ABSTRACT

Autonomous robotic systems require advanced obstacle detection and navigation mechanisms to operate efficiently in dynamic environments. Traditional navigation methods using fixed ultrasonic sensors often result in limited detection range and inaccurate path selection. This project proposes an omnidirectional navigation system that integrates an ultrasonic sensor, a servo motor, and an L293D motor driver to optimize robotic movement. The ultrasonic sensor is mounted on a servo motor, allowing it to scan a 180-degree area and improve obstacle detection accuracy. The L293D motor driver controls the movement of the robotic vehicle, ensuring smooth navigation based on real-time sensor data. The system employs a mean-based algorithm to minimize false positives in obstacle detection and optimize pathfinding. This approach significantly improves robotic efficiency, making it suitable for applications such as industrial automation, search-and-rescue operations, and autonomous vehicles.

Keywords – L293D, Ultrasonic Sensor, 16×2 LCD, Servo Motor

# CHAPTER – 1

EMBEDDED SYTEMS

* 1. Introduction:

An embedded system is a special-purpose computer system designed to perform one or a few dedicated functions, sometimes with real-time computing constraints. It is usually embedded as part of a complete device including hardware and mechanical parts. In contrast, a general-purpose computer, such as a personal computer, can do many different tasks depending on programming. Embedded systems have become very important today as they control many of the common devices we use.

Since the embedded system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product, or increasing the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale

Physically embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

In general, “embedded system” is not an exactly defined term, as many systems have some element of programmability. For example, Handheld computers share some elements with embedded systems — such as the operating systems and microprocessors which power them — but are not truly embedded systems, because they allow different applications to be load and peripherals to be connected.

An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, that is specifically designed for a particular kind of application device. Industrial machines, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines, and toys (as well as the more obvious cellular phone and PDA) are among the myriad possible hosts of an embedded system. Embedded systems that are programmable are provided with a programming interface, and embedded systems programming is a specialized occupation. Certain operating systems or language platforms are tailored for the embedded market, such as Embedded Java and Windows XP Embedded. However, some low-end consumer products use very inexpensive microprocessors and limited storage, with the application and operating system both part of a single program. The program is written permanently into the system’s memory in this case, rather than being loaded into RAM (random access memory), as programs on a personal computer are

* 1. CHARACTERISTIC OF EMBEDDED SYSTEM
* Speed (bytes/sec): Should be high speed
* Power (watts): Low power dissipation
* Size and weight: As far as possible small in size and low weight
* Accuracy (%error): Must be very accurate
* Adaptability: High adaptability and accessibility
* Reliability: Must be reliable over a long period of time
  1. APPLICATIONS OF EMBEDDED SYSTEM

We are living in the Embedded World. You are surrounded with many embedded products and your daily life largely depends on the proper functioning of these gadgets. Television, Radio, CD player of your living room, Washing Machine or Microwave Oven in your kitchen, Card readers, Access Controllers, Palm devices of your work space enable you to do many of your tasks very effectively. Apart from all these, many controllers embedded in your car take care of car operations between the bumpers and most of the times you tend to ignore all these controllers.

* **Robotics:** industrial robots, machine tools, [Robocop](http://en.wikipedia.org/wiki/Robocup) soccer robots
* **Automotive:** cars, trucks, trains
* **Aviation:** airplanes, helicopters
* [Home and Building Automation](http://en.wikibooks.org/wiki/Embedded_Control_Systems_Design/Home_and_Building_Automation)
* **Aerospace:** rockets, satellites
* **Energy systems:** windmills, nuclear plants
* **Medical systems:** prostheses, revalidation machine.
  1. MICROCONTROLLER VERSUS MICROPROCESSOR

What is the difference between a Microprocessor and Microcontroller? By microprocessor is meant the general-purpose Microprocessors such as Intel’s X86 family (8086, 80286, 80386, 80486, and the Pentium) or Motorola’s 680X0 family (68000, 68010, 68020, 68030, 68040, etc). These microprocessors contain no RAM, no ROM, and no I/O ports on the chip itself. For this reason, they are commonly referred to as general-purpose Microprocessors.

A system designer using a general-purpose microprocessor such as the Pentium or the 68040 must add RAM, ROM, I/O ports, and timers externally to make them functional. Although the addition of external RAM, ROM, and I/O ports makes these systems bulkier and much more expensive, they have the advantage of versatility such that the designer can decide on the amount of RAM, ROM and I/O ports needed to fit the task at hand. This is not the case with Microcontrollers.

A Microcontroller has a CPU (a microprocessor) in addition to a fixed amount of RAM, ROM, I/O ports, and a timer all on a single chip. In other words, the processor, the RAM, ROM, I/O ports and the timer are all embedded together on one chip; therefore, the designer cannot add any external memory, I/O ports, or timer to it. The fixed amount of on-chip ROM, RAM, and number of I/O ports in Microcontrollers makes them ideal for many applications in which cost and space are critical.

In many applications, for example a TV remote control, there is no need for the computing power of a 486 or even an 8086 microprocessor. These applications most often require some I/O operations to read signals and turn on and off certain bits

* 1. MICROCONTROLLERS FOR EMBEDDED SYSTEMS

In the Literature discussing microprocessors, we often see the term Embedded System. Microprocessors and Microcontrollers are widely used in embedded system products. An embedded system product uses a microprocessor (or Microcontroller) to do one task only. A printer is an example of embedded system since the processor inside it performs one task only; namely getting the data and printing it. Contrast this with a Pentium based PC. A PC can be used for any number of applications such as word processor, print-server, bank teller terminal, Video game, network server, or Internet terminal. Software for a variety of applications can be loaded and run. Of course, the reason a pc can perform myriad tasks is that it has RAM memory and an operating system that loads the application software into RAM memory and lets the CPU run it.

In this robot as the fire sensor senses the fire, it senses the signal to microcontroller. In an Embedded system, there is only one application software that is typically burned into ROM. An x86 PC contains or is connected to various embedded products such as keyboard, printer, modem, disk controller, sound card, CD-ROM drives, mouse, and so on. Each one of these peripherals has a Microcontroller inside it that performs only one task.

# CHAPTER 2

PROJECT OVERVIEW

2.1 Introduction to Project:

**A**. Background: In this modern world, pathfinding algorithms play a vital role in robotics and automation. However, commonly used pathfinding algorithms for ultrasonic sensors are limited by the monodirectional nature of the sensor. They often rely on simple binary directional decisions (left and right), which is inaccurate [1]. This paper aims to enhance the existing algorithms. These algorithms are crucial for finding paths and avoiding obstacles that come the car’s way. They have various ap plications in the military to reach inaccessible or dangerous areas. Moreover, they have applications in obstacle avoidance systems on extraterrestrial surfaces with an atmosphere. Suresh Kumar Gawre Assistant Professor Department of Electrical engineering MANIT Bhopal sgawre28@gmail.com Fig. 1. Circuit Diagram

**B**. Motivation The motivation behind this project stems from the limita tions of the monodirectional nature of the ultrasonic sensor. This does not provide an accurate way of finding the ideal path as it gives false positives (detailed in the Algorithm section). While other wide-angle methods exist, they tend to be expensive and complicated. This motivated us to create a composite sensor using an ultrasonic sensor mounted on a servo motor. We formulated an algorithm for omnidirectional pathfinding and obstacle detection alongside this sensor.

**C**. Paper Organization In the following section, we’ll give a basic overview of our project, Section II tells us about the design and hardware requirements. Subsequently, Section III tells us about the algorithm and its working principle. Section IV will describe the software requirement and control mechanism, emphasizing their important role. To conclude our discussion, Section V tells us about the advantages, real-world applications, and future potential of the project, meanwhile, Section VI will provide a detailed summary of our discoveries and inputs. Refer to Figures 12 and 13, showing the block diagram and the final model, respectively. Our overall methodology revolves around omni directional pathfinding using a mean-based algorithm, utilizing a single ultrasonic sensor

2.1.1 Existing System:

Conventional robotic navigation systems rely on static ultrasonic sensors that provide monodirectional obstacle detection, limiting their effectiveness in complex environments. These systems suffer from blind spots and inaccurate obstacle readings due to signal reflections, leading to frequent path recalculations and inefficient movement. Additionally, traditional robotic platforms lack servo motor-driven scanning, preventing adaptive obstacle detection. The absence of a motor control mechanism like the L293D driver results in imprecise movement, causing navigation inefficiencies and suboptimal path execution. The limitations of these traditional systems highlight the need for a more advanced robotic navigation approach that integrates dynamic obstacle detection and precise motor control.

2.1.2 Proposed System:

The proposed system represents a significant advancement in robotic navigation by integrating three critical components: an ultrasonic sensor, a servo motor, and an L293D motor driver. The ultrasonic sensor is intelligently mounted on a servo motor, enabling it to scan a wide **180-degree field of view**, which provides enhanced obstacle detection capabilities. This setup allows the system to gather detailed spatial information about its surroundings, ensuring a comprehensive understanding of potential obstacles.

The collected data is processed through a **mean-based algorithm**, a method designed to improve the efficiency and accuracy of path selection. This algorithm minimizes false positives, ensuring reliable and precise navigation even in complex or dynamic environments.

The L293D motor driver serves as the backbone for controlling the robotic vehicle's movement. It enables **precise and responsive control** of the motors, allowing smooth transitions and seamless execution of calculated paths. This level of control enhances the overall stability and efficiency of the robot, making it capable of performing intricate maneuvers with ease.

These features make the system highly suitable for a variety of applications, including **industrial automation, search-and-rescue missions, and autonomous robotic systems**. Its versatility and reliability ensure it can excel in both structured and unstructured environments, addressing a wide range of real-world challenges.

2.2 Block Diagram:

**Arduino UNO**

**Power Supply**

**Ultrasonic Sensor**

**16×2 LCD**

**Servo Motor**

**L293D**

**M1**

**M2**

Fig 2.1: Block diagram of Omni-Directional Robo Using Single Ultrasonic Sensor

# CHAPTER - 3

HARDWARE DESCRIPTION

3.1 Arduino UNO

3.1.1 Introduction to Microcontroller

Microcontroller as the name suggest, a small controller. They are like single chip computers that are often embedded into other systems to function as processing/controlling unit. For example, the control you are using probably has microcontrollers inside that do decoding and other controlling functions. They are also used in automobiles, washing machines, microwaves ovens, toys, etc., where automation is needed.

3.1.2 Arduino UNO Microcontroller:

The Arduino Uno is a microcontroller board based on the Atmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that It does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter. “Uno” means “One” in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board’s power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5Vpin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

* **VIN.** The input voltage to the Arduino board when it’s using an external power source (as opposed to5 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. The** regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
* **3.3V.**A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
* **GND.** Ground pins

3.1.2.1 Memory:

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library)..

3.1.2.2 Input and Output:

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

* **Serial: 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: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, arising or falling edge, or a change in value. See the attach Interrupt() function for details.
* **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
* **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which although provided by the underlying hardware, is not currently included in the Arduino language.
* **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it’s off.
* The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:
* **I2C: 4 (SDA) and 5 (SCL).** Support I2C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

* **AREF.** Reference voltage for the analog inputs. Used with analogReference().
* **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

3.1.2.3 Communication:

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The Atmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An Atmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The ‘8U2 firmware uses the standard USBCOM drivers, and no external driver is needed. However, on Windows, an \*.inf file is required. 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 USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A Software Serial library allows for serial communication on any of the Uno’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

3.1.3 Arduino UNO Board:

The Arduino Uno is a microcontroller board based on the Atmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converters



Fig 3.1: Arduino Uno board

3.1.3.1 Technical Specifications:

Table 3.1. Arduino UNO specifications

|  |  |
| --- | --- |
| **Features** | **Specifications** |
| Microcontroller | ATmega328 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limits) | 6-20V |
| Digital I/O Pins | 14(of which 6 provide PWM output) |
| Analog Input Pins | 6 |
| DC Current pere I/O pin | 40 mA |
| DC Current for 3.3v pin | 50 mA |
| Flash Memory | 32 KB(ATmega328) of which 0.5KB used by boot loader |
| SRAM | 2 KB(ATmega328) |
| EEPROM | 1 KB(ATmega328) |
| Clock Speed | 16 MHz |

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board’s power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

* **USB Interface:**

Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection

* **External power supply:**

Arduino boards can be powered directly from the AC mains power supply by connecting it to the power supply (Barrel Jack).

* **Voltage Regulator:**

The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

* **Crystal Oscillator:**

The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz

* **Arduino Reset:**

It can reset your Arduino board, i.e., start your program from the beginning. It can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

* **Pins (3.3, 5, GND, Vin):**
* 3.3V (6): Supply 3.3 output volt
* 5V (7): Supply 5 output volt
* Most of the components used with Arduino board works fine with 3.3 volt

and 5 volt.

* GND (8)(Ground): There are several GND pins on the Arduino, any of which can be used to ground your circuit.
* Vin (9): This pin also can be used to power the Arduino board from an

external power source, like AC mains power supply.

* **Analog pins:**

The Arduino UNO board has five analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.

* **Main microcontroller:**
* Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.

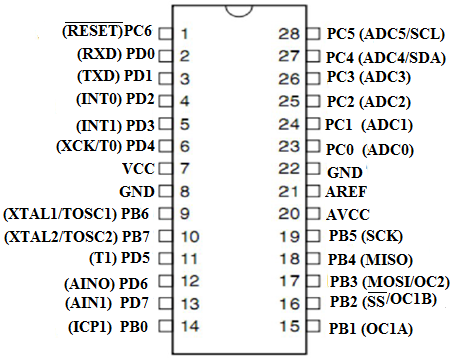


Fig 3.2: Pin diagram

The Atmega8U2 programmed as a USB-to-serial converter. “Uno” means “One” in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards

3.1.3.2 Pin Description:

* **VCC:** Digital Supply Voltage.
* **GND:** Ground.
* **Port B (PB[7:0]) XTAL1/XTAL2/TOSC1/TOSC2:** Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.
* **Port C (PC[5:0]):** Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC[5:0] output buffers have symmetrical drive characteristics with both high sink and source capability.
* **PC6/RESET:** If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a Reset.
* **Port D (PD[7:0]):** Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.
* **AVCC:** AVCC is the supply voltage pin for the A/D Converter, PC[3:0], and PE[3:2]. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC[6:4] use digital supply voltage, VCC.
* **ADC [7:6] (TQFP and VFQFN Package Only):** In the TQFP and VFQFN package, ADC[7:6] serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.
* **12. ICSP pin:** Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an “expansion” of the output. Actually, you are slaving the output device to the master of the SPI bus
* **15. Digital I / O:** The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM.
* **AREF:** AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins working.

# CHAPTER – 4

HARDWARE COMPONENTS