`

DESIGN & FABRICATION OF SEMI AUTOMATIC SELF TRANSFORMING MOBILE ROBOT

A PROJECT REPORT

Submitted by

E. Ajith Kumar	513117114004
R. Arul Manikandan	513117114007
S. Dhivyaswaroop	513117114015
P. Gowtham Raj	513117114025

In partial fulfilment for the award of the degree

of

DEPARTMENT OF MECHANICAL ENGINEERING

THANTHAI PERIYAR GOVERNMENT INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025

ANNA UNIVERSITY, CHENNAI-600 025

BONAFIDE CERTIFICATE

Certified that this project "**DESIGN & FABRICATION OF SEMI AUTOMATIC SELF TRANSFORMING MOBILE ROBOT**"

BATCH STUDENTS:

E. Ajith Kumar	513117114004
R. Arul Manikandan	513117114007
S. Dhivyaswaroop	513117114015
P. Gowtham Raj	513117114025

Who carried out the project work under my supervision

SIGNATURE

DR. P. PRAVEEN RAJ, M.E., Ph.D,
HEAD OF THE DEPARTMENT
Department of Mechanical Engineering,
Thanthai Periyar Govt Inst
Of Technology, Vellore-632002.

SIGNATURE

Dr. T.SUJA, M.E., Ph.D,
ASSIST. PROFESSOR PROJECT GUIDE
Department of Mechanical Engineering,
Thanthai Periyar Govt inst
Of Technology, Vellore-632002.

Submitted for project viva voice examination held on……

INTERNAL EXAMINER

EXTERNAL EXAMINER

`

ACKNOWLEDGEMENT

We thank our beloved Principal **Dr. RAHILA BILAL, M.E, Ph.D,** who always served as source of inspiration and encouraging us throughout our project.

We express our heartfelt thanks to our beloved head of the Department **Dr. P.PRAVEEN RAJ, M.E., Ph.D.,** who has contributed by giving valuable suggestion in doing this project.

We express our sincere thanks to **Dr. T.SUJA, M.E., Ph.D.,** our project guide who gave faithful ideas and constantly guided us to finish this project.

We would also thank our faculty advisors **Dr. T.SUJA, M.E, Ph.D.,** and **Prof. K.BARATHI, M.E.,** for giving constant support in doing this project.

We also like to thank all our staff members of our department for the kind suggestion during this project work. We express our special thanks to our parents and friends who were extremely supporting for successful completion of the project**.**

`

Abstract

Self-transforming robot is a robot which transforms its shape according to the hindrance occurring in the path where the robots are being moved. Such robots have been recognized as very attractive design in exhibiting the reliable transformation according to the situations. Military and defense application needs a robot should possess arbitrary movements like human. In some scenarios transformations are made by biological inspired control strategies using Central Pattern Generators (CPG). CPG is used in the locomotion control of snake robots, quadruped robots, to humanoid robots. This paper presents a Self-transforming robot which possess alteration in its original shape to exhibit a human-like behavior while passing over the particular location. Quadrupedal locomotion on rough terrain and unpredictable environments is still a challenge, where the proposed system will provide the good adaptability in rough terrain. It allows the modulation of locomotion by simple control signal. The necessary conditions for the stable dynamic walking on irregular terrain in common are proposed.

Keywords**: self-transforming robot, central pattern generators (CPG), Quadruped robot.**

`

Contents

`

List of figures

`

List of tables

`

List of Abbrevations

Abbrevation	Expansion
VCC	Voltage common collector
GND	Ground
RxD	Receive Data
TxD	Transmit Data
SCL	Serial clock
SDA	Serial data
LiPo	Lithium Polymer
Vin	Voltage Input
DC	Direct current
MIPS	Millions Instructions Per Second
IC	Integrated Chip
RAM	Random Access Memory
ROM	Read Only Memory
PWM	Pulse Width Modulation
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
Amp	Ampere
mA	Milliampere

`

CHAPTER 1

1. Introduction

The field of robotics is basically divided into two basic categories: wheeled robots and legged robots. Wheeled robots are robots that navigate around the work envelope using motorized wheels to propel them. The design of these robots is simpler than using treads or legs and by using wheels they are easier to design, build, and program for movement in flat environment.

They are also more well-controlled and stable as compared to other types of robots. Legged robots are further divided into one legged, two legged, three legged, four legged and six legged robots. Most successful legged robots have 4 or 6 legs for further stability in movement. Four legged robots are also called quadruped robots. They are preferred on other legged robots due to the factor that they have comparatively a stable gait as well as they are lighter in weight and less expensive than six legged robots.

Robots are usually designed to perform one task very well, whether it's assembling parts in a factory or vacuuming the living room. But ask those robots to perform another task or even the same task in a new environment, and you're asking for trouble.

Self-transforming robots, on the other hand, can reshape themselves as their task or environment changes, ideally without human intervention to take the appearance or form of another object. The robot can morph to resemble everyday objects, machines, or animals, and vice versa.

`

1.1. Robotics

Robotics is the branch of technology that deals with the design, construction, operation, and application of robots. It also includes computer systems for their control, sensory feedback, and information processing.

These technologies deal with automated machines that can perform the duties of humans in dangerous environments or manufacturing processes, or resemble humans in appearance, behavior, and cognition.

Nowadays many robots are inspired by nature and are a part of bio-inspired robotics. Designing, building, programming and testing robots is a combination of physics, mechanical engineering, electrical engineering, mechatronics engineering, structural engineering, mathematics and computing. A new evolving study of legged robots is also a part of robotics. In addition to designing and construction, robotics is used to formulate the stable gaits of these robots.

1.2. Identification of problem

Our goal is to build a four legged robot that can change its shape according to its environment. By default it walks on four legs, in case when an obstacle is fronted by the robot. It changes its shape. It changes its shape and transforms into a car like structure. On wheels it moves beneath the obstacle. On a rough surface it spreads its legs and crawls by again shifting its shape. These three modes (walking, crawling and car) are useful for overcoming an obstacle and changing its shape according to its environment.

`

CHAPTER 2

2. Mechanical design and components

Design considerations are an important factor when compiling a design because they facilitate an informed decision on the limitations, operating conditions and capabilities of the final product. This process may also highlight areas for further investigation that could improve, simplify or make a product more cost effective. This includes commercial aspects that may be undertaken if the product was to become mass produced or if an alternate material or component usage was to be implemented to reduce the potential manufacturing costs.

Design considerations for the self-transforming robot encompassed operating environments, motors, wheels, sensors, microcontroller, power source, safety features and various forms of control. All these considerations were then sub- investigated to determine dimensions, configuration, maintenance requirements, availability, efficiency, etc. Consideration of these factors combine together to guarantee the necessary capability is achieved successfully within the timeframe and resources available.

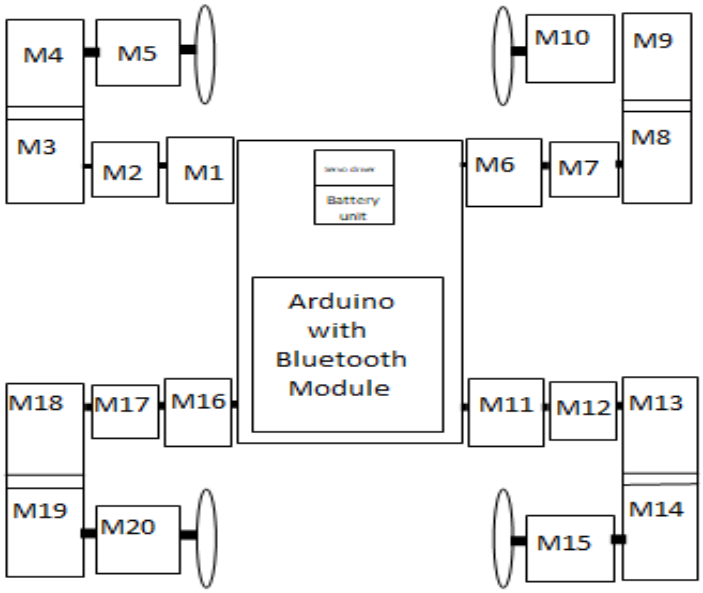

Fig 2.1 Design concept

`

2.1. Materials selection

The materials used in a design limit the durability, strength, maintainability, energy efficiency and operating capability of the product. They also impact on the size and contribute to the total weight of the final design. A heavier material such a metal is typically stronger then lighter materials such as plastics but it requires more energy to move. The size of the design impacts on its ability to navigate around, through or over obstacles as well as its transportability.

Material we choose for our Robot is Aluminum and sheet metal. Sheet metal is used in the base and in the upper most portion of the robot. Sheet metal is light, high-quality, and efficient. It is also very affordable.

Aluminum is used due the following properties:

- Light Weight
- Corrosion Resistance

2.2. Servo motors

Servo motors (or servos) are self-contained electric devices that rotate or push parts of a machine with great precision. Servos are found in many places: from toys to home electronics to cars and airplanes. If you have a radio- controlled model car, airplane, or helicopter, you are using at least a few servos. In a model car or aircraft, servos move levers back and forth to control steering or adjust wing surfaces. By rotating a shaft connected to the engine throttle, a servo regulates the speed of a fuel-powered car or aircraft.

Electronic devices such as DVD players use servos to extend or retract the disc trays. In 21st-century automobiles, servos manage the car's speed Commercial aircraft use servos and a related hydraulic technology to push and pull just about everything in the plane.The heart of a servo is a small direct current (DC) motor.

`

Fig 2.2 Servo motor feedback sensor

These motors run on electricity from a battery and spin at high RPM (rotations per minute) but put out very low torque. An arrangement of gears takes the high speed of the motor and slows it down while at the same time increasing the torque. A tiny electric motor does not have much torque, but it can spin really fast. The gear design inside the servo case converts the output to a much slower rotation speed but with more torque.

The amount of actual work is the same, just more useful. Gears in an inexpensive servo motor are generally made of plastic to keep it lighter and less costly. On a servo designed to provide more torque for heavier work, the gears are made of metal and are harder to damage.

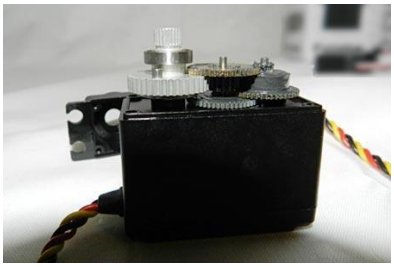

Fig 2.3 Servo motor gear setup

With a small DC motor, you apply power from a battery, and the motor spins. Unlike a simple DC motor, however, a servo's spinning motor shaft is slowed way down.with gears .A positional sensor on the final gear is connected to a small circuit board. The sensor tells this circuit board how far the servo output shaft has rotated. The electronic input signal from the computer or the radio in a remote-controlled vehicle also feeds into that circuit board. The electronics on the circuit board decode the signals to determine how far the user wants the servo to rotate. It then compares the desired position to the actual position and decides which direction to rotate the shaft so it gets to the desired position.

`

2.2.1. Types of servo motors

Servos come in many sizes and in three basic types: positional rotation, continuous rotation, and linear.

- **Positional rotation servo**: This is the most common type of servo motor. The output shaft rotates in about half of a circle, or 180 degrees. It has physical stops placed in the gear mechanism to prevent turning beyond these limits to protect the rotational sensor. These common servos are found in radio- controlled cars and water- and aircraft, toys, robots.
- **Continuous rotation servo**: This is quite similar to the common positional rotation servo motor, except it can turn in either direction indefinitely. The control signal, rather than setting the static position of the servo, is interpreted as the direction and speed of rotation. The range of possible commands causes the servo to rotate clockwise or counterclockwise as desired, at varying speed, depending on the command signal.
- **Linear servo**: This is also like the positional rotation servo motor described above, but with additional gears (usually a rack and pinion mechanism) to change the output from circular to back-and-forth. These servos are not easy to find, but you can sometimes find them at hobby stores where they are used as actuators in larger model airplanes.

2.2.2. Selecting a servo motor

When starting a project that uses servos, look at your application requirements. How fast must the servo rotate from one position to another? How hard will it have to push or pull? Do I need a positional rotation, continuous rotation, or linear servo? How much overshoot is allowable? The less you pay for the servo, the less mechanical power it will have to muster and the less precision it will have in its movements.

`

2.3. Servo holding brackets

In order to develop the robot, the motor must be held tightly. For this servo brackets are use to tightly hold the servo brackets in its position.

2.3.1. U shaped brackets

Fig 2.4 U-Bracket

Fig 2.5 U-Bracket side view

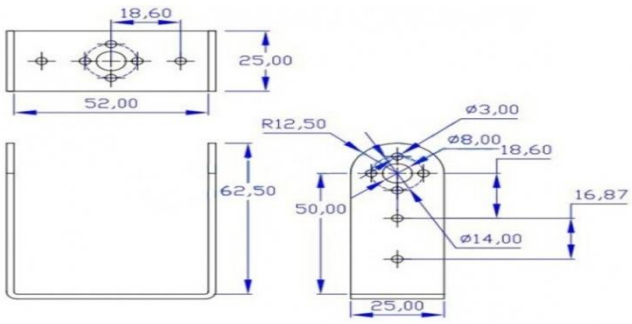

Fig 2.6 U-Bracket sheet diagram

`

2.3.2. Interconnect robot servo bracket aluminium

This Slotted Servo bracket straight board Mount designed to fit most standard sized servos. This bracket is both lightweight and strong. Various holes are located on the bracket that fit most servo horns.

Fig 2.7 Inter connector

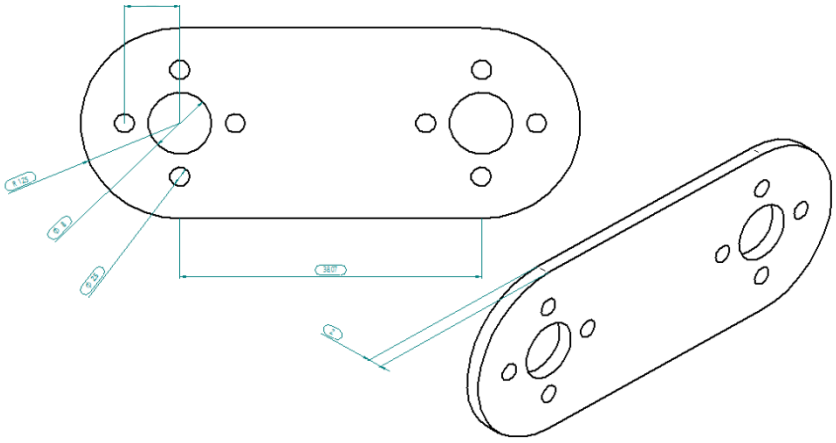

Fig 2.8 Inter connector sheet diagram

`

2.3.3. M brackets

The other bracket which was designed is the M bracket which is shown below.

Fig 2.9 M-Bracket

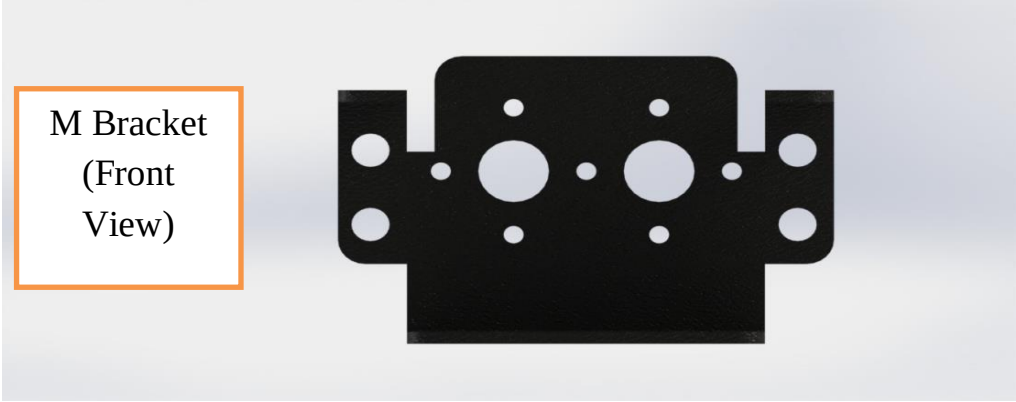

Fig 2.10 M-Bracket front view

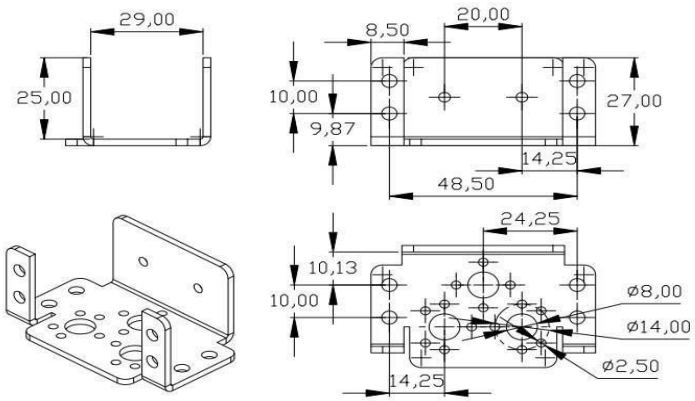

Fig 2.11 M-Bracket sheet diagram

`

2.4. Wheels

The material used for making wheel is wood. It is cheaper than other material and eco-friendly. Rubber gripper is mounted on the wooden wheel to make car mode stable.

2.4.1. Benefits of using wood

- Renewable, Recyclable, Natural.
- Carbon Positive.
- Eco-friendly
- Natural Insulation

Fig 2.12 wheel front view

Fig 2.13 wheel

`

CHAPTER 3

3. Electronics hardware & circuitry

This chapter covers all the electronics hardware that has been used in the project including Microcontroller, Sensors and Power circuitry.

3.1. Microcontroller

The purpose of a microcontroller is to complete computations and process data by executing instructions or programming. They contain all the necessary components expected of a computer system in a single Integrated Chip (IC). These include the processor itself, RAM, ROM and input/outputs. Since they are small in size, they are typically implemented in embedded systems and can be found in numerous household appliances. Ideally, a microcontroller will maintain a fast response with minimal power consumption and low susceptibility to interference. The IC should contain all the devices necessary for operational requirements such as timers, ADC, DAC, and PWM outputs. This reduces the need for additional peripherals and hardware to be incorporated which would only increase the complexity required and the power requirements needed. The microcontroller should also be reprogrammable to allow improved operation and upgrades over it intended lifespan and beyond. The choice of microcontroller was difficult due to the large range and variations available. A 24-bit wide instruction microcontroller was favored over the common 8-bit counterparts due to the extra inherent capabilities. The IC chips are very common and easily available from the manufacturer Microchip, hence they are the preferred supplier. It was also noted that the manufacturer has incorporated lead-free product packaging for its products identifying an environmentally friendly product. The idea of a redundant microcontroller was contemplated but overlooked due to the nature of this robots design. There are 20, 30 and 40 Million Instructions per Second (MIPS) versions with the high performance controller.

`

3.2. Arduino board

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's Intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments. Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software running on a computer (e.g. Flash, Processing.Arduino boards can be purchased pre-assembled or as do-it-yourself kits. Hardware design information is available for those who would like to assemble an Arduino by hand. It was estimated in mid-2011 that over 300,000 officials Arduino had been commercially produced at that point.

3.2.1. Overview

Arduino Uno is a microcontroller board based on 8-bit ATmega328P microcontroller. Along with ATmega328P, it consists other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. Arduino Uno has 14 digital input/output pins (out of which 6 can be used as PWM outputs), 6 analog input pins, a USB connection, A Power barrel jack, an ICSP header and a reset button.

Fig 3.1 Arduino

`

3.2.2. Power

The Arduino Uno board can be powered via a USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector. The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- Vin. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power sources).
- 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. 3V3. A 3.3 volt supply generated by the on-board regulator. The maximum current draw is 50 mA.
- GND. Ground pins.
- IOREF. This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates.

3.2.3. Memory

The ATmega328P has 32 KB of flash memory for storing code, 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

`

3.2.4. Input and output

Each of the 14 digital pins on the arduino uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms.

3.3. Power source

A power source is essential for providing energy to a machine. Without energy, a machine would simply not operate. If an insufficient power source was implemented, a machine would not function correctly nor would it perform adequately. Power sources may be AC through electrical main supplies or generators. They may also be DC from devices such as batteries. Batteries are rated by voltage output and ampere hours (AH). AH refers to the current that they can supply per hour.

Batteries will be an essential component in allowing the robot to maintain autonomous operation. Otherwise a direct connection between the robot and a power source would need to be maintained at all times. Ideally, the best practice is to minimize the total power consumption during this design phase so a smaller power system may be implemented. This is an important factor as power systems are typically the heaviest element of a robot or device.

3.4. Lipo battery

The Sky Cell Lithium-Polymer(Li-Po) Battery for robotic applications. It can give great instantaneous discharge current upto 83A. Very light weight and small size compared to Ni-Cd, Ni-MH and Lead acid batteries.Very long life without losing charging capacity.Weights just 150 gm.

`

3.4.1. Features

- Very small in size and weight compared to Ni-Cd, Ni-MH and Lead Acid Batteries
- Full Charge in 180 minutes with special charger
- Long life with full capacity for upto 1000 charge cycles
- 2X Li-Po 3.7V 3300mAh cells (2S1P)
- Low maintenance
- Discharge Current: 25*3300maH = 83Amp
- Max Charging Current: 1A
- Dimensions: 130mm x 44mm x 14mm
- Weight 150gm

Fig 3.2 lipo battery

3.5. Hc-05 bluetooth module

HC-05 is a Bluetooth module which is designed for wireless communication. This module can be used in a master or slave configuration

HC-05 module has two modes,

1. **Data mode:** Exchange of data between devices.

2. **Command mode:** It uses AT commands which are used to change setting of HC-05. To send these commands to module serial (USART) port is used.

`

3.5.1. Pin description

Fig 3.3 Bluetooth module pin

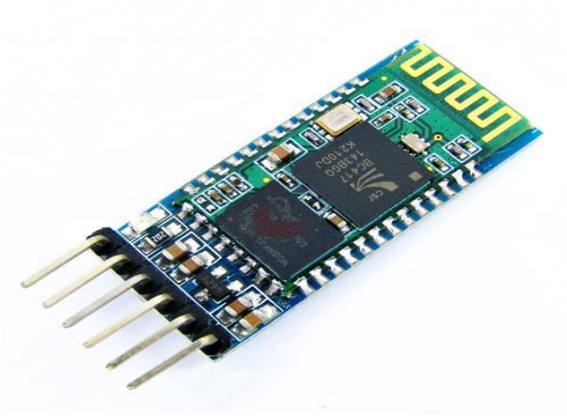

Fig 3.4 Bluetooth module

It has 6 pins,

1. **Key/EN:** It is used to bring Bluetooth module in AT commands mode. If Key/EN pin is set to high, then this module will work in command mode. Otherwise by default it is in data mode. The default baud rate of HC-05 in command mode is 38400bps and 9600 in data mode.

2. **VCC:** Connect 5 V or 3.3 V to this Pin.

3. **GND:** Ground Pin of module.

4. **TXD:** Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin)

5. **RXD:** Receive data serially (received data will be transmitted wirelessly by Bluetooth module).

6. **State:** It tells whether module is connected or not.

`

3.6. Servo driver

Driving servo motors with the Arduino Servo library is pretty easy, but each one consumes a precious pin - not to mention some Arduino processing power. The Adafruit 16-Channel 12-bit PWM/Servo Driver will drive up to 16 servos over I2C with only 2 pins. The on-board PWM controller will drive all 16 channels simultaneously with no additional Arduino processing overhead. What's more, you can chain up to 62 of them to control up to 992 servos - all with the same 2 pins

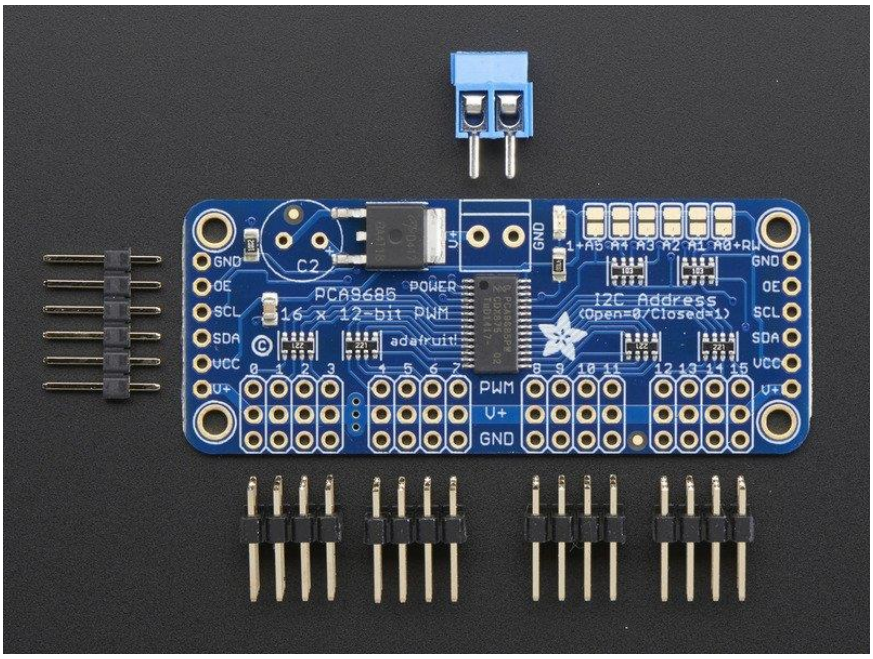

Fig 3.5 Servo motor driver

3.6.1. Power pins

- **GND** - This is the power and signal ground pin, must be connected
- **VCC** - This is the **logic** power pin, connect this to the logic level you want to use for the PCA9685 output, should be 3 - 5V max!
- **V+** - This is an optional power pin that will supply distributed power to the servos. If you are not using for servos you can leave disconnected. It is not used at all by the chip.

`

- You can also inject power from the 2-pin terminal block at the top of the board. You should provide 5-6VDC if you are using servos. If you have to, you can go higher to 12VDC, but if you mess up and connect VCC to V+ you could damage your board!

3.6.2. Control pins

- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line. Can use 3V or 5V logic, and has a weak pull up to **VCC**
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line. Can use 3V or 5V logic, and has a weak pull up to **VCC**
- **OE** - Output enable. Can be used to quickly disable all outputs. When this pin is low all pins are enabled. When the pin is high the outputs are disabled. Pulled low by default so it's an optional pin!
-

3.6.3. Output ports

There are 16 output ports. Each port has 3 pins: V+, GND and the PWM output. Each PWM runs completely independently but they must all have the same PWM frequency. They're set up for servos but you can use them for LEDs! Max current per pin is 25mA.

3.7. Regulators

The module contains two large aluminum heat sinks ensures a better heat dissipation and more stable performance. And it contains five low resistance capacitors of 470uF/35V in parallel specially for the switch mode power supply, a 23mm large magnet and 1.0 copper wire provides high efficiency work. For safe module operation this module contains a TVs Diode to prevent voltage surge from breaking the chip. This module is widely used for storage battery, power transformers, DIY adjustable regulated power supply, industrial equipment, 12V to 3.3V, 12V to 5V, 24V to 5V, 24V to 12V, etc.

`

3.7.1. Specifications

- Input voltage: 5 -40 V (without constant current 5-32V)
- Output current: maximum 12A Constant current range: 0.2-12A (adjustable) (no such function without constant current)
- Output power: The maximum power is about 300W.
- Dimension:6.5cm x 4.8cm x 2.4cm

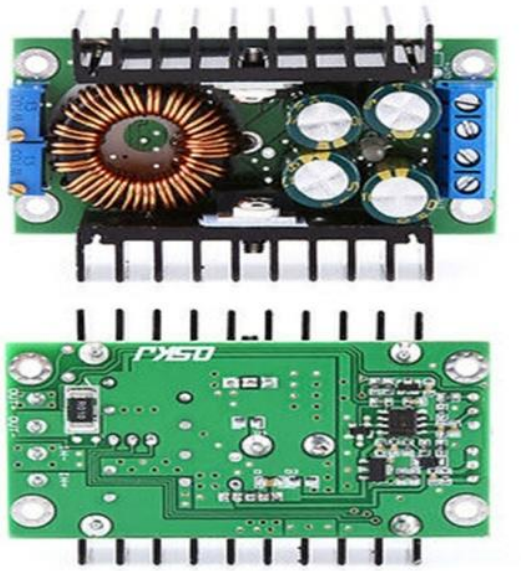

Fig 3.6 Dc-dc regulator

3.8. Imax b3 ac compact balance charger for 2s-3s lipo battery

B3AC Compact Charger is a simple and compact LiPoly balance charger for 2~3s batteries. It features built-in JST-XH balance plug ports and 3 LEDs to indicate charge status. B3AC makes an ideal pocket-sized charger to keep in your field box.

`

Fig 3.7 Lipo battery charger

3.8.1. Specifications

Input Voltage (V AC)	100 ~ 240
Charge Current Range (A)	3 x 800mA
Li-ion/Po cell count	2 ~ 3
Cable length	0.5m
Dimensions (mm) LxWxH	95x55x32

`

CHAPTER 4

4. Specifications and dimensions

- WEIGHT - 2.83 kg
- MOTOR TORQUE – 10 kg/cm
- ROBOT WORKING TIME WITH LIPO BATTERY

  Voltage = 5v

  Amps = 3.3A/hr

  Capacity =25C

  Amps consumed per sec = 83 A/s

  Robot power consumption = 4.5A

  Our robot can run for 40 mins appox

- ROBOT DIMENSIONS:

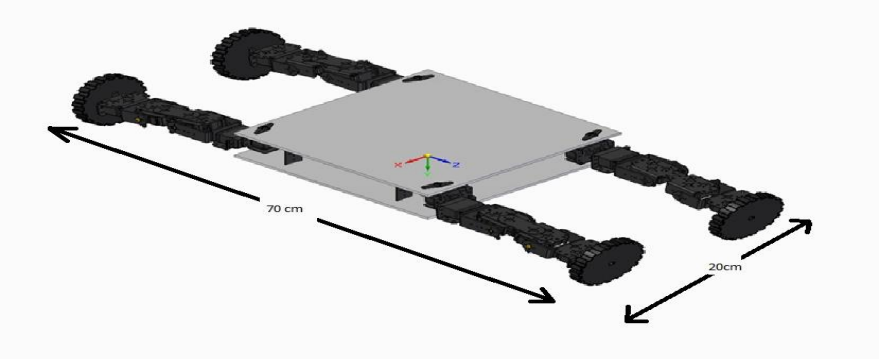

Fig 4.1 Flat mode dimension (70×20cm)

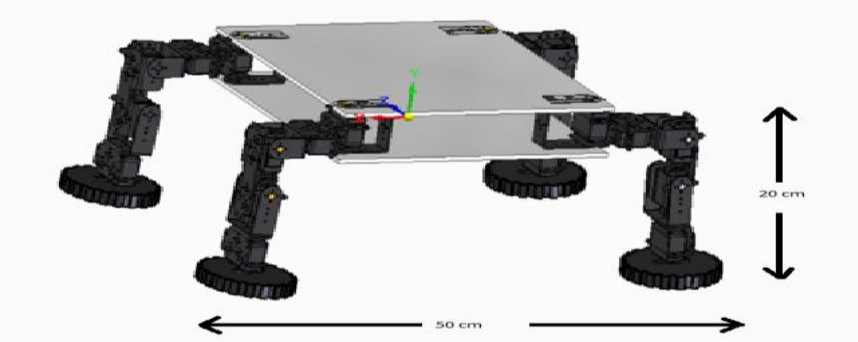

Fig 4.2 Stand mode dimension (50×20 cm)

`

CHAPTER 5

5. Design in solidedge

Solidedge is a solid modeling computer-aided design (CAD) and computer-aided engineering (CAE) software program that runs on Microsoft Windows. The design and development of different configuration of the robot are explained with the help of Solidedge models in this chapter.

5.1. How to attach servos with brackets

After building the servo brackets, the next task was to join the servo motors with the brackets mentioned earlier.

The following figure explains the procedure for joining the servos with brackets.

Fig 5.1 Servo attachment with brackets

A servo is first attached with M bracket and then according to the requirement the U bracket can be attached with M bracket to obtain the required shape. All the servos are attached in such a way that the required structure of transforming robot is developed and the robot is able to change its shape by the movement of the servos at different angles.

`

5.1.1. Walk mode

In this configuration the motors are moves such that it can move on four legs. The solid edge model of the leg of robot for walking is shown below.

Fig 5.2 Servo attachment with brackets

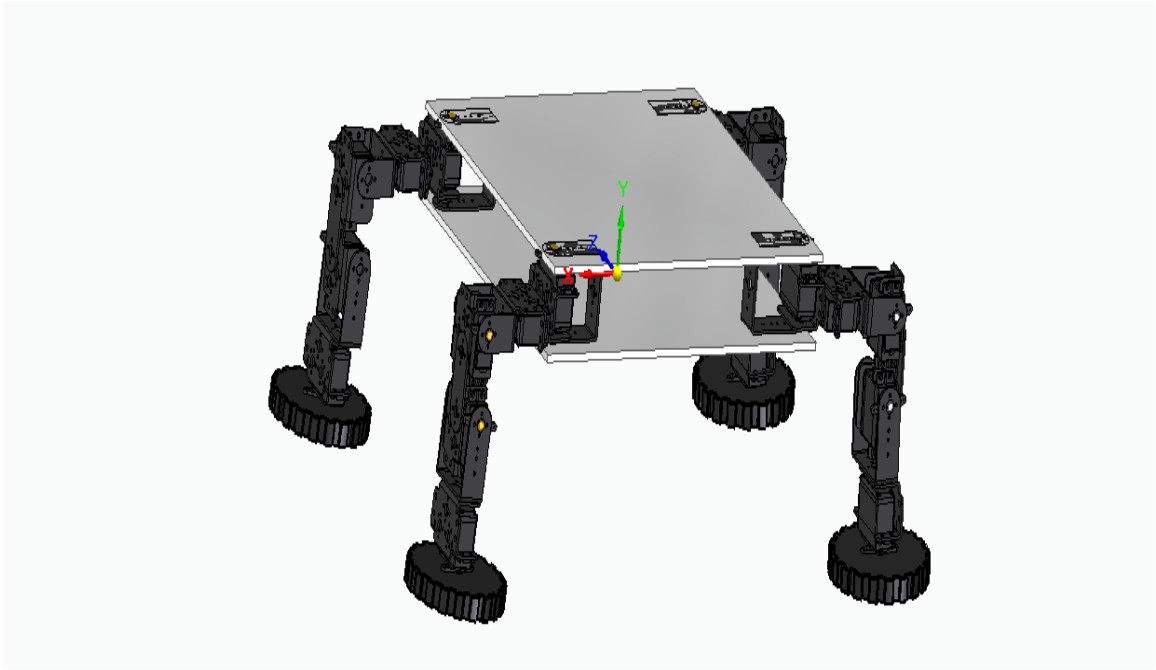

Fig 5.3 Servo attachment stand mode

`

5.1.2. Car mode

In this mode the robot transforms itself in such a way that it can move on four wheels. The robot moves on the wheels attached to the end of each leg.

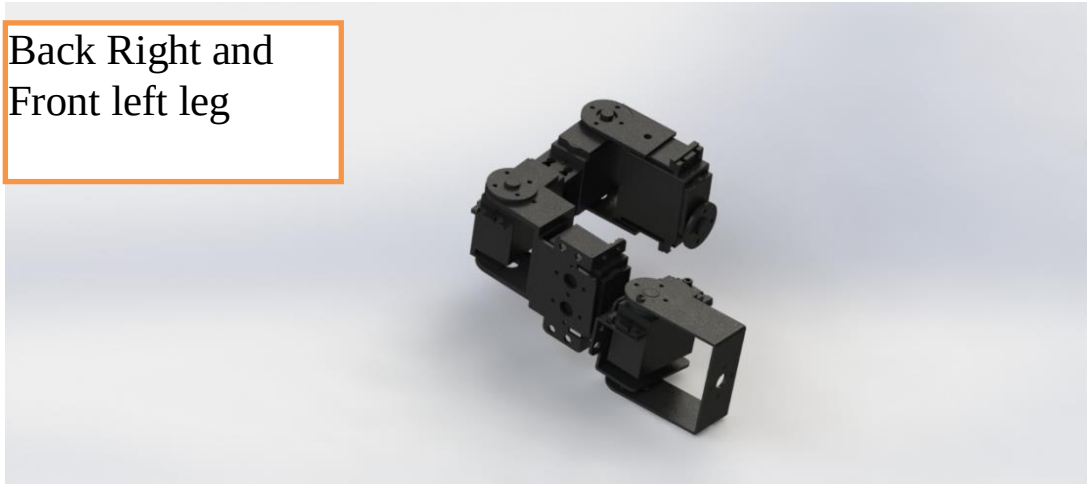

Fig 5.4 Servo attachment with brackets

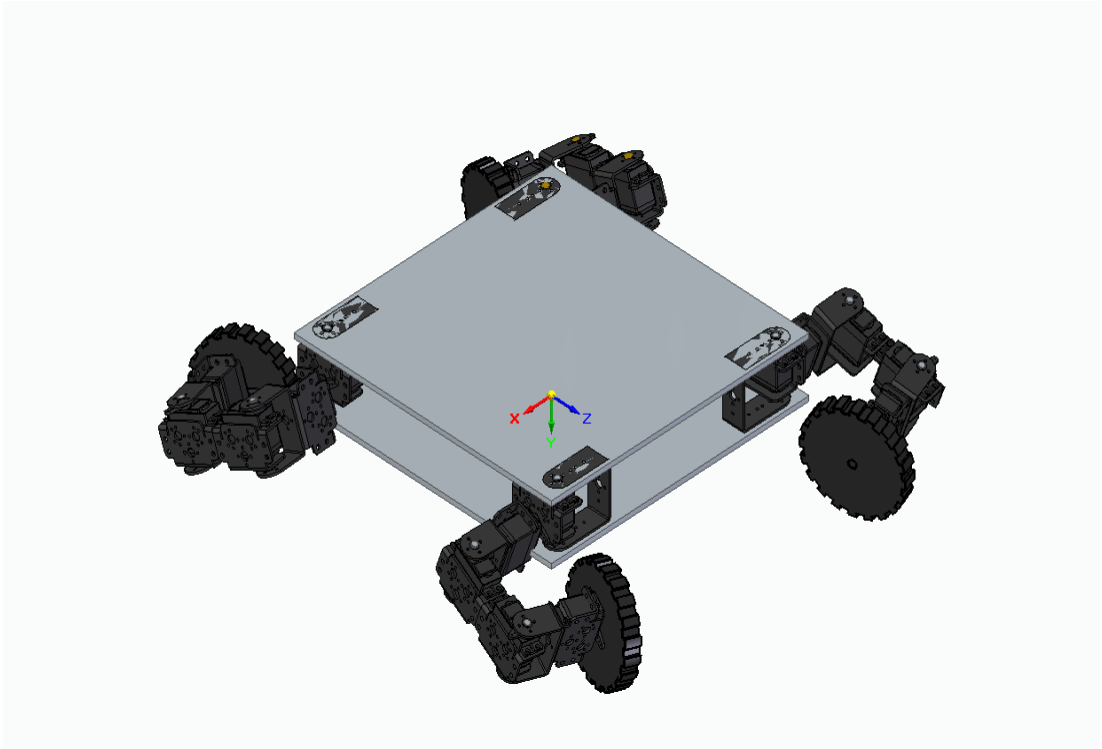

Fig 5.5 Servo attachment car mode

`

5.1.2. Crawling mode

This configuration is obtained for moving on rough surfaces. The solid edge model for this mode is explained below with the help of different figures:

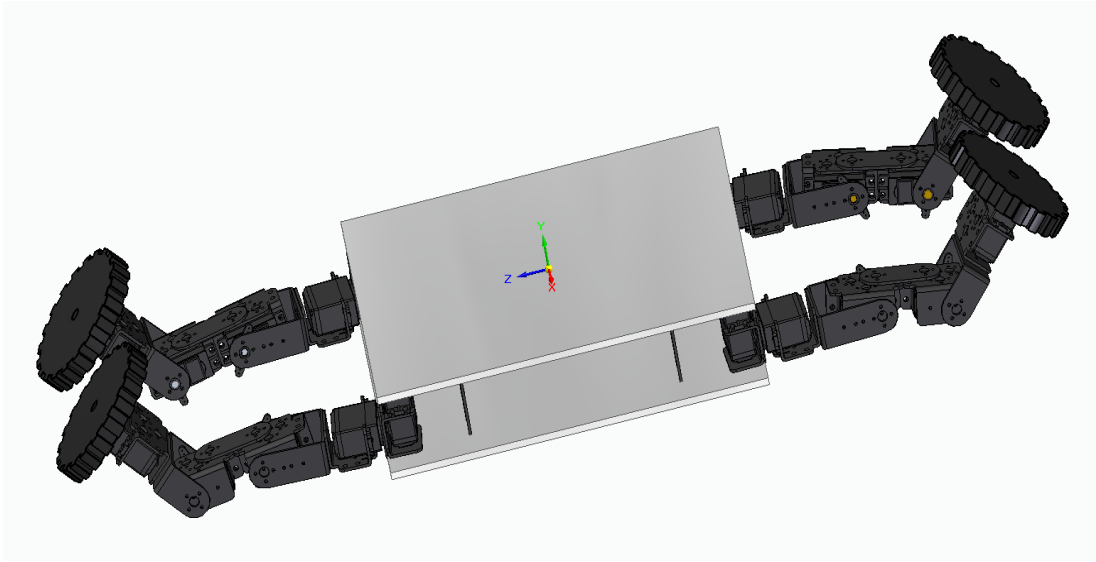

Fig 5.6 Servo attachment crawl mode

5.1.3. Flat car mode

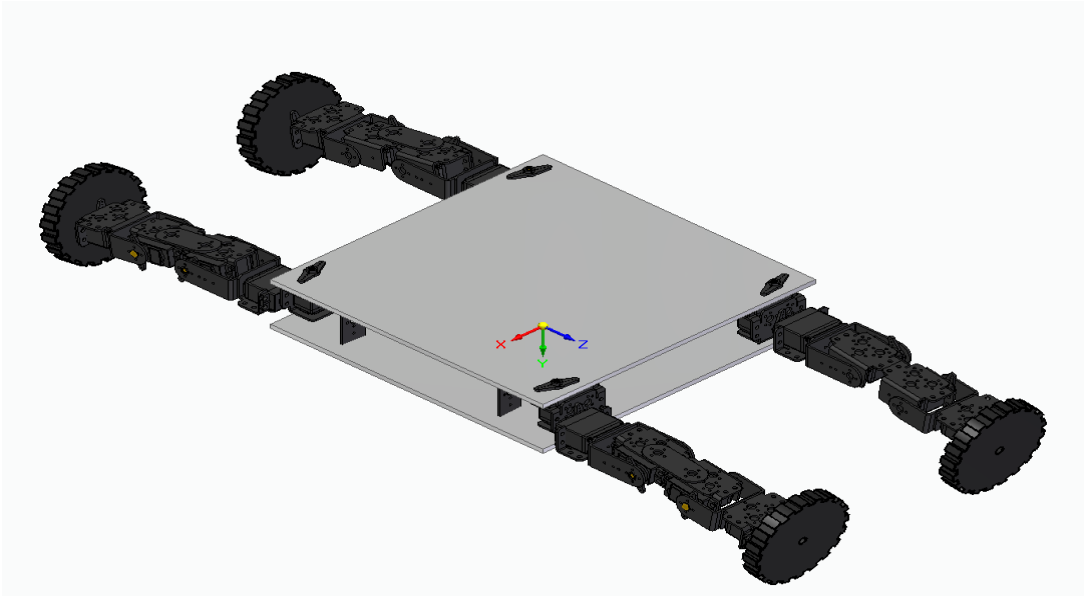

Fig 5.7 Servo attachment flat car mode

`

CHAPTER 6

6. Circuit board

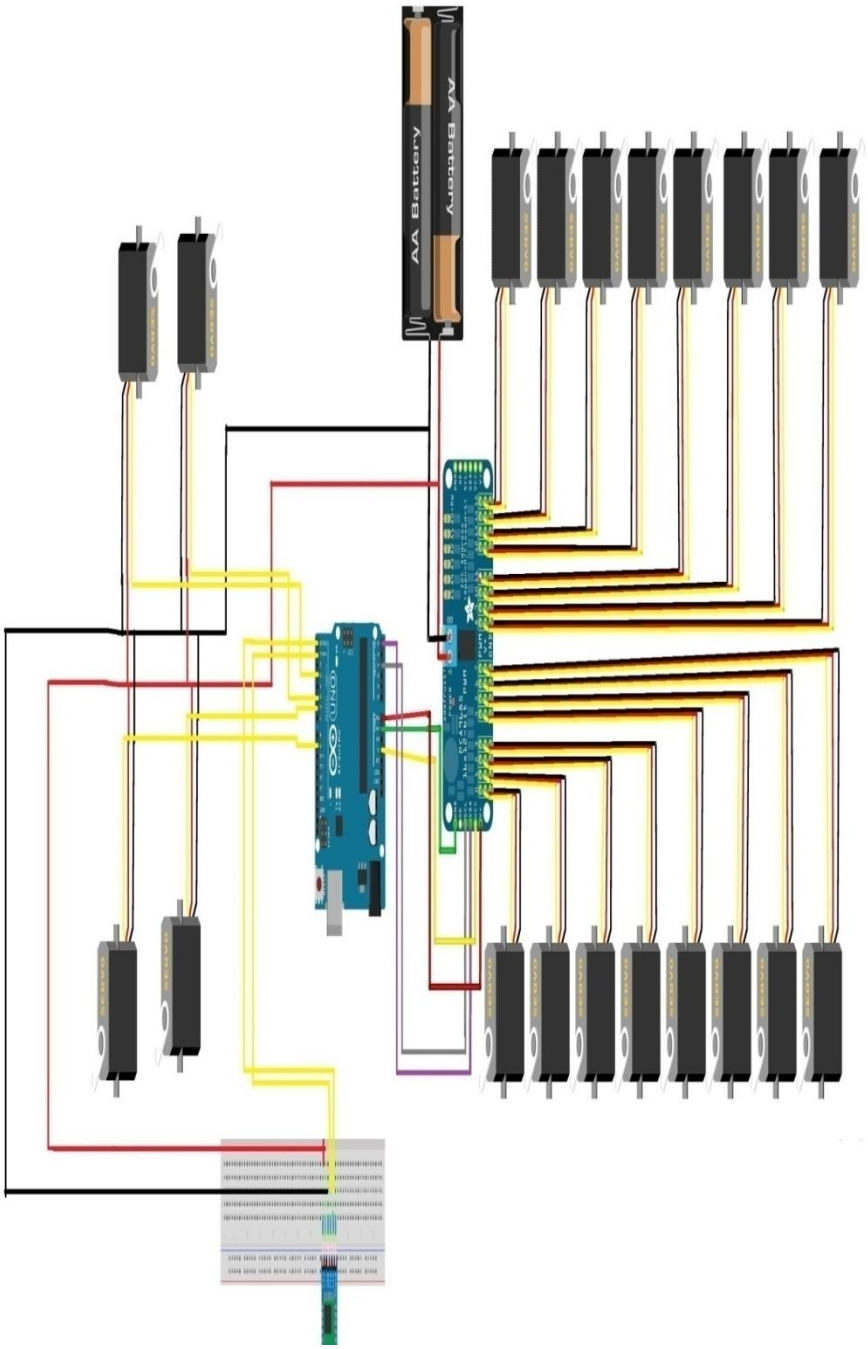

Fig 6.1 circuit diagram

`

CHAPTER 7

7. Gait analysis & representations

In this part, the gait analysis and representations of different configurations and transformations are described using the photos of transforming robot which we developed.

7.1. Default position

In default position the robot stands on four legs. The wheels attached at the end of each leg helps to balance the robot. All the servos are set to 90 degree so that the required default position is obtained.

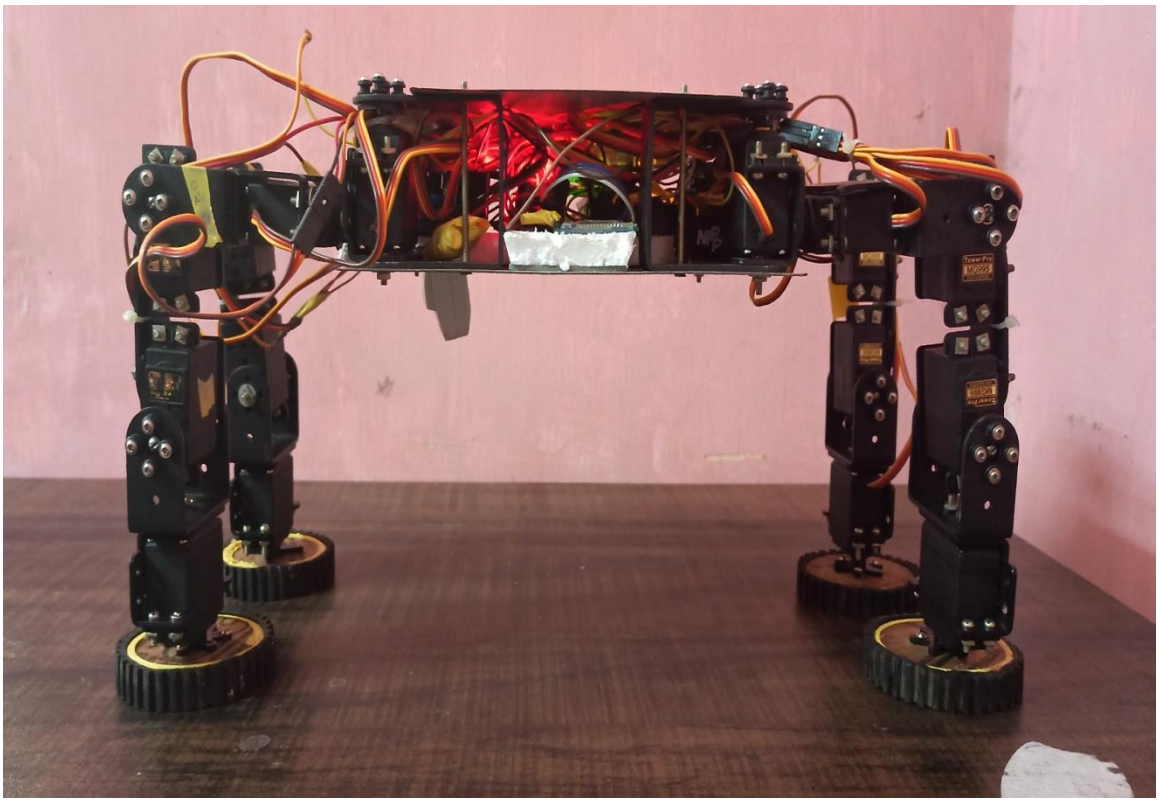

Fig 7.1 Default position

`

7.2. Walk mode

In order to walk on four legs it was necessary to find such a gait so that it could move forward and balance itself as well.

The gait for walking was developed by observing the motion of four legged animals. The motion is obtained by lifting diagonal legs in the air while at the same time the other two diagonal legs which are on the ground move back in order to move the robot forward. For walking mode this gait is repeated again and again so that the robot moves forward like all other four legged animals. The required angles for each leg were found by solving inverse kinematic equations. In this case the end effectors are the wheels attached to the end of each leg considering the base as reference.

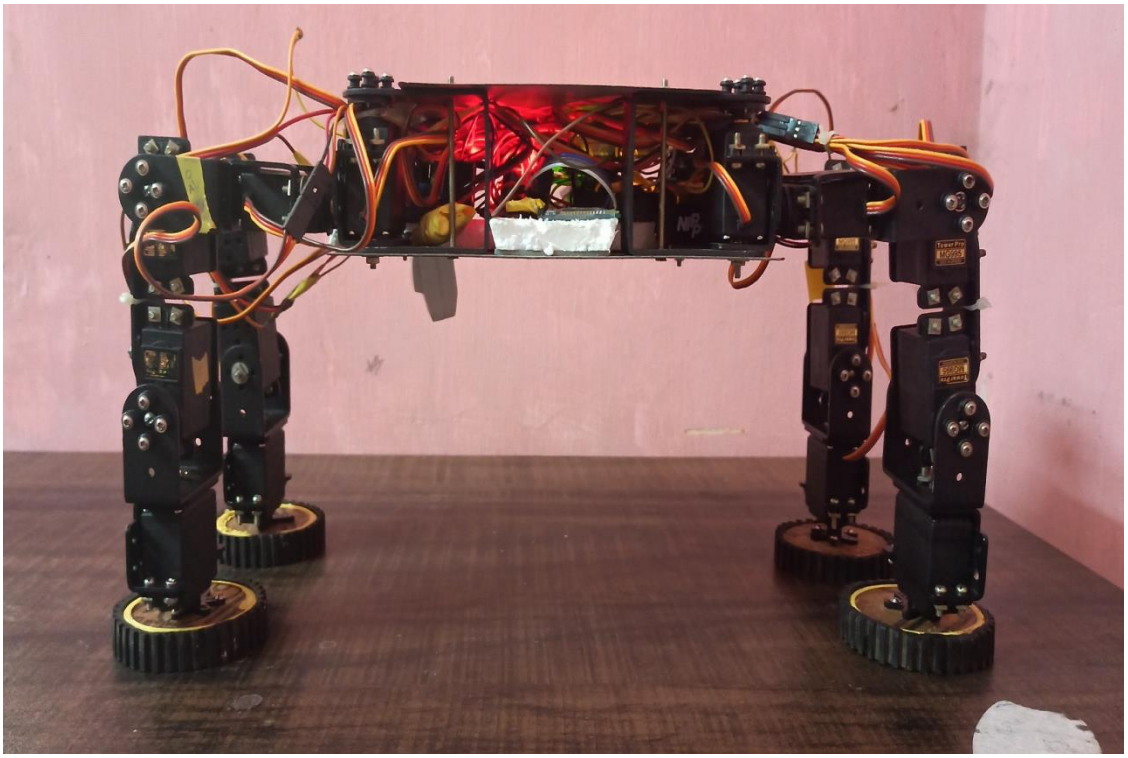

Fig 7.2 Default position

`

(i)Step One:

In first step front left and back right which are diagonal legs are lifted above the ground while the other two legs move back such that robot moves forward.

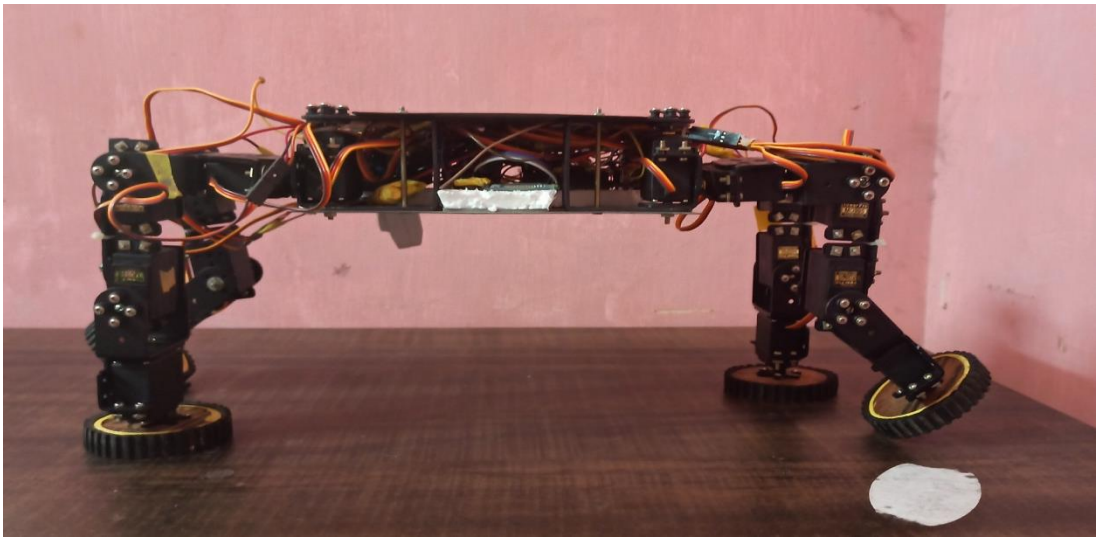

Fig 7.3 walk mode step 1

(ii)Step Two:

This step is the mirror of the first step. In this step front right and back left legs are lifted up from the ground while the motion of other two legs forces the robot to move forward.

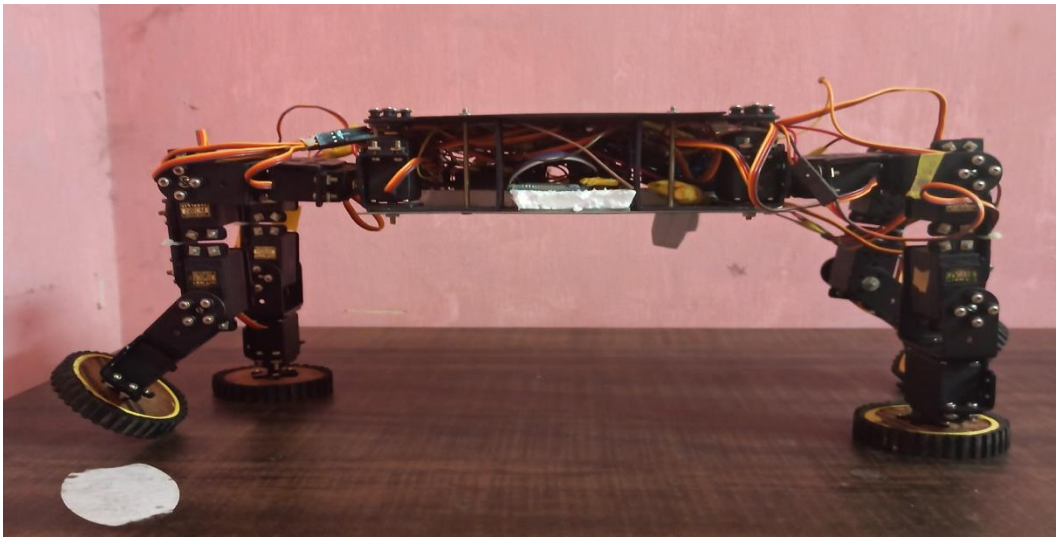

Fig 7.4 walk mode step 2

`

7.3. Transformation to car mode

As mentioned in earlier chapters the robot is capable of changing its shape according to the environment. In order to move through narrow passage it was required to transform the robot in such configuration so that its height is lower than the passage.

This transformation is carried out in steps. In first step the base of the robot can be considered as the end effector. The base is lowered so that it is easier for the robot to balance and transform itself. In next step the base remains stationary while the wheels are considered as end effectors. In the last step the all the legs move in such a way that the base is further lowered and the robot finally transform itself in car mode.

(i) Step one:

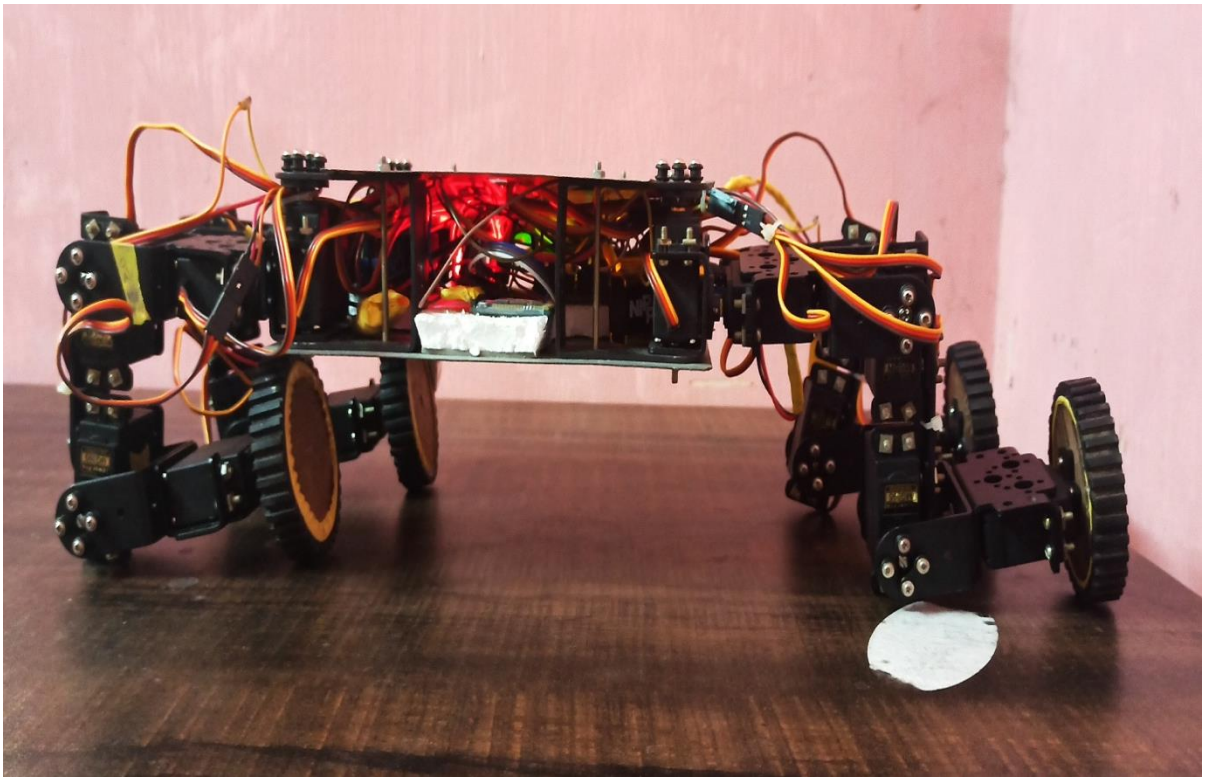

Fig 7.5 car mode step 1

`

(ii) Step two:

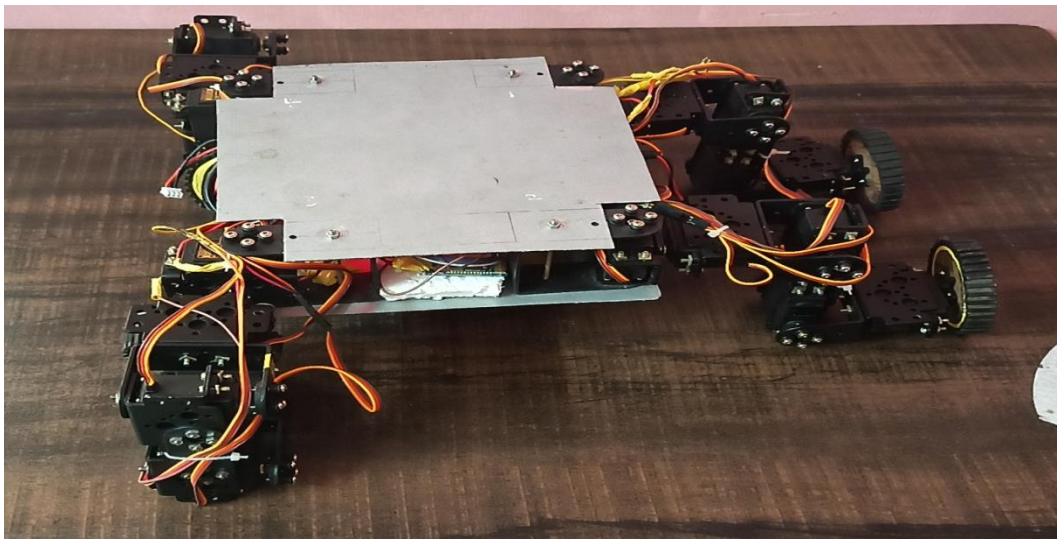

Fig 7.6 car mode step 2

(iii) Step three:

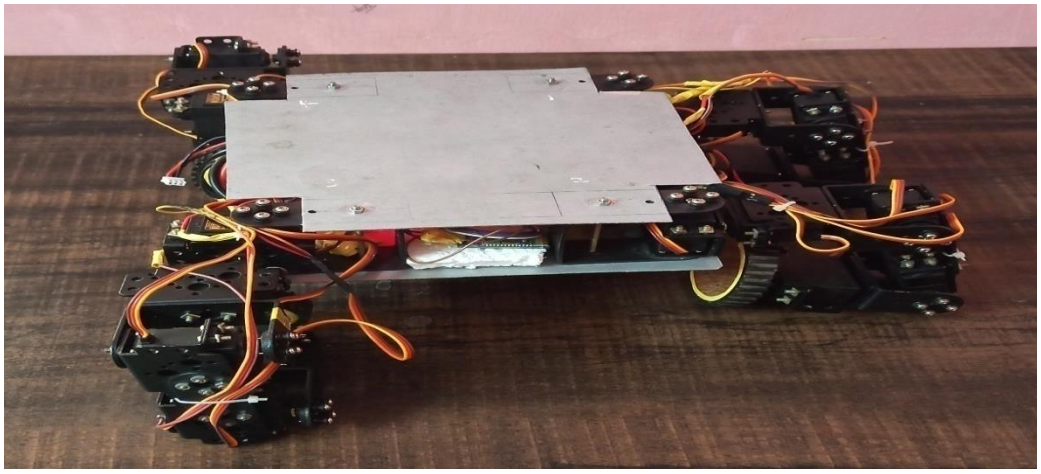

Fig 7.7 car mode step 3

(iv) Step four:

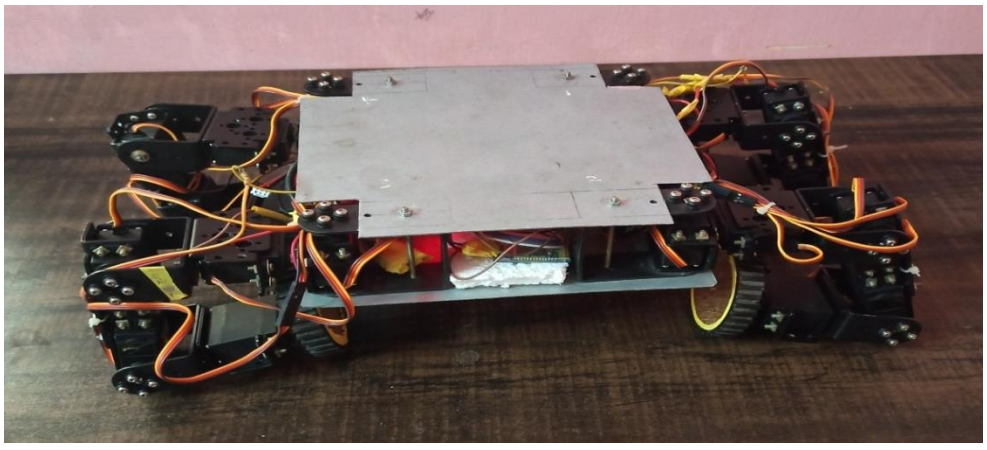

Fig 7.8 car mode step 4

(v) Step five:

`

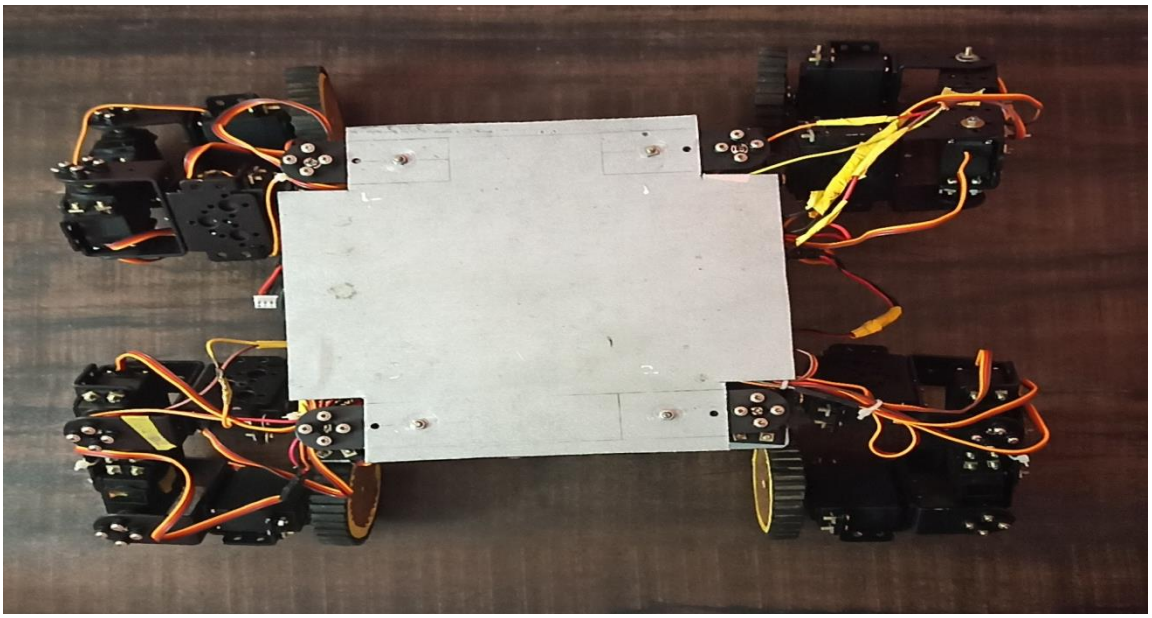

Fig 7.9 car mode step 5

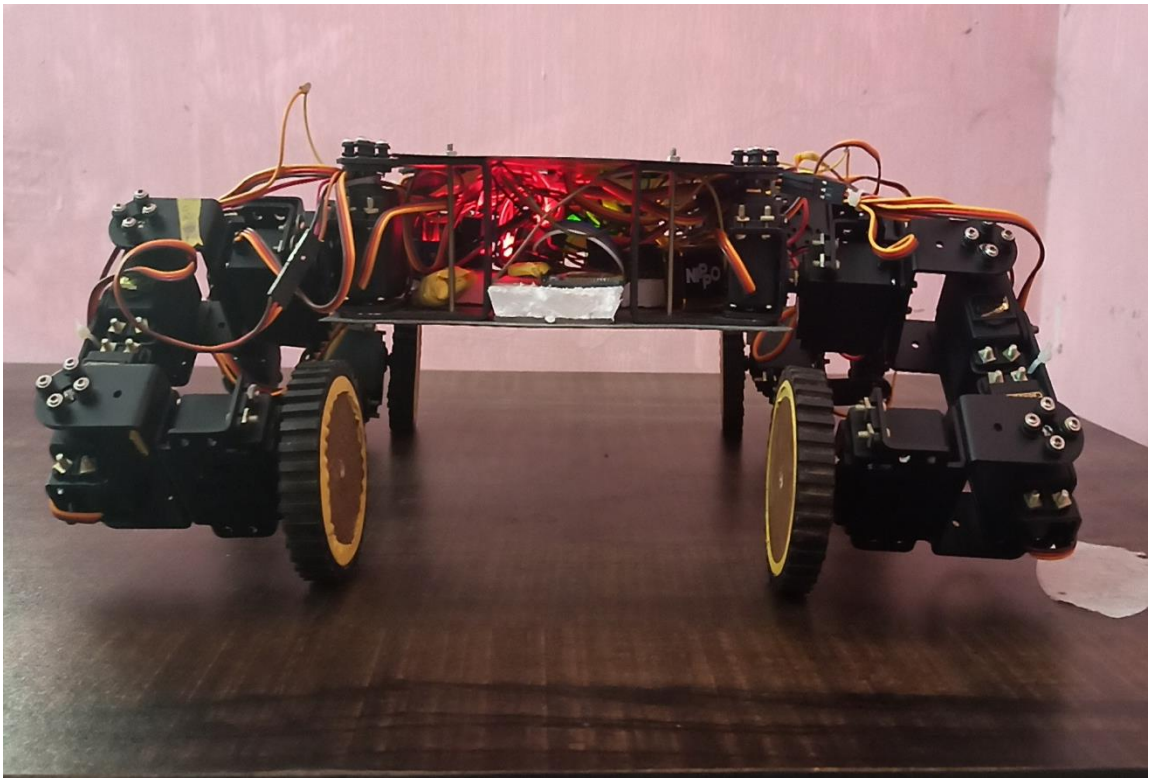

Fig 7.10 Car mode front view

`

7.4. Transformation to crawling mode

The robot can walk on plain surface while for moving through passage it can transform itself into the shape of car. In order to move on rough surfaces such as grass, the robot needs another configuration. This configuration is crawling mode.

Depending upon the output of sensors the robot detects the rough surface and transforms itself into crawling mode. The complete gait analysis of this transformation is explained in the following figures.

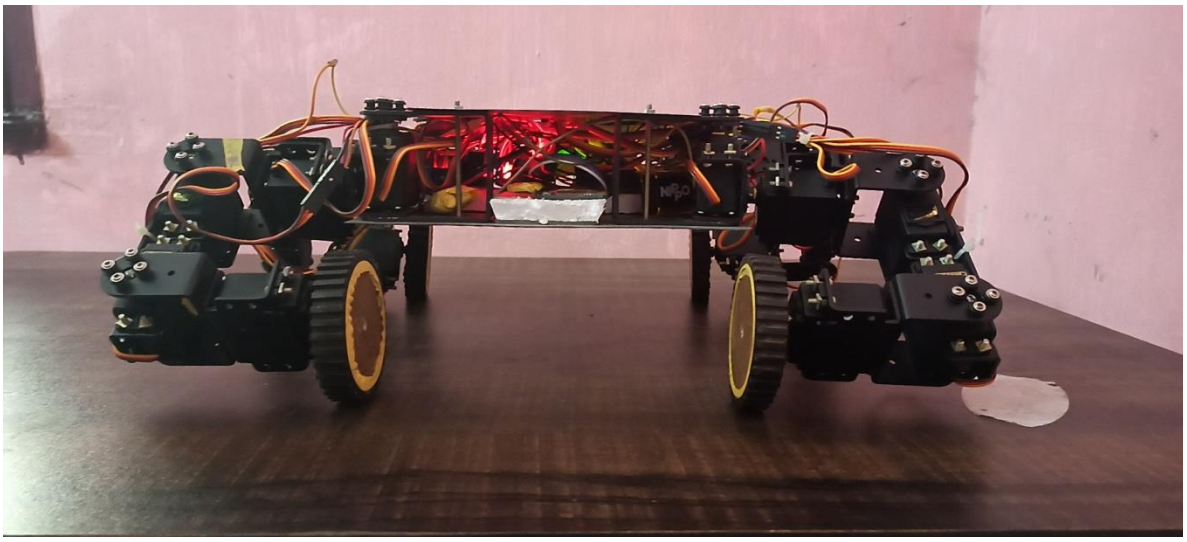

Fig 7.11 Car mode front view

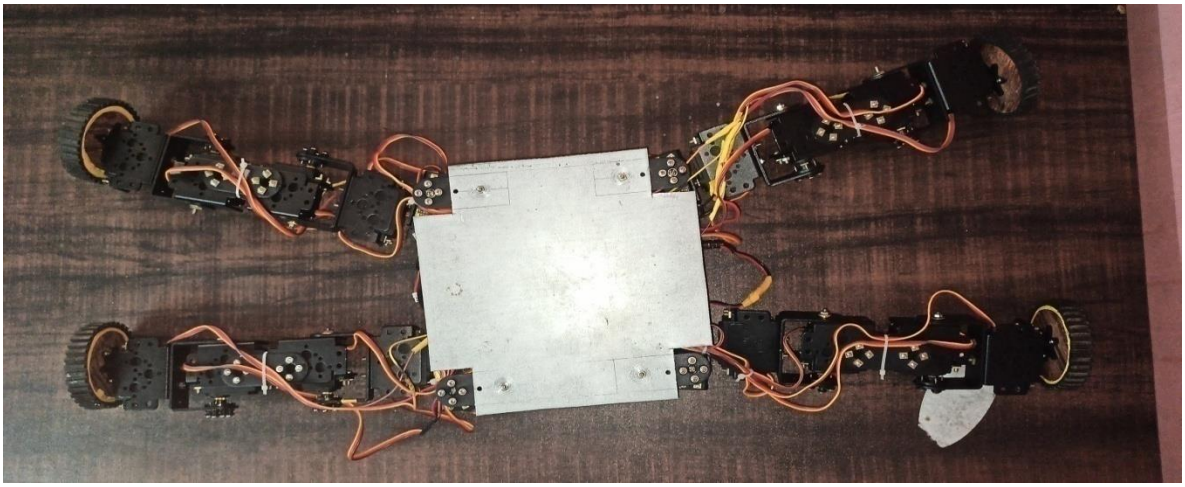

Fig 7.12 Flat Car mode top view

`

7.5. Crawling

The gait analysis for crawling mode was developed by observing the crab. The gait consists of four steps which are repeated for moving in crawling mode. In first step all the legs are lifted up from the ground. In the second step all the legs are moved forward while they are still lifted up from the ground. In next step all legs come back to ground so that the base of the robot is lifted from the ground. Now all legs move forward still touching on the ground while the base is in the air resulting in the forward motion of the root on irregular surface.

(i) Step one:

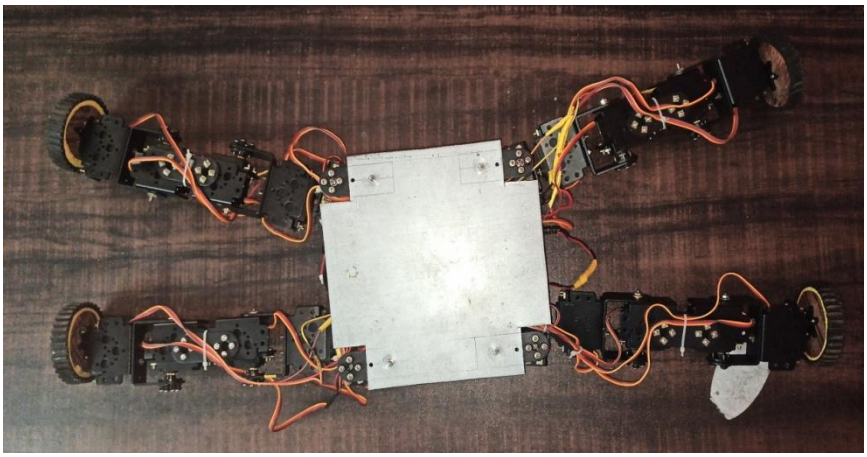

Fig 7.13 Crawl mode step 1

(ii) Step two:

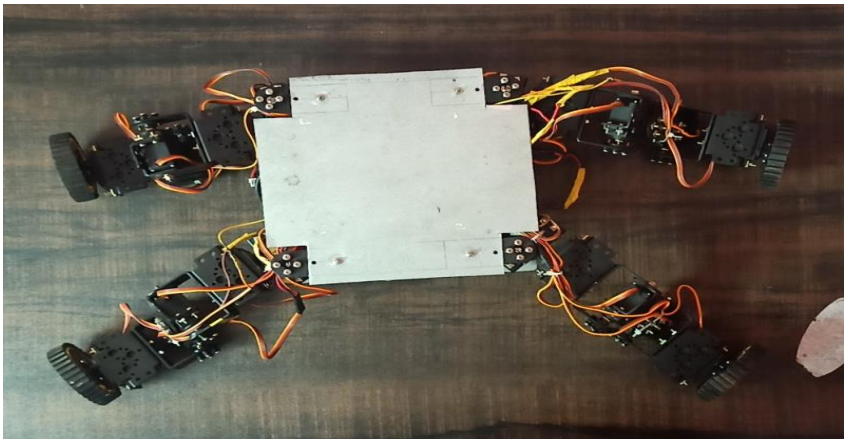

Fig 7.14 Crawl mode step 2

`

(iii) Step three:

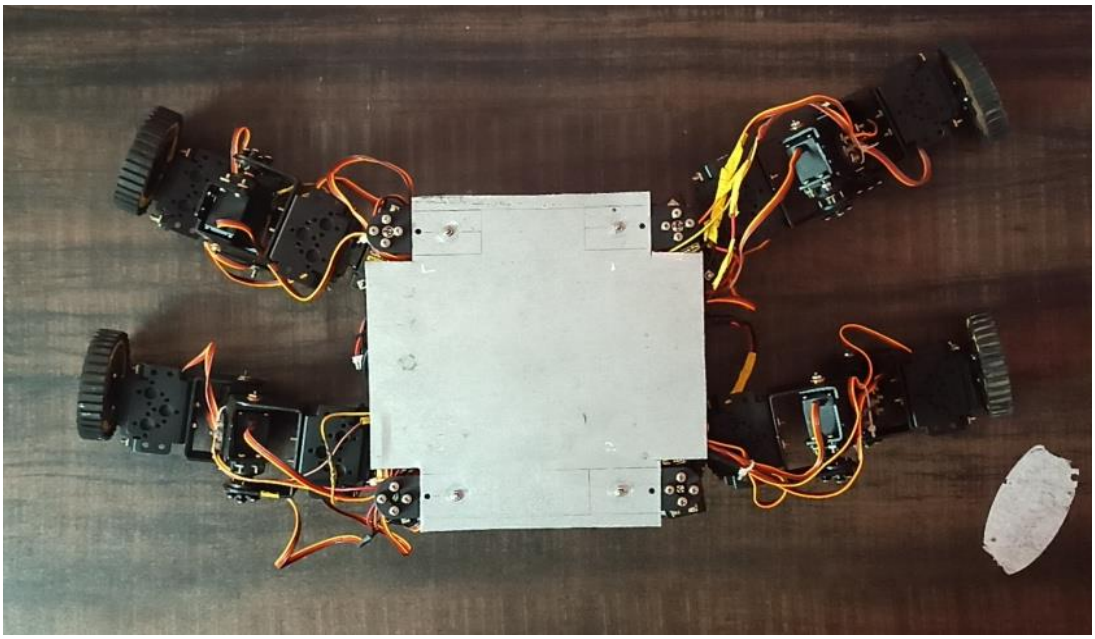

Fig 7.15 Crawl mode step 3

(iv) Step four:

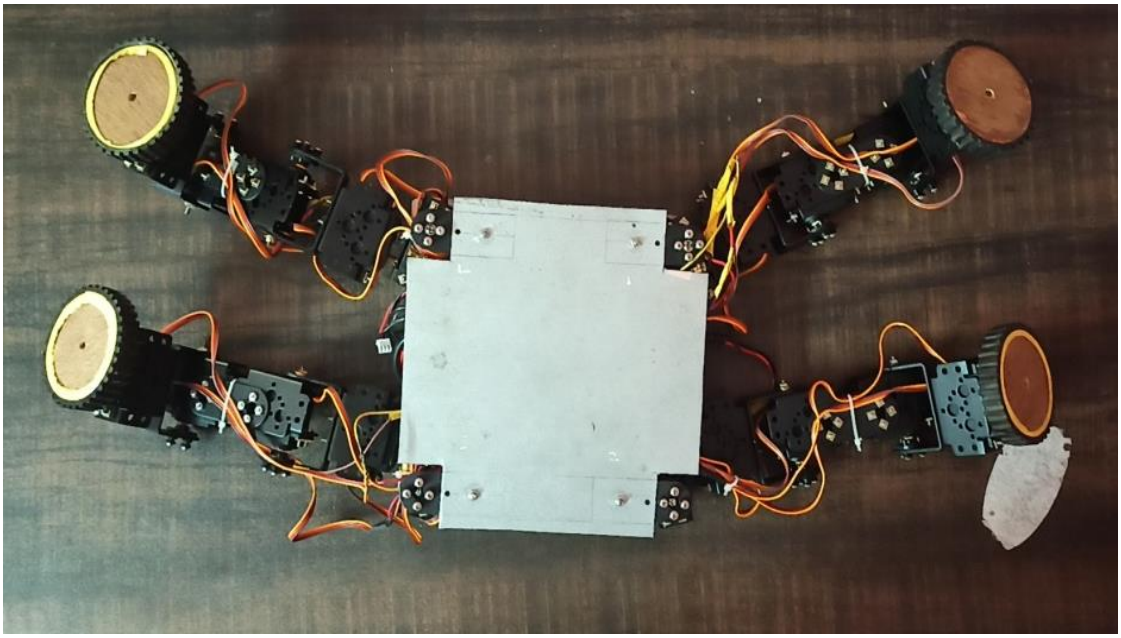

Fig 7.16 Crawl mode step 4

`

CHAPTER 8

8. Mit app inventor

MIT App Inventor is a web application integrated development environment originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT). It allows newcomers to computer programming to create application software(apps) for two operating systems (OS): Android, and iOS, which, as of 8 July 2019, is in final beta testing. It is free and open-source software released under dual licensing: a Creative Commons Attribution Share Alike 3.0 Unported license, and an Apache License 2.0 for the source code.

8.1. MIT webpage interface

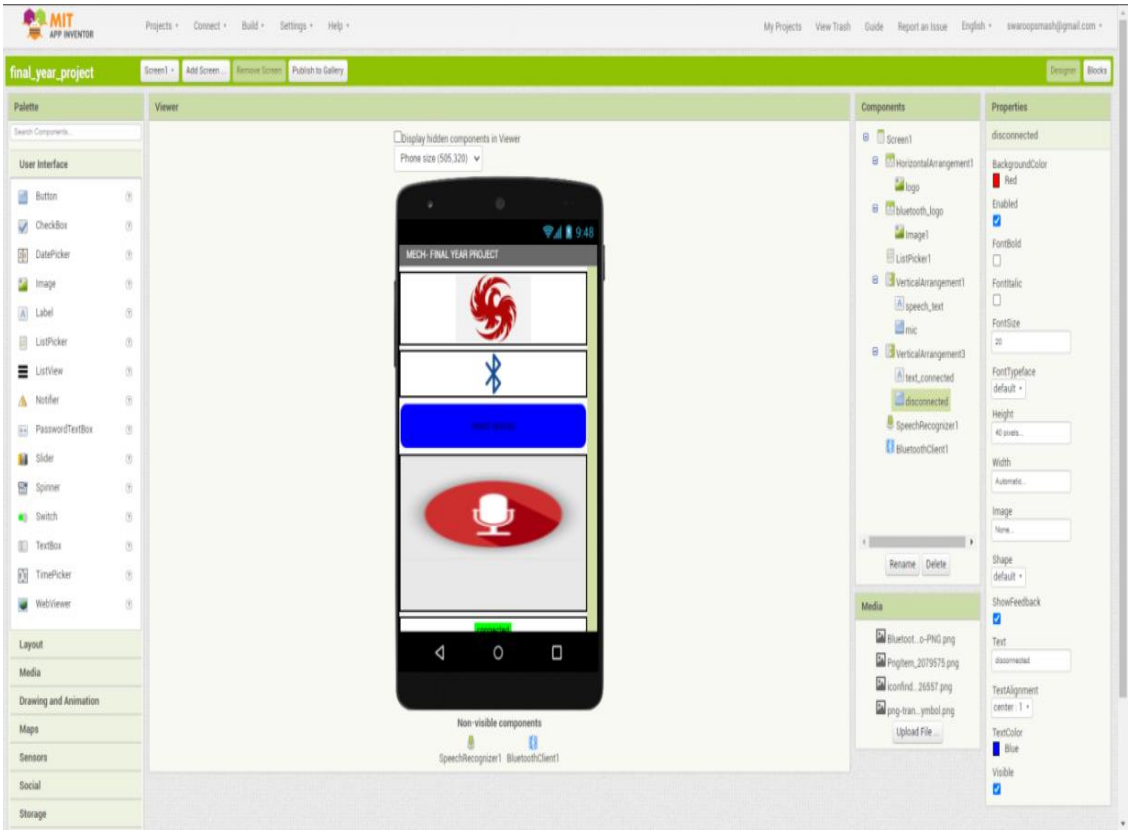

Fig 8.1 MIT webpage interface

`

8.2. MIT app interface

Fig 8.2 Mit app interface

`

8.3. Similar app interface (automation)

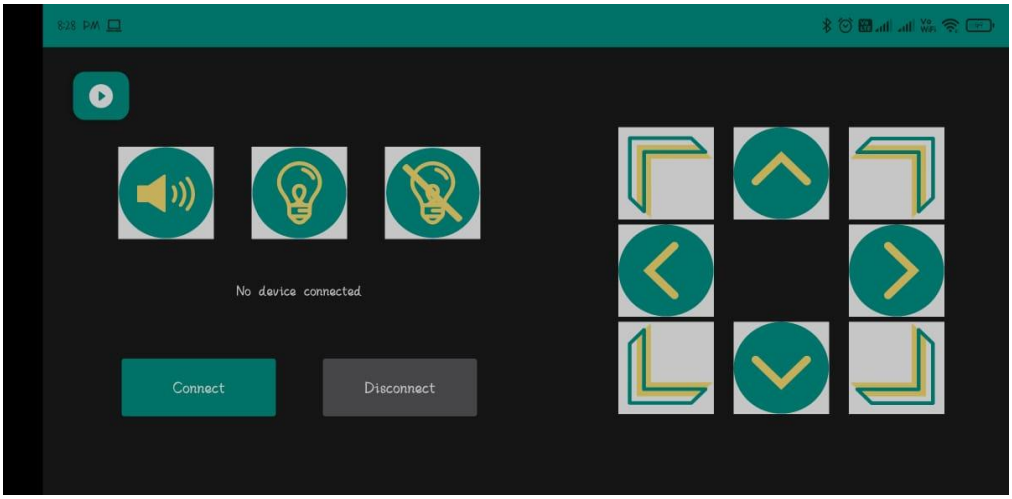

Fig 8.3 Similar app interface

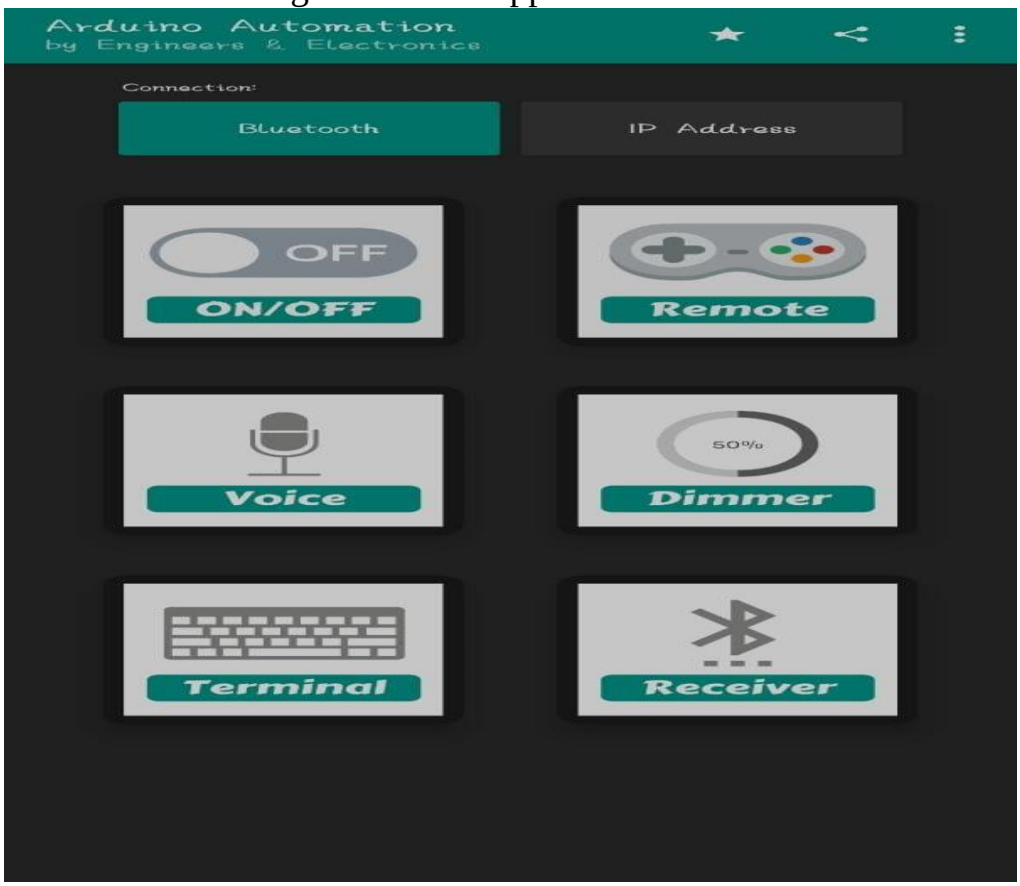

Fig 8.4 Similar app interface Automation

CHAPTER 9

9. Arduino IDE software

The Arduino Integrated Development Environment is a cross-platform application that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of third-party cores, other vendor development boards.

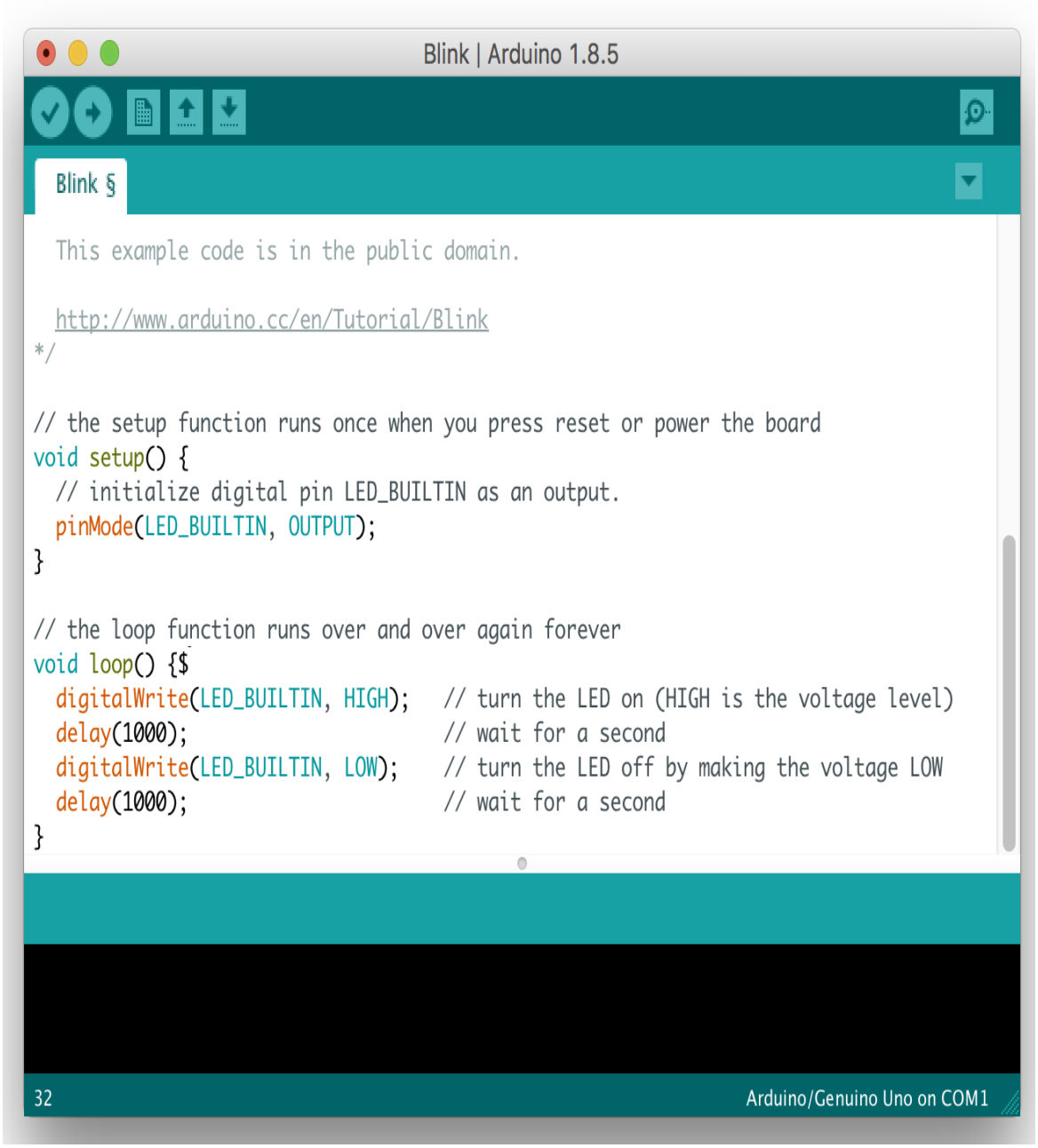

Fig 9.1 Arduino software interface

`

CHAPTER 10

10. Arduino code

```
#include <Servo.h>
#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
int pos0 = 155; // this is the 'minimum' pulse length count (out of 4096)
int pos180 = 630 ;
int flag = 0;
 int q=0;
 int a=0;
Servo w1,w2,w3,w4;
String command;


void setup() {
  Serial.begin(9600);
  Serial.println("fianl year project!");
  pwm.begin();
  pwm.setPWMFreq(60);
  delay(3000);
  w1.attach(3);
  w2.attach(5);
  w3.attach(6);
  w4.attach(9);
}

void setServo(int servo, int angle) {
  int duty;
  duty = map(angle, 0, 180, pos0, pos180);
  pwm.setPWM(servo, 0, duty);

}


void loop(){


   while(Serial.available()) {
  delay(10);
```

```
  command = "";
  command = Serial.readString();

    Serial.print(command);
}
if (command == "start" )
{
   InitialPosition();

}
else if ((command == "walk" ))
{
    InitialPosition();
    delay(100);
    InitialToWalk();
    delay(100 );
    command = "";
 }

else if ((command == "startwalk" ) ){
   Walk();
}
else if ((command == "stopwalk" )){
   InitialToWalk();
  command = "";
  }
else if ((command == "go" )){
InitialPosition();
initialToCar();
command = "";
}
else if ((command == "crawl") or (command == "go3") ){
 InitialToCrawl();
 command = "";
}
else if ((command == "crawlforward" )){
 Crawl();
 }
else if ((command == "crawlstop" )){
   crawlStop();
   command = "";
}
```

`

```
else if ((command == "go2" )){
InitialPosition();
car2();
command = "";
}

else if ((command == "goforward" ) or (command == "FS" )){
  MoveLeft();
}

else if ((command == "gobackward" ) or (command == "BS" )){
  MoveRight();
}
else if ((command == "goright" ) or (command == "RS" )){
  MoveForward();
}
else if ((command == "goleft" ) or (command == "LS" )){
  MoveBackward();
}
else if ((command == "stop" ) or (command == "Y" )){
  Stop();
  command = "";
}
}
void InitialPosition() {

    setServo(0,105);
    setServo(1,80);

    setServo(3,90);


    setServo(4,80);
    setServo(5,80);

    setServo(7,90);

    setServo(8,90);
    setServo(9,80);

    setServo(11,90);

    setServo(12,75);
    setServo(13,80);
```

`

```
      setServo(15,90);


     setServo(2,170);
     setServo(6,170);
     setServo(10,10);
     setServo(14,170);
  }

void InitialToCrawl(){
    setServo(0,90);
    setServo(1,90);
    setServo(2,90);
    setServo(3,90);

    setServo(4,90);
    setServo(5,90);
    setServo(6,90);
    setServo(7,90);

    setServo(8,90);
    setServo(9,90);
    setServo(10,90);
    setServo(11,90);

    setServo(12,90);
    setServo(13,90);
    setServo(14,90);
    setServo(15,90);

}
void Crawl() {

      setServo(3, 160);
      setServo(7, 160);
      setServo(11, 20);
      setServo(15, 160);
if(flag==0){
  for (int i=0;i<=40;i++){
       setServo(0,90+i);
      setServo(4,90+i);
      setServo(8,90-i);
      setServo(12,90-i);
```

```
      delay(20);
}

    delay(1000);
 flag = flag+1;
 }



    for (int i=0;i<=70;i++){
      setServo(2,90+i);
      setServo(6,80+i);
      setServo(10,100-i);
      setServo(14,90+i);
      delay(30);
      }
    delay(2000);



    for (int i=0;i<=70;i++){
      setServo(0, 130-i);
      setServo(4,140-i);
      setServo(8, 40+i);
      setServo(12,50+i);
        delay(20);

      }
    delay(1000);

for (int i=0;i<=70;i++){
      setServo(2,130-i);
      setServo(6,150-i);
      setServo(10,30+(10+i));
      setServo(14,130-i);
      delay(30);
      }
    delay(2000);

for (int i=0;i<=70;i++){
      setServo(0, 60+i);
      setServo(4,70+i);
      setServo(8, 110-i);
      setServo(12,120-i);
```

```
      delay(20);

    }
   delay(1000);

 for (int i=0;i<=30;i++){
  setServo(2,60+i);
  setServo(14,60+i);
 delay(20);

    }
   delay(1000);

   for (int i=0;i<=10;i++){
  setServo(10,110-i);

 delay(20);

    }
   delay(1000);
 }


void initialToCar(){
  setServo(6,160);
   setServo(2,150);
   setServo(10,15);
   setServo(14,150);
 for(int i=0;i<=90;i++){
   setServo(3,90+i);
   setServo(7,90+i);
   setServo(11,90+i);
   setServo(15,90-i);
 delay(20);
 }
   delay(500);

   for(int i=0;i<=90;i++){
   setServo(8,90-i);
   setServo(12,90+i);
   delay(20);
   }
   delay(500);
```

```
    for(int i=0;i<=90;i++){
    setServo(3,90-i);
    setServo(7,90-i);
    delay(20);
    }
    delay(500);

    for(int i=0;i<=90;i++){
    setServo(0,90-i);
    setServo(4,90+i);
    delay(20);
    }
    delay(500);

for(int i=0;i<=90;i++){
    setServo(0,10+i);
    setServo(4,180-i);
    delay(20);
    }
    delay(500);
     for(int i=0;i<=90;i++){
    setServo(8,0+i);
    setServo(12,160-i);
    delay(20);
    }
    delay(500);

  for(int i=0;i<=5;i++){
  setServo(1,80+10*i);
   setServo(5,80-9*i);
    setServo(9,80+10*i);
    setServo(13,80-8*i);
    delay(20);
    }
    delay(500);
    }
void MoveLeft(){
  w1.write(360);
  w2.write(360);
  w3.write(0);
  w4.write(0);
  }
void MoveRight(){
  w1.write(0);
```

```
  w2.write(0);
  w3.write(360);
  w4.write(360);
  }
  void MoveForward(){
  w1.write(360);
  w2.write(360);
  w3.write(70);
  w4.write(70);
  }
  void MoveBackward(){
  w1.write(110);
  w2.write(110);
  w3.write(0);
  w4.write(0);
  }
  void Stop(){
     w1.write(90);
  w2.write(90);
  w3.write(90);
  w4.write(90);
   }



void Walk(){
 for(int i=0;i<=75;i++){
  setServo(3,90+i);
  setServo(2,180);
  setServo(10,-30);
  setServo(11,90-i);
delay(20);
  }
  delay(500);

for(int i=0;i<=45;i++){
  setServo(1,90-i);
   setServo(9,90+i);
delay(20);
  }
  delay(500);

for(int i=0;i<=75;i++){
```

```
  setServo(3,165-i);
  setServo(11,15+i);
  setServo(2,170);
  setServo(10,0);
delay(20);
}
for(int i=0;i<=45;i++){
  setServo(1,40+i);
   setServo(9,125-i);
delay(20);
  }
 delay(500);


// other two legs

for(int i=0;i<=75;i++){
  setServo(7,90+i);
  setServo(6,200);
  setServo(14,180);
  setServo(15,90+i);
delay(20);
 }
 delay(500);

for(int i=0;i<=35;i++){
  setServo(5,90+i);
   setServo(13,90-i);
delay(20);
  }
  delay(500);

for(int i=0;i<=75;i++){
  setServo(7,165-i);
  setServo(15,165-i);
  setServo(6,170);
  setServo(14,170);
delay(20);
}
for(int i=0;i<=35;i++){
   setServo(5,115-i);
   setServo(13,55+i);
delay(20);
  }
```

`

```
 delay(500);

}


void InitialToWalk() {

     setServo(0,5);
     setServo(1,80);

     setServo(3,90);


     setServo(4,175);
     setServo(5,80);

     setServo(7,90);

     setServo(8,0);
     setServo(9,80);

     setServo(11,90);

     setServo(12,160);
     setServo(13,80);

  setServo(15,90);
   setServo(2,160);
    setServo(6,170);
    setServo(10,10);
    setServo(14,160);
  }

  void car2 (){
   setServo(6,160);
   setServo(2,150);
   setServo(10,15);
   setServo(14,150);
  for(int i=0;i<=90;i++){
   setServo(3,90+i);
   setServo(7,90+i);
   setServo(11,90+i);
   setServo(15,90-i);
  delay(20);
```

```
`


}
                    delay(500);
                    for(int i=0;i<=90;i++){
                    setServo(8,90-i);
                    setServo(12,90+i);
                    delay(20);
                    }
                    delay(500);
                    for(int i=0;i<=90;i++){
                    setServo(3,90-i);
                    setServo(7,90-i);
                    delay(20);
                    }
                    delay(500);
                    for(int i=0;i<=90;i++){
                    setServo(0,90-i);
                    setServo(4,90+i);
                    delay(20);
                    }
                    delay(500);
                    for(int i=0;i<=10;i++){
                   setServo(1,80-5*i);
                    setServo(5,80+8*i);
                    setServo(9,80-5*i);
                    setServo(13,80+7*i);
                    delay(20);
                    }
                    delay(500)
                    }
                    void crawlStop(){
                    setServo(3, 160);
                      setServo(7, 160);
                      setServo(11, 20);
                      setServo(15, 160);
                   for (int i=0;i<=40;i++){
                        setServo(0,90+i);
                       setServo(4,90+i);
                       setServo(8,90-i);
                       setServo(12,90-i);
                       delay(20);
                }
                     delay(500)
                }
                }
```

`

11. Application

The applications of the project are Military applications, Surveillance, Rescue operations, Medical applications etc. The proposed robot can do surveillance around areas which is in need of security, areas like boundaries which need to be protected and prone to danger. This feature enables the robot to be used in Militaries. The robot can also be used for rescue purpose as it can change its shape in accordance with the incoming obstacles. So this robot will be able to move through areas having rough surfaces and tiny spaces

12. Advantages

The advantages of the project can be highlighted as follows:

- This robot is capable of performing multi-tasks without much human interventions.
- It can carry out rescue operations in areas where human cannot reach easily.
- It can perform surveillance around areas of boundaries for monitoring of any illegal or suspicious activities inclusions or ceasefires violations

13. Limitations

- The making cost of this robot is high.
- Due to more numbers of motors, the power consumption is high.

`

14. Conclusion

In this project Self-transforming robot was designed for linear motion of robot. This robot will provide a modulation of quadrupedal locomotion on rough terrain by simple control signal. A Hardware prototype of self transforming robot was developed. In future this can be further developed using AI integration and It can be used in the fields of defence, space and also sea exploration, rescue missions by incorporating more powerful and precise motors.

In future the robot can be used to do spying operations even during night for catching terrorist, it can be used for bomb detection and bomb defusal as well. Our robot is manually transformed but in the future we can enable it to perform automatic transformation. The system will be able to spy in areas which are in need of security using the help of Wi-Fi camera, which will stream videos to the control system using Wi-Fi.

`

15. References

[1] YE Changlong, MA Shugen and LI Bin, “Development of a shape-shifting mobile robot for urban search and rescue”, Chinese journal of mechanical Engg. vol.21, no. 2, pp. 31-35, January 2008.

[2] Zhiqing Li, Shugen Ma, “Design and basic experiments of a transformable wheel-track robot with self-adaptive mobile mechanism”, in Proc. Int. Conf. on Rob. And Sys. vol. 21, no. 2, pp. 1334-1339, October 2010.

[3] C. Maufroy, H. Kimura and K. Takase, ”Integration of posture and rhythmic motion controls in quadrupedal dynamic walking using phase modulations based on leg loading/uploading”, Autonomous Robots. vol. 28, pp. 331-353, February 2010.

[4] Sam D. Herbert, Andrew Drenner and N. Papanikolopoulos, Loper: A Quadruped-Hybrid stair climbing robot”, in Proc. Int. Conf. on Rob. and Auto. pp. 799-804, May 2008

[5] K.Tsuchiya, Shinya Aoi and K .Tsujita,” Locomotion Control of a Biped Locomotion Robot using Nonlinear Oscillators”, in Proc. Int. Conf. on Rob.and Sys. pp. 1745-1750, October 2003.

[6] P. Tantichattanont and S. Songschon and S. Laksanacharoen, “Quasi-static analysis of a leg-wheel hybrid vehicle for enhancing stair climbing ability,” in Proc. Int. Conf. on Rob.and Bio., 2007, pp. 1601- 1605.

[7] C.K. Woo, H.D. Choi , Sukjune Yoon ,S. H. Kim and Y. K. Kwak, ” Optimal Design of a New Wheeled Mobile Robot Based on a Kinetic Analysis of the Stair Climbing States”., Journal of intell Robot Syst. vol. 49, pp. 325-354, March 2007.

[8] A. Bschges, H.Scholzand and A.E. Manira,” New movies in motor control”, Current Biology. vol. 21, pp. 513-524, 2011.

`

[9] Y. Fukuoka, H. Kimura and A.H. Cohen," Adaptive dynamic walking of a quadruped robot on irregular terrain based onn biological concepts", International Journal of Robotic Research. vol. 21, pp. 187-202, March 2003.

[10] M. Oliveria, C.P. Santos, L. Costa, V. Mators and M. Ferreira," Multi-Objective parameter CPG optimization for gait generation of quadruped robot considering behavioural diversity", Intelligent Robots and Systems, International Conference. pp. 2286- 2291, March 2010.