A PROJECT REPORT ON

“BLUETOOTH CONTROLLED SURVEILLANCE ROBOT”

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE AWARD OF

DIPLOMA IN ELECTRONICS AND COMMUNICATION ENGINEERING

UNDER THE GUIDANCE OF

SRI. V.VIKAS,M.TECH.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

GOVERNMENT POLYTECHNIC,HYDERABAD

(2018-2021)

CERTIFICATE

This is to certify that the dissertation entitled " Bluetooth Controlled Surveillance Robot" , is being submitted in partial fulfillment of the requirements for the award of diploma in electronics and communication engineering of dece final year of **GOVERNMENT POLYTECHNIC MASAB TANK, HYDERABAD**

GUIDE:
Mr. V. VIKAS, M.TECH.,

HEAD OF DEPARTMENT
Mr. M. LAXMAIAH, M. Tech.,

External Examiner

BATCH CERTIFICATE

" BLUETOOTH COTROLLED SURVEILLANCE ROBOT"

A Dissertation submitted in partial fulfillment of the Requirements for the award of Diploma in Electronics and Communication Engineering

By

NAME	PIN NUMBER
B. ANNA RIBLIN	18001-EC-007
DEERAVATH VENU GOPAL	18001-EC-015
DUVVA DEEPTHI	18001-EC-017
JANMALA KEVIN	18001-EC-024

Under the Guidance of

Mr. V. VIKAS, M.TECH.,

ACKNOWLEDGEMENT

We would like to acknowledge with thanks for the cooperation and guidance by our guide Sri. V.VIKAS, M. Tech., to complete the project within the prescribed tie who provided us with the excellent environment in the lab and encouraging atmosphere which helped us in the successful completion of the project.

We would like to thank our Principal Srimati Rajeshwari and Sri M. Laxmaiah, M. Tech, Head of ECE Section for providing us with various computing facilities. We would like to thank the staff members of the government polytechnic, Masab tank, Hyderabad for being so cooperative throughout the processing of our project work.

Finally, we once again thank all the staff members of our ECE department for their help extended in successful completion of this project.

By

Yours Sincerely

2018-2021
batch

INDEX

Abstract

Tittle: Bluetooth controlled surveillance robot.

The robotics and automation industry which ruled the sectors from manufacturing to household entertainments. It is widely used because of its simplicity and ability to modify to meet changes of needs. And recently there is an intensive research undertaken in the field of intelligence robots and autonomous mobile robot applications. So through this project we wanted to explore this field by building a surveillance robot using Bluetooth module(as this is just

a prototype, the range is limited. later we can upgrade it using other different modules like wi-fi module).For remote operations and with a wireless camera for monitoring purpose the robot along with wireless camera transmit real time video. And this can potentially avoid obstacles as it is integrated with ultrasonic sensor and IR sensor to get information from surrounding areas and there by avoiding any kind off obstacle collisions. This is a kind of robot that can be helpful periodic patrolling in defined or restricted area, spying purpose in war fields and it can be used for gathering information in a particular areas which are dangerous for humans to access. The wi-fi technology is relatively new as compared to other technologies and there is huge potential of its growth and practical application. The android application loaded on mobile devices, can connect with security system and easy to use GUI. The security system then acts on these command and responds to the user. The CMOS camera and the motion detector are attached with security system for remote surveillance. A robot is a machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer. A robot can be controlled by a human operator, sometimes from a great distance. In such type of applications wireless communication is

more important. Surveillance security robot provides safety like man

Keywords And Glossary

Keywords :

Robot, Bluetooth, wi-Fi, Surveillance, sensors, servo motor, camera module , obstacle avoidance, Arduino

Glossary :

A

Arduino

Arduino is an open-source computer hardware and software company, project and user community that designs and manufactures kits for building digital devices and interactive objects that can sense and control the physical world.[1] Arduino boards may be purchased preassembled, or as do-it-yourself kits; at the same time, the hardware design information is available for those who would like to assemble an Arduino from scratch.

ATMEGA 328

The ATmega328 is a single chip micro-controller created by Atmel and belongs to the megaAVR series. The Atmel 8-bit AVR RISC-based microcontroller combines 32Â KB ISP flash memory with read-while-write capabilities, 1Â KB EEPROM, 2Â 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, 6channel 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. The device operates between 1.8-5.5 volts. The device achieves throughputs approaching 1 MIPS per Mhz.

B

BLUETOOTH

Bluetooth is a wireless technology standard for exchanging data over short distances (using short-wavelength UHF radio waves in the ISM band from 2.4 to 2.485Â

GHz[4]) from fixed and mobile devices, and building personal area networks (PANs). Invented by telecom vendor Ericsson in 1994,[5] it was originally conceived as a wireless alternative to RS-232 data cables. It can connect several devices, overcoming problems of synchronization.

E

ESP-32

It is a camera module . The ESP-32 is a powerful 32 bit microcontroller with integrated wi-fi , full TCP/IP stack for internet connection and bluetooth 4.2 . Due to low cost combined with great power and the opportunity to connect the ESP-32 to many other electronic devices . This is used to get the visual information from the robot.

H

HC 05

These small size Bluetooth TTL transceiver modules are designed for serial communication
(SPP - serial port profile). It allows your target device to both send or receive TTL data via Bluetooth technology without connecting a serial cable to your computer.

L

L293D

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 highercurrent signal. This higher current signal is used to drive the motors.

R

Robot

A Robot is an automatic mechanical device often resembling a human or animal. Modern robots are usually an electro-mechanical machine guided by a computer program or electronic circuitry. Robots can be autonomous or semi-autonomous and range from humanoids such as Honda’s Advanced Step in Innovative Mobility (ASIMO) and TOSY’s TOSY Ping Pong Playing Robot (TOPIO) to industrial robots, collectively programmed swarm robots, and even microscopic nano robots.

S

Servo motor

A servo motor is a rotary actuator or linear actuator that allows for precise control of angular (or) linear position , velocity and acceleration . It consists of suitable motor coupled to a sensor for position feedback .This provides fast precision position control for closed loop position control applications.

Surveillance

Surveillance is the monitoring of the behavior, activities, or other changing information, usually of people for the purpose of influencing, managing, directing, or protecting them.

W

Wi-fi

Wi-Fi (or WiFi) is a local area wireless technology that allows an electronic device to participate in computer networking using 2.4Â GHz UHF and 5Â GHz SHF ISM radio bands.

The Wi-Fi Alliance defines Wi-Fi as any "wireless local area network" (WLAN) product based on the Institute of Electrical and Electronics Engineers' (IEEE) 802.11 standards".[1] However, the term "Wi-Fi"is used in general English as a synonym for "WLAN" since most modern WLANs are based on these standards. "Wi-Fi" is a trademark of the Wi-Fi Alliance. The "Wi-Fi CERTIFIED" trademark can only be used by Wi-Fi products that successfully complete Wi-Fi Alliance interoperability certification testing.

Chapter 1

Project Overview

1.1 Introduction

The advent of new high-speed technology and the growing computer Capacity provided realistic opportunity for new robot controls and realization of new methods of control theory. This technical improvement together with the need for high performance robots created faster, more accurate and more intelligent robots using new robots control devices, new drivers and advanced control algorithms.

This project describes a new economical solution of robot control systems .In general; the robots are controlled through wired network. The programming of the robot takes time if there is any change in the project the reprogramming has to be done. Thus they are not user friendly and worked along with the user preferences. To make a robot user-friendly and to get the multimedia tone in the control of the robot, they are designed to make user commanded work. The modern technology has to be implemented to do this.

For implementing the modern technology it should be known by all the users to make use of it. To reach and to full-fill all these needs we are using android mobile as a multimedia for giving directions and voice controlled operations , which is user friendly device to control the robot. In this modern environment everybody uses smart phones which are a part of their day-to-day life. They use all their daily uses like newspaper reading, daily updates, social networking, and all the apps like home automation control, vehicle security, human body anatomy, health maintenance, etc has been designed in the form of applications which can be easily installed in their hand held smart phones. This project approached a robotic movement control through the smart phone application.

The embedded hardware works on the basis of Arduino which has ATMEGA-328 micro-controller and to be controlled by a Smart phone on the basis of Android platform. Arduino is to receive the AT commands from the Smart phone and takes the

data and controls the motors of the robot by the motor driver L293D. The robot can able to move forward, reverse, left and right movements. The Smart phone is been interfaced to the device by using Bluetooth. A Bluetooth device HC-05 module is going to be added to Arduino.

to receive commands from smart phone. A wireless camera is mounted on the robot body for spying purpose even in complete darkness by using infrared lighting.

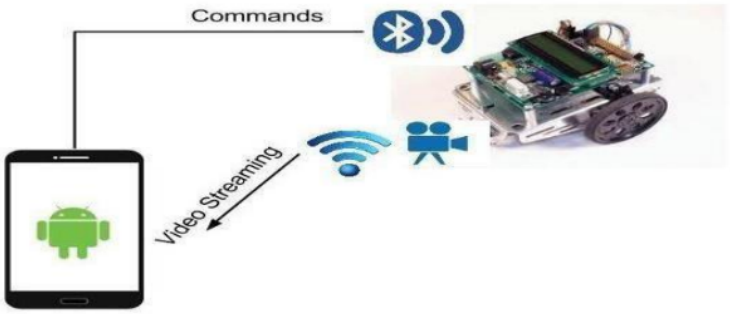
1.2 MOTIVATION

Normally after occurance of any disaster many helping forces will enter into the scene, but there will many half collapsed structures and many places that humans cannot enter due to the incident and some victims may be trapped in such places, such times this robot can be sent and can be used for checking the surroundings with the camera and help people to rescue .

Patrolling can be made to be in the dead zones too, to monitor surroundings and capture images using the camera .

1.3 OVERVIEW

The word surveillance may be applied to observation from a distance by means of electronic equipment such as cameras , CCTV etc. Or interception of electronically transmitted information such as phone video call.



we have successfully implemented the working of Bluetooth controlled surveillance robot. The robot is successfully controlled using the android application through the wireless Bluetooth technology .Even the real time video feel is achieved using the wi-fi technology.

CHAPTER 2: HARDWARE DESCRIPTION

2.1 ARDUINO UNO

The **Arduino Uno** is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is similar to the Arduino Nano and Leonardo. The hardware reference design is distributed under a
Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available.

The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software. The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases. The ATmega328 on the board comes preprogrammed with a bootloader that allows uploading new code to it without the use of an external hardware programmer.

General pin functions:

- **LED**: There is a built-in LED driven by digital pin 13. When the pin is high value, the LED is on, when the pin is low, it is off.
- **VIN**: The input voltage to the Arduino/Genuino board when it is using an external power source (as opposed to 5 volts from the USB connection or

other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

- **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 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board.
- **3V3**: A 3.3 volt supply generated by the on-board regulator. 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. A properly configured shield can read the IOREF pin voltage and select the appropriate power source, or enable voltage translators on the outputs to work with the 5V or 3. 3V.
- **Reset**: Typically used to add a reset button to shields that block the one on the board.

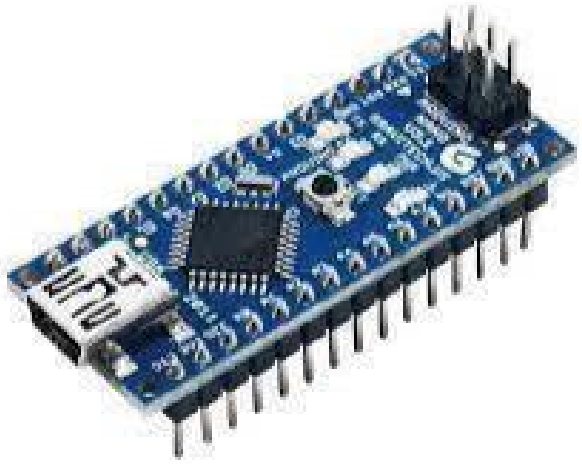
Special pin functions:

Each of the 14 digital pins and 6 analog pins on the Uno can be used as an input or output, under software control (using pinMode(), digitalWrite(), and digitalRead() functions). They operate at 5 volts. Each pin can provide or receive 20 mA as the recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50K ohm. A maximum of 40mA must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller. The Uno has 6 analog inputs, labeled A0 through A5; each provides 10 bits of resolution (i.e. 1024 different values). By default, they measure from ground to 5 volts, though it is possible to change the upper end of the range using the AREF pin and the analogReference() function.

In addition, some pins have specialized functions:

- **Serial** / UART: **pins** 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL serial chip.
- **External interrupts**: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM** (pulse-width modulation): pins 3, 5, 6, 9, 10, and 11. Can provide 8bit PWM output with the analogWrite() function.
- **SPI** (Serial Peripheral Interface): pins 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK). These pins support SPI communication using the SPI library.
- **TWI** (two-wire interface) / I²C: pin SDA (A4) and pin SCL (A5). Support TWI communication using the Wire library.
- **AREF** (analog reference): Reference voltage for the analog inputs.

2.2 ARDUINO NANO



The **Arduino Nano** is a small, complete, and breadboard-friendly board based on the ATmega328P released in 2008. It offers the same connectivity and specs of the Arduino Uno board in a smaller form factor.

The Arduino Nano is equipped with 30 male I/O headers, in a dip-30 like configuration, which can be programmed using the Arduino Software integrated development environment (IDE), which is common to all Arduino boards and running both online and offline. The board can be powered through a type-b micro-USB cable, or through a 9V battery.

The Arduino Nano has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provide UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An FTDI FT232RL on the board channels this serial communication over USB and the FTDI drivers (included with the Arduino software) provide a virtual com port to software on the computer. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Nano's digital pins. The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.

Rather than requiring a physical press of the reset button before an upload, the Arduino Nano is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the FT232RL

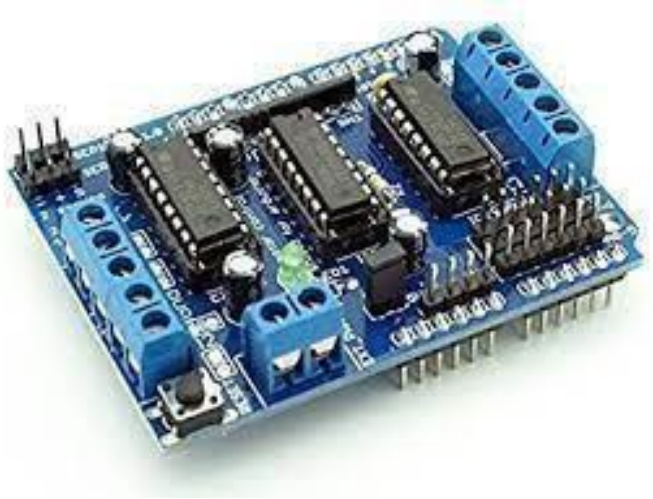
is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip.

This setup has other implications. When the Nano is connected to a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Nano. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened.

TECHNICAL SPECIFICATIONS:

- Microcontroller: Microchip ATmega328P
- Operating Voltage: 5 Volts
- Input Voltage: 6 to 20 Volts
- Digital I/O Pins: 14 (plus 6 can PWM output pins)
- Analog Input Pins: 8
- DC Current per I/O Pin: 40 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB of which 0.5 KB used by bootloader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Speed: 16 MHz
- Length: 45 mm
- Width: 18 mm
- Weight: 7 g

2.3 Motor Driver Sheild:



This is an extension shield that can drive 4 servos, 2 DC motors and one stepper motor. All you need to do is plug the shield into the Uno or Mega2560 board. It is powered by two sources – when connected to a control board, it’s powered by the output of the board; to drive a large-current motor, you can connect an external supply for the Motor Driver Shield and the control board. There's an indicator LED on the shield. When it's not in use, you can power the shield off by the switch and it won't influence the use of the control board. The working voltage is 6.5V-12V.

This shield is a dual-supply one. It can be connected directly to the Uno or Mega2560 board which then serves as the supply. But if you want to drive a DC motor of 5V or higher, you need an external power then. S1 is a single-pole doublethrow (SPDT) switch and controls the power of the shield. Pull it to OFF and the whole shield will not work. S2 is a 6-pin self-lock button switch that controls the external power supply. When you press down the button, the external power source is connected to shield and will also supply the control board. Thus, the external power can be no higher than 12V.

When the Motor Driver Shield is not in use, just pull S1 to OFF. And the shield will not affect the control board.

Connecting DC Motor

L293D

The two DC motors connected are controlled by the L293D chip.

The L293D is a 4-channel monolithic integrated motor driver chip of high voltage and high current. L293D has two pins (Vs and VSS) for power supply. Vs is used to supply power for the motor, while VSS, for the chip. If a small DC motor is used,

connect both pins to +5V. If you use a higher power motor, you need to connect Vs to an external power supply.

2.4 Esp 32 Cam :

ESP32 is a series of low-cost, low-power system on a chip **microcontrollers** with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations or a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and powermanagement modules. ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process. It is a successor to the ESP8266 microcontroller.

Features :

Features of the ESP32 include the following:

- Processors: o CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS

- Ultra low power (ULP) co-processor· Memory: 520 KiB SRAM, 448 KiB ROM· Wireless connectivity: o Wi-Fi: 802.11 b/g/n
- Bluetooth: v4.2 BR/EDR and BLE (shares the radio with Wi-Fi)

• Peripheral interfaces: o 34 × programmable GPIOs o 12-bit SAR ADC up to 18 channels o 2 × 8-bit DACs o 10 × touch sensors (capacitive sensing GPIOs) o 4 × SPI o 2 ×

- SDIO/SPI slave controller
- Ethernet MAC interface with dedicated DMA and IEEE 1588 Precision Time Protocol support o CAN bus 2.0 o Infrared remote controller (TX/RX, up to 8 channels) o Motor PWM o LED PWM (up to 16 channels)
- Hall effect sensor
- Ultra low power analog pre-amplifier

• Security:

- IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI o Secure boot o Flash encryption
- 1024-bit OTP, up to 768-bit for customers o Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)

• Power management: o Internal low-dropout regulator o Individual power domain for RTC o 5 µA deep sleep current o Wake up from GPIO interrupt, timer, ADC measurements, capacitive touch sensor interrupt

2.5 FTDI Programmer:

Future Technology Devices International Limited, commonly known by its acronym **FTDI**, is a Scottish privately held semiconductor device company, specialising in Universal Serial Bus (USB) technology.

It develops, manufactures, and supports devices and their related cables and software drivers for converting RS-232 or TTL serial transmissions to and from USB signals, in order to provide support for legacy devices with modern computers.

The company also provides application-specific integrated circuit (ASIC) design services, and consultancy services for product design, specifically in the realm of electronic devices.

FTDI was founded on 13 March 1992 by its current CEO, Fred Dart. The company is an indirect descendant of Computer Design Concepts Ltd, a former semiconductor technology startup also founded by Dart.

FTDI's initial products were chipsets for personal computer motherboards, the primary customer of which was IBM, which used them in its AMBRA and PS/1 personal computers. It later expanded its product line to include interface translators, such as the MM232R and the USB-COM232-PLUS1, along with other devices for converting between USB and other communication protocols. The headquarters of FTDI is in Glasgow, Scotland. It has offices in Singapore, Taipei (Taiwan), and Portland, Oregon, and a subsidiary in China. The company's manufacturing is handled by subcontractors in the Asia-Pacific region.

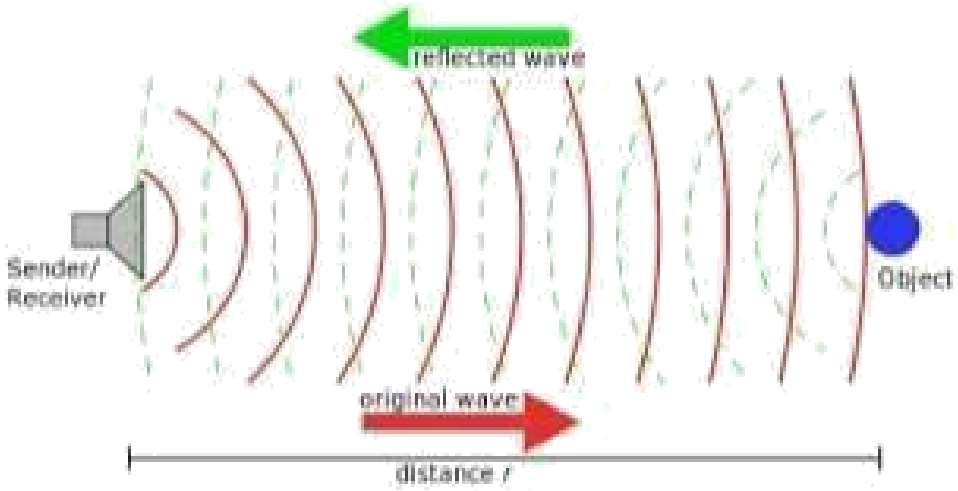
2.6 Ultrasonic Sensor:

Ultrasonic transducers and **ultrasonic sensors** are devices that generate or sense ultrasound energy. They can be divided into three broad categories: transmitters, receivers and transceivers. Transmitters convert electrical signals into ultrasound, receivers convert ultrasound into electrical signals, and transceivers can both transmit and receive ultrasound.

Ultrasound can be used for measuring wind speed and direction (anemometer), tank or channel fluid level, and speed through air or water. For measuring speed or direction, a device uses multiple detectors and calculates the speed from the relative distances to particulates in the air or water. To measure tank or channel liquid level, and also sea level (tide gauge), the sensor measures the distance (ranging) to the surface of the fluid. Further applications include: humidifiers, sonar, medical ultrasonography, burglar alarms, non-destructive testing and wireless charging.

Systems typically use a transducer which generates sound waves in the ultrasonic range, above 18 kHz, by turning electrical energy into sound, then upon receiving the echo turn the sound waves into electrical energy which can be measured and displayed.

This technology, as well, can detect approaching objects and track their positions.



Ultrasonic transducers convert AC into ultrasound, as well as the reverse. Ultrasonics, typically refers to piezoelectric transducers or capacitive transducers. Piezoelectric crystals change size and shape when a voltage is applied; AC voltage makes them oscillate at the same frequency and produce ultrasonic sound. Capacitive transducers use electrostatic fields between a conductive diaphragm and a backing plate.

The beam pattern of a transducer can be determined by the active transducer area and shape, the ultrasound wavelength, and the sound velocity of the propagation medium. The diagrams show the sound fields of an unfocused and a focusing ultrasonic transducer in water, plainly at differing energy levels.

Since piezoelectric materials generate a voltage when force is applied to them, they can also work as ultrasonic detectors. Some systems use separate transmitters and receivers, while others combine both functions into a single piezoelectric transceiver.

Ultrasound transmitters can also use non-piezoelectric principles. such as magnetostriction. Materials with this property change size slightly when exposed to a magnetic field, and make practical transducers.

A capacitor ("condenser") microphone has a thin diaphragm that responds to ultrasound waves. Changes in the electric field between the diaphragm and a closely spaced backing plate convert sound signals to electric currents, which can be amplified.

2.7 IR Sensor:

An **infrared** (**IR**) **sensor** is an electronic device that measures and detects **infrared** radiation in its surrounding environment. ... **IR** is invisible to the human eye, as its wavelength is longer than that of visible light (though it is still on the same electromagnetic spectrum).

A **passive infrared sensor** (**PIR sensor**) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors. PIR sensors are commonly used in security alarms and automatic lighting applications.

PIR sensors detect general movement, but do not give information on who or what moved. For that purpose, an imaging IR sensor is required.

PIR sensors are commonly called simply "PIR", or sometimes "PID", for "passive infrared detector". The term *passive* refers to the fact that PIR devices do not radiate energy for detection purposes. They work entirely by detecting infrared radiation (radiant heat) emitted by or reflected from objects.

Operating Principles:

All objects with a temperature above absolute zero emit heat energy in the form of electromagnetic radiation. Usually this radiation isn't visible to the human eye because it radiates at infrared wavelengths, but it can be detected by electronic devices designed for such a purpose.

The use of infrared sensors to accurately measure the chemical composition of materials or gases in military applications is well known. Now, however, given their rapidly declining price tag, these IR sensors are gaining traction in Internet of Things M2M applications, including medical diagnostics, imaging and industrial process controls, fire detection and remote gas leak detection, pollution monitoring, and realtime combustion control.

There are three major wavelength/frequency categories for the IR spectrum: near-, mid-, and far-IR. Near-IR involves fiber optic, IR sensors in the 700 nm – 1400 nm (0-7 µm – 1.4 µm) range. Mid-IR includes heat-sensing devices in the 1400 nm – 3000 nm (1.4 µm – 3 µm) range, and far, or thermal imaging IR, involves 3000 nm – 1 mm (3 µm – 1000 µm); all markets are seeing an uptick in sales and product development.
According to a recent market research report by ReportsnReports,[1] mid-IR sensor markets totaled $789 million in 2012 and are forecast to reach $7 billion by 2019. The impetus for growth includes price-performance increases and unit cost decreases from a high of $3,000 per unit to $300, in some cases, all the way down to approximately $8 per unit. Size has also migrated from bench sizes to portable units.

IR applications

Infrared sensors are used to sense characteristics in its surroundings by emitting and/or detecting infrared radiation and are capable of measuring the heat being emitted by an object and detecting motion.

Some of the most important tools for maintaining a clean, safe, and healthy environment are sensors, sensor systems, and sensor networks that detect the presence and quantify the amount of specific chemical trace gases. Once the source is located, monitoring also provided by sensors supports mitigation and compliance.

This is also true for industrial process and automotive monitoring and health, especially in breath analysis. Today's standard expensive and time-consuming medical tests will give way to breathalyzers able to diagnose medical conditions on the spot. Medical care will become more proactive and remote care more accurate for today's aging population.

Infrared vision has several applications. It can visualize heat leaks in houses, help doctors monitor blood flow, identify environmental chemicals in the environment, allow art historians to see under layers of paint, and integrate it with contact lenses or wearable electronics.

2.8 Servo Motor :

A **servomotor** is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

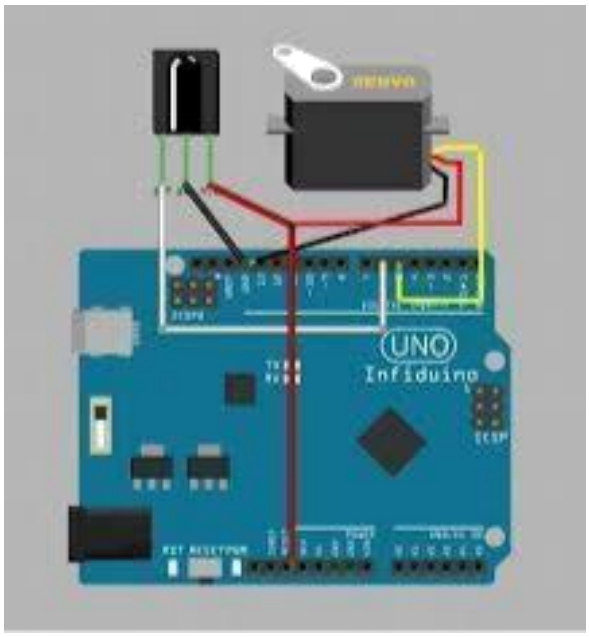
Servomotors are not a specific class of motor, although the term *servomotor* is often used to refer to a motor suitable for use in a closed-loop control system.

Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.

A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft.

The motor is paired with some type of position encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signal is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.

The very simplest servomotors use position-only sensing via a potentiometer and bang-bang control of their motor; the motor always rotates at full speed (or is stopped). This type of servomotor is not widely used in industrial motion control, but it forms the basis of the simple and cheap servos used for radiocontrolled models.



More sophisticated servomotors use optical rotary encoders to measure the speed of the output shaft[2] and a variable-speed drive to control the motor speed. Both of these enhancements, usually in combination with a PID control algorithm, allow the servomotor to be brought to its commanded position more quickly and more precisely, with less overshooting.

The servo motor is most commonly used for high technology devices in the industrial applications like automation technology. It is a self contained electrical device, that rotates parts of machine with high efficiency and great precision. Moreover the output shaft of this motor can be moved to a particular angle. Servo motors are mainly used in home electronics, toys, cars, airplanes and many more devices.

Servo motor works on the PWM (Pulse Width Modulation) principle, which means its angle of rotation is controlled by the duration of pulse applied to its control PIN. Basically servo motor is made up of DC motor which is controlled by a variable resistor (potentiometer) and some gears.

Applications :

1. Robotics : At every joint of the robot, we connect a servomotor. Thus giving the robot arm its precise angle.

2. Conveyor belts : servo motors move , stop , and start conveyor belts carrying product along to various stages , for example , in product packaging/ bottling, and labelling .

3. Camera auto focus : A highly precise servo motor build into the camera corrects a camera lens to sharpen out of focus images.

2.9 DC Gear Motors :

A **DC motor** is any of a class of rotary electrical motors that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor.

DC motors were the first form of motor widely used, as they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight brushed motor used for portable power tools and appliances. Larger DC motors are currently used in propulsion of electric vehicles, elevator and hoists, and in drives for steel rolling mills. The advent of power electronics has made replacement of DC motors with AC motors possible in many applications.

A direct current or **DC motor**, converts electrical energy into mechanical energy. It is one of two basic types of **motors**: the other type is the alternating current or AC **motor**. Among **DC motors**, there are shunt-wound, series-wound, compoundwound and permanent magnet **motors**.

Speed reduction: **Gear motors** are also known as **gear** reducers because as they increase output torque they decrease output speed. A **motor** running at 1,000 rpm fitted with a 5:1 ratio gearhead outputs 200 rpm. This speed reduction improves system performance because many **motors** do not operate efficiently at low rpm.

DC motors include **two key components**: a stator and an armature. The stator is the stationary **part** of a **motor**, while the armature rotates. In a **DC motor**, the stator provides a rotating magnetic field that drives the armature to rotate.

2.10 Bluetooth Module HC-05 :

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

HC-05 has red LED which indicates connection status, whether the Bluetooth is connected or not. Before connecting to HC-05 module this red LED blinks continuously in a periodic manner. When it gets connected to any other Bluetooth device, its blinking slows down to two seconds.

This module works on 3.3 V. We can connect 5V supply voltage as well since the module has on board 5 to 3.3 V regulator.

As HC-05 Bluetooth module has 3.3 V level for RX/TX and microcontroller can detect 3.3 V level, so, no need to shift transmit level of HC-05 module. But we need to shift the transmit voltage level from microcontroller to RX of HC-05 module.

Bluetooth serial modules allow all serial enabled devices to communicate with each other using Bluetooth.

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.

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.

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.

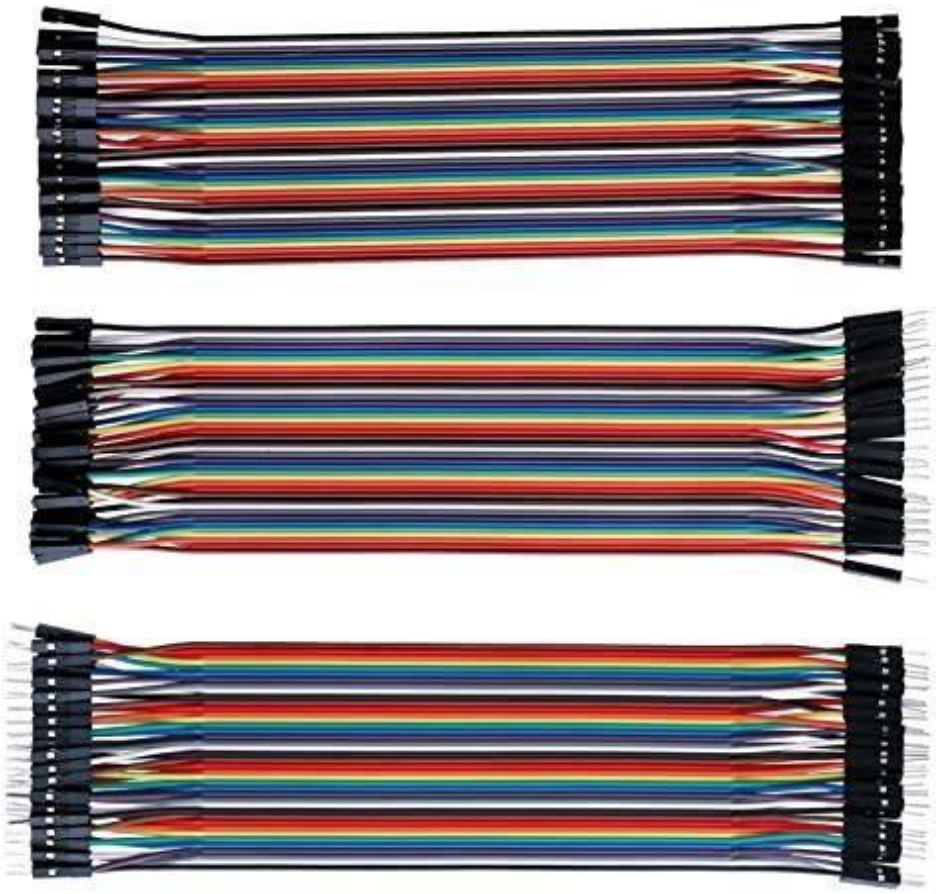
It is used for many applications like wireless headset, game controllers, wireless mouse, wireless keyboard and many more consumer applications.

It has range up to <100m which depends upon transmitter and receiver, atmosphere, geographic & urban conditions.

It is IEEE 802.15.1 standardized protocol, through which one can build wireless Personal Area Network (PAN). It uses frequency-hopping spread spectrum (FHSS) radio technology to send data over air.

It uses serial communication to communicate with devices. It communicates with microcontroller using serial port (USART).

2.11 Jumper Wires:



A **jump wire** (also known as jumper, jumper wire, jumper cable, DuPont wire or cable) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment.

SHOWA **jumper wire** (NSL: New Showa Lead) is a lead-free tin-plated annealed copper **wire**. Tin plating is tin: 99.2%, copper: 0.8%. In general, it is said that hot plating is difficult to control the plating thickness compared with electroplating, but we control the plating thickness by the original processing method.

2.12 Connecting Wires :

A **wire** is a single usually cylindrical, flexible strand or rod of metal. Wires are used to bear mechanical loads or electricity and telecommunications signals. Wire is commonly formed by drawing the metal through a hole in a die or draw plate. Wire gauges come in various standard sizes, as expressed in terms of a gauge number.

The term 'wire' is also used more loosely to refer to a bundle of such strands, as in "multistranded wire", which is more correctly termed a wire rope in mechanics, or a cable in electricity.

Wire comes in solid core, stranded, or braided forms. Although usually circular in cross-section, wire can be made in square, hexagonal, flattened rectangular, or other cross-sections, either for decorative purposes, or for technical purposes such as high-efficiency voice coils in loudspeakers. Edge-wound coil springs, such as the Slinky toy, are made of special flattened wire.

Not all metals and metallic alloys possess the physical properties necessary to make useful wire. The metals must in the first place be ductile and strong in tension, the quality on which the utility of wire principally depends. The principal metals suitable for wire, possessing almost equal ductility, are platinum, silver, iron, copper, aluminium, and gold; and it is only from these and certain of their alloys with other metals, principally brass and bronze, that wire is prepared.

2.13 Chassis (Body Of Robot) :

A **chassis** is the load-bearing framework of an artificial object, which structurally supports the object in its construction and function. An example of a chassis is a vehicle frame, the underpart of a motor vehicle, on which the body is mounted; if the running gear such as wheels and transmission, and sometimes even the driver's seat, are included, then the assembly is described as a rolling chassis.

In an electronic device (such as a computer), the chassis consists of a frame or other internal supporting structure on which the circuit boards and other electronics are mounted.

In some designs, such as older ENIAC sets, the chassis is mounted inside a heavy, rigid cabinet, while in other designs such as modern computer cases, lightweight covers or panels are attached to the chassis.

The combination of chassis and outer covering is sometimes called an *enclosure*.

The **chassis** provides the strength needed for supporting the different vehicular components as well as the payload and helps to keep the automobile rigid and stiff. Consequently, the **chassis** is also an **important** component of the overall safety system.

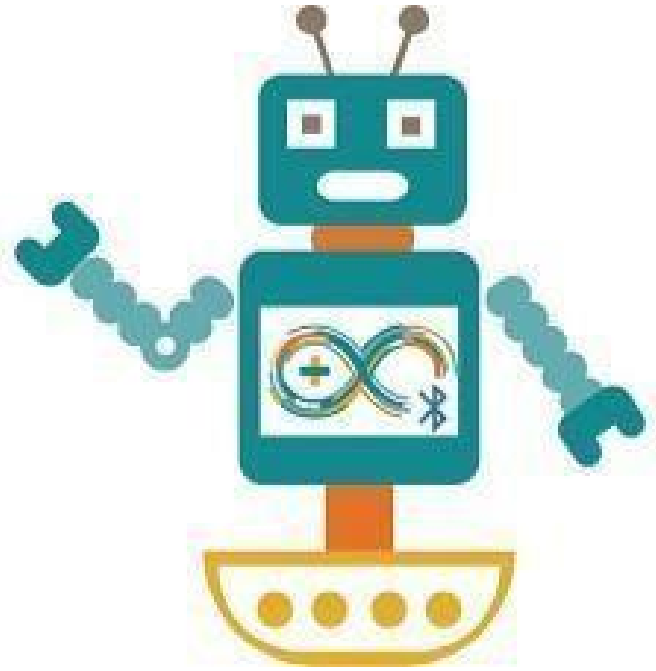
CHAPTER 3: SOFTWARE

3.1 Arduino IDE :

The **Arduino Integrated Development Environment (IDE) is a crossplatform application (for Windows, macOS, Linux) 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.**

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards. Arduino IDE is a derivative of the Processing IDE, however as of version 2.0, the Processing IDE will be replaced with the Visual Studio Code-based Eclipse Theia IDE framework.

3.2 Arduino Bluetooth Control :



Arduino Bluetooth Control is an application that allows you to control your arduino board (and similar boards) via Bluetooth, and so to create awesome and fully customized projects, with the new features available within the app.
The settings section allows you to adapt the application to your needs, through a very simple and intuitive interface.

The application also smartly remembers your bluetooth module and will always try to connect automatically to the latest one you have used, so you won't have to select it every time you use it.

You can also use the application on your wearable device if you have any.

1.Metrics tool
This tool was optimized to receive data via the println() function of arduino, which allows special processing of the data received, like in the "Metrics" tool. It allows you to receive only numbers and fix alarms to get notified about the variations of the value received.Once the alarm triggered, a stop button shows up, allowing you to stop it.Besides you can activate the shaking mode, that will allow you to send data simply by shaking your phone.

2.Arrow keys
This tool provides direction buttons that can fully customized with the data to send, and the sensitivity, which allows to send continuously data to the board by maintainning long press on them.

3.Terminal

This tool is just a classic terminal that receives and sends data to the board, displayed with the timestamp corresponding to each action.

4.Buttons and slider

In portrait orientation, this tool provides 6 buttons fully customized, that will allow you to send specific data when pressed. When you rotate your device, a slider view shows up , to which you can set the range of the data to be sent.

5.Accelerometer

This tool permits you to interpret the gesture commands of your phone, and send the corresponding data to your board, and so , your phone can be the steering wheel of your robot. You can of course set the sensitivity of it through the settings interface.

6. Voice Control

Have you ever dreamed of talking to you robots ? well now your dream is becoming true ! With Arduino Bluetooth Control, you can customize your own vocal commands and use them to control all your microcontroller-based boards.

3.3 Arduino Bluetooth RC Car :

This application is designed to be used with a MODIFIED RC car. You have to replace the car's stock control circuit with a micro controller. This involves programming. The application will not work with a brand new, out of the box RC car. Please visit the website before you download the application.

The application allows you to control an Arduino based RC car over Bluetooth. This is done using a Bluetooth enabled Android phone. Visit this site https://sites.google.com/site/bluetoothrccar/ for the Arduino code and control circuit. The app lets you control the car with either buttons or the phone's accelerometer. A slider bar allows you to control your car's velocity if the car's control circuit has this feature. There are also two buttons for front and back lights. A flashing light lets you know when the phone is connected to the car, and arrows light up letting you know the car's driving direction.

CHAPTER 4: PROGRAMS

Obstacle Avoidance Voice Control Program :

```
#include <AFMotor.h>
#include <NewPing.h>
#include<Servo.h>
#define TRIGGER_PIN A1
#define ECHO_PIN A0
#define MAX_DISTANCE 300
#define IR A5

AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(2, MOTOR12_1KHZ);
AF_DCMotor motor3(3, MOTOR34_1KHZ);
AF_DCMotor motor4(4, MOTOR34_1KHZ);



NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

Servo myservo;

String voice;

void setup() {
 Serial.begin(9600);
 myservo.attach(10);
```

```
myservo.write(90);
pinMode(IR, INPUT); }

void loop() { int distance = sonar.ping_cm();
  //int IR1 = digitalRead(IR);
  //Serial.println(IR1);


if(Serial.available()>0) {
voice="";   delay(2);   voice =
Serial.readString();
delay(2);
  Serial.println(voice);



if (voice == "turn left") {   left();
}else if (voice == "left") {   left();
}else if(voice == "turn right") {   right();
}else if(voice ==
"right") {   right(); }
  }
while(voice == "move forward") {   forward();
}
while(voice == "move backward") {   backward();
}
}
```

```
void forward() {   int distance = sonar.ping_cm();
```

```
 if(distance < 10){
Stop();   voice="";
 }else {
motor1.setSpeed(255);
 motor1.run(FORWARD);
 motor2.setSpeed(255);
 motor2.run(FORWARD);
 motor3.setSpeed(255);
 motor3.run(FORWARD);
 motor4.setSpeed(255);
 motor4.run(FORWARD);
}
}
void backward() {   int IR_Sensor =
 digitalRead(IR);    if(IR_Sensor
== 0) {     Stop();     voice="";
  }else {
motor1.setSpeed(255);
 motor1.run(BACKWARD);
 motor2.setSpeed(255);
 motor2.run(BACKWARD);
 motor3.setSpeed(255);
 motor3.run(BACKWARD);
 motor4.setSpeed(255);
 motor4.run(BACKWARD);


}
} void left() {
myservo.write(180); delay(500);
myservo.write(90);
```

```
delay(500);
motor1.run(BACKWARD);
motor1.setSpeed(255);
motor2.run(BACKWARD);
motor2.setSpeed(255);
motor3.run(FORWARD);
motor3.setSpeed(255);
motor4.run(FORWARD);
motor4.setSpeed(255);
delay(700);
motor1.run(RELEASE);
motor2.run(RELEASE);
motor3.run(RELEASE);
motor4.run(RELEASE);
}

void right() { myservo.write(0); delay(500); myservo.write(90); delay(500);
motor1.run(FORWARD); motor1.setSpeed(255); motor2.run(FORWARD);
motor2.setSpeed(255); motor3.run(BACKWARD); motor3.setSpeed(255);
motor4.run(BACKWARD); motor4.setSpeed(255); delay(700);
motor1.run(RELEASE); motor2.run(RELEASE); motor3.run(RELEASE);
motor4.run(RELEASE);
}

void Stop() {
motor1.run(RELEASE);
motor2.run(RELEASE);
motor3.run(RELEASE);
motor4.run(RELEASE);
}
```

Bluetooth Control Program :

```
#include <Servo.h>
#include <AFMotor.h>
#define Echo A0
#define Trig A1
#define motor 10
#define Speed
170 #define spoint
103 char value; int
distance;  int  Left;
int Right; int L = 0;
int R = 0; int L1 =
0; int R1 = 0;
Servo servo;
AF_DCMotor M1(1);
AF_DCMotor M2(2);
AF_DCMotor M3(3);  AF_DCMotor
M4(4); void
setup()                    {
Serial.begin(9600);
pinMode(Trig, OUTPUT);
pinMode(Echo,    INPUT);
servo.attach(motor);
M1.setSpeed(Speed);
  M2.setSpeed(Speed);
  M3.setSpeed(Speed);
  M4.setSpeed(Speed);
}
void loop() {
```

```
  //Obstacle();   Bluetoothcontrol();
  //voicecontrol();
}
void  Bluetoothcontrol()  {
if (Serial.available() > 0) {
value = Serial.read();

Serial.println(value);
}   if (value  == 'F') {
forward();     }  else   if
(value      ==    'B')     {
backward();   } else if
(value == 'L') {      left();
} else if (value == 'R') {
right();
  } else if (value == 'S') {
    Stop();
  }
}
void Obstacle() {
distance = ultrasonic();
if (distance <= 12) {
Stop();     backward();
delay(100);     Stop();     L
= leftsee();
servo.write(spoint);
delay(800);
    R = rightsee();      servo.write(spoint);      if (L < R) {        right();       delay(500);
Stop();       delay(200);     } else if (L > R) {         left();
```

```
      delay(500);
 Stop();
 delay(200);
    }   } else {       forward();
  }
}
void voicecontrol() {   if
(Serial.available() > 0) {
value = Serial.read();
Serial.println(value);      if
(value == '^') {         forward();
} else if
(value == '-') {         backward();
    } else if (value == '<') {
L = leftsee();
servo.write(spoint);       if
(L >= 10 ) {          left();
delay(500);          Stop();

      } else if (L < 10) {
        Stop();
      }
    } else if (value == '>') {
R = rightsee();
servo.write(spoint);if
(R >= 10 ) {          right();
delay(500);          Stop();
      } else if (R < 10) {
        Stop();
      }
    } else if (value == '*') {
```

```
      Stop();
    }
  }
}
//        Ultrasonic sensor distance reading
function int ultrasonic() { digitalWrite(Trig,
LOW); delayMicroseconds(4);
digitalWrite(Trig, HIGH);
delayMicroseconds(10); digitalWrite(Trig,
LOW); long t = pulseIn(Echo, HIGH); long cm
= t / 29 / 2; //time convert distance return
cm;
}
void forward() {
  M1.run(FORWARD);
  M2.run(FORWARD);
  M3.run(FORWARD);
  M4.run(FORWARD);
}
voidbackward() { M1.run(BACKWARD);

  M2.run(BACKWARD);
  M3.run(BACKWARD);
  M4.run(BACKWARD); }
void left()
{
  M1.run(FORWARD);
  M2.run(BACKWARD);      M3.run(BACKWARD);
  M4.run(FORWARD);
} void right()
{
```

```
  M1.run(BACKWARD);
  M2.run(FORWARD);
  M3.run(FORWARD);
  M4.run(BACKWARD);
}
void              Stop()              {
 M1.run(RELEASE);
  M2.run(RELEASE);
  M3.run(RELEASE);
  M4.run(RELEASE);
} int rightsee() {
 servo.write(20);
 delay(800);
  Left = ultrasonic();
 return Left;  } int
 leftsee() {
 servo.write(180);
 delay(800);   Right =
 ultrasonic();   return
Right;
}
```

Esp 32 Cam Program :

```
#include "esp_camera.h"
#include <WiFi.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality //
Ensure ESP32 Wrover Module or other board with PSRAM is selected
//         Partial images will be transmitted if image exceeds buffer size
//

//Select camera model
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has
PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM //#define
CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM

#include "camera_pins.h"
```

```
const char* ssid = "Airtel-WD670-C2A0"; const char*
password = "D8FCC2A0";

void startCameraServer();

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config; config.ledc_channel
= LEDC_CHANNEL_0;
config.ledc_timer   =   LEDC_TIMER_0;
config.pin_d0       =    Y2_GPIO_NUM;
config.pin_d1       =    Y3_GPIO_NUM;
config.pin_d2       =    Y4_GPIO_NUM;
config.pin_d3       =    Y5_GPIO_NUM;
config.pin_d4       =    Y6_GPIO_NUM;
config.pin_d5       =    Y7_GPIO_NUM;
config.pin_d6       =    Y8_GPIO_NUM;
config.pin_d7       =    Y9_GPIO_NUM;
config.pin_xclk     =   XCLK_GPIO_NUM;
config.pin_pclk     =   PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href     =   HREF_GPIO_NUM;
config.pin_sscb_sda =  SIOD_GPIO_NUM;
config.pin_sscb_scl =  SIOC_GPIO_NUM;
config.pin_pwdn     =  PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
```

```
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

 // if PSRAM IC present, init with UXGA resolution and higher JPEG quality // for
larger pre-allocated frame buffer.
 if(psramFound()){
config.frame_size =
FRAMESIZE_UXGA;        config.jpeg_quality
=         10; config.fb_count =
 2; } else {
   config.frame_size = FRAMESIZE_SVGA; config.jpeg_quality =
12; config.fb_count
= 1;
 }

#if defined(CAMERA_MODEL_ESP_EYE)
pinMode(13,                  INPUT_PULLUP);
pinMode(14, INPUT_PULLUP);
#endif

 // camera init  esp_err_t err = esp_camera_init(&config);
if (err !=
ESP_OK) {
   Serial.printf("Camera init failed with error 0x%x", err);       return;
 }

 sensor_t * s = esp_camera_sensor_get();
 // initial sensors are flipped vertically and colors are a bit saturated
if (s->id.PID == OV3660_PID) { s->set_vflip(s, 1); // flip it back
s->set_brightness(s, 1); // up the brightness just a bit
s->set_saturation(s, -2); // lower the saturation
```

```
  }
  // drop down frame size for higher initial frame rate
s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);    s-
>set_hmirror(s, 1);
#endif

  WiFi.begin(ssid,  password);

  while (WiFi.status() != WL_CONNECTED) { delay(500);

    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi  connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // put your main code here, to run repeatedly:    delay(10000);
}
```

Esp 32 Cam Configuration :

Camera PINS :

```
#if defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
```

APP HTTP :

```
#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "camera_index.h"
#include "Arduino.h"
```

```
#include "fb_gfx.h"
#include "fd_forward.h"
#include "fr_forward.h"

#define ENROLL_CONFIRM_TIMES 5
#define FACE_ID_SAVE_NUMBER 7

#define FACE_COLOR_WHITE 0x00FFFFFF
#define FACE_COLOR_BLACK 0x00000000
#define FACE_COLOR_RED 0x000000FF #define FACE_COLOR_GREEN 0x0000FF00
#define FACE_COLOR_BLUE 0x00FF0000
#define FACE_COLOR_YELLOW (FACE_COLOR_RED | FACE_COLOR_GREEN)
#define FACE_COLOR_CYAN (FACE_COLOR_BLUE | FACE_COLOR_GREEN)
#define FACE_COLOR_PURPLE (FACE_COLOR_BLUE | FACE_COLOR_RED)

typedef struct { size_t size; //number of values used
for filtering size_t index; //current value index size_t
count; //value count int sum;
int * values; //array to be filled with values
} ra_filter_t;

typedef struct {
httpd_req_t *req;
size_t len;
} jpg_chunking_t;

#define PART_BOUNDARY "123456789000000000000987654321" static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixedreplace;boundary="
PART_BOUNDARY; static const char* _STREAM_BOUNDARY = "\r\n--"
PART_BOUNDARY "\r\n"; static const char* _STREAM_PART = "Content-Type:
image/jpeg\r\nContent-Length: %u\r\n\r\n";
```

```
static ra_filter_t ra_filter;
httpd_handle_t stream_httpd = NULL; httpd_handle_t
camera_httpd = NULL;

static mtmn_config_t mtmn_config = {0};
static int8_t detection_enabled = 0; static
int8_t  recognition_enabled  =  0;  static
int8_t is_enrolling  = 0; static face_id_list
id_list = {0};

static ra_filter_t * ra_filter_init(ra_filter_t      * filter, size_t sample_size){         memset(filter,
0, sizeof(ra_filter_t));

    filter->values = (int *)malloc(sample_size * sizeof(int));
if(!filter->values){         return NULL;
    }
    memset(filter->values, 0, sample_size * sizeof(int));

    filter->size = sample_size;         return
filter;
}

static int ra_filter_run(ra_filter_t * filter, int value){
if(!filter->values){         return value;
    }
    filter->sum -= filter->values[filter->index];
filter->values[filter->index] = value;
filter>sum += filter->values[filter->index];
filter->index++;      filter->index = filter>index
```

```
% filter->size;      if (filter->count < filter-
>size) {         filter->count++;
   }
   return filter->sum / filter->count;
}

static void rgb_print(dl_matrix3du_t *image_matrix, uint32_t color, const char *
str){ fb_data_t fb; fb.width = image_matrix->w; fb.height = image_matrix>h;
fb.data = image_matrix->item;      fb.bytes_per_pixel = 3;       fb.format =
FB_BGR888;       fb_gfx_print(&fb, (fb.width - (strlen(str) * 14)) / 2, 10, color, str);
}

static int rgb_printf(dl_matrix3du_t *image_matrix, uint32_t color, const char
*format, ...){
   char loc_buf[64]; char * temp = loc_buf; int len;
va_list arg; va_list copy; va_start(arg, format);
va_copy(copy, arg); len = vsnprintf(loc_buf,
sizeof(loc_buf), format, arg); va_end(copy); if(len >=
sizeof(loc_buf)){ temp = (char*)malloc(len+1); if(temp
== NULL) { return 0;
      }
   }
   vsnprintf(temp, len+1, format, arg);
va_end(arg);      rgb_print(image_matrix, color,
temp);     if(len > 64){         free(temp);
   }     return
len;
}

static void draw_face_boxes(dl_matrix3du_t *image_matrix, box_array_t *boxes, int
face_id){
```

```
    int x, y, w, h, i;
    uint32_t color = FACE_COLOR_YELLOW;
    if(face_id < 0){
        color = FACE_COLOR_RED;
    } else if(face_id > 0){
        color = FACE_COLOR_GREEN;
    }
    fb_data_t fb;      fb.width =
image_matrix->w;      fb.height =
image_matrix->h;      fb.data =
image_matrix->item;
fb.bytes_per_pixel = 3;       fb.format
= FB_BGR888;       for
(i = 0; i < boxes->len; i++){
        // rectangle box        x = (int)boxes-
>box[i].box_p[0];                                y    =
(int)boxes>box[i].box_p[1];                     w  =
(int)boxes>box[i].box_p[2]  - x + 1;            h =
(int)boxes-
>box[i].box_p[3] - y + 1;
fb_gfx_drawFastHLine(&fb, x, y, w, color);
  fb_gfx_drawFastHLine(&fb, x, y+h-1, w, color);
  fb_gfx_drawFastVLine(&fb, x, y, h, color);
  fb_gfx_drawFastVLine(&fb, x+w-1, y, h, color);

#if 0
        // landmark        int x0, y0, j;           for (j = 0; j <
10; j+=2) {             x0 = (int)boxes-
>landmark[i].landmark_p[j];               y0 = (int)boxes-
>landmark[i].landmark_p[j+1];               fb_gfx_fillRect(&fb,
x0, y0, 3, 3, color);
        }
```

```
#endi
  f
  }
}

static int run_face_recognition(dl_matrix3du_t *image_matrix, box_array_t
*net_boxes){ dl_matrix3du_t
*aligned_face = NULL; int matched_id
= 0;

  aligned_face = dl_matrix3du_alloc(1, FACE_WIDTH, FACE_HEIGHT, 3);
if(!aligned_face){
    Serial.println("Could not allocate face recognition buffer"); return
matched_id;
  }
  if (align_face(net_boxes, image_matrix, aligned_face) ==
ESP_OK){ if (is_enrolling == 1){ int8_t left_sample_face =
enroll_face(&id_list, aligned_face);

    if(left_sample_face == (ENROLL_CONFIRM_TIMES - 1)){
      Serial.printf("Enrolling Face ID: %d\n", id_list.tail);
    }
    Serial.printf("Enrolling Face ID: %d sample %d\n", id_list.tail,
ENROLL_CONFIRM_TIMES - left_sample_face);
rgb_printf(image_matrix, FACE_COLOR_CYAN, "ID[%u] Sample[%u]",

id_list.tail, ENROLL_CONFIRM_TIMES - left_sample_face); if
(left_sample_face == 0){ is_enrolling = 0; Serial.printf("Enrolled Face
ID: %d\n", id_list.tail);
    }
  } else { matched_id = recognize_face(&id_list, if
aligned_face); (matched_id >= 0) {
```

```
            Serial.printf("Match Face ID: %u\n", matched_id);
rgb_printf(image_matrix, FACE_COLOR_GREEN, "Hello Subject %u",
matched_id);
          } else {
            Serial.println("No Match Found");
rgb_print(image_matrix, FACE_COLOR_RED, "Intruder Alert!");
matched_id = -1;
          }
        }
    } else {
        Serial.println("Face  Not Aligned");
        //rgb_print(image_matrix, FACE_COLOR_YELLOW, "Human Detected");
    }

    dl_matrix3du_free(aligned_face);      return
matched_id;
}

static size_t jpg_encode_stream(void * arg, size_t index, const void* data, size_t
len){ jpg_chunking_t *j = (jpg_chunking_t *)arg;
if(!index){        j->len = 0;
    }
    if(httpd_resp_send_chunk(j->req,  (const char *)data, len) != ESP_OK){                return
0;
    }j->len
+= len;
return len;
}

static esp_err_t capture_handler(httpd_req _t
*req){ camera_fb_t * fb = NULL; esp_err_t
```

```
res = ESP_OK; int64_t fr_start =
esp_timer_get_time();

    fb = esp_camera_fb_get();
    if (!fb) {
        Serial.println("Camera capture failed"); httpd_resp_send_500(req) ;
return
ESP_FAIL;
    }

    httpd_resp_set_type(req, "image/jpeg"); httpd_resp_set_hdr(req, "Content-
Disposition", "inline; filename=capture.jpg"); httpd_resp_set_hdr(req,
"Access-Control-Allow-Origin", "*");

    size_t out_len, out_width,
out_height; uint8_t * out_buf; bool
s; bool detected = false; int
face_id = 0;
    if(!detection_enabled || fb->width > 400){
size_t fb_len = 0; if(fb->format ==
PIXFORMAT_JPEG){
            fb_len = fb->len; res = httpd_resp_send(req,
(const char *)fb->buf, fb->len);
        } else { jpg_chunking_t jchunk = {req, 0}; res
= frame2jpg_cb(fb, 80, jpg_encode_stream,
&jchunk)?ESP_OK:ESP_FAIL;
            httpd_resp_send_chunk(req, NULL, 0); fb_len
= jchunk.len;
        }
        esp_camera_fb_return(fb);int64_t
fr_end = esp_timer_get_time();
```

```
      Serial.printf("JPG: %uB %ums\n", (uint32_t)(fb_len), (uint32_t)((fr_end -
fr_start)/1000));       return res;
   }

   dl_matrix3du_t *image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);
if    (!image_matrix)    {                                esp_camera_fb_return(fb);
Serial.println("dl_matrix3du_alloc   failed");              httpd_resp_send_500(r eq);
return ESP_FAIL;
   }

   out_buf    =    image_matrix->item;
out_len  = fb->width * fb->height * 3;
out_width = fb->width;     out_height =
fb->height;

   s    = fmt2rgb888(fb->buf, fb->len, fb->format,
out_buf); esp_camera_fb_return(fb);
   if(!s){

dl_matrix3du_free(image_matrix);
Serial.println("to     rgb888     failed");
httpd_resp_send_500(req);       return
ESP_FAIL;
   }

   box_array_t *net_boxes = face_detect(image_matrix, &mtmn_config);

   if (net_boxes){       detected = true;
if(recognition_enabled){           face_id =
run_face_recognition(image_matrix, net_boxes);
      }
```

```
        draw_face_boxes(image_matrix, net_boxes, face_id);            free(net_boxes-
>score);          free(net_boxes->box);          free(net_boxes->landmark);
free(net_boxes);
    }

    jpg_chunking_t jchunk = {req, 0};       s = fmt2jpg_cb(out_buf, out_len, out_width,
out_height,      PIXFORMAT_RGB888,            90,   jpg_encode_stream,   &jchunk);
dl_matrix3du_free(image_matrix);if(!s){
        Serial.println("JPEG compression failed");           return
ESP_FAIL;
    }

    int64_t fr_end = esp_timer_get_time();
    Serial.printf("FACE:  %uB      %ums      %s%d\n", (uint32_t)(jchunk.len),
(uint32_t)((fr_end - fr_start)/1000), detected?"DETECTED ":"", face_id);
return res;
}

static   esp_err_t   stream_handler(httpd_req _t
*req){      camera_fb_t * fb = NULL;          esp_err_t
res  = ESP_OK;              size_t       _jpg_buf_len  = 0;
uint8_t * _jpg_buf = NULL;        char * part_buf[64];
dl_matrix3du_t  *image_matrix = NULL;          bool
detected  = false;        int face_id = 0;          int64_t
fr_start = 0;          int64_t fr_ready = 0;          int64_t
fr_face = 0;        int64_t fr_recognize = 0;      int64_t
fr_encode = 0;

    static int64_t last_frame = 0;
if(!last_frame) {          last_frame =
esp_timer_get_time();
```

```
    }

    res = httpd_resp_set_type(req,
_STREAM_CONTENT_TYPE);          if(res != ESP_OK){              return res;

    }

    httpd_resp_set_hdr(req,  "Access-Control-Allow-Origin", "*");

    while(true){         detected = false;
face_id = 0;          fb
= esp_camera_fb_get();
        if (!fb) {
            Serial.println("Camera capture failed");               res
= ESP_FAIL;
        } else {              fr_start =
esp_timer_get_time();              fr_ready =
fr_start;              fr_face = fr_start;
fr_encode = fr_start;              fr_recognize =
fr_start;              if(!detection_enabled || fb-
>width > 400){                  if(fb->format !=
PIXFORMAT_JPEG){
                    bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
esp_camera_fb_return(fb);                    fb = NULL;                    if(!jpeg_converted){
                        Serial.println("JPEG compression failed");                          res
= ESP_FAIL;
                    }
                } else {
                    _jpg_buf_len = fb->len;
                    _jpg_buf = fb->buf;
                }
```

```
        } else {

            image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);

            if (!image_matrix) {
                Serial.println("dl_matrix3du_alloc  failed");                    res
= ESP_FAIL;
            } else {
                if(!fmt2rgb888(fb->buf, fb->len, fb->format, image_matrix->item)){
Serial.println("fmt2rgb888  failed");                    res = ESP_FAIL;
                } else {                        fr_ready = esp_timer_get_time();
box_array_t *net_boxes = NULL;                        if(detection_enabled){
net_boxes = face_detect(image_matrix, &mtmn_config);
                    }                        fr_face = esp_timer_get_time();
fr_recognize = fr_face;                        if (net_boxes || fb->format
!= PIXFORMAT_JPEG){                        if(net_boxes){                        detected
= true;                        if(recognition_enabled){
                            face_id = run_face_recognition(image_matrix,  net_boxes);
                        }                        fr_recognize =
esp_timer_get_time();
draw_face_boxes(image_matrix, net_boxes, face_id);
free(net_boxes->score);                                                free(net_boxes->box);
free(net_boxes->landmark);                        free(net_boxes);
                    }
                    if(!fmt2jpg(image_matrix->item, fb->width*fb->height*3, fb->width, fb-
>height, PIXFORMAT_RGB888, 90, &_jpg_buf, &_jpg_buf_len)){
                        Serial.println("fmt2jpg failed");                        res
= ESP_FAIL;
                    }
                    esp_camera_fb_return(fb);                        fb
= NULL;
```

```
                } else {
                    _jpg_buf = fb->buf;
                    _jpg_buf_len = fb->len;
                }
                fr_encode = esp_timer_get_time();
            }
            dl_matrix3du_free(image_matrix);
        }
    }
    }
    if(res == ESP_OK){ res = httpd_resp_send_chunk(req,
_STREAM_BOUNDARY, strlen(_STREAM_BOUNDARY));
    }
    if(res == ESP_OK){ size_t hlen = snprintf((char *)part_buf, 64,
_STREAM_PART, _jpg_buf_len); res = httpd_resp_send_chunk(req, (const
char *)part_buf, hlen);
    }
    if(res == ESP_OK){ res = httpd_resp_send_chunk(req, (const char
*)_jpg_buf, _jpg_buf_len);
    } if(fb){
esp_camera_fb_return(fb);
fb = NULL;
      _jpg_buf = NULL;
}  else  if(_jpg_buf){
free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if(res != ESP_OK){ break;
    }
    int64_t fr_end = esp_timer_get_time();
```

```
            int64_t   ready_time   =   (fr_ready   -   fr_start)/1000;
int64_t face_time = (fr_face - fr_ready)/1000;              int64_t
recognize_time = (fr_recognize - fr_face)/1000;          int64_t
encode_time = (fr_encode - fr_recognize)/1000;          int64_t


process_time = (fr_encode - fr_start)/1000;


      int64_t frame_time = fr_end - last_frame;          last_frame
= fr_end;         frame_time /=
1000;
      uint32_t avg_frame_time = ra_filter_run(&ra_filter, frame_time);
      Serial.printf("MJPG:  %uB  %ums  (%.1ffps),  AVG:  %ums  (%.1ffps),
%u+%u+%u+%u=%u %s%d\n",
          (uint32_t)(_jpg_buf_len),
          (uint32_t)frame_time, 1000.0 / (uint32_t)frame_time,            avg_frame_time,
1000.0 / avg_frame_time,
          (uint32_t)ready_time, (uint32_t)face_time, (uint32_t)recognize_time,
(uint32_t)encode_time, (uint32_t)process_time,
          (detected)?"DETECTED ":"", face_id
      );
   }

   last_frame = 0; return
   res;
}


static esp_err_t cmd_handler(httpd_req_t *req){
char* buf; size_t buf_len; char variable[32] = {0,};
char value[32] = {0,};


   buf_len = httpd_req_get_url_query_len(req)   +
1;    if (buf_len > 1) {         buf =
(char*)malloc(buf_len);
```

```
        if(!buf){
httpd_resp_send_500(req);
return ESP_FAIL;
        }
        if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) { if
(httpd_query_key_value(buf, "var", variable, sizeof(variable)) == ESP_OK
&&
                httpd_query_key_value(buf, "val", value, sizeof(value)) == ESP_OK) {
            } else {                    free(buf);
httpd_resp_send_404(req);
return ESP_FAIL;
            }
        } else {                free(buf);
httpd_resp_send_404(req);
return ESP_FAIL;
        }
free(buf);
    }    else   {
httpd_resp_send_404(req);
return ESP_FAIL;
    }

    int val = atoi(value); sensor_t * s = esp_camera_sensor_get(); int
res = 0;

    if(!strcmp(variable, "framesize")) {          if(s->pixformat ==
PIXFORMAT_JPEG) res = s->set_framesize(s, (framesize_t)val);
    }
    else if(!strcmp(variable, "quality")) res = s->set_quality(s, val);         else
if(!strcmp(variable, "contrast"))  res = s->set_contrast(s,  val);             else
if(!strcmp(variable, "brightness")) res = s->set_brightness(s,  val);          else
```

```
if(!strcmp(variable, "saturation")) res = s->set_saturation(s,  val);       else
if(!strcmp(variable,     "gainceiling"))     res     =     s->set_gainceiling(s,
(gainceiling_t)val);     else if(!strcmp(variable, "colorbar")) res = s-
>set_colorbar(s, val);      else if(!strcmp(variable, "awb")) res = s-
>set_whitebal(s, val);      else if(!strcmp(variable, "agc")) res = s>set_gain_ctrl(s,
val);     else if(!strcmp(variable, "aec")) res = s>set_exposure_ctrl(s,  val);       else
if(!strcmp(variable, "hmirror")) res = s->set_hmirror(s, val);       else
if(!strcmp(variable, "vflip")) res = s>set_vflip(s, val);      else if(!strcmp(variable,
"awb_gain")) res = s>set_awb_gain(s,  val);       else if(!strcmp(variable,
"agc_gain")) res = s>set_agc_gain(s, val);       else if(!strcmp(variable,
"aec_value")) res = s-
>set_aec_value(s, val);      else if(!strcmp(variable, "aec2")) res = s-
>set_aec2(s, val);      else if(!strcmp(variable, "dcw")) res = s-
>set_dcw(s, val);    else if(!strcmp(variable, "bpc")) res = s-
   >set_bpc(s, val); else if(!strcmp(variable, "wpc")) res = s-
   >set_wpc(s, val);
   else  if(!strcmp(variable,  "raw_gma"))  res  = s->set_raw_gma(s,     val);          else
if(!strcmp(variable, "lenc")) res = s->set_lenc(s, val);       else if(!strcmp(variable,
"special_effect"))  res  =  s->set_special_effect(s,    val);             else
if(!strcmp(variable, "wb_mode")) res = s->set_wb_mode(s, val);       else
if(!strcmp(variable, "ae_level"))  res = s->set_ae_level(s,  val);          else
if(!strcmp(variable, "face_detect")) {              detection_enabled  = val;
if(!detection_enabled) {            recognition_enabled  = 0;
      }
   }
   else if(!strcmp(variable, "face_enroll")) is_enrolling = val;          else
if(!strcmp(variable, "face_recognize")) {
recognition_enabled  = val;        if(recognition_enabled){             detection_enabled
= val;
```

```
        }    }
else {
res = -1;
   }if(res){
       return httpd_resp_send_500(req);
   }

   httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");       return
httpd_resp_send(req, NULL, 0);
}

static esp_err_t status_handler(httpd_req_t *req){       static
char json_response[1024];

   sensor_t * s = esp_camera_sensor_get();          char
* p = json_response;
   *p++ = '{';

   p+=sprintf(p, "\"framesize\":%u,", s->status.framesize);       p+=sprintf(p,
"\"quality\":%u,", s->status.quality);       p+=sprintf(p,
"\"brightness\":%d,", s->status.brightness);      p+=sprintf(p,
"\"contrast\":%d,", s->status.contrast);      p+=sprintf(p,
"\"saturation\":%d,", s->status.saturation);      p+=sprintf(p,
"\"sharpness\":%d,", s->status.sharpness);       p+=sprintf(p,
"\"special_effect\":%u,", s->status.special_effect);
p+=sprintf(p, "\"wb_mode\":%u,", s->status.wb_mode);
p+=sprintf(p, "\"awb\":%u,", s->status.awb);       p+=sprintf(p,
"\"awb_gain\":%u,", s->status.awb_gain);       p+=sprintf(p,
"\"aec\":%u,", s->status.aec);       p+=sprintf(p, "\"aec2\":%u,", s-
>status.aec2);      p+=sprintf(p, "\"ae_level\":%d,", s>status.ae_level);
p+=sprintf(p, "\"aec_value\":%u,", s-
```

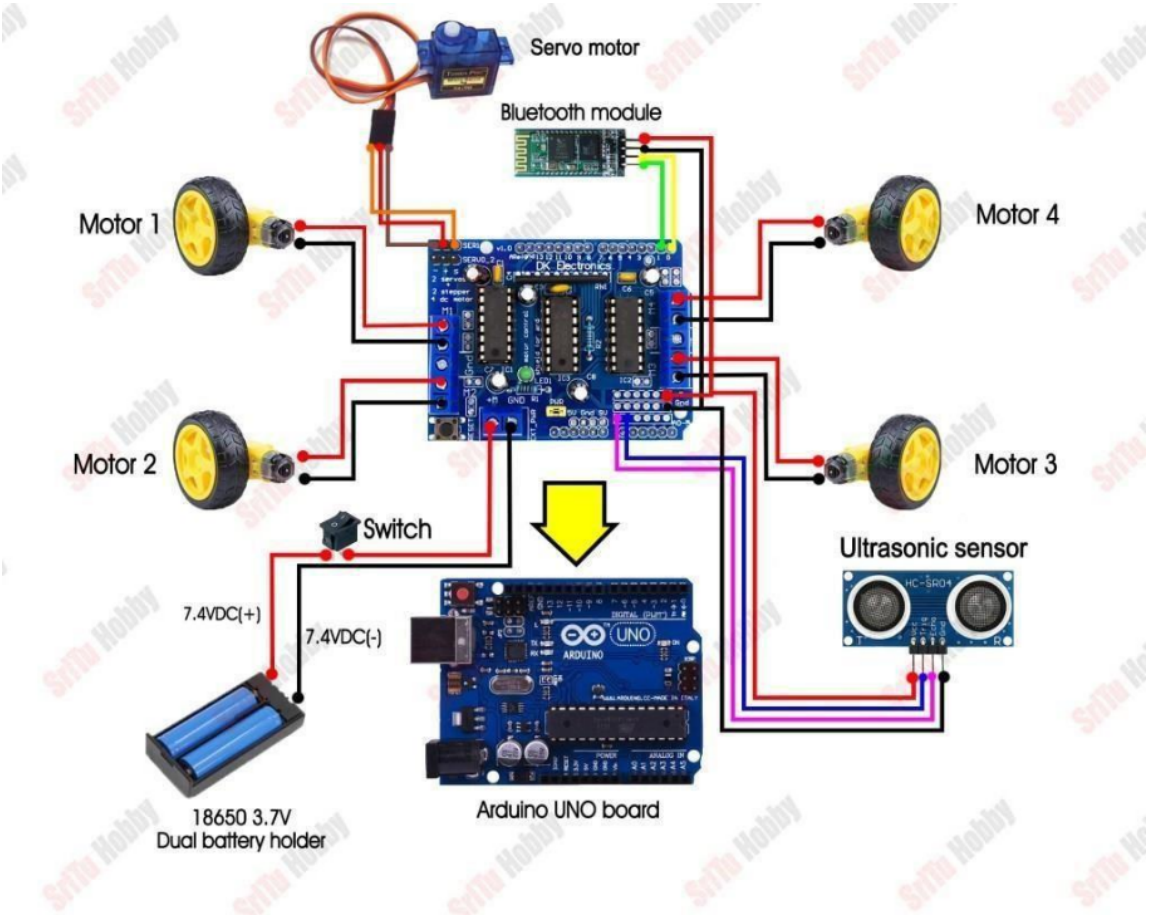
```
>status.aec_value);      p+=sprintf(p, "\"agc\":%u,", s-
>status.agc);        p+=sprintf(p, "\"agc_gain\":%u,",  s>status.agc_gain);
p+=sprintf(p, "\"gainceiling\":%u,",  s-
>status.gainceiling);                p+=sprintf(p,       "\"bpc\":%u,",
s>status.bpc);                  p+=sprintf(p,    "\"wpc\":%u,",       s-
>status.wpc);                  p+=sprintf(p,  "\"raw_gma\":%u,",  s-
>status.raw_gma);              p+=sprintf(p,   "\"lenc\":%u,",      s-
>status.lenc);        p+=sprintf(p, "\"vflip\":%u,", s->status.vflip);
p+=sprintf(p,         "\"hmirror\":%u,",            s->status.hmirror);
p+=sprintf(p,   "\"dcw\":%u,",  s->status.dcw);       p+=sprintf(p,
"\"colorbar\":%u,", s->status.colorbar);
    p+=sprintf(p,  "\"face_detect\":%u,",    detection_enabled);
p+=sprintf(p,           "\"face_enroll\":%u,",            is_enrolling);
p+=sprintf(p,                                  "\"face_recognize\":%u",
recognition_enabled);
    *p++ = '}';      *p++ = 0;         httpd_resp_set_type(req,
"application/json");         httpd_resp_set_hdr(req,  "Access-Control-AllowOrigin",
"*");   return httpd_resp_send(req,  json_response,        strlen(json_response));
}

static esp_err_t index_handler(httpd_req_t    *req){       httpd_resp_set_type(req,
"text/html");    httpd_resp_set_hdr(req,
"Content-Encoding",      "gzip");                    sensor_t    *       s    =
esp_camera_sensor_get();                  if (s->id.PID == OV3660_PID) {
return              httpd_resp_send(req,              (const              char
*)index_ov3660_html_gz, index_ov3660_html_gz_len);
    }
    return httpd_resp_send(req, (const char *)index_ov2640_html_gz,
index_ov2640_html_gz_len);
}
```

```
void startCameraServer(){ httpd_config_t config =
HTTPD_DEFAULT_CONFIG();

    httpd_uri_t index_uri = {
        .uri       = "/",
        .method    = HTTP_GET,
        .handler   = index_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t status_uri = {
        .uri       = "/status",
        .method    = HTTP_GET,
        .handler   = status_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t cmd_uri = {
        .uri       = "/control",
        .method    = HTTP_GET,
        .handler   = cmd_handler,
        .user_ctx  = NULL
    };

    httpd_uri_t capture_uri = {
        .uri       = "/capture",
        .method    = HTTP_GET,
        .handler   = capture_handler,
        .user_ctx  = NULL
    };
```

```
    httpd_uri_t stream_uri = {
        .uri       = "/stream",
        .method    = HTTP_GET,
        .handler   = stream_handler,
        .user_ctx  = NULL
    };


     ra_filter_init(&ra_filter, 20);

    mtmn_config.type                =               FAST;
mtmn_config.min_face                =               80;
mtmn_config.pyramid                 =               0.707;
mtmn_config.pyramid_times           =               4;
mtmn_config.p_threshold.score       =               0.6;
mtmn_config.p_threshold.nms         =               0.7;
mtmn_config.p_threshold.candidate_number = 20;
mtmn_config.r_threshold.score       =               0.7;
mtmn_config.r_threshold.nms         =               0.7;
mtmn_config.r_threshold.candidate_number = 10;
mtmn_config.o_threshold.score       =               0.7;
mtmn_config.o_threshold.nms = 0.7;
mtmn_config.o_threshold.candidate_number = 1;

          face_id_init(&id_list, FACE_ID_SAVE_NUMBER, ENROLL_CONFIRM_TIMES);

    Serial.printf("Starting web server on port: '%d'\n", config.server_port);
if      (httpd_start(&camera_httpd,      &config)      ==      ESP_OK)       {
httpd_register_uri_handler(camera_httpd,                        &index_uri);
httpd_register_uri_handler(camera_httpd,                         &cmd_uri);
```

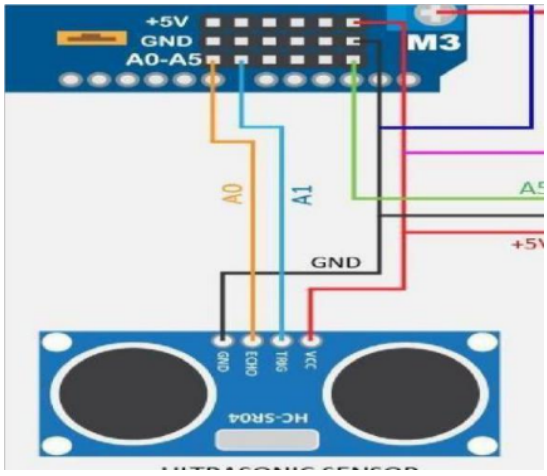
```
httpd_register_uri_handler(camera_httpd, &status_uri);
httpd_register_uri_handler(camera_httpd, &capture_uri);
    }

    config.server_port += 1;
config.ctrl_port += 1;
Serial.printf("Starting stream server on port:
'%d'\n", config.server_port);        if
(httpd_start(&stream_httpd,  &config) ==
ESP_OK) {
httpd_register_uri_handler(stream_httpd,
&stream_uri);
    }
}
```

CHAPTER 5: CIRCUIT DESIGNING



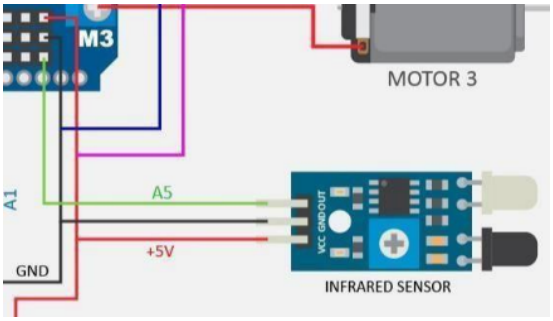
CIRCUIT DIAGRAMOFBLUETTOTH SURVALLIENCE ROBOT

ULTRASONIC SENSOR CONNECTIONS



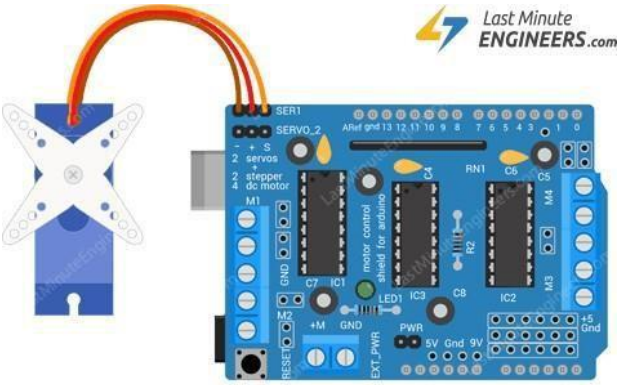
1. Connect Ultrasonic Sensor ’s ECHO PIN to A0 PIN of the motor driver shield.
2. Connect Ultrasonic Sensor ’s TRIGER PIN to A1 PIN of the motor driver shield.
3. Connect Ultrasonic Sensor ’s VCC PIN to 5Volts PIN of the motor driver shield.
4. Connect Ultrasonic Sensor ‘s GROUND PIN TO GROUNG PIN of the motor driver shield.

IR SENSOR CONNECTIONS



1. Connect the OUT PIN of IR Sensor to the A5 PIN of the motor driver shield.
2. Connect the VCC PIN of IR Sensor to the 5Vlots PIN of the motor driver shield.
3. Connect the GROUND PIN of IR Sensor to the GROUND PIN of the motor driver shield.

SERVO MOTOR CONNECTIONS

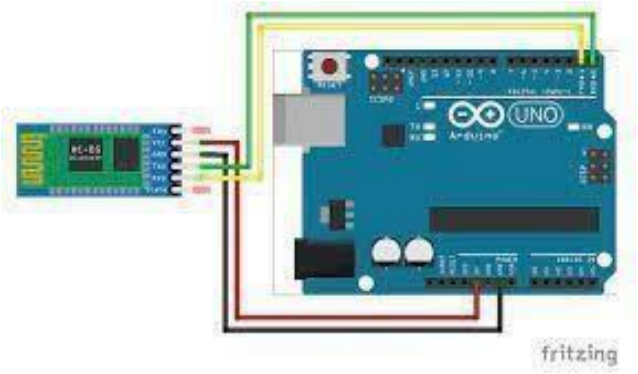


1. the red pin (5volts).
2. The orange pin (PWM).

3. The brown pin (ground).
4. Connect the pins according to the diagram.

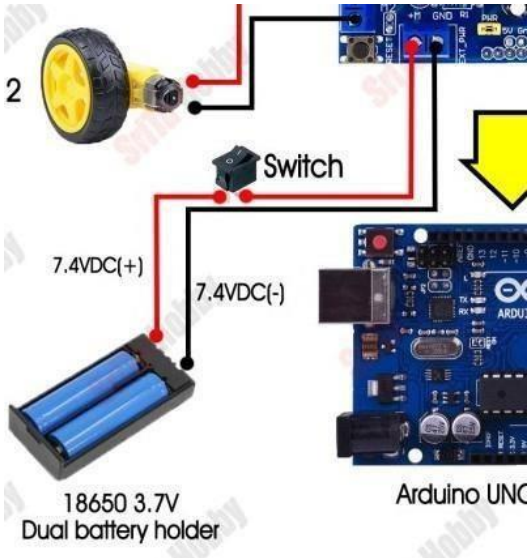
BLUETOOTH MODULE CONNECTIONS

1. Connect the RX PIN of Bluetooth Module to the TX of motor driver shield.
2. Connect the TX PIN of Bluetooth Module to the RX of the motor driver shield.
3. Connect VCC pin to the 5vlots pin of the motor driver shield.
4. Connect ground pin to the ground pin of the motor driver shield.

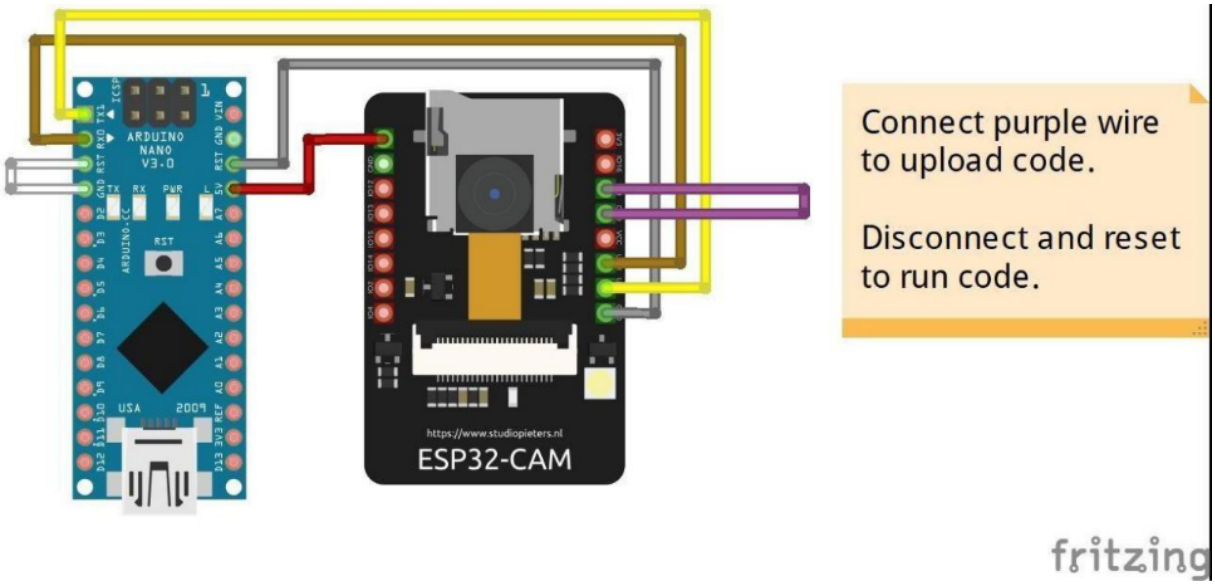


POWER SUPPLY

1. We are using to lithium batteries for power supply.



ESP 32 CAM CONNECTIONS



1. Short the GPI 0 pin with GND pin of ESP32
2. Connect the UOT OF ESP 32 to the TX pin of the NANO.
3. Connect the UOR OF ESP 32 to the RX pin of the NANO.
4. Connect the 5v pin of ESP 32 to the 5v pin of the NANO.
5. Connect the GND pin of ESP 32 to the GND pin of the NANO.
6. Short the Reset pin of NANO with GND of the NANO.

REFERENCES

The sites we used during our preparation of Documentation of project are as follows:

1. https://circuitdigest.com
2. https://en.wikipedia.org
3. https://www.melexis.com
4. https://www.arduino.com
5. https://www.google.com

Chapter 6
CONCLUSION

FUTURESCOPE

Surveillance is needed in almost every field . It could be a great solution to various problems or situation where wireless surveillance is needed.

Our project as tremendous scope as it uses latest technology in the market .

This robot's functionality is even voice controlled and wi-fi connected and the output can be a image or a video which can generate interest as they are all integrated into one entity.

ADVANTAGES

- VOICE CONTROLLED
- OBSTACLE AVOIDANCE
- REMOTE OPEARTIONS
- WI-FI CONNECTED "VIDEO"
- PRECISE MOVEMENT USING SERVO MOTOR

Why Arduino?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems.

Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

Simple, clear programming environment - The Arduino programming environment is easy-to use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino.