Missile Detection and Destruction System

Project Team

Sl. No.	Reg. No.	Student Name
1.	18ETEE003022	Nuthan S M L
2.	18ETEE003027	Rahul Umesh H S
3.	18ETEE003035	Vipul Kumar
4.	18ETEE003038	Vrushabh B Padasalagi

Supervisors

1. Dr. S Nagaraja Rao
2. Mr. Sachin S

January – 2022

B.Tech

FACULTY OF ENGINEERING AND TECHNOLOGY
M. S. RAMAIAH UNIVERSITY OF APPLIED SCIENCES
Bengaluru -560 054

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate

This is to certify that the Project titled "Missile Detection and Destroying System" is a bonafide work carried out in the Department of Electrical and Electronics Engineering by Mr. Nuthan S M L, Mr. Rahul Umesh H S, Mr. Vipul Kumar, Mr. Vrushabh B Padasalagi, bearing Reg. No. 18ETEE003022, 18ETEE003027, 18ETEE003035,18ETEE003038 in partial fulfilment of requirements for the award of B. Tech. Degree in Electrical Engineering of M.S. Ramaiah University of Applied Sciences.

January – 2022

Supervisors: 1. **Dr. Nagaraja Rao S** **2. Mr. Sachin S**
Assistant professors, Department of EEE, RUAS

Dr. S Malathi
Head – Dept. of EEE

Dr.Govind R. Kadambi
Dean-FET

Declaration

Missile Detection and Destroying System

The project work is submitted in partial fulfilment of academic requirements for the award of B. Tech. Degree in the Department of Electrical and Electronics Engineering of the Faculty of Engineering and Technology of M. S. Ramaiah University of Applied Sciences. The project report submitted herewith is a result of our own work and in conformance to the guidelines on plagiarism as laid out in the University Student Handbook. All sections of the text and results which have been obtained from other sources are fully referenced. We understand that cheating and plagiarism constitute a breach of University regulations, hence this project report has been passed through plagiarism check and the report has been submitted to the supervisor.

Sl. No.	Reg. No.	Student Name	Signature
1.	18ETEE003022	Nuthan S M L	
2.	18ETEE003027	Rahul Umesh H S	
3.	18ETEE003035	Vipul Kumar	
4.	18ETEE003038	Vrushabh B Padasalagi	

Date: **13 January 2022**

Acknowledgement

We would like to express our most profound thanks to each one of the individuals who gave us the likelihood to finish this product. We wish to give a special gratitude we give to our project supervisors, Dr. S Nagaraja Rao, assistant professor and Mr. Sachin S, assistant professor of electrical and electronic engineering department, whose commitment in invigorating proposals and consolation, helped us to organize our venture, particularly in composing this simulation results.

We are thankful to Dr. S.Malathi, H.O.D of Electrical and Electronics Engineering, for providing us an opportunity to simulate various topologies using various techniques and obtain the best results.

Our grateful regards to Dr. Srikanth Dutt, Professor and Associate Dean and Dr.Govind R.Kadambi, Dean, Faculty of Engineering and Technology, Ramaiah University of Applied Sciences, Bengaluru, for their constant support and motivation.

We take this opportunity to thank all our lectures and staff of Electrical Laboratory, who have directly or indirectly helped in our project. We pay our respects and love to our parents and all other family members and friends for their love and encouragement throughout our career. Last, but not the least we express our thanks to our friends for their cooperation and support.

Contents

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

List of Figures

List of Tables

Abstract

An ultrasonic module is connected to a microcontroller from the 8051 series in this system. A transmitter and receiver are employed in an ultrasonic transducer. The item reflects the sent waves back to the transducer, which receives them again. Taking into account the velocity of sound, the total time taken from sending the waves to receiving them is estimated. The distance is then calculated by a microcontroller programme and shown on a liquid crystal display screen connected to the microcontroller. The circuit is used to receive 40 KHz reflected signals from the missile object, feed them into a microcontroller programme, and turn on the necessary load while the programme is being executed. When the ultrasonic receiver sends a signal to the microcontroller, it activates the door cannon by triggering the gate of a MOSFET via a transistor or relay. The sensor is attached to the antenna and rotated and operated by a servo motor in a 180-degree rotation. If a target is discovered within the detection range, the programme will direct the launcher to the closest detected target and fire, applying a laser beam to it.

Chapter -1

1.1 Introduction

To detect missile objects, the suggested system uses an ultrasonic module connected to an Arduino family microcontroller. On the same module, an ultrasonic transducer with a transmitter and receiver is employed. Sound waves are produced by the ultrasonic transducer. The item reflects the sent sound waves back to the transducer, which receives them again. Taking into account the velocity of sound, the total time taken from sending the waves to receiving them is estimated. The distance is then calculated and shown on a

microcontroller-interfaced liquid crystal display. When the ultrasonic receiver sends a signal to the microcontroller, it activates the door cannon by triggering the gate of a MOSFET via a transistor or relay. The sensor is mounted on the antenna and rotates and is operated by a stepper motor in a 360-degree rotation. If a target is detected within the detection range, the application will turn the launcher to that target and fire. The antenna is rotated and controlled by a stepper motor on one axis, and it rotates up and down in the direction of the missile object on another axis. The tank vehicle is equipped with a second microcontroller that sends and receives control actions from the key panel via wireless zigbee connection. Embedded C programming using arduino software is used to create programmes for the Arduino series of microcontrollers.

1.2 Proposed system

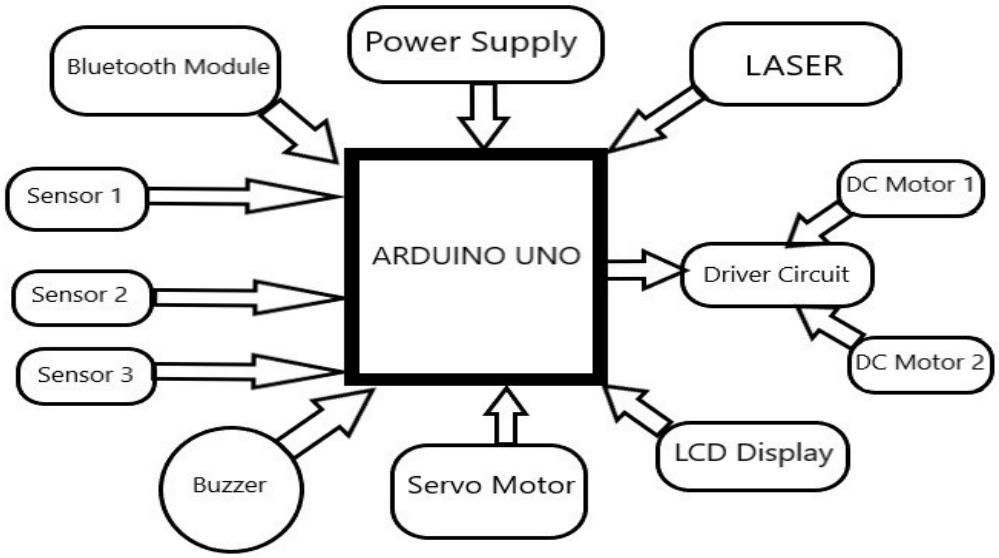


Figure 1: proposed diagram

1.3 Working methodology

To detect missile objects, the suggested system uses an ultrasonic module connected to an Arduino family microcontroller. On the same module, an ultrasonic transducer with a transmitter and receiver is employed. Sound waves are produced by the ultrasonic transducer. The item reflects the sent sound waves back to the transducer, which receives them again. Taking into account the velocity of sound, the total time taken from sending the waves to receiving them is estimated. The distance is then calculated and shown on a microcontroller-interfaced liquid crystal display. When the microcontroller gets the signal

from the ultrasonic receiver, it activates the door cannon by triggering the laser beam gate.

Chapter – 2

2.1 Component requirements

Hardware requirements

1.Arduino uno

2.Bluetooth hc05

3.Ultrasonic sensor (HY – SRF05)

4.servo motor

5.laser light

6.driver circuit

7.dc motor

8.buzzer

9.lcd display

2.2 Overview

2.2.1 Arduino UNO

To detect missile objects, the suggested system uses an ultrasonic module connected to an Arduino family microcontroller. On the same module, an ultrasonic transducer with a transmitter and receiver is employed. Sound waves are produced by the ultrasonic transducer. The item reflects the sent sound waves back to the transducer, which receives them again. Taking into account the velocity of sound, the total time taken from sending the waves to receiving them is estimated. The distance is then calculated and shown on a microcontroller-interfaced liquid crystal display. When the microcontroller gets the signal from the ultrasonic receiver, it activates the door cannon by triggering the laser beam gate.

It has a multitude of sensors and can control lights, motors, and other actuators to impact its surroundings. The Arduino programming language (based on Wiring) and

the Arduino development environment are used to programme the microcontroller on the board (based on Processing). Arduino projects can be self-contained or communicate with computer software (e.g., Flash, Processing, or Max/MSP).

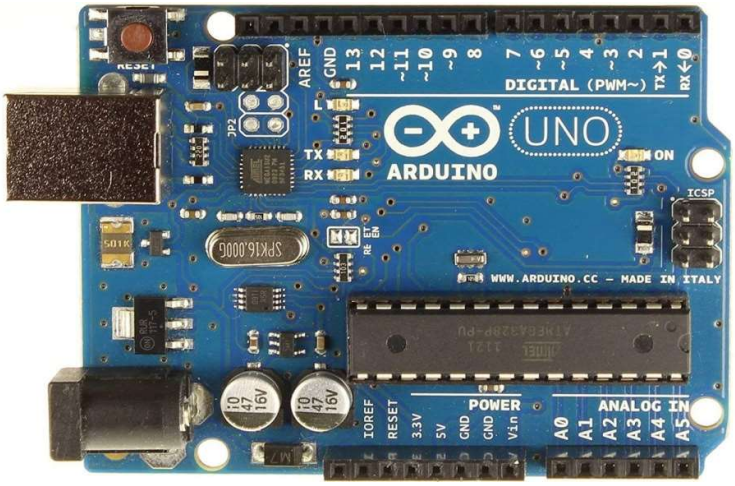


Figure 2.1 Arduino UNO

2.2.2Technical details

- Dimensions (maximum): 75.14 x 53.51 x 15.08mm
- Weight: 27.95g/0.98oz
- Microcontroller: Atmega328
- Operating Voltage: 5V,3.3V
- 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 per I/O Pin: 40 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB (ATmega328) of which 0.5 KB used by bootloader

- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz

The Arduino Uno is a microcontroller board that uses the ATmega328 microcontroller (datasheet). There are 14 digital input/output pins (six of which can be used as PWM outputs), six analogue inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button on the board. It comes with everything you'll need to get started with the microcontroller; simply plug it into a computer with a USB cable or power it with an AC-to-DC adapter or battery.

The Uno is unique in that it does not employ the FTDI USB-to-serial driver chip found on previous boards. Instead, it uses a USB-to-serial converter based on the Atmega16U2 (Atmega8U2 up to version R2). •Atmega 16U2 replaces the 8U2.•Stronger RESET circuit The name "Uno" comes from the Italian word "uno," which means "one." It was chosen to commemorate the debut of Arduino 1.0. Moving

forward, Arduino's reference versions will be the Uno and version 1.0. The Uno is the most recent of a series of USB Arduino boards, and it serves as the platform's reference model; see the index of Arduino boards for a comparison of previous generations.

Atmel ATmega328 microcontroller Voltage Input Voltage Input: 5V (recommended) Input voltage ranges between 7 and 12 volts (limits) 14 digital I/O pins with a voltage range of 6 to 20 volts (of which 6 provide PWM output) Input Pins (Analog) 40 mA DC Current for 3.3V Pin 50 mA DC Current for 6 DC Current per I/O Pin 32 KB (ATmega328) of Flash Memory wherein 0.5 KB of SRAM is consumed by the bootloader KB: 2 (ATmega328) 1 KB EEPROM (ATmega328) 16 MHz clock speed arduino-uno-Rev3-reference-design.zip arduino-uno-Rev3-reference-design.zip arduino-uno-Rev3-reference-design.zip arduino-uno-Re (NOTE: Eagle 6.0 and newer are required.)

arduino-uno-Rev3-schematic.pdf arduino-uno-Rev3-schematic.pdf arduino-uno-Rev3- Note: An Atmega8, 168, or 328 can be used in the Arduino reference design. Current models use an ATmega328, although an Atmega8 is shown in the schematic for reference.

On all three processors, the pin arrangement is the same. Power The Arduino Uno can be fueled either by USB or an external power supply. The power source is automatically selected. An AC-to-DC adapter (wall-wart) or a battery can provide external (non-USB) power. A 2.1mm center-to-center connector can be used to connect the adapter. positive plug into the power jack on the board Battery leads can be put into the POWER connector's Gnd and Vin pin headers. The board can be powered from a 6 to 20 volt external supply. If less than 7V is given, the 5V pin may supply fewer than five volts, making the board unstable. The voltage regulator may overheat and destroy the board if more than 12V is used.

The voltage range recommended is 7 to 12 volts. The following are the power pins: When using an external power source (as opposed to 5 volts via a USB connection or other regulated power source), VIN is the input voltage to the Arduino board. This pin can be used to supply voltage or to access voltage if it is supplied via the power jack. 5V. This pin receives a controlled 5V from the board's regulator. The board can be powered by the DC power jack (7-12V), the USB connector (5V), or the VIN pin (7-12V). Using the 5V or 3.3V pins to supply power bypasses the regulator and may cause damage to your board. We don't recommend it. 3V3. The on-board regulator generates a 3.3-volt supply. The maximum current draw is 50 milliamperes. GND is a generic term that refers to a group of Pins that have been ground. • The ATmega328 has 32 KB of memory (with 0.5 KB used for the bootloader).

There's also 2 KB of SRAM and 1 KB of EEPROM on board (which can be read and written with the EEPROM library). With the pinMode(), digitalWrite(), and digitalRead() routines, any of the Uno's 14 digital pins can be utilised as an input or output. They are powered by 5 volts. Each pin includes a 20-50 kOhm internal pull-up

resistor (disconnected by default) and can deliver or receive a maximum of 40 mA. Furthermore, several pins have specific functions: 0 (RX) and 1 (TX) serial numbers (TX). TTL serial data is received (RX) and transmitted (TX) using this device.

These pins• are connected to the ATmega8U2 USB-to-TTL Serial chip's equivalent pins. 2 and 3 External Interrupts On a low• value, a rising or falling edge, or a change in value, these pins can be set to generate an interrupt. For more information, see the attachInterrupt() function. 3, 5, 6, 9, 10, and 11 are PWM. The analogWrite() method generates an 8-bit PWM signal. • SPI: SS (SS), MOSI (MOSI), MISO (MISO), 13 (MISO) (SCK). Using the SPI library, these pins support SPI communication. LED number: 13 A built-in LED is connected to digital pin 13 by a wire. The• LED is on when the pin is HIGH, and it is off when the pin is LOW.

The Uno includes six analogue inputs, labelled A0 through A5, each with a resolution of 10 bits (i.e. 1024 different values). They measure from ground to 5 volts by default, but the AREF pin and the analogReference() function can be used to adjust the upper end of their range. Some pins also have specialised functions, such as: TWI: A4 (SDA) and A5 (SCL) pins.

Using the Wire library, support TWI communication. • On the board, there are a few more pins: AREF. The analogue inputs have a reference voltage. Used in conjunction with analogReference (). • Restart. To reset the microcontroller, set this line to LOW. Usually used to add a reset button to shields that block the board's reset button. Also see the ATmega328 port mapping and the Arduino pin mapping.

2.2.3 Ultrasonic Sensor

Ultrasonic sensors overcome many of the weaknesses of IR sensors - they provide distance measurement regardless of colour and lighting of obstacles. They also provide lower minimum distances and wider angles of detection to guarantee that obstacles are not

missed by a narrow sensor beam. This particular model is an upgrade from the lower precision HC-SRO4.

The SRF05 Ultra Sonic range finder lets you find the distance of the objects in front of it. It does that by sending ultrasonic pulses and measures the time it takes for the pulses to travel to the obstacle and back. The SRF05 is an evolutionary step from the SRF04, and has been designed to increase flexibility, increase range, and to reduce costs still further. As such, the SRF05 is fully compatible with the SRF04. Range is increased from 3 meters to 4 meters. A new operating mode (tying the mode pin to ground) allows the SRF05 to use a single pin for both trigger and echo, thereby saving valuable pins on your controller. When the mode pin is left unconnected, the SRF05 operates with separate trigger and echo pins, like the SRF04. The SRF05 includes a small delay before the echo pulse to give slower controllers such as the Basic Stamp and Picaxe time to execute their pulse in commands.

Pin Details

5-Pin, One each for VCC, Trigger, Echo, Out and Ground.

Trig : Pulsing a HIGH on this pin will send a Pulse

Echo : The pulse length is proportional to the distance from the obstacle (100usec to 25ms, times out if 30ms)

Technical Specifications :

Trigger Pin Format: 10 uS digital pulse

Sound Frequency: 40 kHz

Echo Pin Output: 0-Vcc

Echo Pin Format: output is DIGITAL and is directly proportional with range.

Measurement Range: 2cm to ~4.5m

Measurement Resolution: 0.3cm

Measurement Angle: up to 15 degrees

Measurement Rate: 40 Hz

Supply Voltage: 4.5V to 5.5V

Supply Current: 10 to 40mA

Connector: standard 5-pin male connector which can plug directly into breadboards.

Static current : less than 2mA

Detection distance: 2cm-450cm

Measurement Procedure

- Send a 10 uS wide pulse to the sensor on the Trigger Pin. The sensor will automatically send out a 40 kHz wave.
- Begin monitoring the output from the Echo Pin and
- When the Echo Pin goes high, begin a timer.

- When the Echo Pin goes low, record the elapsed time from the timer and use the following conversion formula:

Distance (in cm) = (elapsed time * sound velocity (340 m/s)) / 100 / 2.

Timing Diagram

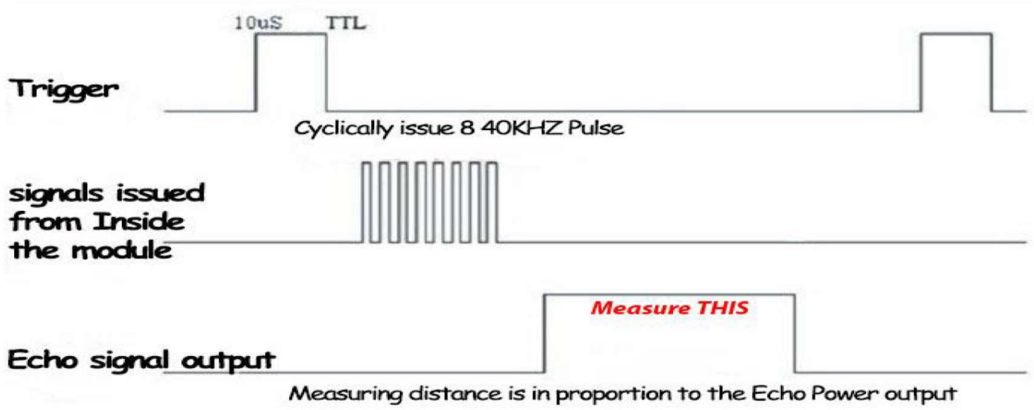


Figure 2.2 Timing Diagram

The basic principle of work:

- Using I/O trigger for at least 10us high level signal
- The module automatically sends eight 40KHz and detect whether there is a pulse signal back

- If the signal back, through high level, time of high output IO duration is the time from sending ultrasonic to returning
- Test distance = high level time x velocity of sound (340m/s) / 2
- Display Format: 16 Characters x 2 lines

Figure 2.3 HY-SRF05 sensor

2.2.4 Buzzer

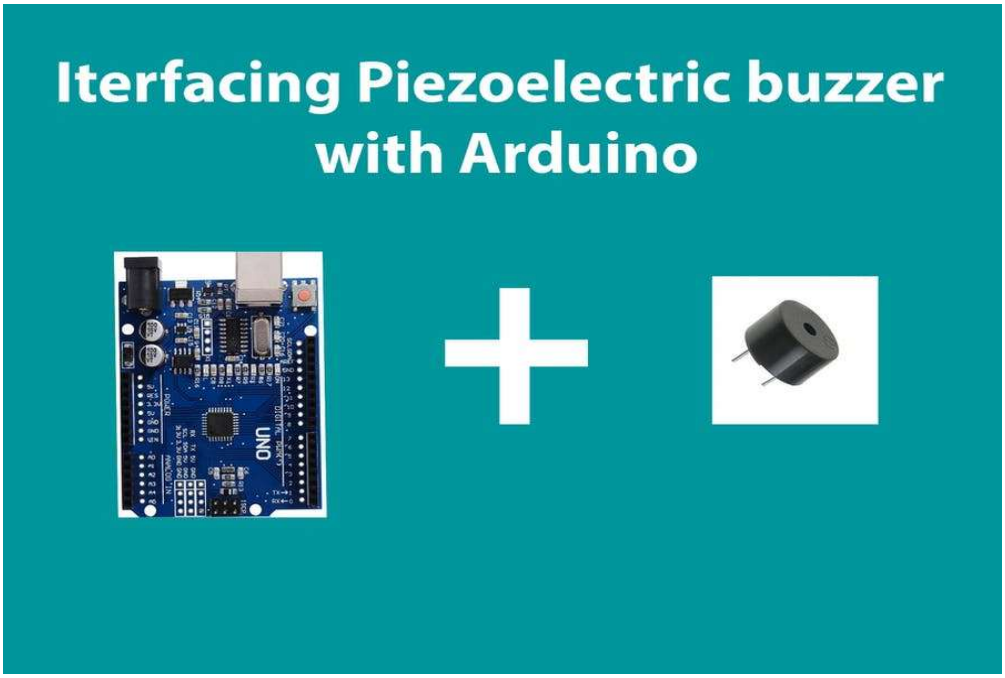
This is a 3V Active Electromagnetic Buzzer that may be mounted on a PCB. Including Audio Alert in your electronic designs is a fantastic idea. It uses a coil element to generate an audible tone and runs on a 3V supply.

Specifications:

- Input Voltage(Max.) : 3V
- Resistance: 30 Ω
- Resonance Frequency: 2048 Hz

- Sound pressure(dB(A)/10cm)min.: 80
- Body Size : 12 x 9.5mm
- Pin Pitch: 6mm
- External Material: Plastic;
- Color: Black

Interfacing Buzzer with Arduino UNO



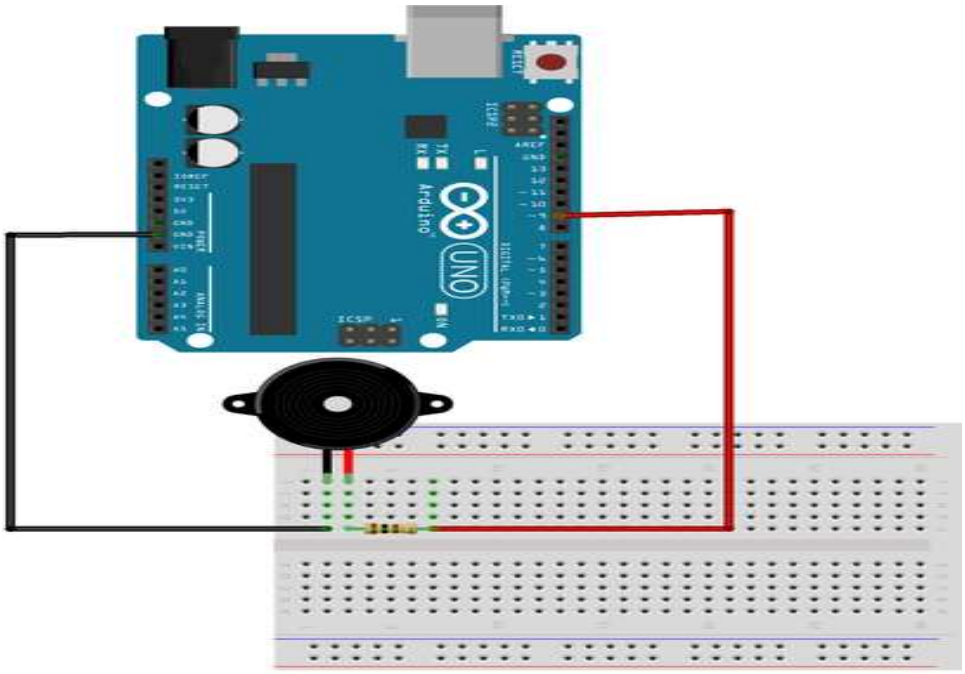


Figure 2.4: Basic Buzzer Connection

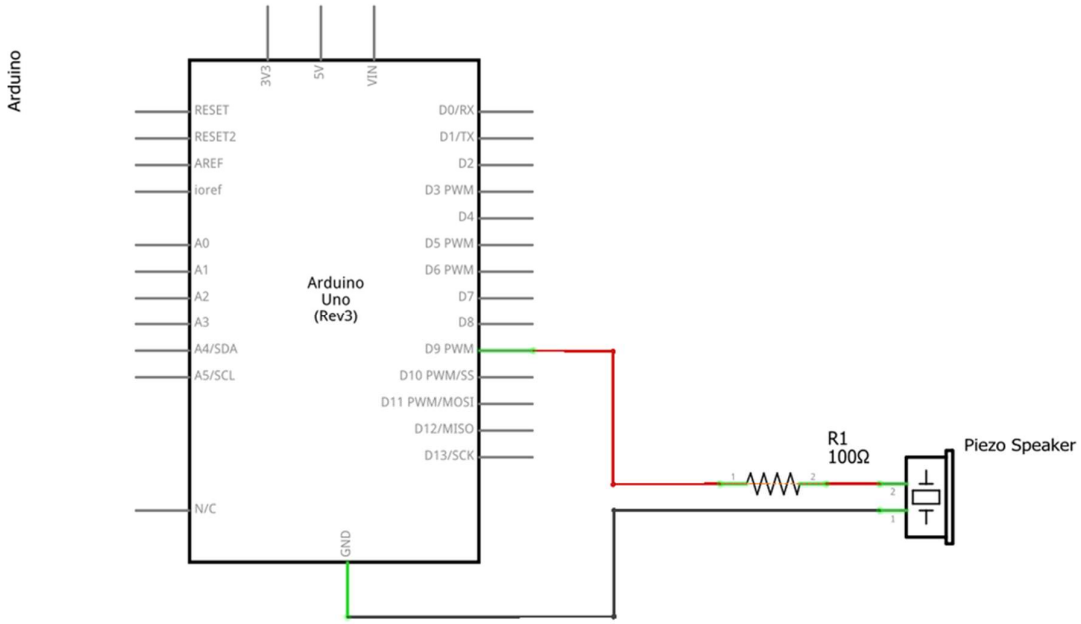
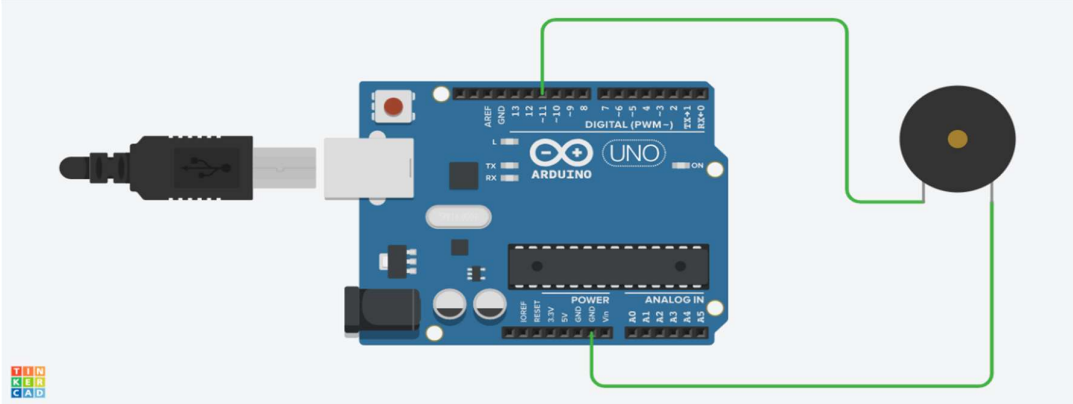


Figure 2.5: Arduino Connected with Buzzer Pin Diagram



The Connections are pretty simple:

1. Connect the Supply wire (**RED**) of the buzzer to the **Digital Pin 9** of the Arduino **through a 100 ohm resistor**.
2. Connect the Ground wire (**BLACK**) of the buzzer to any **Ground Pin** on the Arduino.

Sample Code

```
<p>const int buzzerPin = 9; // declaring the PWM pin</p><p>void setup() {

Serial.begin(8600);

pinMode(buzzerPin, OUTPUT); //addigning pin to Output mode</p><p>}</p><p>void loo
p() {

tone(buzzerPin, 50);

delay(50);

noTone(buzzerPin);

delay(100);

}</p>
```

2.2.5 16*2) LCD Display

LCD16*2 Parallel LCD Display that adds a 162 Black on RGB Liquid Crystal Display to your project in an easy and cost-effective way. The display is 16 characters by 2 lines and has a grey background/backlight with very clear and high contrast black lettering. This is a fantastic LCD monitor with grey backlight. This LCD1602 Gray Backlight Parallel LCD Display is simple to connect to Arduino or other microcontrollers. The values displayed on the display can be simple text or numerical values read from the sensors, such as temperature or pressure, or even the number of cycles performed by the Microcontroller.

Figure 2.6 : LCD Display

Connecting LCD to Arduino UNO

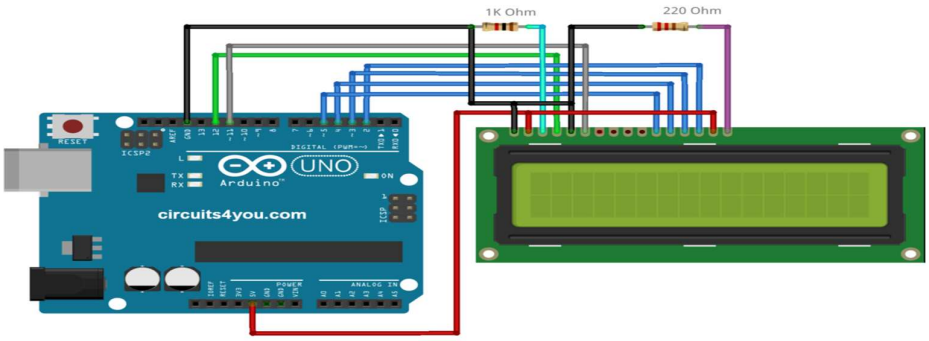


Figure 2.7: Arduino UNO connected with LCD display

All LCDs based on the HD44780 (JHD16x2) driver have a 14- 16-line interface. Through a 330 Ohm current limiting resistor, the backlight connections are connected to the +5V supply. Because we rarely read the display, it's best to link the R/W LCD pin to GND. Connect LCD contrast Pin 3 to ground through a 1K Ohm resistor to set the contrast. It provides the best contrast value. Because we don't need to alter the contrast with a variable resistor every time, it's better to utilise a constant resistor, which saves money and PCB space. The LCDs feature a parallel interface, which means that to operate the display, the microcontroller must manipulate numerous interface pins at once. The pins that make up the interface are as follows:

The register select (RS) pin determines where data is written in the LCD's memory. You can choose between a data register, which stores the information that appears on the screen, and an instruction register, which is where the LCD's controller looks for instructions on what to do next.

A Read/Write (R/W) pin that selects whether the device is in reading or writing mode.

An enable pin that allows you to write to the registers' eight data pins (D0 -D7). When you write to a register, the states of these pins (high or low) represent the bits you're writing, and when you read, the values you're reading.

You can also use a display constrast pin (Vo), power supply pins (+5V and Gnd), and LED Backlight (Bklt+ and BKlt-) pins to power the LCD, manage the display contrast, and turn on and off the LED backlight.

Putting data that forms the image of what you want to display into the data registers, then commands into the instruction register, is how you control the display. You don't need to know the low-level instructions because the Liquid Crystal Library simplifies them for you.

The Hitachi-compatible LCDs can be operated in four different modes: four-bit, eight-bit, and sixteen-bit. The Arduino's 4-bit mode requires seven I/O pins, while the 8-bit mode requires eleven. You can perform almost everything in 4-bit mode for displaying text on the screen, therefore this example explains how to handle a 216 LCD in 4-bit mode.

Sample Code

```
// Import the Liquid Crystal library
#include (LiquidCrystal.h);
//Initialise the LCD with the arduino. LiquidCrystal(rs, enable, d4, d5, d6, d7)
LiquidCrystallcd(12, 11, 5, 4, 3, 2);

void setup() {
 // Switch on the LCD screen
lcd.begin(16, 2);
 // Print these words to my LCD screen
lcd.print("HELLO WORLD!");
}

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

2.2.6 SG90 Servo Motor

The Tower pro SG90S Mini Digital Servo rotates 180 degrees. It's a Digital Servo Motor that accepts and processes PWM signals much more quickly and efficiently. Internal circuitry is complex, allowing for good torque, holding power, and rapid updates in response to external pressures.

Many RC hobbyists prefer our servos because of their excellent tuned performance and dependability.

They come in a tight, robust plastic container that keeps them water and dust resistant, which is a great feature for RC planes, boats, and monster trucks, among other things. It comes with a 3-wire JR servo connector that is also compatible with Futaba connectors.

Wire Description: -

- RED – Positive
- Brown – Negative
- Orange – Signal

Specification: -

- Modulation: Analog
- Speed: 4.8V: 0.11 sec/60°

- 6.0V: 0.10 sec/60°
- Weight: 0.49 oz (14.0 g)
- Dimensions: Length:0.91 in (23.1 mm)
- Width: 0.48 in (12.2 mm)
- Height: 1.14 in (29.0 mm)
- Gear Type: Metal
- Rotation/Support: Dual Bearings
- Rotational Range: 180°
- Pulse Cycle: 20 ms
- Pulse Width: 400-2400 µs

Figure 2.8 : servo motor

Arduino Uno with one Servo Motor

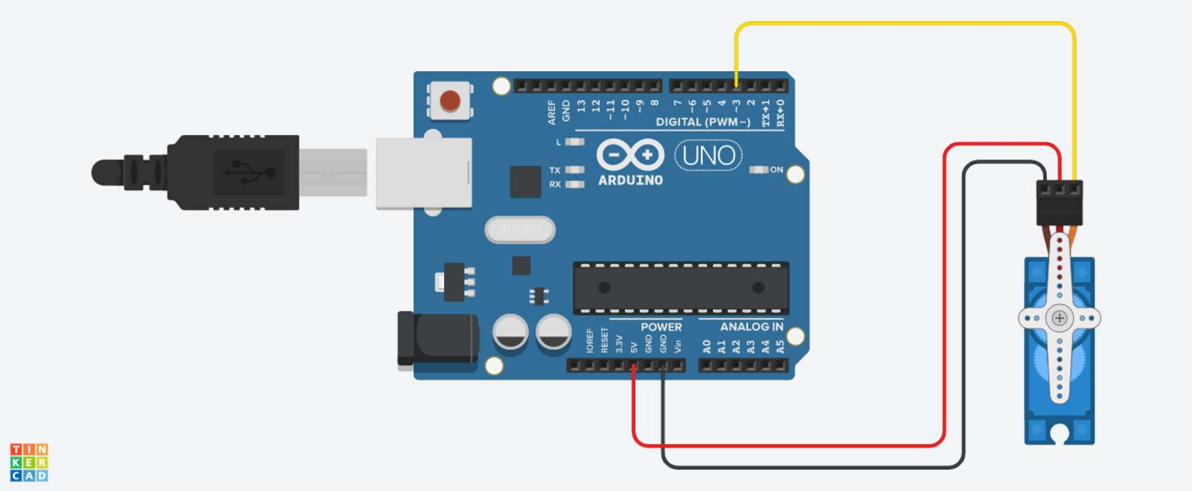


Figure 2.9 : Arduino connected with one servo motor

Sample code

```
#include<Servo.h>
Servo Myservo;
int pos;
void setup()
{
Myservo.attach(3);
}
void loop()
{
for(pos=0;pos<=180;pos++){
Myservo.write(pos);
delay(15);
}
```

2.2.7 Jumper Wires

Jumper wires are used for making connections between items on your breadboard and your Arduino's header pins. Use them to wire up all circuits.

Features :

- Compatible with 2.54mm spacing pin headers
- High quality and in good working condition
- Durable and reusable
- Easy to install and use
- A popular choice for construction or repair
- Be used for the electronic project and Genuine Arduino product
- Flexible Breadboard Jumper Cable Wire allows you to plug and unplug easily for prototyping.
- There are mainly three types male-male ,female-female and male-female jumper wires.

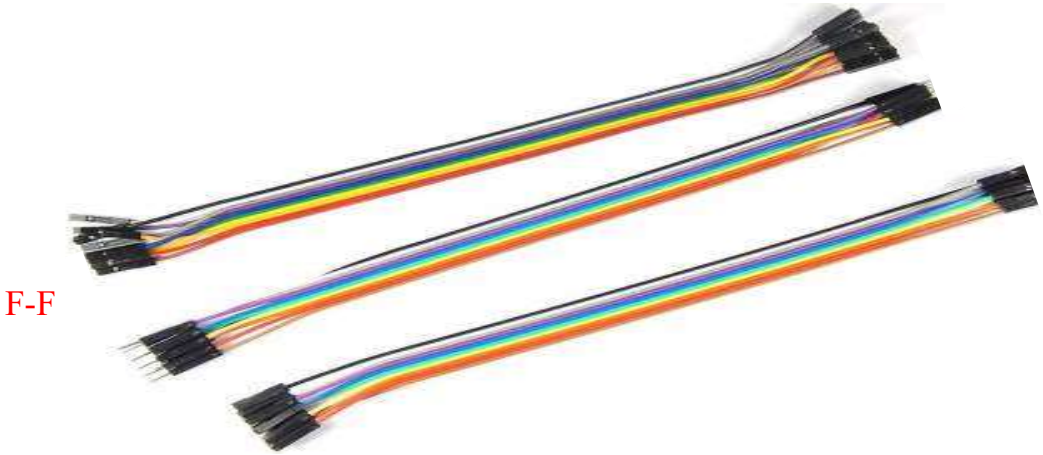


Figure 2.1.0:jumper wires

2.2.8 Bluetooth Module

The Bluetooth Module used here is HC-05. The **HC-05** is a popular bluetooth module which can add two-way (full-duplex) wireless functionality.

HC-05 Pinout Configuration

Pin Number	Pin Name	Description
1	Enable / Key	This pin is used to toggle between Data Mode (set low) and AT command mode (set high). By default it is in Data mode
2	Vcc	Powers the module. Connect to +5V Supply voltage
3	Ground	Ground pin of module, connect to system ground.
4	TX – Transmitter	Transmits Serial Data. Everything received via Bluetooth will be given out by this pin as serial data.
5	RX – Receiver	Receive Serial Data. Every serial data given to this pin will be broadcasted via Bluetooth
6	State	The state pin is connected to on board LED, it can be used as a feedback to check if Bluetooth is working properly.
7	LED	Indicates the status of Module • Blink once in 2 sec: Module has entered Command Mode • Repeated Blinking: Waiting for connection in Data Mode • Blink twice in 1 sec: Connection successful in Data Mode
8	Button	Used to control the Key/Enable pin to toggle between Data and command Mode

Table HC-05 pinout configuration

HC-05 Default Settings

- Default Bluetooth Name: “HC-05”
- Default Password: 1234 or 0000
- Default Communication: Slave
- Default Mode: Data Mode
- Data Mode Baud Rate: 9600, 8, N, 1
- Command Mode Baud Rate: 38400, 8, N, 1
- Default firmware: LINVOR

HC-05 Technical Specifications

- Serial Bluetooth module for Arduino and other microcontrollers
- Operating Voltage: 4V to 6V (Typically +5V)
- Operating Current: 30mA
- Range: <100m
- Works with Serial communication (USART) and TTL compatible
- Follows IEEE 802.15.1 standardized protocol
- Uses Frequency-Hopping Spread spectrum (FHSS)
- Can operate in Master, Slave or Master/Slave mode
- Can be easily interfaced with Laptop or Mobile phones with Bluetooth
- Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.

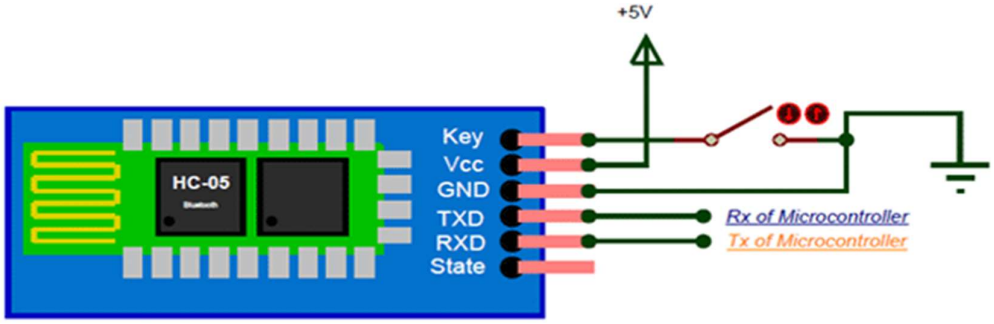


Figure 2.1.1: HC-05 module

2.2.9 Brushed DC Motor

The DC Motor used here is of the type RF – 500TB-12560.

Electrical Specifications

Model	Voltage	No-load Current	No-load Speed	Rated Current	Rated Speed	Rated Torque	Starting Torque	Starting Current
	VDC	*A*	*rpm*	*A*	*rpm*	*mNm*	*mNm*	*A*
RF-500TB-14415	6	0.028	3700	0.13	3060	1.4	8.2	0.65
RF-500TB-12560	6	0.02	2700	0.084	2180	1.1	5.9	0.35

Table 2.1.2 : Electrical specifications

Mechanical Specifications

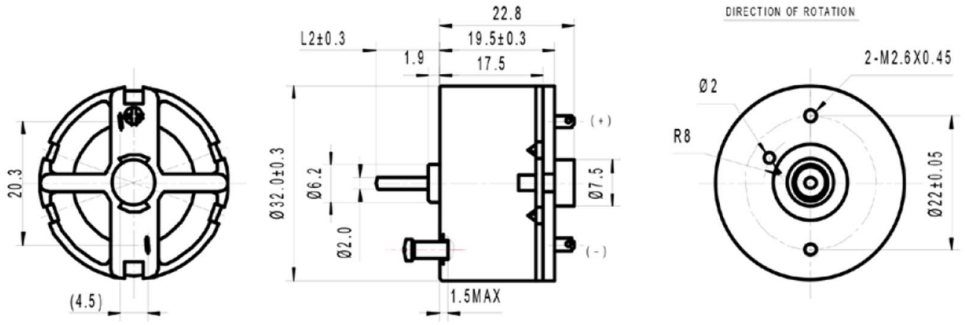


Figure 2.1.3: dc motor mechanical specification

2.2.10 LASER Light

Figure 2.1.4: LASER light

The LLM01 is operated at the top of the module using a labelled rotary selector switch to switch the device on or off and select various lighting options. The device activation can alternately be performed by a pressure switch connected by cable to the LLM01 device.

2.2.11 Driver Circuit

The Model name of the Driver Circuit Used here is **L293D**

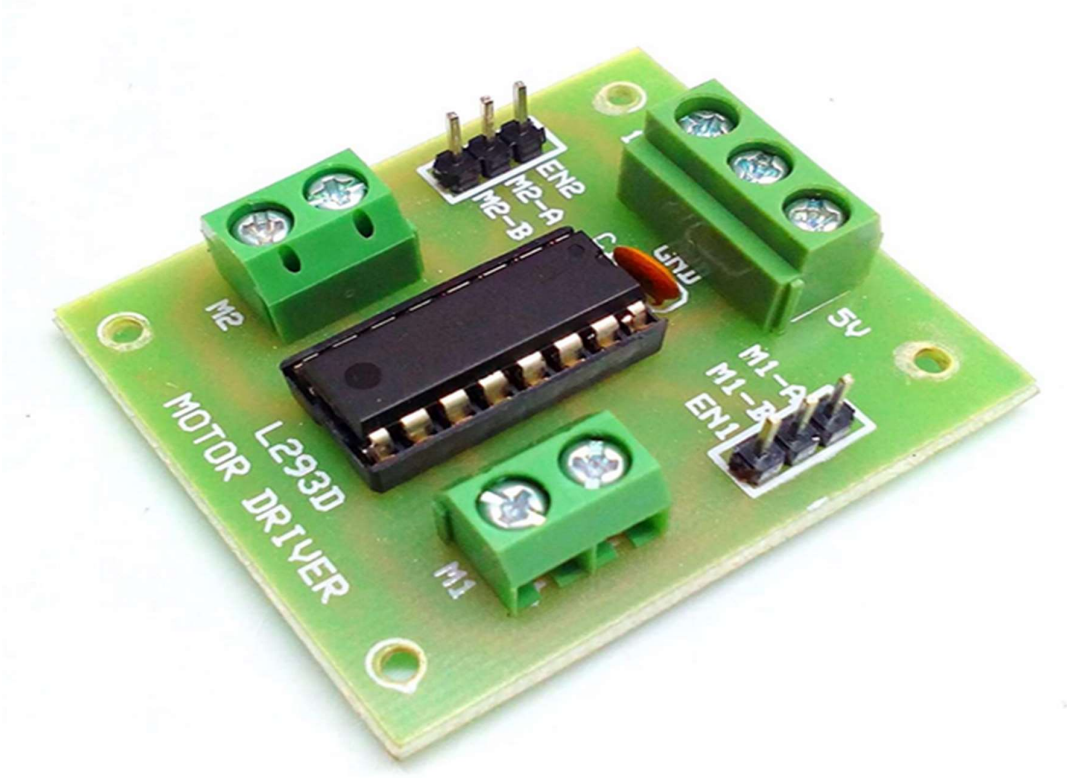


Figure 2.1.5: driver circuit

IN1, IN2, and IN3, IN4 are input pins used for providing a control signal from the controller to run the motor in different directions.

- If input logic at IN1, IN2 is (1,0) the motor rotates in one direction.
- If input logic at IN1, IN2 is (0,1) the motor rotates in the other direction.

EN1 and EN2 are enable pins. Connect 5v DC to EN1 and EN2 pin to operate the motor at its normal speed

- If speed control is needed, then give PWM output at pin EN1 and En2 from the microcontroller.

Power for the motor. If 12V DC gear motor is used then apply 12V.

Chapter 3

Software Requirements

3.1 Embedded C

In microcontrollers, the embedded c programming language is utilised. The embedded C programming language is a general-purpose programming language with code efficiency, organised programming principles, and a large number of operators. Embedded C is a small programming language that isn't intended for a single application. Embedded C is a convenient and effective programming option for a wide range of software jobs because of its lack of restrictions. Many problems can be solved more quickly and easily with embedded C than they can with other more specialised languages. The embedded c language is incapable of performing activities (such as input and output) that would typically necessitate operating system involvement. Because these functions are segregated from the language, embedded c is well-suited to writing code that runs on a variety of systems.

About C Language

Dennis Ritchie created the C programming language in 1969. It is made up of one or more functions, each of which is made up of statements that accomplish a specified task.
The C programming language is a middle-level language because it can handle both high-level and low-level applications. Before diving into the complexities of embedded C programming, it's important to understand how RAM memory is organised.

Salient features of the language

• The C programming language is a collection of keywords, data types, variables, constants, and other features.

• Embedded C is a generic word for a C-based programming language that is linked to a specific hardware architecture.

• Embedded C is a C language extension with certain additional header files. These header files may differ from one controller to the next.

• The microcontroller 8051 is utilised #includereg51.h>.

To create programmes, embedded system designers must understand hardware architecture. External devices are monitored and controlled with the help of these programmes. They can also directly operate and utilise the microcontroller's internal architecture, including as interrupt handling, timers, serial connection, and other functions.

Differences between C and Embedded C

C programming	Embedded C programming
Possesses native development in nature.	Possesses cross development in nature.
Independent of hardware architecture.	Dependent on hardware architecture (microcontroller or other devices).
Used for Desktop applications, OS and PC memories.	Used for limited resources like RAM, ROM and I/O peripherals on embedded controller.

Table 3: Differences between C and Embedded C

The basic additional features of the embedded software

Data types

The term "data type" refers to a comprehensive system for declaring variables of various sorts, such as integers, characters, and floats. The integrated C software stores data in the memory using four different data types.

The variables 'char' and 'int' are used to store single characters, whereas 'float' is used to hold precision floating point values.

The following table shows the size and range of several data types on a 32-bit system. On devices with different word sizes, the size and range may vary.

Data Type	Size	Range
Char or signed char	1byte	-128 to +128
Unsigned char	1byte	0 to 255
Int or singed int	2byte	-32768 to 32767
Unsigned int	2byte	0 to 65535

Table4: Data types

Keywords

There are some terms that should only be used for specific jobs. Keywords are the terms used to describe certain words. In Embedded C, they are standard and predefined.

Lowercase is used for all keywords. Before writing the main programme, these keywords must be defined. The following are some of the most common embedded software terms:

Name	Funtion
sbit	Accessing of single bit
bit	Aceessing of bit addressable memory of RAM
sfr	Accessing of sfr register by another name

Table 5: Keywords

sbit:

When accessing a single bit of the SFR register, this data type is used.

• Syntax: sbit variable name = SFR bit;

• Ex: sbit a=P21;

• Explanation: If we make p2.1 a variable, we may use 'a' instead of p2.1 anyplace in the programme, reducing the program's complexity.

Bit:

This data type is used to get reach RAM's bit addressable memory (20h-2fh).

• Syntax: bit variable name;

• Ex: bit c;

• Definition: A bit sequence setting in a limited data area that a software uses to remember something.

SFR:

This data type is used to access an SFR register that has been given a different name. Capital letters must be used to declare all SFR registries.

• Syntax: SFR variable name = SFR address of SFR register;

• Ex: SFR port0=0x80;

• Explanation: If we declare 0x80 as 'port0,' we can utilise 0x80 instead of port0 throughout the programme, reducing the program's complexity.

SFR Register:

'Special Function Register' is the abbreviation for 'Special Function Register.' The RAM memory of the 8051 microcontroller is 256 bytes. This RAM is divided into two sections: one with 128 bytes for data storage and the other with 128 bytes for SFR registers. The SFR register stores all peripheral devices such as I/O ports, timers, and counters, and each element has its own address.

3.1.1 Arduino IDE

Arduino is an open-source electronics platform that uses simple hardware and software to make it easy to use. Arduino boards can take inputs - such as light from a sensor, a finger on a button, or a Twitter message - and convert them to outputs - such as turning on an LED, triggering a motor, or publishing anything online. By providing a set of instructions to the board's microcontroller, you may tell it what to do. The Arduino programming language (based on Wiring) and the Arduino Software (IDE) (based on Processing) are used to accomplish this.

An integrated development environment (IDE) is a software application that gives computer programmers all the tools they need to create software. A source code editor, build automation tools, and a debugger are typically included in an IDE. Intelligent code completion is available in most current IDEs. Some IDEs, like NetBeans and Eclipse, include a compiler, interpreter, or both, whereas others, like Sharp Develop and Lazarus, don't. The distinction between an integrated development environment and the rest of the software development environment is hazy. A version control system or numerous technologies to make the creation of a graphical user interface (GUI) easier are sometimes included. For object-oriented software development, many modern IDEs include a class browser, an object browser, and a class hierarchy diagram. Integrated development environments (IDEs) are created to help programmers work more efficiently by combining closely related components with similar user interfaces. All development is done in a single programme with an IDE. This application usually comes with a lot of tools for

writing, editing, compiling, deploying, and debugging software. This is in contrast to utilising unrelated tools like vi, GCC, or make to produce software. One of the goals of the IDE is to reduce the amount of configuration required to piece together numerous development utilities by offering the same set of features as a single unit. In circumstances where learning to use the IDE is faster than manually integrating all of the various tools, reducing setup time can boost developer productivity. Beyond assisting with setup activities, tighter integration of all development tasks has the potential to boost overall productivity. For example, code can be continually processed while being changed, providing immediate feedback when syntax problems occur. This can help you learn a new programming language and its libraries faster.

Some IDEs are specialised to a single programming language, allowing them to have a feature set that closely matches the language's programming paradigms. There are, however, several IDEs that support many languages.

While the majority of modern IDEs are graphical, text-based IDEs like Turbo Pascal were widely used before the widespread availability of windowing systems like Microsoft Windows and the X Window System (X11). Function keys or hotkeys are typically utilised to perform frequently used commands or macros.

3.2 Applications:

- To protect against threats;
- it is an all-in-one system, so there is no need to carry several gadgets.
- Once the system is activated, GPS tracking follows the user around.
- When the battery is low, the current position is immediately sent to the pre-registered numbers.
- Because this technology captures the image of the criminal, it aids subsequent investigations in locating the criminal.

3.3 ARDUINO CODE

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>

// Set the LCD address to 0x27 for a 16 chars and 2 line display

LiquidCrystal_I2C lcd(0x27, 16, 2);

#include <Servo.h>

Servo myservo;

int relay1=2;

int relay2=3;

int relay3=4;

int relay4=5;

int relay5=7;

int buzzer=A0;

const int TriggerPin1 = 8; //Trig pin

const int EchoPin1 = 9; //Echo pin

long Duration1 = 0;

const int TriggerPin2 =10 ; //Trig pin

const int EchoPin2 = 11; //Echo pin

long Duration2 = 0;
```

```
const int TriggerPin3 = 13; //Trig pin

const int EchoPin3 = 12; //Echo pin

long Duration3 = 0;

int mdelay=1000;

String datain;

void setup(){

Serial. Begin(9600); // Serial Output

 lcd.init();

 // Turn on the blacklight and print a message.

 lcd.backlight();

 myservo.attach(6);

 myservo.write(0);

pinMode(relay1,OUTPUT);

pinMode(relay2,OUTPUT);

pinMode(relay3,OUTPUT);

pinMode(relay4,OUTPUT);

pinMode(relay5,OUTPUT);

pinMode(buzzer,OUTPUT);

 pinMode(TriggerPin1,OUTPUT); // Trigger is an output pin

 pinMode(EchoPin1,INPUT); // Echo is an input pin
```

```
 pinMode(TriggerPin2,OUTPUT); // Trigger is an output pin

 pinMode(EchoPin2,INPUT); // Echo is an input pin

 pinMode(TriggerPin3,OUTPUT); // Trigger is an output pin

 pinMode(EchoPin3,INPUT); // Echo is an input pin

  }

void loop(){

//  control(); // Call All the Control

  sensor1();

  sensor2();

  sensor3();

 if(Serial. Available())

 {

   while(Serial.available() > 0)

   {

     char c = Serial.read();

     datain +=c;

     delay(10);

     }

     Serial.println(datain);
```

```
if(datain=="1")//if(datain=="1")
{
 Serial.println("FORWARD ON");
 digitalWrite(relay1,LOW);
 digitalWrite(relay2,HIGH);
 digitalWrite(relay3,LOW);
 digitalWrite(relay4,HIGH);
 }
 else if(datain=="0")
 {
  digitalWrite(relay1,HIGH);
  digitalWrite(relay2,HIGH);
  digitalWrite(relay3,HIGH);
  digitalWrite(relay4,HIGH);
//   digitalWrite(relay7,HIGH);
  Serial.println("FORWARD OFF");
  }
  else if(datain=="2")
 {
  // Serial.println("BACKWARD ON");
```

```
 digitalWrite(relay1,HIGH);

 digitalWrite(relay2,LOW);

 digitalWrite(relay3,HIGH);

 digitalWrite(relay4,LOW);

 }

 else if(datain=="3")

{

 Serial.println("LEFT ON");

 digitalWrite(relay1,LOW);

 digitalWrite(relay2,HIGH);

 digitalWrite(relay3,HIGH);

  digitalWrite(relay4,HIGH);

 }

 else if(datain=="4")

{

 Serial.println("RIGHT ON");

 digitalWrite(relay1,HIGH);

 digitalWrite(relay2,HIGH);

 digitalWrite(relay3,LOW);

 digitalWrite(relay4,HIGH);
```

```
        }
       datain="";
   }
}
long Distance1(long time)
{
// Calculates the Distance in mm
// ((time)*(Speed of sound))/ toward and backward of object) * 10

long DistanceCalc1; // Calculation variable
DistanceCalc1 = ((time /2.9) / 2); // Actual calculation in mm
//DistanceCalc = time / 74 / 2; // Actual calculation in inches
return DistanceCalc1; // return calculated value
}
long Distance2(long time)
{
// Calculates the Distance in mm
// ((time)*(Speed of sound))/ toward and backward of object) * 10

long DistanceCalc2; // Calculation variable
```

```
DistanceCalc2 = ((time /2.9) / 2); // Actual calculation in mm

//DistanceCalc = time / 74 / 2; // Actual calculation in inches

return DistanceCalc2; // return calculated value

}

long Distance3(long time)

{

// Calculates the Distance in mm

// ((time)*(Speed of sound))/ toward and backward of object) * 10

long DistanceCalc3; // Calculation variable

DistanceCalc3 = ((time /2.9) / 2); // Actual calculation in mm

//DistanceCalc = time / 74 / 2; // Actual calculation in inches

return DistanceCalc3; // return calculated value

}

 void sensor1()

{

digitalWrite(TriggerPin1, LOW);

delayMicroseconds(2);

digitalWrite(TriggerPin1, HIGH); // Trigger pin to HIGH

delayMicroseconds(10); // 10us high

digitalWrite(TriggerPin1, LOW); // Trigger pin to HIGH
```

```
Duration1 = pulseIn(EchoPin1,HIGH); // Waits for the echo pin to get high

// returns the Duration in microseconds

long Distance_mm1 = Distance1(Duration1); // Use function to calculate the distance

//Serial.print("Duration1 = ");

//Serial.print(Duration1);

//Serial.print("   ");

//Serial.print("Distance1 = "); // Output to serial

//Serial.print(Distance_mm1);

//Serial.println(" mm");

if(Distance_mm1<200)

{

 //lcd.clear();

 lcd.setCursor(0,0);

 lcd.print("Target range:");

 lcd.print(Distance_mm1);

 lcd.setCursor(0,1);

 lcd.print("90 Degree detected");

 delay(2000);

digitalWrite(buzzer,HIGH);

   lcd.clear();
```

```
      lcd.setCursor(0,0);
      lcd.print("Destroying");
       digitalWrite(relay5,HIGH);
 myservo.write(90);
 }
 else
 {  lcd.clear();
digitalWrite(buzzer,LOW);
 digitalWrite(relay5,LOW);
  }
 }
 void sensor2()
 {
//2ND ULTRASONIC SENSOR
digitalWrite(TriggerPin2, LOW);
delayMicroseconds(2);
digitalWrite(TriggerPin2, HIGH); // Trigger pin to HIGH
delayMicroseconds(10); // 10us high
digitalWrite(TriggerPin2, LOW); // Trigger pin to HIGH
Duration2 = pulseIn(EchoPin2,HIGH); // Waits for the echo pin to get high
```

```
// returns the Duration in microseconds

long Distance_mm2 = Distance2(Duration2); // Use function to calculate the distance

//Serial.print("Duration2 = ");

//Serial.print(Duration2);

//Serial.print("   ");

//Serial.print("Distance2 = "); // Output to serial

//Serial.print(Distance_mm2);

//Serial.println(" mm");

if(Distance_mm2<200)

{ lcd.clear();

 lcd.setCursor(0,0);

 lcd.print("Target range:");

 lcd.print(Distance_mm2);

 lcd.setCursor(0,1);

 lcd.print("0 Degree detected");

 delay(2000);

 digitalWrite(buzzer,HIGH);

    lcd.clear();

      lcd.setCursor(0,0);

      lcd.print("Destroying");
```

```
      digitalWrite(relay5,HIGH);

 myservo.write(0);

 delay( mdelay);

 }

 else

 {  lcd.clear();

digitalWrite(buzzer,LOW);

digitalWrite(relay5,LOW);

   }

}

 void sensor3()

{

digitalWrite(TriggerPin3, LOW);

delayMicroseconds(2);

digitalWrite(TriggerPin3, HIGH); // Trigger pin to HIGH

delayMicroseconds(10); // 10us high

digitalWrite(TriggerPin3, LOW); // Trigger pin to HIGH

Duration3 = pulseIn(EchoPin3,HIGH); // Waits for the echo pin to get high

// returns the Duration in microseconds

long Distance_mm3 = Distance3(Duration3); // Use function to calculate the distance
```

```
//Serial.print("Duration3 = ");

//Serial.print(Duration3);

//Serial.print("   ");

//Serial.print("Distance3 = "); // Output to serial

//Serial.print(Distance_mm3);

//Serial.println(" mm");

if(Distance_mm3<200)

{

 lcd.clear();

 lcd.setCursor(0,0);

 lcd.print("Target range:");

 lcd.print(Distance_mm3);

 lcd.setCursor(0,1);

 lcd.print("180 Degree detected");

delay(2000);

digitalWrite(buzzer,HIGH);

    lcd.clear();

      lcd.setCursor(0,0);

      lcd.print("Destroying");

       digitalWrite(relay5,HIGH);
```

```
myservo.write(180);
delay( mdelay);
}
else
{  lcd.clear();
   digitalWrite(buzzer,LOW);
   digitalWrite(relay5,LOW);
  }
}
```

Chapter 4

Prototype

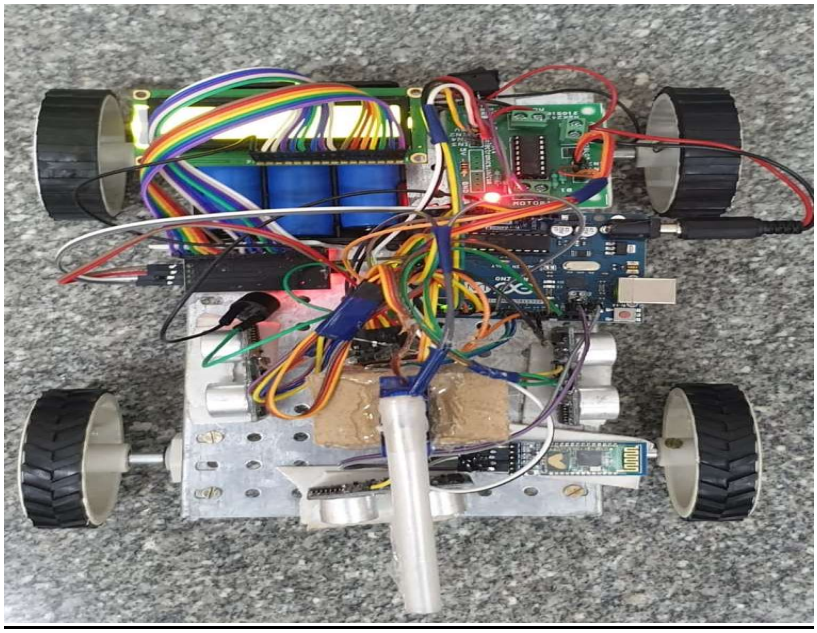


Figure 4.1 Prototype

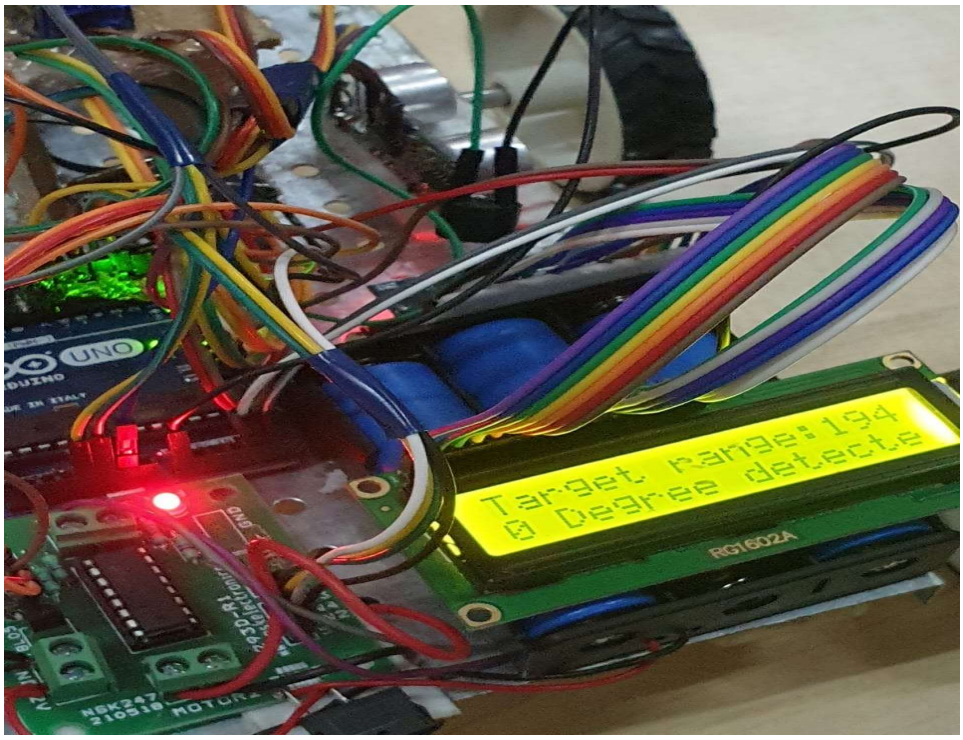


Figure 4.2 Detection System

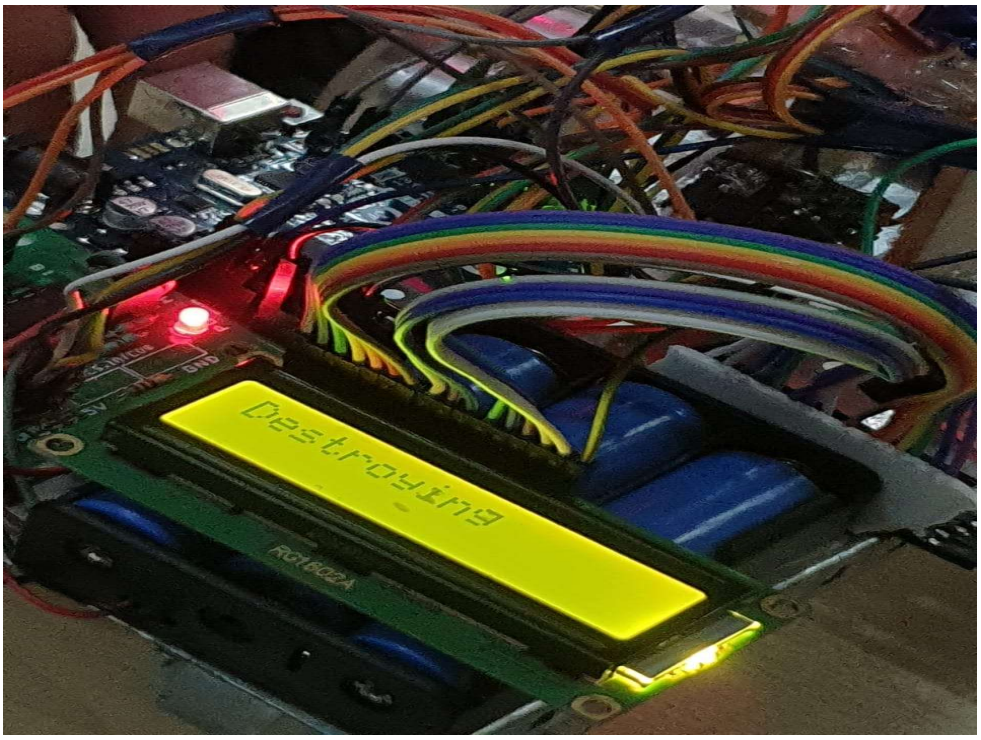


Figure 4.3 Destroying System

Chapter 5

Conclusion

An ultrasonic module is connected to a microcontroller from the 8051 series in this system. A transmitter and receiver are employed in an ultrasonic transducer. The item reflects the sent waves back to the transducer, which receives them again. Taking into account the velocity of sound, the total time taken from sending the waves to receiving them is estimated. The distance is then calculated by a microcontroller programme and shown on a liquid crystal display screen connected to the microcontroller. The circuit is used to receive 40 KHz reflected signals from the missile object, feed them into a microcontroller programme, and turn on the necessary load while the programme is running on the microcontroller. When the ultrasonic receiver sends a signal to the microcontroller, it activates the door cannon by triggering the gate of a MOSFET via a transistor or relay. The sensor is attached to the antenna and rotated and operated by a servo motor in a 180-degree rotation. If a target is discovered within the detection range, the programme will direct the launcher to the closest detected target and fire, applying a laser beam to it.

Chapter 6

References

[1]. Chopra, S., Bharti, S. and Negi, T.S., 2014. Missile Detection by Ultrasonic and Auto Destroy System.

[2]. Soni Kumari, C.K. and Kumar, S., Automatic Missile Detection And Destroying System.

[3]. Swamy, A.A.B. and Augusteen, W.A., MICROCONTROLLER BASED MISSILE DETECTION AND DESTROYING.

[4]. Jeon, I.S., Lee, J.I. and Tahk, M.J., 2010. Homing guidance law for cooperative attack of multiple missiles. *Journal of guidance, control, and dynamics*, *33*(1), pp.275-280.

[5]. Nuryanto, N., Widiyanto, A. and Burhanuddin, A., 2017. Redirection Concept of Autonomous Mobile Robot HY-SRF05 Sensor to Reduce The Number of Sensors. *Proceeding of the Electrical Engineering Computer Science and Informatics*, *4*(1), pp.540-543.

[6]. Diwanji, M., Hisvankar, S. and Khandelwal, C., 2019, September. Autonomous fire detecting and extinguishing robot. In *2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT)* (pp. 327-329). IEEE.

[7]. Agung, I.R., Huda, S. and Wijaya, I.A., 2014, November. Speed control for DC motor with pulse width modulation (PWM) method using infrared remote control based on ATmega16 microcontroller. In *2014 International Conference on Smart Green Technology in Electrical and Information Systems (ICSGTEIS)* (pp. 108-112). IEEE.

[8]. Lauria, M., Nadeau, I., Lepage, P., Morin, Y., Giguère, P., Gagnon, F., Létourneau, D. and Michaud, F., 2006, November. Design and control of a four steered wheeled mobile robot. In *IECON 2006-32nd Annual Conference on IEEE Industrial Electronics* (pp. 4020-4025). IEEE.