

**B. TECH FINAL YEAR PROJECT REPORT**  
**ON**  
**“FIRE FIGHTER ROBOT WITH GAS SENSOR USING**  
**IoT”**

*Submitted in partial fulfillment for the award of degree of Bachelor of  
Technology by Assam Science and Technology University*



Submitted to

**Department of Electronic and Telecommunication Engineering**

**Under the guidance of**  
**Prof. Dr. Bijoy Goswami**

**Submitted By:**

**Bibungsar Brahma** 2006100026014

**Dibankar debnath** 180610026055

**Divya Roy** 210650026002

**Department of Electronic and Telecommunication Engineering**

**Assam Engineering College Guwahati**  
**JALUKBARI- 781013, GUWAHAT**

**July , 2024**

## CERTIFICATE

This is to certify that the project entitled “**FIRE FIGHTER ROBOT WITH GAS SENSOR USING IoT**” has been carried out and presented by

**Bibungsar Brahma**

2006100026014

**Dibankar debnath**

180610026055

**Divya Roy**

210650026002

Students of B. Tech, 8<sup>th</sup> Semester (Electrical Engineering), Assam Engineering College, under my supervision and guidance in a manner satisfactory to warrant its acceptance as prerequisite for the award of Bachelor of Engineering in Electrical Engineering of the Assam Science and Technology University (ASTU).

Further the report has not been submitted in any form for the award of any other degree/diploma.

Date:

**Prof. Bijoy Goswami**

Place- Guwahati

Dept. of Electronic and Telecommunication Engineering

Assam Engineering College, Guwahati- 781013

JULY , 2024

## **DECLARATION**

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when nee

(Signature)

Date:\_\_\_\_\_

## **ACKNOWLEDGMENTS**

**We would like to express our sincere gratitude and appreciation to all those who have contributed to the successful completion of this project. Their support, guidance, and encouragement have been invaluable throughout the entire process.**

**First and foremost, we extend our heartfelt thanks to our supervisor, BIJOY GOSWAMI SIR for their continuous support and insightful guidance. Their expertise and willingness to share knowledge have played a vital role in shaping this project and enhancing its quality.**

**We are also thankful to the faculty members of ETE DEPARTMENT, ASSAM ENGINEERING COLLEGE for providing us with a conducive environment for learning and exploration. Their dedication to imparting knowledge and their commitment to excellence have greatly influenced our project's outcomes.**

**We would like to acknowledge the assistance and cooperation received from our fellow classmates and friends. Their valuable insights, brainstorming sessions, and collaborative efforts have been instrumental in overcoming challenges and achieving our objectives.**

**Additionally, We are grateful to the participants and respondents who generously shared their time and opinions, enabling us to collect relevant data and gather valuable insights for our project.**

**We would also like to extend our gratitude to the library staff for their assistance in accessing relevant resources and materials. Their efforts have been instrumental in broadening our understanding and enriching our project.**

**Finally, we are indebted to our family and friends for their unwavering support, encouragement, and understanding throughout this project. Their belief in us and their constant motivation have been our driving force, inspiring us to strive for excellence.**

**In conclusion, we would like to express our deepest appreciation to all those who have contributed directly or indirectly to the completion of this mini project. Without their support, this endeavor would not have been possible. Thank you all for your valuable contributions.**

**Bibungsar Brahma      200610026014**

**Dibankar Debnath      180610026055**

**Divya Roy                210650026002**



## ABSTRACT

There is no doubt that firefighting is an important job, but it is also a very dangerous occupation. The absence of human beings in detection of fire usually leads to a huge damage. This project aims to design a firefighting robot that can operate remotely. The development of Fire Fighting Robot consists of two elements i.e., hardware and programming. The project is designed to develop a fire fighting robot using Arduino uno . The robotic vehicle is loaded with water pump which is controlled by servos. An microcontroller is used for the desired operation. At the transmitting end using commands are sent to the receiver to control the movement of the robot either to move forward, and left or right etc. At the receiving end tow motors are interfaced to the microcontroller where two of them are used for the movement of the vehicle and the one to position the robot. The sensor adequate range with obstacle detection, while the receiver driver module used to drive DC motors via motor driver IC for necessary work. A water tank along with water pump is mounted on the robot body and its operation is carried out from the microcontroller output through appropriate command from the transmitting end. The whole operation is controlled by an microcontroller. A motor driver IC is interfaced to the microcontroller through which the controller drives the motors,three ir flame sensors are fixed on robot chassis to sense the fire and to reach the destination to put off the fire. The type of locomotion used by a mobile robot is crucial for the robot to perform its task and reach its goal in a given environment. This work focuses on the optimization of the design of a fire fighting robot system subject to optimizing well defined mobility metrics. As robots evolve from industrial fixed base robots to autonomous mobile platforms, the concept of locomotion in robotics becomes much more important. Similar to nature, also robot locomotion must be adapted to the given terrain or task.

# CONTENT

<b>PARTICULARS</b>	<b>Page No.:</b>
Cover Page	1
Certificate	2
Declaration	3
Acknowledgement	4
Abstract	5
Content	6
List of figures	7-8
List of table	9

<b>CHAPTERS</b>	<b>CHAPTER CONTENT</b>	<b>Page No.:</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>9 - 12</b>
	1.1 Indroduction	9
	1.2 Project overview	10
	1.3 Component overview	11-12
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>13 - 14</b>
	2.1 L iterature review	13
	2.2 Problem Statement	14



## LIST OF FIGURES

<b>Figure No.:</b>	<b>Name of the figures</b>	<b>Page No.:</b>
2.1	Overview of system	16
3.1	L293D Motor driver module	20
3.2	Flame sensor module	22
3.3	Internal architecture of dc motor	21
3.4	5v water pump	25
3.5	Servo Motor	26
3.6	Rotating mechanism of servo motor	27
3.7	Arduino UNO board	28
3.8	Pin diagram of Atmega328	30
3.9	Architecture of AVR	32
3.10	Line Diagram of Rocker Bogie Mechanism	35
3.11	Three Dimensional view of Rocker Bogie Mechanism	36
3.12	RBM on uneven path	37
3.13	Screenshort of Arduino IDE	42
3.14	Program compiling using arduino IDE.	43
3.16	Selecting the port	44
3.17	Uploading program to the arduino	44
3.18	Circuit diagram for analyzing characteristics of PV arrays	45

## **LIST OF FIGURES**

<b>Figure No.:</b>	<b>Name of the figures</b>	<b>Page No.:</b>
<b>3.19</b>	Functional description	<b>56</b>
<b>3.20</b>	Flow chart	<b>59</b>
<b>4.1</b>	Microcontroller –flame sensor interfacing	<b>60</b>

## **LIST OF TABLE**

<b>Table No.:</b>	<b>Name of the table</b>	<b>Page No.:</b>
<b>3.14</b>	Pin description table	<b>23</b>
<b>3.15</b>	Description of each pins of ATmega328	<b>31</b>

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1. INTRODUCTION**

Cultural property management is entrusted with the responsibility of protecting and preserving an institution's buildings, collections, operations and occupants. Constant attention is required to minimize adverse impact due to climate, pollution, theft, vandalism, insects, mold and fire. Because of the speed and totality of the destructive forces of fire, it constitutes one of the more serious threats. Vandalized or environmentally damaged structures can be repaired and stolen objects recovered. Items destroyed by fire, however, are gone forever. An uncontrolled fire can obliterate an entire room's contents within a few minutes and completely burn out a building in a couple of hours. Hence it has become very necessary to control and cease the fire to protect the Life and costlier things. For that we purposed to design and fabricate the fire-fighting robot. Autonomous robots can act on their own, independent of any controller. The basic idea is to program the robot to respond in a certain way to outside stimuli. The very simple bump-and-go robot is a good illustration of how this works. This sort of robot has a sensor to detect obstacles. When you turn the robot on, it zips along in a straight line. When it finally hits an obstacle, the impact is on sensors, i.e, sensors may get damaged. Using Ultrasonic sensor and programming logic, the robot is guided to turn right and move forward again, when the robot finds an obstacle in its way. In this way, the robot changes direction any time it encounters an obstacle. Advanced robots use more elaborate versions of this same idea. Roboticists create new programs and sensor systems to make robots more smarter and more perceptive. Today, robots can effectively navigate in a variety of environments.

Robots can be defined as machine resembling a human being but capable of performing complex assignments. In hazardous jobs like firefighting robots can be of significant service. Fire Fighting is an imaginary gameplay of firefighter rescuing the victims and stopping the fire as soon as possible. Many of the times wide reaching fire mishaps commence due to small fire flame leading to the much more vandalization.

The stated firefighting robot is competent of detecting the smoke raised in the air due to flame, with the help of smoke sensor MQ2. Likewise, presence of the fire can be detected by the robot with flame sensors intact on anterior of the prototype robot.

Fire detected gets douse with water from water tank mounted on the robot.

The robot firefighter is designed to look for fire in small houses of specific dimensions. An ideal firefighting robot is also capable of warn the service man about the outrage via SMS or a call. Water pump sprays water on the fire to stop it from further spreading. In addition to being able to be installed in homes, laboratories, stores, shops, etc., firefighting robot is easily portable and can be used once installed.

## **1.2 PROJECT OVERVIEW**

The project is designed to develop a fire fighting robot using Arduino uno . The robotic vehicle is loaded with water pump which is controlled by servos. An ATmega 328 microcontroller is used for the desired operation. At the transmitting end using commands are sent to the receiver to control the movement of the robot either to move forward, and left or right etc. At the receiving end tow motors are interfaced to the microcontroller where two of them are used for the movement of the vehicle and the one to position the robot. The ultrasonic sensor adequate range with obstacle detection, while the receiver driver module used to drive DC motors via motor driver IC for necessary work. A water tank along with water pump is mounted on the robot body and its operation is carried out from the microcontroller output through appropriate command from the transmitting end. The whole operation is controlled by an ATmega 328 microcontroller. A motor driver IC is interfaced to the microcontroller through which the controller drives the motors three flame sensors are fixed on robot chassis to sense the fire and to reach the destination to put off the fire.

## **1.3 COMPONENTS OVERVIEW**

**This system uses the following components.**

### **1.3.1 Microcontroller**

Microcontroller can be described as a computer embedded on a rather small circuit board. To describe the function of a microcontroller more precisely it is a single chip that can perform various calculations and task and send/receive signals from other devices via the available pins. Precisely what tasks and communication with the world it does, is what is governed by what instructions we give to the Microcontroller. It is this job of telling the chip what to do, is what we refer to as programming on it.

However, the microcontroller by itself, cannot accomplish much, it needs several external inputs, power, for one, a steady clock signal, for another. Also, the job of programming it has to be accomplished by an external circuit. So typically, a microcontroller is used along with a circuit which provides these things to it; this combination is called a microcontroller board. The Arduino Uno that you have received is one such microcontroller board. The actual microcontroller at its heart is the chip called Atmega328. The advantages that Arduino offers over other microcontroller boards are largely in terms of reliability of the circuit hardware as well as the ease of programming and using it.

### **1.3.2 Power Supply**

7805 is a voltage regulation IC which is used to supply 5V Direct current to the microcontroller

### **1.3.3 Motor Driver IC**

L293D is a dual H-bridge motor driver integrated circuit (IC). They are use to control



they the 4 motor used in project. There are 2 motor driver IC used in the project one to control front motor and other for rear motors.

### **1.3.4 Computer Interface**

Finally, this project uses IDE compiler for interfacing the arduino with a PC. This interface is used to setup and compile the Arduino.

# **CHAPTER 2**

## **LITERATURE SURVEY**

### **2.1 LITERATURE REVIEW**

The automatic fire fighting robot consists of hardware and software design. The hardware part deals with the mechanical and construction design, electric and electronic circuitry. The software parts deals with the programming . Fire-fighting robots can take many forms, but typically consist of a robotic vehicle with a fire-extinguishing payload, such as a water cannon, foam sprayer, or CO<sub>2</sub> gas dispenser.

These robots can be operated remotely or autonomously, and are usually equipped with cameras and sensors to help them navigate and detect fires. Some robots are even capable of performing basic search-and-rescue operations, such as locating victims and carrying them to safety.

This robot integrates the idea of natural fire detection and corresponding engine control. In order for the robot to be controlled bidirectionally, it makes use of the engine driver. With the assistance of a microcontroller, each guidance for controlling movement is given to the robot . The use of fire-fighting robots has several advantages over traditional firefighting techniques. For example, robots can operate in hazardous environments without putting human firefighters at risk. They can also be used to monitor a fire in real-time, allowing firefighters to better contain and extinguish the blaze. Finally, robots can be used in situations where human access is not possible, such as in collapsed buildings or in remote locations. The paper aims to motivate the robotics community to develop a real-world application based on what it can accomplish . In addition to being able to be installed in homes, laboratories, stores, shops, etc., firefighting robot is easily portable and can be used once installed .

## 2.2 Problem Statement

As explained in the introduction chapter, the realization of complete potential of the sensors and the wired medium in information transfer is the major issue that the following thesis of the following project deals with.

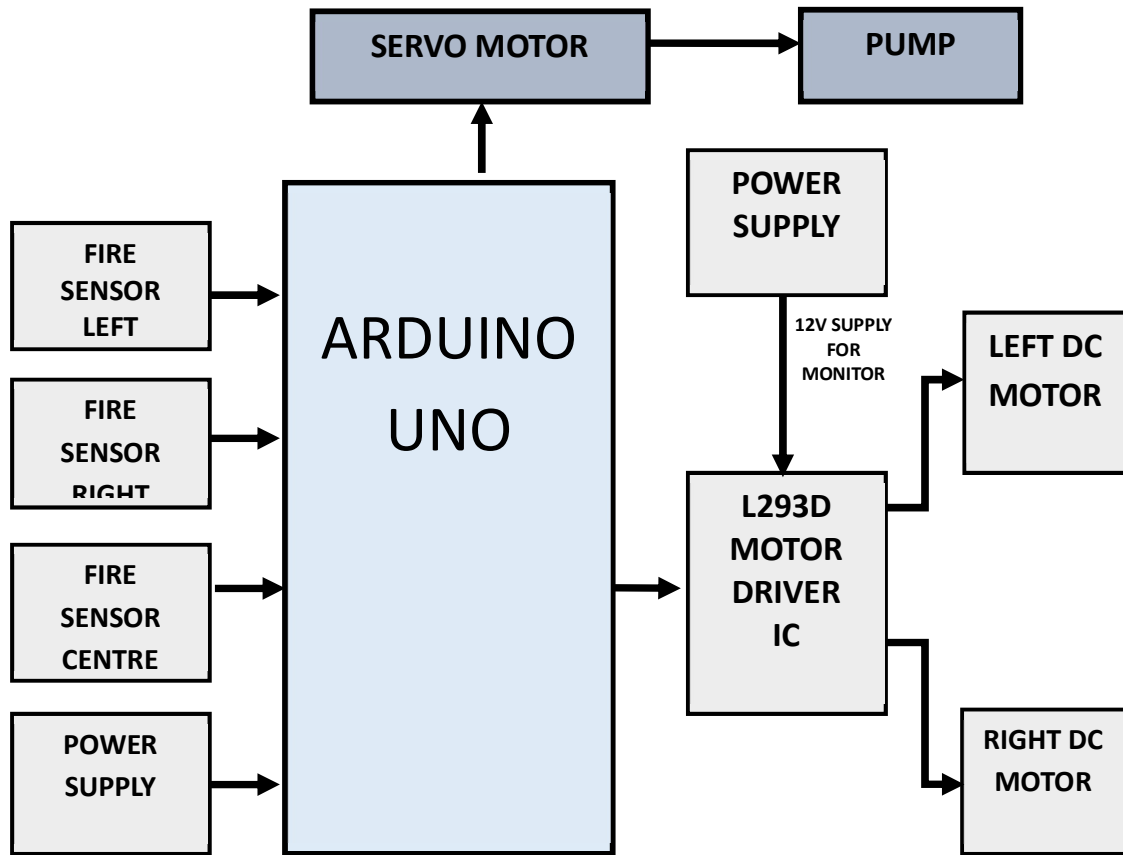


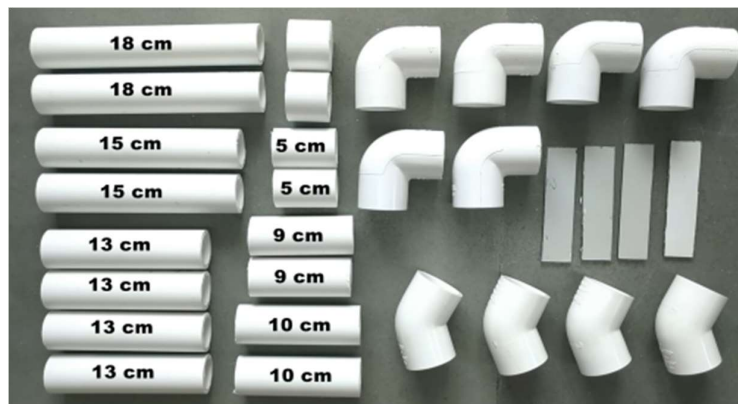
Fig 2.1: Overview of system

From Fig 2.1, there are at least five interfacing circuits, L293d driver module, Arduino-uno with Microcontroller, flame sensors, servo motor and 5v pump here arduino uno acts a heart of our project, in the above block diagram we can see that there are three flame sensors and ultrasonic sensor which acts as input interface to the microcontroller and servomotor, pump, driver module acts a output interface to the microcontroller, here the input and output interface can be indicated with the arrow lines with the respective the microcontroller performs with the respective commands and delay which is programmed on arduino software.



### 3.2 Hardware used

#### 1 .PVC Pipes with 90 and 145 degree connectors



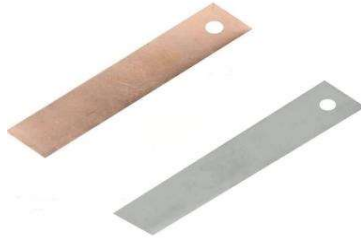
#### 2 .Nuts and Bolts



#### 3. Water Tank and Pipe



#### 4. Metal Strips



#### 5. Metal Clamps



#### 6. Nozzle for accurate speed of water flow



## 7.Drill Machine



### 3.2.1 L293D Driver module

The Motor Driver is a module for motors that allows you to control the working speed and direction of two motors simultaneously. This Motor Driver is designed and developed based on L293D IC. L293D is a 16 Pin Motor Driver IC. This is designed to provide bidirectional drive currents at voltages from 5 V to 36 V.

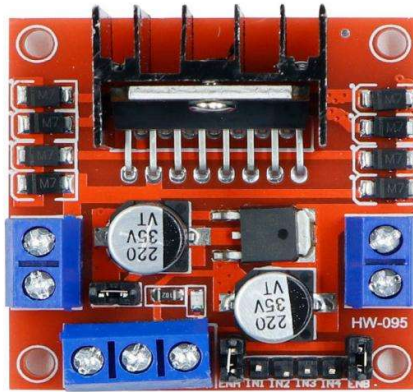


Figure 3.1: L293D motor driver module

L293D is a dual H-bridge motor driver integrated circuit (IC). Motor drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motors. L293D contains two inbuilt H-bridgedriver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively. Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled. As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state.



### **3.2.1.1 Features of L293D driver**

#### **module**

##### **Hardware features**

- ☐ can be used to run Two DC motors with the same IC.
- ☐ Speed and Direction control is possible
- ☐ Motor voltage Vcc2 (Vs): 4.5V to 36V
- ☐ Maximum Peak motor current: 1.2A
- ☐ Maximum Continuous Motor Current: 600mA
- ☐ Supply Voltage to Vcc1(vss): 4.5V to 7V
- ☐ Transition time: 300ns (at 5V and 24V)
- ☐ Automatic Thermal shutdown is available
- ☐ Available in 16-pin DIP, TSSOP, SOIC packages

##### **Applications**

- Used to drive high current Motors using Digital Circuits
- Can be used to drive Stepper motors
- High current LED's can be driven
- Relay Driver module (Latching Relay is possible)

### 3.2.1.1 Pin description

The L293D driver module has 16 pins. They are as follows:

ENABLE :

When enable is pulled low, the module is disabled which means the module will not turn on and it fails to drive motors. When enable is left open or connected to 3.3V, the module is enabled i.e the module remains on and driving of motors also takes place.

VCC:

GND:

Supply voltage 3.3v to 5v

Ground pin

INPUT & OUTPUT:

These two pins acts as an input and output interface for communication.

Pin No	Function	Name
1	Enable pin for Motor 1; active high	Enable 1,2
2	Input 1 for Motor 1	Input 1
3	Output 1 for Motor 1	Output 1
4	Ground (0V)	Ground
5	Ground (0V)	Ground
6	Output 2 for Motor 1	Output 2
7	Input 2 for Motor 1	Input 2
8	Supply voltage for Motors; 9-12V (up to 36V)	Vcc <sub>2</sub>
9	Enable pin for Motor 2; active high	Enable 3,4
10	Input 1 for Motor 2	Input 3
11	Output 1 for Motor 2	Output 3
12	Ground (0V)	Ground
13	Ground (0V)	Ground
14	Output 2 for Motor 2	Output 4
15	Input 2 for Motor 2	Input 4
16	Supply voltage; 5V (up to 36V)	Vcc <sub>1</sub>

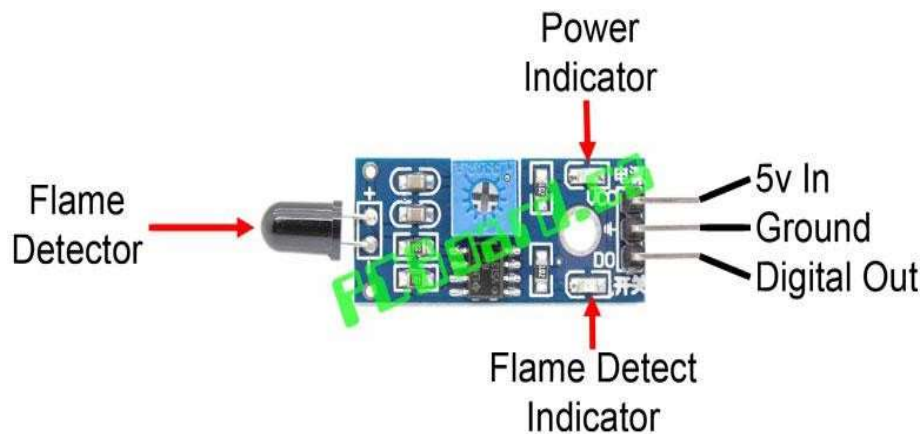
**Table no 3.1. pin description table**

### 3.2.2 Flame Sensor Module

A flame sensor module that consists of a flame sensor (IR receiver), resistor, capacitor, potentiometer, and comparator LM393 in an integrated circuit. It can detect

infrared light with a wavelength ranging from 700nm to 1000nm. The far-infrared flame probe converts the light detected in the form of infrared light into current changes. Sensitivity is adjusted through the onboard variable resistor with a detection angle of 60 degrees.

Working voltage is between 3.3v and 5.2v DC, with a digital output to indicate the presence of a signal. Sensing is conditioned by an LM393 comparator.



**Figure 3.2: flame sensor module**

### 3.2.3 DC Motor:

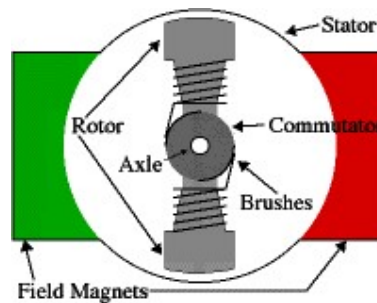
Motors convert electrical energy into mechanical energy. A DC motor is an electric motor that runs on direct current (DC) electricity.

In any electric motor, operation is based on simple electromagnetism. A current-carrying conductor generates a magnetic field; when this is then placed in an external magnetic field, it will experience a force proportional to the current in the conductor, and to the strength of the external magnetic field. As you are well aware of from playing with magnets as a kid, opposite (North and South) polarities attract, while like polarities (North and North, South and South) repel. The internal configuration

of a DC motor is designed to harness the magnetic interaction between a current-carrying conductor and an external magnetic field to generate rotational motion.

Direct current (DC) motors are widely used to generate motion in a variety of products. Permanent magnet DC (direct current) motors are enjoying increasing popularity in applications requiring compact size, high torque, high efficiency, and low power consumption.

In a brushed DC motor, the brushes make mechanical contact with a set of electrical contacts provided on a commutator secured to an armature, forming an electrical circuit between the DC electrical source and coil windings on the armature. As the armature rotates on an axis, the stationary brushes come into contact with different sections of the rotating commutator.



**Fig 3.3 Internal architecture of dc motor**

Permanent magnet DC motors utilize two or more brushes contacting a commutator which provides the direct current flow to the windings of the rotor, which in turn provide the desired magnetic repulsion/attraction with the permanent magnets located around the periphery of the motor.

The brushes are conventionally located in brush boxes and utilize a U-shaped spring which biases the brush into contact with the commutator. Permanent magnet brushless dc motors are widely used in a variety of applications due to their simplicity of design, high efficiency, and low noise. These motors operate by electronic

commutation of stator windings rather than the conventional mechanical commutation accomplished by the pressing engagement of brushes against a rotating commutator.

A brushless DC motor basically consists of a shaft, a rotor assembly equipped with one or more permanent magnets arranged on the shaft, and a stator assembly which incorporates a stator component and phase windings. Rotating magnetic fields are formed by the currents applied to the coils.

The rotator is formed of at least one permanent magnet surrounded by the stator, wherein the rotator rotates within the stator. Two bearings are mounted at an axial distance to each other on the shaft to support the rotor assembly and stator assembly relative to each other. To achieve electronic commutation, brushless dc motor designs usually include an electronic controller for controlling the excitation of the stator windings.

### **3.2.4 Water Pump**

The water pump is operated at 5v which can be interfaced with Arduino



**Fig 3.4 5v water pump**

### **3.2.5 Servo Motor**

A servo is a small DC motor with the following components added: some

gear reduction, a position sensor on the motor shaft, and an electronic circuit that controls the motor's operation. In other words, a servo is to a DC motor what the Arduino is the ATmega microcontroller---components and housing that make the motor easy to use. This will become abundantly clear when we work with unadorned DC motors next week.

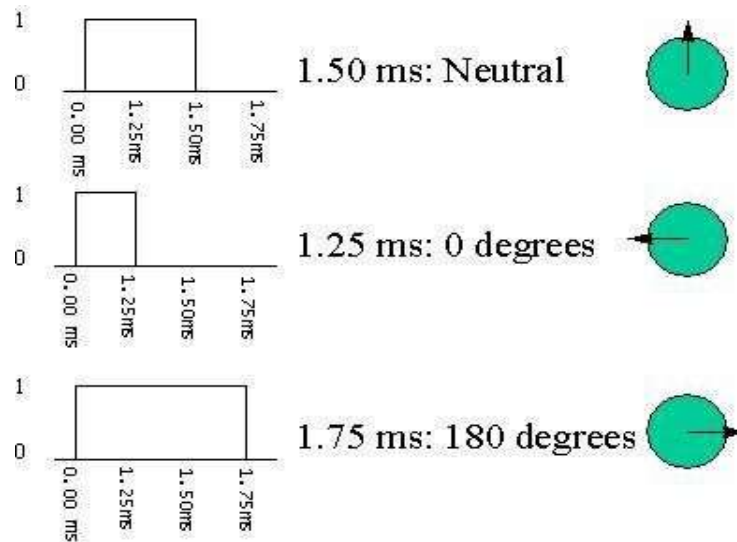
The gear reduction provided in a servo is large; the basic hobby servo has a 180:1 gear ratio. This means that the DC motor shaft must make 180 revolutions to produce 1 revolution of the servo shaft. This large gear ratio reduces the speed of the servo and proportionately increases its torque. What does this imply about small DC motors? Servo motors are typically used for angular positioning, such as in radio control airplanes. They have a movement range of 0 up to 180 degrees, but some extend up to 210 degrees. Typically, a potentiometer measures the position of the output shaft at all times so the controller can accurately place and maintain its position.



**Fig 3.5 Servo Motor**

PPM uses 1 to 2ms out of a 20ms timeperiod to encode its information. The servo expects to see a pulse every 20 milliseconds (.02 seconds). The length of the pulse will determine how far the motor turns. A 1.5 millisecond pulse will make the motor turn to the 90 degree position (often called the neutral

position). If the pulse is shorter than 1.5 ms, then the motor will turn the shaft to closer to 0 degrees. If the pulse is longer than 1.5ms, the shaft turns closer to 180 degrees. The amount of power applied to the motor is proportional to the distance it needs to travel. So, if the shaft needs to turn a large distance, the motor will run at full speed. If it needs to turn only a small amount, the motor will run at a slower speed.

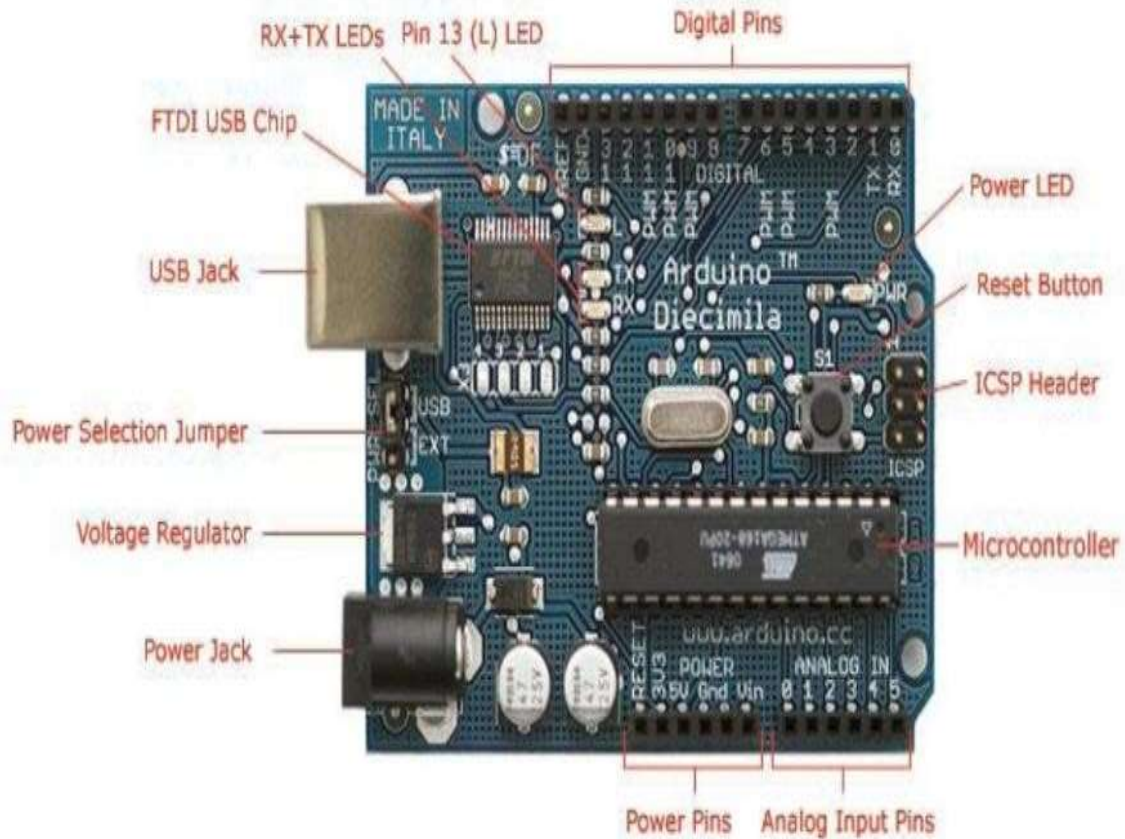


**Fig 3.6 Rotating mechanism of servo motor**

### **3.2.6 MICROCONTROLLER ATMEGA 328**

The Atmel 8-bit AVR RISC-based microcontroller combines 32 KB ISP flash memory with read-while-write capabilities, 1 KB EEPROM, 2 KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes.





**Fig 3.7 Arduino UNO board**

The device operates between 1.8-5.5 volts. The device achieves throughputs approaching 1 MIPS.

### **3.2.6.1. Applications**

Today the ATmega328 is commonly used in many projects and autonomous systems where a simple, low-powered, low-cost micro-controller is needed. Perhaps the most common implementation of this chip is on the popular Arduino development platform, namely the Arduino Uno and Arduino Nano models.



### 3.2.6.2 Features

- 28-pin AVR Microcontroller
- Flash Program Memory: 32 kbytes
- EEPROM Data Memory: 1 kbytes
- SRAM Data Memory: 2 kbytes
- I/O Pins: 23
- Timers: Two 8-bit / One 16-bit
- A/D Converter: 10-bit Six Channel
- PWM: Six Channels
- RTC: Yes with Separate Oscillator
- MSSP: SPI and I<sup>2</sup>C Master and Slave Support
- USART: Yes
- External Oscillator: up to 20MHz

The Atmega328 is a very popular microcontroller chip produced by Atmel. It is an 8-bit microcontroller that has 32K of flash memory, 1K of EEPROM, and 2K of internal SRAM.

The Atmega328 is one of the microcontroller chips that are used with the popular Arduino Duemilanove boards. The Arduino Duemilanove board comes with either 1 of 2 microcontroller chips, the Atmega168 or the Atmega328. Of these 2, the Atmega328 is the upgraded, more advanced chip. Unlike the Atmega168 which has 16K of flash program memory and 512 bytes of internal SRAM, the Atmega328 has 32K of flash program memory and 2K of Internal SRAM.

The Atmega328 has 28 pins, It has 14 digital I/O pins, of which 6 can be used as PWM outputs and 6 analog input pins. These I/O pins account for 20 of the pins.

## PIN DIAGRAM OF ATMEGA328

### ATMEGA 328

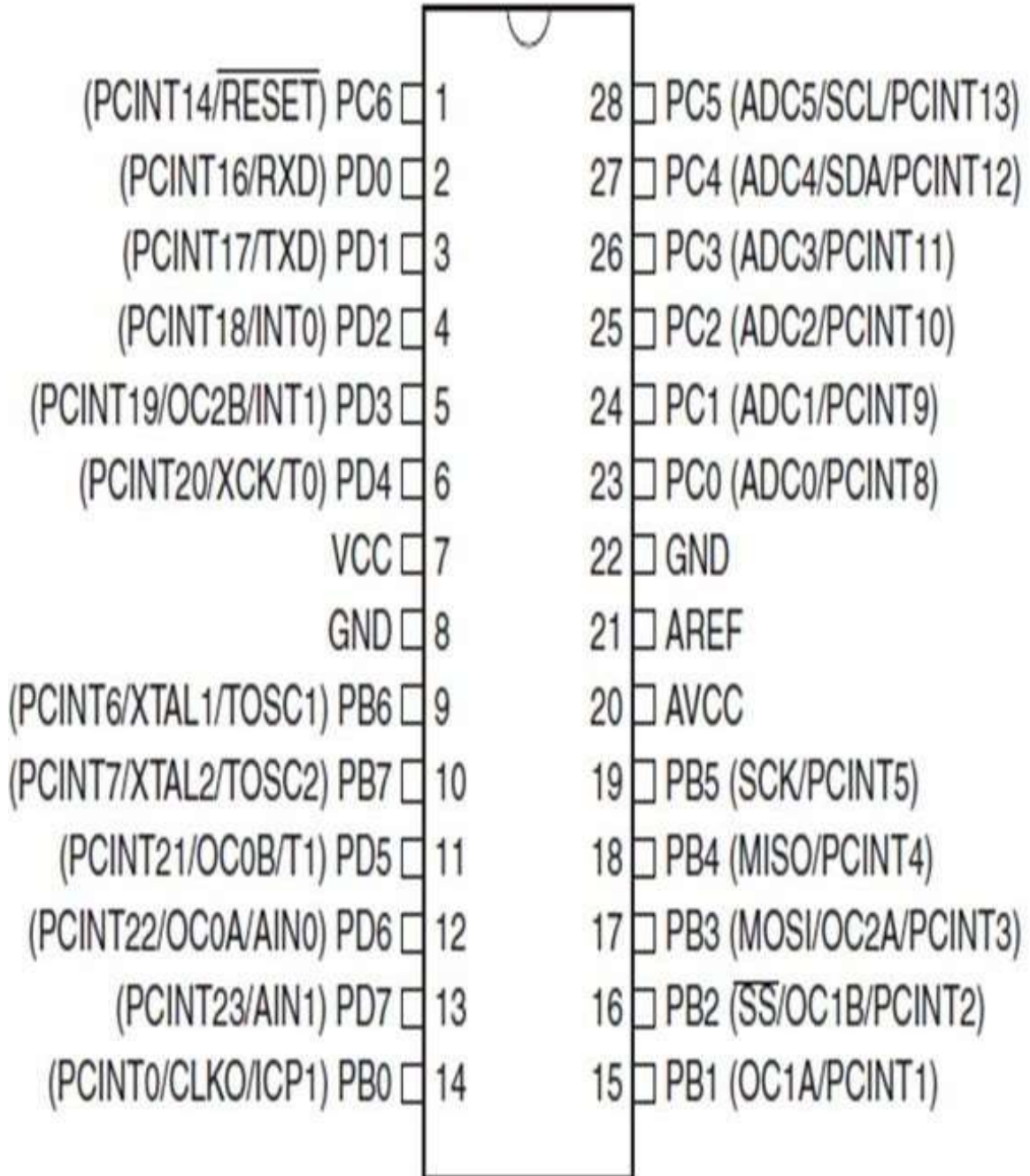


Figure 3.8: Pin diagram of Atmega328

**Table 3.2: Description of each pins of ATmega328**

Pin Number	Description	Function
1	PC6	Reset
2	PD0	Digital Pin (RX)
3	PD1	Digital Pin (TX)
4	PD2	Digital Pin
5	PD3	Digital Pin (PWM)
6	PD4	Digital Pin
7	Vcc	Positive Voltage (Power)
8	GND	Ground
9	XTAL 1	Crystal Oscillator
10	XTAL 2	Crystal Oscillator
11	PD5	Digital Pin (PWM)
12	PD6	Digital Pin (PWM)
13	PD7	Digital Pin
14	PB0	Digital Pin
15	PB1	Digital Pin (PWM)
16	PB2	Digital Pin (PWM)
17	PB3	Digital Pin (PWM)
18	PB4	Digital Pin
19	PB5	Digital Pin
20	AVcc	Positive voltage for ADC (power)
21	AREF	Reference Voltage
22	GND	Ground
23	PC0	Analog Input
24	PC1	Analog Input
25	PC2	Analog Input
26	PC3	Analog Input
27	PC4	Analog Input
28	PC5	Analog Input

As stated before, 20 of the pins function as I/O ports. This means they can function as an input to the circuit or as output. Whether they are input or output is set in the software. 14 of the pins are digital pins, of which 6 can function to give PWM output. 6 of the pins are for analog input/output. Two of the pins are for the crystal oscillator, this is to provide a clock pulse for the Atmega chip. A clock pulse is needed for synchronization so that communication can occur in synchrony between the Atmega chip and a device that

connected to.

The Atmega328 chip has an analog-to-digital converter (ADC) inside of it. This must be or else the Atmega328 wouldn't be capable of interpreting analog signals. Because there is an ADC, the chip can interpret analog input, which is why the chip has 6 pins for analog input. The ADC has 3 pins set aside for it to function- AVCC, AREF, and GND. AVCC is the power supply, positive voltage, that for the ADC. The ADC needs its own power supply in order to work. GND is the power supply ground. AREF is the reference voltage that the ADC uses to convert an analog signal to its corresponding digital value. Analog voltages higher than the reference voltage will be assigned to a digital value of 1, while analog voltages below the reference voltage will be assigned the digital value of 0. Since the ADC for the Atmega328 is a 10-bit ADC, meaning it produces a 10-bit digital value, it converts an analog signal to its digital value, with the AREF value being a reference for which digital values are high or low. Thus, a portrait of an analog signal is shown by this digital value; thus, it is its digital correspondent value. The last pin is the RESET pin. This allows a program to be rerun and start over. And this sums up the pin out of an Atmega328 chip .

### 3.2.6.3. ARCHITECTURE

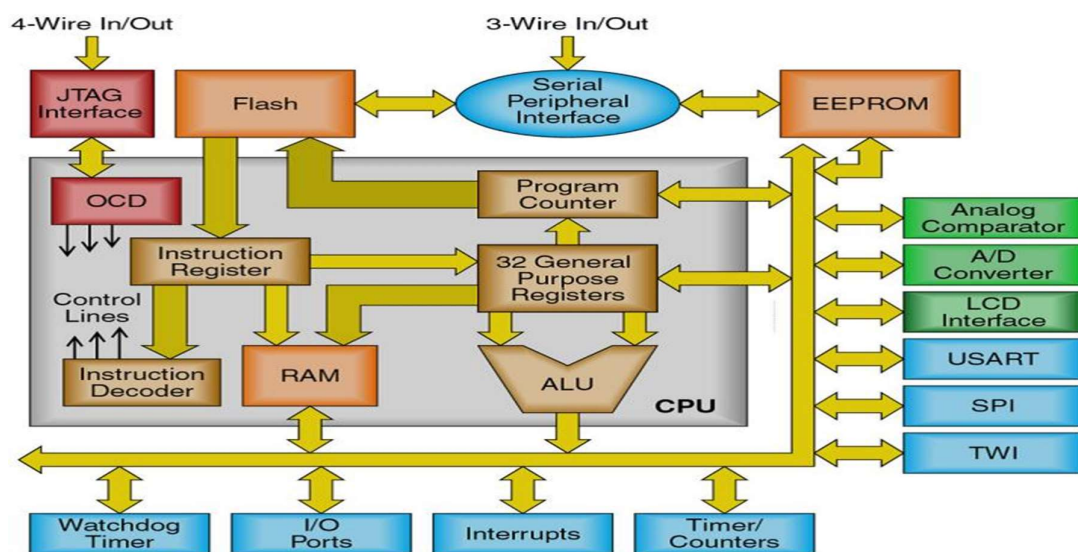


Figure 3.9: Architecture of AVR

## FEATURES OF AVR.

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 32K Bytes of In-System Self-Programmable Flash
- Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
- In-System Programming by On-chip Boot Program
- True Read-While-Write Operation
  - 1024 Bytes EEPROM
- Endurance: 100,000 Write/Erase Cycles
  - 2K Byte Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Pre scalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Pre scaler, Compare Mode, and Capture Mode
    - Real Time Counter with Separate Oscillator
    - Four PWM Channels
- 8-channel, 10-bit ADC
  - 8 Single-ended Channels

- 7 Differential Channels in TQFP Package Only
- 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega32L
  - 4.5 - 5.5V for ATmega32
- Speed Grades
  - 0 - 8 MHz for ATmega32L
  - 0 - 16 MHz for ATmega32
- Power Consumption at 1 MHz, 3V, 25°C for ATmega32L
  - Active: 1.1 mA
  - Idle Mode: 0.35 mA
  - Power-down Mode: < 1  $\mu$ A

## • BASIC TERMINOLOGIES IN ARDUINO

### Analog to Digital Converter (ADC)

- The process of Analog to digital conversion is shown in figure.



- The Arduino has 10 bits of Resolution when reading analog signals.
- $2^{10}=1024$  increments.
- Influence also by how fast you sample.

#### Pulse Width Modulation (PWM)

- The Arduino has 8bit of resolution, when outputting a signal using PWM.
- The range of output voltage is from 0 to 5 Volts.
- $2^8=255$  Increments
- Average of on/off(digital signals to make an average voltage),Duty cycle in 100% of 5Volts.

#### 3. 8 LANGUAGE REFERENCES:

The Microcontroller on the board is programmed using the Arduino programming language (based on wiring) and the arduino development environment (based on processing).

### 3.3.Design of Rocker-Bogie Mechanism

The term “rocker” describes the rocking aspect of the larger links present each side of the suspension system and balance the bogie as these rockers are connected to each other and the vehicle chassis through a modified differential.

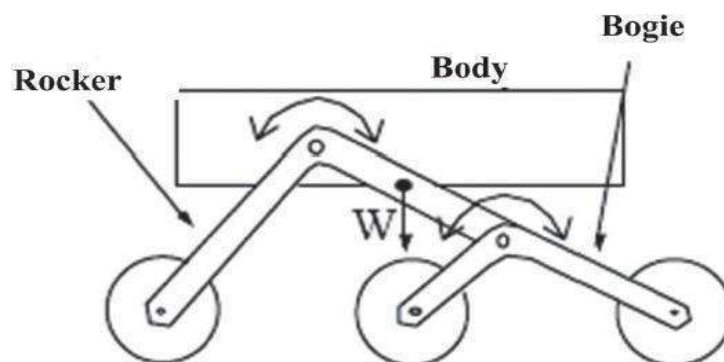


Figure 3.10 Line Diagram of Rocker Bogie Mechanism

In the system, “bogie” refers to the conjoining links that have a drive wheel attached at each end. Bogies were commonly used to bare loading as tracks of army tanks as idlers distributing the load over the terrain. Bogies were also quite commonly used on the trailers of semitrailer trucks as that very time the trucks will have to carry much heavier loa

As accordance with the motion to maintain centre of gravity of entire vehicle, when one rocker moves up-ward, the other goes down. The chassis plays vital role to maintain the average pitch angle of both rockers by allowing both rockers to move as per the situation

The physics of these rovers is quite complex. To design and control these analytical models of how the rover interacts with its environment are essential. Models are also needed for rover action planning. Simple mobility analysis of rocker-bogie vehicles have been developed and used for design evaluation in the available published works.

The rocker-bogie configuration is modeled as a planer system. Improving the performances of a simpler four wheel rover has also been explored

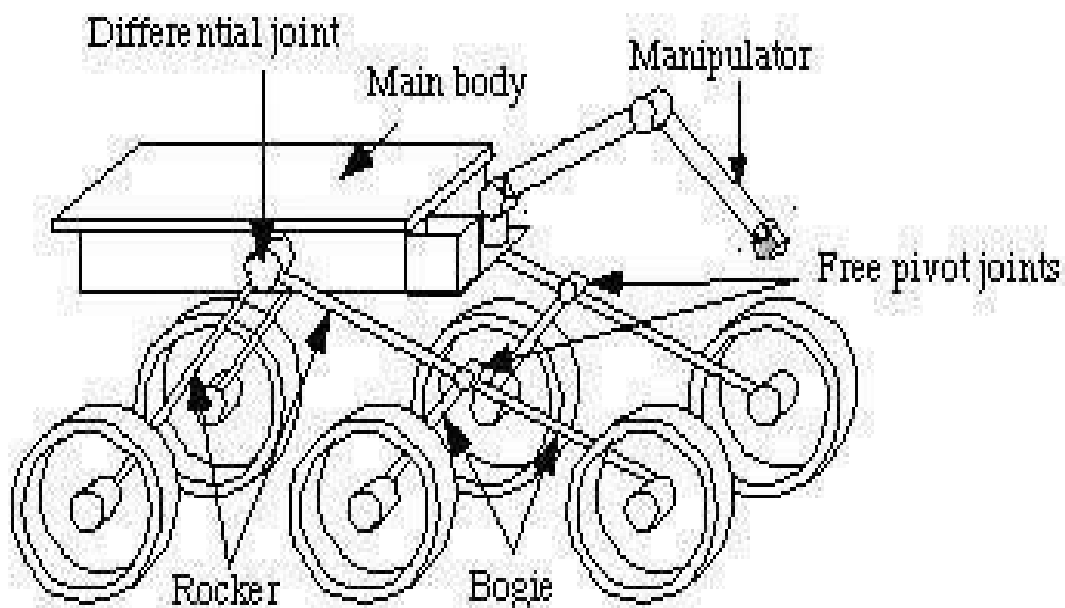


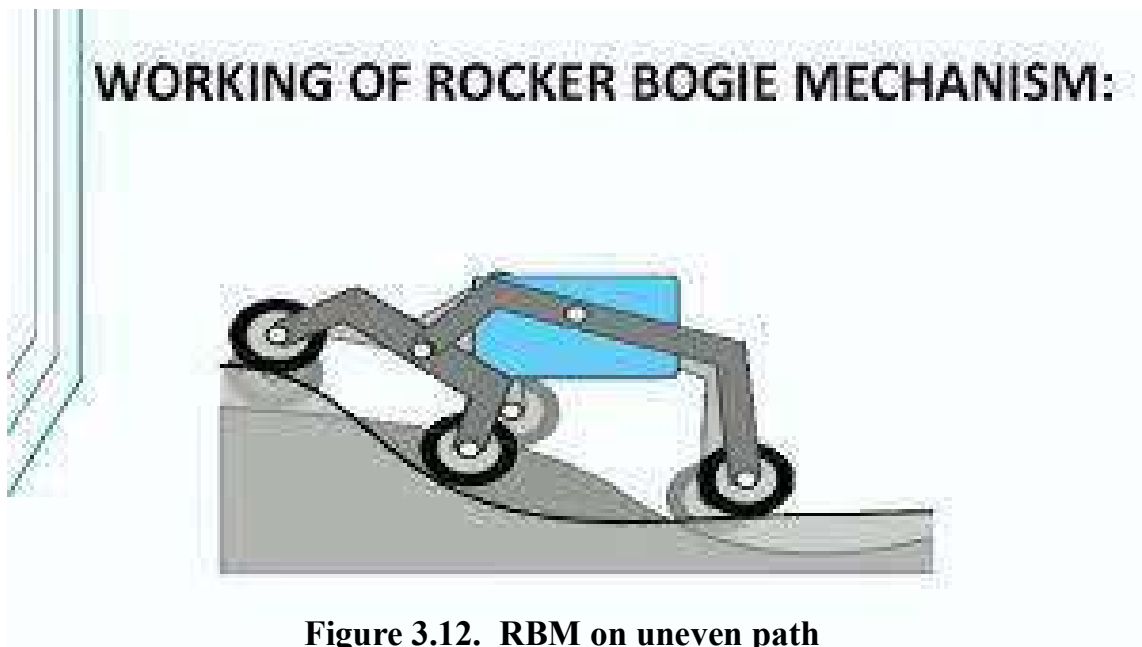
Figure 3.11. Three Dimensional view of Rocker Bogie Mechanism



### 3.3.1 ADVANTAGES

- The design incorporates independent motors for each wheel. There are no springs or axles, making the design simpler and more reliable.
- Rocker Bogie Suspension can withstand a tilt of at least  $50^{\circ}$  in any direction without overturning, which is the biggest advantage of heavy loaded vehicle.
- It can move in harsh environment
- It can work in place which are beyond human reach

Rocker Bogie consisting of no spring and stub axle in each wheel which allows the chasis to climb over any obstacle such as rocks, ditches, sands etc. that are upto double the wheels diameter in size while keeping all the wheels on ground for maximum time.



**Figure 3.12. RBM on uneven path**

### 3.3.2. FUTURE SCOPE

- It can be useful in space mission too, recently it is used in Mars Rover. This mechanism takes consideration on unevenness of the surface it is driving on.
- This rover has larger wheel as compared to obstacles, It can easily operate over most of the Martian rocks.

- It's future application will be assist astronauts during space operations, it will act as a path finder too.
- It is also used in coal mines, act as a spy robot and in military operation too.

### **3.3.3. SELECTION OF MATERIAL**

Selection of material is an important step in designing of any component

The main advantages of material selection are :

It increases the reliability of product .

It reduces the cost of product.

It can also optimize the weight of product.

## **3.4 .SOFTWARE REQUIREMENTS**

### **3.4.1.Embedded C**

Embedded C is a set of language extensions for the C Programming language by the C Standards committee to address commonality issues that exist between C extensions for different embedded systems. Historically, embedded C programming requires nonstandard extensions to the C language in order to support exotic features such as fixed- point arithmetic, multiple distinct memory banks, and basic I/O operations.

### **3.4.2. Difference between C and Embedded C**

Though C and embedded C appear different and are used in different contexts, they have more similarities than the differences. Most of the constructs are same; the difference lies in their applications .

C is used for desktop computers, while embedded C is for microcontroller

based applications. C takes more resources of a desktop PC like memory, OS, etc. while programming on desktop systems what embedded C cannot. Embedded C has to use the limited resources (RAM, ROM, I/O's) on an embedded processor. Thus, program code must fit into the available program memory. If code exceeds the limit, the system is likely to crash.

Compilers for C (ANSI C) typically generate OS dependent executable files. Embedded C requires compilers to create files to be downloaded to the microcontrollers/microprocessors where it needs to run. Embedded compilers give access to all resources which is not provided in compilers for desktop computer applications.

Embedded systems often have the real-time constraints, which is usually not there with desktop computer applications.

Embedded systems often do not have a console, which is available in case of desktop applications.

The C programming language is perhaps the most popular programming language for programming embedded systems. C continues to be a very popular language for micro- controller developers/programmers due to the code efficiency and reduced overhead and development time. C offers low-level control and is considered more readable than assembly language which is a little difficult to understand. Assembly language requires more code writing, whereas C is easy to understand and requires less coding. Plus, using C increases portability, since C code can be compiled for different types of processors. We can program microcontrollers using Atmel Atmega328, AVR or PIC.

Here by developing the programs as per the electronic hardware using Atmel Atmega328 micro controller. For the operations like: blink led, increment decrement counters, token displays etc.

Most C programmers are spoiled because they program in environments where not

only there is a standard library implementation, but there are frequently a number of other libraries available for use. The cold fact is, that in embedded systems, there rarely are many of the libraries that programmers have grown used to, but occasionally an embedded system might not have a complete standard library, if there is a standard library at all. Few embedded systems have capability for dynamic linking, so if standard library functions are to be available at all, they often need to be directly linked into the executable. Oftentimes, because of space concerns, it is not possible to link in an entire library file, and programmers are often forced to "brew their own" standard c library implementations if they want to use them at all. While some libraries are bulky and not well suited for use on microcontrollers, many development systems still include the standard libraries which are the most common for C programmers.

C remains a very popular language for micro-controller developers due to the code efficiency and reduced overhead and development time. C offers low-level control and is considered more readable than assembly. Many free C compilers are available for a wide variety of development platforms. The compilers are part of an IDEs with ICD support, breakpoints, single-stepping and an assembly window. The performance of C compilers has improved considerably in recent years, and they are claimed to be more or less as good as assembly, depending on who you ask. Most tools now offer options for customizing the compiler optimization. Additionally, using C increases portability, since C code can be compiled for different types of processors .

### **3.4.3. Software**

The software used by the arduino is Arduino IDE. The Arduino IDE is a cross- platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring project.



It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit make files or run programs on a command-line interface. Although building on command-line is possible if required with some third-party tools such as Ino.

The Arduino IDE comes with a C/C++ library called "Wiring" (from the project of the same name), which makes many common input/output operations much easier. Arduino programs are written in C/C++, although users only need define two functions to make a runnable program:

- `setup()` – a function run once at the start of a program that can initialize settings
- `loop()` – a function called repeatedly until the board powers off

A typical first program for a microcontroller simply blinks a LED on and off. In the Arduino environment, the user might write a program like this:

```

#define LED_PIN 13

void setup (){
  pinMode(LED_PIN, OUTPUT);// enable pin 13 for digital output
}

void loop (){
  digitalWrite(LED_PIN, HIGH);// turn on the LED
  delay(1000);// wait one second (1000 milliseconds)
}

```



```

sketch_jun22a
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
String inData;
void setup() {
  Serial.begin(9600);
  lcd.begin(16,2);
  lcd.print("Welcome!");
  delay(3000);
  lcd.clear();
}
void loop() {
  int i=0;
  char commandbuffer[100];
  if(Serial.available()){
    delay(100);
    while(Serial.available() && i<99) {
      commandbuffer[i++] = Serial.read();
    }
    commandbuffer[i++] = '\0';
  }
  if(i>0)
  Serial.println((char*)commandbuffer);
  lcd.print((char*)commandbuffer);
  delay(100);
  lcd.clear();
}

```

Figure 3.13: A Screenshot of Arduino IDE

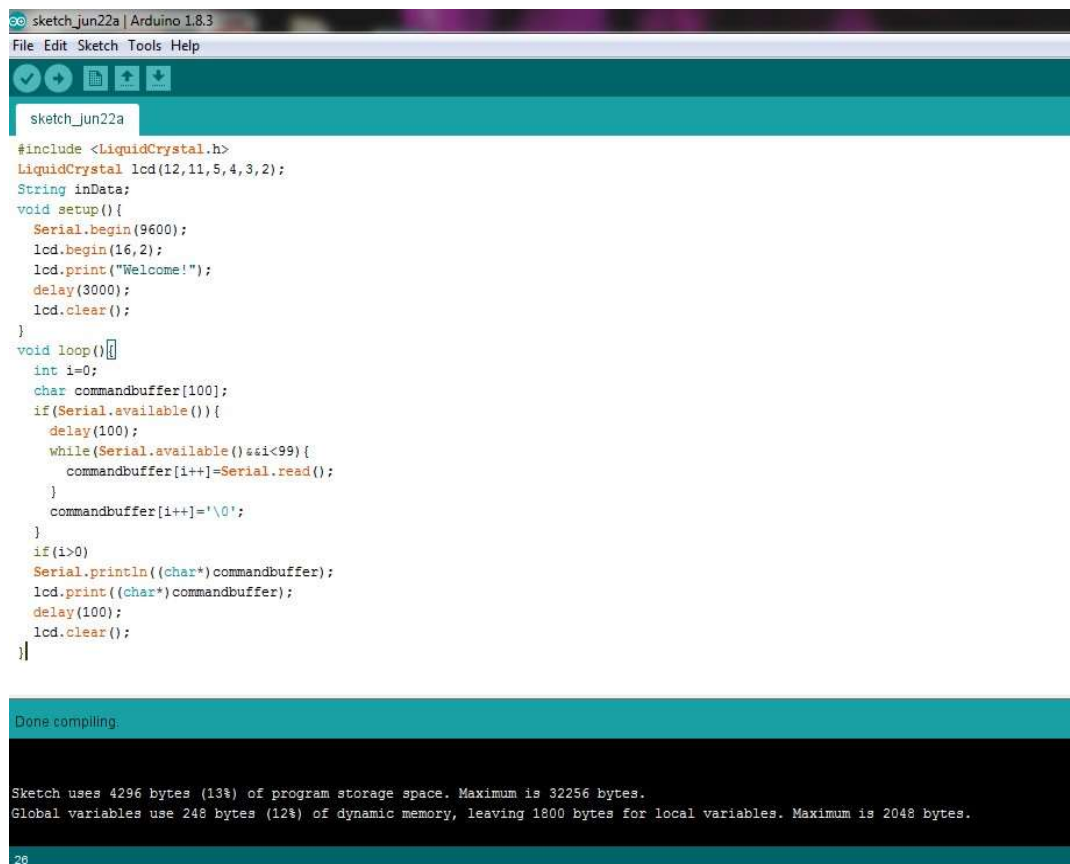
For the above code to work correctly, the positive side of the LED must be connected to pin 13 and the negative side of the LED must be connected to ground. The above code would not be seen by a standard C++ compiler as a valid program,

so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple main() function at the bottom, to make it a valid C++ program.

The Arduino IDE uses the GNU tool chain and AVR Libc to compile programs, and uses AVR dude to upload programs to the board.

For educational purposes there is third party graphical development environment called Mini blog available under a different open source license.

## PROGRAM COMPILING



```
sketch_jun22a | Arduino 1.8.3
File Edit Sketch Tools Help

sketch_jun22a

#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
String inData;
void setup(){
  Serial.begin(9600);
  lcd.begin(16,2);
  lcd.print("Welcome!");
  delay(3000);
  lcd.clear();
}
void loop(){
  int i=0;
  char commandbuffer[100];
  if(Serial.available()){
    delay(100);
    while(Serial.available() && i<99){
      commandbuffer[i++]=Serial.read();
    }
    commandbuffer[i++]='\0';
  }
  if(i>0)
  Serial.println((char*)commandbuffer);
  lcd.print((char*)commandbuffer);
  delay(100);
  lcd.clear();
}

Done compiling.

Sketch uses 4296 bytes (13%) of program storage space. Maximum is 32256 bytes.
Global variables use 248 bytes (12%) of dynamic memory, leaving 1800 bytes for local variables. Maximum is 2048 bytes.

26
```

Figure 3.14: Program compiling using arduino IDE.

## SELECTING BOARD

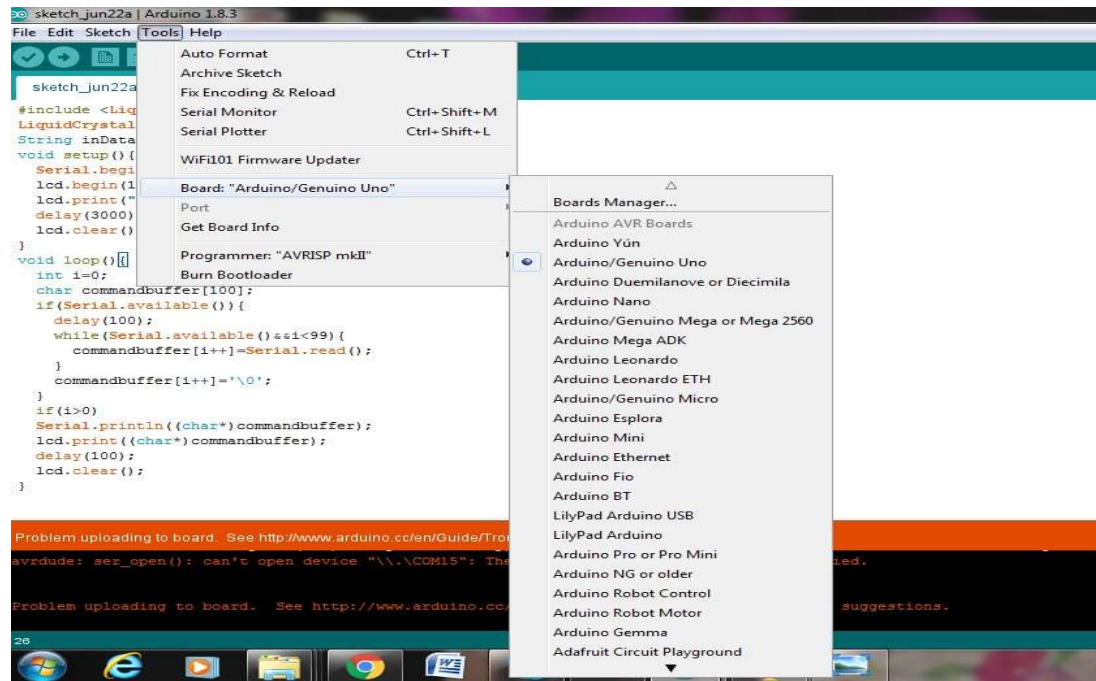


Figure 3.15: Selecting the board from Tools menu

## SELECTING PORT

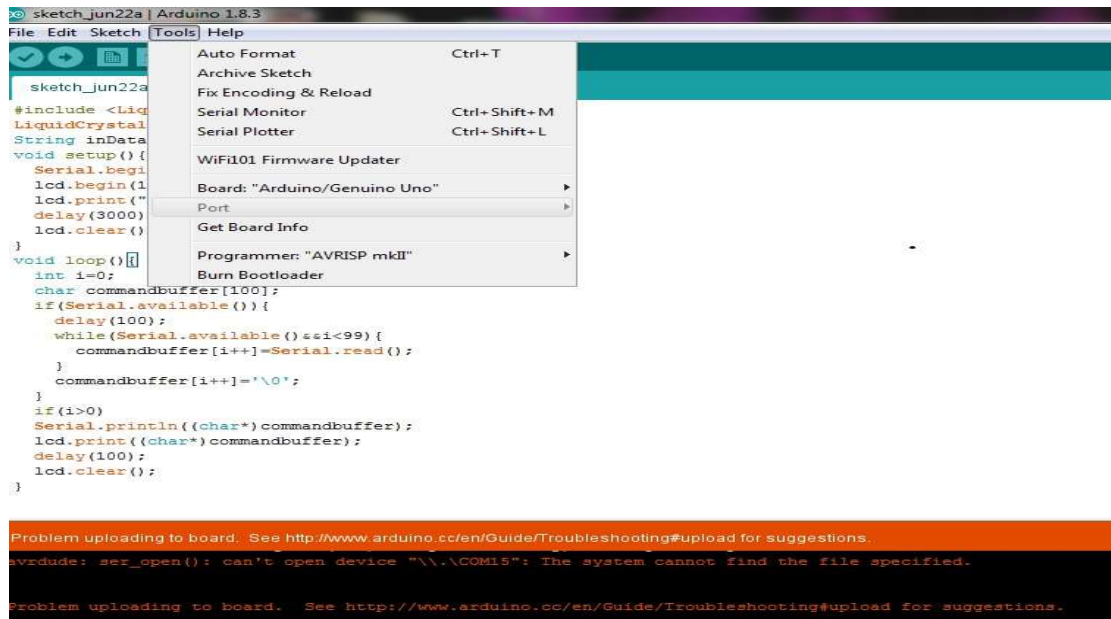


Figure 3.16: Selecting the port



## UPLOADING PROGRAM

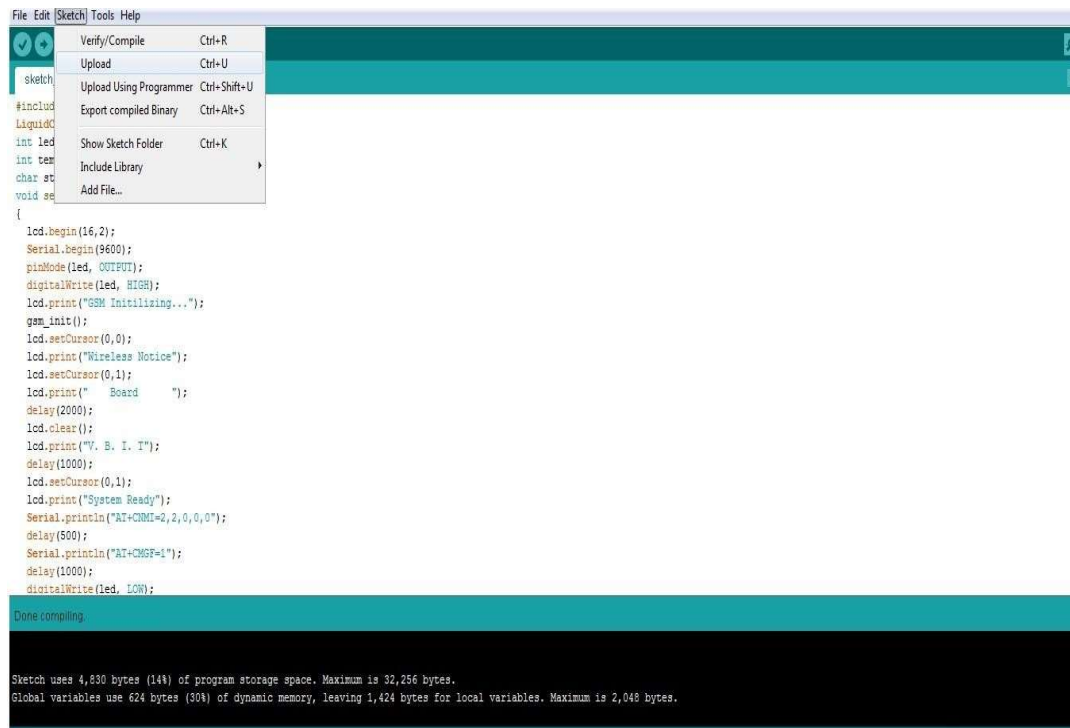


Figure 3.17: Uploading program to the arduino.

### 3.4.4. Language Reference

Arduino programs can be divided in three main parts: structure, values (variables and constants), and functions.

Available data types in ARDUINO IDE are

- void
- boolean
- char ( 0 – 255)
- byte - 8 bit data ( 0 – 255)
- int - 16-bit data (32,767 - -32,768)
- long – 32 bit data (2,147,483,647 to -2,147,483,648)

- float
- double
- string - char array
- String - object
- array

## AT COMMANDS

AT commands are used to control MODEMs. AT is the abbreviation for Attention. These commands come from Hayes commands that were used by the Hayes smart modems. The Hayes commands started with AT to indicate the attention from the MODEM. The dial up and wireless MODEMs (devices that involve machine to machine communication) need AT commands to interact with a computer. These include the Hayes command set as a subset, along with other extended AT commands.

AT commands with a mobile phone can be used to access following information and services:

1. Information and configuration pertaining to mobile device or Bluetooth module.
2. SMS services.
3. MMS services.
4. Fax services.
5. Data and Voice link over mobile network.

The Hayes subset commands are called the basic commands and the commands specific to a Bluetooth network are called extended AT commands.

## 3.5 ARDUINO UNO FINAL CODE

**We have uses two Arduino uno**

### 1. CODE OF BLUETOOTH MODULE

```
#define in1 5 //L298n Motor Driver pins.

#define in2 6

#define in3 10

#define in4 11

#define LED 13

int command; //Int to store app command state.

int Speed = 204; // 0 - 255.

int Speedsec;

int buttonState = 0;

int lastButtonState = 0;

int Turnradius = 0; //Set the radius of a turn, 0 - 255 Note:the robot will malfunction if this is higher
    than int Speed.

int brakeTime = 45;

int brkonoff = 1; //1 for the electronic braking system, 0 for normal.

void setup() {

    pinMode(in1, OUTPUT);

    pinMode(in2, OUTPUT);

    pinMode(in3, OUTPUT);

    pinMode(in4, OUTPUT);

    pinMode(LED, OUTPUT); //Set the LED pin.

    Serial.begin(9600); //Set the baud rate to your Bluetooth module.

}

void loop() {

    if (Serial.available() > 0) {

        command = Serial.read();

        Stop(); //Initialize with motors stoped.

        switch (command) {
```

```
case 'F':  
    forward();  
    break;  
case 'B':  
    back();  
    break;  
case 'L':  
    left();  
    break;  
case 'R':  
    right();  
    break;  
case 'G':  
    forwardleft();  
    break;  
case 'I':  
    forwardright();  
    break;  
case 'H':  
    backleft();  
    break;  
case 'J':  
    backright();  
    break;  
case '0':  
    Speed = 100;  
    break;  
case '1':  
    Speed = 140;  
    break;  
case '2':
```

```

    Speed = 153;

    break;

case '3':

    Speed = 165;

    break;

case '4':

    Speed = 178;

    break;

case '5':

    Speed = 191;

    break;

case '6':

    Speed = 204;

    break;

case '7':

    Speed = 216;

    break;

case '8':

    Speed = 229;

    break;

case '9':

    Speed = 242;

    break;

case 'q':

    Speed = 255;

    break;

}

Speedsec = Turnradius;

if (brkonoff == 1) {

    brakeOn();

} else {

```

```

        brakeOff();
    }
}

void forward() {
    analogWrite(in1, Speed);
    analogWrite(in3, Speed);
}

void back() {
    analogWrite(in2, Speed);
    analogWrite(in4, Speed);
}

void left() {
    analogWrite(in3, Speed);
    analogWrite(in2, Speed);
}

void right() {
    analogWrite(in4, Speed);
    analogWrite(in1, Speed);
}

void forwardleft() {
    analogWrite(in1, Speedsec);
    analogWrite(in3, Speed);
}

void forwardright() {
    analogWrite(in1, Speed);
    analogWrite(in3, Speedsec);
}

```

```

}

void backright() {
    analogWrite(in2, Speed);
    analogWrite(in4, Speedsec);
}

void backleft() {
    analogWrite(in2, Speedsec);
    analogWrite(in4, Speed);
}

void Stop() {
    analogWrite(in1, 0);
    analogWrite(in2, 0);
    analogWrite(in3, 0);
    analogWrite(in4, 0);
}

void brakeOn() {
    //Here's the future use: an electronic braking system!

    // read the pushbutton input pin:
    buttonState = command;

    // compare the buttonState to its previous state
    if (buttonState != lastButtonState) {
        // if the state has changed, increment the counter
        if (buttonState == 'S') {
            if (lastButtonState != buttonState) {
                digitalWrite(in1, HIGH);
                digitalWrite(in2, HIGH);
                digitalWrite(in3, HIGH);
                digitalWrite(in4, HIGH);
                delay(brakeTime);
            }
        }
    }
}

```

```

    Stop();
}
}

// save the current state as the last state,

//for next time through the loop

lastButtonState = buttonState;

}

}

void brakeOff() {

}

```

## 2. CODE OF FLAME SENSOR

```

#define in3 7 //Motor2 L298 Pin in3

#define in4 6 //Motor2 L298 Pin in4
#define enB 5 //Enable2 L298 Pin enB
#define ir_R AO
#define ir_F A1
#define ir_L A2
#define servo A4
#define pump A5

int Speed = 160; // Write The Duty Cycle 0 to 255 Enable for Motor Speed|
int s1, s2, 53;

void setup() { // put your setup code here, to run once
  Serial.begin(9600); // start serial communication at 9600bps pinMode(ir_R, INPUT);//
declare fire sensor pin as input pinMode(ir_F, INPUT);// declare fire sensor pin as input
  pinMode(ir_L, INPUT);// declare fire sensor pin as input pinMode(enA, OUTPUT); // declare
as output for L298 Pin enA
  pinMode(in1, OUTPUT); // declare as output for L298 Pin in1
  pinMode(in2, OUTPUT); // declare as output for L298 Pin in2
}

```



```

pinMode(in3, OUTPUT); // declare as output for L298 Pin in3
pinMode(in4, OUTPUT); // declare as output for L298 Pin in4 pinMode(enB, OUTPUT); //
declare as output for L298 Pin enB
pinMode(servo, OUTPUT);
pinMode (pump, OUTPUT);
for (int angle = 90; angle <= 140; angle += 5) {
servoPulse(servo, angle);
for (int angle = 140; angle >= 40; angle == 5) {
servoPulse(servo, angle);}
for (int angle = 40; angle <= 95; angle += 5) {
servoPulse(servo, angle); }
analogWrite(enA, Speed);
analogWrite(enB, Speed);

delay (500);
void loop(){
s1 = analogRead (ir_R);
s2 = analogRead(ir_F);
s3 = analogRead (ir_L);

Serial.print(s1);
Serial.print ("t") ;
Serial.print(s2);
Serial.print ("t") ;
Serial.println(s3);
delay (50);
if(s1<250){
Stop();
Auto Controll
digitalWrite(pump, 1);
for(int angle = 90; angle >= 40; angle -= 3){
servoPulse(servo, angle);
for(int angle = 40; angle <= 90; angle += 3){
servoPulse (servo, angle);
else if(s2<350){

```

```

Stop();
digitalWrite(pump, 1);
for (int angle = 90; angle <= 140; angle += 3){
servoPulse(servo, angle);
for(int angle = 140; angle >= 40; angle -= 3){
servoPulse(servo, angle);
for (int angle = 40; angle <= 90; angle += 3){
servoPulse(servo, angle);

else if(s3<250){
Stop();
digitalWrite(pump, 1);
for(int angle = 90; angle <= 140; angle += 3){
servoPulse(servo, angle);
for(int angle = 140; angle >= 90; angle == 3){
servoPulse(servo, angle);
else if(s1>=251 && s1<=700){
digitalWrite(pump, 0);
backward();
delay (100);
turnRight();
delay (200);
else if(s2>=251 && s2<=800){|
digitalWrite(pump, 0);
forword ();
else if(s3>=251 && s3<=700){
digitalwrite(pump, 0);
backward();
delay(100);
turnLeft();
delay (200);
Jelsef
digitalWrite (pump, 0);
Stop ();
delay (10);
void servoPulse (int pin, int angle){

```

```

    delay (50);
    // Refresh cycle of servo
    void forward() { //forward
        digitalWrite(in1, HIGH); //Right Motor forward Pin digitalWrite(in2, LOW); //Right Motor
        backword Pin| digitalWrite(in3, LOW); //Left Motor backword Pin digitalWrite(in4, HIGH); //Left
        Motor forward Pin
        void backword() { //backword
            digitalWrite(in1, LOW); //Right Motor forward Pin digitalWrite(in2, HIGH); //Right Motor
            backword Pin digitalWrite(in3, HIGH); //Left Motor backword Pin digitalWrite(in4, LOW); //Left
            Motor forward Pin|
            void turnRight() { //turnRight
                digitalWrite(in1, LOW); //Right Motor forward Pin digitalWrite(in2, HIGH); //Right Motor
                backword Pin digitalWrite(in3, LOW); //Left Motor backword Pin| digitalWrite(in4, HIGH); //Left
                Motor forward Pin|
                void turnLeft() { //turnLeft
                    digitalWrite(in1, HIGH); //Right Motor forward Pin digitalWrite(in2, LOW); //Right Motor
                    backword Pin digitalWrite(in3, HIGH); //Left Motor backword Pin digitalWrite(in4, LOW); //Left
                    Motor forward Pin
                    void Stop() { //stop|
                        digitalWrite(in1, LOW); //Right Motor forward Pin digitalWrite(in2, LOW); //Right Motor
                        backword Pin digitalWrite(in3, LOW); //Left Motor backword Pin
                        digitalWrite(in4, LOW); //Left Motor forward Pin
                    }
                }
            }
        }
    }

```

## 3.6 SYSTEM MODELLING AND DESIGN

### 3.8.1 FUNCTIONAL DESCRIPTION

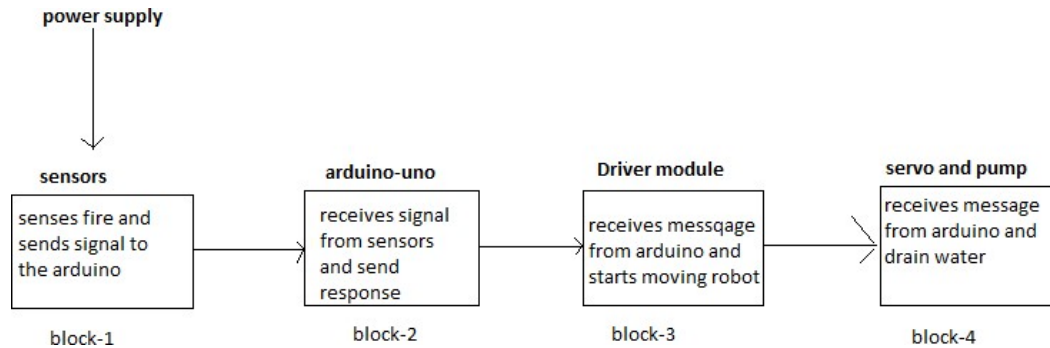


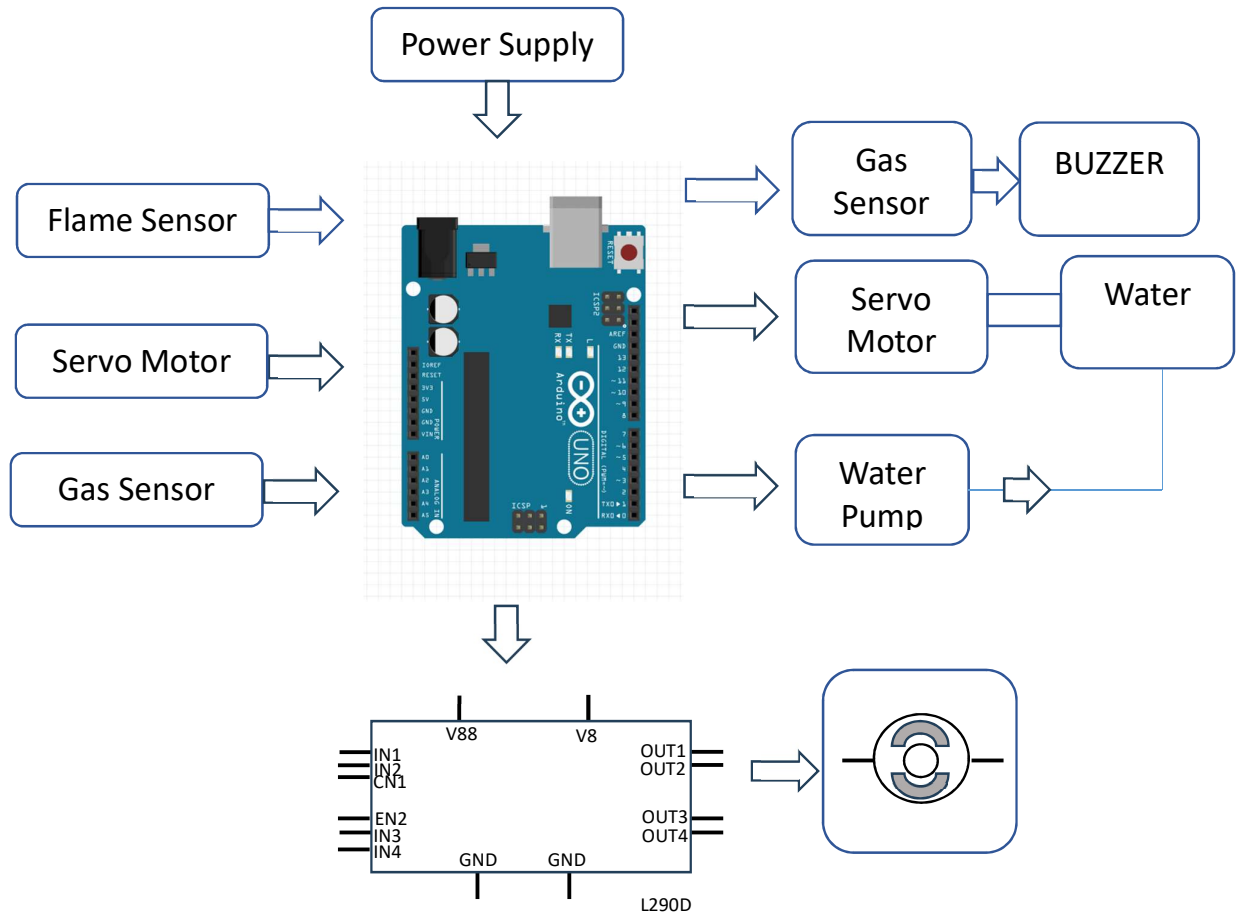
Fig 3.18 : Functional description

diagram The constituent parts involved in the process are

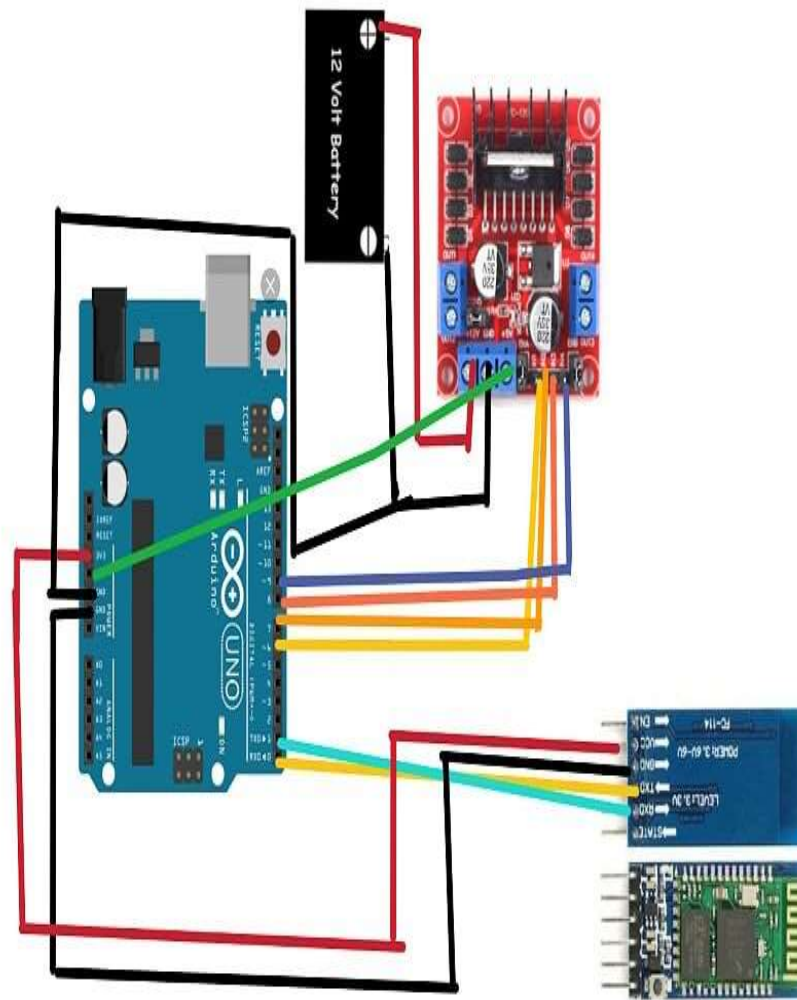
- sensors
- Arduino with Atmel Atmega328 microcontroller
- L293 driver module
- Servo with pump

First block portrays to be sensors which receives, verifies and forwards the message to the Microcontroller. Micro is the second block. Micro processes the message and sends to the driver module. Driver module behaving as the third constituent part and servo pump acts as fourth part which extinguishes the fire.

### 3.6.2. FUNCTIONAL BLOCK DIAGRAM



### 3.6.3. CIRCUIT DIAGRAM



### 3.6.4. DATA FLOW DIAGRAM

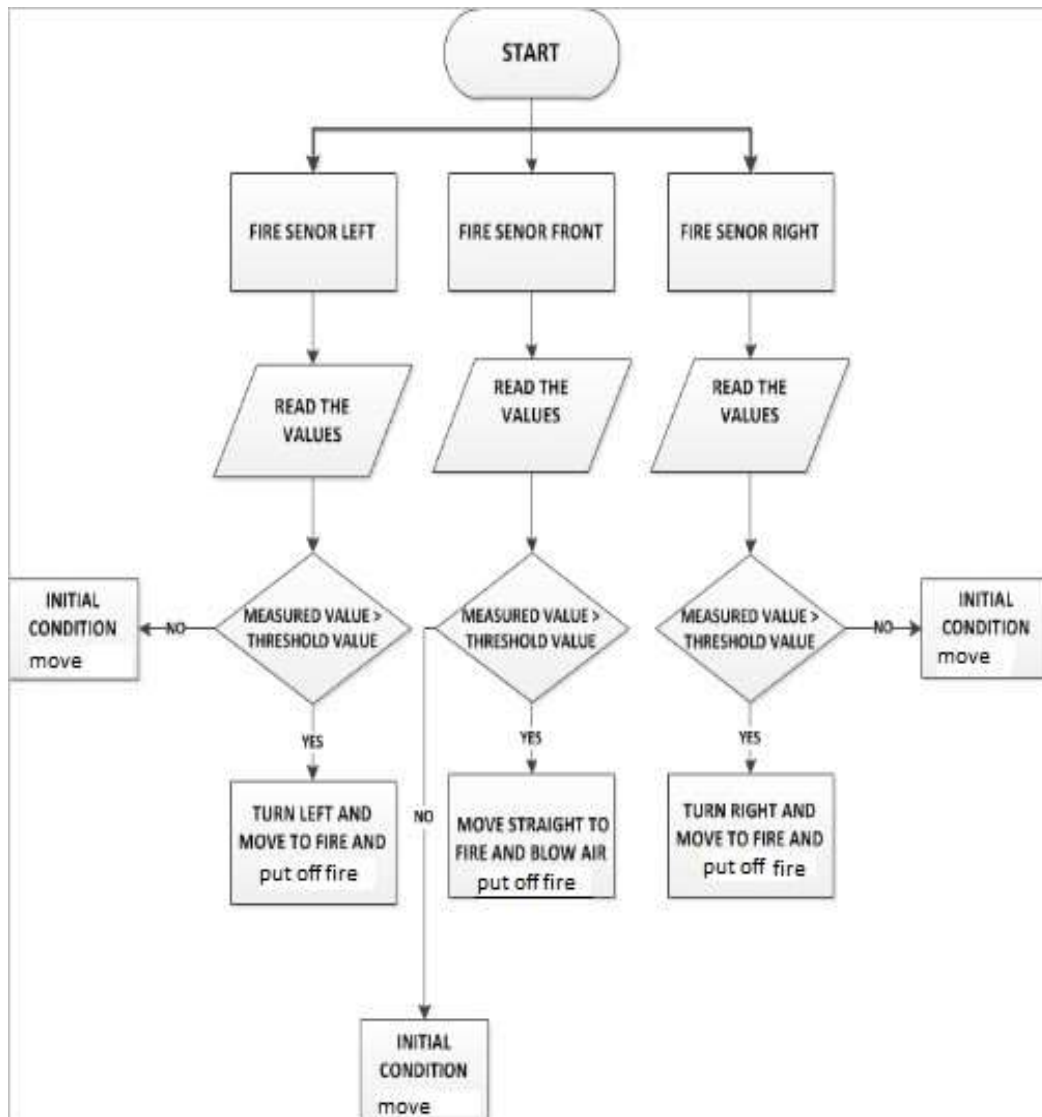


Fig 3.19 : Flow chart

The flow starts by initializing the ports of components. First the power supply should be on to the circuit and three sensors are there one on middle and remaining two on right and left side of chassis whenever the fire is occurred the respective value is read by the sensors when fire is occurred the voltage becomes zero and chassis is moved to the respective and put off fire whenever there is no fire then there is no input is occurred occurred voltage is more than 0 volts and the initial condition is move to other direction

## CHAPTER 4

### IMPLEMENTATION AND TESTING

#### 4.1. MICROCONTROLLER – FLAME SENSOR INTERFACING

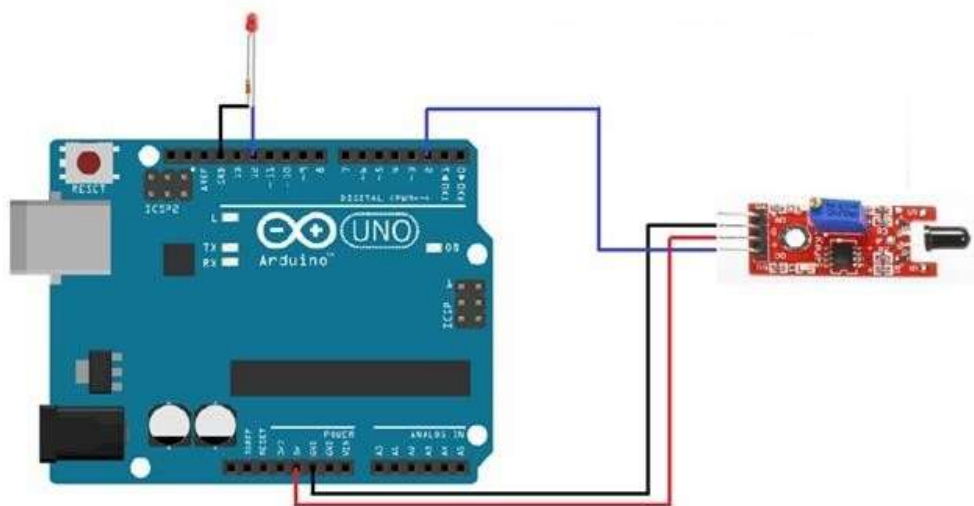


Fig 4.1: Microcontroller –flame sensor interfacing

Fig 4.1 The flame sensor is used to detect the fire or other light sources which are in the range of wavelength from 760nm to 1100nm. The module consists of an IR sensor, potentiometer, OP-Amp circuitry and a led indicator. When a flame will be detected, the module will turn on its red led. This module is sensitive to flame but it can also detect ordinary light. The detection point is 60 degrees. The sensitivity of this sensor is adjustable and it also has a stable performance.

It has both outputs, analog and digital. The analog output gives us a real time voltage output signal on thermal resistance while the digital output allows us to set a threshold via a potentiometer. In our tutorial we are going to use both of these outputs one by one and see how the sensor works. We can use the flame sensor to make an



alarm when detecting the fire, for safety purpose in many projects and in many more ways.

## **4.2 PROGRAMMING OVERVIEW**

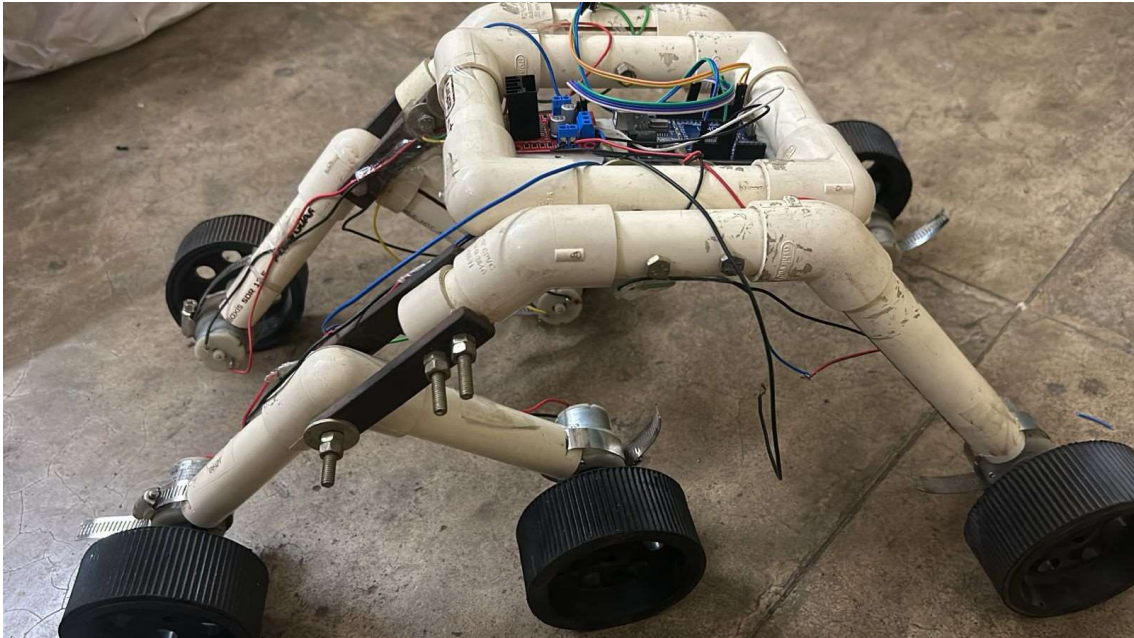
About Arduino Uno R3 Programming To programing Arduino Uno, you need the open source Arduino IDE software that the card manufacturer company wrote. The Arduino IDE Program is a software program written in Java language, used to program Arduino cards and to download Arduino cards to Arduino cards. download the program that I downloaded from the firm's site (<https://www.arduino.cc/>) with this program. It has an editor that uses the Processing / Wiring language, the commands that resemble the C language in some places, and the supporting utilities for the projects (Library - library). Along with this, another company's editor (IDE) has been developed since Arduino includes open source software. A bootloader (boot loader) has already been installed on ATmega328 on Arduino Uno. With this bootloader we can develop software without the need for an external programmer to program Arduino. The programming work can easily be performed by making the necessary settings and definitions in the IDE program .

## **4.3 RESULT ANALYSIS**

The initial stage of the project is the part of Finding fire, fire sensor LM393 The fire sensor detects fire at a certain distance. It does not receive data from areas outside of the determined area. It was decided to use two Reducing motors in order to realize the motion system. Both of these Reducing engines can move forward and backward. According to the obstacle state, if the motor is to be turned, one of the motors is given a reverse current by the processor and the axial rotation is provided and the obstacle less driving is provided. Thus, every obstacle was easily overcome in the environment where the system is located.

# RESULTS

## Snapshots



# **CHAPTER 5**

## **ADVANTAGES AND LIMITATIONS**

### **ADVANTAGES**

1. **Faster Response Time:** Firefighting robots are capable of responding to emergency situations faster than human firefighters. This can be especially beneficial in situations where human firefighters may be delayed due to traffic, bad weather, or other factors.
2. **Increased Safety:** Firefighting robots are able to enter dangerous environments that may be too hazardous for human firefighters. This is especially important when dealing with hazardous materials such as chemicals and toxic substances.
3. **Enhanced Mobility:** Firefighting robots are often equipped with powerful motors and can traverse terrain that may be difficult for humans to access. This can be very helpful in situations where the fire is located in an area that is not easily accessible to human firefighters.
4. **Improved Accuracy:** Firefighting robots are typically equipped with sophisticated sensors that can detect fire, heat, and smoke more accurately than the human eye. This helps to ensure that the fire is extinguished quickly and efficiently.
5. **Cost Savings:** Firefighting robots are often cheaper to operate than human firefighters. This can result in significant cost savings in terms of training, equipment, and manpower.

### **LIMITATIONS**

1. **Robot's Mobility:** Firefighting robots are typically wheeled, tracked or a combination of both, and they rely on their mobility to move around and reach the fire. However, if

the surface is too rough or if there are large obstacles present, the robot may not be able to move around freely.

2. Limited Reach: Firefighting robots have limited reach and may not be able to get close enough to put out the fire.
3. Limited Sensors: Firefighting robots have limited sensors and cameras to detect the fire, making it difficult to determine the exact location and size of the fire.
4. Limited Fire Fighting Capabilities: Firefighting robots are limited in their firefighting capabilities. They are typically limited to using water, foam, or dry chemical extinguishers, which may not be enough to put out large or intense fires.
5. High Cost: Firefighting robots are expensive and may not be affordable for some fire departments.
6. Safety: Firefighting robots pose a safety risk to firefighters, as they may not be able to detect hazardous conditions such as smoke or heat.

## **FUTURE SCOPE**

The project has been motivated by the desire to design a system that can detect fires and take appropriate action, without any human intervention. The development of sensor networks and the maturity of robotics suggests that we can use mobile agents for tasks that involve perception of an external stimulus and reacting to the stimulus, even when the reaction involves a significant amount of mechanical actions. This provides us the opportunity to pass on to robots tasks that traditionally humans had to do but were inherently life-threatening. Fire-fighting is an obvious candidate for such automation. Given the number of lives lost regularly in fire-fighting, the system we envision is crying for adoption. Our experience suggests that designing a fire-fighting system with sensors and robots is within the reach of the current sensor network and mobile agent technologies. Furthermore, we believe that the techniques developed in this work will carry over to other areas involving sensing and reacting to stimulus, where we desire to replace the human with an automated mobile agent.

However, there has been research on many of these pieces in different contexts e.g. coordination among mobile agents, techniques for detecting and avoiding obstacles, on-the-fly communication between humans and mobile agents, etc. It will be both interesting and challenging to put all this together into a practical, autonomous fire-fighting service.

## **CONCLUSION**

The firefighting robot is a promising new technology that has the potential to revolutionize the way fire fighters operate. It is capable of navigating through a burning building and locating the source of the fire and extinguishing it quickly and accurately. Its main advantage is that it can be used in hazardous environments where it would be too dangerous for humans to enter. It also has the potential to save lives, as it is capable of responding to fires more quickly than human firefighters. Therefore, the firefighting robot is an excellent tool to have in the firefighting arsenal.

## REFERENCES

- A. Eswaran, A. Vijay, S. Karthick, C. Sheik Mohammed, M. Vimal, Solae powered automatic firefighting robot, international journal of engineering research & technology (ijert), etedm – 2018 (volume 6 – issue 04), april 2018 .
- B. “Human Wireless Controlling Fire Fighting Robot (FFR) With 3-Axis Hose”, International Journal of Advanced Computer Technology (IJACT), Vol. 2, No. 3, pp. 1- 8.
- C. Mr. Borse Karan Dipak, Mr. Bansode Vishal Laxman, Mr. Gadekar Atish Mahadeo, Mr. Adhav Gitanjali Subhash, Miss. Shelke Amruta Ashok, Review of Fire Fighting Robot, IJSRD - International Journal for Scientific Research & Development | Vol. 4, Issue 01, 2016 | ISSN (online): 2321-0613 .
- D. Firefighting robot: an approach By-Manish Kumbhare, S s kumbhalkar Indian Streams Research Journal Vol.2, Issue. 1/March201412pp.1-4 Dr. Wael Abdulmajeed, Dr. Ali Mahdi and Karzan Taqi.
- E. Muhamad Bukhari Al-Mukmin, M.Zahar, Design And Development Of Automatic Fire Fighting Robot, 2011.
- F. W.Z. Wan Hasan, Mohd Hasimi Mohd Sidek, Suhaidi Shafie, Mohammad Hamiruce Marhaban, Fire Fighting Robot, Conference: APSAEM2010.At: Malaysia, July 2010.
- G. Pramod B.N., Hemalatha K.N., Poornima B.J., Harshita R., Fire Fighting Robot, 2019, IEEE, International Conference On Information and Communication Technology Convergence (ICTC), Jeju, South Korea Oct 2019 [ Kristi Kosasih, E. Merry Sartika, M. Jimmy Hasugian, dan Muliady the Intelligent Fire Fighting Tank Robot, Electrical Engineering Journal , volume 1, issue 1, Posted: 2010-10.

