

An Analysis on Controlling Humanoid Robot Arm Using Robot Operating System (ROS)



**Middlesex
University**

Sriram Murali

A Thesis submitted to Middlesex University in partial fulfillment of the requirements for the degree of

Master of Science (by Research)

**Department of Design Engineering & Mathematics
Middlesex University, London**

Supervisor: Dr. Vaibhav Gandhi

Dr. Zhijun Yang

Acknowledgements

I like to deliver my greatest obligation Dr. Vaibhav Gandhi and Dr. Zhijun Yang for the endless support for my thesis, from initial advice from the early stages of conceptual wellspring and through evolving advice and inspiration to this day. I am thankful to them for guiding me to the Robotics field.

I would like to thank Dr. Tao Geng for serving advice through the project work.

I am thankful to Middlesex University Research team for providing me an opportunity to pursue the Master's degree as a research program.

I like to acknowledge my parents for their intense support who motivated and consoled me to go on my own way, beyond them it would have been difficult to outright my thesis.

Abstract

Humanoid robots are extensively discussed in modern days. The movement task and manipulation of Humanoid Robots is examined based on mobility of platforms and control of the arm. This project describes a robotic arm that is analogous to an arm of a human being. Some important parameters to be considered are reachability, stability and manipulability.

This thesis aims at adapting a humanoid robot arm for performing movement operation that can be used for various purposes. The proposed robot arm has 3 motors on the left arm and 3 motors on the right arm thereby constituting a total of 6 motors. This operation can be achieved by the use of sensor like ultrasonic sensor. Here Beaglebone Black, an open source linux based controller board is used. The Beaglebone Black acts as the main controller for the entire system. A research is also being made to implement the robotic arm using Robot Operating System (ROS) platform. ROS is preferred since it is modular, simple and easy to use tools for development, it provides good hardware support, lots of algorithms are implemented together as package, etc.

Table of Contents

Acknowledgements.....	ii
Abstract.....	iii
List of Figures	vi
List of Acronyms.....	ix
Chapter 1 Introduction	1
1.1 Introduction	1
1.2 Rationale	2
1.3 Aims and Objectives.....	3
1.4 Overview of Thesis.....	3
1.5 Outline of the Thesis	4
Chapter 2 Methods of implementation of the Project	6
2.1 Robot Arm model.....	6
2.2 Implementing Humanoid Robot Arm movement	7
2.2.1 Gesture controlled arm using accelerometer	7
2.2.2 Haptic Controlled Robot Arm.....	7
2.2.3 Robotic Arm using Raspberry Pi.....	8
2.4 Conclusion.....	9
Chapter 3 Overview of working details.....	10
3.1 Hardware Components	10
3.1.1 Beaglebone Black	10
3.1.2 Ultrasonic Sensor	16
3.1.3 USB2Dynamixel Connector	17
3.1.4 Dynamixel AX-12A Motors.....	18
3.1.5 CM-530 Controller	18
3.2 Software Architecture	21
3.2.1 Choice of software used	21
3.2.2 Architecture of ROS.....	21
3.2.3 ROS Concepts	25
3.3 ROS Code Hierarchy	27
3.3.1 Communication between Nodes	27
3.3.2 ROS Ecosystem	27
3.4 Publishing and subscribing using ROS.....	29
3.4.1 Publishing Messages	29

3.4.2 Subscribing Messages	30
3.5 ROS Package.....	30
3.5.1 ROS Dynamixel motor package.....	30
3.6 Implementation of the Project	30
3.6.1 Installing the Operating System.....	30
3.7 Installing Debian OS in Beaglebone Black.....	31
3.7.1 Main Process.....	31
3.7.2 Debian Wheezy on a microSD card.....	32
3.7.3 Flashing Debian Wheezy to eMMC.....	32
Chapter 4 Manoeuver of the Robot arm for Grasping task.....	33
4.1 Introduction	34
4.2 Models and Materials	35
4.2.1 Humanoid Robot Arm specifications	35
4.2.2 Ultrasonic Sensor	35
4.3 The Experimental work	37
4.3.1 Program Structure.....	38
4.4 Working.....	42
4.5 Block Diagram	43
.....	43
4.6 Conclusion.....	43
Chapter 5 Conclusion and Future works.....	44
5.1 Conclusion.....	44
5.2 Contributions to the Research	44
5.3 Limitations.....	46
5.4 Future works	48
Reference List.....	49
Appendix A.....	53

List of Figures

2.1 Humanoid Robot arm model.....	6
3.1 Beaglebone Black Controller.....	11
3.2 Beaglebone Black switch locations.....	12
3.3 Beaglebone Black User Leds.....	13
3.4 Beaglebone Black key components.....	13
3.5 Beaglebone Black Block diagram.....	14
3.6 TPS65217C Block Diagram.....	15
3.7 Ultrasonic Sensor HC-SR04.....	17
3.8 USB2Dynamixel controller.....	18
3.9 Dynamixel Control using PC.....	18
3.10 Dynamixel AX-12A Motor.....	19
3.11 CM-530 controller.....	20
3.12 Parts of CM-530 controller.....	20
3.13 ROS Architecture.....	24
3.14 ROS and Camera Node.....	24
3.15 ROS with camera topic.....	25
3.16 Publishing and Subscribing.....	25
3.17 Code Hierarchy.....	28
3.18 Communication between Nodes.....	29
3.19 ROS Master.....	30
4.1 Flowchart of working model.....	38

4.2 Connecting Ultrasonic sensor with Beaglebone Black.....	39
4.3 Working of Ultrasonic Sensor.....	40
4.4 Flowchart for the working of Dynamixel Motors.....	43
4.5 Interfacing the motors with Beaglebone Black.....	44
4.6 Block Diagram.....	45

List of Tables

Table 3.1 Comparison between Software	23
---	----

List of Acronyms

API	Application Program Interface
ARM	Acorn RISC Machine
CMOS	Complementary Metal Oxide Semiconductor
DMS	Data Management System
DDR3	Double Data Rate Type 3
eMMC	Embedded Multi Media Controller
GA	Genetic Algorithm
GMM	Gaussian Matrix Model
GMR	Gaussian Matrix Regression
GNU	GNU's Not Unix
GPIO	General Purpose Input Output
GPU	Graphics Processing Unit
LCD	Liquid Crystal Display
LCM	Light weight Communications and Marshaling
LED	Light Emitting Diode
OpenCV	Open Source Computer Vision
OS	Operating System
PIC	Peripheral Interface Controller
PID	Proportional Integral Derivative Controller
PWM	Pulse Width Modulation
RAM	Random Access Memory
ROS	Robot Operating System

RL	Reinforcement Learning Algorithm
RPC	Remote Procedure Call
SVM	Support Vector Machine Algorithm
SMPS	Switch Mode Power Supply
TCP/IP	Transmission Control Protocol/Internet Protocol
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
VCS	Version Control System
XML	Extensible markup language
YARP	Yet Another Robot Platform

Chapter 1 Introduction

1.1 Introduction

Humanoid Robot resembles a human body. The purpose of a humanoid robot is to interact with human beings and environments for experimental purposes such as locomotion, picking and placing things etc. In general Humanoid robot consists of Torso, a head, two arms and two legs[1]. Humanoid robot is a challenging research field that has received significant deliberation nowadays[2]. Sustainable researches are carried under the Humanoid Robot platform to make it to move and implement well designated tasks[3]. The Robotic arm resembles a mechanical arm that is identical to a human arm. The networks of the arm are associated with joints that allow rotational or linear displacement. The end effector of the robot can be used to perform many tasks like picking and placing objects [4],[1],[3] spinning, welding depending upon the type of application. There are different types of Robotic arms such as Cartesian Robot, Cylindrical Robot, Polar or Spherical Robot, SCARA Robot, Articulated Robot, Parallel Robot, Anthropomorphic Robot, etc. [5]

One of the most controversial issues related to humanoid robot research to be control mechanisms, modularity and Inter-platform operability in their operating system[6]. To overcome this issue, Robot Operating System (ROS) architecture [7], [6] which is simple to navigate and manage in prototype circumstance has been employed as the software architecture to support both real-time hardware and simulation within this project. Beaglebone Black, an open source Linux based board is used to deploy a Human like robot having 18 degrees of freedom is examined. Robot Operating System (ROS) [8] is implemented in the Beaglebone Black controller to increase the interactivity of the robot. Ultrasonic sensor is attached to the robot arm for detecting the presence of an object. The main objective of this research is to present a method with sufficient flexibility to be made potentially suitable for different scenarios of object perception and handling.

Python Programming and ROS are freely available for the user to develop[9]. There are many sensors like Ultrasonic sensors, IR sensors, DMS Sensor and Gyro sensor are used in the robot. The Beaglebone Black consists of additional ports which can be used to interface additional sensors in future. The robot uses

Dynamixel AX-12A motors. There are six Degrees of Freedom (DOF) in the mechanical end effector, three motors on the left arm and other three on the right arm.

The Robot Operating System (ROS) [6] is practically equivalent to that of a working arrangement of a Personal Computer, which involves accumulation of projects that offers control to clients. If there should be occurrence of ROS, regardless of these projects admit a client to control portability of the robot instead of using it on a PC. The Robot Operating System is preferred because of its modularity, Inter-platform operability and Concurrent resource handling. Beaglebone black [10] is an inexpensive board which possibly lessen framework usage cost. This development platform features a microprocessor AM335x, based on ARM cortex A8 processor, so it is possible to run some Linux distributions in it and is for C++ and additionally Python Programming.

There are few techniques by which Humanoid Robot arm [11] is being modified and planned. The technique utilizes different controllers for controlling the Humanoid robot arm. The controllers are Arduino, Raspberry Pi and so forth. Uncommon programming methods are likewise being taken after to enhance the outline of robot arm. The programming strategy utilizes Arduino Programming[12], C++ programming[12], Python programing[13], MATLAB [12] and so on.

1.2 Rationale

The work exhibited in this research project uses Beaglebone Black as the principle controller and Robot Operating System (ROS) [13] as an operating system. Greater part of the mechanical frameworks don't utilize ROS since it has a few downsides like overhead of the messaging system[14]. A portion of the benefits of ROS are

- Inter-Platform operability – Can be utilized with assortment of programming dialects.
- Modularity – makes the framework more vigorous. In the event that one part crashes the entire framework does not crash.
- Concurrent resource handling – share assets effectively by distributing and subscribing to messages.

The ultimate aspiration of this research project is to present one method with adequate tractability to be possibly applicable to different scenarios of object handling and grasping [15]. Another improvement is the capability to have a robot safely and effortlessly inserted in an unexplored environment. This is desirable due to the introduction of an adaptive guidance used in the method.

1.3 Aims and Objectives

The main aim of this research project is to build a Humanoid Robot arm to simulate the behaviour of human using Beaglebone Black and Robot Operating System (ROS). The next aim of this project is to build the humanoid robot arm for grasping the object. The final aim of this research project is to make full analysis about the Robot Operating System for grasping the object. This project includes robot arm, Beaglebone Black, Ultrasonic Sensor, USB2Dynamixel connector and Dynamixel motors for control of a robotic arm.

The Objectives of this project to achieve the aims are

- To understand the working configurations of the Beaglebone black controller.
- To understand the operation of the dynamixel motors.
- To perform detailed analysis about ROS for performing the grasping operation.

1.4 Overview of Thesis

To perform a grasping action the first step for a humanoid robot is that the several body parts should be able to interact. There are several necessary features required for grasping an object. The first feature is reachability which assures the robot to grasp an object, the second feature is stability that guarantees the robot to remain balanced while moving or grasping an object, and the third feature is manipulability that makes the robot to manipulate the object efficiently after grasping. Many researchers are carrying out research works under this area[18]. This humanoid robot arm grasping problem is carried out in different ways by different researchers[15].

To perform the task of Humanoid robot arm grasping, different controller boards like Raspberry Pi, Arduino etc. are being used. Here GNU octave, which is a freely accessible language identical to MATLAB is also being used. This language is used for numerical computing[12]. It provides considerable graphics proficiency for data visualization and manipulation.

Robotic arm can also perform operations using MATLAB [12] and Arduino based controllers. Here image processing technique is used to control humanoid robot arm using human hand. In this method the motion of the robot arm is manipulated by inspiring human hand in different directions. Here a PIC micro-controller is used which consists of a CMOS camera that can be used, from which the location of the object can be resolved. MATLAB is used to record the human arm position using different X and Y axes to manipulate the robotic arm. Respective position of the hand is designated with different color code to observe the position.

The next method considers the robot grasping problem that uses grippers, section and motion capability. This method reformulates the object grasping task as simple two dimensional repetition resolution issues. The planes defined here are as follows; the first plane is manipulability plane. Here the end effector of the robot is determined with respect to the robot arm. The robot's section and shoulder is determined in support of stability and manipulability of the whole body in the second plane. By linking these two planes ideal solution is obtained.

Humanoid robots are universally reviewed in the modern years. The motion planning and control of humanoid robots can be discussed based on its movement and control of its arms. The robotic arm is designed with any representation of degrees of freedom (DOF). A performance based on inverse kinematics accession is used to solve the problem in terms of fuzzy reasoning was learnt. To demonstrate the path planning and control of the planned system, PC based simulation OpenGL 3D libraries are used. OpenGL is a short language, cross platform application programming interface (API) for analyzing 2D and 3D graphic visuals. The API is combined with a graphics processing unit (GPU), to manage hardware accelerated translation.

1.5 Outline of the Thesis

To understand the work done in the project, the subsequent lines give a small portrayal of what is addressed in each chapter. Chapter 2 details the art work of the Humanoid Robot arm, the Literature review discussing contemporary work about the robotic arm and the types of algorithms used. Chapter 3 presents an overview of the working details. It presents the architecture of the work, the hardware components and the software architecture as well as the packages being used in the project. Chapter 4 gives the manoeuver of the robot arm for

grasping task and lastly Chapter 5 presents the future works and conclusion of the research project.

Chapter 2 Methods of implementation of the Project

This chapter discuss about the implementation of the robotic arm implemented by various methods. It starts with an overview of the robotic arm and then follows with the explanation of some of the algorithms and techniques used for controlling the robot arm.

2.1 Robot Arm model

The Robotic arm considered in this research project consists of 6 DOFs in total. That is the arms of the robot constitutes 6 Dynamixel AX-12A motors [21] in total. The left arm comprises of 3 dynamixel motors and the right arm comprises of other 3 dynamixel motors. The figure 2.1[22] represents the model of humanoid robot arm.

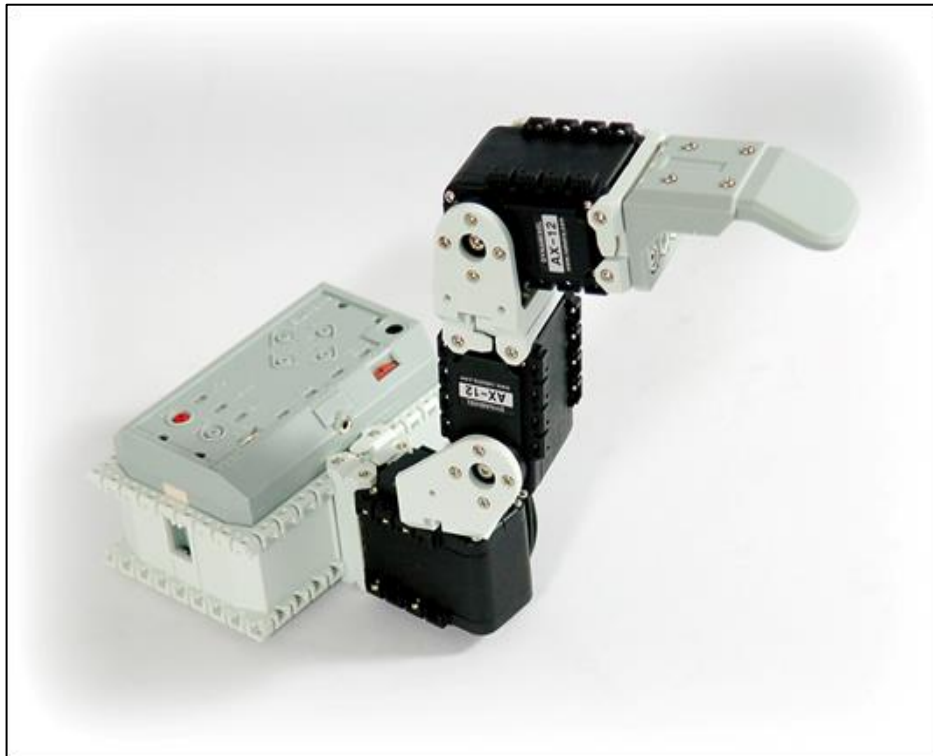


Figure 2.1 Humanoid Robot Arm Model

CM-530 controller is used as the default controller for this robot. The programs are inbuilt within the CM-530 controller for performing different operations. In this research project CM-530 controller is replaced by the Beaglebone Black [23]. Python language is used to program the Beaglebone Black. It has one USB port which is used to connect the USB2Dynamixel and the motors. The USB2Dynamixel connector is capable of driving more than 200 motors at a time[24]. The serial protocol of the dynamixel motor is daisy chained which means several servos can be connected in a chain with each other. Common external power supply of 12v is given to power all the dynamixel motors.

2.2 Implementing Humanoid Robot Arm movement Inverse Kinematics

Inverse kinematics is the use of kinematic equation of the robot to determine the joint parameters which provides desired position of the end effector. Inverse Kinematics [25] requires different approach. Instead of using vector algebra we provide a pure geometrical solution. Taking inverse of rotation/translation matrices will not give inverse kinematics solution.

2.2.1 Gesture controlled arm using accelerometer

Humanoid Robot arm can also be controlled using gesture movement. In this method the robot arm is controlled using human hand gesture and an accelerometer. This method makes use of the transmitter and the receiver units. The transmitting unit is mounted on the transmitting end and it is worn using a glove. This consists of accelerometer and a microcontroller for processing the signals. The receiver end consists of a RF receiver, microcontroller and servo motors. The entire arrangement is placed in a mobile unit that can be controlled using wireless remote control.

The accelerometer is used to capture the human hand behaviours. It measures the accelerations. These readings are analysed to create mathematical relation with PWM control of motors. The communication can be performed using Amplitude shift keying. Codes corresponding to different set of ADC values are transmitted and checked at the receiving end. If the received code is correct then the corresponding PWM value is used to drive the motor.

2.2.2 Haptic Controlled Robot Arm

This method is proposed by other researchers to implement the movement of the robot arm. In this proposed work a device is designed to replicate the operator's actions that also recognise the sensor information to carry out the desired action

by providing desired commands to the actuator. The hardware components used here are Arduino Mega 2560 controller, Flex sensors, Inertial Measurement unit and servo motors.

The robotic arm is an electro mechanical device consisting of arm and hand part. The servos are controlled by the signals from the flex sensors. The arm is controlled by the IMU signals. The implementation of the arm consists of two parts haptic glove and the robot arm.

The haptic glove is fitted with a microcontroller. Inertial measurement unit and flex sensors are connected with the micro controller. A wireless module is also attached with it which can be used to transmit and receive signals. The other part is the robot arm. The robotic arm is fitted with Arduino micro controller. Servo motors are attached with this controller. Here also a wireless module is attached to transmit and receive the signals. IMU is used to sense the motion of the arm. The operation of the robotic arm is performed by moving the haptic glove. The signals are transmitted and received using the wireless modules between the glove and the robotic arm.

2.2.3 Robotic Arm using Raspberry Pi

The movement of the robotic arm in this method makes use of Raspberry Pi controller and Arduino. Raspberry Pi is an open source linux based board. A camera module is also used to capture the image of the object. Image processing technique is used to process the captured image. Using this technique the objects can also be sorted with three different shapes and size. The robotic arm considered here consists of 4 degrees of freedom. The Arduino board used here is Atmega 328 controller. GNU octave software, which is compatible with MATLAB, is used.

A camera module is used with Raspberry Pi for capturing the image of the object. For image processing, first the image is captured through Pi camera and it is stored as image.jpg with 600x400 resolutions. Specific shape detection and colour code detection algorithms are used. Initially the IR sensor detects the presence of the object. If the image is present the raspberry pi camera is switched ON and captures the image. Then the shape and colour of the object is processed and its output is saved in a text file. The Raspberry Pi sends the output signal to Arduino using python script which reads the output from two text files and sends serial message to Arduino which is used to interface Raspberry Pi and the servo motors via the servo controller with the robotic arm.

Arduino receives the incoming message and the motion of the robot arm is controlled accordingly. Motor controller board is connected to the Arduino board and the Arduino also communicates to the controller via serial communication. These instructions are given by the Arduino to the controller accordingly the arm picks the object and sort the object based on colour and shape.

2.4 Conclusion

These are the different methodologies used for implementing the robot arm. These methods are introduced by other researchers to make the robot arm to grasp the object.

Chapter 3 Overview of working details

This Chapter examines the overview of the hardware components and the software used for the project.

3.1 Hardware Components

The hardware peripherals proposed in the research project are Beaglebone Black [23], [10] that acts as the main controller for the robot arm, the ultrasonic sensor which will be used to disclose the existence of objects, camera module that is used to capture the detected image, USB2Dynamixel connector that is used to connect the motors with Beaglebone black controller and Dynamixel AX-12A motors which is the arm [54] of the robot.

3.1.1 Beaglebone Black

Beaglebone Black controller acts as the main controller for the robotic arm. The Beaglebone Black controller [55] is the newest member of Beagleboard family. Beaglebone Black is a modest device, open source and flexible platform for experimentation. This is a linux based board. The board has AM335x [56] 1Gigahertz ARM cortex A8 chip with 512 megabytes of DDR3 RAM. It also has 4gigabytes of 8-bit eMMC on board flash storage for booting and long term storage purposes. The image of the Beaglebone Black controller is shown in figure 3.1 [10].

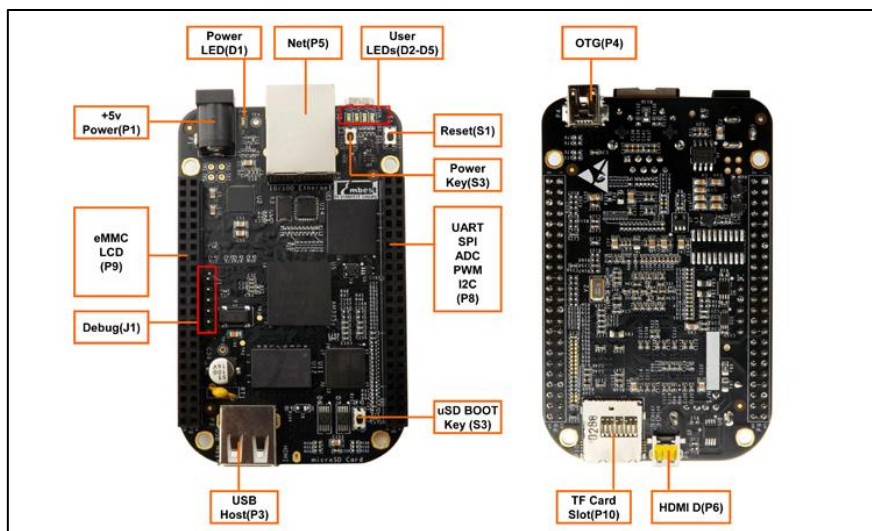


Figure 3.1 Beaglebone Black controller

The Beaglebone Black runs Debian OS [57] and it is programmed using Python which is an open source. One can build assorted software for eloquent purposes. The use of Beaglebone Black in this design has made it a cost adequate solution.

3.1.1.1 Beaglebone Black features

The Beaglebone Black controller's switches and locations are shown in figure 3.2 [10].

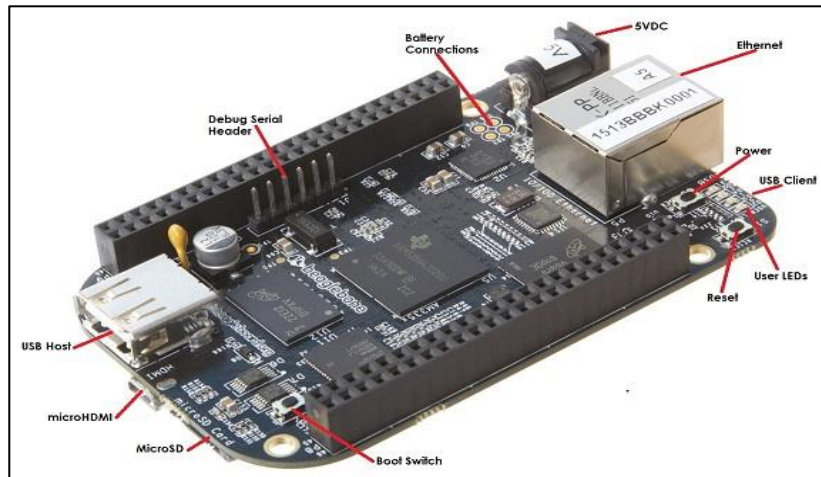


Figure 3.2 Beaglebone Black switch locations

- This controller board accepts 5V power supply.
- Power down sequence is initiated by the processor by alerting the power button.
- The connection to LAN is 10/100 Ethernet.
- Serial Debug is the Serial Debugging port of the controller.
- Mini USB connection is the USB client to PC that powers the board.
- Boot can be forced from the microSD card using BOOT switch.
- Four blue LEDs are there which are usable to the end user.

After the Power LED starts to blink within ten seconds other LEDs starts to flicker in their delinquent configuration. USR0 is constructed at boot to blink in a heartbeat arrangement. USR1 is configured at boot to light during microSD card approach. USR2 is configured at boot to light during CPU action. USR3 is configured at boot to light during eMMC connection.

- The processor can be reset by the reset button.
- MicroSD card is installed in a microSD slot.
- The display is connected to the micro HDMI connector.
- USB interfaces like Wi-Fi, BT, Keyboard, etc. can be connected to USB host.

Figure 3.3 [10] shows the user LEDs of Beaglebone black controller.

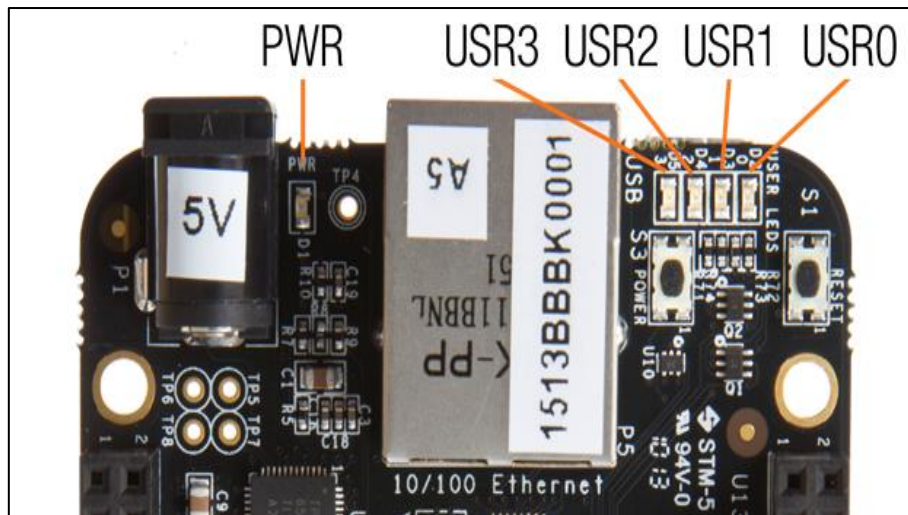


Figure 3.3 Beaglebone Black user LEDs

3.1.1.2 Beaglebone Black Key Components

The figure 3.4 [10] below shows the parts of the key factors on the PCB design of the board.

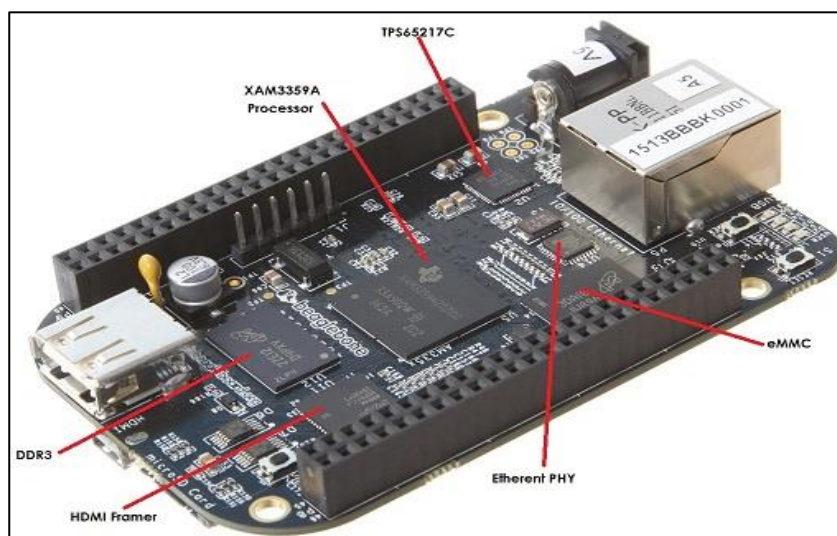


Figure 3.4 Beaglebone Black Key Components

- The main processor of the board is Sitara AM3359AZCZ100.
- The Dual Data Rate RAM memory of the controller is micron 512MB DDR3L.
- The various components of the board are powered by TPS65217C PMIC.
- The physical interface to the network is SMSC Ethernet.
- The on-board MMC chip is micron eMMC which can hold up to 2GB data.
- Control for an HDMI or DVI-D display with an adapter is provided by HDMI framer.

3.1.1.3 Block Diagram

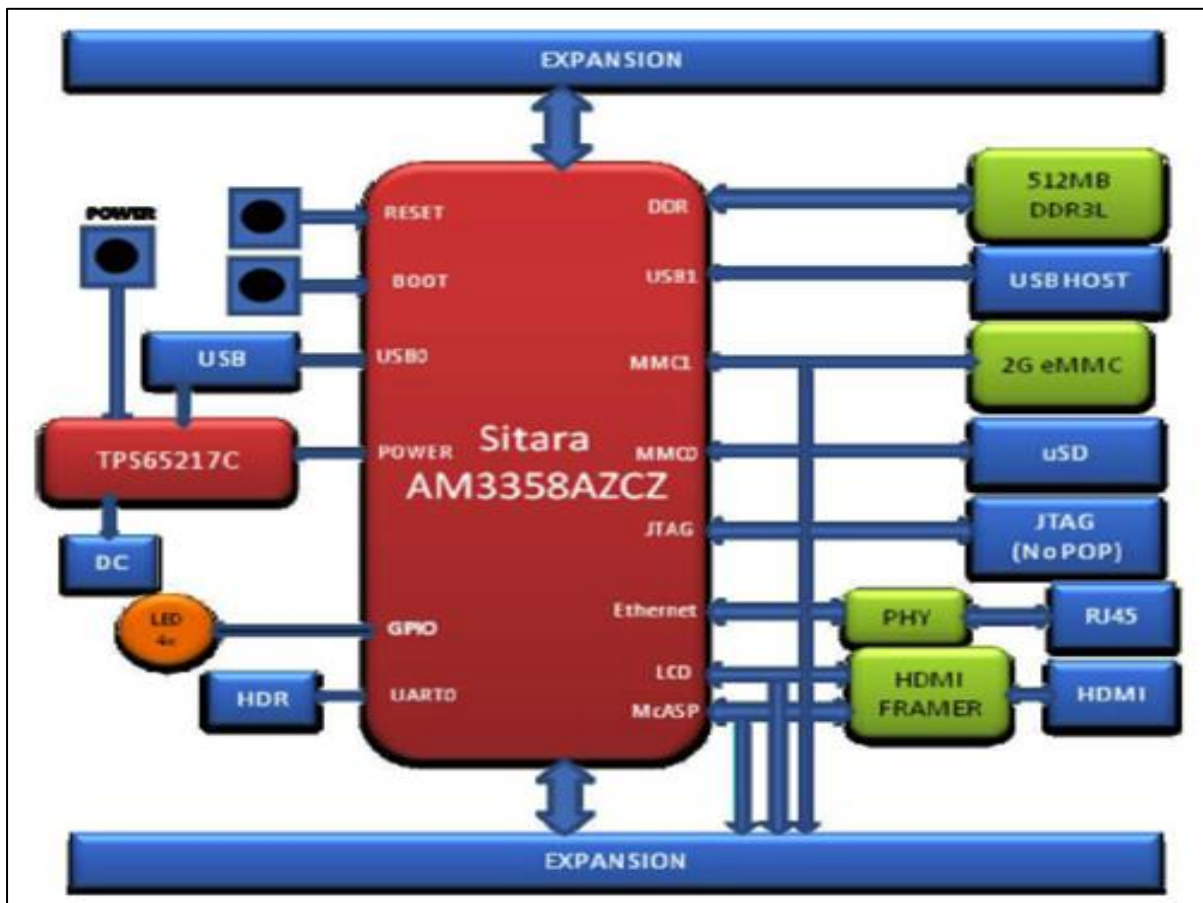


Figure 3.5 Beaglebone Black block diagram

The central Power authority IC for the entire system is TPS65217C [59]. It is a single chip power management IC. It consists of linear dual input power path, three step down converters, and four LDOs. It has three high efficiency 2.25MHz step down converters. They are targeted to provide the core voltage, MPU, and memory voltage for the board. The step down converters gets into low power mode at light capacity for maximum adaptability across broadly achievable range of load currents. Small foot print solution size is attained by the step down converters by allocating small inductors and capacitors. For low noise applications these mechanism can be enforced into fixed frequency PWM using I2C interface [60].

LDO1 and LDO2 support the system standby condition. During normal operation these LDOs support up to 100mA. LDO3 and LDO4 support 285mA each. LDO1 is consistently ON by default but either rail can be configured to remain in SLEEP state. In particular the DCDC converter remains up in low power PFM condition to support processor suspend mode. Flexible power up and power down sequencing is offered by TPS65217C. It provides considerable functions like pushbutton monitor, hardware rest function, temperature sensor to protect the battery and power-good output. The block diagram of TPS65217C is shown in figure 3.6 [59].

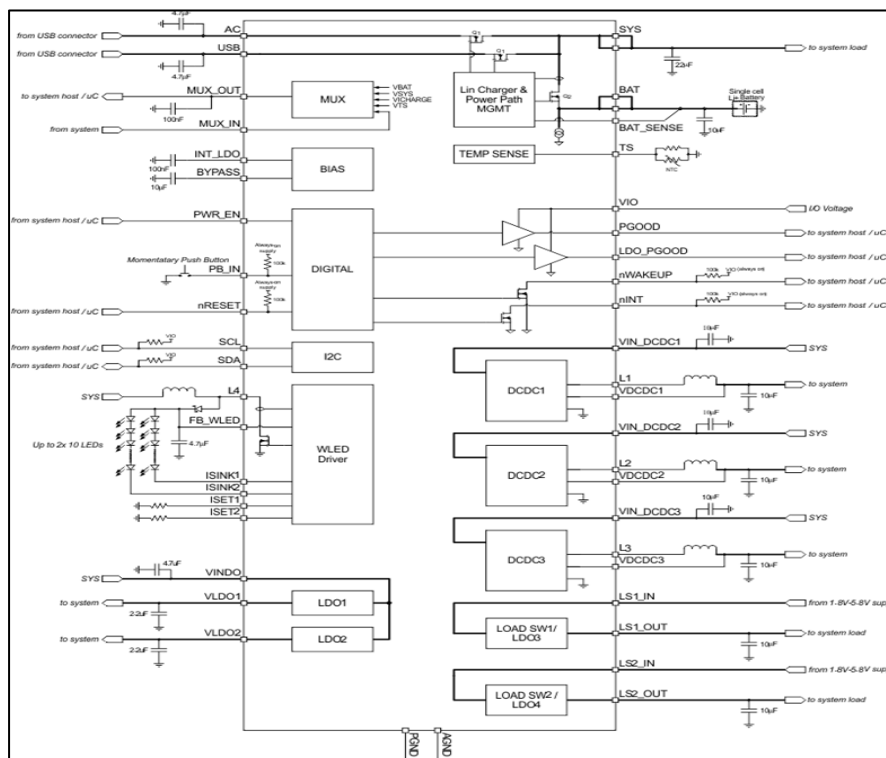



Figure 3.6 TPS65217C Block Diagram

3.1.1.4 Beaglebone black PIN configurations

The pin out diagram of the Beaglebone Black is represented below.

Beaglebone Black Pinout Diagram								
P9					P8			
Function	Physical Pins	Function	Function		Physical Pins	Function		
DGND	1	2	DGND	DGND	1	2	DGND	
VDD 3.3 V	3	4	VDD 3.3 V	MMC1_DAT6	3	4	MMC1_DAT7	
VDD 5V	5	6	VDD 5V	MMC1_DAT2	5	6	MMC1_DAT3	
SYS 5V	7	8	SYS 5V	GPIO_66	7	8	GPIO_67	
PWR_BUT	9	10	SYS_RESET	GPIO_69	9	10	GPIO_68	
UART4_RXD	11	12	GPIO_60	GPIO_45	11	12	GPIO_44	
UART4_TXD	13	14	EHRPWM1A	EHRPWM2B	13	14	GPIO_26	
GPIO_48	15	16	EHRPWM1B	GPIO_47	15	16	GPIO_46	
SPIO_CSO	17	18	SPIO_D1	GPIO_27	17	18	GPIO_65	
I2C2_SCL	19	20	I2C_SDA	EHRPWM2A	19	20	MMC1_CMD	
SPIO_DO	21	22	SPIO_SLCK	MMC1_CLK	21	22	MMC1_DAT5	
GPIO_49	23	24	UART1_TXD	MMC1_DAT4	23	24	MMC1_DAT1	
GPIO_117	25	26	UART1_RXD	MMC1_DAT0	25	26	GPIO_61	
GPIO_115	27	28	SP11_CSO	LCD_VSYNC	27	28	LCD_PCLK	
SP11_DO	29	30	GPIO_112	LCD_HSYNC	29	30	LCD_AC_BIAS	
SP11_SCLK	31	32	VDD_ADC	LCD_DATA14	31	32	LCD_DATA15	
AIN4	33	34	GND_ADC	LCD_DATA13	33	34	LCD_DATA11	
AIN6	35	36	AIN5	LCD_DATA12	35	36	LCD_DATA10	
AIN2	37	38	AIN3	LCD_DATA8	37	38	LCD_DATA9	
AIN0	39	40	AIN1	LCD_DATA6	39	40	LCD_DATA7	
GPIO_20	41	42	ECAPWMO	LCD_DATA4	41	42	LCD_DATA5	
DGND	43	44	DGND	LCD_DATA2	43	44	LCD_DATA3	
DGND	45	46	DGND	LCD_DATA0	45	46	LCD_DATA1	

The Beaglebone Black [61], [62] consists of large number of pins. These pins are divided into two headers. The pins present on the left side are represented as header P9 and the pins present on the right side of the controller are represented as header P8. The pins are represented using different colours used for various purposes.

Red colour pins – They are the power, ground and reset pins. The power pins may be of 5V, 3.3V or 1.8V. The VDD_ADC is 1.8V power supply used to provide reference for analog read functions.

Green colour pins – These pins are general purpose GPIO pins. Some of these pins can be used for UART serial communication.

Purple colour pins – These pins are generally used for PWM output. These pins can also be used to simulate analog output between 0 and 3.3V.

Light blue colour pins – These pins are used as analog in. The analog in reads between 0 and 1.8V.

White colour pins – These pins are used for shared I2C bus connections.

Orange colour pins – These pins are generally used for LCD screen applications.

3.1.2 Ultrasonic Sensor

Ultrasonic sensor is used to detect the presence of object. They calculate the acoustic waves having frequency higher than hearing range. They act by bringing about high frequency vibration of sound and then it receives and evaluates the echo pulse. Ultrasonic sensor is based on three physical principles. The principles are

- Time of flight – The distance can be measured by time of flight method
- Doppler shift – The velocity can be sensed using this method
- Amplitude attenuation – Attenuation coefficient, distance and directionality can be sensed.

This sensor does not require substantial association with the target. The evaluations are very conscious to temperature and angle of target. It has high accuracy and the measurements are not affected by sun light or black material.

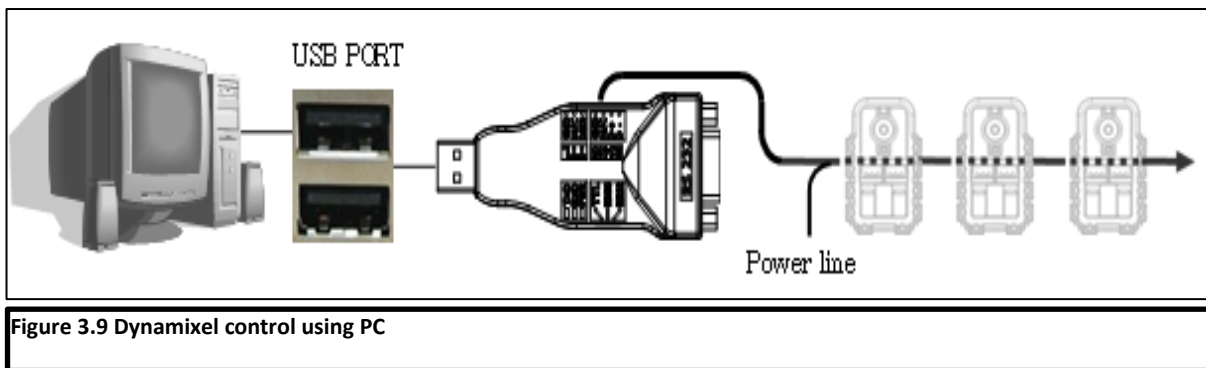
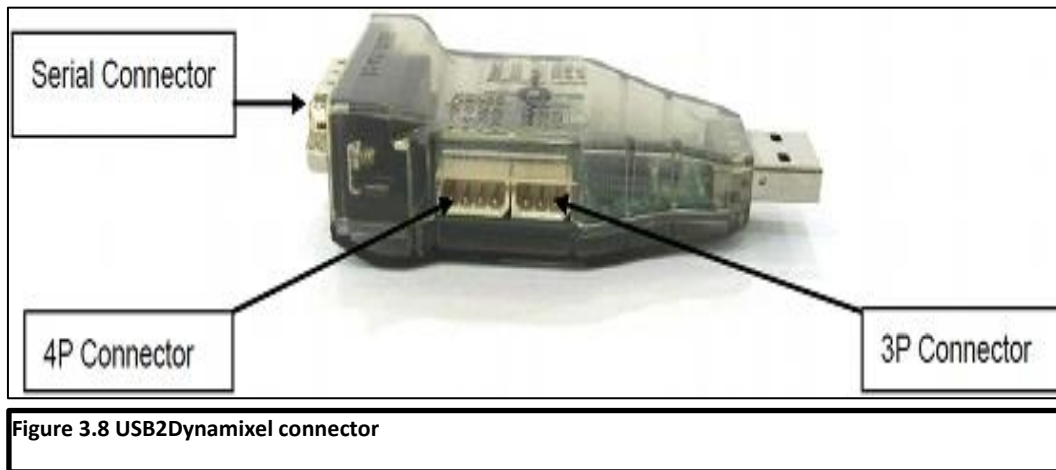
The figure 3.7 [63] represents an ultrasonic sensor.



Figure 3.7 Ultrasonic Sensor HC-SR04

3.1.3 USB2Dynamixel Connector

The USB2Dynamixel [24] is a device used to connect the dynamixel motors with the main controller or to the PC through USB port. It is equipped with 3p and 4p connectors to be connected with different dynamixel motors. The USB2Dynamixel does not provide power to the dynamixel. Power must be given separately in order to operate the dynamixel motors. Figure 3.8 [24] is an USB2Dynamixel connector.



The figure 3.9 [24] shows the dynamixel control using a computer.

Serial connector – Changes from USB port to serial port through RS-232 communication.

3P connector – used to connect Dynamixel of AX series through TTL communication.

4P connector – used to connect dynamixel of DX, RX series through RS-485 communication.

3.1.4 Dynamixel AX-12A Motors

Dynamixel AX-12A motor [64], [65] is deliberately controlled servo motor which can provide 50 regulations and evaluation parameters that allows full control over the robot applications. It is a new version of AX 12+ with advanced external design and improved gear durability. The motors are connected in daisy chain, which means that several servo motors can be associated with each other in a group. Servo motor works on PWM [62](Pulse width modulation) principle which means that the angle of rotation can be controlled by the extent of the enforced pulse. The servo motor is made of DC motor. It can be controlled by a variable resistor (potentiometer) and few gears. The position of the motor is decided by electrical pulse. The Dynamixel motors are stronger and have a robust control system that reports the position, temperature and torque. The communication in dynamixel motor takes place over a half-duplex UART port. Figure 3.10 [21] shows a dynamixel AX-12A motor.



Figure 3.10 Dynamixel AX-12A motor

3.1.5 CM-530 Controller

The CM-530 controller [66] is a small controller used in robotic applications. The CM-530 controller consists of 32 bit Arm Cortex M3 processor, a CPU, TTL Communication board (3 pin dynamixel bus), status LED, input buttons, GPIO ports, buzzer and sound sensor. The CM-530 [66] is suitable with



Figure 3.11 CM-530 Controller

dynamixel AX and MX series motors. This controller support Bluetooth and Zigbee communication. It can be programmed with Embedded C and it is compatible with RoboPlus software. The CM-530 controller [66] is connected with PC through USB port. The configuration of the controller is shown in figure 3.11 [66] and figure 3.12 [66].

PC Link – It is used to interface the serial cable with CM-530 controller. It is used for communication with PC.

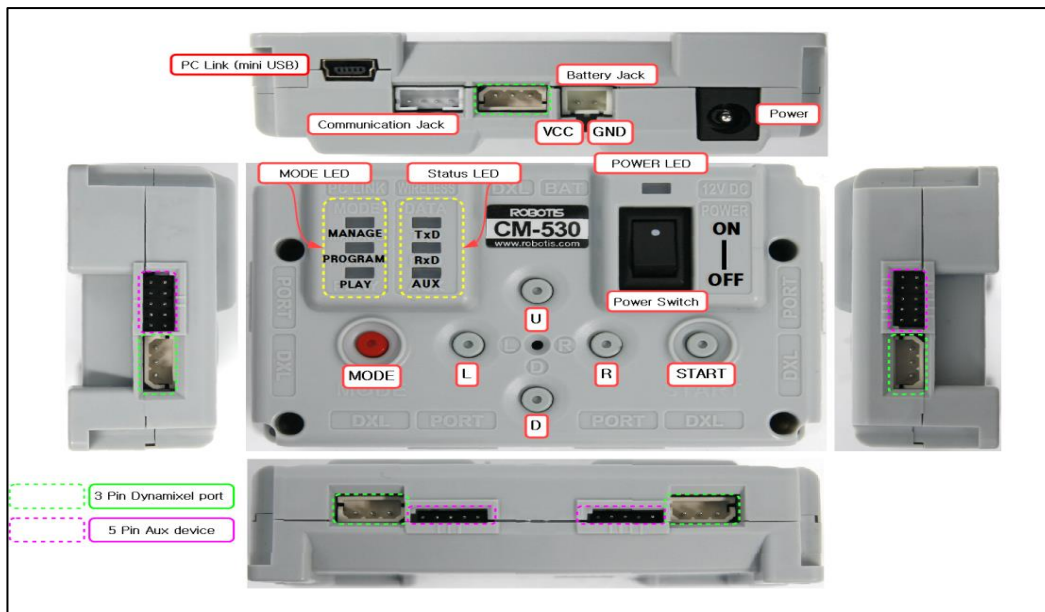


Figure 3.12 Parts of CM-530 Controller

Communication Device Connection Jack – this jack is used as a wireless communication medium with Zigbee and Bluetooth.

Battery Jack – battery is connected using this jack.

Power Jack –the SMPS power supply is connected using this jack.

Power LED – The ON and OFF status are shown.

Power Switch – It is used to turn on or off the robot.

Mode Button –the mode of the controller can be changed using the mode button.

Start Button – Selected mode can be started from the controller using start button.

U/L/D/R button – These are the navigation buttons used to navigate the robot when it is switched on.

AX/MX series port –This port is used to connect Dynamixel AX and MX series motors

Peripheral device connection port – Peripheral devices and sensors like touch sensor, IR sensors are connected.

Mode display LED – Present operating mode of the controller is shown by this LED.

CM-530 controller is not used as the main controller in this research project. This controller is used to learn about the basic working mechanisms of the dynamixel motors used in robot arm, its operation when connected in daisy chain method. The CM-530 controller has many disadvantages when compared to Beaglebone black controller. It has only less memory which is not compatible with many of the industrial robot applications. Also the Beaglebone black can be used with any OS distributions whereas the latter controller does not support OS distributions. The controller is pre-programmed with some minimal functionality whereas our main controller supports C++ and Python program [17] which can be re-programmed any number of times according to our need.

3.2 Software Architecture

The Software architecture used in this project is Robot Operating System (ROS) [6]. It is a malleable structure for programming the robot. Complicated and vigorous robot performance can be facilitated by the presence of libraries, conventions and tools. ROS [67] is an open-source software package which is similar to an operating system for the robot. It provides assistance like hardware abstraction, low level machine control, implementation of frequently used components, message passing between process and package administration.

3.2.1 Choice of software used

The software used in this project is ROS [57]. There are many choice of robot software that can be used in robots as shown in table 3.5. They are

- Light weight Communications and Marshalling (LCM)
- Yet Another Robot Platform (YARP)
- Robot Operating System (ROS)

3.2.2 Architecture of ROS

The ROS [67] Master constitutes the main part of the Robot Operating System. The communication between the nodes takes place with ROS Master. The Master grants all other ROS [68] fragments (nodes) for communicating with each other. It consists of many resources like packages, meta packages, repositories etc.

The responsibility of communication with the camera takes place with the help of a camera node, the image data is processed by image handling node on the robot arm and the image can be displayed using image display node on the screen. All the nodes should be certified with the master to start communication with each other. Figure 3.13 [69] shows the architecture of ROS.

Table 3.1 Comparison between software

LCM	YARP	ROS
It is a group of libraries and tools for message passing and data marshalling.	It is an open source software package for interconnecting sensors, processors and actuators.	It is a collection of software frameworks that provides services such as message passing, hardware abstraction, package management etc., The action takes place in nodes that receive post and multiplex sensor, control, state, planning, actuator and other messages.
It contributes publish/subscribe message passing model.	Publish / subscribe message passing model available.	ROS consists of nodes that publish and subscribe messages. A single node can publish/subscribe to multiple topics.
Supported Languages: C, C++, Java	Supported Languages: C, C++	Supported Languages: C++, Java, Python, LISP
Graphical tools are not available.	Graphical tools are not available.	It supports graphical representation which uses Rviz for graphical representation.
It cannot sync several different services to one device.	It cannot sync several different services to one device.	ROS can sync several different services to one simple device.
It does not have separate packages for everything.	It does not have separate packages for everything.	It has separate packages for each and everything that can be installed easily.
It has only few dependencies.	It has only few dependencies.	ROS has many dependencies.
Features: low latency, efficient broadcast mechanism, user friendly.	Features: great flexibility with sensors, hardware. Used in mobile robot vision applications, efficient.	Features: publish/subscribe all data streams, multiplexing information, segmentation and recognition, motion control and planning, grasping, object identification etc.

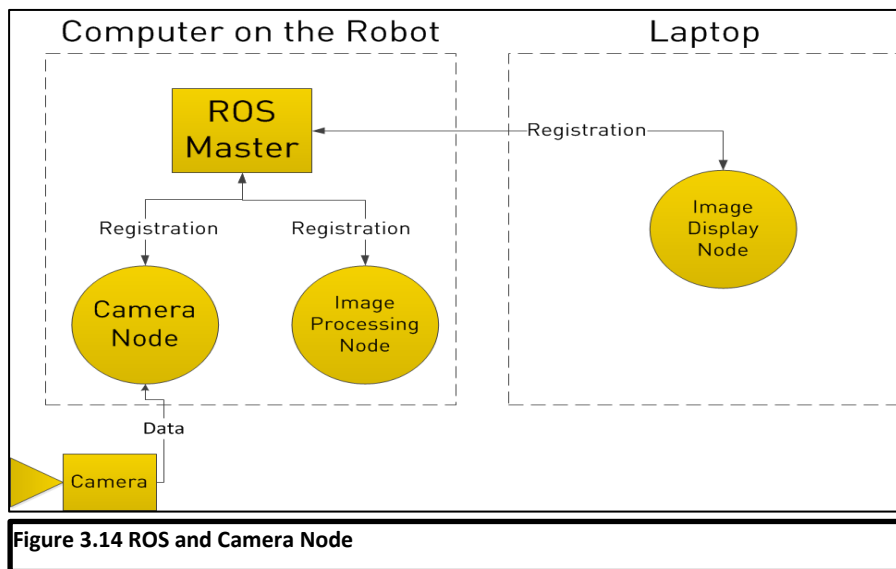
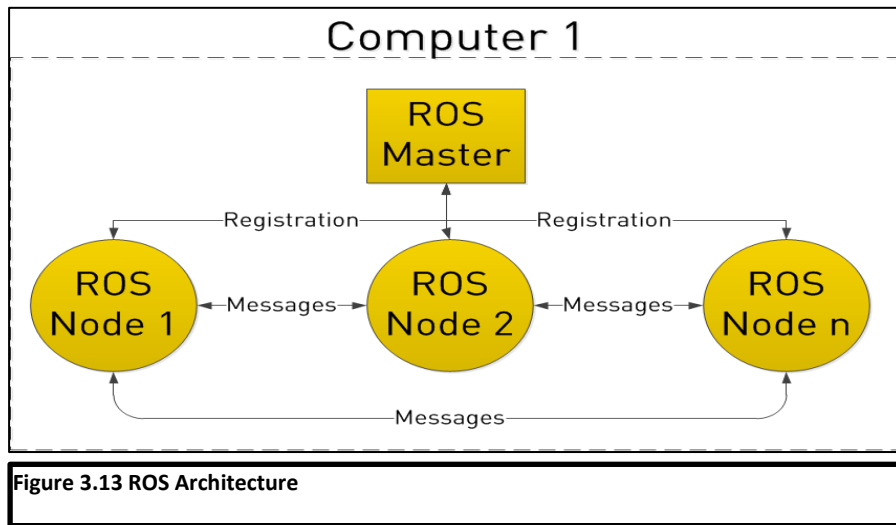


Figure 3.14 [69] shows communication between ROS and the camera node.

A topic can be published by the camera node only after registering with the ROS master and the other nodes are subscribed to that respected topic. Once the camera node starts to receive some data from the camera, the topic is assigned directly to the other two nodes. A node can register a unique service with the ROS [67] master. The image processing node requests the topic and the camera node gathers the data from camera and then sends the acknowledgement. The camera topic is shown in figure 3.15 [69].

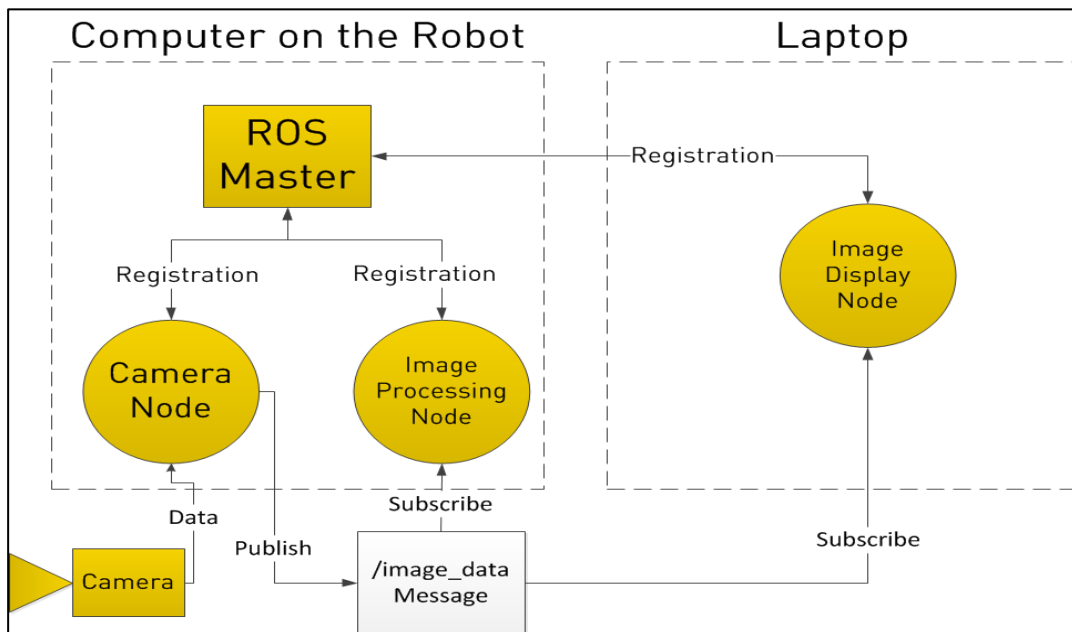


Figure 3.15 ROS with camera topic

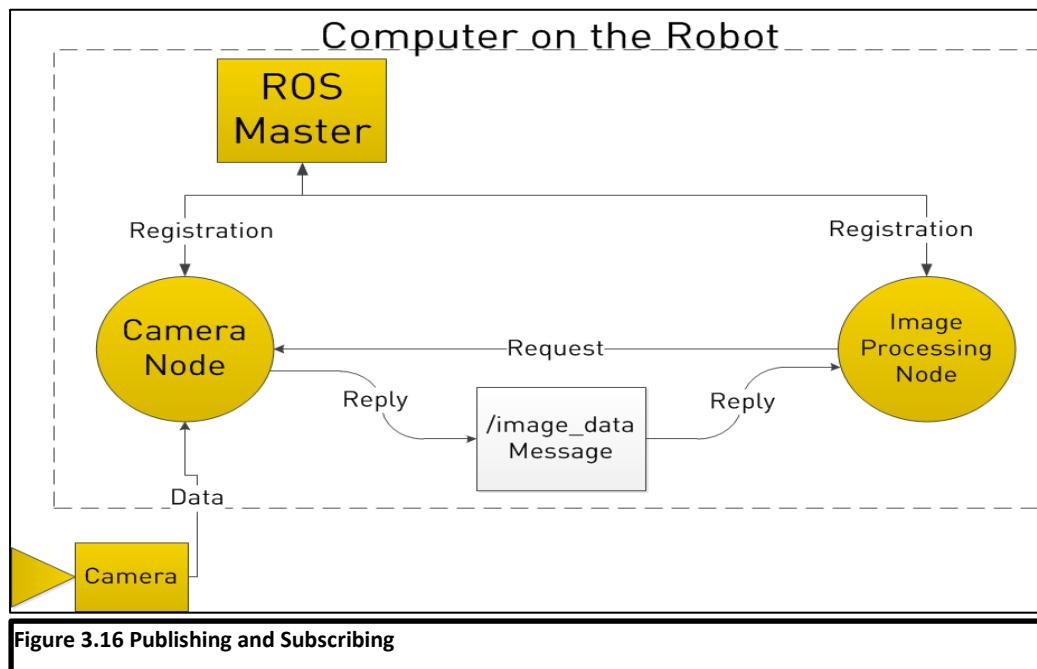


Figure 3.16 Publishing and Subscribing

The publishing and subscribing topics is shown in figure 3.16 [69]. The ROS ecosystem consists of its core stack (roscore and catkin), dedicated tools (Rviz and rqt_graph) and third party ROS packages. Roscore is a compilation of nodes and programs which are prerequisites of ROS based structure. The roscore should run all the time in order for ROS [8] node to broadcast. Cmake macros and combined python codes are used to frame parts of ROS that are categorized under Catkin. The sensor data and state data from ROS are visualized from Rviz, which is a 3D visualizer. It is possible to visualize current structure using Rviz on a virtual model of the robot. Live representations of the

sensor values can be demonstrated subsequent over ROS [6] topics including information from a camera, measuring distance using infrared sensor, data from sonar and more. Rqt_graph builds a dynamic graph of what is going on, on the system. It is a component of rqt package. It Provides Rqt_graph as a GUI plugin for visualizing ROS graphs [70].

3.2.3 ROS Concepts

Robot Operating System has three concepts. They are File system level, computational graph level, and community level. The ROS [71] File system level consists of Packages, Meta packages, Package demonstrate, Repositories, Message (msg) types, and Service (srv) types.

Packages are the fundamental entity for organizing software in ROS. A package consists of nodes, ROS-reliant libraries, datasets, and structure of files that are formulated together.

Meta Packages are particular packages that only serve a group of other related packages. Meta Packages [6] do not install files other than their package.xml and they do not contain any tests, codes, files or other items that are present in packages.

Package Manifest is an XML file called package.xml that must be included with any catkin compliant package root folder. Manifest contributes metadata about a package along with its name, description, version, dependencies and other Meta information like exported packages.

A Repository [68] is a collection of package that shares a trivial VCS system. Packages that share a VCS, shares same version. ROS repository contains only one package.

The value of the data is described by simple message description language published by ROS nodes [13]. The source code can be easily and automatically generated by ROS tools for type of message in several target languages by the message type information. .msg file consists of two parts. Fields and Constants are the two types of .msg file. Fields are the data that is sent inside of the message whereas the constant defines the useful values that illustrates those fields. Message types are assigned to use package resource name.

Robot Operating System (ROS) [6] uses a reduced service description language for describing its service types. This builds directly upon the ROS message configuration to enable request/response connection between nodes. These

descriptions are gathered in .srv files in the srv/subdirectory of the package. The service descriptions are referred to use package resource name.

The associative network of Robot Operating System (ROS) is computation graph level that processes the data together.

Node is a process which performs calculation. They communicate through remote procedure call services, parameter servers and streaming topics. A robot control system usually consists of many nodes and these are represented to operate at fine grained scale. The naming and listing services are implemented by the ROS to rest of the nodes in ROS system. It also records publishers and subscribers topics and also the services. The ROS master facilitates the individual nodes to locate one another. Once the nodes are located they communicate with each other peer to peer. The command to initialize the ROS master is roscore [6]. Once this command is executed the ROS master loads along with all other essential components. A Parameter Server is a common, multi-variant dictionary. It can be accessed through network APIs. ROS naming convention [14] is used to name the parameters.

The messages are published to the topics that facilitate the communication between the nodes with each other. Transferring request and responding messages are performed by nodes which is a part of ROS [6] service call. Srv files define the request and response messages. The data structure of a message can be specified by message files. They are stored in a package inside a message subdirectory. The Topics are the communication medium where the messages between the nodes are exchanged. They have unidentified publishing and subscribing definition from where the production of instructions are separated from utilization. Multiple numbers of publisher nodes and subscriber nodes are present in a topic. Topics are performed solely for communicating only in one direction. Publishing to a topic is done by ROS message types whereas the nodes receive the message types that are identical. ROS node [68] provides service beneath a string name and client calls the service by sending the desired message and waiting for the acknowledgement.

Bags are used for saving and playing back ROS message data. They are designed by tool called **roscap** which can subscribe to more than one ROS topic. These files can be played back in ROS to the same topics from where they are recorded.

3.3 ROS Code Hierarchy

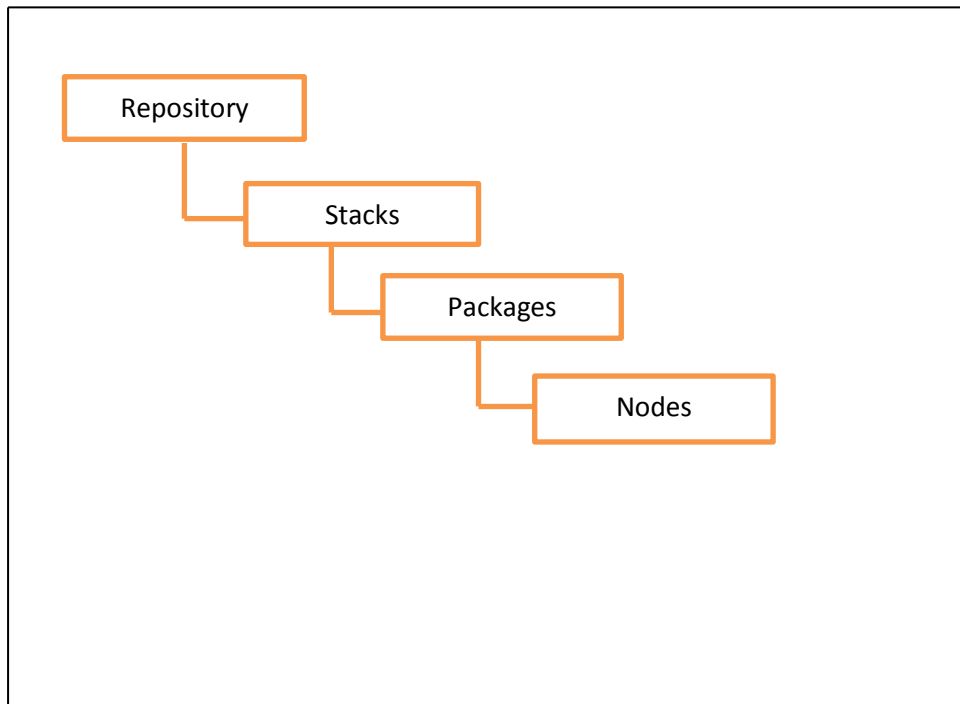


Figure 3.17 Code Hierarchy

Repository – It contains all codes from a particular development group.

Stack – It groups all the code on a particular subject or device.

Packages – They separate the modules and provide different services.

Nodes – They are the executable which exists in every model

3.3.1 Communication between Nodes

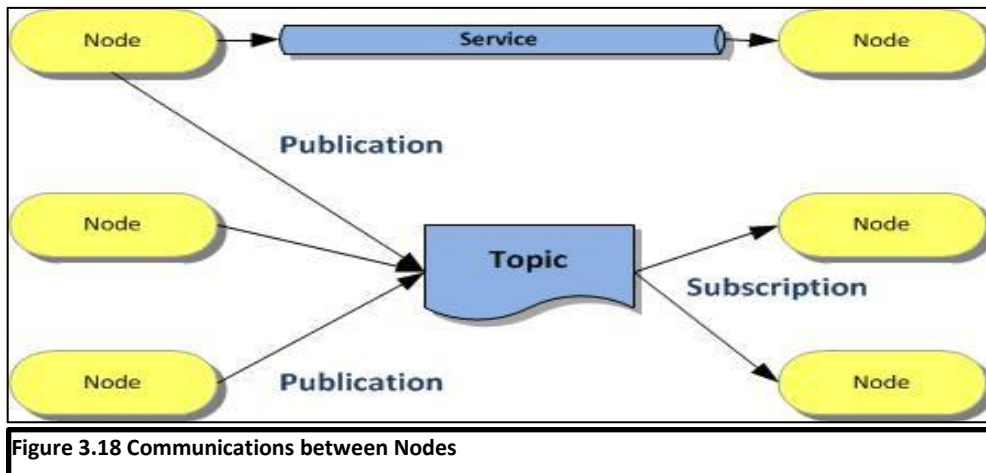
A running instance of a ROS program is called as a node. Messages are sent by nodes by distributing it to a particular topic. The content of the message is identified by the topic name. A node interested in certain kind of data will subscribe to a relevant topic. There are many publishers and subscribers for a topic and a single node can subscribe many topics.

A ROS node can be written in rospy or roscpp. Here roscpp is defined for C++ and rospy is defined for Python code.

3.3.2 ROS Ecosystem

The ROS Ecosystem [13] consists of its core stack (roscore and catkin), dedicated tools (Rviz and rqt_graph) and third party ROS packages. These

packages represent the bulk ecosystem having specific features that can be reused by other packages.



- **Core stack**

Roscore is a collection of nodes and programs that are prerequisites of ROS based system. The nodes that are launched as part of roscore are determined in roslaunch/roscore.xml. Roslaunch treats nodes listed in roscore.xml differently. The nodes running with same name are being searched and are launched.

- **Dedicated tools**

Rviz (ROS Visualization) is a 3D visualizer that displays sensor data and states the information from ROS. Using Rviz any user can visualize current configuration on a virtual model of the robot. The live representations of the sensor values are displayed that includes camera data, Infrared distance measurements, sonar data and more. Rviz can be initialised using the command “roslaunch rviz rviz”.

Rqt_graph creates a dynamic graph of the process going on, on the system. It is a part of rqt package. Rqt_graph provides a GUI plugin for visualizing ROS graphs. This can be initialised using the command “roslaunch rqt_graph rqt_graph”.

- **Third party ROS packages**

The third party ROS packages [6] are non-catkin packages. These ROS packages are compiled using third party library code that makes it available using rosbuilt’s dependency system.

```
started roslaunch server http://ubuntu:48429/
ros_comm version 1.4.8

SUMMARY
=====

PARAMETERS
* /rosversion
* /roscdistro

NODES

auto-starting new master
process[rosmaster]: started with pid [2463]
ROS_MASTER_URI=http://ubuntu:11311/

setting /run_id to 2299b920-2406-11e4-8013-000c29efce79
process[roscout-1]: started with pid [2478]
started core service [/roscout]
```

Figure 3.19 ROS Master

3.4 Publishing and subscribing using ROS

3.4.1 Publishing Messages

Messages can be published using C++ or Python programming. Every ROS topic is associated with a message type that has a corresponding header file. This header file should be #include in the program. The package name should be the name of the package containing the message. The object which is needed is created by the line

```
ros::Publisher pub = node_handle advertise <message_type>(topic_name,
queue_size);
```

The scope resolution operator is used to separate the package name from the type name. Here

- node_handle is the object of class ros::NodeHandle.
- message_type is the data type of the message that needs to be published.
- topic_name is the string which contains the name of the topic to be published.
- advertise represents the size of the message queue for the publisher.

Publishing messages to multiple topics can be done by creating separate ros::Publisher object for each topic. After all the work is completed the messages can be published using the command

```
pub.publish(msg);
```

After using this command the messages will be sent to any subscribers of the corresponding topic.

3.4.2 Subscribing Messages

The subscriber node does not know about the arrival of the messages. A code must be placed inside the callback function to respond to the incoming messages. For subscribing to a topic subscriber object should be created.

```
ros::Subscriber sub = node_handle.subscribe(topic_name, queue_size,
pointer_to_callback_function);
```

Here,

- node_handle is the same node that handles the object.
- topic_name is the name of the topic to be subscribed.
- queue_size is the integer size of the message queue for the subscriber.
- Pointer to callback function is executed by ROS when a message arrives.

When ros::Subscriber is constructed connection with any publishers of the named topic is being established.

3.5 ROS Package

ROS package is a comprehensible collection of files, that includes both executable and supporting file that serves a specific purpose. A package contains ROS nodes. Packages that we create should live together in a directory called workspace. A subdirectory is created inside a workspace directory as it contains the source code for the package.

3.5.1 ROS Dynamixel motor package

This contains the package that is used to interface various series of Dynamixel motors. This package supports AX and MX dynamixel motor models.

3.6 Implementation of the Project

3.6.1 Installing the Operating System

The Beaglebone Black acts as the main controller in this system. The Beaglebone black is compatible with much software. One can use Debian [76], Android, Ubuntu, Cloud9 IDE etc. in the Beaglebone black controller. Here Debian Wheezy is used as the operating system for Beaglebone black.

Choice of the Operating System

Among many operating systems Debian is used as the Operating system in Beaglebone black controller. The reasons for using Debian as an Operating system are

- All the packages are unified.
Debian outpaces all other software collections in terms of package integration. Here all software is packaged by a systematic group; all packages can be found at a single site. This integration between packages makes Debian more robust.
- Supports multiple architecture and kernels
Debian supports multiple architectures like alpha, amd64, armel, hppa, powerpc, s390, and sparc. It supports kernels like GNU Hurd, FreeBSD besides linux.
- Stability
Debian has good stability. It can run for over a year without rebooting unlike other systems that crashes multiple times a day.
- Good System security
Debian has good system security. It is very responsive in fixing security problems. The availability of open source code in Debian prevents poor security models being implemented.
- Easy Upgrade
Due to easy packaging system in Debian it can be easily upgraded to its latest version.
- Easy Installation
Debian can be installed easily using CD, DVD, USB stick or even using the network.

3.7 Installing Debian OS in Beaglebone Black

The Debian OS [56] has to be installed in Beaglebone Black controller. For installing the OS following steps should be followed.

3.7.1 Main Process

The instructions mentioned below are for users with Windows computer.

- The first step is to download the Debian Wheezy image from the internet. The image can be booted every time from a microSD card or the image can be flashed into the eMMC of the Beaglebone black controller.
- After downloading the .img.xz file, it should be extracted.
- Then the microSD card is inserted into the computer.
- The image is written into the microSD card using Win 32 disk imager software.
- After writing the image the microSD card should be removed.

3.7.2 Debian Wheezy on a microSD card

After completing the main process Debian image [14] is mounted in a microSD card. This method does not flash the image into eMMC. This method provides information to boot the image from the microSD card every time.

- Insert the microSD card containing the Debian Wheezy image into powered off Beaglebone black. Make sure that Ethernet is connected.
- Power is applied to the board by holding the boot button. The boot button should be held continuously until the led flashes. Use external power supply.
- After some time screen shows a login prompt. Give the user and password as Debian.
- Boot button should be hold every time to boot Debian image from microSD card otherwise the Beaglebone black boots whatever image found on its eMMC.

3.7.3 Flashing Debian Wheezy to eMMC

This method describes about flashing Debian image into the eMMC of the Beaglebone black after completing the main process.

- Insert the microSD card containing Debian image into powered off Beaglebone black.
- Power should be applied to the board while the boot button is held. The boot button is held continuously until the user LEDs starts to flash.
- After some time screen shows a login prompt.
- The board should be left for about 10 minutes until the image is being flashed into eMMC of Beaglebone black controller.

- After some time the LEDs will be solid and not flashing. Remove the power and microSD card and then the power should be re-applied to the controller.
- It takes some time to boot to the login screen. Give the user name as Debian and password as temppwd.
- Now the Beaglebone black will boot the Debian image from eMMC.

Chapter 4 Manoeuvre of the Robot arm for Grasping task

This chapter discusses the task of grasping an object using the Robot arm.

4.1 Introduction

This research project is based on Bioloid Robot arm [78], [79] for grasping objects using Robot Operating System (ROS) [6]. This system is chosen after careful consideration of the hardware and software components. The fundamental objective of this project is to establish a platform to integrate both the hardware and software components to make the robot arm to grasp the object.

The hardware part is the Beaglebone black [22], which is considered as the main controller of the entire system. It runs Debian [76] operating system with running ROS which is the main framework to integrate the entire software running in Beaglebone black controller. Ultrasonic sensor is used to detect the presence of object. The robotic arm uses dynamixel AX-12A motors to move the arm. The Beaglebone black consists of additional ports that can be used to interface other sensors. ROS architecture [70] is used as the main software framework because it is easy to manage and navigate. In case of ROS the programs acknowledges the user to manipulate the mobile operations of the robot arm. ROS [69] is preferred because of its modularity, inter-platform operability and concurrent resource handling. In previous working models the robotic arm is implemented using some algorithms. There are many controllers that can be used to implement the arm of the robot. Here Beaglebone black [80] is used because it is cost effective system which is a freely available linux based board.

The experimental setup consists of Beaglebone black, USB2Dynamixel connector, Ultrasonic sensor [81], Dynamixel motors [82]. Initially the ultrasonic sensor detects the presence of the object. If the presence of object is detected, the signal is stored in Beaglebone black controller. The entire process is controlled by the ROS.

So, the overall system behaviour can be explained as follows.

- ROS provides communication between the modules that codes and decodes messages between ROS [71] and the communication medium

used in the controller as it supports localization and motion planning modules.

- The packages which are being installed in ROS acts as a driver that allows the communication of the data between and allows data conversion that can be read by ROS applications.
- The communication of the data between the modules takes place easily by publishing and subscribing to the appropriate topics.

To understand the interaction between different systems, the following section describes the used hardware and software architecture covering the operating mode of each of the components.

4.2 Models and Materials

This part describes about the hardware peripherals used in this project.

4.2.1 Humanoid Robot Arm specifications

The Humanoid Robot arm [83] considered in this project consists of six dynamixel AX-12A motors. It has 6 degrees of freedom. The left arm [4]consists of three dynamixel motors and the right arm consists of other three dynamixel motors. These motors are connected with usb2dynamixel connector which in turn is connected with the main controller. The dynamixel motor is provided with 12V power supply. The motors are connected is daisy chain in which all the motors are connected in series with each other where a common power supply is given to all the motors.

4.2.2 Ultrasonic Sensor

The sensor used in this project is ultrasonic sensor [62]. It works according to three principles. They are time of flight, Doppler Effect¹ and attenuation of sound waves. These sensor measurements are very sensitive to temperature and the angle of target. The Doppler shift can be explained as follows. When the wave reflects from a moving object its frequency is shifted by an amount proportional to the velocity of the object.

$$f_e - f_r = 2f_e(v/c) \cos A \quad \dots (5.1)$$

Where,

A - Angle between the target's and the pulse's lines of motion

f_e - Frequency of emitted pulse

f_r - Frequency of received pulse

v - Velocity of the target

Usually the ultrasonic sound propagates faster than audible sound. The measurement of the distance transmitted can be calculated using the equation given below.

$$I = I_0 e^{-ax} \quad \dots (5.2)$$

Where,

I – It is the received intensity

I_0 –It is the original intensity

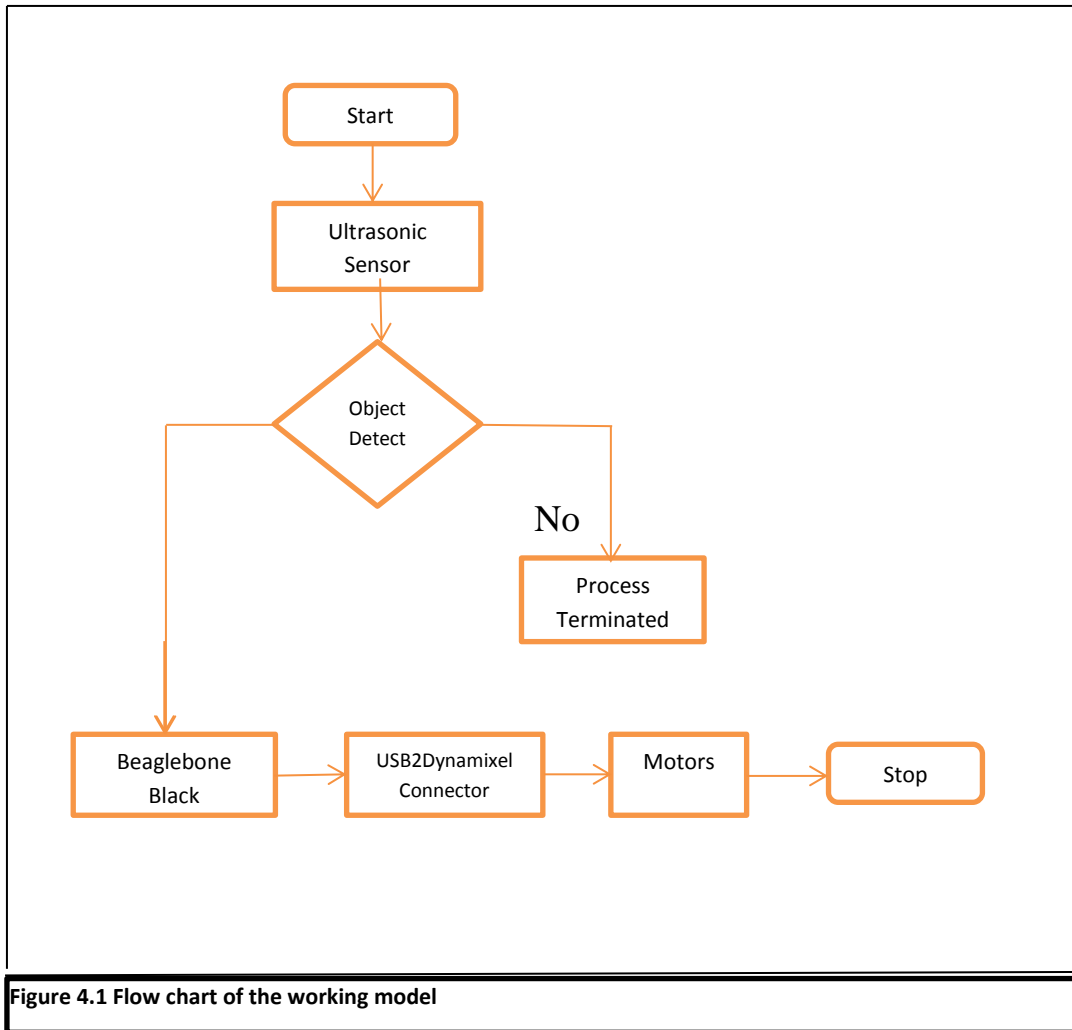
a –The attenuation coefficient

x – Distance transmitted by the wave

¹Doppler effect is the change in frequency or wavelength of a wave for an observer moving relative to its source.

4.3 The Experimental work

The main task of this project is to implement the robot arm to grasp the object [84] using ROS. Here Python script is used to implement the project. The flow diagram given below is the working of the robot arm for grasping [9] the object.



The flowchart of the working model is explained as follows. Initially the process gets started. After starting the process the ultrasonic sensor detects the presence of the object [63]. If no object is detected the process gets terminated. After receiving the signal the Beaglebone black [7] sends the information in the form of Python script to usb2dynamixel connector. After the connector receives the necessary signal the corresponding signal is transferred to the dynamixel motors to perform the task of grasping [77].

4.3.1 Program Structure

In this project python programming is used to perform the action. The hardware wiring of the ultrasonic sensor with the Beaglebone black [62] is shown. The Ultrasonic sensor consists of four pins namely Ground, Power, Trigger and Echo. The Power pin of the sensor is connected with pin 5 of the Beaglebone black and the ground of the sensor is connected to the pin 1 of the main controller. The trigger and echo are connected to the pins 12 and 11 pins of the P8 side of the main controller. Here the resistors are used because the Beaglebone black input pins are capable of delivering only 3.3V whereas the sensor emits 5V supply. In order to avoid the damage to the controller board the resistors are used. Figure 30 shows connection between beaglebone black and ultrasonic sensor. [62]

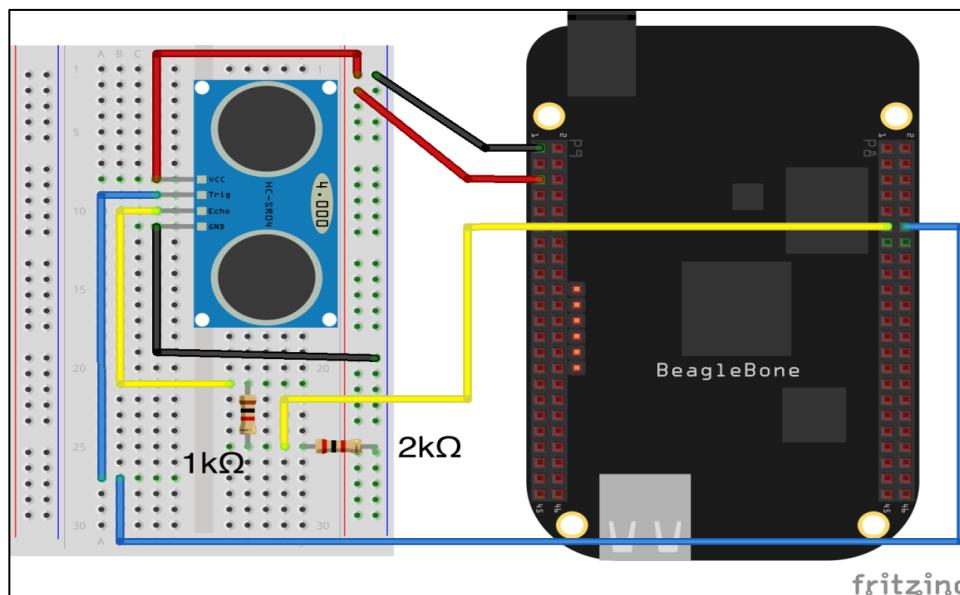


Figure 4.2 Connecting Ultrasonic sensor with Beaglebone Black

The flowchart explains the working of the ultrasonic sensor.

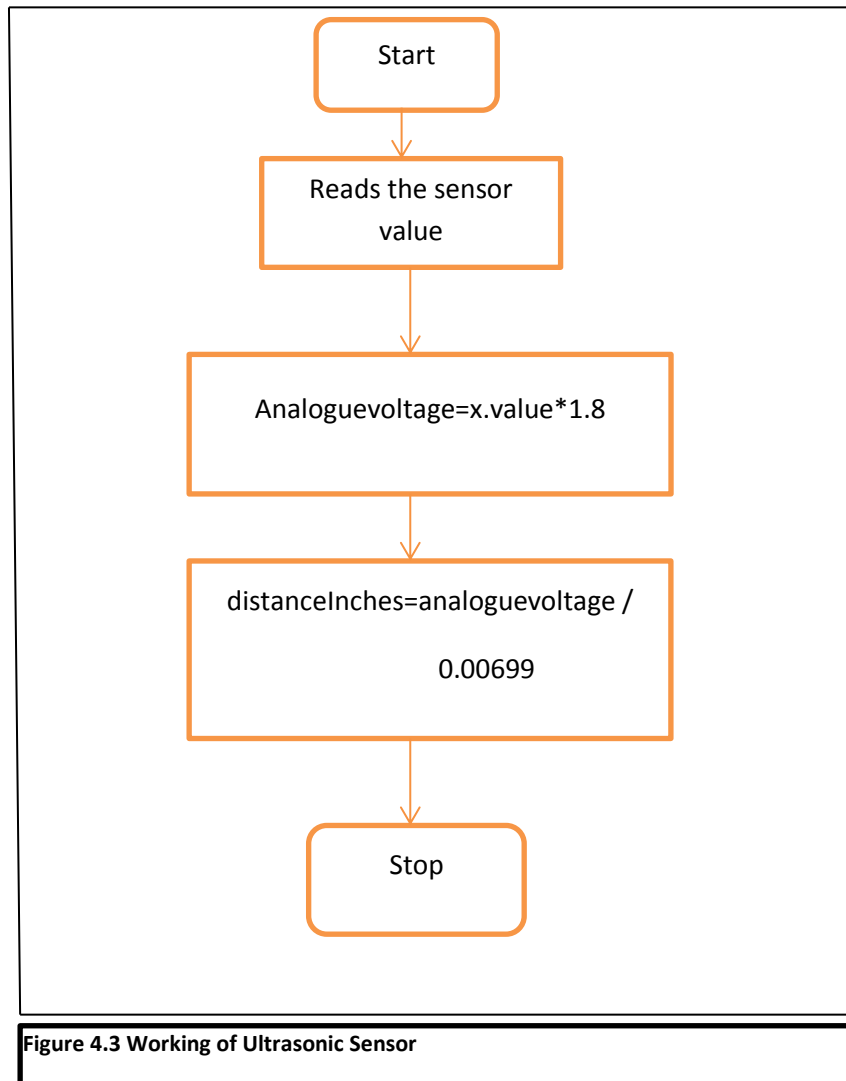


Figure 4.3 Working of Ultrasonic Sensor

4.3.1.1 Algorithm

Step 1: The process of object detection is started.

Step 2: The sensor value is read for every 2 seconds. The value is read in order to detect the presence of the object. The value can be read using a function.

Step 3: The ADC value is converted into Analogue voltage using the formula $\text{Analoguevoltage} = x.\text{value} * 1.8$

Step 4: The distance is calculated into inches. The formula used to convert into inches is $\text{distanceInches} = \text{analoguevoltage} / 0.00699$.

Step 5: The program is terminated.

The distance is calculated in centimetre. If the distance increases the voltage of the ultrasonic sensor also increases whereas if the distance is reduced the voltage of the sensor will fall. This can be shown in a graphical form as represented below.

4.3.1.2 Algorithm

Step 1: Initially the process for motor operation is started.

Step 2: The dynamixel checksum is calculated. The checksum is calculated using if condition. If not ($0 \leq id \leq 0xFD$) is used to calculate the dynamixel checksum. If the condition is false it displays as the respective ID is not legal. If the condition is true it proceeds to the next step.

Step 3: The enwiring of the dynamixel motor is checked. It is checked using if statement. If not ($0 \leq v \leq 1023$). If the condition is not satisfied it shows enwiring is illegal. If the condition is true, the control moves to the next step.

Step 4: After checking the necessary conditions the servos are interfaced with the controller.

Step 5: Serial port is provided to the servos in order for its operation to be performed.

Step 6: After providing the serial port reset is provided to the servos. If the servo motor misbehaves reset can be used.

Step 7: After providing the reset the next step is to define the set position. It is done using if statement. If not ($0 \leq position \leq 1023$) is the condition. If it is not satisfied then it is displayed as invalid position. If the condition is satisfied, the position of the servos should be set in degrees.

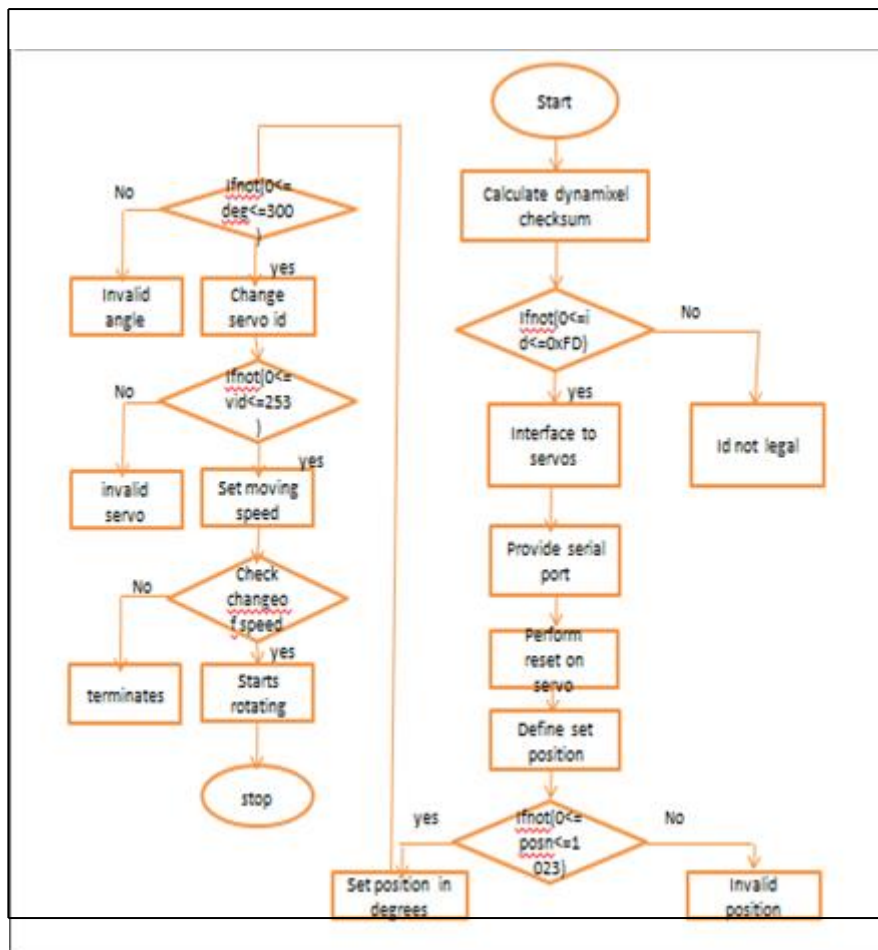


Figure 4.4 Flowchart for the working of dynamixel motors

Step 8: The maximum degree this servo will rotate is 300. If the angle is in between 0 and 300 the condition is satisfied otherwise it shows as invalid angle.

Step 9: The ID of the servo motor is changed since there may be multiple servos running at the same time. The ID of the servo should be in-between 0 to 253. If not $(0 \leq \text{id} \leq 253)$. If the condition is false it displays invalid servo ID. If the condition is true the speed of the motor is set.

Step 10: If the speed is in-between the stipulated range the motor starts to move else the process will be terminated thereby showing an error message. The steps 8 to 10 are repeated to perform the grasping action by putting the necessary conditions inside a loop.

Step 11: Stop the program.

4.4 Working

The block diagram consists of Beaglebone Black [22], ultrasonic sensor, USB2Dynamixel connector and robot arm [77].

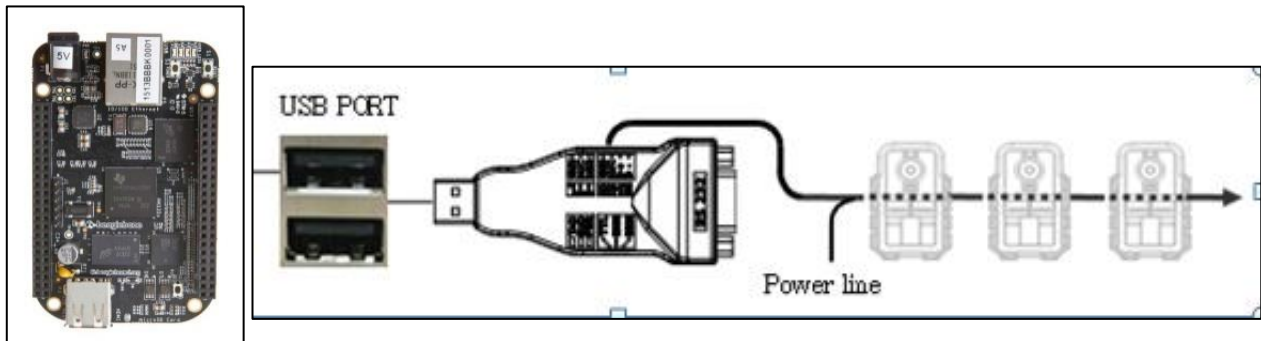


Figure 4.5 Interfacing the motors with Beaglebone black

USB2Dynamixel connector is used to interface the dynamixel motors with Beaglebone black. The USB port of the controller is used to connect the connector. Initially the object is detected using the ultrasonic sensor. After detecting the object, the controller transfers the information using python coding [17] and reads the output. The Beaglebone black in addition sends the message to USB2Dynamixel connector using python code that reads the output and sends serial message to dynamixel motors connected with the robot arm. The USB2Dynamixel connector receives the incoming message from Beaglebone black and appropriately it controls the motion of the arm. Here initially the robot arm is connected with wires to perform the action. Respective ROS packages are installed within the controller to perform all the work successfully.

This is a low cost mechanism that finds its way to perform pick and place the object thereby making this system in industries very cheap along with the same accuracy as that of a dedicated PC.

Outcome of the Project

The arm of the robot is successful in case of motion. Both the arms represent the movement. The speed of the motors can be controlled by the program and it can be changed according to our needs. The arms of the robots are unable to perform the grasping operation as it needs fine tuning and the program needs to be even more specific.

4.5 Block Diagram

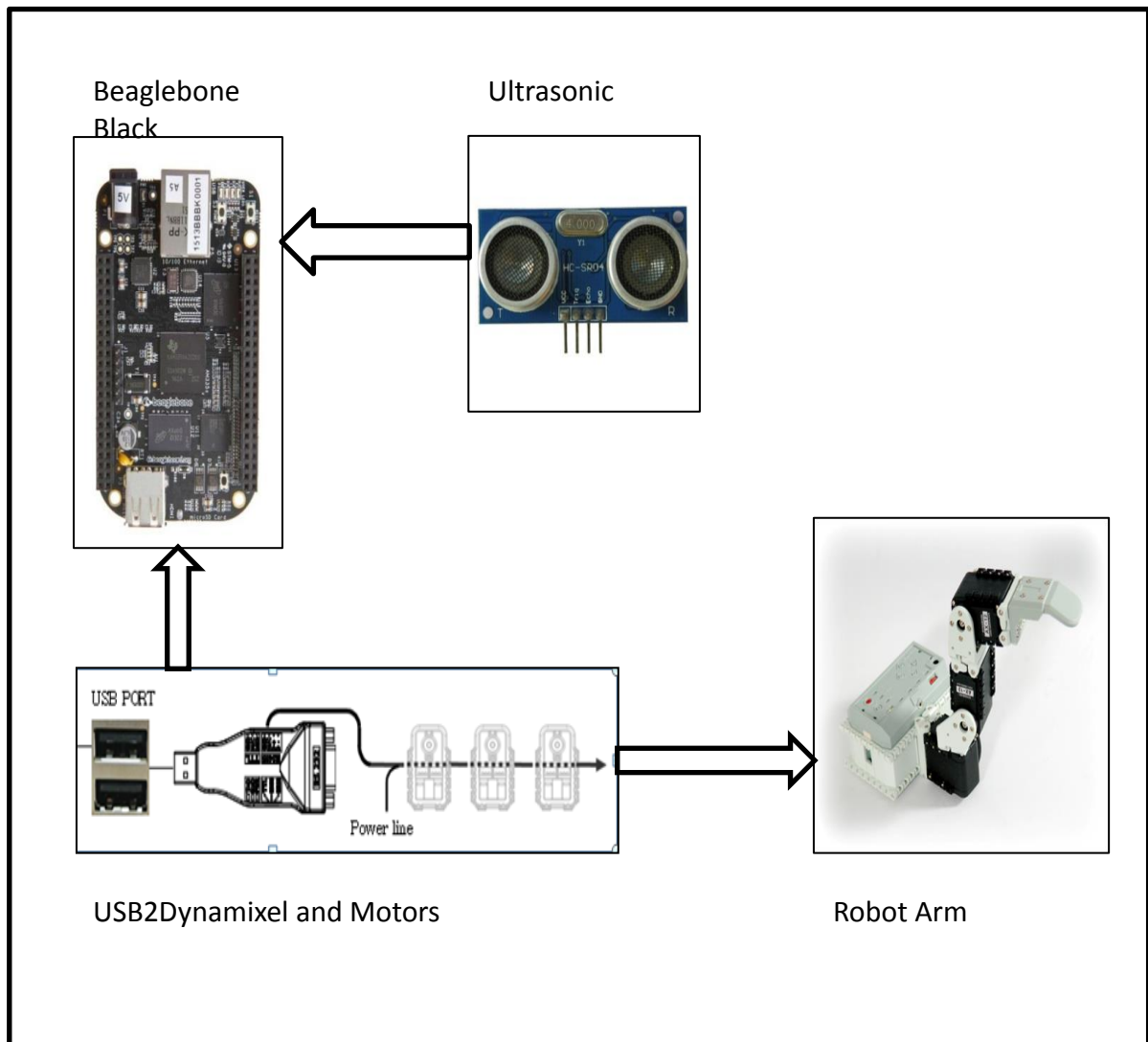


Figure 4.6 Block Diagram

4.6 Conclusion

In this chapter, using the ultrasonic sensor the distance can be detected and the motors were made to run using the python code.

Chapter 5 Conclusion and Future works

5.1 Conclusion

In this research project a robotic arm is being considered. We have explored the robotic arm for a humanoid robot with 6 degrees of freedom (DOF). The technology developed in this project consists of Robot Operating System software, robotic arm structure design, dynamixel actuators and ARM based controller.

- The Beaglebone black acts as a powerful controller for this project which comes along with inbuilt flash storage, higher processing speed, having more GPIO pins and much more. One disadvantage of Beaglebone black is that, it has lower graphic processing power so when using camera modules this controller may lag a bit behind when compared to other controllers.
- The USB2Dynamixel connector is used as it has the capability to run more than 200 dynamixel motor using a single connector by connecting the motors serially with each other.
- The use of dynamixel motors in this project adds advantage as it has high torque; high accuracy and the motors can be connected using daisy chain.
- The Ultrasonic sensor is used to detect the presence of object in this project. But this sensor have minimum sensing distance, also changes in environment affect ultrasonic response.
- The advantages of ROS include Inter-platform operability, modularity and concurrent resource handling. Relatively long learning curve may be necessary to master ROS.

5.2 Contributions to the Research

My Research concentrates for the most part on the movement of the robotic arms. The fundamental thought behind this is to execute the robot arm using Robot Operating System. My work focuses on the investigation of ROS and deduction techniques to encourage the arm utilizing ROS. In doing that I have sought after a few interrelated professions that traverse the numerous part of the issue from collecting the model using CM-530 controller, introducing the Robot Operating System in Beaglebone black, making the motors to run using ROS and python script, interfacing the Ultrasonic sensor, study on ROS and more. The following is a short synopsis of some of my key research commitments.

- . The principle contributions of this thesis are

- I have experienced many papers, productions and journals with respect to the robot arm usage that helped me to comprehend the utilization of various platforms. This correlation between the diverse stages turns into the establishment of the exploration of ROS, which is utilized as software platform.
- I have given an arrangement of standard strategies to enhance the execution parameter of the model. I have utilized ultrasonic sensor and Beaglebone black which goes about as the fundamental controller for the proposed framework as it backings extensive variety of equipment.
- To execute the robot arm utilizing ROS, I have made an examination between other working stages on how they slack when contrasted with ROS. Amid the investigation of utilizing ROS, I have looked into about it which helped me think about the components of the automated stage that upgrade my comprehension on Publishing and subscribing topics.

The other contributions of the research are as follows.

Operating Motors using ROS and Python Script: The dynamixel motor is used in the arm of the robot. To begin with the ROS dynamixel motor bundle is introduced, so it can bolster the dynamixel motor for operation. First the ROS dynamixel motor package is installed, so that it can support the dynamixel motor for operation. A package is created that contains the necessary files to run the motor. The controller manager is started by launching the launch file which connects the motors and publish feedback data. A joint controller is made, then a dispatch document is made and it is propelled. The controller manager node is begun first and it is published to a desired angle to move the motor. The motor is likewise keep running with python script. I have executed independently the running of motors utilizing ROS and python script.

Interfacing the Ultrasonic sensor: I have used the ultrasonic sensor to detect the presence of the object. The ultrasonic sensor is worked utilizing the python script. I confronted trouble when attempting to actualize the sensor utilizing ROS. I have connected the Echo and Trigger of the sensor is to the pin 11 and 12 of the header P8 in the controller. The ground and Vcc is associated with the individual pins of the Beaglebone Black. I have used a resistor between the Echo pin as the input pins are 3.3V while the sensor emits 5V in order to keep harm of the board.

Operating the motors using Python Script: I have utilized dynamixel AX-12A motors in the robot arm. Python coding is utilized to run the motors. To begin with I have encircled the code and I have stacked it in Beaglebone Black controller. At first I actualized the operation of a solitary motor by interfacing it with the controller utilizing usb2dynamixel connector. After fruitful execution of the work I have re-encircled the python code to make it work for running 3 dynamixel motors and succeeded working three dynamixel motors all the while in the meantime.

Survey on ROS: ROS is utilized as the robot platform for the entire research extend. I have considered and reviewed about the ROS commands, its usefulness that can be utilized as a part of this research project. I have made an examination about the software that can likewise be utilized like ROS however having a few disservices when compared with ROS. Communication between the nodes is also studied. These are trailed by numerous dozen papers which I have used to review about ROS.

5.3 Limitations

Beaglebone Black:

A portion of the impediments of the Beaglebone Black are as per the following.

- The microSD card cannot be used as a storage device when booting.
- Software Guarded Extensions is not given in 3.8 kernels.
- The latest image does not work on lap dock.
- This controller lingers somewhat behind when utilizing camera modules.

Limitations of ROS:

The Robot Operating System has some limitations. The limitations of ROS are

- Getting started with ROS is convoluted and troublesome as it is a fairly huge framework enveloping a geometry library, robot specification language, pose estimation, localization and mapping, navigation packages all entwined with a message-passing structure. It accepts entirely great working information of networking paradigms like remote procedure calls and publisher-subscriber. Working with ROS requires binding the code to ROS's package build system and gathering against its

dependencies. One of the greatest torment purposes of integrating third-party code or libraries into my particular undertakings is simply making sense of getting things to compile. If the build system software like Cmake, catkin and its basics of how the compilers and linkers work figuring out its establishment will be troublesome.

- The intersystem communication is marginally troublesome in rosbuilt contrasted with that of catkin. Both rosbuilt and catkin are build systems. Both build systems generate executable files from the source file. rosbuilt was the build system of ROS since the early circumstances. It utilizes CMake however summons it from a makefiles and builds each package independently and in source. Catkin has been introduced in groovy. It is utilizing CMake more neatly where it falls short on features. It builds packages out of source as a matter of course, handles dependencies better and improves the job of packaging. So in the event that you we pick we might need to run with catkin it might take a bit longer to learn CMake. But whenever we depend on a rosbuilt package we are forced to stick to rosbuilt. However catkin packages can only depend on other catkin packages. So, if we have a dependency that is still on rosbuilt, it needs to be converted first or rosbuilt should be used for the packages.
- ROS is good for material handling because of the fact that it endures powerfully changing conditions as new impediments are presented. But ROS has limitations in process control applications. It was never designed for processes like painting, spot welding or arc welding. Using ROS for small projects is intricate as teach pendants can be used to train robots.
- ROS shows error messages while trying to install and run the code. It is also one of the limitations. The error messages can be troubleshooting by finding the alternative codes over the internet and while installing the error messages can be rectified by installing some dependencies.
- ROS is not really the best for tasks which are totally mission-critical, totally needs to be scalable, secure or need to run on some other operating systems other than Ubuntu which is fairly common.

Dynamixel Motor:

The constraints of Dynamixel motors are

- Upon continuous rotation the motors get heated.
- The cost of the motor is higher.

5.4 Future works

There are some ideas to advance into:

- This system may be built for use in the industrial application for pick and place tasks.
- The Bioloid Robot arm is light in weight and can provide reasonable accuracy in handling objects.
- This arm can be taken to a higher level by providing a voice controller thereby implementing it using voice command.
- This system uses Linux-based commands. The same system can be implemented using an Android application over which the full system can be controlled using the internet anywhere from the world.
- A camera module can be used to detect the presence of the object in the environment.
- The system can be implemented by interfacing the sensor and motor together using ROS.
- The system can be implemented using Artificial Intelligence in the future as this may provide intelligent behaviour to the robot in unpredictable and changing environments.

Reference List

- [1] Robotics and Automation Society, Ed., 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012): Osaka, Japan, 29 November - 1 December 2012. Piscataway, NJ: IEEE, 2012.
- [2] Chung-Hsien Kuo, Yu-Wei Lai, Kuo-Wei Chiu, and Shih-Tseng Lee, "Motion planning and control of interactive humanoid robotic arms," 2008, pp. 1–6.
- [3] Masayuki Shamoto, Kunihito Kato, and Kazuhiko Yamamoto, "A implementation of Humanoid Vision -analysis of eye movement and implementation to robot-," 2007, pp. 744–747.
- [4] T. Tremblay and T. Padir, "Modular Robot Arm Design for Physical Human-Robot Interaction," 2013, pp. 4482–4487.
- [5] "Robotics/Types of Robots/Arms - Wikibooks, open books for an open world." [Online]. Available: https://en.wikibooks.org/wiki/Robotics/Types_of_Robots/Arms. [Accessed: 03-Oct-2016].
- [6] J. M. O’Kane, A gentle introduction to ROS. Leipzig: Amazon, 2014.
- [7] P. Tavares and A. Sousa, "Flexible pick and place architecture using ROS framework," 2015, pp. 1–6.
- [8] J. Rozman, R. Luza, and F. V. Zboril, "ROS-based remote controlled robotic arm workcell," 2014, pp. 101–106.
- [9] "Documentation - ROS Wiki." [Online]. Available: <http://wiki.ros.org/>. [Accessed: 03-Oct-2016].
- [10] "BeagleBoard.org - black." [Online]. Available: <https://beagleboard.org/black>. [Accessed: 03-Oct-2016].
- [11] I. Zubrycki and G. Granosik, "Test setup for multi-finger gripper control based on robot operating system (ROS)," 2013, pp. 135–140.
- [12] P. S. Lengare and M. E. Rane, "Human hand tracking using MATLAB to control Arduino based robotic arm," 2015, pp. 1–4.
- [13] L. Joseph, Learning robotics using Python: design, simulate, program, and prototype an interactive autonomous mobile robot from scratch with the help of Python, ROS, and Open-CV! Birmingham: Packt Publishing open source, 2015.
- [14] "wiki.ros.org." .
- [15] Y. Lin, Shaogang Ren, M. Clevenger, and Y. Sun, "Learning grasping force from demonstration," 2012, pp. 1526–1531.
- [16] K. Li and M. Meng, "Multilevel part-based model for object manipulation," 2013, pp. 1114–1119.
- [17] tutorialspoint.com, "Python Basic Syntax," [www.tutorialspoint.com](http://www.tutorialspoint.com/python/python_basic_syntax.htm). [Online]. Available: http://www.tutorialspoint.com/python/python_basic_syntax.htm. [Accessed: 03-Oct-2016].
- [18] F. Husain, A. Colome, B. Dellen, G. Alenya, and C. Torras, "Realtime tracking and grasping of a moving object from range video," 2014, pp. 2617–2622.
- [19] "2.1. Gaussian mixture models — scikit-learn 0.18.1 documentation." [Online]. Available: <http://scikit-learn.org/stable/modules/mixture.html>. [Accessed: 09-Dec-2016].
- [20] "Dynamixel AX-12A Robot Actuator from Robotis." [Online]. Available: <http://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx>. [Accessed: 03-Oct-2016].
- [21] "Robots for Beginners." [Online]. Available: http://support.robotis.com/en/product/bioloid/beginnerkit/download/bioloid_beginner_app.htm. [Accessed: 03-Oct-2016].

- [22] D. F. Vera, D. M. Cadena, and J. M. Ramirez, "Iris recognition algorithm on BeagleBone Black," 2015, pp. 282–286.
- [23] "USB2Dynamixel." [Online]. Available: http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.htm. [Accessed: 03-Oct-2016].
- [24] "What are the kinematics for a 3DOF robot arm? - Quora." [Online]. Available: <https://www.quora.com/What-are-the-kinematics-for-a-3DOF-robot-arm>. [Accessed: 03-Oct-2016].
- [25] "Genetic algorithm," Wikipedia. 19-Nov-2016.
- [26] "Introduction to Genetic Algorithms." [Online]. Available: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html. [Accessed: 08-Dec-2016].
- [27] "Genetic Algorithms Overview." [Online]. Available: <http://geneticalgorithms.ai-depot.com/Tutorial/Overview.html>. [Accessed: 08-Dec-2016].
- [28] "FAQ: comp.ai.genetic part 2/6 (A Guide to Frequently Asked Questions) - Q1.1: What's a Genetic Algorithm (GA)?" [Online]. Available: <https://www.cs.cmu.edu/Groups/AI/html/faqs/ai/genetic/part2/faq-doc-2.html>. [Accessed: 08-Dec-2016].
- [29] J. Brownlee, "A Tour of Machine Learning Algorithms," Machine Learning Mastery, 25-Nov-2013. .
- [30] "Machine learning," Wikipedia. 05-Dec-2016.
- [31] "List of machine learning concepts," Wikipedia. 13-Jun-2016.
- [32] "Essentials of Machine Learning Algorithms (with Python and R Codes)," Analytics Vidhya, 10-Aug-2015. .
- [33] "An Introduction to Neural Networks." [Online]. Available: <http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html>. [Accessed: 08-Dec-2016].
- [34] "Artificial neural network," Wikipedia. 28-Nov-2016.
- [35] A. R. F. Quiros, A. C. Abad, and E. P. Dadios, "Object locator and collector robotic arm using artificial neural networks," 2015, pp. 1–6.
- [36] M. A. Nielsen, "Neural Networks and Deep Learning," 2015.
- [37] "Neural Networks." [Online]. Available: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html. [Accessed: 08-Dec-2016].
- [38] [Online]. Available: http://www-bcf.usc.edu/~ioannou/RobustAdaptiveBook95pdf/Robust_Adaptive_Control.pdf. [Accessed: 08-Dec-2016].
- [39] [Online]. Available: <http://www.landau-adaptivecontrol.org/Slides%20Ch1.pdf>. [Accessed: 08-Dec-2016].
- [40] "Adaptive control," Wikipedia. 14-Nov-2016.
- [41] "cmumrsdproject - Adaptive Monte Carlo Localization (Details of Theory and ROS Package)." [Online]. Available: [http://cmumrsdproject.wikispaces.com/Adaptive+Monte+Carlo+Localization+\(Details+of+Theory+and+ROS+Package\)](http://cmumrsdproject.wikispaces.com/Adaptive+Monte+Carlo+Localization+(Details+of+Theory+and+ROS+Package)). [Accessed: 29-Nov-2016].
- [42] "Reinforcement learning," Wikipedia. 07-Dec-2016.
- [43] "Reinforcement Learning - Algorithms." [Online]. Available: <http://www.cse.unsw.edu.au/~cs9417ml/RL1/algorithms.html>. [Accessed: 08-Dec-2016].
- [44] "Reinforcement Learning Introduction." [Online]. Available: <http://reinforcementlearning.ai-depot.com/>. [Accessed: 08-Dec-2016].
- [45] [Online]. Available: http://www.cs.columbia.edu/~kathy/cs4701/documents/jason_svm_tutorial.pdf. [Accessed: 08-Dec-2016].

- [46] "Introduction to Support Vector Machines — OpenCV 2.4.13.1 documentation." [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html. [Accessed: 08-Dec-2016].
- [47] "Support vector machine," Wikipedia. 30-Nov-2016.
- [48] "Understanding Support Vector Machine algorithm from examples (along with code)," Analytics Vidhya, 06-Oct-2015. .
- [49] "Evolutionary algorithm," Wikipedia. 05-Dec-2016.
- [50] "Evolutionary Algorithms 1 Introduction." [Online]. Available: <http://www.geatbx.com/docu/algindex.html>. [Accessed: 08-Dec-2016].
- [51] "Evolutionary computation," Wikipedia. 03-Dec-2016.
- [52] "Chapter 1. The Power of Evolution." [Online]. Available: <http://watchmaker.uncommons.org/manual/ch01.html>. [Accessed: 08-Dec-2016].
- [53] Y. Pititeeraphab and M. Sangworasil, "Design and construction of system to control the movement of the robot arm," 2015, pp. 1–4.
- [54] "Overview | Controlling a Servo with a BeagleBone Black | Adafruit Learning System." [Online]. Available: <https://learn.adafruit.com/controlling-a-servo-with-a-beaglebone-black/overview>. [Accessed: 03-Oct-2016].
- [55] A. Nayyar and V. Puri, "A Review of Beaglebone Smart Board's-A Linux/Android Powered Low Cost Development Platform Based on ARM Technology," 2015, pp. 55–63.
- [56] "Bootstrapping the BeagleBone Black with Debian." [Online]. Available: <http://thethingsystem.com/dev/Bootstrapping-the-BeagleBone-Black-with-Debian.html>. [Accessed: 03-Oct-2016].
- [57] [Online]. Available: https://courses.cs.washington.edu/courses/cse466/11au/calendar/ros_cc_1_intro-jrsedit.pdf. [Accessed: 03-Oct-2016].
- [58] "TPS65217 | Power Management Multi-Channel IC (PMIC) Solutions | Power Management | Description & parametrics." [Online]. Available: <http://www.ti.com/product/TPS65217>. [Accessed: 03-Oct-2016].
- [59] the_admin, "i2c communication: Beaglebone Black and MPU6050 (acc+gyro) | intuition >> data >> analysis." .
- [60] "Programming the BeagleBone Black with Python | Random Nerd Tutorials." .
- [61] "PWM | Setting up IO Python Library on BeagleBone Black | Adafruit Learning System." [Online]. Available: <https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/pwm>. [Accessed: 03-Oct-2016].
- [62] "Using ultrasonic Sensor with Arduino and Stm32." .
- [63] A. R. Chaidir, A. B. Satriya, and G. Dias Kalandro, "Design of a gripping imitator robotic arm for taking an object," 2016, pp. 1–5.
- [64] "dynamixel_motor - ROS Wiki." [Online]. Available: http://wiki.ros.org/dynamixel_motor. [Accessed: 03-Oct-2016].
- [65] "CM-530." [Online]. Available: <http://support.robotis.com/en/product/auxdevice/controller/cm530.htm>. [Accessed: 03-Oct-2016].
- [66] "Robotic Arm Controller Bachelor of Engineering (Honours) Sustainable Design Engineering Year 4, Semester 7 2012/13 Sustainable Design Practice 2 Robotic Arm Controller Robotic Arm Controller Table of Contents." [Online]. Available: https://www.academia.edu/7296388/Robotic_Arm_Controller_Bachelor_of_Engineering_Honours_Sustainable_Design_Engineering_Year_4_Semester_7_2012_13_Sustainable_Design_Practice_2_Robotic_Arm_Controller_Robotic_Arm_Controller_Table_of_Contents. [Accessed: 03-Oct-2016].

- [67] H. Wei, Z. Huang, Q. Yu, M. Liu, Y. Guan, and J. Tan, "RGMP-ROS: A real-time ROS architecture of hybrid RTOS and GPOS on multi-core processor," 2014, pp. 2482–2487.
- [68] M. Quigley, B. Gerkey, and W. D. Smart, Programming robots with ROS, First edition. Sebastopol: O'Reilly & Associates Incorporated, 2015.
- [69] "ROS 101: Intro to the Robot Operating System | Robohub." .
- [70] E. Fernandez, Learning ROS for robotics programming: your one-stop guide to the Robot Operating System. 2015.
- [71] J. Lentin, Mastering ROS for robotics programming: design, build, and simulate complex robots using Robot Operating System and master its out-of-the-box functionalities. Birmingham Mumbai: Packt Publishing, 2015.
- [72] "Robotic Mapping: Simultaneous Localization and Mapping (SLAM)," GIS Lounge, 15-Jan-2013. [Online]. Available: <https://www.gislounge.com/robotic-mapping-simultaneous-localization-and-mapping-slam/>. [Accessed: 29-Nov-2016].
- [73] "Marginal Particle and Multirobot Slam: SLAM='SIMULTANEOUS LOCALIZATION AND MAPPING' By Marc Sobel (Includes references to Brian Clipp Comp Robotics) - ppt download." [Online]. Available: <http://slideplayer.com/slide/4546667/>. [Accessed: 29-Nov-2016].
- [74] denna, "SLAM: Simultaneous Localization and Mapping: Part I," SlideServe, 08-Sep-2012. [Online]. Available: <http://www.slideserve.com/denna/slam-simultaneous-localization-and-mapping-part-i>. [Accessed: 29-Nov-2016].
- [75] "navigation/Tutorials/RobotSetup." [Online]. Available: [http://library.isr.ist.utl.pt/docs/ros/wiki/navigation\(2f\)Tutorials\(2f\)RobotSetup.html](http://library.isr.ist.utl.pt/docs/ros/wiki/navigation(2f)Tutorials(2f)RobotSetup.html). [Accessed: 29-Nov-2016].
- [76] "Bootstrapping the BeagleBone Black with Debian." [Online]. Available: <http://thethingsystem.com/dev/Bootstrapping-the-BeagleBone-Black-with-Debian.html>. [Accessed: 03-Oct-2016].
- [77] R. Khurshid, N. Fitter, E. Fedalei, and K. Kuchenbecker, "Effects of Grip-Force, Contact, and Acceleration Feedback on a Teleoperated Pick-and-Place Task," IEEE Trans. Haptics, pp. 1–1, 2016.
- [78] S. Rosa, A. Russo, A. Saglinbeni, and G. Toscana, "Vocal interaction with a 7-DOF robotic arm for object detection, learning and grasping," 2016, pp. 505–506.
- [79] Robotics and Automation Society and Engineering in Medicine and Biology Society, Eds., 2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechanics (BioRob 2012): Rome, Italy, 24 - 27 June 2012 ; [including the first edition of the Symposium on Surgical Robotics ; part of the Bioengineering Week (June 20 - 29, 2012)]. Piscataway, NJ: IEEE, 2012.
- [80] I. Siradjuddin, S. P. Tundung, A. S. Indah, and S. Adhisuwignjo, "A real-time Model Based Visual Servoing application for a differential drive mobile robot using Beaglebone Black embedded system," 2015, pp. 186–192.
- [81] "dinuxbg/pru-gcc-examples," GitHub. [Online]. Available: <https://github.com/dinuxbg/pru-gcc-examples>. [Accessed: 03-Oct-2016].
- [82] M. Karimi, A. Ahmadi, N. Khazaee Korghond, E. Babaians, and S. S. Ghidary, "ReMoRo; A mobile robot platform based on distributed I/O modules for research and education," 2015, pp. 657–662.
- [83] L. Pauly, M. V. Baiju, P. Viswanathan, P. Jose, D. Paul, and D. Sankar, "CAMbot: Customer assistance mobile manipulator robot," 2015, pp. 1–4.
- [84] S. Pradeep, A. Rani, V. Singh, and S. Agarwal, "An Approach for Object Picking using Correlation analysis and Robotic Manipulation," 2015, pp. 1–6.

Appendix A

Specifications of the devices used.

Table A.1 Specifications of Beaglebone Black

Features	
Processor	AM335x ARM cortex A8 chip
Video out	HDMI
DRAM	512MB DDR3
On-board Flash	4GB Embedded MMC
Power source	Mini USB or 5VDC External Jack
Indicator	1-power, 2-Ethernet, 4-user controllable LEDs.
Ethernet	10/100, RJ45
SD/MMC connector	micro SD, 3.3V
User Input	Reset button Boot button Power button
Video out	16b HDMI, 1280x1024 (max) 1024x768, 1280x720, 1440x900, 1920x1080@24hz w/EDID support
Audio	Via HDMI interface, stereo
Weight	1.4 Oz (39.68gms)

Table A.2 Specifications of Ultrasonic Sensor

Power supply	+5V DC
Quiescent current	<2mA
Working current	15mA
Effectual angle	<15°
Range	2cm – 400cm

Resolution	0.3cm
Measuring angle	30 degree
Trigger input pulse width	10uS

Table A.3 Specifications of Dynamixel Ax-12A Motor

Specifications	
Weight	54.6gms
Dimension	32mm*50mm*40mm
Resolution	0.29°
Gear reduction ratio	254:1
Stall torque	1.5Nm (at 12.0V, 1.5A)
No load speed	59rpm (at 12V)
Running degree	0°~300°
Voltage	9~12V
Command signal	Digital packet
Protocol type	Half duplex Asynchronous serial communication
ID	254 (0~253)
Communication speed	7343bps ~ 1mbps
Feedback	Position, temperature, load, input voltage etc.

Table A.4 Specifications of CM-530 Controller

Specifications	
Weight	54gms
CPU	ARM Cortex STM32F103RE
Operating Voltage	6V to 15V
Current consumption	Idle: 50mA External I/O max current: 0.3A Overall max current: 10A
Operating Temperature	-50° to 70°C
Internal I/O Device	Button: 6 (Reset 1;port 5) MIC: 1 Voltage Sensor: 1

External I/O device	OLLO Compatible I/O 5pin port: 6 AX/MX series 3pin connector: 5
Compatible products	Gyro sensor, DMS sensor, IR Sensor, Touch Sensor, Servo motors etc.