# INTEGRATION OF SENSORS IN ROBOTICS

**Centre for Indian Knowledge Systems (CIKS), IIT Guwahati**

**Prepared by**

Shikshita Chhetri

Intern IIT Guwahati, Assam

B.Tech 4th Year, NERIST

June 2024

**Under the guidance of**

Dr. Uday Shanker Dixit

Department of Mechanical Engineering

# CERTIFICATE

This is to certify that the work contained in this report entitled **"The Role of Sensors in IoT: A Comprehensive Exploration"** by **Shikshita Chhetri**, has been carried out at the Centre for Indian Knowledge Systems, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a certificate.

**Prof. U.S. DIXIT**

Department of Mechanical Engineering

July 2024

Indian Institute of Technology Guwahati

# STUDENT DECLARATION

I hereby declare that the entire work embodied in this report entitled **"The Role of Sensors in IoT: A Comprehensive Exploration"** has been carried out by me. No part of it has been submitted for any certificate of any institution previously.

Date: 06/07/2024

**Shikshita Chhetri**

**Intern IIT Guwahati**

# Acknowledgement

I would like to express my sincere gratitude and appreciation to **IIT Guwahati** for providing me with the opportunity to be a part of their internship program. This experience has been invaluable in shaping my professional growth and furthering my understanding.

I am also immensely grateful to my supervisor, **Prof. Uday Shanker Dixit**, Head of Centre, Centre of Indian Knowledge Systems (CIKS) for allowing us to undergo an internship under himand his unwavering support and mentorship. His guidance, constructive feedback, and patience have been crucial in honing my skills and allowing me to take on new challenges with confidence.

I am also deeply grateful to **Dr. Faladrum Sharma** for providing continuous support and directing our efforts to ensure the successful completion ofthe internship. His insightful input and valuable suggestions greatly contributed to the quality of our work. I would like to extend my thanks to **Mr. Kalyan Boro** for providing valuable assistance during our work. I would also like to express my gratitude to **Mr. Abhijeet Dhulekar** and **Intern Mr. Archya Roy** for their constant assistance throughout the internship. His efforts in providing the necessary materials, facilitating access to workshops helped a lot in the successful execution of our project.

I am grateful to **Pandit Hemchandra Goswami** Foundation for providing support to this work through the project **"Pandit Hemchandra Mission for Popularization of Science and Assamese Culture".**

I would also like to express my gratitude to all my colleagues and teammates for their warm welcome, camaraderie, and collaborative spirit. The positive work environment and the opportunity to work alongside such talented individuals have significantly contributed to my learning experience. Lastly, I would like to thank the Think India organization for providing me with a platform to help attain this internship.

Sincerely,

Shikshita Chhetri

# Abstract

This report explores practical applications of Internet of Things (IoT) and embedded systems through hands-on projects, focusing on Arduino-based platforms. It begins with an introduction to fundamental IoT concepts and Arduino's role in sensor integration, emphasizing its versatility and applicability in modern technology. The report then delves into specific projects showcasing real-world applications. The ESP32-CAM project demonstrates the module's capabilities in live streaming and its potential across security, education, healthcare, and retail sectors. It addresses technical aspects like coding and configuration while discussing broader implications for surveillance and monitoring systems. Another work involves radar sensor integration with Arduino and ultrasonic sensors, coupled with data visualization using the Processing Development Environment (PDE). This highlights how sensor data can be processed and visualized to enhance decision-making in environment monitoring and object detection. Moreover, the obstacle avoidance robot car has fabricated using ultrasonic sensor and Arduino Mega 2560 that illustrating principles of autonomous navigation with integration of the ultrasonic sensor. Overall, these projects provide practical insights into IoT and embedded systems, encouraging exploration and application in diverse technological contexts.

# Contents

# Chapter 1: Introduction

## 1.1 IoT (Internet of Things)

The term "Internet of Things" (IoT) describes a network of common place physical objects that are linked to the internet so they may trade, gather, and use data. These items might include anything from automobiles, wearable technology, and industrial gear to home goods like refrigerators and thermostats. These gadgets can send and receive data since they are outfitted with sensors and software *t*hat enables internet communication.

**Key aspects usually seen in IoT:**

*Physical Devices and Objects:* These include everyday items like refrigerators, thermostats, cars, and wearable, as well as industrial machines and infrastructure components.

*Sensors and Actuators:* Sensors collect data from the environment (e.g., temperature, motion, moisture), while actuators perform actions based on processed data (e.g., adjusting a thermostat, locking a door).

*Connectivity:* Devices are connected through various communication protocols like Wi-Fi, Bluetooth, Zigbee, and cellular networks, enabling them to send and receive data.

*Data Processing:* Data collected by IoT devices is processed locally (edge computing) or sent to centralized cloud servers where it is analyzed and used to make informed decisions.

*User Interface:* Interfaces like mobile apps, web dashboards, and voice assistants allow users to interact with IoT devices, monitor their status, and control their functions remotely. Example:

Consider a smart refrigerator. It can:

- Track its contents and notify you when supplies like milk are running low.
- Suggest recipes based on the ingredients available inside.
- Send alerts to your phone if the door is left open.

This level of functionality is achieved because the refrigerator is connected to the internet and can communicate with other devices, such as our smart phone.

IoT represents a transformative technology that is reshaping various industries and aspects of daily life. By enabling devices to communicate and make intelligent decisions, IoT improves efficiency, enhances convenience, and drives innovation. However, addressing challenges related to security, privacy, interoperability, and scalability is essential for realizing the full potential of IoT. As technology continues to evolve, IoT will play an increasingly vital role in creating a more connected and intelligent world.

## 1.2 Sensors

Sensors are fundamental components of modern technology, essential for the Internet of Things (IoT), automation, and various industrial and consumer applications. A sensor is a device that detects and responds to changes in the environment, converting physical parameters (such as temperature, pressure, or humidity) into readable signals (electrical, optical, or digital) that can be further processed and analyzed. This capability makes sensors critical for monitoring, control, and data collection across diverse fields. These sensors are classified into many types:

*1. Temperature Sensors:* Devices like thermostats and thermocouples detect temperature variations by changing their electrical resistance or generating voltage. They find use in HVAC systems, medical equipment, and environmental monitoring.

*2. Pressure Sensors:* Strain gauges and piezoelectric sensors measure pressure changes by detecting deformation or electrical charge fluctuations. Applications include automotive tire pressure monitoring and industrial equipment.

*3. Humidity Sensors:* These sensors, such as capacitive and resistive types, gauge moisture levels by monitoring changes in electrical capacitance or resistance. They are integral to climate control systems and agricultural monitoring.

*4. Proximity Sensors:* Using technologies like inductive, capacitive, and ultrasonic methods, proximity sensors detect the presence of objects without physical contact. They are essential in robotics and security systems.

*5. Light Sensors:* Photodiodes and phototransistors convert light levels into electrical signals. They are used in ambient light detection for smart phones, automated lighting systems, and optical safety devices.

*6. Motion Sensors:* Accelerometers and gyroscopes measure acceleration and angular velocity, crucial for applications in smart phones, vehicle navigation systems, and gaming controllers.

*7. Gas Sensors:* Sensors like electrochemical, infrared, and metal oxide types detect gases by analyzing changes in electrical properties or absorption spectra. They are vital for air quality monitoring and industrial safety.

*8. Biosensors:* Enzyme-based, immune sensors, and DNA sensors detect biological substances through biochemical reactions, used extensively in medical diagnostics and environmental monitoring.

*9. Ultrasonic Sensors:* An ultrasonic sensor is a device that uses high-frequency sound waves to detect objects, measure distances, or monitor levels without physical contact. It emits ultrasonic pulses and calculates distances based on the time it takes for echoes to return.

*10. IR Sensors:* IR (Infrared) sensors are electronic devices that detect infrared radiation emitted or reflected by objects. They are used to measure distance, detect motion, or sense the presence of objects without direct contact. IR sensors operate by emitting infrared light and then measuring the reflection or absorption of this light by nearby objects.

Sensors offer benefits such as improved accuracy, automation, and real-time monitoring capabilities, contributing to enhanced safety, efficiency, and informed decision-making. However, challenges such as calibration needs, power consumption, data management complexities, and susceptibility to interference require attention for optimal sensor performance and reliability.

In conclusion, sensors are indispensable components in modern technology, facilitating advancements across various sectors by providing crucial data for monitoring, control, and automation. As sensor technology continues to evolve, addressing challenges and optimizing sensor capabilities will be the key to unlocking their full potential and enabling further innovation in IoT, healthcare, industry, agriculture, and beyond.

## 1.3 Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments.

# Chapter 2: ESP32 CAM: Live Streaming Web Server

## 2.1 Introduction

The ESP32-CAM is a small-size, low-power camera module based on ESP32. It comes with an OV2640 camera and provides an onboard TF card slot. This board has 4MB PSRAM which is used for buffering images from the camera into video streaming or other tasks and allows you to use higher quality in your pictures without crashing the ESP32. It also comes with an onboard LED for flash and several GPIOs to connect peripherals. The ESP32-CAM is an innovative and highly versatile development board designed by Espressif Systems. It has garnered significant attention in the tech and DIY communities due to its compact size, powerful features, and extensive application possibilities. Below is an in-depth exploration of the ESP32-CAM, detailing its hardware specifications, software capabilities, typical use cases, and potential for innovation.

## 2.2 Hardware Specifications

- Microcontroller: At the heart of the ESP32-CAM is the ESP32-S microcontroller, a robust and highly capable System on Chip (SoC) that includes a dual-core Tensilica LX6 processor. The ESP32-S operates at a clock frequency of up to 240 MHz and features 520 KB of SRAM, which allows it to handle complex tasks and multitasking with ease. The dual-core architecture ensures efficient performance and responsiveness, making it suitable for real-time applications.

- Camera: The integrated camera module on the ESP32-CAM is the OV2640, a highly efficient camera sensor capable of capturing high-resolution images and video produced by OmniVision(OV). The OV2640 supports a variety of resolutions, including VGA (640x480) and SVGA (800x600), and can output images in formats such as JPEG, BMP, and GRAYSCALE. This flexibility makes it ideal for a wide range of image processing tasks, from simple snapshots to complex video streaming applications.

- Memory and Storage: The ESP32-CAM comes with 4 MB of built-in flash memory, which is adequate for storing firmware and some image data. For projects requiring additional storage, the board features a micro SD card slot that supports cards up to 4 GB. This additional storage can be used for saving images, videos, and other data-intensive files.

- Connectivity: One of the standout features of the ESP32-CAM is its integrated Wi-Fi and Bluetooth capabilities. The board supports 802.11 b/g/n Wi-Fi, which enables it to connect to wireless networks for internet access and communication with other devices. Bluetooth 4.2 (BLE) is also supported, providing low-energy connectivity options for various applications.

- I/O Interfaces: The ESP32-CAM includes several General Purpose Input Output (GPIO) pins, which can be used to connect to other sensors, actuators, and peripherals. The board also features a UART (Universal Asynchronous Receiver-Transmitter) interface, which is used for programming and communication. Additionally, the board includes SPI and I2C interfaces, further expanding its connectivity options.

## 2.3 Programming Environments

The ESP32-CAM can be programmed using a variety of development environments, making it accessible to both beginners and experienced developers. Some of the popular programming environments include:

- Arduino IDE: The ESP32-CAM is fully compatible with the Arduino IDE, a popular platform among hobbyists and makers. The Arduino IDE provides an easy-to-use interface and a vast library of resources and examples to get started with the ESP32-CAM.
- MicroPython: For those who prefer Python, the ESP32-CAM can be programmed using MicroPython, a lightweight implementation of Python designed for microcontrollers. MicroPython offers a simple and intuitive syntax, making it an excellent choice for rapid prototyping and experimentation.
- ESP-IDF: For more advanced users, Espressif's own IoT Development Framework (ESP-IDF) provides a comprehensive set of tools and libraries for developing applications on the ESP32 platform. ESP-IDF offers greater control over the hardware and more advanced features, suitable for complex and large-scale projects.

## 2.4 Camera Specifications

The ESP32-CAM includes an OV2640 camera module. The device also supports OV7670 cameras. The OV2640 has the following specifications:

- 2 Megapixel sensor
- Array size UXGA 1622×1200
- Output formats include YUV422, YUV420, RGB565, RGB555 and 8-bit compressed data
- Image transfer rate of 15 to 60 fps

## 2.5 ESP32 CAM and ESP32 CAM MB

The ESP32-CAM AI-Thinker module is an ESP32 development board with an OV2640 camera, microSD card support, on-board flash lamp and several GPIOs to connect peripherals. And it costs just a few bucks. However, one of the biggest hassles when working with the ESP32-CAM AI-Thinker module is uploading code to the board. The AI-Thinker board doesn’t have a built-in USB programmer. So we have to use FTDI Programmer, USB TTL Programmer or ESP32-CAM-MB USB programmer.

In this case for Live Streaming we will use ESP32 CAM MB USB Programmer and Arduino IDE for uploading the code. The ESP32-CAM AI-Thinker MB programmer is a shield that is attach to ESP32-CAM board GPIOs. Figure 2.1 shows the programmer and the ESP32-CAM side by side.

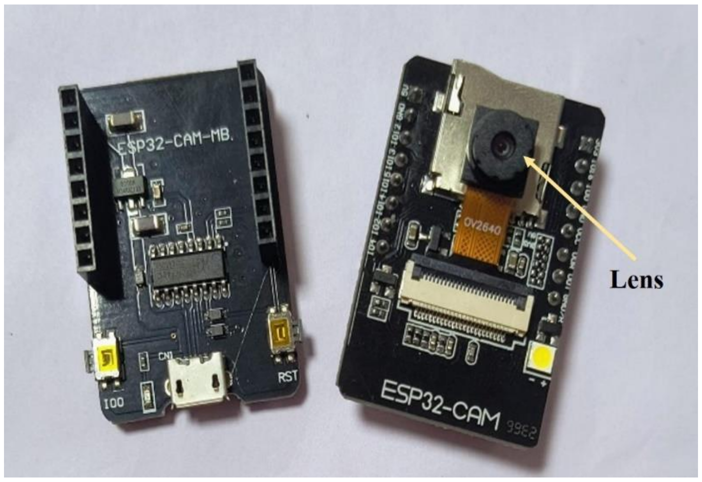


**Fig 2.1**: ESP32-CAM-MB and ESP32 CAM

The programmer comes with the CH340C USB to serial chip. This allows you to program the ESP32-CAM using the USB port on the shield. Additionally, the shield also comes with a RESET and a BOOT (IO0) buttons. This may be useful to easily reset the ESP32-CAM or put it into flashing mode.

### 2.6 Programming ESP32 CAM

- To program the ESP32 CAM, the ESP32 board in the Arduino IDE is installed first using the following instructions:
    1. Files > Preferences > Additional Board Manager URLs
    2. Copying the following line to the Additional Boards Manager URLs field.
    3. https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
    4. Add OK.
    5. Boards manager > Search for ESP32 and press the install button for esp32 by Espressif Systems version 3.X.
- The ESP32 CAM MB programmer is connected to the ESP32-CAM as shown in the following image and then is connect the board to computer or PC using a USB cable.

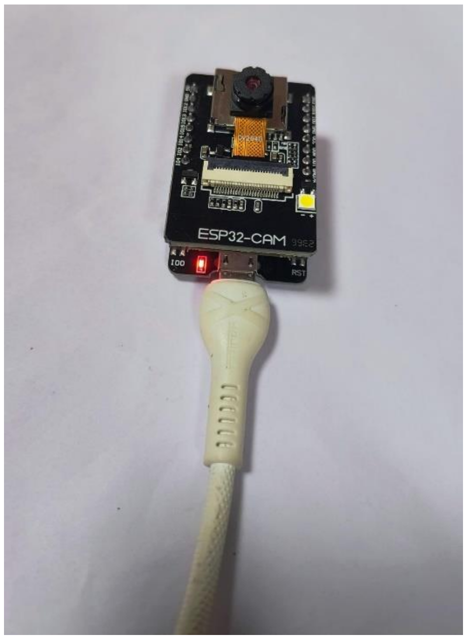


**Fig 2.2**: Connection

- In Arduino IDE, Tools > Board and select AI-Thinker ESP32-CAM and Tools > Port and select the COM port the ESP32-CAM is connected to.
- Files > Examples > ESP32 > Camera > CameraWebServer
- The code for video streaming will be displayed in the screen.
- The ESP32 CAM operates in two modes: Station Mode and Soft Access Point Mod
- We will make our ESP32 CAM work in Soft Access Point mode for this we have to make a small change in the program code.
- We have to customize the SSID and Password. We can set any name to SSID and Password according to our choice.
- In the *setup()*, remove the following lines (set the ESP32 as a station):

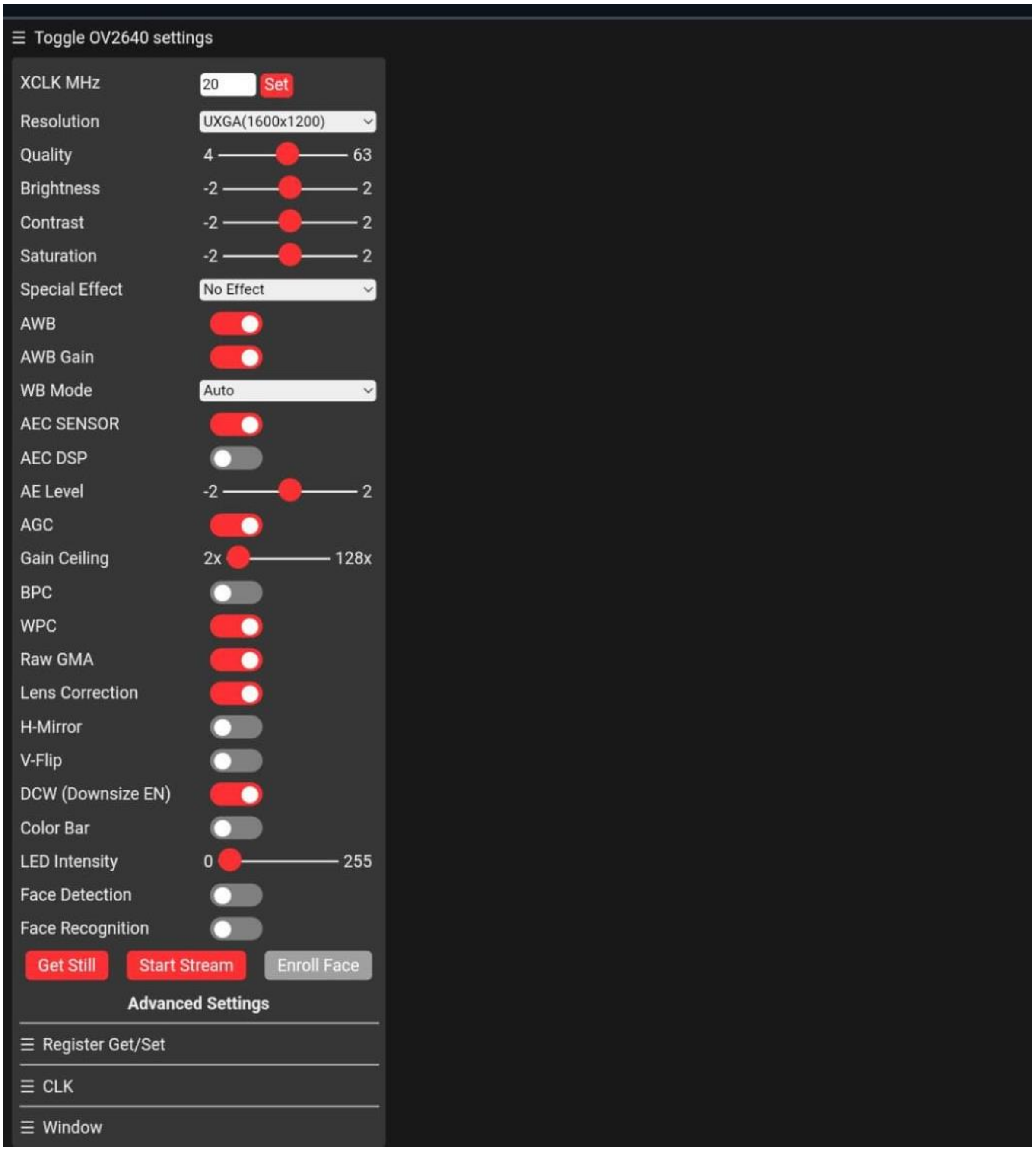
```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
```

And add the following to set the ESP32 as an access point using the *softAP()* method:

```
WiFi.softAP(ssid, password);
```

- We're ready to upload the code in the ESP32 CAM board.

- After uploading the code, you can connect to the ESP32-CAM access point to access the web server.
- Open your web browser and type the IP address **192.168.4.1**. The video streaming web server page should load:

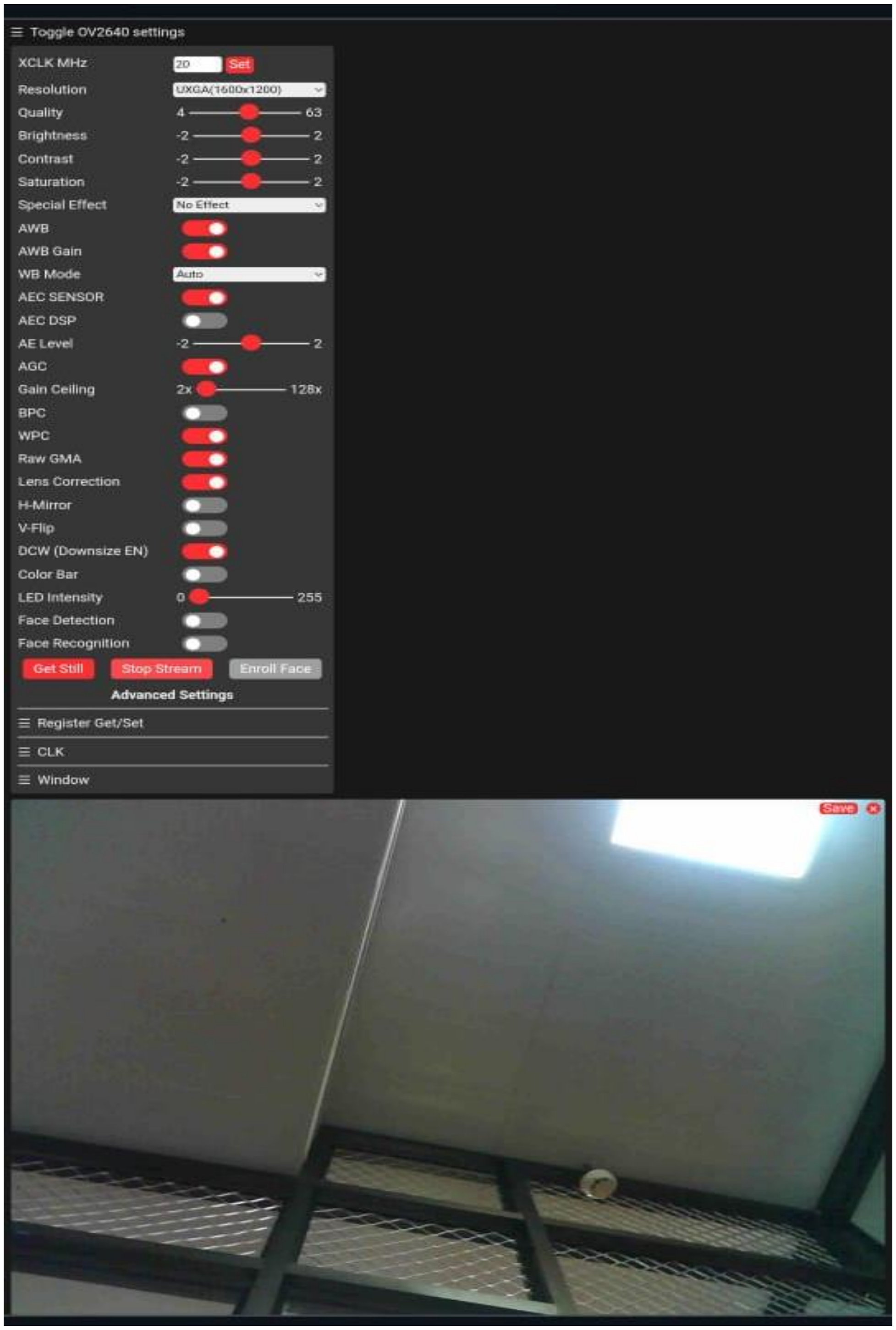


☰ Toggle OV2640 settings

| | |
|---|---|
| XCLK MHz | 20 Set |
| Resolution | UXGA(1600x1200) |
| Quality | 4 – 63 |
| Brightness | -2 – 2 |
| Contrast | -2 – 2 |
| Saturation | -2 – 2 |
| Special Effect | No Effect |
| AWB | |
| AWB Gain | |
| WB Mode | Auto |
| AEC SENSOR | |
| AEC DSP | |
| AE Level | -2 – 2 |
| AGC | |
| Gain Ceiling | 2x – 128x |
| BPC | |
| WPC | |
| Raw GMA | |
| Lens Correction | |
| H-Mirror | |
| V-Flip | |
| DCW (Downsize EN) | |
| Color Bar | |
| LED Intensity | 0 – 255 |
| Face Detection | |
| Face Recognition | |

**Fig 2.3**: Web Server Page

- Click on Start stream and the camera is ready for live streaming.



☰ Toggle OV2640 settings

| | |
|---|---|
| XCLK MHz | 20 Set |
| Resolution | UXGA(1600x1200) |
| Quality | 4 ─●─ 63 |
| Brightness | -2 ─●─ 2 |
| Contrast | -2 ─●─ 2 |
| Saturation | -2 ─●─ 2 |
| Special Effect | No Effect |
| AWB | ● |
| AWB Gain | ● |
| WB Mode | Auto |
| AEC SENSOR | ● |
| AEC DSP | ○ |
| AE Level | -2 ─●─ 2 |
| AGC | ● |
| Gain Ceiling | 2x ●── 128x |
| BPC | ○ |
| WPC | ● |
| Raw GMA | ● |
| Lens Correction | ● |
| H-Mirror | ○ |
| V-Flip | ○ |
| DCW (Downsize EN) | ● |
| Color Bar | ○ |
| LED Intensity | 0 ●── 255 |
| Face Detection | ○ |
| Face Recognition | ○ |

**Fig 2.4**: Demonstration of Live Stream

- We can save the image using the Save button in the stream.

## 2.7 Applications

- Home Security and Surveillance: Enables DIY home security with live video streaming and remote monitoring capabilities.
- IoT (Internet of Things) Projects: Facilitates environmental monitoring and smart agriculture through visual data capture and transmission.
- Educational and DIY Projects: Used in STEM education for school students and DIY automation for integrating camera functionalities.
- Healthcare and Telemedicine: Supports remote patient monitoring and telehealth with visual data for diagnostics.
- Retail and Customer Analytics: Enhances retail operations with customer behavior analysis and inventory management using visual data verification.

## 2.8 Pros and Cons

Pros

- Cost-Effective: The ESP32-CAM is very affordable, making it accessible for a wide range of users, from hobbyists and students to professionals. This cost-effectiveness allows for extensive experimentation and application development without significant financial investment.
- Integrated Camera and Wireless Connectivity: The board features an OV2640 camera module capable of capturing high-quality images and video, combined with built-in Wi-Fi and Bluetooth capabilities. This integration makes it ideal for applications in surveillance, IoT, and wireless communication, providing a comprehensive solution in a single package.
- Versatile Software Support: The ESP32-CAM is compatible with multiple development environments, including Arduino IDE, MicroPython, and ESP-IDF. This wide range of software support provides flexibility for developers, enabling them to choose the platform that best fits their skills and project requirements.
- MicroSD Card Slot: The onboard microSD card slot allows for expanded storage capabilities, which is particularly useful for applications that require storing large amounts of data, such as video recordings or extensive image captures. This feature enhances the board's utility in data-intensive applications.

Cons

- Camera Quality: The ESP32-CAM's OV2640 camera module, while functional, may not meet the resolution and clarity requirements of more demanding applications, limiting its suitability for tasks requiring high-quality visual output.
- Lag and Performance Issues: During active video streaming or image capture, the ESP32-CAM can experience lag and performance degradation, especially under simultaneous tasks like Wi-Fi transmission and image processing, impacting real-time applications such as surveillance and robotics.
- Complex Initial Setup: Setting up the ESP32-CAM for the first time, including installing drivers and configuring the development environment, can be challenging for beginners unfamiliar with embedded systems and IoT devices.
- Memory Constraints: With only 520 KB of SRAM available, the ESP32-CAM may struggle with memory-intensive applications, especially those requiring extensive image processing or handling large buffers of data.

# Chapter 3: Arduino Radar Sensor

Radar (Radio Detection and Ranging) is a system that uses radio waves to determine the distance (ranging), direction (azimuth and elevation angles), and radial velocity of objects relative to the site. It is a radio determination method used to detect and track aircraft, ships, spacecraft, guided missiles, motor vehicles, map weather formations, and terrain. A radar system consists of a transmitter producing electromagnetic waves in the radio or microwaves domain, a transmitting antenna, a receiving antenna (often the same antenna is used for transmitting and receiving) and a receiver and processor to determine properties of the objects.

Radar was developed secretly for military use by several countries in the period before and during World War II. The modern uses of radar are highly diverse, including air and terrestrial traffic control, radar astronomy, air-defense systems, anti-missile systems, marine radars to locate landmarks and other ships, aircraft anti-collision systems, ocean surveillance systems, outer space surveillance and rendezvous systems, meteorological precipitation monitoring, radar remote sensing, altimetry and flight control systems, guided missile target locating systems, self-driving cars, and ground-penetrating radar for geological observations. Modern high tech radar systems use digital signal processing and machine learning and are capable of extracting useful information from very high noise levels.

## 3.1 Project Model

- To show the RADAR system using Arduino has been made using an Arduino, Ultrasonic sensor, servo motor, jumper wires and the output is generated in the software- Processing Development Environment (PDE) along with Arduino IDE.
- Apparatus:
    1. Arduino Uno Board: Arduino UNO is a microcontroller board based on the ATmega328P. 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.

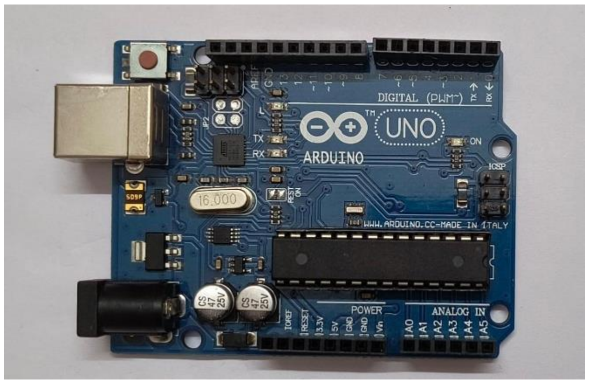


**Fig 3.1**: Arduino Uno Board

2. Ultrasonic Sensor HC-SR04: The HC-SR04 is an affordable and easy to use distance measuring sensor which has a range from 2cm to 400cm (about an inch to 13 feet). The sensor is composed of two ultrasonic transducers. One is transmitter which outputs ultrasonic sound pulses and the other is receiver which listens for reflected waves.

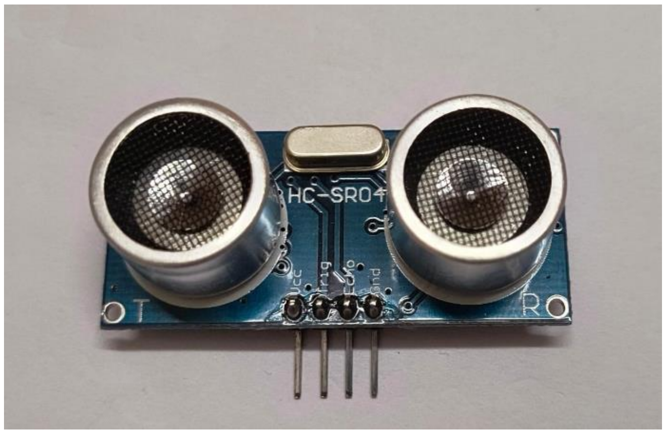


**Fig 3.2:** Ultrasonic Sensor HC-SR04

3. Servo motor SG90: Micro Servo Motor SG90 is a tiny and lightweight server motor with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. It has three wires, RED- power supply, BLACK-ground, YELLOW-signal.

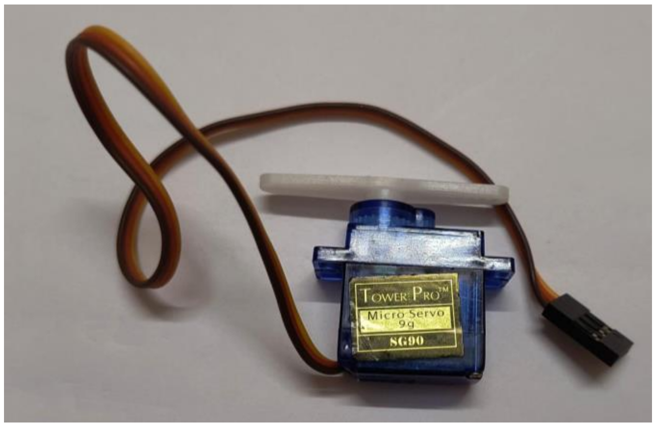


**Fig 3.3:** Servo motor SG90

4. Jumper wires: Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboards and other prototyping tools in order to make it easy to change a circuit as needed.

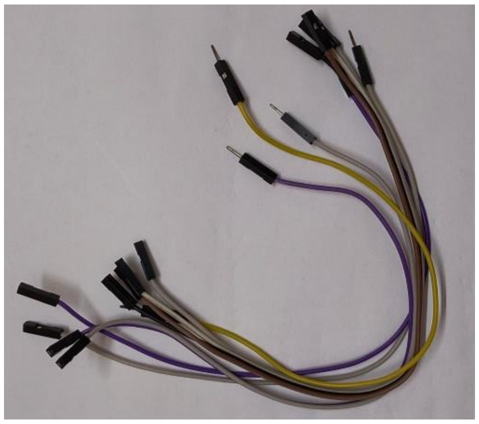


**Fig 3.4**: Jumper wires

5. USB 2.0 Cable for Arduino board: This cable is used to power the Arduino Uno board and for uploading the code in the Uno board.

**Fig 3.5**: USB 2.0 Cable

## 3.2 Software or platforms:

- Arduino IDE: The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.
- Processing Development Environment (PDE): Processing is free, open source software based on Java. It is designed for the visual arts community for creating drawings, animations, and interactive programs. Processing is open source and is available for macOS, Windows, and Linux. Projects created with Processing are also cross-platform, and can be used on macOS, Windows, Android, Raspberry Pi, and many other Linux platforms.

## 3.3 Connections:

- The Signal pin of Servo motor is connected to Pin 11 of Arduino Board. The GND and Power signals are connected to the GND and 5V of the Arduino Board.
- The Trig and Echo pins of the Ultrasonic sensor is connected to the Pin 9 and 10 of the Arduino Board. The GND and VCC pins are connected to the GND and 5V of the Arduino Board. The following circuit diagram is made using Tinkercad.

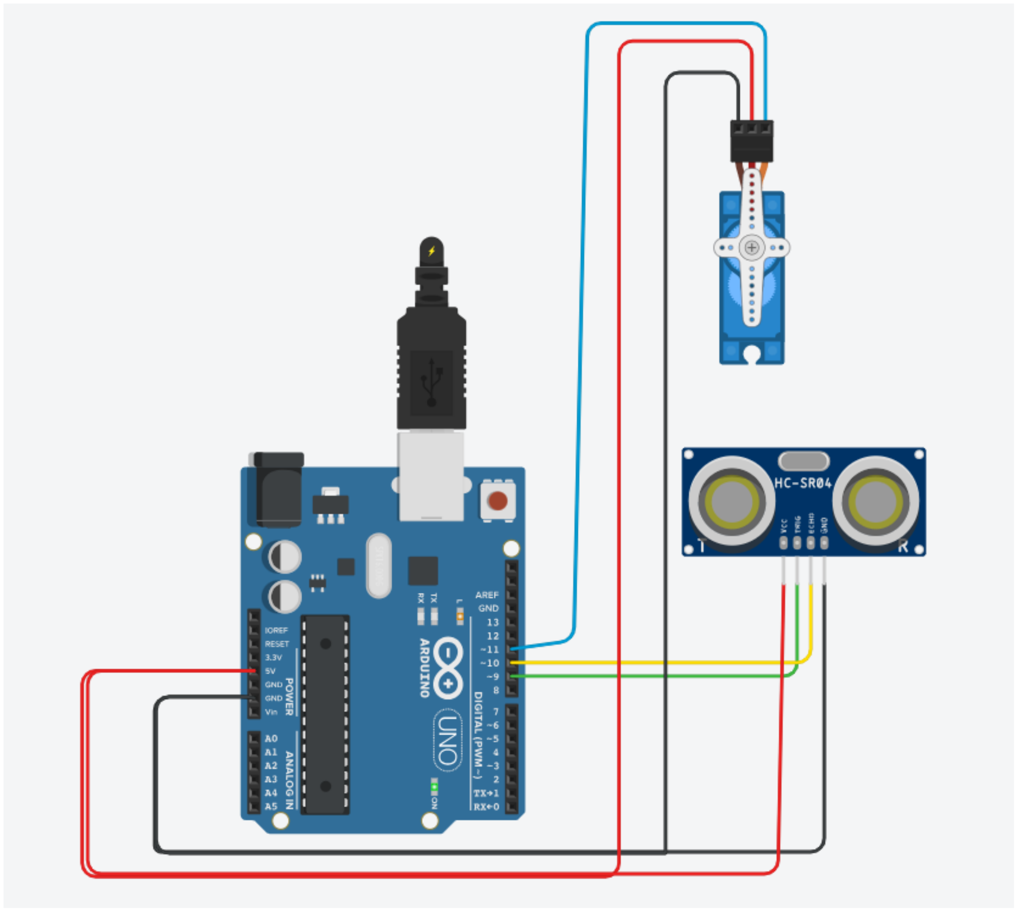


**Fig 3.6**: Circuit connection for the Arduino RADAR Sensor

## 3.4 Working:

- The Arduino Code is uploaded to the Arduino Uno Board through Port COM7 using a USB Cable and the servo motor starts rotating in 180 degree. The ultrasonic sensor is attached above the servo motor.
- The Processing code is run to display the graphical representation of the data. The ultrasonic sensor is represented in a radar type display. When the ultrasonic sensor detects any object within its range, the object will be shown on screen graphically.
- The code of both Arduino IDE and Processing is mentioned in the **Appendix**.

## 3.5 Performance:

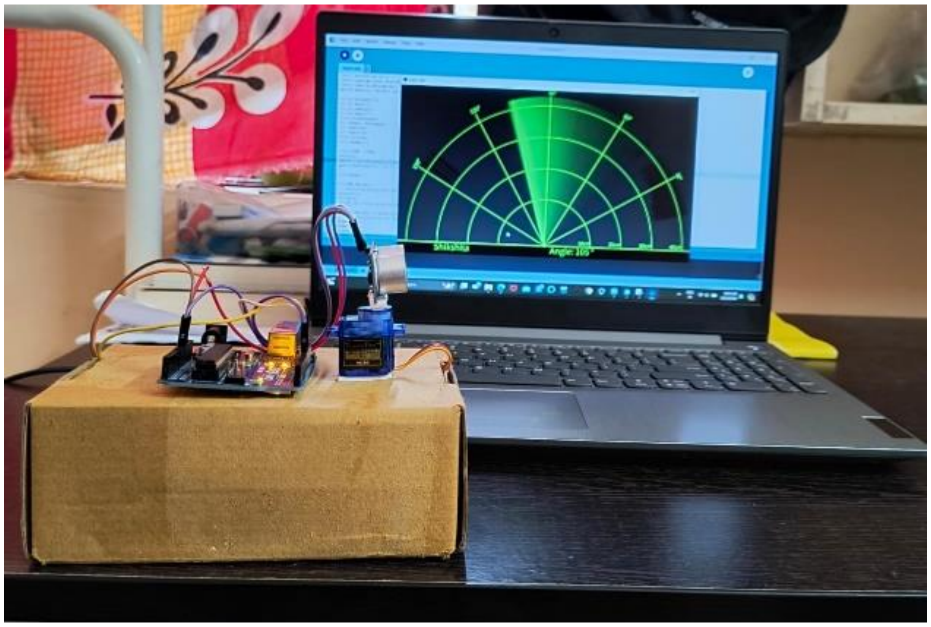


**Fig 3.7:** No obstacle detected by the ultrasonic radar sensor

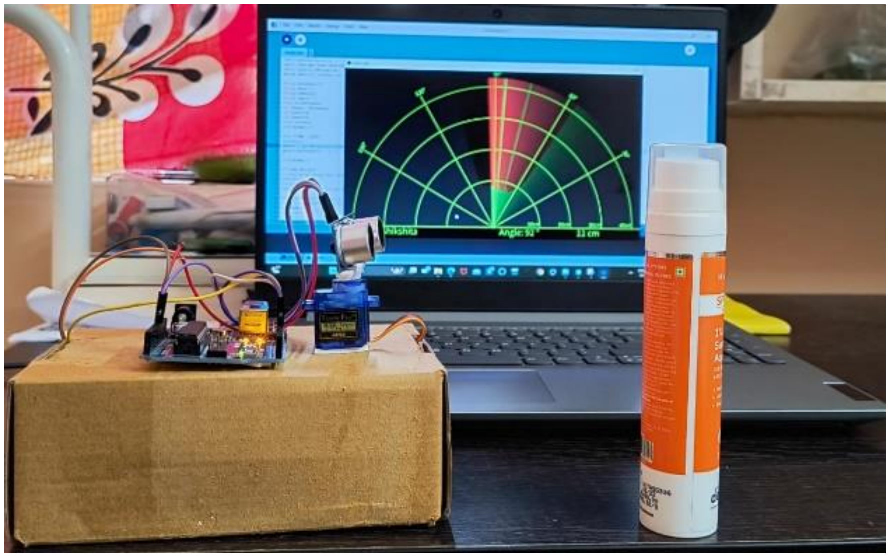


**Fig 3.8**: Obstacle detected by the ultrasonic radar sensor

# Chapter 4: Obstacle Avoidance Robot Car

Obstacle avoidance, in robotics, is a critical aspect of autonomous navigation and control systems. It is the capability of a robot or an autonomous system/machine to detect and circumvent obstacles in its path to reach a predefined destination. This technology plays a pivotal role in various fields, including industrial automation, self-driving cars, drones, and even space exploration. Obstacle avoidance enables robots to operate safely and efficiently in dynamic and complex environments, reducing the risk of collisions and damage.For a robot or autonomous system to successfully navigate through obstacles, it must be able to detect such obstacles. This is most commonly done through the use of sensors, which allow the robot to process its environment, make a decision on what it must do to avoid an obstacle, and carry out that decision with the use of its effectors, or tools that allow a robot to interact with its environment.

## 4.1 Project Model

- There are several methods for robots or autonomous machines to carry out their decisions in real-time. Some of these methods include sensor-based approaches, path planning algorithms, and machine learning techniques. In our case, we are using sensors-based approach using Ultrasonic sensor.
- Apparatus:

  1. Arduino Mega 2560: The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), 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 an AC-to-DC adapter or battery to get started. Fig 4.1 shows an image of Arduino Mega 2560.

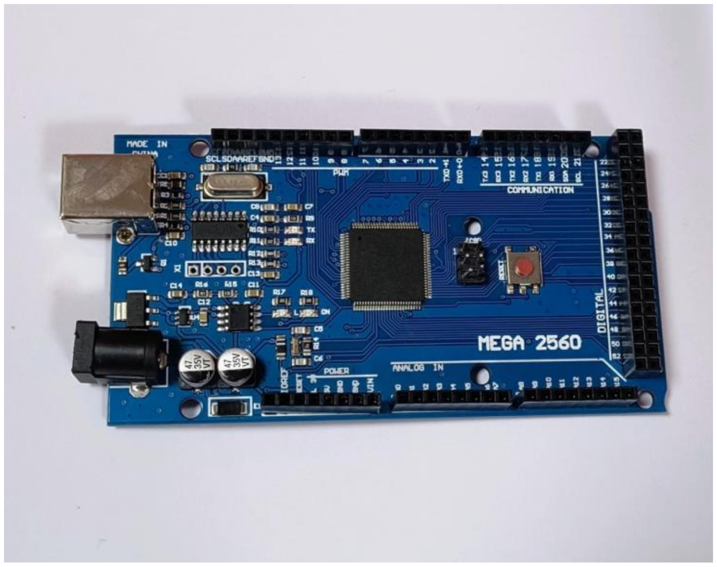


**Fig 4.1:** Arduino Mega 2560

2. Ultrasonic Sensor: Ultrasonic sensor works on the principle of propagation of ultrasonic waves in air. Ultrasonic waves are sound waves which are of the frequencies above 20 KHz, this is below the audible level of human ear. The sonar system in ultrasonic sensor can detect an object in the range of 2 cm to 400 cm. The presence of an object in front of the sensor is detected by the phenomenon of reflection. Ultrasonic sensor cannot detect reflected waves when the angle of reflection is high.An ultrasonic sensor consists of 4 pins, VCC, TRIG, ECHO and GND. TRIG and ECHO pins of this sensor are designed for transmitting and receiving ultrasonic waves respectively.

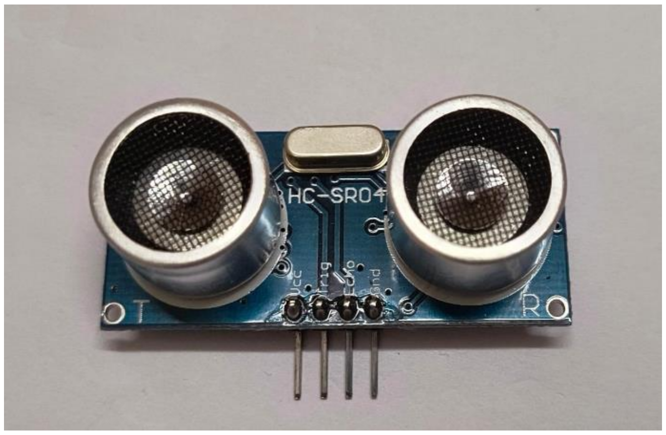


**Fig 4.2:** Ultrasonic Sensor HC-SR04

3. L298N Motor Driver: L298N is a high voltage, high current dual full-bridge motor driver module. It is used for controlling DC Motor, and can control both the speed and rotation direction of two DC motors. The module consists of an L298N dual-channel H-Bridge motor driver IC. It is a double H-Bridge design, with a low level of heat and interference. It can drive inductive loads such as relays, solenoids, DC and stepping motors.

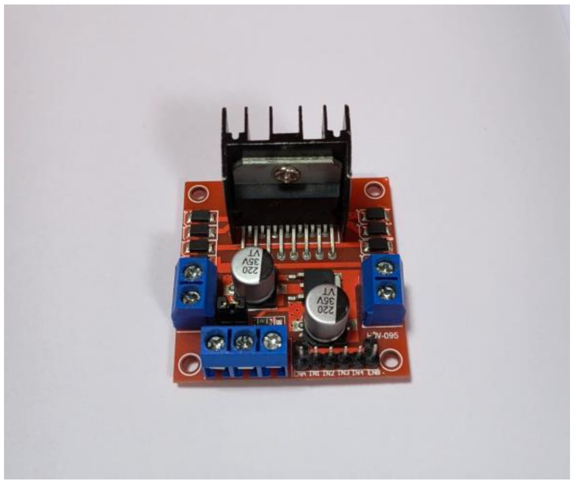


**Fig 4.3**: L298N Motor Driver

4. DC Motor: Dual shaft geared motors are useful in robotics applications. Shaft on both sides allows the user to use a Wheel and an Encoder simultaneously. Not only this, the user gets enough headroom for the orientation of the motor and can be used in any orientation as per the application is required. It is a plastic based geared Dual shaft DC motor operating between a voltage range of 3V to 9V and has a torque of 0.8 Kg/cm with an RPM of 100 which is pretty decent for most of the applications. It is recommended to use this motor with more than 5V to have optimum torque in working condition.

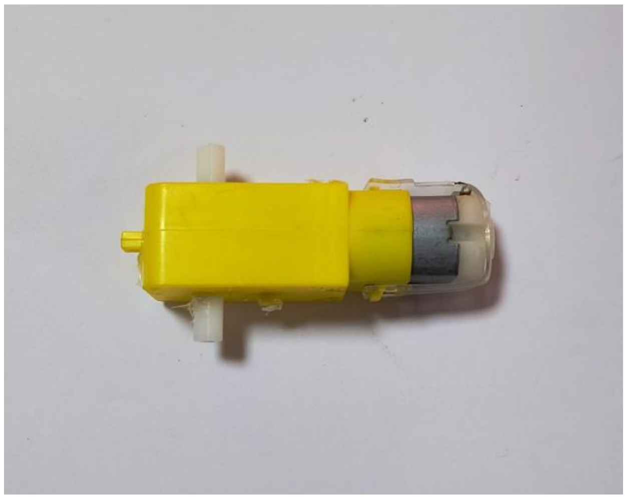


**Fig 4.4**: DC Motor

5. Wheels: These are 65mm Robot Wheels for a Dc geared motor. The wheels are suitable for all BO motors and they are one of the most commonly used components in robotics.
Robot Wheel Composition: High quality rubber wheel on plastic hub.

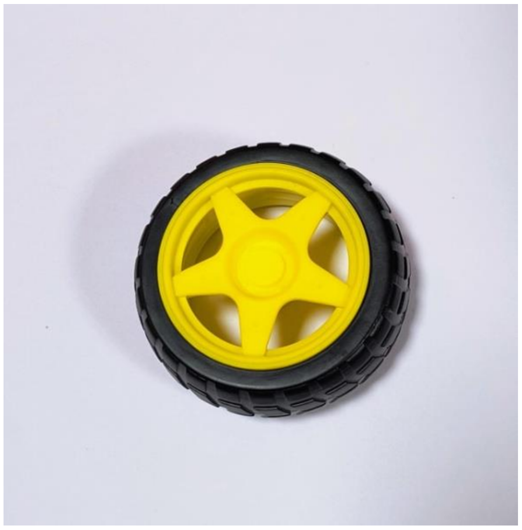


**Fig 4.5**: Wheel

6. Castor Wheel: A castor wheel is a relatively small un driven wheel, meaning that it is free-rolling (as opposed to powered). They are designed to be attached to the bottom of a larger

object, to enable easy movement across a floor or other hard surface. Castor wheels are often referred to simply by the standalone term 'castors'. They are also commonly identified as 'caster wheels' or 'casters' by some suppliers and manufacturers.

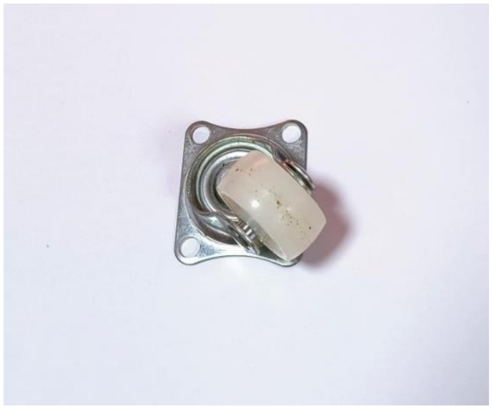


**Fig 4.6**: Castor Wheel

7. Battery: Batteries in robotic cars provide power, store energy, and enable zero-emission operation. They offer quiet and smooth acceleration, autonomy, and faster charging times. Advancements in battery technology improve performance and range, making electric vehicles increasingly viable and efficient. Batteries are a crucial component in the development of robotic cars.

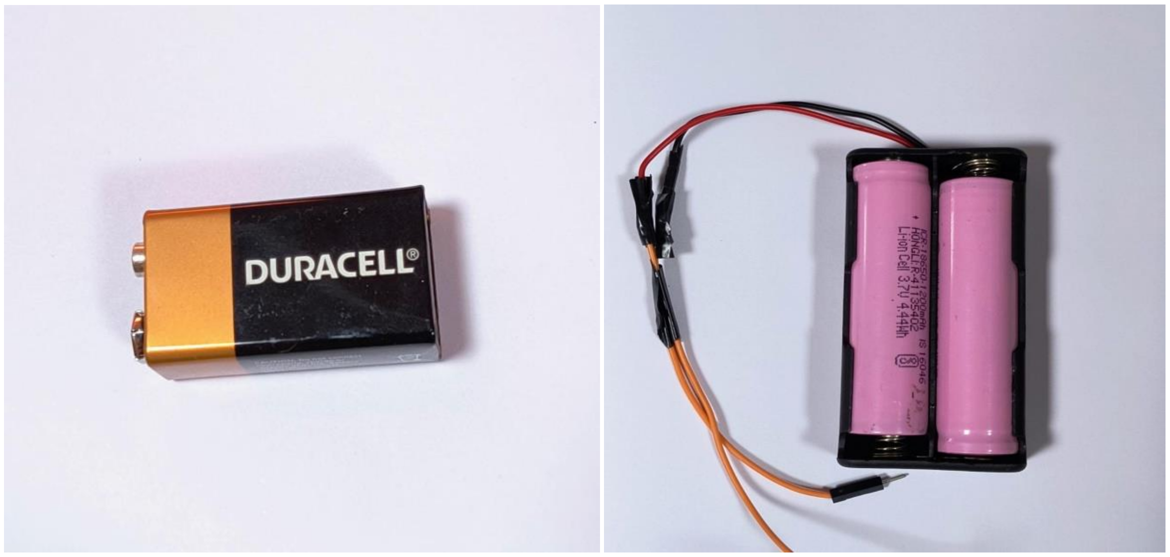


**Fig 4.7**: Battery (Power Supplies)

8. Jumper Wires: Jumper wires: Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper

wires are typically used with breadboards and other prototyping tools in order to make it easy to change a circuit as needed.

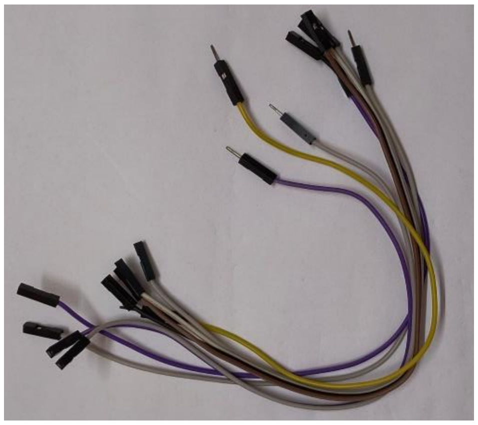


**Fig 4.8**: Jumper wires

9. USB 2.0 Cable for Arduino board: This cable is used to power the Arduino Uno board and for uploading the code in the Uno board.

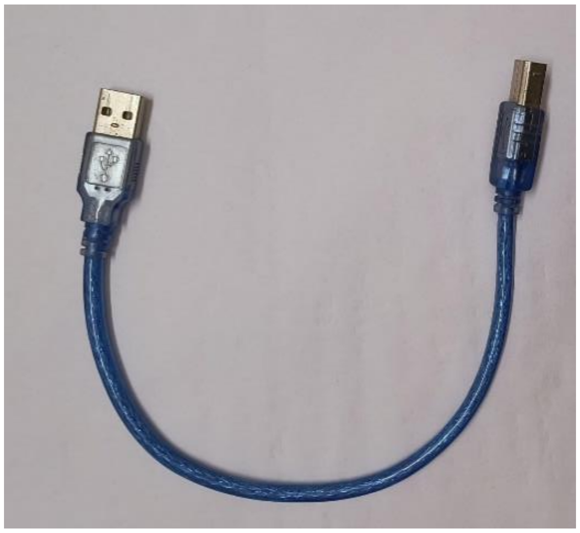


**Fig 4.9: USB 2.0 Cable**

10. Robot Car Base plate: The base plate of a robot car is the foundation that mounts various components like motors, batteries, electronics, and suspension. It's designed to be strong, durable, and lightweight, typically made of materials like aluminium, steel, or carbon fibre, and plays a crucial role in determining the vehicle's performance and stability.

## 4.2 Software

- Arduino IDE: The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

## 4.3 Connections

- The Trig pin and Echo pin of the ultrasonic sensor is connected to the pin 16 and 17 of the Arduino board and the GND and VCC is connected to the GND and 5V pin of the Arduino.
- The IN1, IN2, IN3 and IN4 pins of L298N motor is connected to pins 2, 3, 14 and 15 of the Arduino. A battery of 9V is connected to the 12V pin of the driver. The supply must be 6-12V because the motors draw a lot of current.
- Two motors are used in this model. Motor A is connected to OUT1 and OUT2 of the driver and Motor B is connected to OUT3 and OUT4 of the driver.
- A separate supply is given to power the Arduino board.
- The entire set up is mounted to the base plate as shown in Fig 4.10:

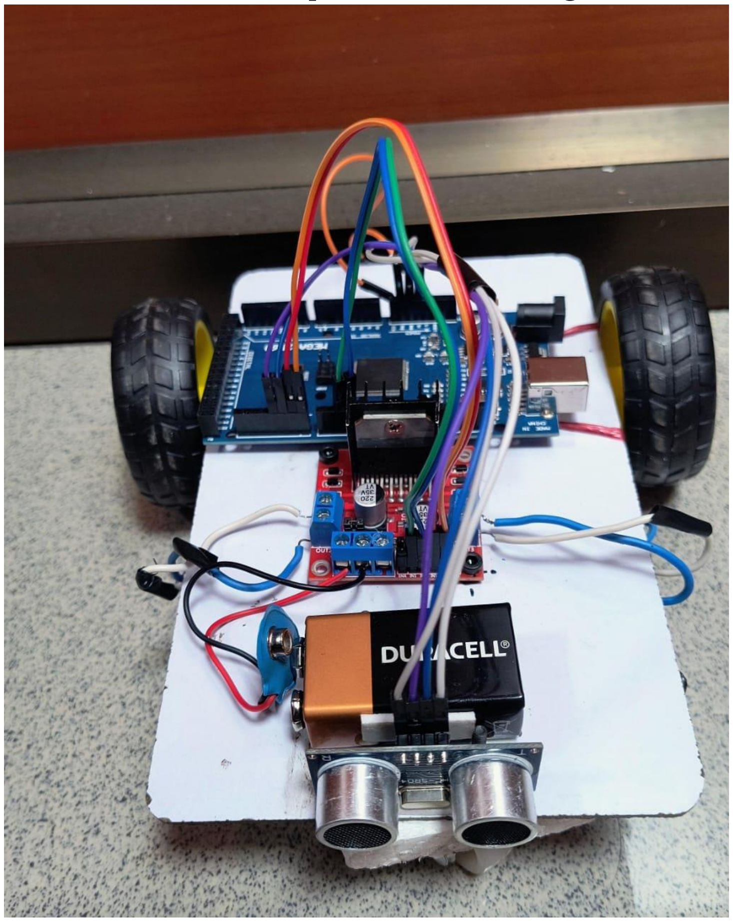


**Fig 4.10:** Robot Car Setup

## 4.4 Working

- The Arduino code to run the robot car is uploaded to the Arduino board and the robot car is ready to run on its own. The code is given in the **Appendix**.

## Conclusion

In conclusion, this report has explored various practical applications of Internet of Things (IoT) and embedded systems through hands-on Arduino-based projects. It has highlighted Arduino's versatility in integrating sensors and its role in powering real-world applications across different sectors. From the ESP32-CAM's live streaming capabilities to radar sensor integration and obstacle avoidance robotics, each project showcased Arduino's potential in enhancing surveillance, environmental monitoring, and autonomous navigation systems. These applications not only underscore Arduino's adaptability but also demonstrate its effectiveness in translating theoretical concepts into functional prototypes. Moreover, the projects emphasized the importance of interdisciplinary skills such as coding, hardware interfacing, and data visualization, essential for successful IoT implementations. They illustrated how IoT technologies can address contemporary challenges and improve operational efficiencies across industries. Looking ahead, the field of IoT continues to evolve rapidly, presenting opportunities for innovation and growth. Arduino's accessibility and community support make it a valuable platform for enthusiasts and professionals alike to explore and advance IoT solutions further.

In essence, this report serves as a practical guide and inspiration for leveraging Arduino and IoT technologies in creating impactful solutions that address real-world needs, driving forward the integration of smart and connected systems into everyday life.

## Appendix

### Arduino Code for Radar Sensor:

```
#include <Servo.h>.
const int trigPin =9 ;
const int echoPin = 10;
// defining time and distance
long duration;
int distance;
Servo myServo; // Object servo
void setup() {
  pinMode(trigPin, OUTPUT); // trigPin as an Output
  pinMode(echoPin, INPUT); // echoPin as an Input
  Serial.begin(9600);
  myServo.attach(11); // Pin Connected To Servo
}
void loop() {
  // rotating servo i++ depicts increment of one degree
  for(int i=15;i<=165;i++){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();

  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
  // Repeats the previous lines from 165 to 15 degrees
  for(int i=165;i>15;i--){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
}
int calculateDistance(){

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance= duration*0.034/2;
  return distance;}
```

**Processing Code for Display of RADAR Sensor:**

```
import processing.serial.*;
import java.awt.event.KeyEvent;
import java.io.IOException;
Serial myPort;// defubes variables

String distance="";
String data="";
String noObject;
String angle="";
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;
void setup() {

size (1280  ,720);
smooth();
myPort = new Serial(this,"COM7", 9600); // change this accordingly
myPort.bufferUntil('.'); // reads the data from the serial port up to the character '.'. So actually it
reads this: angle,distance.
}
void draw() {

fill(98,245,31);
// simulating motion blur and slow fade of the moving line
noStroke();
fill(0,4);
rect(0, 0, width, height-height*0.065);

fill(98,245,31); // green color
// calls the functions for drawing the radar
drawRadar();
drawLine();
drawObject();
drawText();
}
void serialEvent (Serial myPort) { // starts reading data from the Serial Port
// reads the data from the Serial Port up to the character '.' and puts it into the String variable
"data".
```

```
data = myPort.readStringUntil('.');
data = data.substring(0,data.length()-1);

index1 = data.indexOf(','); // find the character ',' and puts it into the variable "index1"
angle= data.substring(0, index1); // read the data from position "0" to position of the variable
index1 or thats the value of the angle the Arduino Board sent into the Serial Port
distance= data.substring(index1+1, data.length()); // read the data from position "index1" to the
end of the data pr thats the value of the distance

// converts the String variables into Integer
iAngle = int(angle);
iDistance = int(distance);
}
void drawRadar() {
pushMatrix();
translate(width/2,height-height*0.074); // moves the starting coordinats to new location
noFill();
strokeWeight(2);
stroke(98,245,31);
// draws the arc lines
arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI);
arc(0,0,(width-width*0.27),(width-width*0.27),PI,TWO_PI);
arc(0,0,(width-width*0.479),(width-width*0.479),PI,TWO_PI);
arc(0,0,(width-width*0.687),(width-width*0.687),PI,TWO_PI);
// draws the angle lines
line(-width/2,0,width/2,0);
line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));
line((-width/2)*cos(radians(30)),0,width/2,0);
popMatrix();
}
void drawObject() {
pushMatrix();
translate(width/2,height-height*0.074); // moves the starting coordinats to new location
strokeWeight(9);
stroke(255,10,10); // red color
pixsDistance = iDistance*((height-height*0.1666)*0.025); // covers the distance from the sensor
from cm to pixels
// limiting the range to 40 cms
if(iDistance<40){
// draws the object according to the angle and the distance
line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)),(width-
width*0.505)*cos(radians(iAngle)),-(width-width*0.505)*sin(radians(iAngle)));
}
popMatrix();
}
void drawLine() {
pushMatrix();
strokeWeight(9);
stroke(30,250,60);
translate(width/2,height-height*0.074); // moves the starting coordinats to new location
line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-height*0.12)*sin(radians(iAngle)));
// draws the line according to the angle
popMatrix();
}
void drawText() { // draws the texts on the screen

pushMatrix();
if(iDistance>40) {
noObject = "Out of Range";
}
else {
noObject = "In Range";
}
fill(0,0,0);
noStroke();
rect(0, height-height*0.0648, width, height);
fill(98,245,31);
textSize(25);

text("10cm",width-width*0.3854,height-height*0.0833);
text("20cm",width-width*0.281,height-height*0.0833);
text("30cm",width-width*0.177,height-height*0.0833);
text("40cm",width-width*0.0729,height-height*0.0833);
textSize(40);
text("Shikshita", width-width*0.875, height-height*0.0277);
text("Angle: " + iAngle +" °", width-width*0.48, height-height*0.0277);
text("", width-width*0.26, height-height*0.0277);
if(iDistance<40) {
text(" " + iDistance +" cm", width-width*0.225, height-height*0.0277);
}
textSize(25);
fill(98,245,60);
translate((width-width*0.4994)+width/2*cos(radians(30)),(height-height*0.0907)-
width/2*sin(radians(30)));
rotate(-radians(-60));
text("30°",0,0);
resetMatrix();
```

```
translate((width-width*0.503)+width/2*cos(radians(60)),(height-height*0.0888)-
width/2*sin(radians(60)));
rotate(-radians(-30));
text("60°",0,0);
resetMatrix();
translate((width-width*0.507)+width/2*cos(radians(90)),(height-height*0.0833)-
width/2*sin(radians(90)));
rotate(radians(0));
text("90°",0,0);
resetMatrix();
translate(width-width*0.513+width/2*cos(radians(120)),(height-height*0.07129)-
width/2*sin(radians(120)));
rotate(radians(-30));
text("120°",0,0);
resetMatrix();
translate((width-width*0.5104)+width/2*cos(radians(150)),(height-height*0.0574)-
width/2*sin(radians(150)));
rotate(radians(-60));
text("150°",0,0);
popMatrix();
```

**Arduino code for Robot Car:**

```
 const int trigPin =16;
const int echoPin =17;
const int in1 = 2;
const int in2 = 3;
const int in3 = 14;
const int in4 = 15;

void setup()
{
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode (in1, OUTPUT);
  pinMode (in2, OUTPUT);
  pinMode (in3, OUTPUT);
  pinMode (in4, OUTPUT);
}
long duration, distance;

void loop()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration/58.2;
   if (distance >20) {
    // Stop the car
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);
    delay(500); // Wait for half a second


  } else {
    // Move forward
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
  }
  delay(100);
}
```

**References:**

1. https://randomnerdtutorials.com/upload-code-esp32-cam-mb-usb - ESP32 and ESP32-CAM-MB.
2. https://randomnerdtutorials.com/esp32-cam-access-point-ap-web-server - Programming ESP32.
3. https://processing.org/download -Processing IDE.
4. https://en.wikipedia.org/wiki/Radar - RADAR
5. https://www.arduino.cc/education/visualization-with-arduino-and-processing - Processing IDE
6. https://www.domestika.org/en/blog/11113-what-is-tinkercad-learn-3d-design-from-scratch - Tinker Cad
7. https://en.wikipedia.org/wiki/Obstacle_avoidance - Obstacle Avoidance