allocation (*lease*) is valid.

Depending on implementation, the DHCP server may have three methods of allocating IP-addresses:

- *dynamic allocation*: A network administrator reserves a range of IP addresses for DHCP, and each client computer on the LAN is configured to request an IP address from the DHCP server during network initialization. The request-and-grant process uses a lease

concept with a controllable time period, allowing the DHCP server to reclaim (and then reallocate) IP addresses that are not renewed [36]

- *automatic allocation*: The DHCP server permanently assigns an IP address to a requesting client from the range defined by the administrator. This is like dynamic allocation, but the DHCP server keeps a table of past IP address assignments, so that it can preferentially assign to a client the same IP address that the client previously had.
- static allocation - only a table of MAC addresses made by the network administrator will receive a previous allocated IP address [36]

4.4.5 Ethernet Shield

The Arduino Ethernet Shield(Fig. 4.53) allows an Arduino board to connect to the internet. It is based on the Wiznet W5100ethernet chip (datasheet). The Wiznet W5100 provides a network (IP) stack capable of both TCP and UDP. It supports up to four simultaneous socket connections. Use the Ethernet library to write sketches which connect to the internet using the shield. The ethernet shield connects to an Arduino board using long wire-wrap headers which extend through the shield. This keeps the pin layout intact and allows another shield to be stacked on top. [37]

The shield contains a number of informational LEDs:

- PWR: indicates that the board and shield are powered
- LINK: indicates the presence of a network link and flashes when the shield transmits or receives data
- FULLD: indicates that the network connection is full duplex
- 100M: indicates the presence of a 100 Mb/s network connection (as opposed to 10 Mb/s)
- RX: flashes when the shield receives data
- TX: flashes when the shield sends data
- COLL: flashes when network collisions are detected [37]

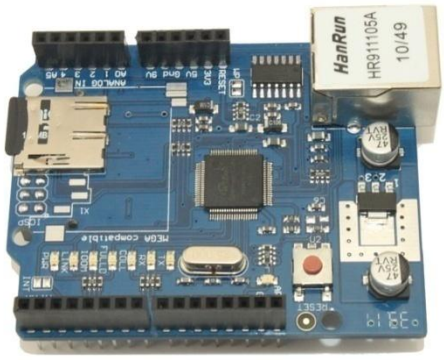


Figure 4.53 Ethernet Shield[37]

Arduino communicates with both the W5100 and SD card using the SPI bus (through the ICSP header). This is on digital pins 10, 11, 12, and 13 on the Uno and pins 50, 51, and 52 on the Mega. On both boards, pin 10 is used to select the W5100 and pin 4 for the SD card. These pins cannot be used for general I/O. On the Mega, the hardware SS pin, 53, is not used to select either the W5100 or the SD card, but it must be kept as an output or the SPI interface won't work.

One should take into account also that the W5100 and SD card share the SPI bus, only one can be active at a time. Therefore if both peripherals need to be used in the program just like in our case , they have to be active one at a time so manual activation/deactivation by controlling the SPI SS pin has to be performed . To do this with the SD card, set pin 4 as an output and write a high to it. For the W5100, set digital pin 10 as a high output.

As I mentioned earlier the Ethernet shield is configured as a web server so in order to allow a client to initiate an HTTP connection (Hypertext Transmission Protocol) it has to respect also the default protocol.

So the Arduino will listen the Ethernet connection for incoming client http connections requests.

If something has been received we have to check that is a valid HTTP /1.1 GET request so we have to check if it's terminated with a blank line and an end line character (/n).

If this condition is full field the server will respond with one of the standard HTTP headers - the response for successful connection : HTTP/1.1 200 OK and an entity describing the requested resource Content-Type: text/html. In Arduino code the only way to send information as characters is by using the public print function which can be used by all classes defining

microcontroller peripherals . So in order to send the http headers and also later the HTMP tags client.print() function is used having under quotes the headers / tags to be sent

```
EthernetClient client = server.available();

if (client) {  // got client?

booleancurrentLineIsBlank = true;

while (client.connected()) {

if (client.available()) {   // client data available to read

char c = client.read(); // read 1 byte (character) from client

// last line of client request is blank and ends with \n

      // respond to client only after last line received

if (c == '\n' &&currentLineIsBlank) {

// send a standard http response header

client.println("HTTP/1.1 200 OK");

client.println("Content-Type: text/html");

client.println("Connection: close");

client.println("<!DOCTYPE HTML>"); // creating a new HTML file for the sensor data

// the first one was only for the google maps API

client.println("<html>");  // all HTML headers have to be sent like this
```

4.4.5.1 Auto IP assignement

Usually in order to create a server you need a static IP to host the webpage . Using the arduino code you have to initially choose a free fixed IPv4 address from the LAN. IPAddress ip(192, 168, 0, 10); and along with MAC address

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; the server is created at port 80 corresponding to http : EthernetServer server(80);

This practice of course is very simple and it works if we connect the server to a router that doesn't have many devices connected to it, especially if it's also a wireless access point(usually one's home router) so it's easy to look for a free IP and upload your code to the microcontroller. The first 2 bytes of the IP address , in my case 192 and 168 are very common for LAN's in Romania so guessing them is easy , the 3 rd byte is usually specific to the router type so we are left with the last byte to try and find a free IP And even though many devices can be connected to your router , you can always check them on the routers webpage , the DHCP client list and find a free IP or use a network scanner software and find a free IP. Trouble is that you don't always have access to a router for which you know the password and if multiple devices connect to the wireless access point , the DHCP server will dynamically assign an IP and even if they disconnect their devices , the server will memorize their MAC and IP address for a period known as the lease time (default lease time is 2 hours) so you may find yourself searching manually through the 255 possibilities for a free IP which is time consuming .

DHCP servers solves this problem for client devices , so I thought that the best way to solve this issue is to use them to assign also to my server a free IP to host my webpage . I'm able to do this since the user can access my page just by typing the IP address to the browser and not by typing a website name hosted on a domain.

In order to do this I used the Ethernet.h library function which is responsible for communicating with the DHCP server :Ethernet.localIP()[thisByte] which returns one byte at a time from the IPv4 address assigned by the DHCP. They will be stored in a software defined buffer called ipAddr and this way the address obtained will be used to initialize the server's static IP

```
for (byte thisByte = 0; thisByte< 4; thisByte++) {

   // read the value of each byte of the IP address:

ipAddr[thisByte] = Ethernet.localIP()[thisByte];

    }

IPAddressip (ipAddr[0],ipAddr[1],ipAddr[2],ipAddr[3]); // server IP address assignment
```

I also had to keep in mind that , because of using also the microSD card and the Ethernet connection which are using the same SPI bus to receive data, it's impossible to keep them both active at a time because as I experimented ,serious issues appeared . Either the server could not receive an IP from the DHCP server or the microSD card couldn't be initialized , so I had to manually deselect one of the SPI slaves at a time.

This is done in a simple way , you have to set HIGH digital pin 4 (since Slave Select is active when LOW)which corresponds to the SD card or either pin 10 which corresponds to the Ethernet chip. This has to be done properly at particular moments , otherwise it won't work since whenever call a Ethernet class member functions or an SD class member functions they deal with this pins automatically.

So as a solution , in void setup() first thing to do before initializing the Ethernet Shield is deactivate the SD card and after acquiring the IP address from the DHCP the SD card will be reactivated by calling it's class member functions

```
void setup () {

pinMode(4,OUTPUT);

digitalWrite(4,HIGH);

// next  start the Ethernet connection

....

}
```

4.4.5.2 MicroSD card

MicroSD cards are one of the smaller versions of the classic SD card which stands for Secure Digital and are a very reliable fast accessing non-volatile random access memories based on flash technology .

For our application is the best suited solutions to store HTML files since it offers an 8Gb storing capacity.

Unfortunately since is accessed by the SPI bus we are unable to make data transfer as fast as a class 4 SD card will allow us (25 Mbytes/second) , but for our applications is enough

4.4.5.3 HTML & CSS

In this subchapter I will present briefly the role of this 2 scripting languages and introduce the tags that I used for developing my projects website

HyperText Markup Language, commonly referred to as **HTML**, is the standard markup language used to create web pages.

A **markup language** is a system for annotating a document in a way that is syntactically distinguishable from the text.[

Web browsers can read HTML files and compose them into visible or audible web pages. Browsers do not display the HTML tags and scripts, but use them to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language. [38]

It is written in the form of HTML elements consisting of *tags* enclosed in angle (like <html>). HTML tags most commonly come in pairs like <h1> and </h1>,

<!DOCTYPE html> used to declared the type of document being sent to the browser

<html> , </html> are used to create the limit of the page
<head></head> - the head pair tells the browser that anything contained in it will be displayed in the header
<title></title>Page Title
<body></body> describes
<h1></h1>a Heading , need to be inside body
paragraph

<style></style> can be used to establish the style for a category of tags , for example all paragraphs can be set to have the same font and size .

<u></u> encloses a list of elements

 elements from the list , has to be inside <u></u > [38]

Cascading Style Sheets (**CSS**) is a style sheet language used for describing the look and formatting of a document written in amarkup language. While most often used to change the style ofweb pages and user interfaces written in HTML and XHTML [39]

Nowadays we don't need any more to have separate HTML and CSS files since HTML 5 uses also some of the CSS attributes directly . They can be called either separately or mixed together

p {font-size :160% } This will set all paragraphs fonts to 160%

ul {font-size :120% } This will set all list elements fonts to 120%

<ul style="text-align:left"> This HTML tag concerning lists contains also a CSS attribute , namely style which can also be declared separately just like in the previous line

4.3.5.4 Google Maps API

The Google Maps API is a JavaScript library. It can be added to a HTML web page inside a <script></script> pair of tags.

So 4 steps have to be performed in order to have a functional API on our website

1. Load the Google maps API from google server

<script src="http://maps.googleapis.com/maps/api/js"></script>

2. Set map properties

An in order to use it , it has to be initialize using a functions with the same name which will have to define three properties :

The **center** property specifies where to center the map. Create a **LatLng** object to center the map on a specific point. Pass the coordinates in the order: latitude, longitude.

How much will we see initially on the map , so therefore the **zoom property** which specifies the zoom level for the map. zoom: 0 shows a map of the Earth fully zoomed out. Higher zoom levels zoom in at a higher resolution. [40]

- And what kind of map we'll see given by the **mapTypeId property** :
- ROADMAP (normal, default 2D map)
- SATELLITE (photographic map)
- HYBRID (photographic map + roads and city names)
- TERRAIN (map with mountains, rivers, etc.)

```
function initialize() {

var mapProp = {
  center:new google.maps.LatLng(51.508742, -0.120850),
  zoom: 7,
  mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

3. Create a Map Container

Create a element to hold the map. Use CSS to size the element:

```
<div id="googleMap" style="width:500px;height:380px;"></div>
```

4. Create a Map Object

The code below creates a new map inside the element with id="googleMap", using the parameters that are passed (mapProp).

```
var map=new google.maps.Map(document.getElementById("googleMap"), mapProp);
```

5. Add an Event Listener to Load the Map

Add a DOM listener that will execute the initialize() function on window load (when the page is loaded):

google.maps.event.addDomListener(window, 'load', initialize);

But in our case we don't need to display a fixed location but the robot's location .(Fig. 4.54)

But it's not necessary to read GPS data from the robot , instead it's enough to know where de server is since for my applications the server has to be placed in the area served by the patrolling robot.

In order to know where the server is we use geolocation function

//reference

Figure 4.54 Google maps API

4.3.5.5 Website

The web server's page is represented in figures 4.55 and 4.56 . It can provide the same information to a user remote connected to the LAN of the target patrolling area as to the user in the field which asks the robot directly of what information he needs to know.

The server will display on its webpage temperature information as in figure 4.55 , gas or smoke presence information and in short all the information that the user asked the robot. But the most

important parameters which have to be monitored from a command centre are the monitored parameters warnings such as Low battery warning as in figure 4.58 ,high temperature warning , gas / smoke presence warning in order the enable the emergency teams to react quickly in case of danger. In order to pinpoint on the map the locations of the event , the servers location information is also displayed by using the Google maps API.

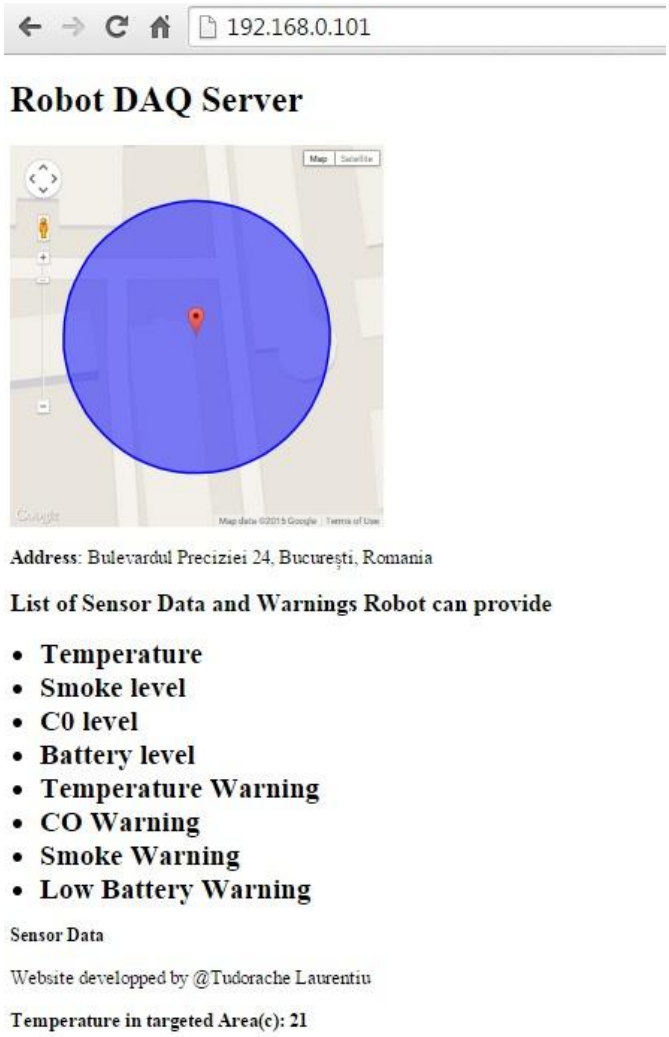


Figure 4.55 Webpage showing temperature measurement

Figure 4.56 Webpage showing Low battery Warning

5. Conclusion

The developed prototype demonstrates a promising patrol vehicle that can be used for prevention and security due to its ability to monitor the content of toxic gas. Its environmental sensors inform the user about air quality in the surrounding area using warning procedures implemented ,so, in this way if a certain parameter especially gas concentration exceeds a safety threshold, the happening is reported to the user via ZigBee wireless connection and displayed by the special watch .

I presented all the steps followed in order to design an automatic speech recognitions system which can recognize user voice commands and all the experiments I've made in order to find the best suitable solution and to improve command recognition accuracy with the purpose of serving the control system of the voice controlled robot .

The facility to be voice controlled makes it reliable and easy to use because everything the user has to do is to order and the robot will act as a teammate.

And finally as an enhancement ,a third block , a server has been added to the Zigbee network formed by the robot and it's remote control which collects the environment parameters information along with the possible event warnings and displays them on a its hosted web page which can be accessed remote by a command in centre in order to increase the time of response of an emergency team in case of special situations.

6. References

1. Frenoy O. Madre Deus , Nishit S. Borker, “Multi-purpose Robot for Hostile Environment Monitoring and Aid to Rescue Operations” , in Internațional Conference on Computing and Control Engineering (ICCCE 2012), 12 & 13 April,

2. Ovidiu Vermesan , Peter Friess , " Internet of Things - Converging technologies for smart environments and integrated ecosystems "

3. Robert Faludi Building Wireless Sensor Networks with Zigbee , XBee , Arduino and Processing

4. http://asimo.honda.com/gallery/

5 . http://digilentinc.com/Data/Products/PMOD-CLS/PmodCLS_rm_RevD-E.pdf

6. https://cloud.google.com/speech/

7. http://www.politepix.com/openears/

8.https://learn.adafruit.com/introducing-the-raspberry-pi-model-b-plus-plus-differences-vs-model-b/overview

9. Indrumar Proiect de cercetare-dezvoltare in tehnologia vorbirii S.l Dr. Ing. Horia Cucu
10. http://en.wikipedia.org/wiki/Foster-Miller_TALON
11.http://digilentinc.com/Data/Products/CHIPKIT-UNO32/chipKIT-Uno32-RevC_rm.pdf
12. http://digilentinc.com/Data/Products/PMOD-HB5/PmodHB5_RevD_rm.pdf
13. http://en.wikipedia.org/wiki/Pulse-width_modulation
14 . http://garagelab.com/profiles/blogs/tutorial-what-is-and-how-to-use-pwm-pulse-width-modulation
15. http://en.wikipedia.org/wiki/Analog-to-digital_converter
16. http://www.ti.com/lit/ds/symlink/lm50.pdf
17. https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf 62
18 http://inmotion.pt/documentation/pololu/POL-1480/MQ-2.pdf
19 http://www.sharpsma.com/webfm_send/1487
20 http://ftp1.digi.com/support/documentation/90000976_G.pdf
21. http://en.wikipedia.org/wiki/ZigBee
22 https://www.raspberrypi.org/products/model-b-plus/
23 http://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html
24 https://reference.digilentinc.com/_media/pmod:pmod:pmodCLS_rm.pdf
25 https://learn.sparkfun.com/tutorials/i2c
26 https://en.wikipedia.org/wiki/I%C2%B2C
27. https://www.arduino.cc/en/Reference/WireBeginTransmission
28. http://www.tutorialspoint.com/unix/unix-processes.htm
29 https://www.raspberrypi.org/documentation/linux/usage/rc-local.md
30 http://arduino.cc/en/Main/arduinoBoardMega
31 http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus
32 http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/
33 http://en.wikipedia.org/wiki/Server_(computing)
34 http://en.wikipedia.org/wiki/OSI_model
35 http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
36 http://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol
37 http://www.arduino.cc/Main/ArduinoEthernetShield
38 http://en.wikipedia.org/wiki/HTML
39 http://en.wikipedia.org/wiki/Cascading_Style_Sheets
40 http://www.w3schools.com/googleapi/google_maps_basic.asp

41. Wikipedia SSH , SFTP,