MULTIFUNCTION ROBOT CONTROLLED BY COMPUTER VISION SYSTEM

MOHAMMED SULAIMAN MUSTAFA

OCTOBER 2014

MULTIFUNCTION ROBOT CONTROLLED BY COMPUTER VISION SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES OF
ÇANKAYA UNIVERSITY

BY
MOHAMMED SULAIMAN MUSTAFA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF
COMPUTER ENGINEERING

OCTOBER 2014

Title of the Thesis : **Multifunction Robot Controlled by Computer Vision System**

Submitted by **Mohammed Sulaiman MUSTAFA**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.

[illegible]

Prof. Dr. Taner ALTUNOK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

[illegible]

Assist. Prof. Dr. Murat SARAN
Head of Department

This is to certify that we have read this thesis and that in our opinion; it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

[illegible]

Assist. Prof. Dr. Yuriy ALYEKSYEYENKOV
Supervisor

Examination Date : 31.10.2014

Examining Committee Members

Assist. Prof. Dr. Yuriy ALYEKSYEYENKOV	(Çankaya Univ.)	[illegible]
Assoc. Prof. Dr. Fahd JARAD	(THK Univ.)	[illegible]
Assist. Prof. Dr. Abdül Kadir GÖRÜR	(Çankaya Univ.)	[illegible]

STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name	: Mohammed Sulaiman MUSTAFA
Signature	: [illegible]
Date	: 31.10.2014

ABSTRACT

MULTIFUNCTION ROBOT
CONTROLLED BY COMPUTER VISION SYSTEM

Mohammed Sulaiman, MUSTAFA
M.Sc., Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Yuriy ALYEKSYEYENKOV
October 2014, 82 pages

In this thesis, we try to come up and build a robot platform with multifunction capabilities, easy to add, modify and delete those functions without redesigning, by using easy use technology that can create a suitable efficient platform. The process of building platform is by using figures, tables and programming code to make this thesis capable to apply and implement in real world, showing obstacles and challenges that lead to the key of success until it reaches the final goal. This thesis requires only basic level in electronic and computer programming because we are using a simplified way for building robot. The multifunction platform is a unique idea and opens new space to experimenters to get benefits from this opinions or ideas to use these functions in raw state, with no need to study hardware and software material of robot. The final robot form is shown in the last pages of this thesis as appendix.

Keywords: Robotics, Computer Vision, Machine Vision, Microcontroller.

ÖZ

BİLGİSAYAR GÖRÜNTÜ SİSTEMİ İLE KONTROL EDİLEN ÇOK FONKSİYONLU ROBOT

Mohammed Sulaiman, MUSTAFA
Yüksek Lisans, Bilgisayar Mühendisliği Anabilim Dalı
Tez Yöneticisi: Assist. Prof. Dr. Yuriy ALYEKSYEYENKOV
Ekim 2014, 82 sayfa

Bu tez çalışmasında, uygun verimlilikte platformları oluşturabilen kolay teknolojiyi kullanarak, tekrar tasarlanmasına gerek kalmaksızın fonksiyonlarına bir şeyler eklenmesi, modifiye edilmesi ve silinmesi kolay olan bir robot platformu yapmaya çalıştık. Bu platform oluşturma süreci, figürleri oluşturma, tablolar ve programlama kodlarından oluşarak, bu tezin gerçek hayatta uygulanmasını sağlayacak ve nihai amacına ulaşıncaya kadar başarı anahtarının zorluklarını gösterecektir. Bu tez için sadece temel düzeyde elektronik ve bilgisayar programlaması gerekir çünkü biz robot yapımı için basitleştirilmiş bir yol uyguluyoruz. Çok fonksiyonlu platform kendine has bir fikirdir ve deney yapanların robotun donanım ve yazılımı üzerinde çalışmaya gerek duymadan bu fonksiyonları ham hali ile kullanması amacıyla bu fikirlerden faydalanması için onlara bir yol açar. Robotun son hali, bu tezin son sayfası olan ekte sunulmuştur.

Anahtar Kelimeler: Robotik, Bilgisayar Gösterim, Makine Görme, Mikro Kontrolör.

ACKNOWLEDGEMENTS

I would like to express my special thanks to those names below, for them supports, efforts, supervision, special guidance, suggestions, and encouragement through the development of this thesis and for the department staff.

Special thanks to my teachers in university especially my supervisor:
Assist. Prof. Dr. Yuriy ALYEKSYEYENKOV.

Special thanks to my family especially my brother:
Assist. Samer Sulaiman MUSTAFA.

Special thanks to my friend’s especially my close friend:
Assist. Ahmed Burhan MOHAMMED.

TABLE OF CONTENTS

STATEMENT OF NON-PLAGIARISM iii

ABSTRACT iv

ÖZ v

ACKNOWLEDGEMENTS vi

TABLE OF CONTENTS vii

LIST OF FIGURES x

LIST OF TABLES xiii

LIST OF ABBREVIATIONS xiv

CHAPTERS

1. OVERVIEW 1

1.1. Introduction 1

1.2. Definitions 2

1.3. Background and Early Systems 5

1.4. Fields of Applications 7

1.4.1. Submarines Field 8

1.4.2. Space Field 8

1.4.3. Military Field 9

1.4.4. Industry Field 9

1.5. Thesis Goal 10

1.6. Thesis Outline 10

2. DESCRIPTION OF HARDWARE SYSTEM 11

2.1. Introduction 11

2.2. Chassis of My Project Robot 12

2.3. Peripheral Hardware 12

2.3.1. Motors 13

2.3.2. Relays 16

2.3.3. Robot Arm 17

2.3.4. Power Supply of System (Battery) 20

2.3.5. Camera 22

2.3.6. Steering..23

2.4. Core Hardware 25

2.4.1. Bridge to Hardware..25

2.4.2. Bridge to Software 27

2.4.3. Linking Two Bridges 28

2.5. Summary 28

3. DESCRIPTION OF THE SOFTWARE SYSTEM 29

3.1. Introduction 29

3.2. Software in SBM 29

3.3. Communication SBM with SBC (SBM Side)......................................35

3.4. Software in SBC 36

3.5. Communication SBC with SBM (SBC Side)......................................39

3.6. OpenCV (Add-On Library)..41

3.7. Image Structure in OpenCV..47

3.8. Vision Input in OpenCV 49

3.9. Object Representation in OpenCV 50

3.10. Function: Find Laser Spot Using RGB Manipulating 51

3.11. Function: Find Green Ball Using HSV Morphological Manipulating ...59

3.12. Function: Find Image Inside Image Using Feature Detection 64

3.13. Function: Trained Object Finder Using Cascade Classifier Training 68

3.14. Homing Guidance..73

3.15. Summary 76

4. CONCLUSION 77

4.1. Characterization and Future Works 77

4.2. Obstacles and Challenges 78
4.3. Recommendations and Developments..80

REFERENCES R1

APPENDICES A1

A. PROJECT ROBOT PICTURES AND MEASUREMENTS.........................A1
B. CURRICULUM VITAE A2

LIST OF FIGURES

FIGURES

Figure 1 The robots 2

Figure 2 The unmanned vehicles 3

Figure 3 The pods platform 4

Figure 4 The truck platform 4

Figure 5 The wheel platform 5

Figure 6 Isaac Asimov 5

Figure 7 TT-26 6

Figure 8 Raymond Goertz using E1 tele arm 7

Figure 9 Mr. Barack Obama playe soccer with ASIMO 7

Figure 10 Kingfish UUV 8

Figure 11 NASA space robots 8

Figure 12 Military robots, sword units 9

Figure 13 KUKA industry robots 9

Figure 14 Overview of my project robot 11

Figure 15 Chassis used in project robot 12

Figure 16 The motors 13

Figure 17 The gear motors 14

Figure 18 The brushless motors 14

Figure 19 The servo motors 15

FIGURES

Figure 20 The steeper motors 16

Figure 21 The relay 16

Figure 22 Relay scheme 17

Figure 23 Robot arm with five servo motors..17

Figure 24 Arm panned to right by motor “A” 18

Figure 25 Arm claw from close to open state by motor “E” 19

Figure 26 Arm in “pistol mode” 19

Figure 27 Arm in “cobra mode” 19

Figure 28 Arm in “park mode” 20

Figure 29 Comparison between three type batteries 21

Figure 30 Cells of Li-ion 22

Figure 31 Internal web camera for laptop 22

Figure 32 Parallel steering..23

Figure 33 All wheel steering..24

Figure 34 Front wheel steering 24

Figure 35 My project robot wheel system..25

Figure 36 Arduino uno R3..26

Figure 37 Arduino uno R3 datasheet schema 27

Figure 38 Laptop board using it as SBC 27

Figure 39 Comparison among connection mediums..28

Figure 40 Arduino IDE..30

Figure 41 Visual C++ 2010 ultimate, “selecting template” 38

FIGURES

Figure 42 Project configuration properties, “general” 38

Figure 43 Project configuration properties, “precompiled headers”.................39

Figure 44 Project configuration properties for openCV, “C\C++ general”........43

Figure 45 Project configuration properties for openCV, “linker general” 44

Figure 46 Project configuration properties for openCV, “linker input”............44

Figure 47 Source input test image...58

Figure 48 Obtain result mask image 58

Figure 49 Mask image enhancement...58

Figure 50 HSV color representation..60

Figure 51 Dilate operation 61

Figure 52 Erode operation 61

Figure 53 Opening operation 61

Figure 54 Closing operation 62

Figure 55 Gradient operation 62

Figure 56 Top hat operation..62

Figure 57 Black hat operation...63

Figure 58 SIFT detection key points 65

Figure 59 SIFT description with FLANN matching methods 65

Figure 60 Vector file “.vec” content for rubik's cube 70

Figure 61 Passive homing using “eye in hand” 75

Figure 62 Multifunction system illustrating diagram.......................................78

LIST OF TABLES

TABLES

Table 1 Difference Between Image Processing and Vision Processing 41

Table 2 Difference Between Machine Vision and Computer Vision 42

LIST OF ABBREVIATIONS

UGV	Unmanned Ground Vehicles
UAV	Unmanned Aerial Vehicles
UUV	Unmanned Underwater Vehicles
ASIMO	Advanced Step in Innovative Mobility
PWM	Pulse-Width modulation
GND	Ground
NiMH	Nickel–Metal Hydride
Li-ion	Lithium-ion
Li-Po	Lithium Polymer
SBM	Single-Board Microcontroller
ATMEL	Advanced Technology for Memory and Logic
RISC	Reduced Instruction Set Computing
AVR	Alf and Vegard RISC
PIC	Peripheral Interface Controller
STM	Sustainability Microelectronics
ARM	Acorn RISC Machine
R3	Revision 3
SBC	Single-Board Computer
LAN	Local Area Network
GPS	Global Positioning System
GPRS	General Packet Radio Service
IDE	Integrated Development Environment
GCC	GNU Compiler Collection
ATM	Automated Teller Machine
CLR	Common Language Runtime
GUI	Graphical User Interface
CV	Computer Vision

RGB	Red, Green and Blue
HSV	Hue, Saturation and Value
HSB	Hue, Saturation and Brightness
FAST	Features from Accelerated Segment Test
MSER	Maximally Stable Extremely Regions
SURF	Speeded Up Robust Features
SIFT	Scale-Invariant Feature Transform
FLANN	Fast Approximate Nearest Neighbor
XML	Extensible Markup Language
HARR	Alfred Haar
LBP	Local Binary Patterns
HOG	Histogram of Oriented Gradients
Mi-Fi	Mobile Wireless Fidelity

CHAPTER 1

OVERVIEW

1.1 Introduction

The major goal of robotics is to imitate human—like activities. Robots can solve several complex tasks without caring for the natural of the environment. All robots until this day are unable to self learn and creativity [1], so robots need pre programmed and sometimes need to be supervised and remote controlled or fully automotive by themselves or hyper. Remote activity of robots is very important because in some environment it is very dangerous or impossible to reach for replacing or adding on operation without risking human life because human soul is more precious than robots (Fig. 1).

Robotics involves three main processes [2]:
• Sense
• Think
• Act

• "Sense" is the technologies that demonstrate the innovative ways used by robots to collect data about the world, including machines with vision, motion detection, and ultrasonic mapping abilities.

• "Think" process includes many exhibits that explore how robots are programmed to process information and act accordingly, including the many facets of robot. Intelligence is basic for advanced and artificial programming. Intelligence systems simulate human thoughts and emotions.

• “Act” is an obvious feature of most of robots, it is the theme of "acting" that demonstrates how robots walk, roll, climb, fly, grasp and use tools, collect materials, and build structures.

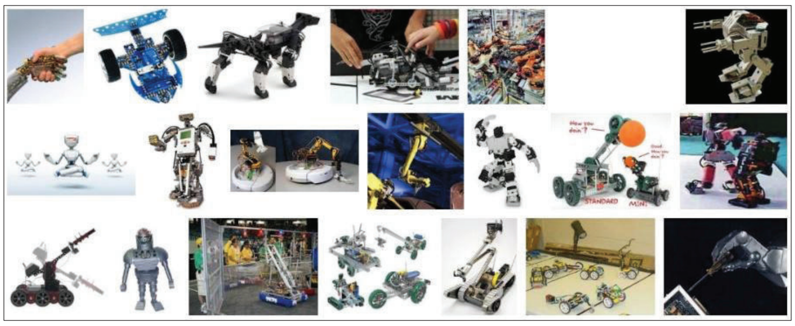


Figure 1: The robots

1.2 Definitions

The term robot is a mechanical or virtual artificial agent it is an electro-mechanical machine that is usually guided by a computer programmer or electronic circuitry [3].

Some robots are fully automatic control and they do not need any type of observation or control and named as autonomous robots, or other robots have some functions and need human control and some other automatic functions. This type is named semi-autonomous robots. Some others need fully remote control and without any human need, they are dump machines. This type is named non-autonomous robots [4]. When robots are autonomous they usually have less and fixed function like factory robots that can have less function as moving some objects from one place to another, and fixed function that grip the object from its accurate place. When autonomous robots have a lot of flexible functions they need complex programming and a lot of environments, and probabilities. The hardest one is when robots try to be human. Until now they never achieve that but they are on their way to do it.

Building all robots stared through from non-autonomous and programmed and tested in indoor environment like labs or workplaces then advance to semi-autonomous with testing outdoor environment. If robots need to be fully autonomous then the human controlling and observing is amputation from function to another with loop back checking advanced robots.

Platforms include two main types, station robots and mobile robots. Station robots have no capability to move from one place to another like arm robots used in medicine and constructions. Also we have mobile robots used for various tasks and include all environment operations like air, sea and ground. This robots are usually named as unmanned vehicles because human is not on board. These vehicles maybe observed and remotely controlled from far stations by human. Therefore, these vehicles should be in minimum size as much as possible and designed careless on creating internal environment for human. \since they are not ridden by human they cheap in their price.

We have three main environments: Air, sea and ground thus these unmanned vehicles are named as for ground “Unmanned Ground Vehicles” or for short “UGV”, and for air named “Unmanned Aerial Vehicles” or for short “UAV”, and for sea named “Unmanned Underwater Vehicles” or for short “UUV” [5] (Fig. 2).

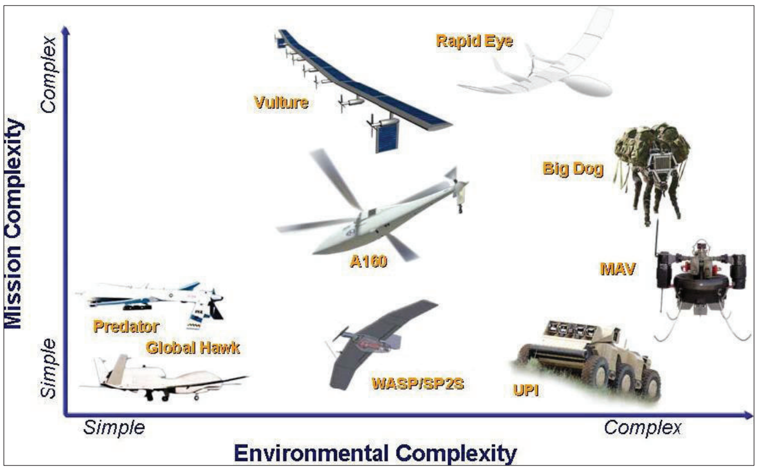


Figure 2: The unmanned vehicles

The “UGV” usually have mobile ability with specific platform changed according to terrain and natural operation. These platforms include three main types: Pods, wheel and tracks [6]. Every platform has unique factors like: Speed, complex and torque.

UGV platforms are important and well-selected to count on factors that can fit with UGV natural operation. Pod platforms are very useful for complex mobile similar to human nature but the process of building and balancing it is very complex. Pods are different in number of legs, bipeds (two legs) and quadrupeds (four legs) and hexapods (6 legs) are most popular pods, and increasing the pod legs resulted efficiency with complexity (Fig. 3).

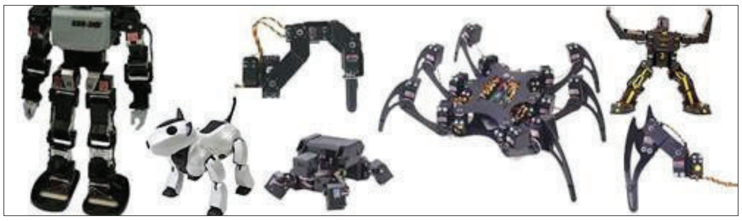


Figure 3: The pods platform

UGV track platforms are useful for torque factor that is used almost in all terrains, but have high energy consumption. They are commonly used by military and space explorer. Track's platform scheme is not static and changed according to the kind of the operation (Fig. 4).

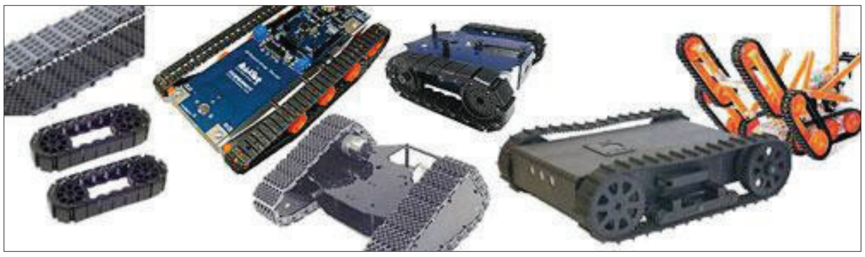


Figure 4: The truck platform

UGV wheel platforms are useful for general and normal use, easy building and cheaper than other types, and can go faster. The number of wheels is not static, more wheels will be better for critical situations when some wheel are broken, others wheels can be equivalent to the broken one (Fig. 5).

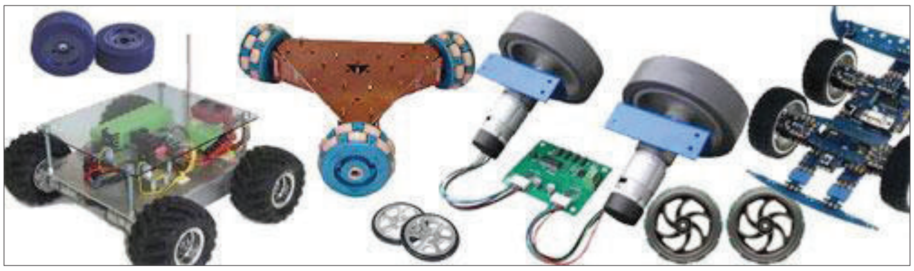


Figure 5: The wheel platform

1.3 Background and Early Systems

In 1921 the term “Robot” was first used by the writer Karel Capek for “Rossum's Universal Robots” to denote fictional automata. In 1941 the term “Robotics” had been used by the writer Isaac Asimov (Fig. 6) to describe technology of robots, in 1942 Asimov Write “Runaround” described three laws of robotics [7]:

1 - Robot must not harm human.
2 - Robot must obey human orders (except 1'st law).
3 - Robot must protect itself if the operator tries to contradict law 1 and 2.

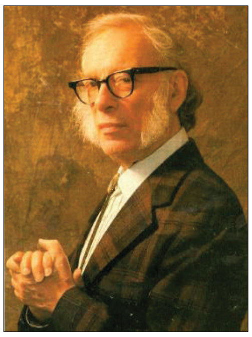


Figure 6: Isaac Asimov

In the same period of time, i.e. 1940's, UGV version was started to be developed. UGV first design is used for military purposes only for wars, especially in the Word Wide War 2 (WWII 1939-1945). TELETANK is the first UGV version designed by the Union of Soviet Socialist Republic (USSR) in 1930's in winter war against Finland and also in WWII against German in 1941 [8]. This UGV was non autonomous vehicle that remotely controlled from fix station like bunkers or mobile station like tank that ranges from 500 to 1500 km. This UGV functionality is firing bullets from guns machines, throwing heat or smoke bombs, mine explosive, also treatment with chemical weapons that can suffocate humans if they are done by themselves. These UGV have two possible frequencies to be controlled in order to avoid interference and jamming with 16 to 24 different commands. The "TT" was indicated to TELETANKS, for example TT-26 (Fig. 7).

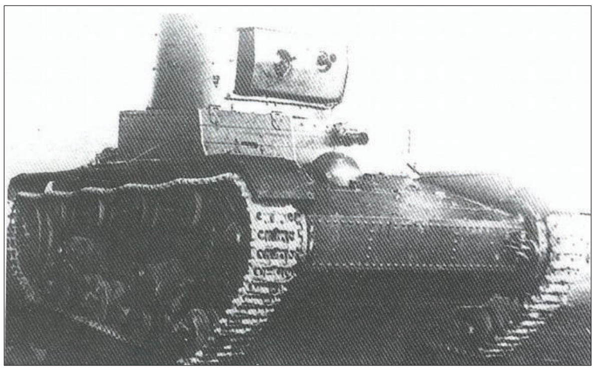


Figure 7: TT-26

In 1950s the atomic energy started to be used, the radioactive substances that come from atomic materials are dangerous for humans. This leads to the used station robots in the field of work. In 1951 France, Raymond GOERTZ designed remote controlled electronic arm that is used to manipulate radiation materials [9]. This arm is named E1. E1 arm which has two arms: Master and slave. The master arm is attached and controlled by human and slave arm repeats master arm action. The two arms connected by steel cables and pulleys (Fig. 8).

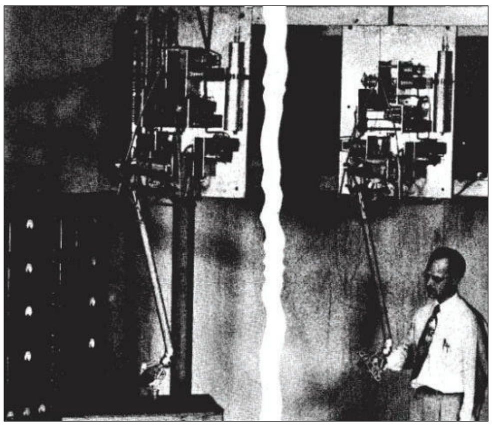


Figure 8: Raymond Goertz using E1 tele arm

The most high technology and humanoid robots are called "ASIMO" (Advanced Step in Innovative Mobility) (Fig. 9) [10], which are designed and developed by the Japanese company Honda and, completed and introduced to public in November 2000, and continue in developing and creating new versions. Honda celebrates the 10th anniversary of ASIMO in November 2010 and creating application called "Run with ASIMO" for smart phone to invoke virtually ASIMO walking [11].

Figure 9: Mr. Barack Obama play soccer with ASIMO

1.4 Fields of Applications

As we mentioned before robots have a lot of various types and forms and platforms, which lead to be used in various applications, Technology and human needs are increasing every day and this all leads to the capability of using robots in any application, and minimizing the jobs as we can decrease the cost.

Fields of applications include:

1.4.1 Submarines Field

Robots can be used for water and submarines with water proof designed. They can tolerate high sea pressure when dive too deep that is impossible for human. The best example of submarine robot is “Kingfish” which has the ability to dive 600 meters depths in water [12] (Fig. 10).

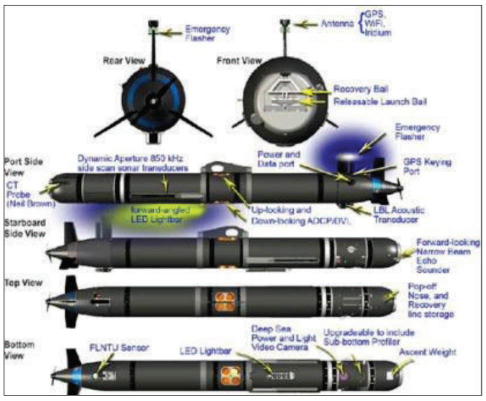


Figure 10: Kingfish UUV

1.4.2 Space Field

The space environment is lifeless, and sending humans to space is dangerous and costly process. National Aeronautics and Space Administration (NASA) sends a lot of robots to space, planets and moons [13] (Fig. 11).

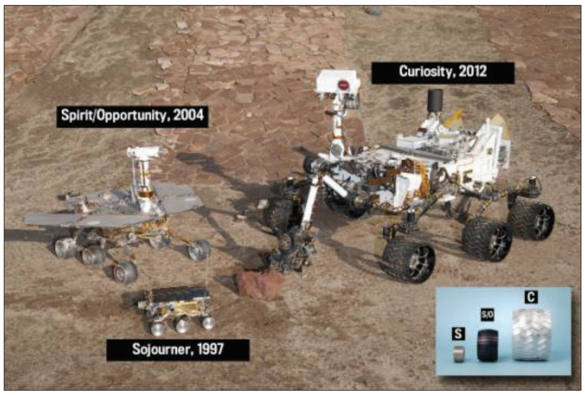


Figure 11: NASA space robots

1.4.3 Military Field

Human life is more precious than machines thus robots are mostly used in military operations such as, mine detecting and air surveillance photographs. The future of this application is prosperous and the government will develop this kind of technology nowadays [14] (Fig. 12).

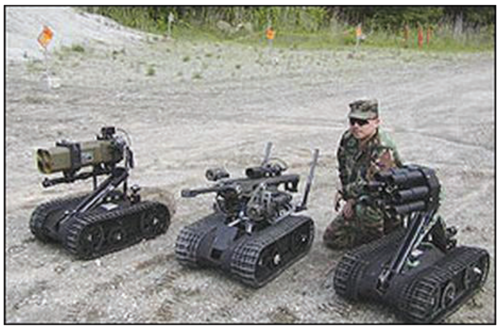


Figure 12: Military robots, swords units

1.4.4 Industry Field

A lot of Robots are used in factories because they have low error ratio and never become tired from repeated works, also these robots are purchased only for one time and need maintenance but their low cost for human employed insisted, no injury or harm if they work for long hours like humans and very fast. Robots in industry have few functions because every robot has a job to do repeatedly as it is preprogrammed and re-executed. KUKA arms are commonly used in Industry Field [15] (Fig. 13).

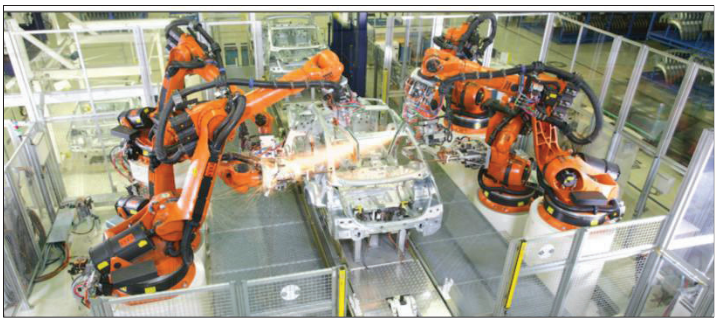


Figure 13: KUKA industry robots

1.5 Thesis Goal

The current thesis aim at building simple unmanned ground vehicle robot and controlling software, showing and discussing steps to achieve the goal, and showing the challenges and failure or success in the way to achieve that goal. This thesis also aims at adding some primary functions and building platforms that accept and modify new functions easily without redesigning.

Moreover, it aims at designing computable software to be used easily and widely, and using computer vision programs. Thus, this thesis is entitled a multifunction robot controlled by computer vision.

1.6 Thesis Outline

This thesis is divided into chapters. Every chapter completes the material presented by the previous one, in order to avoid any ambiguity.

• Chapter one includes basic information and introduction to thesis goal concerning building a robot and a type of robot, which is different from other types, to be ready to enter the other chapters, in detail.

• Chapter two describes my hardware work and building robots in detail and describes the system, it includes two parts the peripheral hardware and the core hardware. Each part studies in details.

• Chapter three describes the software and functions that can be produced from our system and what they are capable for with showing simple function examples.

• Chapter four discusses the conclusion, shows the power and weak point of system, and works challenges and future recommendations.

CHAPTER 2

DESCRIPTION OF HARDWARE SYSTEM

2.1 Introduction

There is no robot without hardware. The hardware gives the robot mobility and physical actions and feeds it with environment data. Selecting which hardware to be used will affect the whole robot system. The hardware of the robot can be divided into three parts: robot chassis, peripheral hardware and core hardware. Each section will be covered in detail. The Project robot is will result final hardware form (Fig. 14).

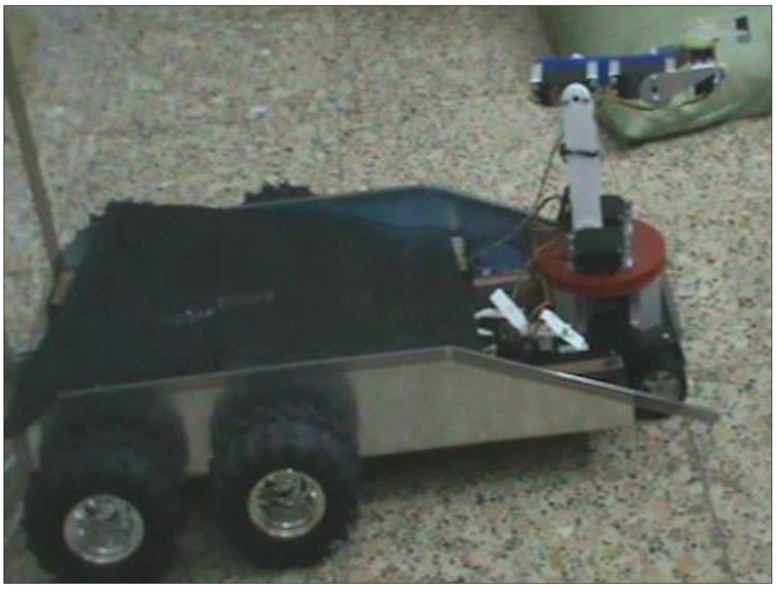


Figure 14: Overview of my project robot

2.2 Chassis of My Project Robot

Selecting the type of chassis is very important, using floppy metal gain light weight and losses firmness of chassis and using solid metals gain firmness and lost light weight, vise versa. Selecting wrong metals maybe makes the project to fail and redesigned from the beginning. Final chassis size is also important, optimize designed chassis size must be as small as possible. If chassis too small that will make problems of suffocation area for robot parts.

For project robot I select Aluminum (Al) polished plates that are used in some parts of home building like kitchens, they bay as plat [16], using special tools for cutting and denting (Fig. 15).

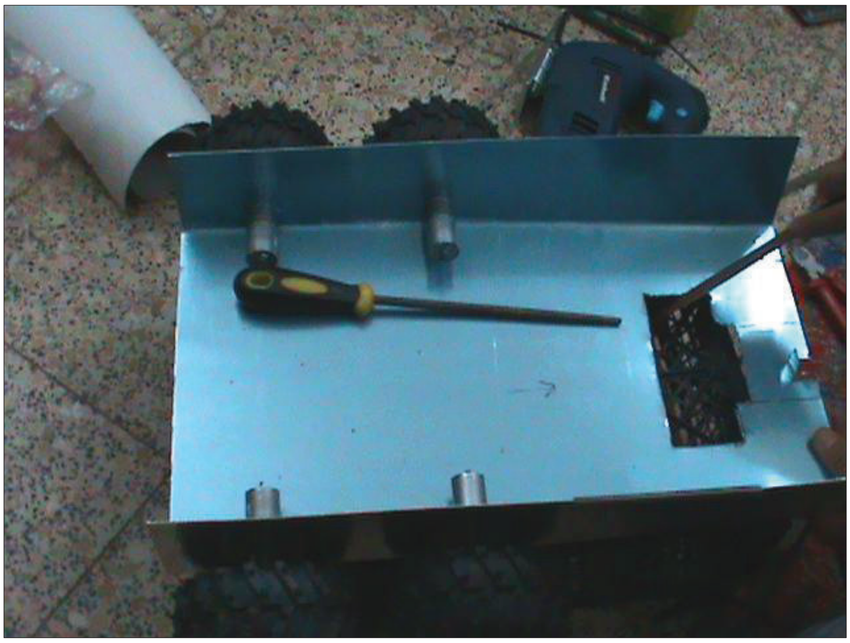


Figure 15: Chassis used in my project robot

2.3 Peripheral Hardware

Peripheral hardware in robots is the terminal electronic device that any robot has to apply input or output to environment, like mobility, applying some operations, fetching data vision from outside of robot. This type of hardware is not smart and unable to process any information. Without this type of hardware the robot converts to an empty box and loses connection with real world. Peripheral hardware can include:

2.3.1 Motors

Electric motors have a main role in robots designed, make a lot of tasks, like moving and doing custom action, using various types of motors, each motor has a place of use. Motors in simple form are electric coil with electric flow current inside two magnetic poles, using magnetic field with repulsion and attraction to obtain motion. This motion leads to infinite rotation [17] (Fig. 16).

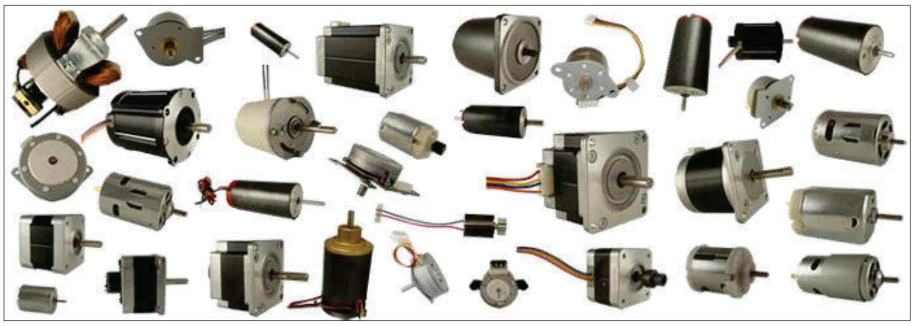


Figure 16: The motors

Usually, there are four common types of motor used in robots [18]:

1- Gear Motors

2- Brushless Motors

3- Servo Motors

4- Stepper Motors

Every Motor is characterized by specific factors, as follows [19]:

- Speed, measure by "rpm", times of motor rotation in one minute.
- Working electronic status, like voltage, current, power. Where the volt is static for each motor, power and current depend on moment of motion status.
- Torque, measure by "KP" (kilopond or kilo gram per 1 cm for motors) [20].
- Physical status, like motor dimension and weights.

"Gear Motors" have specific features that are used in mobility medium because they have high torque and high power consumption [21]. These motors have usually two pinout repressing power supply line for motor. In my project I used four gear motors connected with four wheels (Fig. 17).

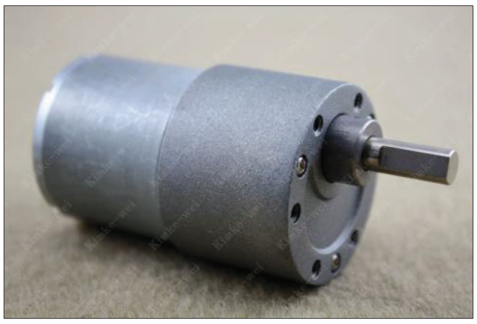


Figure 17: The gear motors

“Brushless Motors” have specific features that are used in flying and take off with middle power consumption. The rotation is very fast and high rotation in one second through using multi coils in motor. Normal motors contain two or four coils and this type of motors contain eight or more coils [22]. These types of motors are usually connected to blades that make object able to take off. These motors have three pinouts which are controlled by decoder circuit. In my project this type is not included (Fig. 18).

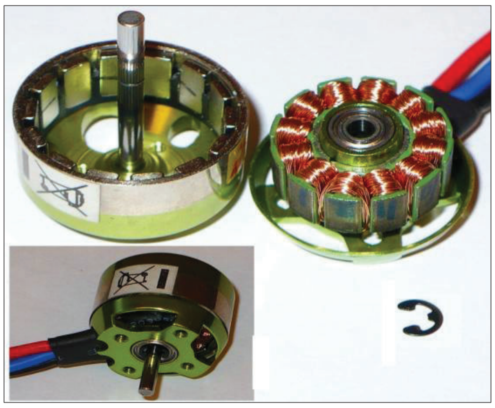


Figure 18: The brushless motors

For “Servo Motors”, the rotation of this type of motors is limited for 180 degree which means that these motors never rotated in continuous circle, so these motors will be usually used in custom actions and movements. In my work, I used these motors in robot arm and robot motion steering. These motors have three pinouts, two pinouts for the power supply and one pinout for degree rotation [23] (Fig. 19). This line is PWM.

PWM mean “Pulse-Width Modulation” it’s type of modulation that controls the width or duration of pulse [24], when PWM adjusted width of plus to 1% duty cycle the servo motor rotate to 1 decimal degree and reach to limit of far less of rotation, If PWM is 100% duty cycle will rotate to 180 decimal degree and reach to limit of far most rotation.

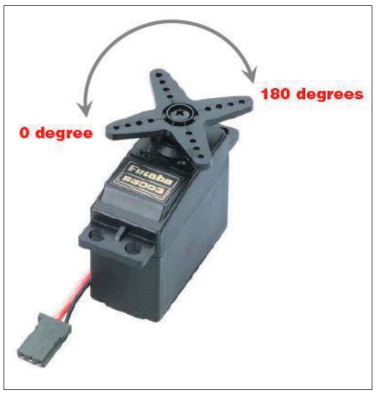


Figure 19: The servo motors

Servo motors, as mentioned above have three pinouts. Two pinouts represent power supply and one pinout is PWM. The following paragraph shows the color for each pinout and its purpose:

Pinout color: Yellow (Version 1) or White (Version 2) → PWM.
Pinout color: Red (Version 1 and Version 2) → Power (+5V DC).
Pinout color: Brown (Version 1) or Black (Version 2) → GND (Ground).

“Stepper Motors” are also brushless motors but they have at most four coils each coil has one line input, for example if we have stepper motor with four coils each coil have two pinouts, totally we have four multiply two equal eight pinout. This makes to overall motor controlling and lead to fine control in speed of rotation and fine stop. It is used a lot in robotics but in my project there is no need to use robot system (Fig. 20).

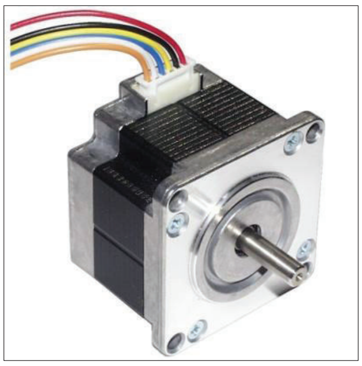


Figure 20: The steeper motors

2.3.2 Relays

Relays are majorly used in controlling tasks. Their benefit is to separate controlling from the device power supply, and known as driven circuit [25] (Fig. 21). There are different types of really, such as like working volts and amount of ampere that can be supported, usually in computer interfacing and robotics used relays of 5 volt because of core volt, and supporting ampere is different depending on its use. As far as the current project is concerned, supporting relays used to control robots wheels.

Figure 21: The relay

Relay scheme are almost the same. They have two pinouts which represent the magnetic coil. When there are currents in coil, the relay will be in action and when there are no currents the relay will be mute. The work volt of this coil is representing the working of the relay. The other pinout for relay is representing the way of controlling, usually has one way in simple relays or two or more. If we take one way as an example we will have three pinouts, one pinout is a common line and the other two points represent the “normally open line” and “normally closed line”. One pinout is always connected to the common line in no-action state and called “normally closed line", and the other line is unconnected and called "normally open line". In action state the operation will be visa versa (Fig. 22).

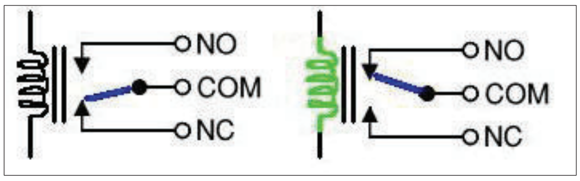


Figure 22: Relay scheme

2.3.3 Robot Arm

Robots arms have a lot of shape and a lot of types, it depending on its work and what is expected from arms to do. Therefore, we must focus on the arms that are commonly used in robots and also in my project robot system as well. The common arm must have claw to pick-and-place, also free rotation and tilt forward. This motion is done by a series of motors, especially servo motor with a strong shaft holding all motors together (Fig. 23).

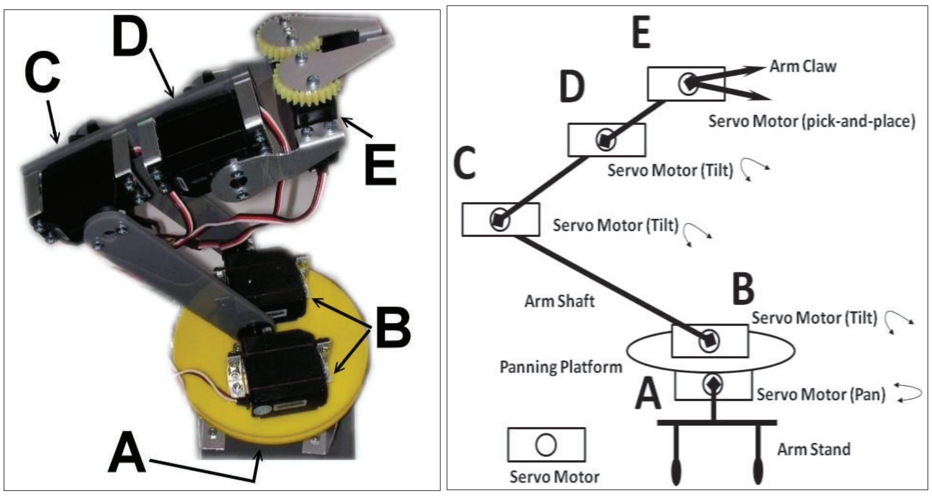


Figure 23: Robot arm with five servo motors

In the above figure we can see all five servo motors. For motor "A" it is responsible to rotate and pan whole arm 180 degree from far left to far right, and motor "B" is responsible to tilt the arm to front and rare, for motors "C" and "D" they are also responsible to tilt forward and backward but in fine steps that make the arm exactly approach to target object.

And for the last motor "E" it controls on arm claw to pick-and-place objects. All motors as we early state range from (1 to 180) degree, and adjusting this degree to specific degrees can make arm applying the procedure of jobs. Applying these jobs in sequence makes, the arms move as they want. Let us take an example of arm modes:

Note:
Style of examples will be as:
Name of Motor [A,B,C,D,E] (Degree of that motor from 1 to 180) → Action.

Motor "A" controlled on arm pan from left to right of arm, if
A(1) → Arm turn to far limit left,
A(180) → Arm turn to far limit right,
A(90) → Arm facing front middle of far left and far right and this always will be the robot start point (Fig. 24).

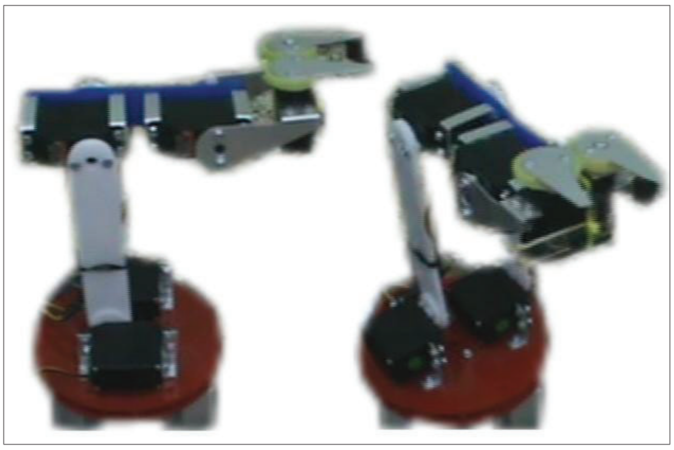


Figure 24: Arm panned to right by motor "A"

Motor "E" controlled on arm claw only, when
E(1) → claw is closed,
E(90) → claw is half open,
E(180)→ claw full open (Fig. 25).

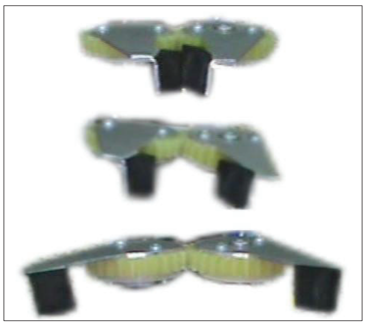


Figure 25: Arm claw from close to open state by motor "E"

The remaining motor will control the tilt arm from frontward to backward, we can obtain different patterns of arm orders, and but the common and most used patterns will be:

B(90),C(90),D(90) → "pistol mode" (Fig. 26).

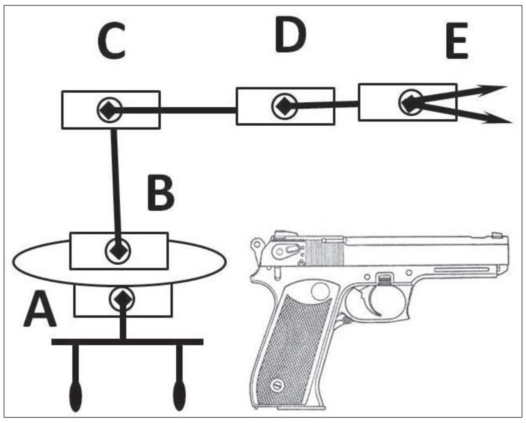


Figure 26: Arm in "pistol mode"

B(90),C(180),D(90) → "cobra mode" (Fig. 27).

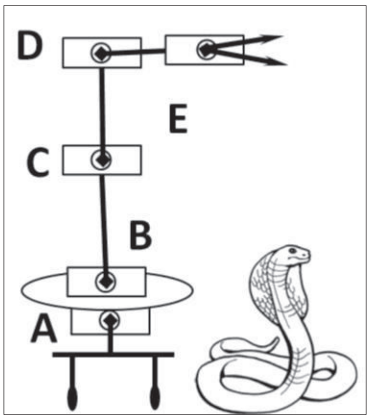


Figure 27: Arm in "cobra mode"

B(180),C(45),D(45) → “park mode” (Fig. 28).

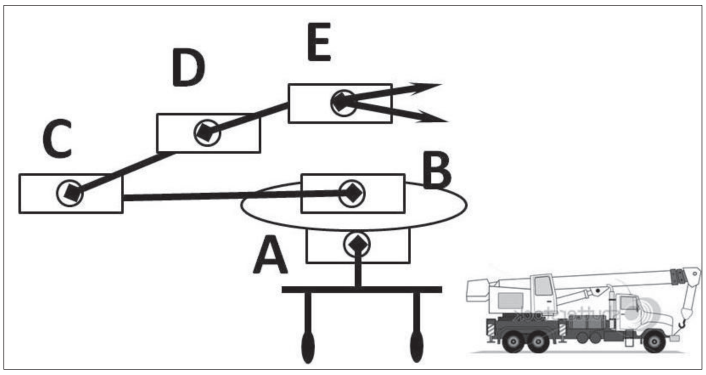


Figure 28: Arm in “park mode”

Robot arms have a number of concrete pinouts. Two pinouts represent power supply of arm (+, GND) as inputs to all motors they contain. And the pinout of each servo motor represents the PWM and we have five PWM inputs. Totally we have seven pinouts.

2.3.4 Power Supply of System (Battery)

Any robotic system will never be in action without power supply. There are a lot of power sources like nuclear, solar systems and petroleum, etc. but we need simple and efficient sources, therefore, the optimize selection is battery.

Batteries are simple, portable and easy to use. Also there are a lot of batteries, the most important one is the multi reachable battery. Famous reachable battery includes the following three types [26] (Fig. 29):

Lead-acid, Nickel–metal hydride (NiMH) and Lithium-ion (Li-Po or Li-ion).

For lead-acid battery it is well used in cars and generators. They have multi size and each size has specific Watt, measurement by Ah (Ampere per hour).
These batteries have good long hours of usage and very good for use in robotic but also have cons, which are heavy for small and middle size robots.

For NiMH battery, it is used in electronic device and some reachable batteries in toys. These batteries have a major problem known as "Dreaded Memory Effect" [27], which means when it loses their capacity for full recharging if they are discharged repeatedly and then recharge the same amount without overcharge before they have fully drained.

For Li-ion or know as lithium polymer battery, it is the best battery for using in electronic device such as mobile phones, characterized by high energy density, no memory effect, and only a slow loss of charge when they not in use.

These batteries commonly used in laptops, contained inside laptop battery cover small cells of Li-ion battery, each cell has static core volt, using it in serial order gaining increase in volts.

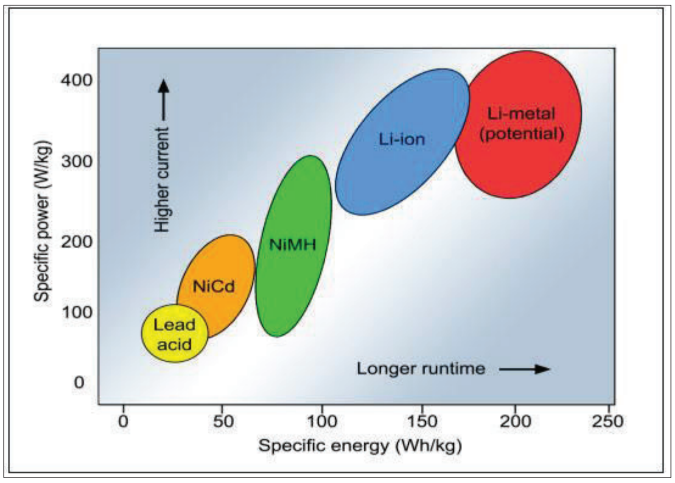


Figure 29: Comparison between three type batteries

For my project robot, I used Li-ion cells, their optimal choice, each cell has 3.7 volt and as we state above we can get multi amounts of volts when they are connected serial order that my project robot system needs multi volts, also I connect them in a parallel order to get backed power (Fig. 30).

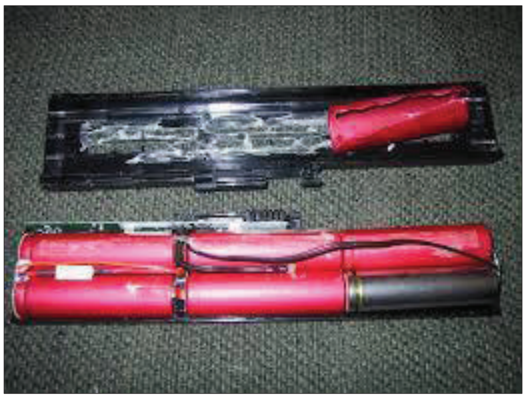


Figure 30: Cells of Li-ion

2.3.5 Camera

Most of robots contain cameras, and have data vision input to observe and apply procedures. There are hundred's types of camera and multi sizes, but we must be sure to choose the one which is combatable with its system. The core hardware plays a major role in the selection of the type of camera. In my project robot, USB camera like web camera is combatable with the core hardware. USB camera has four pinouts (in USB 1.x – 2.0), the two pinouts in edge is power supply (+,-) of camera and its 5 Volt. The other two pins represent data in and data out for the camera which is in super fast serial order sending [28]. In my project robot system, I used two cameras, one in front of the robot to see the front and the second is attached with robot arm claw to see the pick able object clearly (Fig. 31).

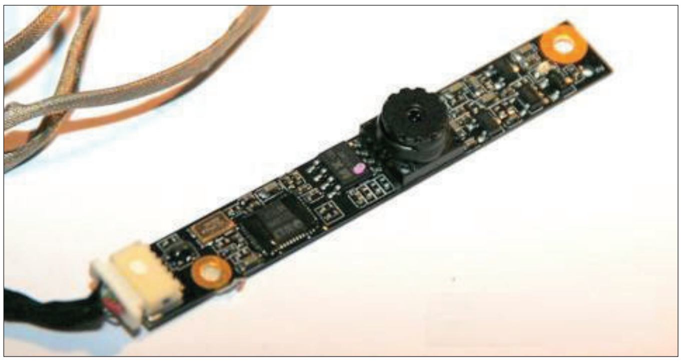


Figure 31: Internal web camera for laptop

2.3.6 Steering

Steering robot means rotation and guides it to a specific point, if no steering robot will turn left or right just in straight direction. Steering has a lot of methods. Choosing the method can affect robot mobility in a good way or in a bad way. There are three major types of steering method [29]:
Parallel wheel steering, all wheel steering and front wheel steering.

In parallel wheel steering there are four by four or six by six or more, wheels which are divided into two parallel part left side and right side, every side can rotate clock wise or counter clock wise and this movement lead the robot to go to different directions for each status (Fig. 32).

In This type of steering no wheels have angling, this leads to robust in wheel shaft construction but have some cons and this type of steering cannot do all types of rotation and losing soft steering. This type of steering is used in army vehicles and also in tracked robot platform.

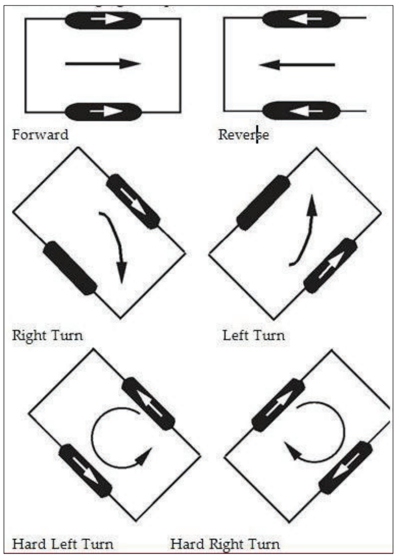


Figure 32: Parallel steering

In all wheel steering, all wheels come in action of steering. In this type of steering all wheels have mount of steer angle and also all wheels have specific rotation from other wheels. This action leads to gain soft steering and custom rotation and motion

(Fig. 33). This type of steering has cons, complexity of building and a lot of defects. This type of steering used in small and light robots which usually have four wheels.

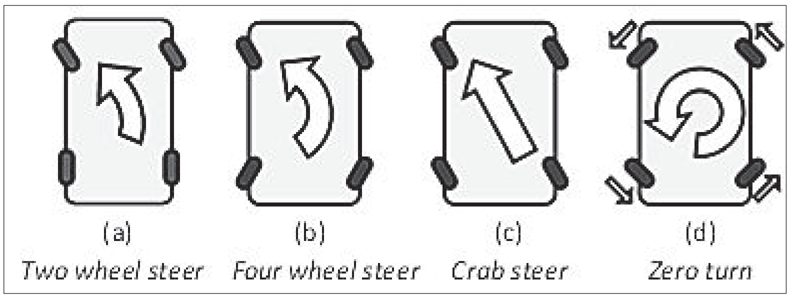


Figure 33: All wheel steering

In front wheel steering, wheels are divided into two sides, front and remaining wheels. The front wheels have the ability of steer angling while remaining wheels cannot angling and just can be used as be as driver wheels giving robot thrust of moving (Fig. 34). This type of steering has the same benefits of other two types of steering and easy implementation.

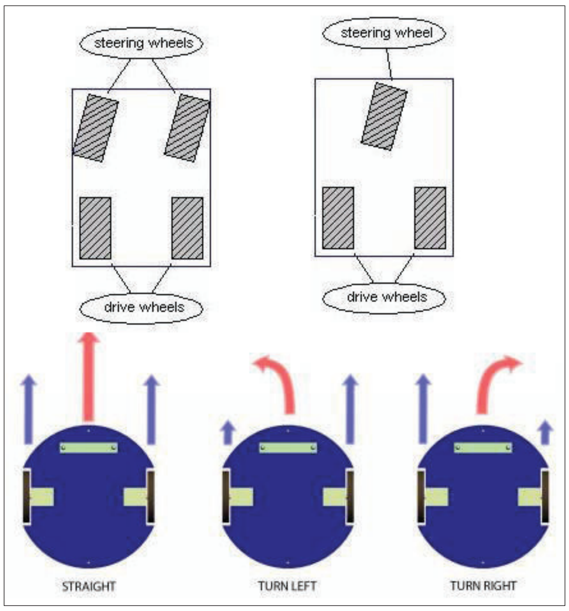


Figure 34: Front wheel steering

In my project robot system, I used front wheel steering, using two front wheels as steering and remains wheels as driver wheels that have big size four by four wheels and two small size wheels (Fig. 35).

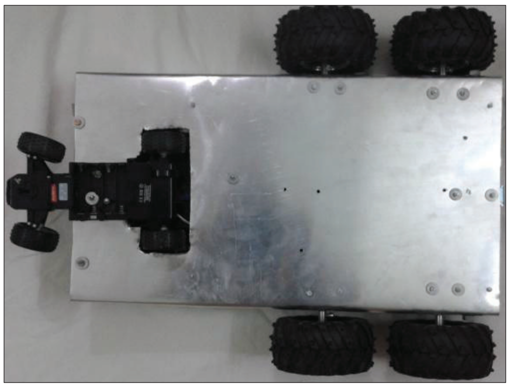


Figure 35: My project robot wheel system (bottom view)

2.4 Core Hardware

All peripheral hardware will be useless if there is no electronic board to control it, and connect between the operator and the peripheral hardware. This type of hardware must have the ability to connect and control motors and relays and other types of hardware. Simultaneously it must be specialist in operating system and programming language. These entire characteristic in one device is rare and semi impossible. The idea is to use two core boards, the first core device bridged to robot hardware and the second core board bridged to robot software, in the same time both core boards connect each other as well. In this case it is easy to find each core board characteristic alone and independent from each other. So we can divide core hardware, as follows:

2.4.1 Bridge to Hardware

In this section of bridge, we must select the best electronic board that can be connected easily with robot hardware as motors, relays and other peripheral devices and professional for control in electric input and output with compatible pinout. The best selection for these devices is “Single-Board Microcontroller” or knows as “SBM”. In markets now, there are a lot of SBM with different specifications like Atmel AVR, PIC, STM32 and ARM. Every SBM has pros and cons [30]. The best SBM is Atmel AVR which is represented by “Arduino” product.

Arduino is SBM contain microprocessor (single chip) with electronic card. This SBM is built in Italy with wide distribute and cheap cost, Arduino has multi product, we will focus on "Arduino Uno R3" (Fig. 36).

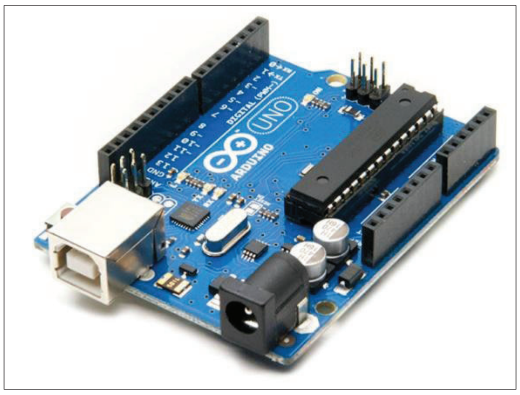


Figure 36: Arduino UNO R3

Arduino Uno R3 has multi pinouts. If we hold Arduino as aiming USB port to upper and Microprocessor chip to front, we see in the upper one USB type "B" jack (female) using for programming and uploading code, and one DC jack for power supply that can be in range 5 Volt to 12 Volt.

In the right side we see a group of pins, pins from 3 to 13 which can be used as digital input or output (core volt is 5 volt), pins that mark in tilde (~) (3,5,6,9,10,11) can also be used as analog output (from 0 volt to 5 volt) or used as PWM that control Servo Motors. Pin 13 is connected to internal small led that is used for test prosper. Pin groups in the left (A0 to A5) side are used for analog input (from 0 volt to 5 volt). The remaining of pins are used as multi volt source "3.3" Volt and ground signal "GND" and other pins are used for custom operation [31] (Fig. 37).

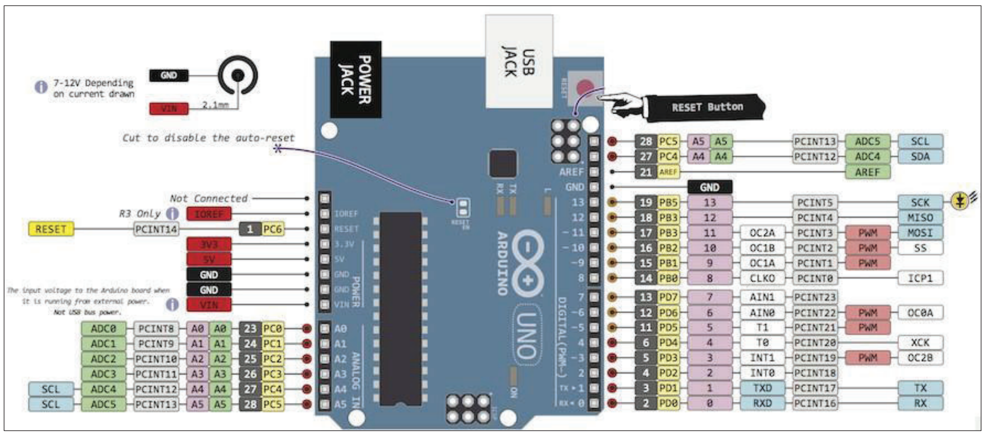


Figure 37: Arduino UNO R3 datasheet schema

2.4.2 Bridge to Software

In this section of bridge we must select electronic board that is professional with software and compatible with programming language and easy to be connected to networks and ability to remote. These types of boards must have an operation system inside and internal storage to save control code. These boards are named as "Single-Board Computer" or know as "SBC" [32]. In markets there are limited SBC that satisfy our requirements. Result of this research indicate that the best option is to use laptop mother board, getting rid of some extra part as cover, laptop screen, keyboard, DVD driver. While keeping of hard drive and battery. Using laptop board as SBC in robot has a lot of benefits, as uniform programming environment, and very compatible with a lot of interface like networks and internets (Fig. 38).

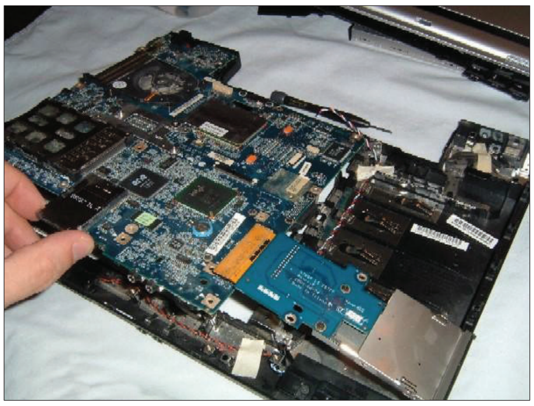
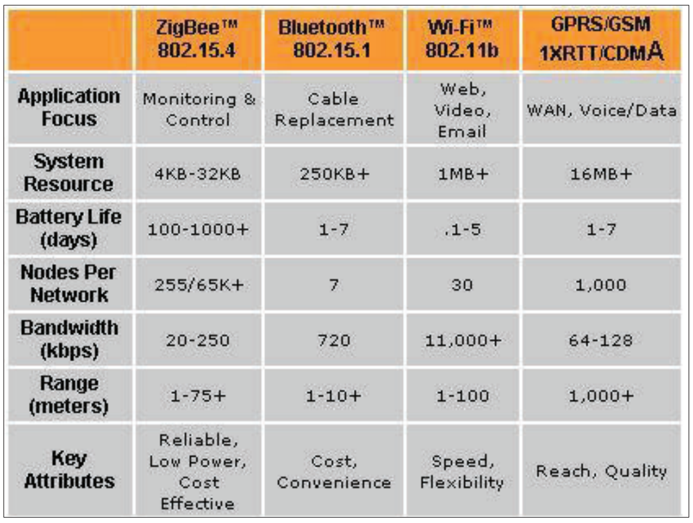


Figure 38: Laptop board using it as SBC

2.4.3 Linking Two Bridges

Arduino as SBM and laptop board as SBC, as the last step are two connected parts. Arduino is connected by USB port but also has extended ability to be connected with LAN, Bluetooth, Wi-Fi and ZigBee and GPRS [33], while laptops have the ability to connect most of the above ways. Every connection medium has pros and cons, and choosing the right one depends on the system structure itself [34] (Fig. 39).



	ZigBee™ 802.15.4	Bluetooth™ 802.15.1	Wi-Fi™ 802.11b	GPRS/GSM 1XRTT/CDMA
Application Focus	Monitoring & Control	Cable Replacement	Web, Video, Email	WAN, Voice/Data
System Resource	4KB-32KB	250KB+	1MB+	16MB+
Battery Life (days)	100-1000+	1-7	.1-5	1-7
Nodes Per Network	255/65K+	7	30	1,000
Bandwidth (kbps)	20-250	720	11,000+	64-128
Range (meters)	1-75+	1-10+	1-100	1,000+
Key Attributes	Reliable, Low Power, Cost Effective	Cost, Convenience	Speed, Flexibility	Reach, Quality

Figure 39: Comparison among connection mediums

2.5 Summary

The final robot hardware system of my project be include: Wheels with DC motors, arms with servo motors, steering with servo motors, relays as electronic switches, powers supply as a Lio-ion battery (laptop battery), Arduino as a single boarded microcontroller and laptop board as a single boarded computer.

CHAPTER 3

DESCRIPTION OF THE SOFTWARE SYSTEM

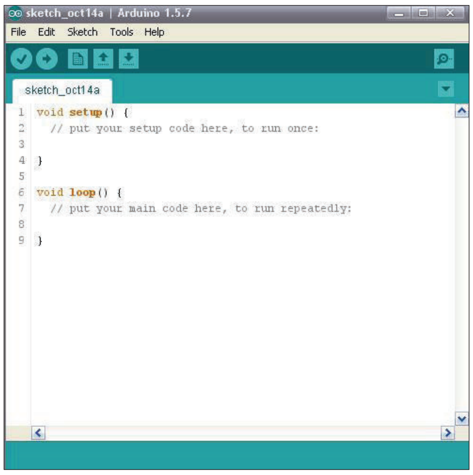
3.1 Introduction

In the previous chapter we talked about hardware and all accessories in detail, but none of them can achieve any object without software to control it. Robot software is complex and to make it easy we must first select the best suitable software and programming language. Moreover, we talked about peripheral robot hardware and core robot hardware, for peripheral hardware they do not accept any programming language just electric signal from core hardware. Core hardware is divided into two parts, SBM and SBC. Every part has its own software and programming language. We will study every part in detail.

3.2 Software in SBM

In the previous chapter we represented SBM as Atmel Microcontroller as "Arduino" product and we talked about board pinout. Now it is the time to program it. The first step is connecting Arduino with PC by using USB cable and software driver and programming IDE (Integrated Development Environment).

The official website for Arduino is "http://arduino.cc", to download the driver and programming IDE from "http://arduino.cc/en/Main/Software". The software package contains both driver and IDE. The latest version includes more Arduino products and more language code development. Below is the latest version which has multi operating systems like windows, Linux and MAC OSX. Then we select the type of software package as self installation or portable compressed file, best chooses is ZIP file (compressed file for non-administrator install). The best place extract ZIP file to last drives that does not contain an operating system and not risk or reformatting that drive. After we extract it we will see multiple folders, folder named “DRIVERS” contain driver software for Arduino and we can select this folder when the operating system asks for Arduino driver. Folder name “EXAMPLES” contains the example file that we can import and modify it to match our jobs. In root folder file name “ARDUINO.EXE”, this file opens Arduino IDE for programming and uploading code. Work file extension is “*.INO”. After running Arduino IDE we will see programming environment (Fig. 40).



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 40: Arduino IDE

We have some important commands in file menu. First we must choose which Arduino board we want to upload by following: “File Menu → Tools → Board”. Then selecting the number of port that Arduino is connected to PC by USB (communication port), it will auto detect by IDE they just need to confirm selection, because sometimes they have more than one board to connect, by following:
“File Menu → Tools → Port”. To start with any project some static steps:

1- Beginning: We have three choices: new project or load previous work or load example, and after finishing we have to save and close, all of this operation in:
“File Menu → File”.

2- Writing or editing codes.

3- Verifying: We check if the code has any errors by following:
“File Menu → Sketch → Verify / Compile”. If there is any error we should go back to step (2).

4- Uploading: We must be sure that Arduino is connected to PC and selecting the right board and right communication port, then we upload code to microprocessor ATMEL chip is existed that exist inside Arduino. This code has not volatile and does not need power supply to keep programming code and also re writable multiple time [35].

Arduino IDE use “AVR-GCC” as programming compiler [36]. Programming language similar to C++ style, Inside Arduino IDE we have two main functions: “Setup” and “Loop”.

“Setup” function is used to set variable and alter the option and runs once when Arduino boot is up. Where “Loop” function runs repeatedly in loop form until power supply is cut from Arduino.

In Arduino IDE we have a lot of commands, but in general we have the major operation that is always repeated in project, to make these commands run once we write them inside “Setup” function or want run repeatedly then write in “Loop” function.

Note:

- All codes ended by “;”.
- For comment inside code used “//” for one line, and “/*” “*/” for multiline.
- Capital and small letters sensitive.
- All below codes (SBM only) applied to “Arduino UNO R3”.
- Codes are between these comments **“//CODE STAR**” to “**//CODE END**” in Italic text font form.

We can divide operations into:

1- Digital output, used pins from (2 to 13):

```
//CODE START
DigitalWrite(4,1);
DigitalWrite(4,0);
/*
Where 4 is pins number and 0 as zero volt or 1 as 5 volt in digital output form
*/
//CODE END
```

2- Analog output, used only PWM pins (3, 5, 6, 9, 10, 11):

```
//CODE START
AnalogWrite(6,0);
AnalogWrite(6,128);
AnalogWrite(6,255);
/*
Where 6 is pin number and output is range from (0 to 255), 0 is zero volt, 128 is 2.5
volt and 255 is 5 volt
*/
//CODE END
```

3- Digital input, used pins from (2 to 13):

```
//CODE START
int d;
d = DigitalRead(4);
/*
 Where "d" is integer variable contain input digital value 1 or 0
*/
//CODE END
```

4- Analog input, used pins from (A0 to A5):

```
//CODE START
int a;
a = AnalogRead(A2);
/*
Where "a" is integer variable contain analog value from pin A2 in range from (0 to
1023) where 1023 is 5 volt.
*/
//CODE END
```

5- Delaying, if the program is too fast we can delay it:

```
//CODE START
delay(1000);
// where 1000 is millisecond (1000 millisecond = 1 second)
//CODE END
```

6- Mapping, remap the number from one range to another range, used for analogs range:

```
//CODE START
int a;
a = AnalogRead(A1);
float m;
m = map(a,0,1023,0,5);
/*
 m is float variable remap from (0-1023) range to (0-5) range, m now is true value of
input analog volt.
*/
//CODE END
```

7- Programming library, we can extend commands by using extra library:

```
//CODE START
// This command writes outside of functions.
#include <Servo.h>
// where "Servo" an example for represent library name
//CODE END
```

8- Servo motor controlling, used only PWM pins (3, 5, 6, 9, 10, and 11), we follow a fixed style:

```
//CODE START
#include <Servo.h>
// import library, this command write outside functions only.
Servo motorA;
Servo motorB;
// declaration for servo motor, this command write outside functions only.
motorA.attach(3);
motorB.attach(5);
/*
Attaching motors to destination pin, motors in this level start work and lock there
angle, this command write inside "Setup" function only.
*/
motorA.write(33);
motorB.write(134);
// move motors to destination angle, where angle between (0 to 180)
motorA. detach();
motorB. detach();
//release motors, motors in this level will free
//CODE END
```

The other operation in Arduino is including the eight operations above. Arduino is built as a self-sufficiency environment and instruction execution. These operations mean that instruction verify and compile then upload to chip, after that will repeat the execution of this code and not accept any order from PC any more, but for robotics operation we need real-time controlling. So we will use Arduino as a real-time interface board.

3.3 Communication SBM with SBC (SBM Side)

We mean to make Arduino ready to control by using SBC in any time for input or output operation without delays. In Arduino we have this capability by using same the communication when we upload codes before.

To make Arduino receive and send instruction with SBC in real-time we should use some fixed steps:

1- Initialize communication port (serial port) with specific data bit speed 9600:

```
//CODE START
Serial.begin(9600);
// this command will write on "Setup" function only
//CODE END
```

2- For sending data to SBC:

```
//CODE START
Serial.print("DATA1");
Serial.print("DATA2");
Serial.println("");
Serial.flush;
delay(2);
// where "DATA" replace by variables or any information want to send
//CODE END
```

3- For receive data from SBC:

```
//CODE START
int d=0;
if (Serial.available()) d = Serial.read();
/*
 when any data send and available for SBC will import it in "d" variable this
command will write in "loop" function only.
*/
if (d== X) // Instructions
if (d ==Y) // Instructions
if (d == Z) // Instructions
/*
X, Y and Z are numbers that SBC send to SBM. In this style we can do more than one
Instruction when data send from SBC
*/
//CODE END
```

To test these communications when we have no SBC available in mean time, we can open simulation window in Arduino IDE to substitute SBC by following:
"File Menu → Tools → Serial Monitor".

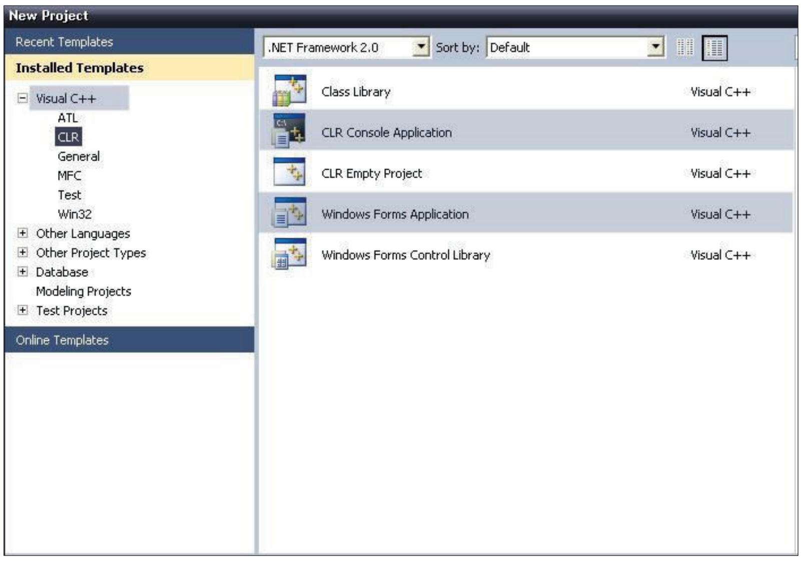
3.4 Software in SBC

In the previous chapter we talk about the best select for SBC, and we end to choose laptop motherboard as SBC board. Using of LMB (laptop mother board) has a lot of benefits in software part. The main benefit is very compatible in giving us the same environment that already given to students and a lot of people working on. Power supply is not problematic anymore in SBC because LBM already battery has supply, with great storage that may contain any operating software with any software programming language.

The best compatible software with conventional program language is "Windows" operating system like "Windows XP" and "Windows 7". Lower or upper version is not recommended. Also this system is used in ATM (Automated teller machine) [37] to simplify network and database connection. Normally we install software to LBM and we continue further to prepare programming software, for Windows operating system conventional programming language is Visual Studio (VS), version 2010 can install in Windows XP and Windows 7 [38]. Visual Studio also includes multi types as: Ultimate, Premium, Pro and Express. Where Ultimate is the best package but not frees and express is free and downloads from this link "http://www.visualstudio.com/downloads/download-visual-studio-vs#d-2010-express".

After fixed procedure to complete install. Visual Studio is multi platform programming language include: VBasic, VC#, VC++ and VF#. Every language can be connected connect to SBM by using fixed style in every each platform, but in the same time we need to add on programming library to extended program language capability like using computer vision operation. The best platform is flexible and accepts all kinds of libraries are VC++. So the final choose is "Visual C++ 2010 Ultimate".

In VC++ we have two types of codes managed and unmanaged [39], the difference between them are code managed by the third party or intermediate language when it compiled or not, unmanaged code is also called native code that CPU can directly execute to machine code [40], VC++ and has the ability to be in both forms, to be perfect connecting to add-on library we must sure that what we select should be managed by CLR (Common Language Runtime), when creating the new project we must select the following template: "Visual C++ → CLR". We would also select "NET Frame 2.0" to be compatible with early operating system, and then we decide whether our project needs graphical user interface (GUI) or not just as black windows. Template "Windows Forms Application" support GUI interface and "CLR Console Application" is without GUI (Fig. 41).


New Project

Recent Templates

Installed Templates

- Visual C++
  - ATL
  - CLR
  - General
  - MFC
  - Test
  - Win32
- Other Languages
- Other Project Types
- Database
- Modeling Projects
- Test Projects

Online Templates

.NET Framework 2.0 Sort by: Default

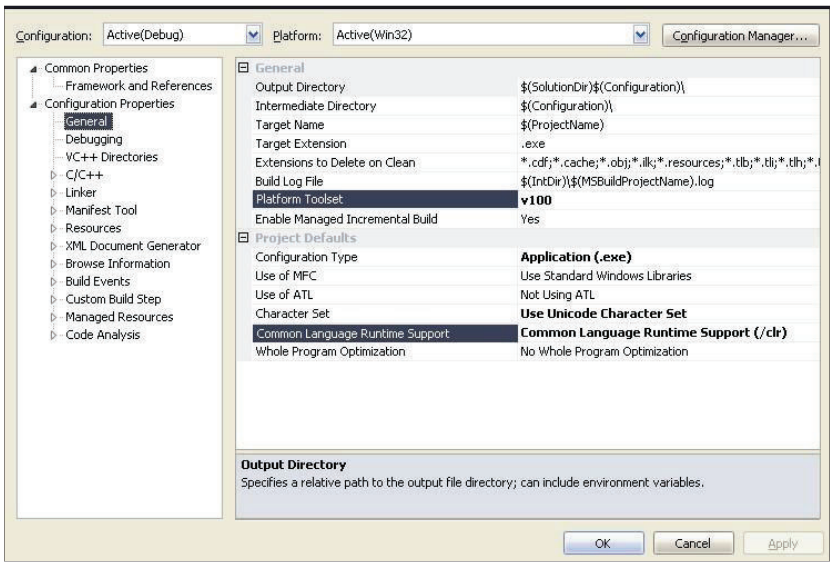
Class Library	Visual C++
CLR Console Application	Visual C++
CLR Empty Project	Visual C++
Windows Forms Application	Visual C++
Windows Forms Control Library	Visual C++

Figure 41: Visual C++ 2010 ultimate, “selecting template”

When we create consoled or windows formed application we need to prepare it to accept add-on library. These steps is fixed and done once for each one project, and the steps are:

1- Press ALT+F7 inside environment of project, this will show project configuration properties.

2- In “General” section highlighted setting must be changed, as shown below (Fig. 42).


Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

- Common Properties
  - Framework and References
- Configuration Properties
  - General
  - Debugging
  - VC++ Directories
  - C/C++
  - Linker
  - Manifest Tool
  - Resources
  - XML Document Generator
  - Browse Information
  - Build Events
  - Custom Build Step
  - Managed Resources
  - Code Analysis

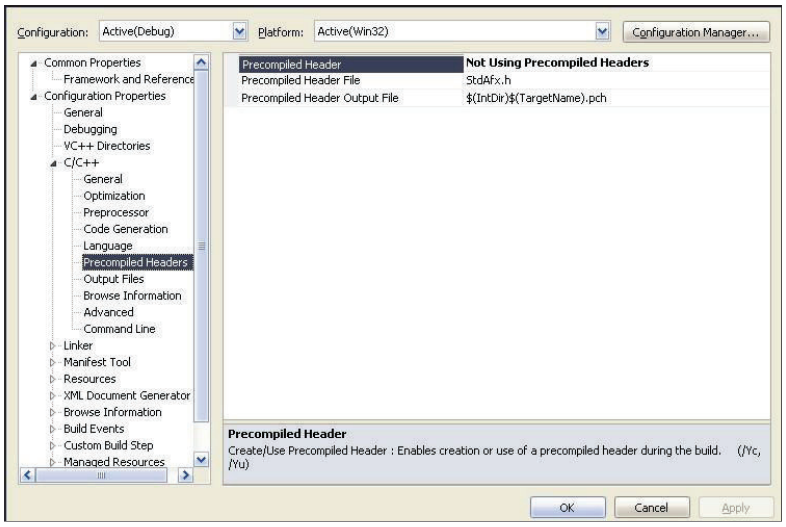
General	
Output Directory	$(SolutionDir)$(Configuration)\
Intermediate Directory	$(Configuration)\
Target Name	$(ProjectName)
Target Extension	.exe
Extensions to Delete on Clean	*.cdf;*.cache;*.obj;*.ilk;*.resources;*.tlb;*.tli;*.tlh;*.
Build Log File	$(IntDir)\$(MSBuildProjectName).log
Platform Toolset	**v100**
Enable Managed Incremental Build	Yes
Project Defaults	
Configuration Type	**Application (.exe)**
Use of MFC	Use Standard Windows Libraries
Use of ATL	Not Using ATL
Character Set	**Use Unicode Character Set**
Common Language Runtime Support	**Common Language Runtime Support (/clr)**
Whole Program Optimization	No Whole Program Optimization

Output Directory
Specifies a relative path to the output file directory; can include environment variables.

OK Cancel Apply

Figure 42: Project configuration properties, “general”

3- In “Precompiled Headers” section highlighted setting must be changed, as shown below (Fig. 43).



Precompiled Header	**Not Using Precompiled Headers**
Precompiled Header File	StdAfx.h
Precompiled Header Output File	$(IntDir)$(TargetName).pch

Precompiled Header
Create/Use Precompiled Header : Enables creation or use of a precompiled header during the build. (/Yc, /Yu)

Figure 43: Project configuration properties, “precompiled headers”

After that the project is ready to accept, add-on library, and this is done once for every new project.

3.5 Communication SBC with SBM (SBC Side)

In the previous section we prepared SBM to accept input and output from SBC, now it is the time to prepare SBC to send and receive commands to SBM using communication port, all code writings will be on base file (the same name of the project is named with “.CPP” extension), following the fixed steps below [41]:

1- Open communication port from SBC side:

```
//CODE START
System::IO::Ports::SerialPort^ link;
link = gcnew System::IO::Ports::SerialPort;
link->PortName= "COM4";
```

```
link->Handshake= System::IO::Ports::Handshake::None;
link->BaudRate = 9600;
link->Parity= System::IO::Ports::Parity::None;
link->StopBits = System::IO::Ports::StopBits::One;
link->DtrEnable = false;
link->RtsEnable = false;
link->Open();
/*
Where "4" in "COM4", represent the number that SBM used it as communication
port
*/
//CODE END
```

2- Sending data from SBC to SBM will be as:

```
//CODE START
link->Write("DATA");
// where "DATA" represent output data to SBM
//CODE END
```

3- Receiving data from SBM will be as:

```
//CODE START
String^  k;
k = link->ReadLine();
// where k is system string variable
//CODE END
```

4- Closing port when all operations would finish:

```
//CODE START
Link->Close();
//CODE END
```

We can also precede with fixed protocols between SBC and SBM and concurrent both sides with the same command.

3.6 OpenCV (Add-On Library)

The above section explains in detail the connection between control board and robot device and how, this path was lead in. Now the other path must lead out, mean the connection between robot and outside environment by using computer vision operation to feed robot with images and video. The difference between image processing and vision processing are shown below [42] (Table 1):

Operation	Image Processing	Vision Processing
Input	Raw Image(Source)	Enhancement Image
Procedure	Smoothing, Sharpening, Contrasting and Stretching	Shape and Object Recognition, Training and Learning
Output	Enhancement Image	Inference, Decision and Action

Table 1: Difference Between Image Processing and Vision Processing

Therefore, to apply some procedures in vision processing we need first to go through from image processing first. Vision processing is called for computer vision and machine vision, the two vision processes are similar and hard to find difference in both methods, therefore, a lot of people mixed between them and the difference can be as in the above table [43] (Table 2):

Feature	Machine Vision	Computer Vision
Natural usage	Practice	Academic
Cost	Critical Concern	Minor Concern
Dedicated electronic hardware usage	Yes, Very Likely	No
None Algorithm solution usage	Yes	No
Input Data	Real Time Inputs	Computer File
Knowledge of human vision influences system design	Most Unlikely	Very Likely
Nature of Solution	Satisfactory Performance	Optimal Performance
Output Data	Simple signal to control external equipment	Complex

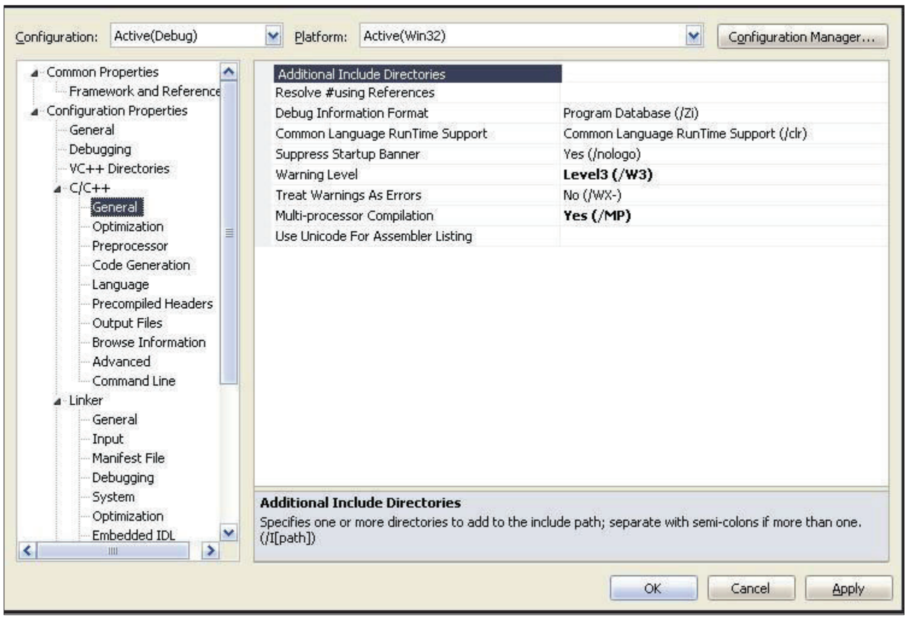
Table 2: Difference Between Machine Vision and Computer Vision

In VC++ Using original command for computer vision is very hard and very slow processing. The best thing is to add-on library to extend VC++ ability, to gain speed and uniform style programming. The best library to achieve it is OpenCV, OpenCV mean open source computer vision developed by Intel Russia research center in Nizhny Novgorod, and now supported by Willow Garage and Itseez. It is free to download and install from "www.opencv.org". Setting up OpenCV in VC++ is not easy and a lot of people have problems and there are many ways to setting it up.

The best and simple way is to follow the fixed stable for any new project (after prepare it for CLR):

1- Press ALT+F7 inside environment of Project, this will make to show project configuration properties.

2- In “C\C++ → General” section highlighted setting must be added with new settings, which are shown in the following figure (Fig. 44):



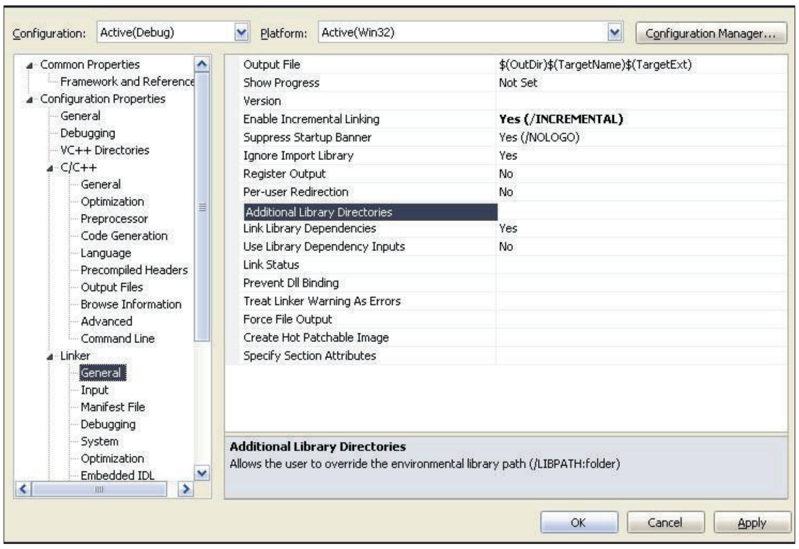
Additional Include Directories	
Resolve #using References	
Debug Information Format	Program Database (/Zi)
Common Language RunTime Support	Common Language RunTime Support (/clr)
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	No (/WX-)
Multi-processor Compilation	**Yes (/MP)**
Use Unicode For Assembler Listing	

Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])

Figure 44: Project configuration properties for opencv, “C\C++ general”

Item Add: header folder path “*.H” and “*.HPP”, that are found in *“[PATH]\opencv\build\include”.*

3- In “Linker → General” section highlighted setting must be added with new settings, which are shown in the following figure (Fig. 45):



Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	**Yes (/INCREMENTAL)**
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	Yes
Register Output	No
Per-user Redirection	No
Additional Library Directories	
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Link Status	
Prevent Dll Binding	
Treat Linker Warning As Errors	
Force File Output	
Create Hot Patchable Image	
Specify Section Attributes	

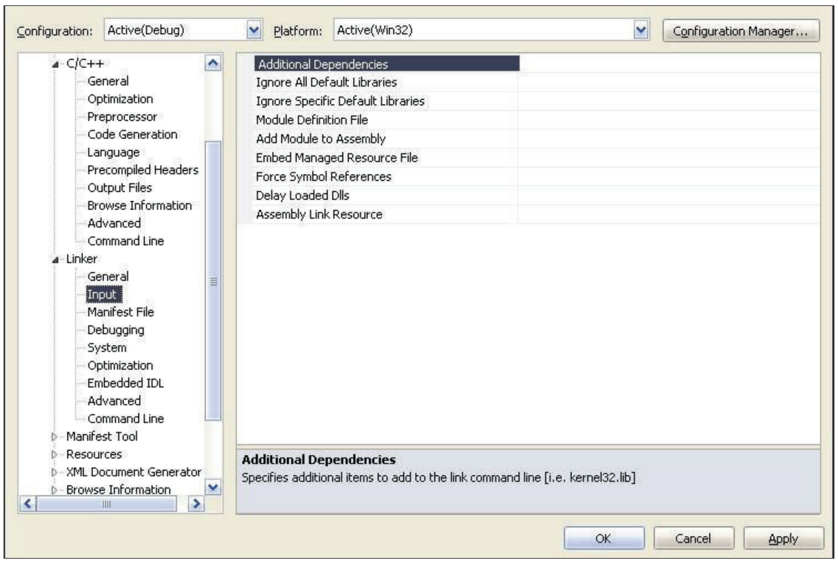
Additional Library Directories
Allows the user to override the environmental library path (/LIBPATH:folder)

Figure 45: Project configuration properties for opencv, “linker general”

Item add: Static library folder path “*.LIB”, that found in *“[PATH]\opencv\build\x86\vc10\lib”.*

Where “x86” for 32bit operating system and “vc10” for VC++ 2010.

4- In “Linker → Input” section highlighted setting must be added with new settings, which are shown in the following figure (Fig. 46):



Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

Additional Dependencies	
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link command line [i.e. kernel32.lib]

Figure 46: Project configuration properties for opencv, “linker input”

Item add: Static library files "*d.LIB", that found in
"[PATH]\opencv\build\x86\vc10\lib".

Where "x86" for 32bit operating system and "vc10" for VC++ 2010.
Every LIB file name that is ended with "d" letter must be written in the highlighted setting. The files have fixed style:
("opencv_")(LIBRARY NAME)(Version Number)("d").("LIB").

For current stable version number when this thesis is written is 2.4.9 so the files will be: (Note: files must separate by ";")
"opencv_calib3d249d.lib;opencv_contrib249d.lib;opencv_core249d.lib;opencv_features2d249d.lib;opencv_flann249d.lib;opencv_gpu249d.lib;opencv_highgui249d.lib; opencv_imgproc249d.lib;opencv_legacy249d.lib;opencv_ml249d.lib;opencv_nonfree249d.lib;opencv_objdetect249d.lib;opencv_ocl249d.lib;opencv_photo249d.lib;opencv_stitching249d.lib;opencv_superres249d.lib;opencv_video249d.lib"

5- Dynamic library "*d.DLL" that is ended with "d" must be attached with the project current folder to reach within the project code or a copy of these files to: "Windows → System32" folder, these library files that are found in
"[PATH]\opencv\build\x86\vc10\bin".

Where "x86" for 32bit operating system and "vc10" for VC++ 2010
The files have fixed style:
("opencv_")(LIBRARY NAME)(Version Number)("d").("DLL").

6- The project file coding must be written in the top of all commands calling for header file by using "*#include*" command like:

//CODE START
```
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
```

```
#include <opencv2/nonfree/nonfree.hpp>
#include <opencv2/nonfree/features2d.hpp>
#include <opencv2/legacy/legacy.hpp>
#include <opencv2/video/video.hpp>
#include <opencv2/ml/ml.hpp>
#include <opencv2/objdetect/objdetect.hpp>
#include <opencv2/flann/flann.hpp>
#include <opencv2/calib3d/calib3d.hpp>
//CODE END
```

Note:
".H" and ".HPP" files are header files that contain a definition of the function that points to ".LIB" files, ".LIB" files known as static library contain body of function compatible only with specific program language (OpenCV ".LIB" files compatible with VC++), it is called once at compiling and combine with project. ".DLL" files known as dynamic library is compatible with almost all programming language, it is called in run-time period and not combine with the project therefore must be kept attached with the project.

To ensure adding OpenCV is successful, we will execute these sample codes that import image and show it:

```
//CODE START
//Includes written here
int main(void)
{
cv::Mat k;                          //creating matrix image file
k=cv::imread("c:\\test.jpg");  //path and name image file (must exist)
cv::imshow("test",k);        //showing image file with test name window
cv::waitKey();                 //press key to end project
}
//CODE END
```

If windows appear with the above specific above image, all things are alright and we can keep moving.
Also in
“[PATH]\opencv\build\doc”,
There are very good documentation material and user guide.

3.7 Image Structure in OpenCV

Image in OpenCV is represented by a matrix dimension of matrix image size. The Matrix image dimension is represented by [44]: (With different calls names)

1- N = Columns = Width = Horizontal = X = Left to Right Range of image.
2- M = Rows = Height = Vertical = Y = Top to Bottom Range of image.
3- K = Layer or Channel of Image (only in color image).

The value of this image matrix depends on image type, in monochrome image there are only white or black (pixel spot or empty) and there is no need for the third dimension(K), the value of matrix will be only 1 or 0 (255 or 0 in OpenCV) where 1 or 255 is white and 0 is back.

In gray scale image there are also the same two dimensions, but the value ranges from 0 to 255, where the color is scaled as old films. Both images are symbolized by “*CV_8UC1*” that mean 8bit (1Byte) unsigned integer (0 to 255) with one channel (no third layer):

```
//CODE START
//OpenCV Code, create gray or monochrome black image (zero image) with 800x600
// dimension
cv::Mat gray;
gray = Mat::zeros(Size(800,600), CV_8UC1);
// access to specific pixel at location 20 row, 50 column
int pix=0;
pix = gray.at<unsigned char>(20,50);
```

```
// change to specific pixel at location 20 row, 50 column
int pix = 33;
gray.at<unsigned char>(20,50) = pix;
//CODE END
```

In color image the structure is a bit different because colored image file consists of three source color: red, green and blue. Every source color have ranges from (0 to 255, 256 unique pixel) so the number of the possible color will be 256 * 256 * 256 = 16777216 unique color, this image is known as RGB image (every source color first latter). The third dimensions represent the layers of the source code, the first layer is (Layer 2) is Red, the second layer (layer 1) is green and the third layer (layer 0) is blue [45]:

```
//CODE START
//OpenCV Code, create color empty image with 800x600 dimension
cv::Mat rgb;
rgb = Mat::zeros(Size(800,600), CV_8UC3);

// access to specific pixel at location 20 row, 50 column
int red_pix=0;
int grn_pix=0;
int blu_pix=0;
red_pix = rgb.at<cv::Vec3b>(20,50)[2]; //Red pixel
grn_pix = rgb.at<cv::Vec3b>(20,50)[1]; //Green pixel
blu_pix = rgb.at<cv::Vec3b>(20,50)[0]; //Blue pixel
// change to specific pixel at location 20 row, 50 column
int red_pix=40;
int grn_pix=6;
int blu_pix=248;
rgb.at<cv::Vec3b>(20,50)[2] = red_pix; //Red pixel
rgb.at<cv::Vec3b>(20,50)[1] = grn_pix; //Green pixel
rgb.at<cv::Vec3b>(20,50)[0] =  blu_pix; //Blue pixel
//CODE END
```

3.8 Vision Input in OpenCV

OpenCV procedure can apply only to image files, therefore if we need to procedure live video or storage video we must convert to sequence of image [46], in OpenCV there are simple commands to do that:

```
//CODE START
// Load Image from storage
cv::Mat img;                    //creating matrix image file
img=cv::imread("c:\\pic.jpg");  //path and name image file (must exist)

// Show Image
cv::imshow("myproj",img);       //showing image file with "myproj" name window
cv::waitKey();                  //wait to show image, non present number will wait until
                       //key press, if number represent will wait to specific milliseconds


// Save Image to storage
cv::imwrite("c:\\pic2.jpg", img );



// Getting sequence image from live video (web cam)
cv::VideoCapture vid;  //Create video stream
cv::Mat img; //Create Image for Stream
vid.open(0);       //Open live video stream from camera id 0 (any camera)
while(1) //DO
{
vid >> img;  //Send live video stream to Image, now all procedure will on this file
// Procedures will here
} // LOOP
vid.release(); //Close stream when we finish
```

```
// Getting sequence image from storage video (video file in drive)
cv::VideoCapture vid;  //Create video stream
cv::Mat img; //Create image for stream
vid.open("c:\\video.avi"); // Open video stream from storage path (file must exist)
vid.set(CV_CAP_PROP_POS_FRAMES,0);   //Start from frame 0
while(1) //DO
{
vid >> img;
// Procedures
} // LOOP UNTIL (img.data == NULL), video file reach to end
vid.release(); //Close stream when we finish
```

//CODE END

3.9 Object Representation in OpenCV

Object in OpenCV is a multi type data related to something. This set of data refers to the details of that object with various data type and this data can extract it from raw data like raw image that is fed by live camera. We can create the framework of this object with specific data model, in programming language and call it structure data or record data. In VC++ we can create this type of data by command "*struct*" [47]. Therefore the object code will be: (Note: must write this code above main function)

//CODE START

```
struct obj
{
        int id;
//Object identify number (ID), we can put any number to suit program

        cv::String name; // Object name, any name inside  " "
        int loc_x; // location of object in raw image (horizontal location)
        int loc_y; // location of object in raw image (vertical location)
        int shf_x;  //displacement horizontal location from center of image
        int shf_y; //displacement horizontal location from center of image
        int area_x1; //  area location from x1 in horizontal space
        int area_x2;  // area location from x2 in horizontal space
        int area_y1;  // area location from y1 in vertical space
        int area_y2; //  area location from y2 in vertical space
        int hgt;  //hight of object
        int wdt; //width of object
        int size; //size of object area (hgt * wdt) or diameter
        int raw_hgt; //hight of raw image that object found in
        int raw_wdt; //width of raw image that object found in
        cv::Mat img; //image of object
        int found;
/*
Found variable is very important indicate the object has been found last time or  not,
if found = 0 mean the object is disappear and all above information is for object is
seen last time, if found = 1 then object is available and on sight
*/
};
obj myobj;   // create object name "myobj" using above structural
//CODE END
```

We can access the object structure by using "." like *myobj.name*, "*myobj.found*", "*myobj.size*", and etc. Most information feed objects inside manipulating function in robot system we can enumerate the major object manipulating functions that will be followed below.

3.10 Function: Find Laser Spot Using RGB Manipulating

In this function we can try to find green laser beam that can be highlighted on wall or floor in front of the robot system. Laser is pure green color while other source color is less (red and blue), this will be our key to find, if Green ranges from (0 to 255) then the most high will be above 128, while red and green are almost low and will be under 128, then we must find where this spot light dense (spot dense mean is focus or gather target pixel or true pixel in spot of image) will find in spot dense procedure location and size of target object.

```
//CODE START
int mf_find_spot(cv::Mat rawimg,obj &myobj,int red_low=0,int red_hgh=255,int
grn_low=0,int grn_hgh=255,int blu_low=0,int blu_hgh=255)
{
int lfts = rawimg.cols;
int tops = rawimg.rows;
myobj.raw_wdt = lfts;
myobj.raw_hgt = tops;
cv::Mat maskimg = cv::Mat::zeros(cv::Size(myobj.raw_wdt, myobj.raw_hgt),
CV_8UC1);

int Redp=0;
int Grnp=0;
int Blup=0;
int ilft=0;
int itop=0;
int flag_red=0;
int flag_grn=0;
int flag_blu=0;
int fnd=0;
myobj.found=0;
for (ilft = 0; ilft < lfts; ilft++){
for (itop=0; itop < tops; itop++){
Redp = rawimg.at<cv::Vec3b>(itop,ilft)[2];
Grnp = rawimg.at<cv::Vec3b>(itop,ilft)[1];
Blup = rawimg.at<cv::Vec3b>(itop,ilft)[0];
flag_red=0;
flag_grn=0;
flag_blu=0;
if ( (Redp>=red_low) && (Redp<=red_hgh) ) flag_red=1;
if ( (Grnp>=grn_low) && (Grnp<=grn_hgh) ) flag_grn=1;
if ( (Blup>=blu_low) && (Blup<=blu_hgh) ) flag_blu=1;
if  (flag_red+flag_grn+flag_blu >= 3)
{
fnd=fnd+1;
maskimg.at<unsigned char>(itop,ilft)=255;
}
else maskimg.at<unsigned char>(itop,ilft)=0;
}
}
if        (fnd > 0)
{
int x=0; int x1=0; int x2=0;
int y=0; int y1=0; int y2=0;
mf_find_density(maskimg,x,y,x1,x2,y1,y2);
mf_inf2obj(myobject,x,y,0,0,x1,x2,y1,y2,0,0, maskimg);
return 1;
}
return 0;
}
//CODE END
```

Function “***mf_find_density***” find where “true” or “1” point is gathered using Distance Transform “chessboard” distance metric [48]:

```
//CODE START
int mf_find_density(cv::Mat mask,int& x, int& y, int &x1,int& x2,int& y1,int& y2)
{
int n = mask.cols;
int m = mask.rows;
cv::Mat dns = cv::Mat::zeros(cv::Size(n, m), CV_8UC1);
int p=0;
int i=0;
int j=0;
int d=0;
int nd = 0;
int sd = 0;
int ed = 0;
int wd = 0;
int no = 0;
int so = 0;
int eo = 0;
int wo = 0;
int ng = 0;
int sg = 0;
int eg = 0;
int wg = 0;
int prd=1;
int max=0;
int im=0;
int jm=0;
int tmp=0;
for (i = 0; i < n  -1; i++){
for (j=0; j < m - 1; j++){
p=mask.at<unsigned char>(j,i);
if (p>0)
{
d=0;
for(;;)
{
d=d+1;
nd = i - d;
sd = i + d;
ed = j - d;
wd = j + d;
if (nd < 1) nd = 1;
if (ed < 1) ed = 1;
if (sd >= n) sd = n-1;
if (wd >= m) wd = m-1;
no = mask.at<unsigned char>(j,nd);
so = mask.at<unsigned char>(j,sd);
eo = mask.at<unsigned char>(ed,i);
wo = mask.at<unsigned char>(wd,i);
ng=0;
if (no > 0) ng =  1;
if (no < 0) ng = -1;
sg=0;
if (so > 0) sg =  1;
if (so < 0) sg = -1;
eg=0;
if (eo > 0) eg = 1;
if (eo < 0) eg = -1;
wg=0;
if (wo > 0) wg = 1;
if (wo < 0) wg = -1;
prd = ng * sg * eg * wg;
if (prd == 0)
{
if (d > max)
{
max = d;
im = i;
jm = j;
}
dns.at<unsigned char>(j,i)=d;
break;
}}}}}
x = im;
```

```
y = jm;
x1 = im - (max - 1);
y1 = jm - (max - 1);
if (x1 < 0) x1 = x1 * -1;
if (y1 < 0) y1 = y1 * -1;
x2 = max + im;
y2 = max + jm;
if (x2 > n) x2 = n;
if (y2 > m) y2 = m;
if (x1 > x2) { tmp = x1; x1 = x2; x2 = tmp; }
if (y1 > y2) { tmp = y1; y1 = y2; y2 = tmp; }
return 0;
}
//CODE END
```

Function “***mf_inf2obj***” converts information to object:

```
//CODE START
int mf_inf2obj(obj &myobj,int x,int y,int width=0,int hight=0,int x1=0,int x2=0, int
y1 = 0, int y2=0, int raw_width=0,int raw_hight=0, cv::Mat rawimg =
cv::Mat::zeros(1,1,CV_8UC3))
{
if ( (x<1) || (y<1) )
{
myobj.found=0;
return 1;
}
myobj.found=1;
int tmp=0;
if (x1 > x2) { tmp = x1; x1=x2; x2=tmp; }
if (y1 > y2) { tmp = y1; y1=y2; y2=tmp; }
if (width==0 ) width= x2-x1;
if (hight==0 ) hight= y2-y1;
myobj.wdt = width;
myobj.hgt = hight;
myobj.size = width * hight;
myobj.loc_x = x;
myobj.loc_y = y;
myobj.area_x1 = x1;
myobj.area_x2 = x2;
myobj.area_y1 = y1;
myobj.area_y2 = y2;
if (raw_width == 0) raw_width = rawimg.cols;
if (raw_hight == 0) raw_hight = rawimg.rows;
myobj.shf_x = raw_width -  myobj.loc_x;
myobj.shf_y = raw_hight -  myobj.loc_y;
if (rawimg.data != NULL)
{
cv::Rect roi = cv::Rect(x1,y1,x2-x1,y2-y1);
rawimg(roi).copyTo(myobj.img);
}
return 0;
}
//CODE END
```

Function “***mf_find_spot***” receives several parameters like: raw image, object, red range, green range, blue range. And then try to mark to white on true condition color ranges and black for not matching the range to new image as it is a mask image.

Then by using “***mf_find_density***” to obtain point location where white points are gathered (Fig. 47, 48). Then we can apply blur filter (like median blur) [49] to get rid of noise and small true white points on mask image (Fig. 49).

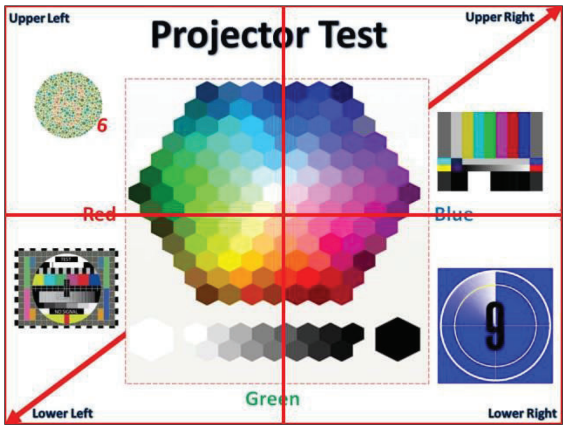


Figure 47: Source input test image

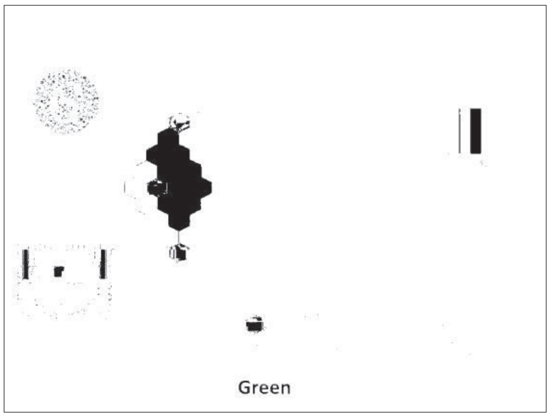


Figure 48: Obtain result mask image

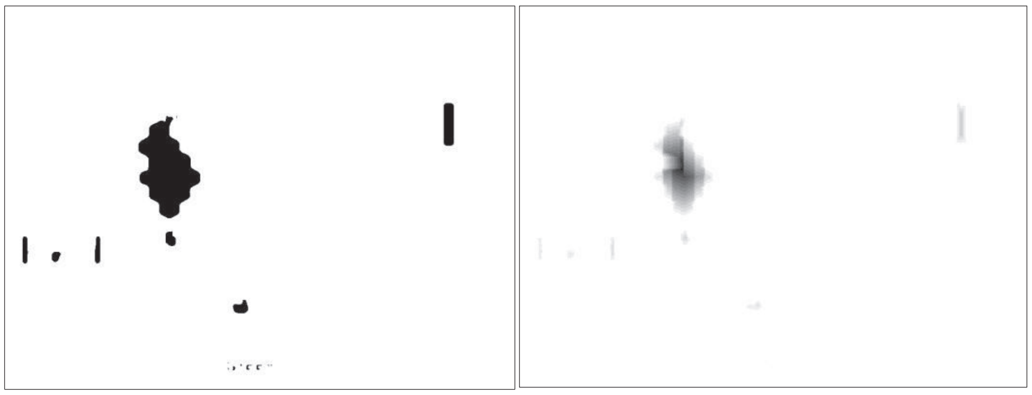


Figure 49: Mask image enhancement and dense image

The main call code to track and find laser spot will be like:

```
//CODE START
//Initialize live video (web camer) send it to →frm image (check previous code)
myobj.name = "Laser"
mf_find_spot(frm,myobj,128,255,0,128,0,128); //Green laser
/*
myobj.loc_x, myobj.lox_y   we obtain all information that robot
 need it as object form
*/
//CODE END
```

Robot function software mechanism has a specific parameter and a specific result. Robot homing guidance will depend only on object data.

3.11 Function: Find Green Ball Using HSV Morphological Manipulating

The RGB manipulation above is not an optimal solution because it is very sensitive to environment change like brightness and noise. Therefore, we can use HSV color system [50]. HSV refers to (Hue, Saturation, and Value) or sometimes is called HSB (parameter Value can be also called Brightness).

In Hue (H) parameter determines the list of gradual colors(0 to 179) starting from red → gradual red with green → green → gradual green with blue → blue → gradual blue with red → red.

The Saturation (S) parameter determines the rate saturation the hue color with white color (0 to 255).

The Value (V) or Brightness (B) parameter determines the rate of hue color with black (0 to 255). Different software's uses different scales. So if they are compared to OpenCV values, we will need to normalize these ranges.

Note:
Brightness is the amount of the light let in; contrast is the difference between dark areas and light areas.

Let us take example and pick a specific color like cyan, cyan is blue white color that is worn by surgery doctors and it is also color of the sky in the morning. The RGB code for cyan is R=0, G=255, B=255, if we need to get more bright cyan color we need to change green and blue by a specific amount and in other color we need to change all three parameters and getting very sensitive and high noise affection.

In HSV cyan code color is H=90, S=255, V=255, to make cyan more bright, we only need to change parameter (V) and if we want to change the rate of cyan in color, we only change the parameter (V). This is useful to control color range (Fig. 50).

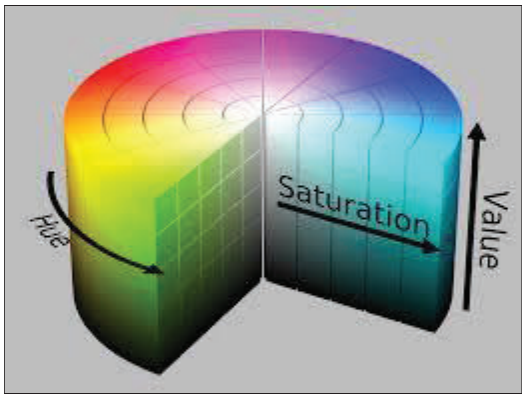


Figure 50: HSV color representation

Morphological manipulation is a set of operations that can be done on mask binary image and changing the sequence of operations can obtain different results. If we have white spots on black background (like mask binary image), the operation that makes those spot be more expanded and more thick is called “Dilate” (Fig. 51), whereas the operation that makes those spots more narrow and thinner is called “Erode” [51] (Fig. 52).

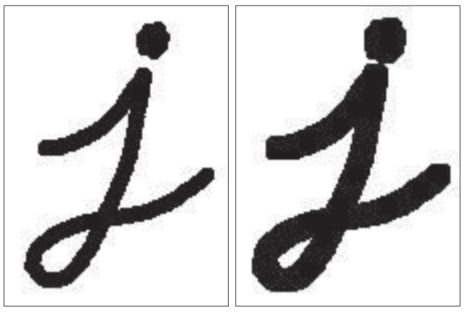


Figure 51: Dilate operation

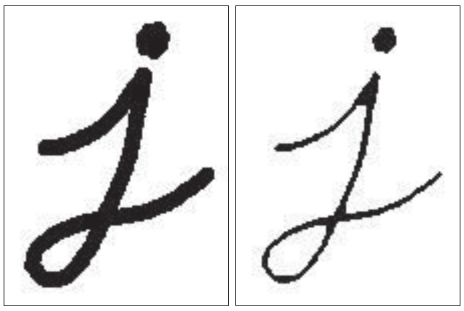


Figure 52: Erode operation

If we apply “Erode” then apply “Dilate” we obtain “Opening”, useful to remove a noisy and small true white spot and mix it in one big spot (Fig. 53).

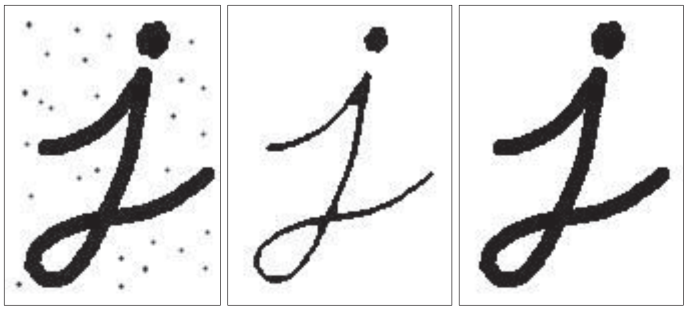


Figure 53: Opening operation

If we apply “Dilate” then apply “Erode” we obtain “Closing” operation, useful to close small gaps inside spots (Fig. 54).

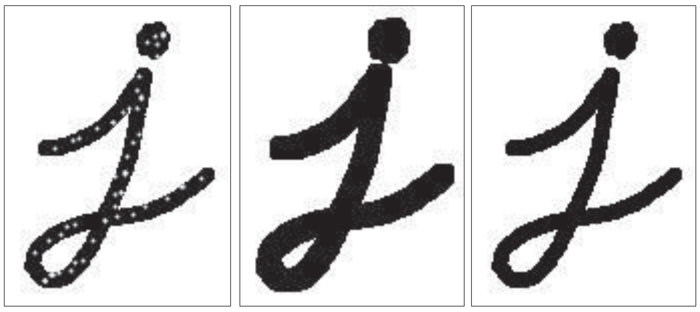


Figure 54: Closing operation

If we apply subtract “Erode” from “Dilate” we obtain “Gradient” operation, useful to find edge detection (Fig. 55).

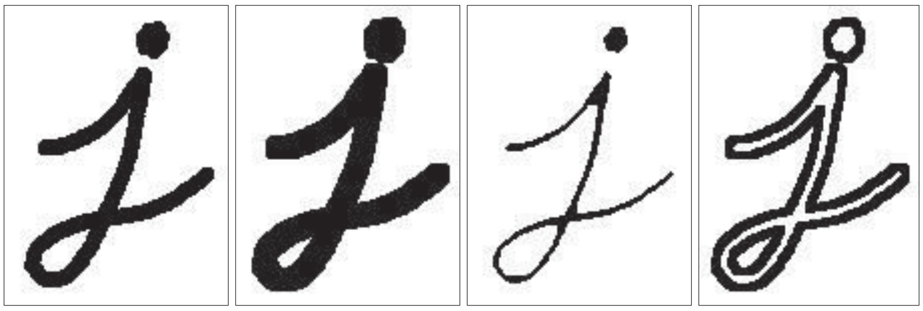


Figure 55: Gradient operation

If we apply subtract original input image from “Opening” we obtain “Top Hat” operation, useful to fracture spot from connection points (Fig. 56).

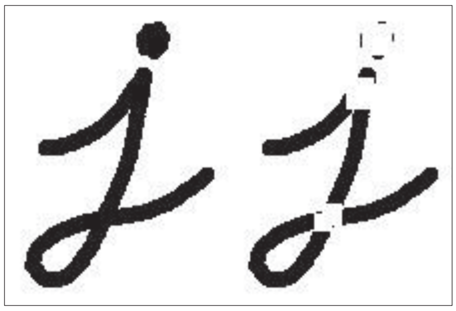


Figure 56: Top hat operation

If we apply subtract “Closing” from original input image obtain “Black Hat” operation, useful to find the connect point among spots (Fig. 57).

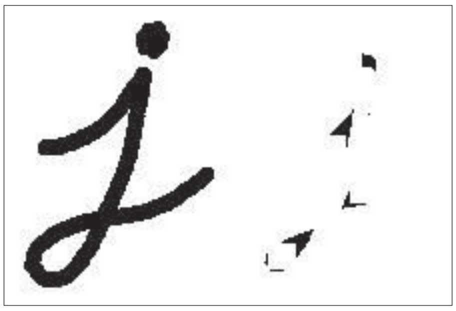


Figure 57: Black hat operation

“Dilate” and “Erode” used matrix, usually used “3x3”. In our project we will use “Opening” to beat small true white spot in binary mask to obtain the best result then save the truest white spot in an object form for later use:

```
//CODE STAR
/*
Green Ball
H : 43 ~  92
S : 48 ~ 162
V : 17 6 ~ 255
*/

int mf_find_hsv(cv::Mat img,obj& myobj,int H_min, int H_max, int S_min,int
S_max, int V_min, int V_max,int erot=3,int dilt=3)
{
myobj.found=0;
cv::Mat frame = img.clone();
cv::Mat frame_HSV;
cv::Mat thresh_HSV;
int H_MIN=H_min;  int H_MAX=H_max;
int S_MIN=S_min; int S_MAX=S_max;
int V_MIN=V_min;  int V_MAX=V_max;
int EROT=erot;   int DILT=dilt;
cv::Mat erodeElement;
cv::Mat dilateElement;
cv::Mat thresh_tmp;
frame_HSV=NULL;
thresh_HSV=NULL;
erodeElement = cv::getStructuringElement(
cv::MORPH_RECT,cv::Size(EROT,EROT));
dilateElement = cv::getStructuringElement(
cv::MORPH_RECT,cv::Size(DILT,DILT));
thresh_tmp=NULL;
cvtColor(frame,frame_HSV,cv::COLOR_BGR2HSV);
inRange(frame_HSV,cv::Scalar(H_MIN,S_MIN,V_MIN),cv::Scalar(H_MAX,S_MAX
,V_MAX),thresh_HSV);
erode(thresh_HSV,thresh_HSV,erodeElement);
erode(thresh_HSV,thresh_HSV,erodeElement);
dilate(thresh_HSV,thresh_HSV,dilateElement);
dilate(thresh_HSV,thresh_HSV,dilateElement);
int x=0,x1=0,x2=0;
int y=0,y1=0,y2=0;
mf_find_density(thresh_HSV,x,y,x1,x2,y1,y2);
mf_inf2obj(myobj,x,y,0,0,x1,x2,y1,y2,0,0,frame);}
//CODE END
```

3.12 Function: Find Image Inside Image Using Feature Detection

Feature detection is a method used to extract and find interest point inside image and relationship between the locations of this point for later use. There are many methods like "FAST" (Features from Accelerated Segment Test), "MSER" (Maximally Stable Extremely Regions) and "SURF" (Speeded Up Robust Features) [52]. The famous method of feature detection is "SIFT" (Scale-Invariant Feature Transform). This method can be applied only in gray scale images (carless for color) then try to find location of interest point inside image, if the location of interest "SIFT" hit one in that place or hit zero [53]. The point of interest may be one the following types:

1 – Isolated points
2 – Connected region points
3 – high-contrast region points
4 – Blob region points
5 – Rapped change (edge)

The result of this method is vector of key points (hits) then "SIFT" descriptor convert this key point from vector form to describe domain, like orientation and relationship between points [54] (Fig. 58).

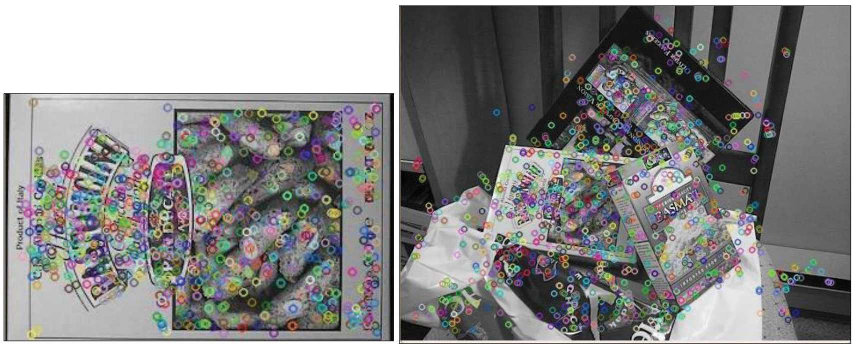


Figure 58: SIFT detection key points

Then try to find match between the two description domains of the target image and source image and find rate of matching using specific threshold value using "FLANN" (Fast Approximate Nearest Neighbor) search library, perform a quick and efficient matching [55] (Fig. 59). Then convert final result to object form.

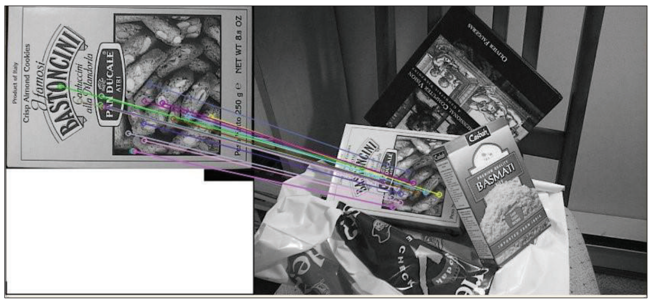


Figure 59: SIFT description with FLANN matching methods

```
//CODE STAR
int mf_find_sift(cv::Mat img1,cv::Mat img2,obj& myobj)
{
myobj.found=0;
cv::Mat IMG_GRY1;
cv::Mat IMG_GRY2;
if (img1.type() == 16)
cv::cvtColor(img1,IMG_GRY1,CV_RGB2GRAY);
else
IMG_GRY1 = img1.clone();
if (img2.type() == 16) cv::cvtColor(img2,IMG_GRY2,CV_RGB2GRAY);
else IMG_GRY2 = img2.clone();
cv::SiftFeatureDetector sift1;
cv::vector<cv::KeyPoint> keyps1;
cv::Mat dsc1;
sift1.detect (IMG_GRY1, keyps1);
sift1.compute(IMG_GRY1, keyps1,dsc1);
cv::SiftFeatureDetector sift2;
cv::vector<cv::KeyPoint> keyps2;
cv::Mat dsc2;
sift2.detect (IMG_GRY2, keyps2);
sift2.compute(IMG_GRY2, keyps2,dsc2);
cv::FlannBasedMatcher matcher;
cv::vector< cv::DMatch > matches;
matcher.match( dsc1, dsc2, matches );
double max_dist = 0; double min_dist = 100;
for( int i = 0; i < dsc1.rows; i++ )
{
double dist = matches[i].distance;
if( dist < min_dist ) min_dist = dist;
if( dist > max_dist ) max_dist = dist;
}
```

```
cv::vector< cv::DMatch > good_matches;
int thresh = 2*min_dist; // = 0.8
for( int i = 0; i < dsc1.rows; i++ )
if( matches[i].distance < thresh ) good_matches.push_back( matches[i]);
if (good_matches.size() >= 3)  myobj.found=1;
if (myobj.found == 1) {
cv::Mat mask=cv::Mat::zeros(cv::Size(IMG_GRY1.cols, IMG_GRY1.rows),
CV_8UC1);
int ptx=0;
int pty=0;
for(int i=0;i< good_matches.size();i++)
{
ptx=keyps1[good_matches[i].queryIdx].pt.x;
pty=keyps1[good_matches[i].queryIdx].pt.y;
mask.at<unsigned char>(pty,ptx)=255;
}
int EROT=21;
int DILT=51;
cv::Mat erodeArray = cv::getStructuringElement(
cv::MORPH_RECT,cv::Size(EROT,EROT));
cv::Mat dilateArray = cv::getStructuringElement(
cv::MORPH_RECT,cv::Size(DILT,DILT));
dilate(mask,mask,dilateArray);
dilate(mask,mask,dilateArray);
erode(mask,mask,erodeArray);
erode(mask,mask,erodeArray);
cv::medianBlur(mask,mask,9);
int x=0,x1=0,x2=0;
int y=0,y1=0,y2=0;
mf_find_density(mask,x,y,x1,x2,y1,y2);
mf_inf2obj(myobj,x,y,0,0,x1,x2,y1,y2,0,0,img1);
return 0;
}//CODE END
```

3.13 Function: Trained Object Finder Using Cascade Classifier Training

In this method we try also to find image inside image but using multi images as input to increase the chance of detection by converting a set of images to database file and then try to find it inside source image. This operation is called "Cascade Training". The cascade training depends on some factors as:

1 - High mount number for train images resulting high accuracy and vice versa.
2 - Long amount of time taking for train images resulting high accuracy and vice versa.

Cascade training divides the set of images into two parts: Positive images (or called object images) and negative images (or called background images). Positive images include images that contain one or more target object inside those images, all the objects inside the positive images need information related to it, like location and size of that object inside positive images. The style of this order needs to put positive images inside specific folder [56], for example "pos":

/pos (Folder "pos")
img001.jpg (positive images inside folder)
img002.jpg (positive images inside folder)
.... Etc.

The form of information file will be like "Pos.inf" that contains:

pos/img001.jpg 1 20 20 100 100 (1 object → x1,y1= 20,20 → x2,y2 = 100,100)
pos/img.002.jpg 2 80 80 190 190 40 40 120 120 (2 objects and locations of objects)
….. Etc.

This "pos.inf" file must be in root with "pos" folder (not inside with images).
Where negative images will contain any image and not include object, any image can be used, with caring about the size of the negative images must be equal or large that

from positive images. The style of this order needs to put negative images inside specific folder, for example "neg":

/neg (Folder "neg")
img001.jpg (negative images inside folder)
img002.jpg (negative images inside folder)
…. Etc.

The form of information file will be like "neg.inf" contains:
neg/img001.jpg
neg/img.002.jpg
….. Etc.

The result of the above operation could be summarized, as follows:
1 - Folder "pos" contains positive object images.
2 - File "pos.inf" contains information related to images inside "pos" folder.
3 - Folder "neg" contain negative background images.
4 - File "neg.inf" contains a list of names images that are inside "neg" folder.

Note:

- All the above images (positive and negative) must be in gray scale form.
- All folder and files above must put to:
  "[PATH]\opencv\build\x86\vc10\bin".

The next step is creating Vector file for positive images, and all positive images will be converted to one file that contains them all. Using command line in side above path:
"[PATH]\opencv\build\x86\vc10\bin\ opencv_createsamples.exe
-info pos.inf -num "X" -w "W" –h "H" -vec pos.vec".

Where "X" number of samples (positive images), "W" width of output vector, "H" height of output vector, where output of this step is vector file "pos.vec". To check our step and go successfully we can show vector file and see inside images by:

“[PATH]\opencv\build\x86\vc10\bin\ opencv_createsamples.exe
-w “W” –h “H” -vec pos.vec”.

Make vector file appear inside windows with turning between images files that contains it.

Note:
For width and height of output vector file, not fixed change depends on the size of the object inside positive images. The small value will correspond to 24 and other big values will correspond to 24 * (big/small) value. Let us take an example if we have objects size 640X480 (WXH) then the vector file must be (32X24) where 480 small from 640 corresponded to 24 (480 → 24) and 640 corresponded to 24 * (640/480) = 32. These calculations keep vector image form change in Aspect ratio and save the style dispose.

The result of the above operation could be summarized as:
Output vector file “pos.vec” [57] (Fig. 60).

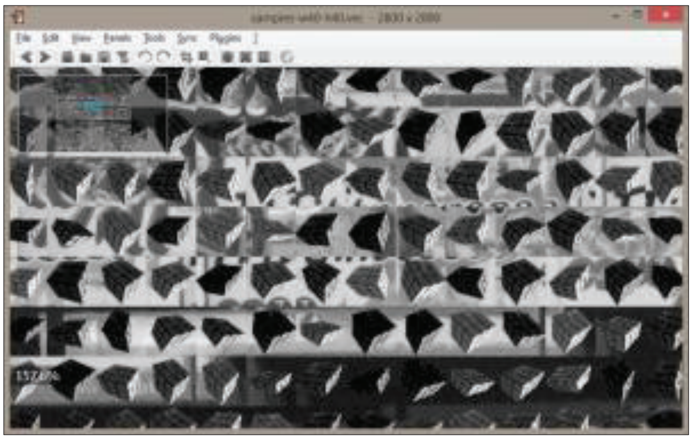


Figure 60: Vector file “.vec” content for rubik's cube

The next and last step to create cascade by training positive vector file with negative file creating cascade trained database and extension of output will be “.XML” (Extensible Markup Language) that keeps all information inside it and there is no need to images. To start trainings we must create empty folder named XML that keep

“.XML” files inside and make sure that they are empty. Then we try to train all images by using command line:

```
[PATH]\opencv\build\x86\vc10\bin\ opencv_traincascade.exe
 -data XML -vec pos.vec  -bg neg.inf   -numPos “P”  -numNeg “N” -numStages “S”
-w “W” -h “H”  -featureType “F”.
```

Where “P” number of positive images, “N” number of negative images, “S” number of stage level (more level more accuracy, more time), “F” feature type for creating cascade by using different methods like : HARR (ratio to Hungarian mathematician Alfred Haar in 1910), LBP (Local Binary Patterns) and HOG (Histogram of Oriented Gradients) [58]. Every cascade method has pros and cons, if the time taken is a couple of hours, in LBP to train cascade may take a couple of weeks in HARR [59].
The final output will be “Cascade.XML” we may give it another name in our object like “ball.xml”. This XML file will be used as database for code in OPENCV, we can also use any XML database that has been already taken from the internet or from other sources and get rid of any effort to create a new one. This technique is already used in face detection and other human body detection.
There are already trained sample XML file in:
[PATH]\opencv\sources\data\haarcascades”,
and we can use any of these files.

The OpenCV code can be represented as “***mf_find_cascade***”:

```
//CODE START
int mf_find_cascade(cv::Mat img,obj &myobj, cv::String xml,double
par3_ScaleFactor=1.1,int par4_MinNeighbors=2,int par5_flags=0,int
par6_MinSiz=30,int par7_MaxSize=100)
{
myobj.fnd=0;
cv::Mat frame = img.clone();
struct strc3
{
```

```
double par3_ScaleFactor;
int par4_MinNeighbors;
int par5_flags;
int par6_MinSize;
int par7_MaxSize;
};

strc3 cas;
int TgT_Left =0;
int TgT_Top =0;
int TgT_Height =0;
int TgT_Width  =0;
int Tgt_FindIt  =0;
cv::String obj_cascade_name;
cv::CascadeClassifier obj_cascade;
cv::vector<cv::Rect> objk;
cv::Mat frame_gray;
cas.par3_ScaleFactor = par3_ScaleFactor;
cas.par4_MinNeighbors = par4_MinNeighbors;
cas.par5_flags = par5_flags;
cas.par6_MinSize = par6_MinSiz;
cas.par7_MaxSize = par7_MaxSize;
obj_cascade_name = xml;
if( !obj_cascade.load( obj_cascade_name ) )
{std::cout<<"--(!)Error loading XML cascade\n"; return 1;}
cv::cvtColor( frame, frame_gray, cv::COLOR_BGR2GRAY );
cv::equalizeHist( frame_gray, frame_gray );
obj_cascade.detectMultiScale
( frame_gray, objk,  cas.par3_ScaleFactor, cas.par4_MinNeighbors,
cas.par5_flags,cv::Size(cas.par6_MinSize, cas.par6_MinSize,
cv::Size(cas.par7_MaxSize,cas.par7_MaxSize ));
```

```
if (objk.size() >= 1)
{
TgT_Left = objk[0].x+ int(objk[0].width / 2);
TgT_Top =  objk[0].y+ int(objk[0].height /2);
TgT_Height = objk[0].height;
TgT_Width  = objk[0].width;
Tgt_FindIt = 1;
}
if (Tgt_FindIt == 1)
mf_inf2obj(myobj,TgT_Left,TgT_Top, TgT_Width, TgT_Height,0,0,0,0,0,0,frame);
return 0;
}
//CODE END
```

3.14 Homing Guidance

We talk in chapter one about autonomous robot function. That robot is full controlled without any human intervention and here in this section we try to do autonomous function for homing robot arm to make target object on sight. There are three types of homing system: active homing, semi active homing and passive homing [60].

In active homing robot we try to send signal or sound wave to target and these waves will be reflected from object to robot sensor and then try to act peripheral device depending on reflect signal. This homing system keeps the robot to continue lock target until peripheral device finishes the duty where this device will not take any order from the environment, they receive an order from robot central system only.

In semi active homing all peripheral devices send and receive information with object target at outside using radio or ultra sound systems.

In passive homing the peripheral device will only receive signals that emission from target like unique color, unique shape and infra red. This system will be used almost in all vision robotic systems using camera, camera mount in a different place on robot if the camera not mount in peripheral device like robot arm.

This called "Eye at Hand", if camera mounts on peripheral device that makes camera in moving position this call "Eye In Hand". Every style has pros and cons, in my system I used "Eye in Hand" where target object stay locked into the screen center using information that is gathered from detection functions. We use from pervious vision function and make it useful by using object information, we have two information:
"myobj.sft_x" and "*myobj.sft_y"* indicating mount of shift of object from the center of the screen. The robot arm with camera mount in try to be put in the center of screen, using arm motor (A) as pan (mean paining from far left to far right) and using arm motor (C) as tilt (mean tilt from bottom to upper), moving in two axel and free aiming to any location:

```
//CODE START
int mf_homing(obj &myobj) {
int tgtx=myobj.sft_x;
int sgnx=0
if (tgtx > 0) sgnx=1
if (tgtx < 0) sgnx=-1
tgtx= tgtx * sgnx;
int stpx=0;
if (tgtx >= 5)   stpx=stpx+1;
if (tgtx >= 55)  stpx=stpx+1;
if (tgtx >= 105) stpx=stpx+1;
if (tgtx >= 205) stpx=stpx+1;
if (tgtx >= 255) stpx=stpx+1;
stpx = stpx * sgnx;
int armx = stpx;
int tgty=myobj.sft_y;
int sgny=0
if (tgty > 0) sgny=1
if (tgty < 0) sgny=-1
tgty= tgty * sgny;
int stpy=0;
if (tgty >= 5)   stpy=stpy+1;
if (tgty >= 55)  stpy=stpy+1;
if (tgty >= 105) stpy=stpy+1;
if (tgty >= 205) stpy=stpy+1;
if (tgty >= 255) stpy=stpy+1;
stpy = stpy * sgny;
int army = stpy;
link->Write(armx+3000);
link->Write(army+4000);
return 0;
}
 //CODE END
```

Output will be (Fig. 61).

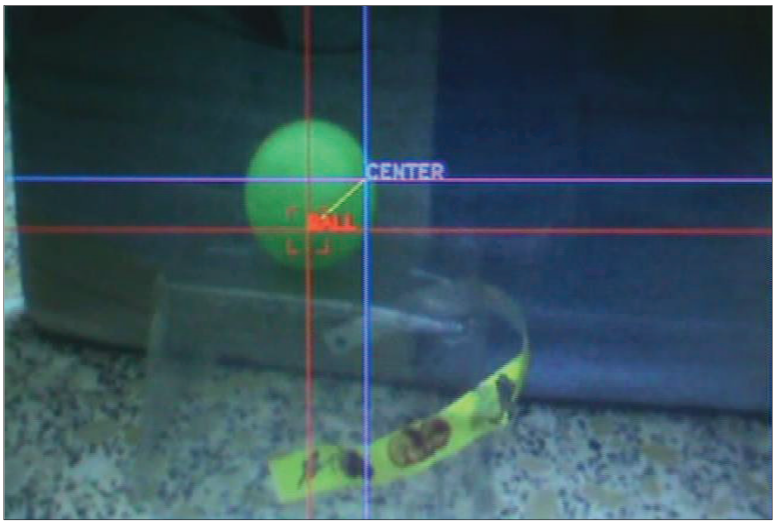


Figure 61: Passive homing using "eye in hand"

3.15 Summary

The software we used Arduino IDE as inside Arduino as SBM board, Windows operating system with Visual Studio C++ 2010 Ultimate combine OpenCV library using it in light version inside laptop board as SBC board and then connecting SBM with SBC by communication port “COM” from both sides.

CHAPTER 4

CONCLUSION

In this chapter we discuss the obtained result and draw some conclusion to be discussed and utilized for the future work, for our system and advantages of this system and what characterize it from other systems, then we discuss the obstacles and challenges for building our system and how to avoid those disadvantages in the future, and last section will be about recommendations and the development of a system for those who want to improve the work with the system and upgrade it in future.

4.1 Characterization and Future Works

This system is characterized from other systems in being applicable for future work but first let us summarize this characterization as follow:

1- Very uniform software language that any person coded on his own pc he can execute on this system.

2- Very compatible with extra device without need to be relearn, because compatible with university students materials and famous devices in markets.

3- Flexible and expandable because the structure of this system has a lot of interface port that easily adds a new device without redesigned it.

4- No need to study internal hardware or software, we just need a brief explanation of the main instruction of this system and any one can just add a specific function without re changing other functions or redesigning core software (Fig. 62).

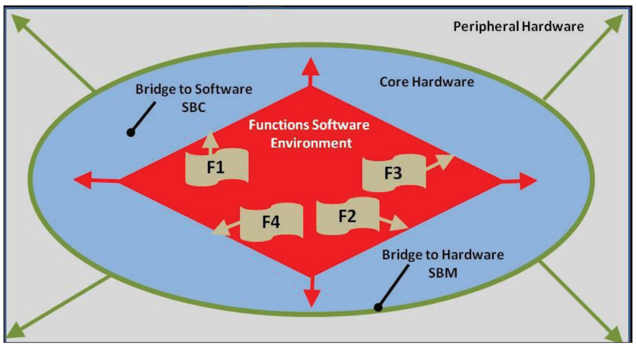


Figure 62: Multifunction system illustrating diagram

This unique structure leads to the application of useful and scientific application easily and in multi ways. We can apply it as a stationary or mobility status, this means that we can remove the wheels or unused them to stay only in the same place and application function and this is called stationary, or we can use them as mobility status like patrolling. Whatever used, in both cases we can use as: remote camera, motion detection and useful form sensors that can mount inside the system and used as weather sensors and risk managements. Moreover, that great useful from of this system is using computer vision algorithms and methods to detect and find the object that can be used as national security to find car sticky bombs and road mount bombs, and a lot of things even I cannot imagine the things that make computer vision to be alive.

4.2 Obstacles and Challenges

No project or working without getting obstacles, its life natural but smart way to convert this obstacle to our advantage by using to future development. We can summarize it as follow:

1- Real-Time system problems: Because this system is immediately in act procedures there are high ratio of errors effect on system stability. Fatal errors could only be known when the system in duty, can eliminate these errors goes for ever [61].

• Avoidance: This problems cannot be eliminated easily but we can avoid it by using team testers that make fatal errors appear quickly and try to eliminated them, and we should not go to the next step if we are not sure that the present step is working perfectly because if we go to the next step without checking it the incoming errors will be ambiguous and very hard to be found and fixed.

2- Hardware representation: A lot of theses and projects are done without any implementation in real, but in robot system with hardware representation will be easy to understand and apply the methods but in the same time will be a hard work because the producers that we apply in our imagination do not get the same result which are got by real presentation. And dealing with hardware representation is not easy because they need maximum expertise and knowledge in electronics.

• Avoidance: Using simulations, there are limited robot simulations, we need to find the best suitable one with real presentation and our development and optimization will be virtually inside simulation and in the final stage we can build the system by using the final form of robot simulation [62]. Simulator call "LabView" create by Ni (National Instruments) company, in LabView 2012 a package add-on called "Robotic Simulator" that is easy to the implementation of robot and easy to convert from virtual world to real word [63-64].

3- Cost fee: Building robotic system cost a lot of fees and buying the device and its development, without financial supply from the government and university, the robot building will be high costing.

• Avoidance: Using again simulation [65], for its best choice you should be sure that have complete financial fee or never try to work with any system because you will stop in the middle.

4- Duration of the work: As thesis work you should have hardware presentation and coding, you also need to write the thesis documentation. The whole thing you need much time and you go beyond normal thesis, specialty when you learn new things such as, dealing with a new device or new coding language. You must remember that you have limited time and you should consider that fact or you will be late in schedule.

• Avoidance: You are not recommended to work with a new thing which you do not understand and take a lot of time. Try to work with the thing you already know to save time and to start a plan early from schedule time. Keep the portions of work balanced in time dividing do not focus on part and leave the other three main parts: hardware, software and documentation.

5- National security: Some countries as the third world countries see that this work threats their national security because there are chances to use robot materials in wrong way. Therefore, some robot devices in the third world countries is illegal and hard to be found in the market. Moreover, dealing with these devices exposes you to arrest.
• Avoidance: If you live in the third world countries try never to work in hardware representation and keep your work in low profile as much as possible like using simulation.

4.3 Recommendations and Developments

In this section we will talk about the possible development of the system in future or for the person who wants to continue develop what I finished. These commendations and developments are summarized as follow:

1- Using small SBC as possible, light version of laptop board has a lot of benefits but in the same time, its disadvantage is the size and weight of the board with its battery and storage device and these things make the robot need more power.
• Recommendation: Use “Raspberry pi” [65] or “PCduino” [66] or “beaglebone black” [67] or “pandaboard ES” [68]

2- Using high power motor for movements (not for steering) to avoid robot weight problems.
• Recommendation: Use the motor that has high torque as possible and marked with “HP” (Horse Power customization) [69].

3- Using high and full power battery.
• Recommendation: Use high mount of mApH (mille ampere per hour) as possible [70].

4- Using online charge system that can charge robot itself without turning it off, by using sockets electric charging or solar cell charging system [71].

• Recommendation: Use a charge system similar to the aerial refueling by using "Probe-and-drogue" [72], where robot park in custom station and using specific probe that mounts in robot and then connects it to drogue charging system.

5- Using ultrasonic to determine and measure any obstacle in front of the robot and avoid it. Using pulse signal to send and wait for response and measurement of the period of time taken from sending until receiving, with using the ultrasonic sensor four pinouts type or three pinouts type.

• Recommendation: Use "HC-SR04" Ultrasonic Sensor [73].

6- Using GPS "Global Positioning System" to locate the robot on the map, by applying three variables that GPS board provides system (X, Y, and Z):

Where X is location in Horizontal Line (Latitude "LAT").

Where Y is location in Vertical Line (Longitude "LON").

Where Z is location in elevation (Altitude from mean sea level "ALT").

Range of X: -90N to 90N (N: North).

Range of Y: -180E to 180E (E: East).

Range of Z: -10924M to 8848M (M: Meters).

And apply the above information to the map [74].

• Recommendation: Use "Sparkfun Arduino GPS Shield" [75].

7- Using IR (Infra Red) cameras to duty in night and dark place.

• Recommendation: Use cheap CCTV (Closed-Circuit Television) camera that is use in survival DVR (Digital Video Recorder) systems [76].

8- Using mobile internet inside robot system to be remote, observing and controlling for far distance.

• Recommendation: Use ”Mi-Fi” (Mobile Wireless Fidelity) that is connected to self phone internet provider network and provide system by using LAN cables (Local Area Network), which is sometimes, called “3G Router” [77].

REFERENCES

1. **Marvin L., (1982),** *"Why People Think Computers Can't"*, AI Magazine, California, vol. 3, no. 4, pp. 4-9.

2. **Scheier C., Pfeifer R., (2001),** *"Understanding Intelligence"*, MIT Press, Boston, pp. 56-60.

3. **Mittal R. K., Nagrath I., (2003),** *"Robotics and Control",* Tata McGraw-Hill Education, Noida, pp. 6-10.

4. **Parisi D., (2014),** "*Future Robots: Towards a Robotic Science of Human Beings"*, John Benjamins, Amsterdam, vol. 7, pp. 36-39.

5. **Kent A., Williams G. J., (1994),** *"Encyclopedia of Microcomputers: Reporting on Parallel Software to SNOBOL"*, CRC Press, Florida, vol. 15, pp. 108-111.

6. **Shimon Y. N., (1999),** "*Handbook of Industrial Robotics"*, John Wiley & Sons, New York, vol. 1, pp. 148-153.

7. **Luokkala B. B., (2013),** *"Exploring Science through Science Fiction"*, Springer Science & Business Media, Berlin, pp. 96-99.

8. **Springer J. P., (2013),** *"Military Robots and Drones: A Reference Handbook Contemporary World Issues"*, ABC-CLIO, California, pp. 1-6.

9. **Sheridan B. T., (1992),** *"Telerobotics, Automation, and Human Supervisory Control"*, MIT Press, California, pp. 99-104.

10. **Cohen B. Y., Marom A., Hanson D., (2009),** *"The Coming Robot Revolution: Expectations and Fears About Emerging Intelligent, Humanlike Machines"*, Springer Science & Business Media, Berlin, pp. 29-34.

11. **CNS News**, http://cnsnews.com/mrctv-blog/curtis-kalin/obama-admits-hes-scared-japanese-robots. (Data Download Date: 19.07.2014).

12. **Peters A., (2014),** *"Thunder Under the Sea"*, Dorrance Publishing, Pennsylvania, pp. 2-5.

13. **Launius D. R., McCurdy E. H., (2008)**, *"Robots in Space"*, JHU Press, Maryland, pp. 22-26.

14. **Springer J. P., (2013),** *"Military Robots and Drones: A Reference Handbook Contemporary World Issues"*, ABC-CLIO, California, pp. 205-210.

15. **Bernhardt R., Bernard R., Dillman R., Hormann K., Tierney K., (1991),** "*Integration of Robots into CIM"*, Springer Science & Business Media, Berlin, pp. 302-307.

16. **Cook D., (2010),** *"Intermediate Robot Building",* Apress Media, New York, pp. 410-414.

17. **Phipps A. C., (1999),** "*Fundamentals of Electrical Control"*, Fairmont Press, Georgia, pp. 162-166.

18. **Onwubolu G., (2005),** "*Mechatronics: Principles and Applications"*, Butterworth-Heinemann, Oxford, pp. 245-249.

19. **Bakshi A. U., Bakshi V. M., (2009)**, *"Electrical Drives and Control",* Technical Publications, Wolverhampton, pp. 68-73.

20. **Husain I., (2003)**, *"Electric and Hybrid Vehicles: Design Fundamentals"*, CRC Press, Florida, pp.96-100.

21. **Funakubo H., (1991*), "Actuators for Control"*, CRC Press, Florida, vol. 2, pp. 55-59.

22. **Niku S., (2010),** *"Introduction to Robotics"*, John Wiley & Sons, New York, pp. 286-289.

23. **Barrett S., (2010),** *"Arduino Microcontroller Processing for Everyone",* Morgan & Claypool Publishers, California, vol. 28, pp. 199-202.

24. **Parab S. J., (2008),** "*Practical Aspects of Embedded System Design Using Microcontrollers"*, Springer Science & Business Media, Berlin, pp. 53-58.

25. **Gurevich V., (2005),** *"Electric Relays: Principles and Applications"*, CRC Press, Florida, pp. 118-121.

26. **Chakraborty S., Simoes G. M., Kramer E. W., (2013),** *"Power Electronics for Renewable and Distributed Energy Systems"*, Springer Science & Business Media, Berlin, pp. 534-539.

27. **Gookin D., (2012),** *"Laptops for Dummies",* John Wiley & Sons, New York, pp. 190-195.

28. **James L. K., (2013),** *"Computer Hardware: Installation, Interfacing, Troubleshooting and Maintenance"*, PHI Learning, Delhi, pp. 47-52.

29. **Wong Y. J., (2008),** *"Theory of Ground Vehicles"*, John Wiley & Sons, New York, pp. 454-458.

30. **Singh K. A., Kumar A. S., (2008),** "*Microcontroller and Embedded System"*, New Age International, Kolkata, pp. 6-11.

31. **Barrett S., (2010)**, *"Arduino Microcontroller Processing for Everyone",* Morgan & Claypool Publishers, California, vol. 28, pp. 9-13.

32. **Kent A., Williams G. J., (1991),** *"Encyclopedia of Computer Science and Technology"*, CRC Press, Florida, vol. 25, pp. 278-283.

33. **Grimmett R., (2014),** *"Arduino Robotic Projects*", Packt Publishing, Birmingham, pp. 210-214.

34. **Wu Y., (2011),** *"Advances in Computer, Communication, Control and Automation"*, Springer Science & Business Media, Berlin, vol. 121, pp. 466-469.

35. **Barrett F. S., (2012),** *"Arduino Microcontroller: Processing for Everyone",* Morgan & Claypool Publishers, California, vol. 38, pp. 41-46.

36. **Premeaux E., Evans B., (2011),** "*Arduino Projects to Save the World"*, Apress Media, New York, pp. 52-56.

37. **The Conversation UK News**, http://theconversation.com/when-the-atm-runs-windows-how-safe-is-your-money-32823. (Data Download Date: 18.08.2014).

38. **Microsoft Developer Network Library**, http://msdn.microsoft.com/en-us/library/gg265786%28v=vs.100%29.aspx. (Data Download Date: 03.08.2014).

39. **Wienholt N., (2003),** *"Maximizing.NET Performance"*, Apress Media, New York, pp. 95-99.

40. **Rubin E., Yates R., (2003),** *"Microsoft.NET Compact Framework"*, SAMS Publishing, Indianapolis, pp. 37-40.

41. **Web Tuts Depot**, http://webtutsdepot.com/2010/05/01/arduino-visual-c-plus-plus-tutorial. (Data Download Date: 05.08.2014).

42. **Nagabhushana S., (2005),** *"Computer Vision and Image Processing"*, New Age International, Kolkata, pp. 164-169.

43. **Driggers G. R., (2003***), "Encyclopedia of Optical Engineering",* CRC Press, Florida, vol. 2, pp. 1212-1216.

44. **Laganiere R., (2014),** *"OpenCV Computer Vision Application Programming Cookbook"*, Packt Publishing, Birmingham, pp. 92-96.

45. **Brahmbhatt S., (2013),** *"Practical OpenCV"*, Apress Media, New York, pp. 24-29.

46. **Dalal J., Patel S., (2013),** *"Instant Opencv Starter"*, Packt Publishing, Birmingham, pp. 48-52.

47. **Petersen R. L., (2006),** *"Introductory C with C++"*, Surfing Turtle Press, California, pp. 618-622.

48. **Goshtasby A. A., (2005),** *"2-D and 3-D Image Registration"*, John Wiley & Sons, New York, pp. 90-93.

49. **Burger W., Burge M. J., (2009),** "*Digital Image Processing"*, Springer Science & Business Media, Berlin, pp. 108-113.

50. **Girod B., Niemann H., Seidel H. P., (1999),** "*Vision Modelling and Visualization"*, IOS Press, Amsterdam, pp. 95-99.

51. **Bradski G., Kaehler A., (2008)**, *"Learning OpenCV"*, O'Reilly Media, California, pp. 155-159.

52. **Siegwart R., Nourbakhsh I. R., Scaramuzza D., (2011),** *"Introduction to Autonomous Mobile Robots"*, MIT Press, Boston, pp. 234-237.

53. **Brahmbhatt S., (2013),** *"Practical OpenCV",* Apress Media, New York, pp. 125-128.

54. **Kalra P., (2006),** *"Computer Vision, Graphics and Image Processing"*, Springer Science & Business Media, Berlin, pp. 59-64.

55. **Sandri S., Sanchez M. M., Cortes U., (2009),** *"Artificial Intelligence Research and Development"*, IOS Press, Amsterdam, vol. 202, pp. 13-16.

56. **Bradski G., Kaehler A., (2008),** *"Learning OpenCV"*, O'Reilly Media, California, pp. 509-514.

57. **Nomacs Image Lounge**, http://www.nomacs.org/2-0-2-supports-vec-file. (Data Download Date: 10.09.2014).

58. **Zhang C., Zhang Z., (2010),** *"Face Detection and Adaptation"*, Morgan & Claypool Publishers, California, pp. 12-16.

59. **Kornyakov A. S., (2013),** *"Instant Opencv for IOS"*, Packt Publishing, Birmingham, pp. 32-36.

60. **Maini A. K., (2013),** *"Lasers and Optoelectronics"*, John Wiley & Sons, California, pp. 306-309.

61. **Lauwereins R., Madsen J., (2008),** *"Design, Automation, and Test in Europe"*, Springer Science & Business Media, Berlin, pp. 97-101.

62. **Cetto J. A., Ferrier J. L., Filipe J., (2009),** *"Informatics in Control, Automation and Robotics"*, Springer Science & Business Media, Berlin, pp. 93-98.

63. **Sumathi S., Surekha P., (2007),** *"LabVIEW Based Advanced Instrumentation Systems"*, Springer Science & Business Media, Berlin, pp. 644-649.

64. **Liu H., Ding H., Xiong Z., Zhu X., (2010),** *"Intelligent Robotics and Applications"*, Springer Science & Business Media, Berlin, pp. 351-354.

65. **Richardson M., Wallace S., (2012),** *"Getting Started with Raspberry Pi"*, O'Reilly Media, California, pp. 8-13.

66. **Kurniawan A., (2014),** *"Getting Started with pcDuino3"*, PE Press, California, pp. 6-10.

67. **Richardson M., (2013*),* ***"Getting Started with BeagleBone"*, O'Reilly Media, California, pp. 8-12.

68. **Auler B. C., (2014*),* ***"Getting Started with LLVM Core Libraries"*, Packt Publishing, Birmingham, pp. 235-240.

69. **Smith R. L., Herman S. L., (2004),** *"Electrical Wiring Industrial"*, Cengage Learning, Boston, pp. 121-124.

70. **Jha A. R., (2012),** *"Next-Generation Batteries and Fuel Cells for Commercial, Military, and Space Applications"*, CRC Press, Florida, pp. 266-270.

71. **Barsukov Y., Qian J., (2013),** *"Battery Power Management for Portable Devices"*, Artech House, Boston, pp. 90-94.

72. **Langton R., Clark C., Hewitt M., Richards L., (2009),** *"Aircraft Fuel Systems"*, John Wiley & Sons, New York, vol. 24, pp. 111-114.

73. **Karvinen K., Karvinen T., (2014),** *"Getting Started with Sensors"*, Maker Media, California, pp. 70-74.

74. **Gabler R., Petersen J, Trapasso L., Sack D., (2008),** *"Physical Geography"*, Cengage Learning, Boston, pp. 38-42.

75. **Boxall J., (2013),** *"Arduino Workshop"*, No Starch Press, California, pp. 261-264.

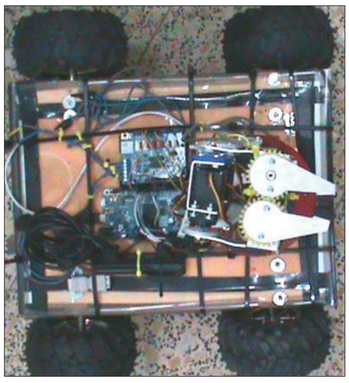
76. **Kruegle H., (2011),** *"CCTV Surveillance"*, Butterworth-Heinemann, Oxford, pp. 56-59.

77. **Parsons J. J., Oja D., (2013)**, *"New Perspectives on Computer Concepts"*, Cengage Learning, Boston, pp. 327-332.

APPENDICES A

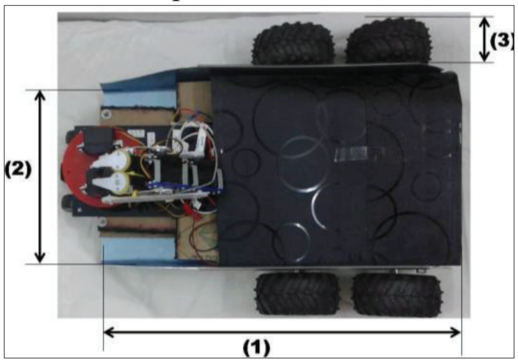
PROJECT ROBOT PICTURES AND MEASUREMENTS

First Prototype of MF Robot Project:

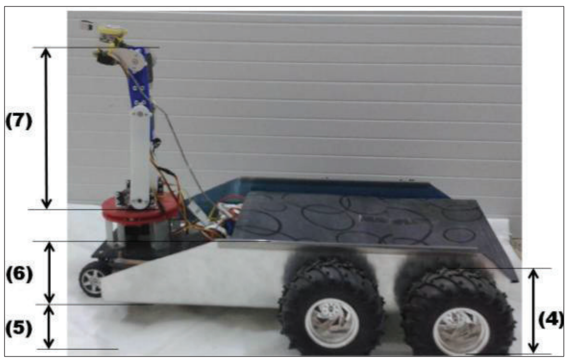


Final Version of MF Robot Project:

Top View



Side View



(1). Length of Project Robot: 50 Cm.
(2). Width of Project Robot: 29 Cm.
(3). Width of Project Robot Wheel: 6 Cm.
(4). Diameter of Project Robot Wheel: 16 Cm.
(5). Height Project Robot from ground: 13 Cm.
(6). Height of Project Robot: 10 Cm.
(7). Height of Project Robot Arm (Cobra Mode): 25 Cm.
(•). Total Weigh of Project Robot: 5 Kg.
(•). Weigh of Project Robot for: (SBC: 1.5 Kg, Battery: 0.5 Kg, Arm: 0.5 Kg).

APPENDICES B

CURRICULUM VITAE

PERSONAL INFORMATION

Name, Surname: Mohammed Sulaiman, MUSTAFA
Date and Place of Birth: 30 December 1983, Kirkuk/Iraq
Marital Status: Single
Phone: Iraq +964.770.936.67.57 Turkey +9.0538.270.48.11
Email: moh.kirkuk@gmail.com
Skype: moh_kbc2
Webpage: http://www.facebook.com/moh.kirkuk

EDUCATION

Degree	**Institution**	**Year of Graduation**
M.Sc.	Çankaya University, Computer Engineering	2014
B.Sc.	Kirkuk Technical Collage, Software Engineering	2007
Diploma	Kirkuk Technical Institute, Computer Systems	2003

WORK EXPERIENCE
Free Works, Unemployed.

FOREIN LANGUAGES
Advanced English, Advanced Arabic, Beginner Turkish.