**ABSTRACT**

This project focuses on the development of a human detection system using ultrasonic sensors, designed to assist in search and rescue operations during disaster scenarios, such as earthquakes, floods, or fires. The system employs ultrasonic technology to detect human presence by measuring the distance between the sensor and objects in its vicinity. The sensor continuously emits sound waves and calculates the time taken for the sound waves to return, allowing it to estimate the distance to nearby objects. When a human is detected within a specific range (typically between 20 cm and 200 cm), the system triggers an alert through a buzzer and a visual indicator (LED). This alert helps rescue teams identify areas where humans may be trapped under debris or in hard-to-reach places. The system operates in real-time and provides immediate feedback, making it a valuable tool for disaster recovery efforts. The detection algorithm is based on simple distance measurements, ensuring accurate identification of human presence while reducing false positives from other objects. This solution offers a cost-effective, reliable, and easy-to-deploy tool for emergency responders, improving efficiency in locating survivors and accelerating recovery operations. The system can be further enhanced with additional sensors or wireless communication for broader applications.

# 1. INTRODUCTION

### OVERVIEW OF THE PROJECT

In recent years, robotics and Internet of Things (IoT) technologies have evolved rapidly, leading to the development of smart systems capable of performing tasks autonomously and interacting with the cloud for data collection and analysis. One such application is human detection robots, which can be used in various fields such as surveillance, security, automation, and healthcare.

This project focuses on building a Human Detection Robot using an ESP32 microcontroller, an HC-SR04 ultrasonic sensor, an L298N motor driver, and DC motors. The robot’s core functionality is to detect humans or objects within a specific range using the ultrasonic sensor. When a human is detected, the robot is programmed to respond by moving forward or performing specific tasks. Additionally, the ESP32 microcontroller allows the robot to communicate with an IoT platform (such as ThingSpeak), providing real-time data updates on human detection events.

The combination of robotics and IoT opens up exciting possibilities for monitoring, automation, and human-computer interaction. By leveraging the ESP32's wireless capabilities, this project enables remote monitoring and control, making it a flexible and scalable solution for a wide range of applications. The system is designed to be cost-effective, utilizing widely available components, while offering an easily customizable platform for future enhancements such as advanced motion control, alerts, and additional sensors.

This introduction highlights the growing potential for IoT-based robotics, and sets the foundation for creating a smart, human-aware system capable of real-time decision-making and communication.

# SYSTEM STUDY

### **EXISTING SYSTEM**

Currently, several systems integrate human detection using various sensors such as ultrasonic, infrared, and cameras. Most existing systems use ultrasonic sensors (like the HC-SR04) to detect obstacles or humans by measuring the distance to objects. These systems are often integrated with microcontrollers like Arduino or ESP32 for processing data and controlling motors. For movement, motor drivers like the L298N are used to control DC motors, enabling the robot to navigate autonomously. Many of these systems are designed for security and surveillance purposes, alerting users when a human is detected. IoT-based platforms like ThingSpeak and Blynk are commonly used to send real-time data and provide remote monitoring. However, these systems often have limitations in terms of sensor accuracy and range. Some systems require manual intervention, and the communication between devices is not always secure. There is also a focus on reducing power consumption, as many systems operate on batteries. These existing systems lay the foundation for building more advanced, reliable, and energy-efficient human detection robots.

* + 1. **DISADVANTAGES OF EXISTING SYSTEM**

1. Limited Sensor Accuracy
   * Ultrasonic and infrared sensors can produce false positives or negatives, especially in cluttered or noisy environments.
2. Short Detection Range
   * Most affordable sensors like HC-SR04 have a limited range (up to 4 meters), restricting coverage area.
3. Manual Intervention Required
   * Many systems still need human interaction for reset, updates, or control, reducing automation.
4. Unreliable Communication
   * IoT platforms like Blynk or ThingSpeak may experience latency, connectivity issues, or data loss, especially in low-bandwidth areas.
5. Security Vulnerabilities
   * Data transmitted over IoT platforms may be insecure, making the system vulnerable to hacking or unauthorized access.
6. High Power Consumption
   * Continuous sensing and communication drain batteries quickly, affecting long-term deployment.
7. Lack of Intelligence
   * Most systems cannot differentiate between a human and other moving objects, leading to incorrect alerts.
8. Poor Terrain Adaptability
   * Many rovers have limited movement capabilities on uneven or rough surfaces.
9. Limited Scalability
   * Hard to expand for large-scale monitoring or integrate with other smart devices without redesigning the system.
10. Environmental Sensitivity

* Sensors may be affected by lighting, temperature, or weather conditions, reducing reliability outdoors.

### **2.2 PROPOSED SYSTEM**

The proposed system aims to enhance human detection and automation by using an ESP32 microcontroller, an HC-SR04 ultrasonic sensor, L298N motor driver, and DC motors integrated with IoT communication. The ultrasonic sensor will detect humans within a specified range, triggering the robot to move forward or take predefined actions. The ESP32 will process sensor data, control motor movement, and send real-time status updates to an IoT platform like ThingSpeak for remote monitoring. The system will also feature an energy-efficient design, optimizing power consumption by utilizing low-power modes in the ESP32. Additionally, security measures will be implemented to protect communication between the robot and the IoT platform, ensuring safe data transmission. The robot will be designed for scalability, allowing future enhancements like additional sensors or advanced motion control. With this system, human detection will be more reliable, offering both autonomous operation and real-time monitoring for various applications, including surveillance, automation, and healthcare. The system's flexibility allows for integration with other devices, making it adaptable for different scenarios.

**2.2.1 ADVANTAGES OF PROPOSED SYSTEM**

1. Improved Accuracy and Responsiveness
   * The system uses real-time processing with the ESP32, allowing quick response to human detection events.
2. IoT-Based Remote Monitoring
   * Integration with platforms like ThingSpeak enables remote tracking and status updates, making it ideal for surveillance and automation.
3. Energy Efficiency
   * Utilizes the low-power modes of ESP32, ensuring longer battery life and efficient power management.
4. Enhanced Security
   * Implements secure data transmission protocols, reducing the risk of unauthorized access or data breaches.
5. Autonomous Operation
   * The robot can take predefined actions upon detection without human intervention, increasing automation and efficiency.
6. Scalability and Flexibility
   * Designed for easy integration of additional sensors and modules, allowing future upgrades (e.g., camera, PIR sensor, GPS, etc.).
7. Cost-Effective Design
   * Uses affordable components like the ESP32 and ultrasonic sensor, making the system budget-friendly for multiple applications.
8. Real-Time Decision Making
   * The system can instantly react to sensor inputs, making it suitable for critical applications like healthcare monitoring or emergency response.
9. Compact and Lightweight
   * The use of compact hardware ensures the rover is lightweight and portable, ideal for indoor and outdoor use.
10. Multiple Application Areas

* Can be used in surveillance, automation, healthcare, disaster response, and smart homes, making it a versatile solution.

**2.2.2 PROJECT DESCRIPTION**

* Purpose:  
   To assist in search and rescue operations during disaster scenarios like earthquakes, landslides, and building collapses.
* Autonomous Navigation:  
   The rover navigates autonomously using ultrasonic sensors to detect obstacles and avoid collisions.
* Human Detection:  
   Detects the presence of humans using proximity sensors and can be upgraded with thermal or camera-based detection.
* IoT Monitoring:  
   Sends real-time alerts and data to a remote server for monitoring by rescue teams.
* Power Efficiency:  
   Uses low-power modes of the ESP32 to maximize battery life in the field.
* Secure Communication:  
   Ensures secure data transmission between the rover and monitoring platform.
* Scalability:  
   Easily upgradable with additional sensors (IR, camera, gas, temperature) for improved detection.
* Applications:  
   Useful in rescue missions, hazardous zone inspection, and remote human detection.
* Cost-Effective Design:  
   Uses affordable, easily available components, making it ideal for large-scale deployment.

# 3.SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

**3.1.1 ESP 32 Dev Kit**

The ESP32 Dev Kit is a development board based on the ESP32 system-on-chip (SoC) manufactured by Espress if Systems. The ESP32 is a popular microcontroller that offers both Wi-Fi and Bluetooth connectivity, making it suitable for a wide range of IoT (Internet of Things) and embedded projects. Here's some key information about the ESP32 Dev Kit:

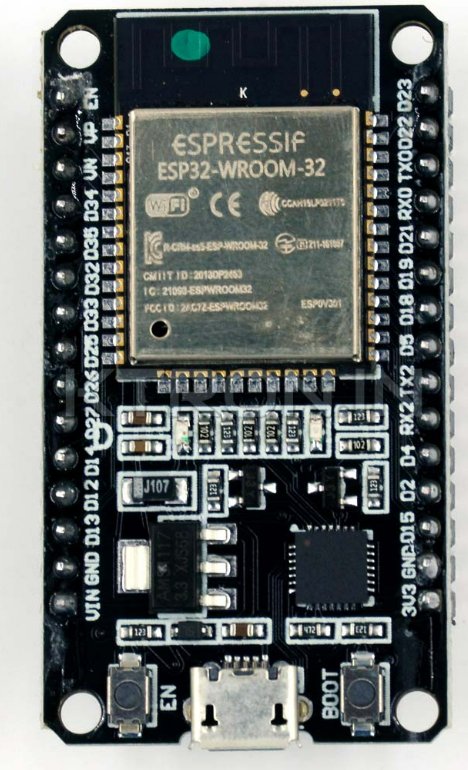
**ESP32 SoC**: The ESP32 is a dual-core microcontroller with a 32-bit CPU architecture. It features integrated Wi-Fi (802.11 b/g/n) and Bluetooth (BLE) capabilities. It also has a variety of peripherals, such as GPIO pins, UART, SPI, I2C, ADC, and more, making it versatile for different applications.

**Development Board**: The ESP32 Dev Kit is a development board designed to facilitate the prototyping and development of projects using the ESP32. It typically includes the ESP32 SoC, onboard power regulation, GPIO pins exposed for easy access, and a USB-to-Serial converter for programming and debugging.

**Programming**: You can program the ESP32 Dev Kit using the Arduino IDE, PlatformIO, Espressif's ESP-IDF (IoT Development Framework), and other development environments. It supports the C/C++ programming language.

**Power Supply**: The board can be powered through a USB connection or an external power supply, typically ranging from 5V to 12V, depending on the specific board and application requirements.

**I/O Pins**: The ESP32 Dev Kit typically exposes a variety of GPIO pins that you can use to connect sensors, displays, actuators, and other peripheral devices.

**Wi-Fi and Bluetooth**: The integrated Wi-Fi and Bluetooth capabilities of the ESP32 make it suitable for projects that require wireless communication. You can create Wi-Fi access points, connect to existing networks, and establish Bluetooth Low ****Energy (BLE) connections.

**Fig. ESP32 Dev Kit**

**Pin Description**

**Common GPIO Pins (General-Purpose Digital I/O)**:

**GPIO0**:

Common Uses: Boot mode selection, general-purpose digital I/O.

**GPIO2**:

Common Uses: Bootstrapping, general-purpose digital I/O.

**GPIO4**:

Common Uses: General-purpose digital I/O.

**GPIO5**:

Common Uses: General-purpose digital I/O.

**GPIO12**:

Common Uses: General-purpose digital I/O.

**GPIO13**:

Common Uses: General-purpose digital I/O.

**GPIO14**:

Common Uses: General-purpose digital I/O.

**GPIO15**:

Common Uses: Boot mode selection, general-purpose digital I/O.

**GPIO16**:

Common Uses: General-purpose digital I/O.

**GPIO17**:

Common Uses: General-purpose digital I/O.

**GPIO18**:

Common Uses: General-purpose digital I/O, often used as SPI clock (SCK) in SPI communication.

**GPIO19**:

Common Uses: General-purpose digital I/O, often used as SPI MISO in SPI communication.

**GPIO21**:

Common Uses: General-purpose digital I/O, often used as I2C SDA in I2C communication.

**GPIO22**:

Common Uses: General-purpose digital I/O, often used as I2C SCL in I2C communication.

**GPIO23**:

Common Uses: General-purpose digital I/O, often used as SPI MOSI in SPI communication.

**GPIO25**:

Common Uses: General-purpose digital I/O.

**GPIO26**:

Common Uses: General-purpose digital I/O.

**GPIO27**:

Common Uses: General-purpose digital I/O.

**I2C Pins (SDA and SCL)**:

* **SDA (Serial Data)** (Commonly GPIO 21): This pin is used for the data line in I2C communication. It's used for sending and receiving data between the ESP32 and I2C-compatible sensors and devices.
* **SCL (Serial Clock)** (Commonly GPIO 22): This pin is the clock line in I2C communication, which synchronizes data transfer between the ESP32 and I2C devices.

**UART Pins (TX and RX)**:

* **TX (Transmit)** (Commonly GPIO 1): This pin is used for transmitting data from the ESP32 to an external UART device.
* **RX (Receive)** (Commonly GPIO 3): This pin is used for receiving data sent to the ESP32 from an external UART device.

**SPI Pins (MOSI, MISO, SCK, and CS)**:

* **MOSI (Master Out Slave In)** (Commonly GPIO 23): MOSI is the data line for sending data from the ESP32 to SPI devices.
* **MISO (Master In Slave Out)** (Commonly GPIO 19): MISO is the data line for receiving data from SPI devices.
* **SCK (Serial Clock)** (Commonly GPIO 18): SCK is the clock line used for synchronizing data transfer in SPI communication.
* **CS (Chip Select)** (Commonly GPIO 5): The Chip Select pin is used to select a specific SPI device when multiple devices share the same SPI bus. Note that the CS pin assignment can vary depending on the board.

**ADC Pins**:

* Common ADC pins include GPIO 32 to GPIO 39. These pins are often labeled as ADC1\_CH0 to ADC1\_CH7, corresponding to channels 0 to 7. They are used for analog-to-digital conversion and can read analog voltage levels from sensors and other analog sources.

**PWM Pins**:

* Various GPIO pins on the ESP32 support PWM (Pulse Width Modulation) output. Common PWM-capable GPIO pins include GPIO 0, GPIO 2, GPIO 4, GPIO 5, GPIO 12, GPIO 13, GPIO 14, GPIO 15, GPIO 16, GPIO 17, GPIO 18, GPIO 19, GPIO 21, and GPIO 22. These pins can generate PWM signals and are often used for controlling the brightness of LEDs, motor speed, and other applications that require variable output levels.

**Enable Pin**:

* The Enable (EN) pin is typically not a GPIO pin, and it may not be directly accessible as a GPIO. It's used to enable or disable the ESP32 module and control its power state. The exact implementation can vary between boards.

**Boot Mode Pins**:

* The boot mode pins are GPIO 0 and GPIO 2. Their configuration during startup determines the boot mode of the ESP32. GPIO 0, in particular, is commonly used for this purpose.

**Interrupt-Capable GPIO Pins** (GPIO pins that can be used for external interrupts):

**GPIO0** (Interrupt-capable):

Common Uses: Boot mode selection, general-purpose digital I/O, external interrupt input.

**GPIO2** (Interrupt-capable):

Common Uses: Bootstrapping, general-purpose digital I/O, external interrupt input.

**GPIO4** (Interrupt-capable):

Common Uses: General-purpose digital I/O, external interrupt input.

**GPIO5** (Interrupt-capable):

Common Uses: General-purpose digital I/O, external interrupt input.

**GPIO13** (Interrupt-capable):

Common Uses: General-purpose digital I/O, external interrupt input.

**GPIO14** (Interrupt-capable):

Common Uses: General-purpose digital I/O, external interrupt input.

**GPIO15** (Interrupt-capable):

Common Uses: Boot mode selection, general-purpose digital I/O, external interrupt input.

**Specifications**

* Microcontroller: Dual-core Tensilica LX6 microprocessor
* CPU Frequency: Adjustable up to 240 MHz
* Operating Voltage: 2.2V to 3.6V
* Wireless Connectivity:
  + Wi-Fi: 802.11 b/g/n (2.4 GHz)
  + Bluetooth: Bluetooth Classic (BT 4.2 BR/EDR) and Bluetooth Low Energy (BLE)
* Memory:
  + SRAM: 520KB
  + Flash: 4MB
* GPIO Pins: Varies by specific Dev Kit model but usually includes numerous GPIO pins (e.g., GPIO, PWM, UART, SPI, I2C, ADC)
* Analog-to-Digital Converter (ADC): 12-bit SAR ADC with up to 18 channels
* Interfaces: UART, SPI, I2C, I2S, PWM, CAN, SD/SDIO, Ethernet, etc.
* Power Consumption: Ultra-low power consumption in sleep modes
* Security: Hardware-based security features, including secure boot and flash encryption
* Operating Temperature: -40°C to 85°C
* Dimensions: Vary by specific Dev Kit model

**Physical Specifications (for a typical ESP32 Dev Kit):**

* Dimensions: Approximately 100mm x 60mm
* Power Supply: Typically powered via USB or an external 5V to 12V power source
* USB-to-Serial Converter: Onboard USB-to-UART bridge for programming and debugging

Programming and Development:

* Development Environment: Arduino IDE, PlatformIO, Espressif's ESP-IDF (IoT Development Framework), and more
* Programming Language: C/C++
* Programming Interfaces: Supports flashing and debugging via USB
* Firmware Updates: Over-the-air (OTA) firmware updates supported

Wireless Connectivity:

* Wi-Fi: Supports 802.11 b/g/n with WPA/WPA2, WEP, and WPS security
* Bluetooth: Supports Bluetooth Classic (BT 4.2 BR/EDR) and Bluetooth Low Energy (BLE)

**3.1.2 Ultra Sonic Sensor**

An ultrasonic sensor is a device that uses high-frequency sound waves (above the range of human hearing) to detect objects, measure distances, and perform other functions such as proximity sensing or level detection. The sensor emits a sound wave, which bounces off an object and returns to the sensor. The sensor then calculates the distance to the object based on the time it takes for the sound wave to return.



**Fig.Ultra Sonic Sensor**

**Pin Description:**

**VCC (Power Supply, +5V):** This pin is used to provide the operating voltage to the sensor, typically +5V. Connect it to the positive terminal of your power supply.

**Trig (Trigger, Input):** The Trigger pin is an input pin that initiates the ultrasonic measurement. To start a measurement, you need to send a short pulse (at least 10 microseconds) to this pin. The sensor emits an ultrasonic pulse after receiving this pulse.

**Echo (Output):** The Echo pin is an output pin that provides a pulse whose duration corresponds to the time it takes for the ultrasonic pulse to travel to the object and back. The length of this pulse is proportional to the distance between the sensor and the object.

**GND (Ground):** Connect this pin to the ground (0V) of your power supply to complete the circuit.

Please note that the HC-SR04 sensor doesn't have a dedicated-OUT pin for direct distance information. Instead, you'll need to calculate the distance based on the duration of the pulse received on the Echo pin.

**Specifications**

Operating Voltage 5V DC

Operating Current 15mA

Operating Frequency 40KHz

Min Range 2cm / 1 inch

Max Range 400cm / 13 feet

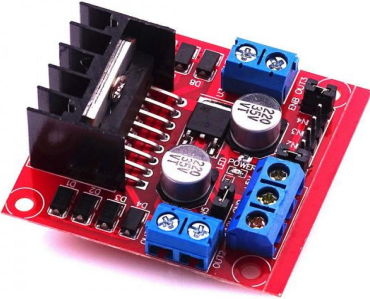
Accuracy 3mm

Measuring Angle <15°

Dimension 45 x 20 x 15mm

**3.1.3 Motor Driver IC**

In this system the motor driver is connected with analog output pin of ATmega328P microcontroller. But this microcontroller can produces maximum 5V of DC output voltage. The DC motor connected with this system requires 12V DC input voltage to operate. The motor driver IC is an integrated circuit chip that controls motors in autonomous robot and embedded circuits. L293D motor driver IC is connected between microcontroller output pin and DC motor input. This IC acts as a current amplifier. It takes low current signal from microcontroller and after amplification provides high current signal to DC motor. This high current signal is useful to drive the DC motor. L293D IC consists of 16 pins which are used to control a set of two DC motors instantly in any direction L293D consist of two inbuilt H-bridge driver circuits. It means by using a L293D IC we can control two DC motors. Input logic at pins 2 and 7 and 10 and 15 controls the operation of two motors. To stop the first motor input logic 00 is used. Logic 11 is used to stop the other motor. Logic 01 and 10 will the motor clockwise and anticlockwise directions, respectively. Pin 1 and 9 are enable pins for two connected motors. To start this motors the enable pin of this motor must be high.



**Fig. Motor Driver IC**

**Pin Description**

**Enable 1, 2 (EN1, EN2):** These pins are used to enable or disable the motor outputs on channel 1 and 2, respectively. When a logic HIGH signal is applied to these pins, the outputs are enabled, allowing the motors to operate. When a logic LOW signal is applied, the outputs are disabled, and the motors stop.

**Input 1, 2 (IN1, IN2):** These pins control the direction of motor rotation for channel 1. By applying different logic levels (HIGH or LOW) to these pins, you can make the motor rotate in either direction.

**Input 3, 4 (IN3, IN4):** These pins control the direction of motor rotation for channel 2. Similar to inputs 1 and 2, they determine the direction of the motor connected to channel 2.

**Output 1, 2 (OUT1, OUT2):** These pins are the outputs for channel 1. Depending on the input signals (IN1 and IN2), the voltage on these pins will change, driving the connected motor in the desired direction.

**Output 3, 4 (OUT3, OUT4):** These pins are the outputs for channel 2. Similar to outputs 1 and 2, they control the motor connected to channel 2.

**Vs (Supply Voltage):** This is the motor supply voltage pin. It's the positive supply voltage for the motors. The voltage should be within the specified operating range of the L298 IC.

**GND (Ground):** This is the ground pin for the motor supply voltage.

**VSS (Logic Supply Voltage):** This is the logic supply voltage pin. It provides the power supply for the internal logic circuitry of the L298. This voltage is typically around 5V.

**Specifications**

Operating Voltage 5 – 45V

Operating Current 0 – 36 mA

Peak Current 2A

Maximum Current Consume 20W

Driver or IC L298n H Bridge

Dimensions 3.4 x 4.3 x 2.7 cm

Operating Temperature -23°c to – 130°c

Thermal Resistance Heat Sink

**3.1.4 DC GEAR MOTOR**

A geared DC Motor has a gear assembly attached to the motor. The gear assembly helps in increasing the torque and reducing the speed. Using the correct combination of gears in a gear motor, its speed can be reduced to any desirable figure. This concept where gears reduce the speed of the vehicle but increase its torque is known as gear reduction. The DC motor works over a fair range of voltage. The higher the input voltage more is the RPM of the motor. In any DC motor, RPMand torque are inversely proportional.

****

**Fig. DC GEAR MOTOR**

**Pin Description**

**Positive (+) Power Supply:** This pin is where you connect the positive terminal of your 12V DC power source. It provides the voltage necessary for the motor to operate.

**Negative (-) Power Supply or Ground:** This pin is where you connect the negative terminal of your 12V DC power source. It completes the electrical circuit and provides a reference point for the motor's operation.

**Specifications**

Operating Voltage 12V

No Load Current 60mA

Load Current 300mA (Depends upon RPM)

Dimensions 75 x 32 mm (L x D)

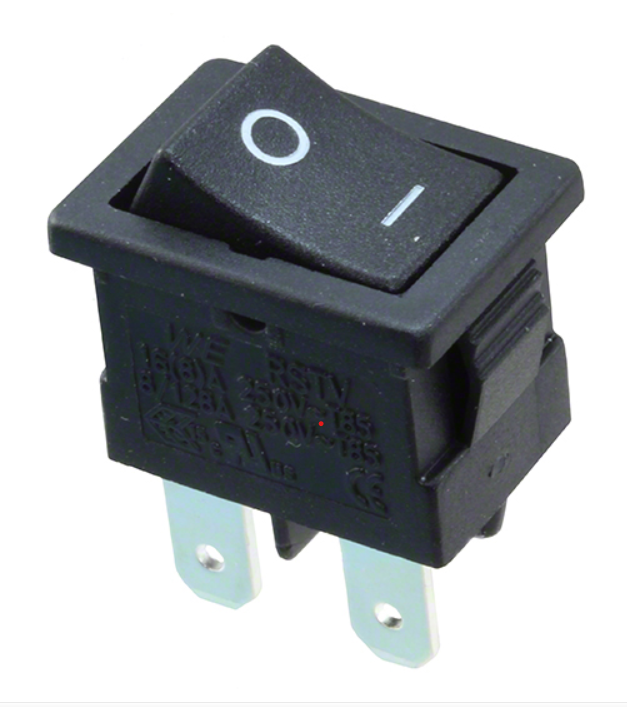
Gear Type Plastic

Gear Dimensions Gear Box Diameter 37mm

Shaft Dimensions 22 x 6 mm (L x D)

**3.1.5 ON – OFF Switch**

An on-off switch, also known as a toggle switch or simply a switch, is an electrical component used to control the flow of electric current through a circuit. It has two positions: "on" and "off". When the switch is in the "on" position, it allows current to flow through the circuit, enabling the connected device or system to operate. When the switch is in the "off" position, it interrupts the flow of current, effectively turning off the connected device or system.



**Fig. ON – OFF Switch**

**Pin Description**

**Terminal 1:** This terminal is the input or supply terminal. It's the point where the incoming current or voltage is connected.

**Terminal 2:** This terminal is the output or load terminal. When the switch is in the "on" position, it connects Terminal 1 and Terminal 2, allowing current to flow between them.

**3.1.6 ESP32-CAM**

The ESP32-CAM is a low-cost development board with a camera and Wi-Fi/Bluetooth capabilities, based on the ESP32-S chip. It's widely used for projects like remote surveillance, face recognition, smart doorbells, and IoT camera applications.



**Fig. ESP32-CAM**

**4. SYSTEM DESIGN**

**4.1 BLOCK DIAGRAM:**

Esp32

IoT

Ultrasonic Sensor

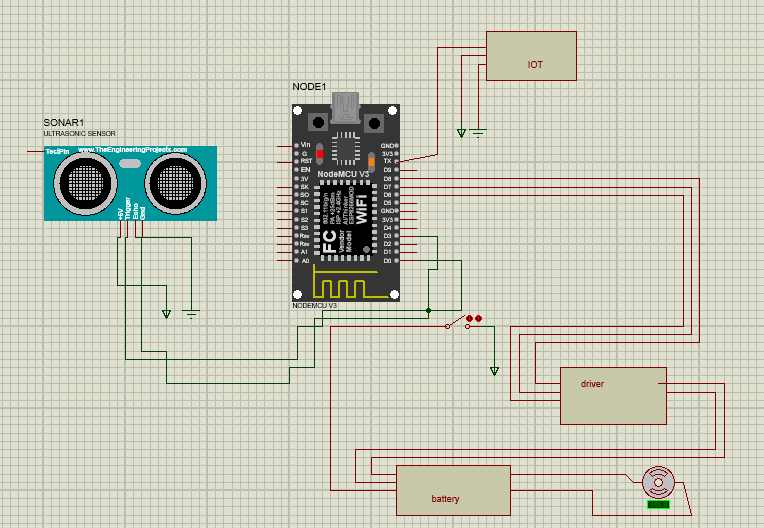
battery

L298n driver

Dc motor

Switch

**4.2 CIRCUIT DIAGRAM:**



* 1. **WORKING PRINCIPLE:**

1.Mobility and Navigation:

Chassis and Motors: The rover is built on a rugged chassis with all-terrain wheels or tracks, powered by motors.

Obstacle Avoidance: Uses ultrasonic, IR, or LiDAR sensors to detect and avoid obstacles like debris or collapsed structures.

Autonomous Navigation: Incorporates GPS and/or SLAM (Simultaneous Localization and Mapping) for autonomous movement in unknown environments.

2. Human Detection System:

Thermal Camera/IR Sensors: Detect human body heat, especially useful in low visibility or under debris.

CO₂ Sensors: Detect exhaled carbon dioxide as an indicator of human presence in enclosed spaces.

Microphones/Audio Sensors: Capture sounds like cries or movement from trapped individuals.

Computer Vision with AI: A regular or infrared camera processes images through AI models (like YOLO or OpenPose) to detect human shapes or postures.

3.Communication System:

Wireless Modules (e.g., Wi-Fi, LoRa, ZigBee): Transmit data like location, images, and sensor readings to the control center.

Real-time Video Streaming: Helps rescue teams monitor conditions remotely.

4.Power Supply:

Battery Operated: Often rechargeable lithium-ion batteries, sometimes with solar panels for extended missions.

**5. SYSTEM TESTING AND IMPLEMENTATION**

**5.1 SYSTEM TESTING:**

System testing was carried out in multiple phases to ensure each component and the overall rover system functioned correctly and efficiently in a simulated disaster environment.

**1. Unit Testing**

* **Ultrasonic Sensor:** Tested for accuracy in detecting objects within a 2–300 cm range.
* **Camera (ESP32-CAM):** Verified image clarity and response time when triggered.
* **GPS Module:** Validated real-time location accuracy using serial monitoring.
* **Motors & Motor Driver (L298N):** Checked forward and stop actions based on detection input.

**2. Integration Testing**

* All components were connected to the ESP32 and tested together.
* The rover was checked for correct responses—stopping upon detection, capturing an image, and logging location data.

**3. Functional Testing**

* Simulated obstacles (to represent humans) were placed in front of the rover.
* The rover successfully:
  + Detected the object using the ultrasonic sensor
  + Captured an image using the ESP32-CAM
  + Fetched GPS coordinates
  + Stopped momentarily and resumed movement

**4. Performance Testing**

* The system was tested in different lighting and terrain conditions:
  + Verified consistent performance indoors and outdoors.
  + Measured detection delay and movement responsiveness.
  + Tested battery usage for power efficiency.

**5.2 SYSTEM IMPLEMENTATION:**

**OVERVIEW**

The autonomous rover is designed to navigate through disaster-affected areas (earthquakes, collapsed buildings, fire zones) to detect human presence using sensors and a camera. It sends real-time data to a remote station via Wi-Fi.

**Hardware Components**

* ESP32 Microcontroller: Acts as the central processing unit for the rover. Handles sensor input, motor control, and Wi-Fi communication.
* Ultrasonic Sensor: Used for obstacle detection and to avoid collisions during movement.
* DHT11 Sensor: Monitors environmental conditions such as temperature and humidity, useful in assessing the disaster environment.
* ESP32-CAM Module: Captures real-time video/images and streams them over Wi-Fi for remote monitoring. Also used for human detection using AI/ML models.
* Motor Drivers (L298N): Interface between ESP32 and the DC motors to control movement.
* DC Motors with Wheels: Provide mobility to the rover in rugged terrain.
* Wi-Fi Module (built-in ESP32): Enables real-time communication with a base station or cloud server.
* Battery Pack: Powers the rover.

**Software Implementation**

* Embedded C/C++ (Arduino IDE): Used for programming ESP32.
* Object Detection Algorithm: TensorFlow Lite model deployed on ESP32-CAM for basic human detection.
* Wi-Fi Streaming: Real-time video stream is sent to a mobile app or PC interface.
* Obstacle Avoidance Logic: Ultrasonic sensor data processed to control movement and avoid obstacles.

**Functionality Workflow**

* Initialization: ESP32 initializes all sensors and motors.
* Navigation: Ultrasonic sensors guide the rover through safe paths.
* Surveillance: ESP32-CAM streams video and captures potential human presence.
* 4. Detection: If a human is detected, GPS (optional) can mark location and send alert.
* Monitoring: Real-time video and sensor data are displayed on the remote interface.

**Use Case Scenario**

* A building has collapsed due to an earthquake.
* The rover is deployed into narrow spaces.
* It moves autonomously, avoids debris using sensors, and streams video.
* Upon detecting a human figure, the system sends an alert and location to the rescue team.

**6.CONCLUSION**

In conclusion, the proposed Human Detection Robot using ESP32, ultrasonic sensors, L298N motor driver, and IoT integration presents a practical solution for human detection and automation in various applications. The integration of IoT allows for real-time monitoring and remote control, providing enhanced functionality and flexibility. By leveraging the ESP32’s Wi-Fi capabilities and energy-efficient design, the system ensures reliable communication and power optimization. Additionally, the use of ultrasonic sensors for detecting humans within a specific range ensures accurate and effective detection. The proposed system lays the foundation for further advancements in autonomous robots and offers scalability for future enhancements, such as the inclusion of additional sensors or advanced algorithms. Ultimately, this project offers significant potential in fields like security, healthcare, and automation, making it a valuable tool for real-world applications.

**7. SCOPE FOR FUTURE ENHANCEMENT**

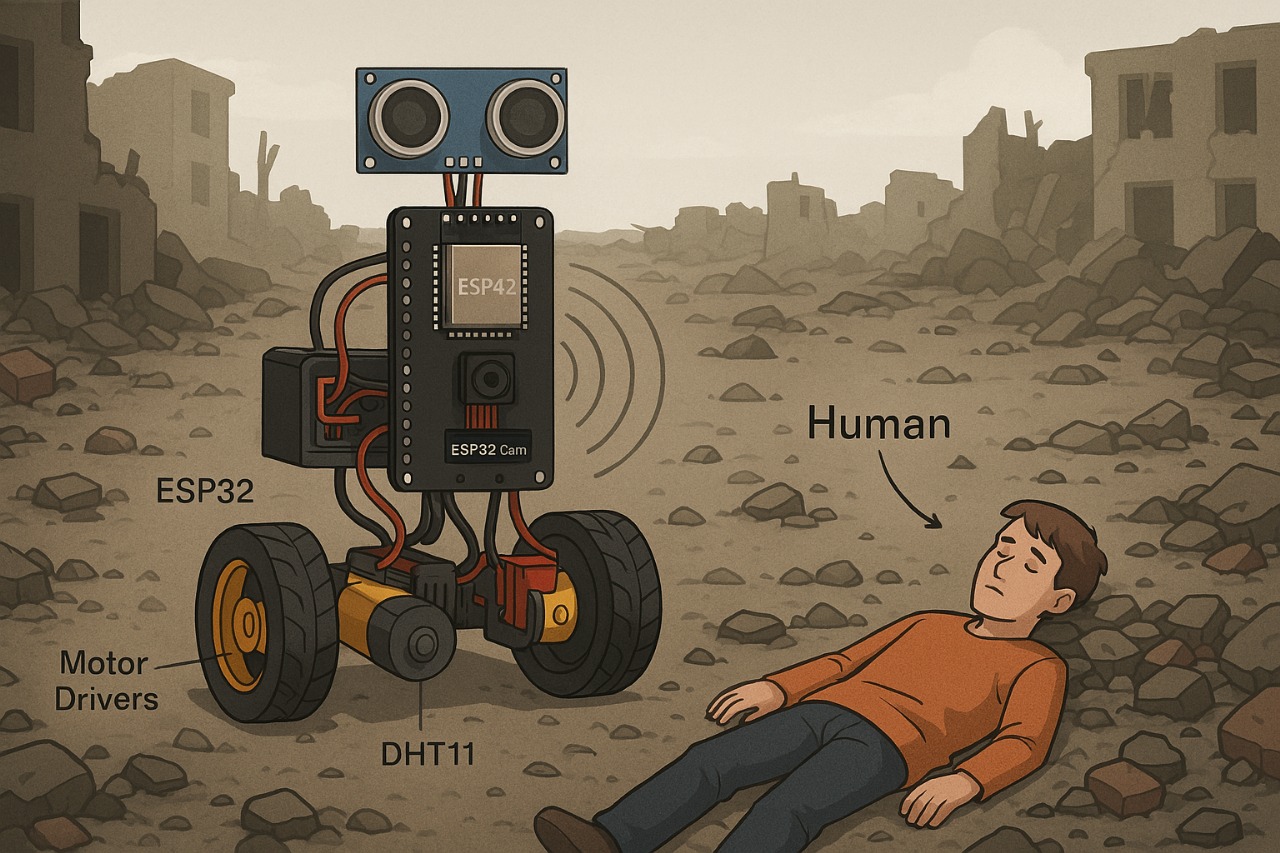
The proposed human detection rover has significant potential for future enhancement to improve its effectiveness in disaster scenarios. One major upgrade is the integration of a thermal imaging camera, allowing the detection of human body heat in low-visibility environments like smoke or debris-covered areas. Additionally, audio sensors and voice recognition could be used to identify human sounds such as shouting or tapping, further aiding in rescue operations. Incorporating AI and machine learning can enhance the system’s ability to distinguish between humans and other objects, enabling smarter obstacle avoidance and path planning. The addition of GPS and real-time location tracking would help responders pinpoint exact locations of detected individuals. Future versions could also support swarm robotics, where multiple rovers communicate and collaborate to cover large areas more efficiently. Enhancing the rover’s terrain adaptability with stronger wheels or tracks would allow it to operate in more rugged environments. Implementing solar charging or self-charging mechanisms would extend its operational time in the field. To ensure uninterrupted performance in remote zones, offline data storage and later syncing could be enabled. Lastly, integration with aerial drones could provide an overhead view to complement the rover’s ground-level detection, making the system more robust and comprehensive for search and rescue missions.

**BIBLIOGRAPHY:**

1. Patel, M., & Shah, D. (2018). "Obstacle Detection and Avoidance Using Ultrasonic Sensors in Mobile Robots". *International Journal of Advanced Research in Electrical, Electronics, and Instrumentation Engineering*, 7(8), 3035-3041.
2. Bicheno, S., & Milne, A. (2020). "IoT-Enabled Mobile Robots: A Survey of Design and Implementation". *Journal of Robotics and Autonomous Systems*, 130, 103601. https://doi.org/10.1016/j.robot.2020.103601
3. Chaudhary, S., Kumar, R., & Arora, A. (2021). "ESP32-Based IoT Platform for Autonomous Mobile Robots". *Proceedings of the International Conference on IoT and Robotics*, 245-253. IEEE. https://doi.org/10.1109/ICIRT.2021.9235698
4. Sahu, S., & Kumar, P. (2019). "Design and Control of Mobile Robot Using L298N Motor Driver". *International Journal of Engineering Research and Technology*, 8(5), 1-5.
5. Wang, Y., & Li, H. (2020). "Human Detection and Tracking for Security Robots Using Ultrasonic and Infrared Sensors". *IEEE Access*, 8, 132902-132912. https://doi.org/10.1109/ACCESS.2020.3009395
6. Sasikala, S., & Kannan, S. (2020). "IoT-Based Human Detection Robot for Elderly Care in Healthcare". *International Journal of Advanced Science and Technology*, 29(8), 155-164.
7. Yadav, S., & Verma, R. (2021). "Remote Monitoring and Control of IoT-Based Robots Using ThingSpeak". *International Journal of Computer Applications*, 176(7), 10-14. https://doi.org/10.5120/ijca2021921682
8. Espressif Systems. (2022). "ESP32 Technical Reference Manual". Espressif Systems. Retrieved from https://www.espressif.com/en/products/socs/esp32
9. ThingSpeak. (2023). "ThingSpeak IoT Platform for Data Analytics and Cloud Applications". Retrieved from <https://www.thingspeak.com/>
10. R. K. Gupta, A. P. Pandey. (2019). "Robotic Systems: A Review of Human Detection Technologies". *International Journal of Robotics and Mechatronics Engineering*, 5(3), 34-42.

**APPENDIX:**

**A.SCREENSHOT**



**B.SAMPLE CODING**

#include <HardwareSerial.h>

#include <TinyGPS++.h>

#include <WiFi.h>

#include "esp\_camera.h"

// Pin Definitions (adjust based on your ESP32 model)

#define TRIG\_PIN 12

#define ECHO\_PIN 13

#define ENA 14

#define IN1 27

#define IN2 26

#define ENB 25

#define IN3 33

#define IN4 32

// GPS

static const int RXPin = 16, TXPin = 17;

TinyGPSPlus gps;

HardwareSerial GPSSerial(1);

// Camera settings (for ESP32-CAM)

#define PWDN\_GPIO\_NUM -1

#define RESET\_GPIO\_NUM -1

#define XCLK\_GPIO\_NUM 0

#define SIOD\_GPIO\_NUM 26

#define SIOC\_GPIO\_NUM 27

#define Y9\_GPIO\_NUM 35

#define Y8\_GPIO\_NUM 34

#define Y7\_GPIO\_NUM 39

#define Y6\_GPIO\_NUM 36

#define Y5\_GPIO\_NUM 21

#define Y4\_GPIO\_NUM 19

#define Y3\_GPIO\_NUM 18

#define Y2\_GPIO\_NUM 5

#define VSYNC\_GPIO\_NUM 25

#define HREF\_GPIO\_NUM 23

#define PCLK\_GPIO\_NUM 22

long duration;

int distance;

void setup() {

Serial.begin(115200);

// Motor setup

pinMode(ENA, OUTPUT);

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

pinMode(ENB, OUTPUT);

pinMode(IN3, OUTPUT);

pinMode(IN4, OUTPUT);

// Ultrasonic setup

pinMode(TRIG\_PIN, OUTPUT);

pinMode(ECHO\_PIN, INPUT);

// GPS setup

GPSSerial.begin(9600, SERIAL\_8N1, RXPin, TXPin);

// Camera setup

camera\_config\_t config;

config.ledc\_channel = LEDC\_CHANNEL\_0;

config.ledc\_timer = LEDC\_TIMER\_0;

config.pin\_d0 = Y2\_GPIO\_NUM;

config.pin\_d1 = Y3\_GPIO\_NUM;

config.pin\_d2 = Y4\_GPIO\_NUM;

config.pin\_d3 = Y5\_GPIO\_NUM;

config.pin\_d4 = Y6\_GPIO\_NUM;

config.pin\_d5 = Y7\_GPIO\_NUM;

config.pin\_d6 = Y8\_GPIO\_NUM;

config.pin\_d7 = Y9\_GPIO\_NUM;

config.pin\_xclk = XCLK\_GPIO\_NUM;

config.pin\_pclk = PCLK\_GPIO\_NUM;

config.pin\_vsync = VSYNC\_GPIO\_NUM;

config.pin\_href = HREF\_GPIO\_NUM;

config.pin\_sscb\_sda = SIOD\_GPIO\_NUM;

config.pin\_sscb\_scl = SIOC\_GPIO\_NUM;

config.pin\_pwdn = PWDN\_GPIO\_NUM;

config.pin\_reset = RESET\_GPIO\_NUM;

config.xclk\_freq\_hz = 20000000;

config.pixel\_format = PIXFORMAT\_JPEG;

config.frame\_size = FRAMESIZE\_QVGA;

config.jpeg\_quality = 10;

config.fb\_count = 1;

if (!esp\_camera\_init(&config)) {

Serial.println("Camera initialized.");

} else {

Serial.println("Camera failed to initialize!");

}

moveForward();

}

void loop() {

// Measure distance

digitalWrite(TRIG\_PIN, LOW);

delayMicroseconds(2);

digitalWrite(TRIG\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG\_PIN, LOW);

duration = pulseIn(ECHO\_PIN, HIGH);

distance = duration \* 0.034 / 2;

Serial.print("Distance: ");

Serial.println(distance);

if (distance < 30) {

stopMotors();

Serial.println("Human/Obstacle Detected!");

captureImage();

getGPSLocation();

delay(5000);

moveForward();

}

delay(500);

}

void moveForward() {

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

analogWrite(ENA, 200);

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

analogWrite(ENB, 200);

}

void stopMotors() {

digitalWrite(IN1, LOW);

digitalWrite(IN2, LOW);

analogWrite(ENA, 0);

digitalWrite(IN3, LOW);

digitalWrite(IN4, LOW);

analogWrite(ENB, 0);

}

// Capture and save image

void captureImage() {

camera\_fb\_t \*fb = esp\_camera\_fb\_get();

if (!fb) {

Serial.println("Camera capture failed");

return;

}

Serial.println("Image Captured (Simulated)");

esp\_camera\_fb\_return(fb);

}

// Fetch GPS coordinates

void getGPSLocation() {

while (GPSSerial.available() > 0) {

gps.encode(GPSSerial.read());

if (gps.location.isUpdated()) {

Serial.print("Latitude: ");

Serial.println(gps.location.lat(), 6);

Serial.print("Longitude: ");

Serial.println(gps.location.lng(), 6);

}

}

}