

IOT BASED UNMANNED ARMED GROUND VEHICLE



Group Members

MUHAMMAD ZAKRIYA ZAHID	(190487)
KHIZAR HAYAT	(190503)
MUHAMMAD YASEEN	(190507)

**Bachelor of Electrical Engineering
(2019-BEEP-2023)**

PROJECT SUPERVISOR

MAM SEEMA TAHIR

ASSISTANT PROFESSOR

DEPARTMENT OF ELECTRICAL ENGINEERING

FACULTY OF ENGINEERING

AIR UNIVERSITY, ISLAMABAD

A Final Year Project Report
presented to
AIR University
in partial fulfillment of the requirements
for the Degree of
Bachelor of Electrical Engineering

IOT BASED UNMANNED ARMED GROUND VEHICLE

SUBMITTED BY

MUHAMMAD ZAKRIYA ZAHID	(190487)
KHIZAR HAYAT	(190503)
MUHAMMAD YASEEN	(190507)

(19-BEEP-23)

Department of Electrical Engineering



Approval for Submission

It is to certify that the project report titled
"IOT BASED UNMANNED ARMED GROUND VEHICLE"
has met the required standard of submission
in partial fulfillment of requirements
for the award of degree of
Bachelor of Electrical Engineering
at Air University, Islamabad.

Project Supervisor

Mam Seema Tahir

Assistant Professor

Head of Department

Dr Hafiz Ashiq Hussain

Acknowledgments

The delight of completing the assignment successfully would be incomplete without acknowledging all individuals whose counsel and encouragement helped to make the effort a success.

We gratefully thank "**AIR UNIVERSITY ISLAMABAD**" for providing us the opportunity to pursue our degree program, which is B.E in **ELECTRICAL ENGINEERING (POWER)**.

We will always be grateful to our Honorable Chairman **Air Marshal Javaid Ahmed** and **Registrar Air CDRE @ Abdul Wahab Motla**.

We would like to convey our sincere gratitude to our respected **HOD Dr.HAFIZ ASHIQ HUSSAIN** for providing us with such a wonderful learning environment and for mentoring us in becoming excellent citizens.

We truly appreciate the inspiration, prompt assistance, and direction provided to us by our supervisor **SEEMA TAHIR, ASSISTANT PROFESSOR** to finish our project.

We would like to thank **Mr. Mumajjed UL Mudassir**, the coordinator for our final year projects, for his encouragement and prompt assistance and direction given to us.

We also thank our department's instructors and respected faculty **Dr.Fida Muhammad Khan, Air CDRE @ Tahir Mahmood Khalid, Mam Amina Asif, Mam Madeeha Uzma, Engineer Faizan Munir, Engineer Akram Rashid, Dr.Ashfaq Ahmad, Dr.Waseem Khan** , who have always been patient in answering our questions.

Finally, we would like to express our gratitude to our parents and friends for serving for both motivation and inspiration for us while we pursued and completed this project

Abstract

Surveillance/Security and Terrorism is one of the the leading problems which is suffering almost all the countries in the world. Every country is looking forward to strengthen its defence system by including latest and advanced technologies in their defence /surveillance department. Many underdevelopment countries including Pakistan and India are spending billion of dollars on advancing their defence to tackle any security situation. These days where every field is based on automation and robotics, many armed vehicle is also automated to execute many dangerous operations on war fields . These vehicles are not even efficient but also there is almost zero risk for human life in war field. The project we are working on is **“IOT based unmanned armed ground vehicle”** which is designed for surveillance with dedication to serve humanity. Vehicle will be Remotely controlled by wireless means. We will be able to see live streaming of the area and a weapon will be attached to target any thing in the range. Vehicle will consist of two main parts one is control unit and other is body of armed vehicle with a controlled weapon. Remote controller will get live video of the camera attached on vehicle. Web page will send the information to Esp-32 which will control the micro-controller for controlling the stepper motors, servo motors, dc motors wireless data reception, and transmission.

Contents

1	Introduction	1
1.1	Internet of things	1
1.2	ADVANTAGES	1
1.2.1	Access to low-cost, low-power sensor technology	2
1.2.2	Connectivity	2
1.2.3	Cloud computing platforms	2
1.2.4	Machine learning and analytics	3
1.2.5	Conversational artificial intelligence (AI)	3
1.3	IOT APPLICATIONS	4
1.3.1	INDUSTRIAL	4
1.3.2	MILITARY	5
1.3.3	Commercial	6
1.4	UAGV	6
1.4.1	Tele-operated	6
1.4.2	Security	7
1.4.3	Autonomous	8
1.5	Problem Statement	9
1.6	Objectives of the project	10

2 Literature review	11
3 Methodology	13
3.1 Components	13
3.1.1 DC Gear Motors	13
3.1.2 Stepper Motor	14
3.1.3 Servo motor	15
3.1.4 L298N Motor driver	16
3.1.5 Battery and voltage regulator	17
3.1.6 PIC18F452	18
3.1.7 ESP-32 Cam	18
3.1.8 Arduino	21
3.1.9 A4988 Motor Driver	23
3.2 Design Procedure	24
3.2.1 Control Unit / First Portion	24
3.2.2 Vehicle Body / Second Portion	25
4 Results and Discussion	27
4.1 Simulation Results	27
4.1.1 Explanation	27
4.1.2 Specifications and Calculations	28
4.2 Hardware	31
4.2.1 Components Explanation	31
4.2.2 External Hardware	32
4.2.3 User Interface	33
5 Conclusion	34

6	Impact On Environment and SDG's	35
6.1	Impact on Environment	35
6.1.1	Low Cost	35
6.1.2	Life Saving	35
6.1.3	Easy to Operate	35
6.1.4	Pollution Free	35
6.2	Sustainable Development Goals	36

List of Figures

1.1	Internet of things	1
1.2	IoT Operated things	2
1.3	Cloud Connectable things	2
1.4	IoT Platform	3
1.5	Data Processing steps in cloud	3
1.6	IoT Data Processing steps	4
1.7	IoT used in Industry	5
1.8	IoT Usage in Military	5
1.9	Tele-operated UGV	7
1.10	Autonomous UGV	8
3.1	Autonomous UGV	14
3.2	Stepper Motor	15
3.3	Stepper Motor Simulation	15
3.4	Servo Motor with arms	16
3.5	Servo Motor with Arduino	16
3.6	L298N motor driver	17
3.7	Battery	17
3.8	Pic18F452 Pin Configuration	18

3.9	ESP-32 cam module	19
3.10	Specifications of ESP 32 cam	19
3.11	ESP-32 cam Pin Configuration	20
3.12	Arduino Pin outs	23
3.13	A4988 Driver Module	23
3.14	Control Unit + Vehicle Body Flow diagram	24
3.15	Android Processing steps	24
3.16	UGV Controllers Processing steps	25
4.1	Simulation of project	27
4.2	Body Parts Movement Combinations	28
4.3	Hardware labelled parts	31
4.4	External Look of hardware	32
4.5	User Interface	33

CHAPTER 1

Introduction

1.1 Internet of things

The Internet of things (IoT) describes physical objects (or groups of such objects) with sensors, processing ability, software, and other technologies that connect and exchange data with other devices and systems over the Internet or other communications network



Figure 1.1: Internet of things

1.2 ADVANTAGES

Even while the idea of IoT has been around for a while, it has only just come to fruition as a result of numerous recent technological developments.

1.2.1 Access to low-cost, low-power sensor technology

IoT technology may be used by more firms due to the sensors' low cost and accuracy.

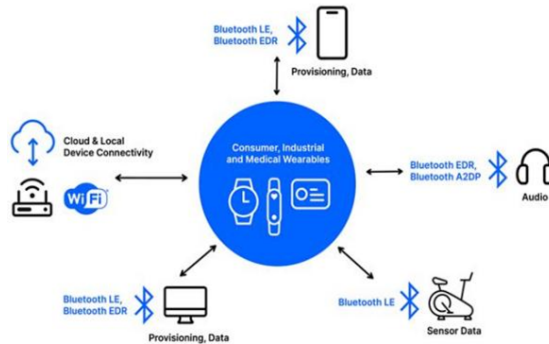


Figure 1.2: IoT Operated things

1.2.2 Connectivity

Due to the vast variety of communication protocols available for the internet, it is now simple to connect sensors to the cloud and other "things" for efficient data transfer.



Figure 1.3: Cloud Connectable things

1.2.3 Cloud computing platforms

Thanks to the emergence of cloud platforms, businesses and consumers may now access the infrastructure they need to scale up without having to manage it all.

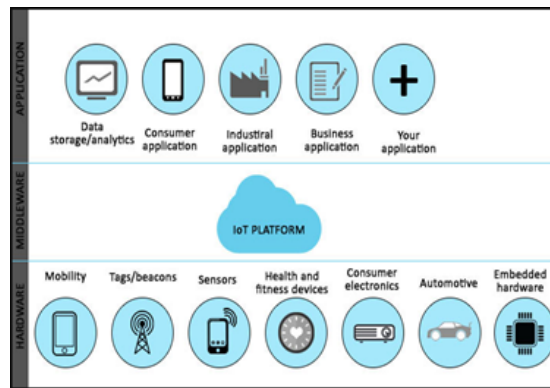


Figure 1.4: IoT Platform

1.2.4 Machine learning and analytics

Because of advancements in machine learning and analytics, as well as access to diverse and enormous amounts of data stored on the cloud, organizations may gain insights more rapidly and conveniently. The potential of IoT are increased by these technical advancements, and these alternative technologies are supported by the data that IoT generates.

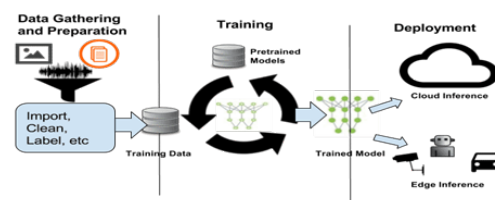


Figure 1.5: Data Processing steps in cloud

1.2.5 Conversational artificial intelligence (AI)

Internet of Things (IoT) devices increasingly support natural language processing (NLP), notably digital personal assistants like Alexa, Cortana, and Siri. IoT gadgets are becoming more enticing, useful, and affordable to use at home as a result.



Figure 1.6: IoT Data Processing steps

1.3 IOT APPLICATIONS

One of the most important 21st-century technologies has recently arisen, and that is the Internet of Things. Thanks to the capacity to link everyday objects to the internet via embedded devices, such as home appliances, cars, thermostats, and baby monitors, continuous communication between people, processes, and things is now feasible. Low-cost computers, the cloud, big data, analytics, and mobile technologies allow physical things to share and gather data with the least amount of human involvement. In today's networked environment, digital technologies can record, monitor, and alter every contact between connected entities. The physical and digital worlds communicate with one another and work together. Here, some of the major applications of IoT are covered.

1.3.1 INDUSTRIAL

Industrial IoT devices gather and analyse data from linked equipment, locations, people, and operational technology (OT). When used in conjunction with OT monitoring systems, IoT becomes more effective in controlling and monitoring industrial systems. Additionally, the same technology may be utilised for automatic record updates of asset placement in commercial storage units because the size of the assets may range from a little screw to a whole motor replacement component. Such assets can be lost, costing time, money, and human resources. Industry 4.0, the fourth wave of the industrial revolution, is another name for IoT. The following are some common uses for IoT:

- Connected assets, smart manufacturing



Figure 1.7: IoT used in Industry

- Preventive and predictive maintenance
- Smart digital supply chains
- Connected logistics
- Smart cities and smart power stations.

1.3.2 MILITARY

The Internet of Military Things (IoMT), which is the use of IoT technology in the military, is focused on monitoring, intelligence, and other combat-related tasks. It is significantly impacted by the likelihood of deploying sensors, weapons, vehicles, robots, wearable biometrics for people, and other intelligent technology that is beneficial in war.

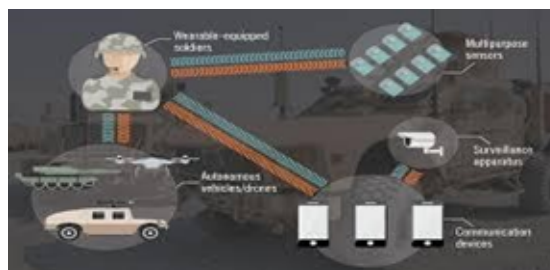


Figure 1.8: IoT Usage in Military

1.3.3 Commercial

Commercial IoT refers to a network of objects that communicate with one another using the Internet protocol entirely autonomously. Thanks to embedded software, sensors, actuators, electronics, network connectivity, and other "Things," the "Things" can collect data and share it with other Things. Commercial IoT refers to the systems and devices utilised by companies and corporations. Industry-specific variations might be significant. IoT is being used more and more in several industries, including health-care, recreation areas, workplaces, and others. Consumer IoT refers to B2C products or systems such mobile phones, gadgets, and personal healthcare alert systems.

1.4 UAGV

Unmanned armed ground vehicles (UAGVs) are military robots that soldiers may operate via the internet of things. This vehicle has the ability to conduct ground operations in a hostile environment without any actual human presence. A soldier can operate the vehicle according to the scenario and shoot the enemy with a control pistol that is attached to the top of the vehicle's body while viewing a live video of the battle area. Generally speaking, if we discuss unmanned robotics. For many tasks, both civilian and military use is quite valuable.

There are two general classes of unmanned ground vehicles:

1. Tele-operated
2. Autonomous

1.4.1 Tele-operated

A teleoperated UGV which is a vehicle that is managed by an operator over a communications link from a distance. Based on sensory data from either line-of-sight visual observation or remote sensory input from devices like cameras, the operator performs all cognitive functions.

An easy-to-understand illustration of the fundamentals of teleoperation would be a remote-control toy automobile. All of the unmanned vehicles is operated remotely by the user based solely on the observed performance of the vehicle, whether that connection is wired or wireless. There are several different types of teleoperated UGVs in use right now. These vehicles are typically utilized in dangerous situations to replace people. Explosives and vehicles that disable bombs are two examples.

1.4.2 Security

Cryptography and steganography have different levels of security. Cryptography can provide strong security if implemented correctly, and the key is kept secret. However, if the key is compromised or the algorithm is broken, the message's confidentiality can be compromised. In contrast, steganography can provide a high level of security if the message is hidden in a seemingly innocuous file or medium. However, if the file is analyzed or the attacker has prior knowledge of the steganographic technique used, the message can be detected and compromised.

While cryptography and steganography have different objectives and techniques, they can be used together to provide a high level of security. Cryptography can be used to encrypt the contents of a message, while steganography can be used to conceal the message's existence. It is important to note that while cryptography focuses on making the contents of a message unreadable, steganography focuses on making the message undetectable. Therefore, the choice between these two techniques depends on the specific needs and objectives of the communication.

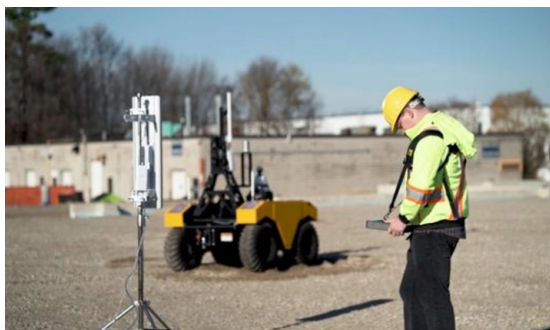


Figure 1.9: Tele-operated UGV

1.4.3 Autonomous

In essence, an autonomous UGV is a robot that drives itself, although it is more particularly an automobile that moves over the ground. In the actual world, a fully autonomous robot is capable of learning about its surrounding.



Figure 1.10: Autonomous UGV

- Perform for protracted periods of time without human interaction.
- Transport from point A to point B without the use of a human navigator.
- Avoid situations that could be harmful to people, property, or the structure itself unless they are specifically allowed by the design standards.
- It can Repair itself without the help from outside.
- It can Spot suspicious objects, such as people's and cars.
- Learn or acquire new skills independently
- Autonomous learning includes the capacity to adapt to your surroundings on your own.

Like other machinery, autonomous robots still need routine maintenance.

1.5 Problem Statement

Due to various economic elements, manpower is being replaced by machines in the modern age. Conflicts now, in the fifth generation of warfare, are wasting valuable lives. Therefore, using armed robots in battles to accomplish various objectives without causing any casualties has proven to be quite beneficial. The objective of this project is to create a ground vehicle that can be manually handled by a person from a distance in order to aim from a weapon at targets.

1.6 Objectives of the project

Following are the objectives of our project:

Command & Control

To control the UGV from some distance by giving commands through wireless means and to move in all directions.

Remote Access / IOT

We will have wireless control over the vehicle through internet using IOT.

Live Video Streaming

Our Esp-32 module will provide us a live video streaming of far objects to target through gun.

CHAPTER 2

Literature review

Due to the rise in terrorist attacks, **Dumbre et al.** used this vehicle to improve security. The vehicle could be operated over a long distance between the user and the vehicle as it could be controlled remotely over the internet. A webcam was linked with Raspberry Pi which captured several numbers number of photos, which were ultimately transmitted over the internet to the server-side web browser. The user can either give a command from a keyboard or from a webpage, and the robot would begin to move according to the command.

A system that will provide visual feedback to aid in both sectors, from the defense system for surveillance purposes in sensitive locations to any home surveillance, was proposed by **A. Ashraya and Munaswamy**. The Raspberry Pi3 CPU was utilized to operate the robot. This robot was capable of high-quality transmission, high-temperature sensing, and fire suppression via sprinklers.

The robot's high-definition video was its main focus. **Tarek A. Tutunji, Mohammed Salah-Eddin, and Hisham Abdalqader** created a prototype for IOT-based surveillance in uncharted territory. The vehicle had an IP address which was used to connect to the operator via the cloud. It could be operated manually using an Android-app to-command speed and direction or autonomously using control algorithms to avoid obstacles. Raspberry offers cloud-based video streaming, and Arduino regulates motor speed.

Ashitha V. Naik, Balram Rayappa Kage, Vikrant Krishna, Tarun Pal, and Tanmay Raj presented a war field spying robot that could be manually operated and can provide

location information to a server for surveillance purposes. It had an IR sensor and a bomb detector and was driven by a PIC microcontroller and a NodeMCU. A buzzer fitted to detect underground mines. A mobile android application was implemented to control the robot system proposed by **Mallikarjun Mudda, Jahangir Badhasha, D. Vamshi Krishna, B. V. Ramana Reddy, and R. Sampath Mahendra**. The wifi module at the receiving station receives video from the ESP-32 camera and broadcasts it. Video can be captured for later use. For better outcomes, the robot has a smoke sensor and night vision. Defense and rescue activities were its primary uses.

S. MITHILEYSH performed an experiment named as unmanned ground vehicle. His vehicle was working in manual and autonomous both modes of control. A camera was installed on the vehicle for live streaming of the location. A gun was also attached with the vehicle to target enemy with the of video. He successfully uses the results, such as motion tracking, obstacle detection, path planning , gesture control and GPS. He used 3G services to provide easy accessibility to his UGV.

Unmanned ground vehicles were created by Kamran Shahani, Hong Song, Chaopeng Wu, Syed Raza Mehdi, Kazim Raza, and Noorudin Khaskheli for use in search and rescue operations. Without any human operator, this car was operating all by itself. With visual feedback, vehicles can be utilised for both surveillance and rescue operations. The four-wheeled vehicle can move within an 8-meter radius and uses Bluetooth as a communication method. The vehicle was equipped with an aluminium arm that could pick up anything. The military search and surveillance positions benefited from this effort.

By adopting an easy-to-use graphic user interface for UGV navigation, the created system—named UGV and remote control station proposed by Jun Pyo Lee was simple to use and enabled a large reduction in station operators' burden. A single station operator could control several UGVs simultaneously due to the system. Numerous experiments were conducted in a real-world setting to confirm the efficiency of the proposed system.

CHAPTER 3

Methodology

We are going to develop an Unmanned armed Ground Vehicle which is being operated by the teleported method. So we explain the method components as under.

3.1 Components

3.1.1 DC Gear Motors

The dc gear motor is typically assembled and with the DC motor, which combines a gearbox and a motor. A built-in assembly of gears within the frame distinguishes a dc gear motor's construction from a typical motor. The gear ratio is selected for the appropriate speed reduction and the gear assembly is provided as an extended assembly.”

“Household appliances, blowers, pumps, fans, machine tools, power tools, and disc drives all employ electric motors. They can be started or operated using either the battery or electricity that is supplied by the grid. Brushless Dc motors are very efficient, have a long lifespan, and require less maintenance. However, they do have one drawback, namely a high starting cost and more challenging speed controllers.

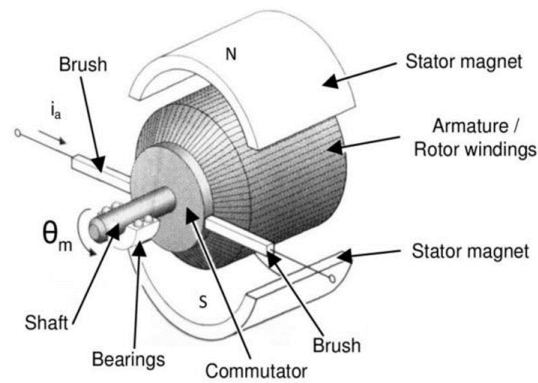


Figure 3.1: Autonomous UGV

3.1.2 Stepper Motor

A DC motor that moves in distinct steps is called a stepper motor. It contains several coils arranged in what are referred to as "phases". The motor will revolve by sequentially energising each phase, one at a time. We can regulate our stride and obtain accurate location or pace using a computer or an Android. We chose it for the arm actuator action because it is exact in the moment. It can rotate the gun actuator to right, left, up and down precisely. If we apply DC voltage to brushed DC motors terminals which can rotate continuously. Basically the stepper motor is used for precise control and work. Where efficiency is required in work we use stepper motor because it can do the task more precisely due to its torque control and speed. The shaft rotates through a set angle with each pulse. Several "toothed" electromagnets organised as a stator and rotating around an iron rotor make up stepper motors in reality.

The electromagnets are powered by a microcontroller or an external driving circuit. To magnetically draw the gear teeth and rotate the motor shaft, one electromagnet is operated. When the gear teeth are lined up with the first electromagnet, they are somewhat spaced apart from the subsequent magnets. In other words, when the first electromagnet is shut off and the second one is put on, the gear slightly turns to align with the second electromagnet. The procedure is then repeated. "Paraphrase of each number of steps of each number of steps of As a result the motor can turn itself to a precise angle.

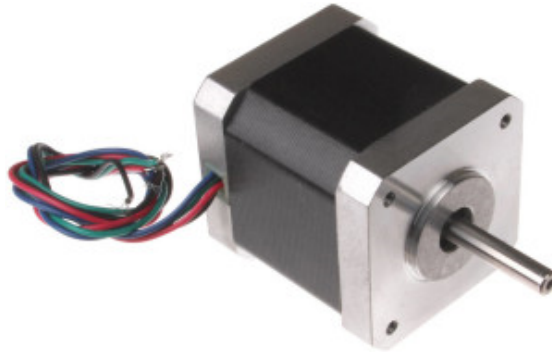


Figure 3.2: Stepper Motor

Simulation of Stepper Motor

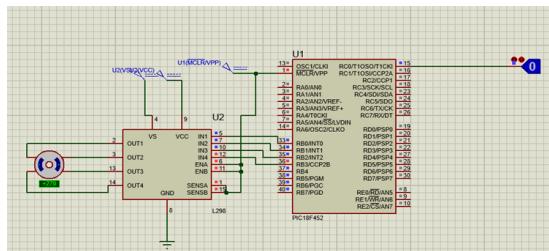


Figure 3.3: Stepper Motor Simulation

3.1.3 Servo motor

It is an electrical device that works in pushing and rotating objects. It works best to precisely controls the objects. In our project we use it for Gun triggering. We have done the coding in the Micro controller which control the servo motor according to our requirement.



Figure 3.4: Servo Motor with arms

Principle

Servo motor is working on the principle of servo mechanism. As we know it's not a special type of motor because we can be found it in robots like baby toy cars, helicopters, and other childhood toys. It is also used in industry in manufacturing plants. It is highly preferable because it can produce torque. It is having Electric motor, Reduction gear, Position feedback potentiometer, Actuator arm.

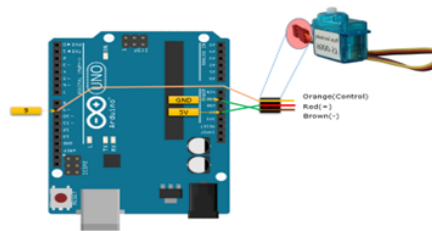


Figure 3.5: Servo Motor with Arduino

3.1.4 L298N Motor driver

Two DC motors and one stepper motor can be driven using the dual-channel H-Bridge motor driver L298N. It implies that it has the ability to independently drive up to two DC motors for any application, such as DC locks, solenoids, 2WD robots, small drill machines, and solenoids.

An L298N motor driver module (IC) is made up of an L298N motor driver chip. This is an integrated monolithic circuit that comes in a 15-lead Multiwatt package. It is a twin full-bridge driver that can support standard TTL logic levels and has high voltage and current.

This driver we use in project, it protect the motor and controller from damage because if we directly link dc motors they can draw power in variation so it might effect the controller.

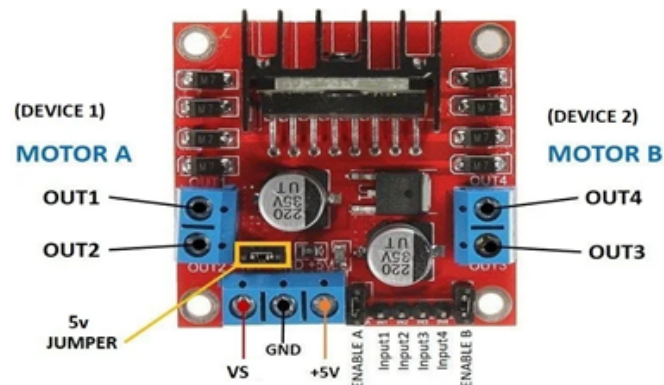


Figure 3.6: L298N motor driver

3.1.5 Battery and voltage regulator

Lithium-ion polymer batteries are rechargeable batteries. Typically batteries are made out of a few indistinguishable auxiliary cells in equal expansion to build the release current capability. These batteries have very low resistance, and charge fast as compared to led acid. The almost 99% efficiency of these batteries reduces charge-related losses. When compared to conventional lead-acid batteries, which have an efficiency of 75–80 percent, lithium batteries can last up to ten times as long. It offers the purest clean energy. No pollution, no gas and no fumes. It protect the environment from pollution so its basically beneficial for users.



Figure 3.7: Battery

3.1.6 PIC18F452

An enhanced flash microcontroller with great performance, the PIC18F452-I/P has 8 channels of 10-bit analog-to-digital (A/D) converters.

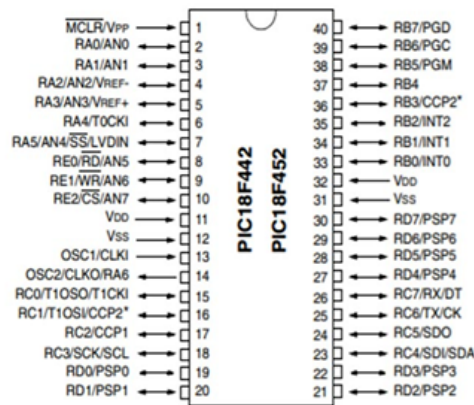


Figure 3.8: Pic18F452 Pin Configuration

An 8-bit, 10-MIPS, CMPS, and FLASH-based microcontroller called the PIC18F452 contains 34 I/O pins out of 40 Pin packages. It is a strong microcontroller having one 8-bit timer, three 16-bit timers, eight 10-bit analog-to-digital converter channels, and peripherals for I2C, SPI, and USART. It is a low power microcontroller device that operates at 5 volts and 4 MHz while using less than 0.2 uA of standby current and 1.6 mA of regular current. Pic18F452 is used for motor control. Esp 32 output pins are input pins for pic18F452 which are actually combination that can rotate or move the motors as per given sequence

3.1.7 ESP-32 Cam

The ESP32-CAM is a fairly priced ESP CHIP development board that includes a camera and Bluetooth. It aims to link devices together over a reasonably short distance. Two powerful 32-bit LX6 CPU's (BLE was developed with IoT applications in mind, which has significant implications for its design) are installed on the board.

With a footprint of just 27*40.5*4.5mm and a deep sleep current of up to 6mA that may operate independently as a minimal system, the ESP32-CAM features a highly compet-

itive small-size camera module.

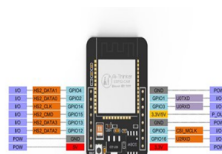
The ESP-32CAM may be used in a wide range of IoT applications. It is suitable for Internet of Things applications such as wireless positioning system signals, wireless monitoring, wireless control in industrial settings, wireless QR identification, and smart home appliances. It is the ideal choice for IoT applications. In order to facilitate speedy product manufacturing while providing customers with a high-reliability connection option that is useful for usage in a range of IoT hardware terminals, the ESP-32CAM employs a DIP package and may be inserted directly into the backplane.



Figure 3.9: ESP-32 cam module

Pin Configuration

The 48 pins on the ESP32 chip may be used for a variety of purposes. Some ESP32 development boards do not expose all of the pins, while others cannot be utilized.



CAM	ESP32	SD	ESP32
D0	PIN5	CLK	PIN14
D1	PIN18	CMD	PIN15
D2	PIN19	DATA0	PIN2
D3	PIN21	DATA1/flash	PIN4
D4	PIN36	DATA2	PIN12
D5	PIN39	DATA3	PIN13
D6	PIN34		
D7	PIN35		
XCLK	PIN0		
PCLK	PIN22		
VSYNC	PIN25		
HREF	PIN23		
SDA	PIN26		
SCL	PIN27		
POWER PIN	PIN32		

Figure 3.10: Specifications of ESP 32 cam

SPECIFICATIONS

Module Model	ESP32-CAM
Package	DIP-16
Size	27*40.5*4.5 (±0.2) mm
SPI Flash	Default 32Mbit
RAM	520KB SRAM+4M PSRAM
Bluetooth	Bluetooth 4.2 BR/EDR and BLE standards
Wi-Fi	802.11 b/g/n/
Support interface	UART、SPI、I2C、PWM
Support TF card	Maximum support 4G
IO port	9
UART Baudrate	Default 115200 bps
Image Output Format	JPEG(OV2640 support only),BMP,GRAYSCALE
Spectrum Range	2412 ~2484MHz
Antenna	Onboard PCB antenna, gain 2dBi
Transmit Power	802.11b: 17±2 dBm (@11Mbps) 802.11g: 14±2 dBm (@54Mbps) 802.11n: 13±2 dBm (@MCS7)
Receiving Sensitivity	CCK, 1 Mbps: -90dBm CCK, 11 Mbps: -85dBm 6 Mbps (1/2 BPSK): -88dBm 54 Mbps (3/4 64-QAM): -70dBm MCS7 (65 Mbps, 72.2 Mbps): -67dBm
Power Dissipation	Turn off the flash lamp: 180mA@5V Turn on the flash lamp and turn on the brightness to the maximum: 310mA@5V Deep-sleep: Minimum power consumption can be achieved 6mA@5V Modern-sleep: Minimum up to 20mA@5V Light-sleep: Minimum up to 6.7mA@5V
Security	WPA/WPA2/WPA2-Enterprise/WPS
Power Supply Range	5V
Operating Temperature	-20 °C ~ 85 °C
Storage Environment	-40 °C ~ 90 °C , < 90%RH

Figure 3.11: ESP-32 cam Pin Configuration

KEY FEATURES

- The smallest 802.11b/g/n Wi-Fi BT SoC Module
- Low-power 32-bit CPU, which may also function as an application processor
- Up to 160MHz clock speed Summary computing power up to 600 DMIPS
- Built-in Flash bulb, support for the OV2640 and OV7670 cameras.
- Built-in 520 KB SRAM, external 4MPSRAM
- It helps UART/SPI/I2C/PWM/ADC/DAC
- Support for WiFi picture upload and TF card
- Supports a variety of sleep modes.

- Embedded FreeRTOS and Lwip
- STA/AP/STA+AP operation mode is supported.
- Support the AirKit/Smart Config technology
- Support for local and remote firmware updates (FOTA) features for serial ports

Camera Specification and working

The ESP32-CAM has the following specifications:

Output formats include YUV422, YUV420, RGB565, RGB555, and 8-bit compressed data. Image transfer rate is 15 to 60 frames per second. With 2megapixel camera.

Using the ESP32-CAM is very easy because it is built in module so we no need to attach extra cam module to get video streaming. Just do the Arduino coding and control the esp module camera. We can done the coding in Arduino and access the camera module.

Working

Dc motor is working for vehicle movement. We can design a coding on Arduino and microcontroller which can rotates the motor by combinations. We assign the combination through coding and attach the pins as per requirement and it can rotate left or right and move forward or backward with proper speed.

3.1.8 Arduino

The Arduino electronics platform is composed entirely of open-source hardware and software. For developing interactive electronic projects, it offers a flexible and user-friendly framework.

The hardware of the Arduino is often built around microcontrollers, which are compact, programmable devices that can sense inputs from the outside world and operate outputs like lights, motors, and sensors. For a variety of project needs, Arduino boards come in a variety of sizes and designs.

To develop, build, and upload code to the Arduino board, utilise the Arduino soft-

ware, sometimes referred to as the Arduino IDE (Integrated Development Environment). Even those without substantial programming knowledge can use it because it employs a condensed version of the C++ programming language.

Some key features and benefits of Arduino include

Open-source

Being open-source, the hardware, software, and design files for the Arduino platform are all freely accessible. This promotes teamwork, knowledge exchange, and the growth of a sizable community.

Easy prototyping

Electronic project prototyping is made easier with Arduino. It offers a large range of integrated libraries and functionalities that abstract complicated processes, allowing users to concentrate on the needs of their particular projects.

Extensibility Developing electronic project prototypes is made easier using Arduino. It offers a large selection of pre-built libraries and features that abstract complicated processes, allowing users to concentrate on the needs of their own projects.

Community and resources

Arduino has a large and active community of users, makers, and developers. This community provides support, tutorials, sample projects, and code examples, making it easier for beginners to get started and for advanced users to find solutions to complex problems.

Versatility: Robotics, home automation, IoT projects, interactive art pieces, sensor networks, and many other fields can all benefit from the use of Arduino.

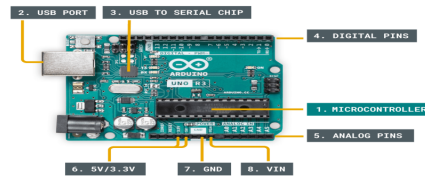


Figure 3.12: Arduino Pin outs

3.1.9 A4988 Motor Driver

A common stepper motor driver, the A4988, is employed in a number of projects involving robotics, CNC machines, and 3D printers. It is made to control bipolar stepper motors by giving them the current and signals required for accurate positioning and movement. Here are some of the A4988 motor driver's salient characteristics.

Bipolar stepper motors are frequently utilised in numerous applications, and the A4988 is specifically made to control them. With a voltage range of 8V to 35V, it can handle motors with up to 2A per coil.

The A4988 is capable of microstepping, which enables smoother and more accurate control of stepper motors. To enable more precise control over the motor, it can be adjusted to give full-step, half-step, quarter-step, eighth-step, or sixteenth-step microstepping modes.

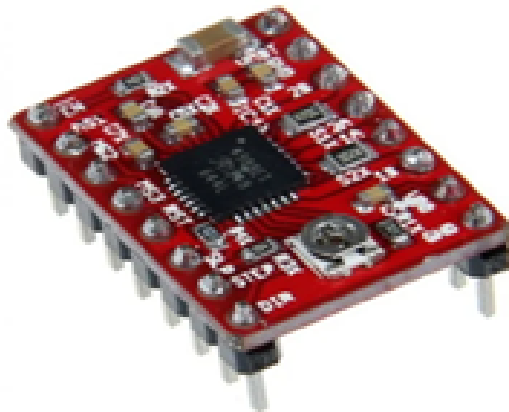


Figure 3.13: A4988 Driver Module

3.2 Design Procedure

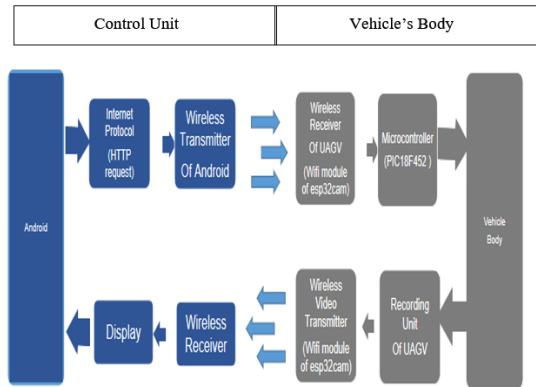


Figure 3.14: Control Unit + Vehicle Body Flow diagram

Flow diagram of this project consists of two main portions. These two portions physically separated i-e there is no physical link between these two Portions. But these two parts are wirelessly linked by transceivers. One Portion may be known as control unit that will control other part wirelessly. Other portion may be called vehicle's main body.

3.2.1 Control Unit / First Portion

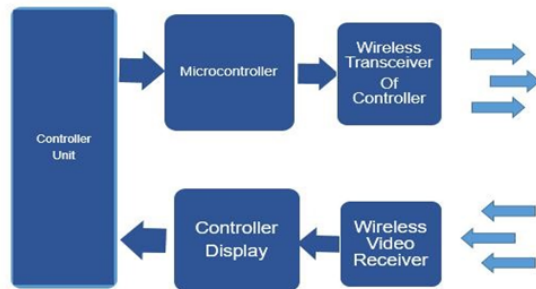


Figure 3.15: Android Processing steps

First portion may be called as controller unit as it would be operated from a remote location to send commands and get live video from second portion.

Division

Control unit would consist of:

- Console

- Keys
- Microcontroller
- Transceiver

We design a web page which is open on android and control the UAGV. There is a Arduino coding that generates combination for different controls.

Control Flow

When a specific key would be pressed.

- Microcontroller would detect and manipulate it.
- Microcontroller then interact with transceiver.
- And command would be transmitted by transceiver.
- Transceiver would also continuously receiving live video coming from other portion and displaying on console.

3.2.2 Vehicle Body / Second Portion

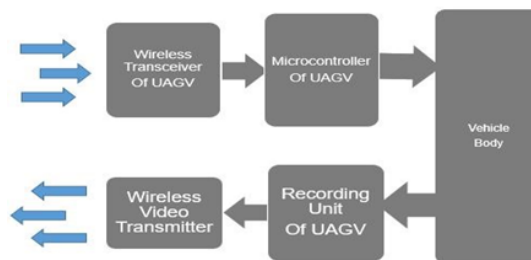


Figure 3.16: UGV Controllers Processing steps

Second portion of project flow diagram is showing vehicle's body.

Division

Vehicle's body can be divided into different parts

- Wireless transceiver
- Armed Actuator

- Microcontroller
- Camera Unit
- Vehicle Structure

Control Flow

- Command Sent by Control unit's transceiver would be received by Vehicle's transceiver.
- Microcontroller would immediately get the command and process it.
- Microcontroller would Process its Peripheral according to the command.
- According to command vehicle's components (Motors and actuators) would work.
- Also Vehicle's transceiver would continuously take like video and from camera and send it to transceiver. And Transceiver then transmit live video.

CHAPTER 4

Results and Discussion

4.1 Simulation Results

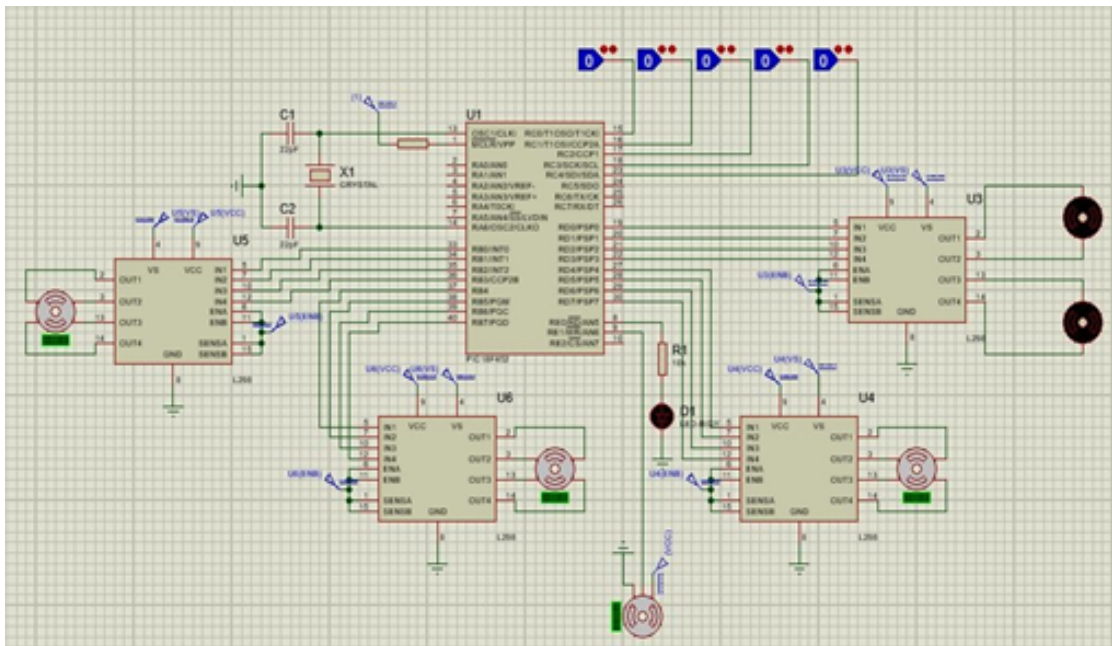


Figure 4.1: Simulation of project

4.1.1 Explanation

In the above ,five inputs that are basically outputs of the ESP-32 module going into the microcontroller PIC18f452.These combinations are basically controlling the three stepper motors,two dc gear motors and a servo mottor.

Binary Combination	Associated Action
00000	No Function
00001	Actuator Upward
00010	Actuator Right
00011	Actuator Downwards
00100	Actuator Left
00101	Actuator Upward Right
00110	Actuator Downwards Right
00111	Actuator Downwards Left
01000	Vehicle Upward Left
01001	Actuator speed++
01010	Actuator speed--
01011	Vehicle Forward
01100	Vehicle Rotate Right
01101	Vehicle Backward
01110	Vehicle Rotate Left
01111	Trigger the weapon

Figure 4.2: Body Parts Movement Combinations

The controller L298N is controlling the two dc motors according to commands given in code. These motors are placed for movement of the UGV. Then each controller L298N is controlling a single stepper motor. Steppers motors will work according to the commands given in Mikro C code through microcontroller.

A single servo motor is also there without any controller to trigger the gun. This whole circuit will run and all motors will work according to the outputs that are inputs of the microcontroller PIC18f452. With each combination the working will change/move forward, backward, gun up and down, and triggering of the gun etc.

4.1.2 Specifications and Calculations

Battery

We are using a 12V 5Ah rated sealed Lead Acid battery.

Charging Current

We know that Charging Current for Battery = 20% of Battery Ah rating
 $= 0.20 \times 5 = 1A$

Charging Time

We know Charging Time of Battery = Battery Ah rating / Charging Current

$$= 5\text{Ah} / 1\text{A} = 5 \text{ hours}$$

12V sealed lead acid batteries are fully charged at around 12.89 volts and fully discharged at around 12.23 volts (assuming 50% max depth of discharge).

Voltage (V)	Capacity (In percenatage)
12.89	100 %
12.78	90 %
12.65	80 %
12.51	70 %
12.41	60 %
12.23	50 %
12.11	40 %
11.96	30 %
11.81	20 %
11.70	10 %
11.63	0 %

Dc Motor

Features of High Torque at 150 RPM 12V DC motors with a metal gearbox and gears.

12V DC gear motor.

base motor, 18000 RPM.

M3 thread hole on a 6mm diameter shaft.

37 mm is the gearbox diameter.

28.5 mm is the motor diameter.

length without the shaft, 63 mm.

30mm shaft length.

weight of 180gm.

torque of 32 kg/cm.

No-load current = 8.0 mA, Load current = upto 7.5 A(Max).

Servo Motor

Weight.: 55g

Size Dimention: 40.7mm, 19.7mm, and 42.9mm

10 kg-cm (4.8V) and 12 kg-cm (6V) of stall torque

Operating Speed: 0.23 s/60° (4.8 V); 0.2 s/60° (6.0 V)

Voltage Range: 4.8 to 6.6 volts

Type of Gear: Metal Gear

Range of temperatures: 0°C to 55°C

1us Dead Band Width

Servo wire length is 32 cm, and its idle current draw is 10 mA.

Operating Current Without Load: 170mA

Current Stall: 1.2A

Included are Servo Arms and Screws.

Stepper Motor

PERFORMANCE SPECIFICATIONS

0.59 Nm of holding torque

Voltage Desired 3.6 V

3.0 mH for phase inductivity

resistance to isolation at 100 M/min at 500 V DC

Class B isolation (130 °C)

82 g/cm² of rotational inertia

Torque detent 0.02 Nm

Operating range: -10 to 50 °c

Step Angle 1.8° 5% No. of Phases Rated Current 2.0 A 1.8 2 Phase Resistance

4.2 Hardware

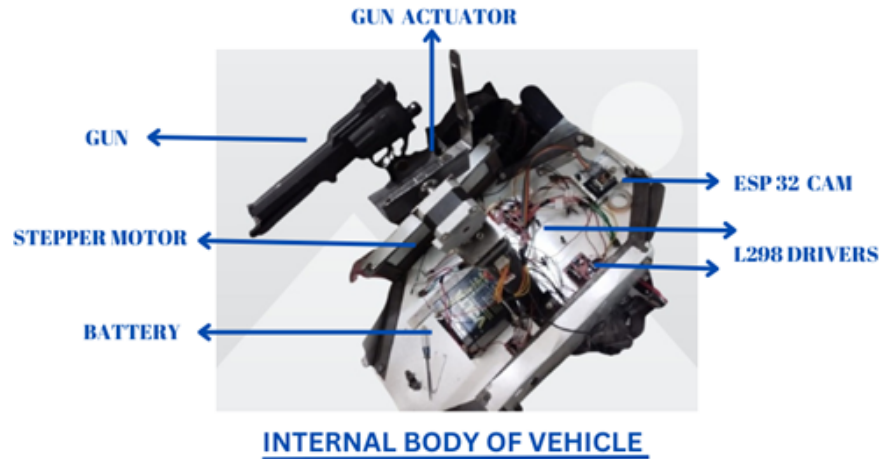


Figure 4.3: Hardware labelled parts

4.2.1 Components Explanation

Battery

Battery is used to power up the robot. Each motor and all of the components like esp32cam, Arduino by using regulators because they can operate at 5v.

Esp32cam

It can provide us a live video streaming and working as transceiver (Receiver and Transmitter) for video stream.

Dc Motors

Two Dc motors are used in our project that rotate the vehicle body front, back, right and left. It operates at 12v and draw current of 7.5A.

Stepper Motor

Two stepper Motors are used, one is for horizontal movement of gun and second one is for vertical movement of gun. These motors can rotate around 360 degree in horizontal and vertical also. Each Stepper motor can operate at 12v. It can draw 7.5-8 mA current.

One L298 two A4988 Motor Driver

These driver are used to operate dc and stepper motors. One driver can controll 2 dc motors and 2 drivers can be used for two stepper motors.

Gun

It is used to aim/hit the target by through live esp32cam video streaming.

Servo Motor

It is used to trigger the gun.

4.2.2 External Hardware

It will consists of vehicle body and user interface. **Vehicle Body**

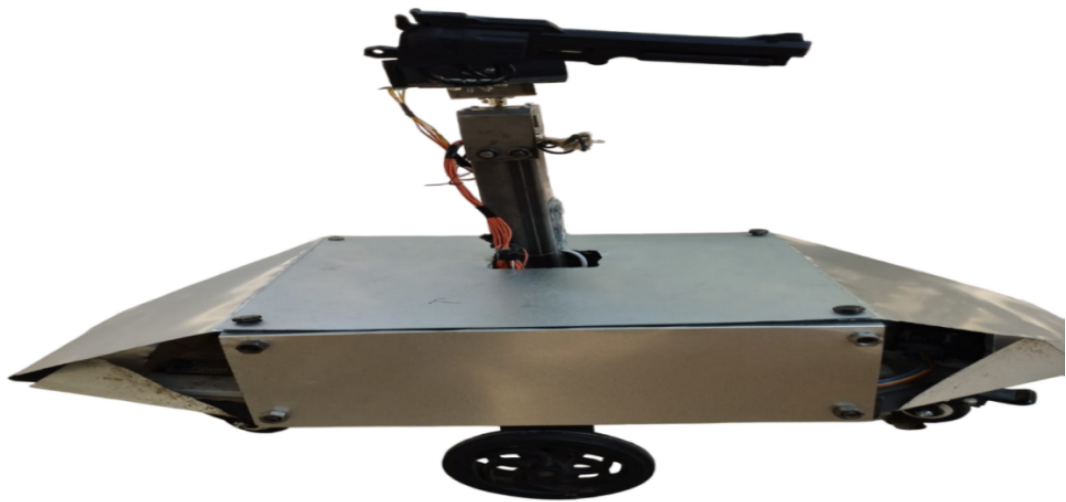


Figure 4.4: External Look of hardware

4.2.3 User Interface

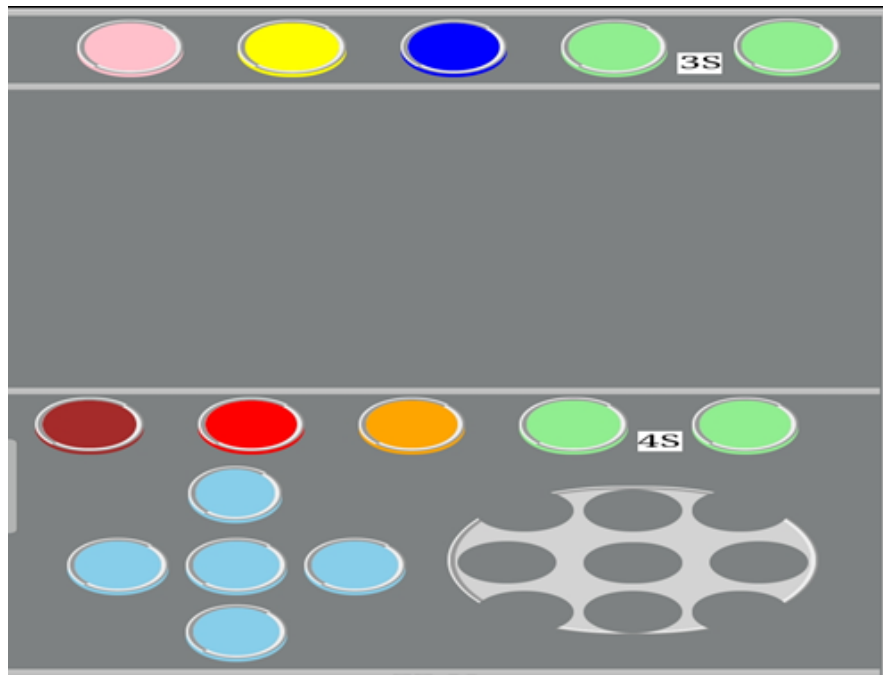


Figure 4.5: User Interface

- Big Round Button > To Control armed actuator
- Red Button > To Trigger Weapon
- Big Round Button > To Control armed actuator
- Red Button > To Trigger Weapon
- Blue Buttons > Vehicle Control Buttons
- Green Buttons > To vary the speed of Actuator
- Brown Buttons > To Reload Weapon
- Yellow Button > To Switch Laser
- Box > To demonstrate Live video

CHAPTER 5

Conclusion

The combination of various developments under has given us the method for achieving goals that have never been recognize before. These developments accomplish a controlled machine to deal with Circumstances isolated and work with a human's occupation in circumstances. In this Project we are supposed to come with such an unmanned ground vehicle that would be armed with a gun and would be able to shot down different targets in real time. This machine would be wirelessly controlled from a remote location by a person. Goal of this machine would only be the well being of man kind.

CHAPTER 6

Impact On Environment and SDG's

6.1 Impact on Environment

6.1.1 Low Cost

We use the module Esp 32 Cam Which is less costly as compared to other modules/ methods like Rasperry pi, FGPA and some other single chip computers.

6.1.2 Life Saving

As it is unmanned so no need of an power on fighting spot so no causalities expected so it is a life saving war machine.

6.1.3 Easy to Operate

There is no specific device and application required to operate/ control the UAGV. We can simply control it by using web page user interface. S its easy to operate.

6.1.4 Pollution Free

We are not usisng mechanical engines or other polluting equipumnts so it's a environ-ment friendly robot. We can use batteries which can be charged electrically and give no

bad effects to environment.

6.2 Sustainable Development Goals

It is being used for human well being. To avoid life losses robots can be used. It can be useful for armed operation and also use in security surveillance. We can replace man by robots because some robots are autonomous (which can operate automatically) So they can protect the location at which they are placed or fixed.

“It is affordable and having clean energy. Any armed forces can develop it because it is not too much costly and it can also save the human lives during conflicts so it is very beneficial for our forces to protect sovereignty and border security. It is reliable , sustainable, affordable and modern technology product. As it is having no pollution generating component As it is very clean energy product so it is not pollute our environment.

This product can also be used for reduced inequalities within and among the countries. Some countries having less population if they can use robots for their security purpose, for fight among soldiers and safety purpose then they can easily reduce inequalities. As it can work in stand alone systems also so they can use robots in private security like in rare things protection. When a country having less population can manufacture robots for its strength increasing then it can reduced inequalities.

By using robots we can get peace justice and strong institutions. In order to achieve sustainable development, we may encourage inclusive and peaceful communities, ensure that everyone has access to justice, and create inclusive institutions at all levels.

Because when we stop invaders, thieves and suicide attackers by robots then a peace come in a society. When we achieve peace in a society then we can develop a strong institutions. If we are surrounded by invaders, our center of attraction is peace then how we can get strong institutions , and development. So if we can apply the robotic power and get peace then why we not using them.

References

1. VOL. 13, NO. 3, FEBRUARY 2018 ISSN 1819- 6608 ... ARPJ Journal of Engineering and Applied Sciences ... Kolapo Sulaimon Alli, Moses Oluwafemi Onibonjoje, Akinola S. Oluwole, Michael Adegoke ...
2. V.Abilash, J.Paul Chandra Kumar, "Arduino controlled landmine detection robot", Third International Conference on Science Technology Engineering and Management (ICONSTEM), 2017.
3. Rathesh Kumar, Amulya Niraj Khalko, "Design and implementation of metal detector using DTMF technology", International Conference on Signal Processing, Communication, Power and Embedded System, 2016.
4. Ramneet Kaur¹ and Balwinder Singh, "design and implementation of car parking system on FPGA" International Journal of VLSI design & Communication Systems (VLSICS) Vol.4, No.3, June 2013
5. P. V. K. Borges, A. Tews, and D. Haddon, "Pedestrian detection in industrial environments: Seeing around corners," in Proc. IEEE/RSJ Int.Conf. Intell. Robots Syst., IROS'12, 2012, pp. 4231–4232.
6. J. Rodriguez-Araujo, J. Rodriguez-Andina, "Field programable system on chip for localization of UGVs in an indoor iSpace," IEEE Transaction on industrial informatics, vol.10,no.2,MAY 2014
7. M. Sathiyarayanan, Syed Azharuddin, Santhosh Kumar and Gibran Khan, Gesture Controlled Robot for Military Purpose, International Journal for Technological Research in Engineering, Vol 1, issue 11, pp. 1300-1303, July 2014.

8. Jiabao Wang, Yafei Zhang, Jianjiang Lu and Weiguang Xu, " A framework for moving target detection, recognition and tracking in uav videos.", *Affective Computing and Intelligent Interaction*, Springer-Verlag Berlin Heidelberg 2012, pp. 69-76.
9. Z. M. A. N. MEHMOOD, "DESIGN AND IMPLEMENTATION OF LOW COST REMOTE-OPERATED UNMANNED GROUND VEHICLE (UGV)," IN INTERNATIONAL CONFERENCE ON ROBOTICS AND EMERGING ALLIED TECHNOLOGIES IN ENGINEERING (ICREATE), 2014.
10. DC MOTORS - STEVEN ENGINEERING," [ONLINE]. AVAILABLE: STEVENENGINEERING.COM/TECH_SUPPORT/PDFS/44MOTORS.PDF.
11. J.Pyo, Future Unmanned System Design for Reliable Military Operations, *International Journal of Control and Automation*. Korea, vol. 5, pp. 173-186, September 2012

Appendices

Coding For Pic18F452 in Mikro c

```
unsigned int ActDelay;
```

```
unsigned int ci;
```

```
unsigned int S1Step=1;
```

```
unsigned int S2Step=1;
```

```
void delay(unsigned int a)
```

```
switch(a) case 1: delayms(5); break;
```

```
case 2: delayms(10); break;
```

```
case 3:delayms(15); break;
```

```
case 4:delayms(20); break;
```

```
case 5:delayms(25); break;
```

```
case 6:delayms(30); break;
```

```
case 7:delayms(35); break;
```

```
void clock1()
```

```
if(S1Step==1)LATB=0b00000001; delay(ActDelay); S1Step++;
```

```
else if(S1Step==2)LATB=0b00000100; delay(ActDelay); S1Step++;
```

```
else if(S1Step==3)LATB=0b00000010; delay(ActDelay); S1Step++;
```

```
else if(S1Step==4)LATB=0b00001000; delay(ActDelay); S1Step=1;
```

```
void Anticlock1()
```

```
if(S1Step==1)LATB=0b00000100; delay(ActDelay); S1Step++;  
else if(S1Step==2)LATB=0b00000001; delay(ActDelay); S1Step++;  
else if(S1Step==3)LATB=0b00001000; delay(ActDelay); S1Step++;  
else if(S1Step==4)LATB=0b00000010; delay(ActDelay); S1Step=1;
```

```
void clock2()
```

```
if(S2Step==1) LATB=0b00010000; delay(ActDelay); S2Step++;  
else if(S2Step==2) LATB=0b01000000; delay(ActDelay); S2Step++;  
else if(S2Step==3) LATB=0b00100000; delay(ActDelay); S2Step++;  
else if(S2Step==4) LATB=0b10000000; delay(ActDelay); S2Step=1;
```

```
void Anticlock2()
```

```
else if(S2Step==1) LATB=0b01000000; delay(ActDelay); S2Step++;  
else if(S2Step==2) LATB=0b00010000; delay(ActDelay); S2Step++;  
else if(S2Step==3) LATB=0b10000000; delay(ActDelay); S2Step++;  
else if(S2Step==4) LATB=0b00100000; delay(ActDelay); S2Step=1;
```

```
void clock3()
```

```
LATD=0b00010000; delay(ActDelay);  
LATD=0b01000000; delay(ActDelay);  
LATD=0b00100000; delay(ActDelay);  
LATD=0b10000000; delay(ActDelay);
```

```
void Anticlock3()
```

```
LATD=0b01000000; delay(ActDelay);
```

```
LATD=0b00010000; delay(ActDelay);  
LATD=0b10000000; delay(ActDelay);  
LATD=0b00100000; delay(ActDelay);
```

```
void main()  
ActDelay=4;  
ci=1;  
TRISB=0b00000000;  
TRISC=0b11110000;  
TRISD=0b00000000;  
while(1)  
while(PORTC.F7==1)  
if(ActDelay<=7 && ci==1)  
ActDelay++; ci=0;
```

Coding For Esp 32 Cam in Arduino IDE

```
:[(/* 192.168.4.1 */  
#include "esphttpserver.h"  
#include "esptimer.h"  
#include "espcamera.h"  
#include "imgconverters.h"  
#include "Arduino.h"  
#include "espcamera.h"  
#include "esp32-hal-ledc.h"  
#include "soc/soc.h"  
#include "soc/rccntreg.h"  
#include <WiFi.h>  
#include <WiFiClient.h>
```

```
#include <WiFiAP.h>
#define CAMERAMODELAITHINKER
#define PWDNGPIONUM 32
#define RESETGPIONUM -1
#define XCLKGPIONUM 0
#define SIODGPIONUM 26
#define SIOCGPIONUM 27
#define Y9GPIONUM 35
#define Y8GPIONUM 34
#define Y7GPIONUM 39
#define Y6GPIONUM 36
#define Y5GPIONUM 21
#define Y4GPIONUM 19
#define Y3GPIONUM 18
#define Y2GPIONUM 5

#define VSYNCGPIONUM 25
#define HREFGPIONUM 23
#define PCLKGPIONUM 22

/* Wifi Crdentials // Replace your SSID and Password */
const char* ssid = "UnmannedGroundVehicle";
const char* password = "190487";

void startCameraServer();

/* Defining DC motor, command and Flash LED pins */
const int RMotor1 = 14;
const int RMotor2 = 15;
const int LMotor1 = 13;
```

```
const int LMotor2 = 12;
```

```
int com1=6;
```

```
int com2=2;
```

```
int com3=3;
```

```
int com4=4;
```

```
const int FlashPin = 4;
```

```
/* Defining initial values */
```

```
int speed = 255;
```

```
int panVal = 4875;
```

```
int tiltVal = 4875;
```

```
#define PARTBOUNDARY "123456789000000000000987654321"
```

```
static const char* STREAMCONTENTTYPE = "multipart/x-mixed-replace;boundary="
```

```
PARTBOUNDARY;
```

```
httpdhandler streamhttpd = NULL;
```

```
httpdhandler camerahttpd = NULL;
```

```
/* Stream handler */
```

```
static esprtt streamhandler(httpdreq *req) ""
```

```
camerafbt * fb = NULL;
```

```
esprtt res = ESPOK;
```

```
size_t jpgbuflen = 0;
```

```
uint8t * jpgbuf = NULL;
```

```
char * partbuf[64];
```

```
static int64t lastframe = 0;
```

```
if (!lastframe)
```

```
lastframe = esptimergettime();
```

```
res = httpdrespsettype(req, STREAMCONTENTTYPE);
if (res != ESPOK)
return res;

while (true)
fb = espcamerafbget();
if (!fb)
Serial.println("Camera capture failed");
res = ESPFAIL;
else

if (fb->format != PIXFORMATJPEG)
bool jpegconverted = frame2jpg(fb, 80, &jpgbuf, &jpgbuflen);
espcamerafbreturn(fb);
fb = NULL;
if (!jpegconverted)
Serial.println("JPEG compression failed");
res = ESPFAIL;

else
jpgbuflen = fb->len;
jpgbuf = fb->buf;

if (res == ESPOK)
```

```
size_t hlen = snprintf((char *)partbuf, 64, STREAMPART, jpgbuflen);
res = httpdrespsendchunk(req, (const char *)partbuf, hlen);

if (res == ESPOK)
res = httpdrespsendchunk(req, (const char *)jpgbuf, jpgbuflen);

if (res == ESPOK)
res = httpdrespsendchunk(req, STREAMBOUNDARY, strlen(STREAMBOUNDARY));

if (fb)
espcamerafbreturn(fb);
fb = NULL;
jpgbuf = NULL;
else if (jpgbuf)
free(jpgbuf);
jpgbuf = NULL;

if (res != ESPOK)
break;

int64t frend = esptimergettime();
int64t frametime = frend - lastframe;
lastframe = frend;
frametime /= 1000;
/*Serial.printf("MJPG: %uB %ums (%.1ffps)",
(uint32t)jpgbuflen,
(uint32t)frametime, 1000.0 / (uint32t)frametime
);*/

lastframe = 0; return res;”
```

```

/* Command handler */ static esperrt cmdhandler(httpdreq *req) “ char* buf; sizer
buflen; char variable[32] = 0,; char value[32] = 0,;

buflen = httpdreqgeturlquerylen(req) + 1; if (buflen > 1) buf = (char*)malloc(buflen);
if (!buf) httpdrespsend500(req); return ESPFAIL; ” if (httpdreqgeturlquerystr(req, buf,
buflen) == ESPOK) if (httpdquerykeyvalue(buf, "var", variable, sizeof(variable)) ==
ESPOK &&
httpdquerykeyvalue(buf, "val", value, sizeof(value)) == ESPOK)
else “
free(buf);
httpdrespsend404(req);
return ESPFAIL;

else
free(buf);
httpdrespsend404(req);
return ESPFAIL;

free(buf);
else
httpdrespsend404(req);
return ESPFAIL;

int val = atoi(value);
sensort * s = espcamerasensorget();
int res = 0;
/* Flash LED control */
if (!strcmp(variable, "flash"))

ledcWrite(3, val);

```



```
else if (!strcmp(variable, "speed"))
```

```
/* Setting the motor speed */
```

```
if (val > 255) val = 255;
```

```
else if (val < 0) val = 0;
```

```
speed = val;
```

```
/* Robot direction control */
```

```
else if (!strcmp(variable, "car"))
```

```
if (val == 1)
```

```
Serial.println("Forward");
```

```
ledcWrite(4, speed);
```

```
ledcWrite(5, 0);
```

```
ledcWrite(6, 0);
```

```
ledcWrite(7, speed);
```

```
else if (val == 2)
```

```
Serial.println("Turn Left");
```

```
ledcWrite(4, speed);
```

```
ledcWrite(5, 0);
```

```
ledcWrite(6, speed);
```

```
ledcWrite(7, 0);
```

```
else if (val == 3)
```

```
Serial.println("Stop");
```

```
ledcWrite(4, 0);
```

```
ledcWrite(5, 0);
```

```
ledcWrite(6, 0);
```

```
ledcWrite(7, 0);
```

```
digitalWrite(com1, LOW);
```

```
digitalWrite(com2, LOW);
digitalWrite(com3, LOW);

else if (val == 4)
Serial.println("Turn Right");
ledcWrite(4, 0);
ledcWrite(5, speed);
ledcWrite(6, 0);
ledcWrite(7, speed);

else if (val == 5)
Serial.println("Backward");
ledcWrite(4, 0);
ledcWrite(5, speed);
ledcWrite(6, speed);
ledcWrite(7, 0);

/* Pan and Tilt servo control */
else if (!strcmp(variable, "pantilt"))

if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3,
LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");

else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3,
LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");

else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3,
LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3, LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3, LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3, LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3, LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3, LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3, LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3, LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3, LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3, LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3, LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3,  
LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3,  
LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3,  
LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3,  
LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3,  
LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3,  
LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else if (val == 4) digitalWrite(com1, LOW); digitalWrite(com2, HIGH); digitalWrite(com3,  
LOW); digitalWrite(com4, LOW); ledcWrite(3, 255);Serial.println("Down");
```

```
else
```

```
Serial.println("variable");
```

```
res = -1;
```

```
if (res)
```

```
return httpdrespsend500(req);
```

```

httpdrespsethdr(req, "Access-Control-Allow-Origin", "*");
return httpdrespsend(req, NULL, 0);

static esperrt statushandler(httpdreq *req)
static char jsonresponse[1024];

sensort * s = espcamerasensorget();
char * p = jsonresponse;
*p++ = `;

p += sprintf(p, "framesize:%u,", s->status.framesize);
p += sprintf(p, "quality:%u,", s->status.quality);
*p++ = `;
*p++ = 0;
httpdrespsettype(req, "application/json");
httpdrespsethdr(req, "Access-Control-Allow-Origin", "*");
return httpdrespsend(req, jsonresponse, strlen(jsonresponse));

/* Index HTML page design */
static const char PROGMEM INDEXHTML[] = R"rawliteral(
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width,initial-scale=1">
<title>Unmanned Armed Vehicle</title>

<style>

```

```
.buttonfirebackground-color:red; border:outset; width:60px; height:60px; border-radius:50%;
```

```
.buttoncontrolbackground-color:skyblue; border:outset; width:60px; height:60px; border-  
radius:50%;
```

```
.buttonactuatorbackground-color:lightgrey; border:outset; width:60px; height:60px; border-  
radius:50%;
```

```
#firemargin-right:60px;
```

```
#forwardmargin-right:60px;
```

```
#downmargin-right:60px;
```

```
#backmargin-right:60px;
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div align=center>
```

```
<label class="label">Flash</label>
```

```
<input type="range" class="slider" id="flash" min="0" max="255" value="0"
```

```
onchange="tryfetch(document.location.origin+'/control?var=flash&val='+this.value);catch(e)">
```

```
</div>
```

```
<br/>
```

```
<div align=center> <img id= "camstream" src="" style='width:300px;'></div>
```

```
<br/>
```

```
<div align=center>
```

```
<label class="label">Speed</label>
```

```
<input type="range" class="slider" id="speed" min="0" max="255" value="255"
```

```
onchange="tryfetch(document.location.origin+'/control?var=speed&val='+this.value);catch(e)">
```

```
</div>
```

```
<br/>
```

```

<p align=center>
<button class="buttonfire" id="fire" onclick="fetch(document.location.origin+'/control?
var=pantilt&val=3');" ></button>
<button class="buttoncontrol" id="forward" ontouchstart="fetch(document.
location.origin+'/control?var=car&val=1');
ontouchend="fetch(document.location.origin+'
/control?var=car&val=3');" > </button>

```

```

</p>

```

```

<p align=center> <button class="buttonactuator"
ontouchstart="fetch(document.location.origin+'/control?var=pantilt&val=1');"
ontouchend="fetch(document.location.origin+'
/control?var=pantilt&val=2');" > </button>

```

```

<button class="buttoncontrol"
ontouchstart="fetch(document.location.origin+'/control?var=car&val=2');"
ontouchend="fetch(document.location.origin+'/control?var=car&val=3');" > </button>

```

```

<button class="buttoncontrol"
ontouchstart="fetch(document.location.origin+'/control?var=car&val=3');"
ontouchend="fetch(document.location.origin+'/control?var=car&val=3');" ></button>

```

```

<button class="buttoncontrol"
ontouchstart="fetch(document.location.origin+'/control?var=car&val=4');"
ontouchend="fetch(document.location.origin+'/control?var=car&val=3');" ></button>

```

```

</p>

```

```

<p align=center>

```

```
<button class="buttonactuator" id="down"
ontouchstart="fetch(document.location.origin+'/control?var=pantilt&val=4');"
ontouchend="fetch(document.location.origin+'/control?var=pantilt&val=2');"></button>
```

```
<button class="buttoncontrol" id="back"
ontouchstart="fetch(document.location.origin+'/control?var=car&val=5');"
ontouchend="fetch(document.location.origin+'/control?var=car&val=3');"></button>
```

```
</p>
```

```
<script>
```

```
window.onload = document.getElementById("camstream").src = window.location.href.slice(0,
-1) + ":81/stream";
```

```
</script>
```

```
</body>
```

```
</html>
```

```
)rawliteral";
```

```
static esperrt indexhandler(httpdreq *req)
```

```
httpdresptype(req, "text/html");
```

```
return httpdrespsend(req, (const char *)INDEXHTML, strlen(INDEXHTML));
```

```
void startCameraServer()
```

```
httpdconfig config = HTTPDDEFAULTCONFIG();
```

```
httpdurit indexuri =
```

```
.uri = "/",
```

```
.method = HTTPGET,
```

```
.handler = indexhandler,
```



```
.userctx = NULL  
;
```

```
httpdurit statusuri =  
.uri = "/status",  
.method = HTTPGET,  
.handler = statushandler,  
.userctx = NULL  
;
```

```
httpdurit cmduri =  
.uri = "/control",  
.method = HTTPGET,  
.handler = cmdhandler,  
.userctx = NULL  
;
```

```
httpdurit streamuri =  
.uri = "/stream",  
.method = HTTPGET,  
.handler = streamhandler,  
.userctx = NULL  
;
```

```
Serial.printf("Starting web server on port: '%d'", config.serverport);  
if (httpdstart(&camerahttpd, &config) == ESPOK)  
httpdregisterurihandler(camerahttpd, &indexuri);  
httpdregisterurihandler(camerahttpd, &cmduri);  
httpdregisterurihandler(camerahttpd, &statusuri);
```

```
config.serverport += 1;
config.ctrlport += 1;
Serial.printf("Starting stream server on port: '%d'", config.serverport);
if (httpdstart(&streamhttpd, &config) == ESPOK)
httpdregisterurihandler(streamhttpd, &streamuri);

void initMotors()

/* Configuring PWM channels for DC motors */ ledcSetup(Channel, Frequency, Res-
olution) */
ledcSetup(4, 2000, 8); /* 2000 hz PWM, 8-bit resolution and range from 0 to 255 */
ledcSetup(5, 2000, 8);
ledcSetup(6, 2000, 8);
ledcSetup(7, 2000, 8);
/* Attaching the channel to the GPIO to be controlled */
/* ledcAttachPin(GPIO, Channel) */
ledcAttachPin(RMotor1, 4);
ledcAttachPin(RMotor2, 5);
ledcAttachPin(LMotor1, 6);
ledcAttachPin(LMotor2, 7);

void coms()
pinMode(com1, OUTPUT);
pinMode(com2, OUTPUT);
pinMode(com3, OUTPUT);

void initFlash()
/* Configuring PWM channels for Falsh LED */
ledcSetup(3, 5000, 8); /* 5000 hz PWM, 8-bit resolution and range from 0 to 255 */
ledcAttachPin(FlashPin, 3);
```

```
void setup()
WRITEPERIREG(RTCCNTLBROWNOUTREG, 0);
Serial.begin(115200);
Serial.setDebugOutput(true);

initMotors();
coms();
initFlash();
Serial.println();
cameraconfigt config;
config.ledcchannel = LEDCCHANNEL0;
config.ledctimer = LEDCTIMER0;
config.pind0 = Y2GPIONUM;
config.pind1 = Y3GPIONUM;
config.pind2 = Y4GPIONUM;
config.pind3 = Y5GPIONUM;
config.pind4 = Y6GPIONUM;
config.pind5 = Y7GPIONUM;
config.pind6 = Y8GPIONUM;
config.pind7 = Y9GPIONUM;
config.pinxclk = XCLKGPIONUM;
config.pinpclk = PCLKGPIONUM;
config.pinvsync = VSYNCGPIONUM;
config.pinhref = HREFGPIONUM;
config.pinscsda = SIODGPIONUM;
config.pinscscl = SIOCGPIONUM;
config.pinpwn = PWDNGPIONUM;
config.pinreset = RESETGPIONUM;
config.xclkfreqhz = 20000000;
config.pixelformat = PIXFORMATJPEG;
if (psramFound())
```

```
config.framesize = FRAMESIZEQVGA;
config.jpegquality = 10;
config.fbcount = 2;
else
config.framesize = FRAMESIZEQVGA;
config.jpegquality = 12;
config.fbcount = 1;

esperrrt err = espcamerainit(&config);
if (err != ESPOK)
Serial.printf("Camera init failed with error 0x%x", err);
return;

sensort * s = espcamerasensorget();
s->setframesize(s, FRAMESIZEQVGA);
s->setvflip(s, 1);
s->sethmirror(s, 1);

/* Connecting to WiFi */
WiFi.softAP(ssid, password);
IPAddress IP = WiFi.softAPIP();
Serial.print("AP IP address: ");
Serial.println(IP);
startCameraServer();
Serial.println("Server started");
for (int i = 0; i < 5; i++)
ledcWrite(3, 10);
delay(100);
ledcWrite(3, 0);
delay(100);
void loop()
)"]:
```

Coding For Arduino Uno in Arduino IDE

```
#include <Servo.h>
Servo myservo;
#define H 12
int x1=0;
int x2=0;
int StepperDelayx=4;
int VehicleSpeedx=2;
int ActuatorLimitX=50;
int ActuatorLimitX=100;
int LightStatus=0;
int pos = 0;
int NoRepeat = 0;
void StepperDelay(int a)
if(a==7)delayMicroseconds(2000);
else if(a==6)delayMicroseconds(3000);
else if(a==5)delayMicroseconds(4000);
else if(a==4)delayMicroseconds(5000);
else if(a==3)delayMicroseconds(6000);
else if(a==2)delayMicroseconds(7000);
else if(a==1)delayMicroseconds(8000);

void StepperSpeed(int a)
if(a==1) if(StepperDelayx<=6) StepperDelayx++;
if(a==2) if(StepperDelayx>=2) StepperDelayx--;

void VehicleSpeed(int a)
if(a==1) if(VehicleSpeedx<=4) VehicleSpeedx++;
if(a==2) if(VehicleSpeedx>=2) VehicleSpeedx--;
void VehicleForward(int a)
if(a==1) digitalWrite(M1A, HIGH); digitalWrite(M2A, HIGH); delayMicroseconds(20);
digitalWrite(M1A,LOW);
```

```
digitalWrite(M2A, LOW); delayMicroseconds(80);
else if(a==2) digitalWrite(M1A, HIGH); digitalWrite(M2A, HIGH); delayMicrosec-
onds(40);
digitalWrite(M1A,LOW); digitalWrite(M2A, LOW); delayMicroseconds(60);
else if(a==3) digitalWrite(M1A, HIGH); digitalWrite(M2A, HIGH); delayMicrosec-
onds(60);
digitalWrite(M1A,LOW); digitalWrite(M2A, LOW); delayMicroseconds(40);
else if(a==4) digitalWrite(M1A, HIGH); digitalWrite(M2A, HIGH); delayMicrosec-
onds(80);
digitalWrite(M1A,LOW); digitalWrite(M2A, LOW); delayMicroseconds(20);
else if(a==5) digitalWrite(M1A, HIGH); digitalWrite(M2A, HIGH);
```

```
void VehicleLeft(int a)
```

```
if(a==1) digitalWrite(M1A, HIGH); digitalWrite(M2B, HIGH); delayMicroseconds(20);
digitalWrite(M1A,LOW);
digitalWrite(M2B, LOW); delayMicroseconds(80);
else if(a==2) digitalWrite(M1A, HIGH); digitalWrite(M2B, HIGH); delayMicrosec-
onds(40);
digitalWrite(M1A,LOW); digitalWrite(M2B, LOW); delayMicroseconds(60);
else if(a==3) digitalWrite(M1A, HIGH); digitalWrite(M2B, HIGH); delayMicrosec-
onds(60);
digitalWrite(M1A,LOW); digitalWrite(M2B, LOW); delayMicroseconds(40);

else if(a==4) digitalWrite(M1A, HIGH); digitalWrite(M2B, HIGH); delayMicrosec-
onds(80);
digitalWrite(M1A,LOW); digitalWrite(M2B, LOW); delayMicroseconds(20);
else if(a==5) digitalWrite(M1A, HIGH); digitalWrite(M2B, HIGH);
```

```
void VehicleRight(int a)
```

```
if(a==1) digitalWrite(M1B, HIGH); digitalWrite(M2A, HIGH); delayMicroseconds(20);
```

```

digitalWrite(M1B,LOW); digitalWrite(M2A, LOW); delayMicroseconds(80);
else if(a==2) digitalWrite(M1B, HIGH); digitalWrite(M2A, HIGH); delayMicrosec-
onds(40);
digitalWrite(M1B,LOW); digitalWrite(M2A, LOW); delayMicroseconds(60);
else if(a==3) digitalWrite(M1B, HIGH); digitalWrite(M2A, HIGH); delayMicrosec-
onds(60);
digitalWrite(M1B,LOW); digitalWrite(M2A, LOW); delayMicroseconds(40);
else if(a==4) digitalWrite(M1B, HIGH); digitalWrite(M2A, HIGH); delayMicrosec-
onds(80);
digitalWrite(M1B,LOW); digitalWrite(M2A, LOW); delayMicroseconds(20);
else if(a==5) digitalWrite(M1B, HIGH); digitalWrite(M2A, HIGH);
void VehicleBackward(int a)
if(a==1) digitalWrite(M1B, HIGH); digitalWrite(M2B, HIGH); delayMicroseconds(20);
digitalWrite(M1B,LOW); digitalWrite(M2B, LOW); delayMicroseconds(80);
else if(a==2) digitalWrite(M1B, HIGH); digitalWrite(M2B, HIGH); delayMicrosec-
onds(40);
digitalWrite(M1B,LOW); digitalWrite(M2B, LOW); delayMicroseconds(60);
else if(a==3) digitalWrite(M1B, HIGH); digitalWrite(M2B, HIGH); delayMicrosec-
onds(60);
digitalWrite(M1B,LOW); digitalWrite(M2B, LOW); delayMicroseconds(40);
else if(a==4) digitalWrite(M1B, HIGH); digitalWrite(M2B, HIGH); delayMicrosec-
onds(80);
digitalWrite(M1B,LOW); digitalWrite(M2B, LOW); delayMicroseconds(20);
else if(a==5) digitalWrite(M1B, HIGH); digitalWrite(M2B, HIGH);
void ActuatorUp() if(ActuatorLimit1<=99)Clock1(); ActuatorLimit1++;
void ActuatorDown() if(ActuatorLimit1>=1)AntiClock1(); ActuatorLimit1--;
void ActuatorRight() if(ActuatorLimit2<=199)Clock2(); ActuatorLimit2++;
void ActuatorLeft() if(ActuatorLimit2>=1)AntiClock2(); ActuatorLimit2--;

void ActuatorUpRight() if(ActuatorLimit1<=99)Clock1();
ActuatorLimit1++; if(ActuatorLimit2<=199)Clock2(); ActuatorLimit2++;

```

```

void ActuatorUpLeft() if(ActuatorLimit1<=99)Clock1();
ActuatorLimit1++; if(ActuatorLimit2>=1)AntiClock2(); ActuatorLimit2--;
void ActuatorDownRight() if(ActuatorLimit1>=1)AntiClock1(); ActuatorLimit1--;
if(ActuatorLimit2<=199)Clock2(); ActuatorLimit2++; void ActuatorDownLeft()
if(ActuatorLimit1>=1)AntiClock1(); ActuatorLimit1--;
if(ActuatorLimit2>=1)AntiClock2(); ActuatorLimit2--;

```

```

void AntiClockX()
if(digitalRead(S1Dir)==0) digitalWrite(S1Dir, HIGH);

```

```

digitalWrite(S1Step, HIGH);
StepperDelay(StepperDelayx);
digitalWrite(S1Step, LOW);
StepperDelay(StepperDelayx);

```

```

void ClockX()
if(digitalRead(S1Dir)==1) digitalWrite(S1Dir, LOW);
digitalWrite(S1Step, HIGH);
StepperDelay(StepperDelayx);
digitalWrite(S1Step, LOW);
StepperDelay(StepperDelayx);
void AntiClockX()
if(digitalRead(S2Dir)==0) digitalWrite(S2Dir, HIGH);

```

```

digitalWrite(S2Step, HIGH);
StepperDelay(StepperDelayx);
digitalWrite(S2Step, LOW);
StepperDelay(StepperDelayx);
void ClockX()
if( digitalRead(S2Dir)==1)digitalWrite(S2Dir, LOW);

```



```

digitalWrite(S2Step, HIGH);
StepperDelay(StepperDelayx);
digitalWrite(S2Step, LOW);
StepperDelay(StepperDelayx);

void TrigerG()
for (pos = 0; pos <=60; pos += 1) // goes from 0 degrees to 180 degrees
// in steps of 1 degree
myservo.write(pos); // tell servo to go to position in variable 'pos'
delay(15); // waits 15ms for the servo to reach the position

for (pos = 60; pos >= 0; pos -= 1) // goes from 180 degrees to 0 degrees
myservo.write(pos); // tell servo to go to position in variable 'pos'
delay(15); // waits 15ms for the servo to reach the position

void ActuatorInitialize()
ActuatorLimit1=50;
ActuatorLimit2=100;
void Align()
while(ActuatorLimit1!=50 || ActuatorLimit2!=100)
if(ActuatorLimit1>=51) AntiClock1(); ActuatorLimit1--;
if(ActuatorLimit1<=49) Clock1(); ActuatorLimit1++;
if(ActuatorLimit2>=101) AntiClock2(); ActuatorLimit2--;
if(ActuatorLimit2<=99) Clock2(); ActuatorLimit2++;

void setup()
myservo.attach(T);
pinMode(X,OUTPUT); pinMode(X,OUTPUT); pinMode(X,OUTPUT); pinMode(X,OUTPUT);
//pinMode(X,OUTPUT); pinMode(X,OUTPUT); pinMode(X,OUTPUT); pinMode(X,OUTPUT);

```

```
pinMode(X,OUTPUT); pinMode(X,OUTPUT);
pinMode(X,OUTPUT); pinMode(X,OUTPUT);
```

```
pinMode(X,OUTPUT); pinMode(X,OUTPUT);
```

```
pinMode(X,INPUT); pinMode(X,INPUT); pinMode(X,INPUT); pinMode(X,INPUT);
pinMode(X,INPUT);
```

```
void loop()
```

```
while(digitalRead(C5)==0 && digitalRead(C4)==0 && digitalRead(C3)==0 && dig-
italRead(C2)==0 && digitalRead(C1)==0) // 0
```

```
while(digitalRead(C5)==0 && digitalRead(C4)==0 && digitalRead(C3)==0 && dig-
italRead(C2)==0 && digitalRead(C1)==1) // 1 Actuator Initialize
```

```
ActuatorInitialize();
```

```
while(digitalRead(C5)==0 && digitalRead(C4)==0 && digitalRead(C3)==0 && dig-
italRead(C2)==1 && digitalRead(C1)==0) // 2 Light
```

```
if(NoRepeat==0) digitalWrite(L, !digitalRead(L)); NoRepeat=1;
```

```
while(digitalRead(C5)==0 && digitalRead(C4)==0 && digitalRead(C3)==0 && dig-
italRead(C2)==1 && digitalRead(C1)==1) // 3 Horn
```

```
while(digitalRead(C5)==0 && digitalRead(C4)==0 && digitalRead(C3)==1 && dig-
italRead(C2)==0 && digitalRead(C1)==0) // 4 Vehicle Speed Decrement
```

```
if(NoRepeat==0) VehicleSpeed(2); NoRepeat=1;
```

```
while(digitalRead(C5)==0 && digitalRead(C4)==0 && digitalRead(C3)==1 && dig-
italRead(C2)==0 && digitalRead(C1)==1) // 5 Vehicle Speed increment
```

```
if(NoRepeat==0) VehicleSpeed(1); NoRepeat=1;
```

```
while(digitalRead(C5)==0 && digitalRead(C4)==0 && digitalRead(C3)==1 && dig-
italRead(C2)==1 && digitalRead(C1)==0) // 6 Reload
```

```
while(digitalRead(C5)==0 && digitalRead(C4)==0 && digitalRead(C3)==1 && dig-
```

```

digitalRead(C2)==1 && digitalRead(C1)==1) // 7 Fire
if(NoRepeat==0) Triger(); NoRepeat=1;
while(digitalRead(C5)==0 && digitalRead(C4)==1 && digitalRead(C3)==0 && dig-
italRead(C2)==0 && digitalRead(C1)==0) // 8 Laser

while(digitalRead(C5)==0 && digitalRead(C4)==1 && digitalRead(C3)==0 && dig-
italRead(C2)==0 && digitalRead(C1)==1) // 9 Actuator Speed Decrement
if(NoRepeat==0) StepperSpeed(2); NoRepeat=1; while(digitalRead(C5)==0 && digi-
talRead(C4)==1 && digitalRead(C3)==0 && digitalRead(C2)==1 && digitalRead(C1)==0)
// 10 Actuator Speed Increment
if(NoRepeat==0) StepperSpeed(1); NoRepeat=1;
while(digitalRead(C5)==0 && digitalRead(C4)==1 && digitalRead(C3)==0 && dig-
italRead(C2)==1 && digitalRead(C1)==1) // 11 Vehicle Up
VehicleForward(VehicleSpeedx);
while(digitalRead(C5)==0 && digitalRead(C4)==1 && digitalRead(C3)==1 && dig-
italRead(C2)==0 && digitalRead(C1)==0) // 12 Vehicle Right
VehicleRight(VehicleSpeedx);
while(digitalRead(C5)==0 && digitalRead(C4)==1 && digitalRead(C3)==1 && dig-
italRead(C2)==0 && digitalRead(C1)==1) // 13 Vehicle Down
VehicleBackward(VehicleSpeedx);
while(digitalRead(C5)==0 && digitalRead(C4)==1 && digitalRead(C3)==1 && dig-
italRead(C2)==1 && digitalRead(C1)==0) // 14 Vehicle Left
VehicleLeft(VehicleSpeedx);
while(digitalRead(C5)==0 && digitalRead(C4)==1 && digitalRead(C3)==1 && dig-
italRead(C2)==1 && digitalRead(C1)==1) // 15 Actuator Up
Up();
while(digitalRead(C5)==1 && digitalRead(C4)==0 && digitalRead(C3)==0 && dig-
italRead(C2)==0 && digitalRead(C1)==0) // 16 Actuatot Up Right
UpRight();
while(digitalRead(C5)==1 && digitalRead(C4)==0 && digitalRead(C3)==0 && dig-
italRead(C2)==0 && digitalRead(C1)==1) // 17 Actuator Right

```

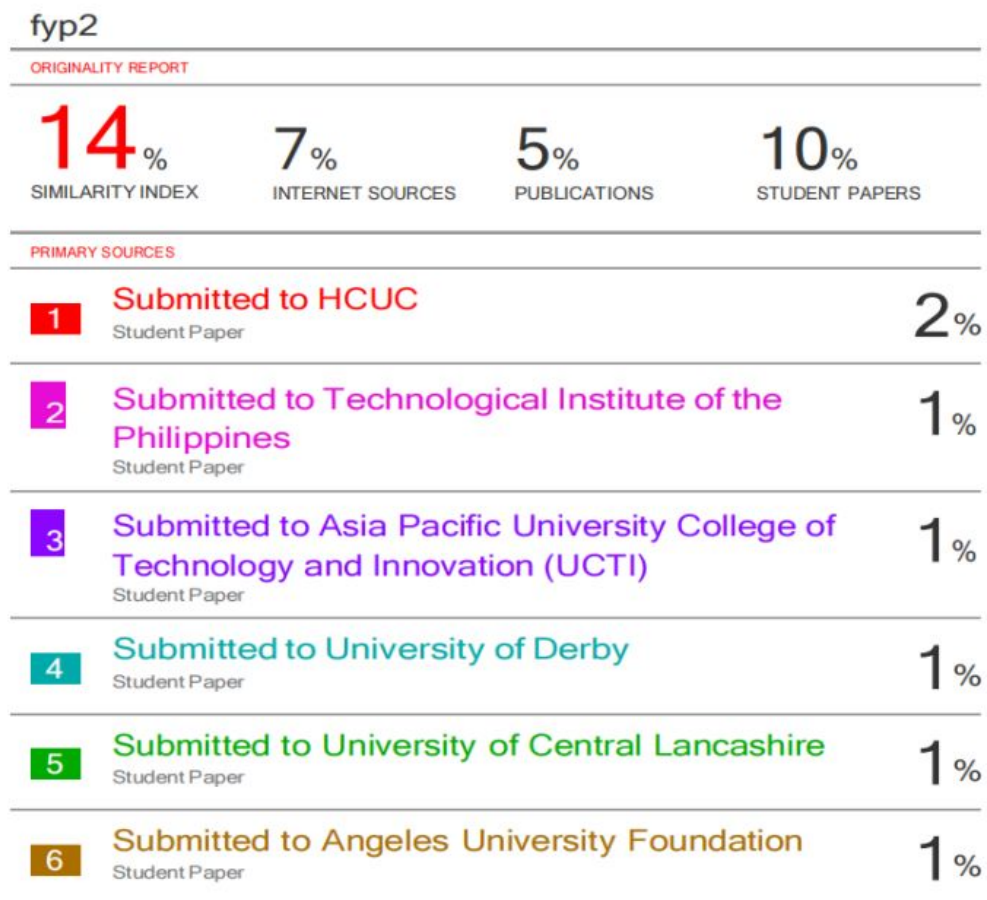
```

Right();
while(digitalRead(C5)==1 && digitalRead(C4)==0 && digitalRead(C3)==0 && dig-
italRead(C2)==1 && digitalRead(C1)==0) // 18 Actuator Down Right
DownRight();
while(digitalRead(C5)==1 && digitalRead(C4)==0 && digitalRead(C3)==0 && dig-
italRead(C2)==1 && digitalRead(C1)==1) // 19 Actuator Down rDown();
while(digitalRead(C5)==1 && digitalRead(C4)==0 && digitalRead(C3)==1 && dig-
italRead(C2)==0 && digitalRead(C1)==0) // 20 Actuator Down Left
DownLeft();
while(digitalRead(C5)==1 && digitalRead(C4)==0 && digitalRead(C3)==1 && dig-
italRead(C2)==0 && digitalRead(C1)==1) // 21 Actuator Left
Left();
while(digitalRead(C5)==1 && digitalRead(C4)==0 && digitalRead(C3)==1 && dig-
italRead(C2)==1 && digitalRead(C1)==0) // 22 Actuator Up Left
UpLeft();
while(digitalRead(C5)==1 && digitalRead(C4)==0 && digitalRead(C3)==1 && dig-
italRead(C2)==1 && digitalRead(C1)==1) // 23 Actuator ReAlign
ReAlign();
while(digitalRead(C5)==1 && digitalRead(C4)==1 && digitalRead(C3)==0 && dig-
italRead(C2)==0 && digitalRead(C1)==0) // 24

while(digitalRead(C5)==1 && digitalRead(C4)==1 && digitalRead(C3)==0 && dig-
italRead(C2)==0 && digitalRead(C1)==1) // 25

digitalWrite(M1A, LOW); digitalWrite(M1B,LOW); digitalWrite(M2A, LOW); digi-
talWrite(M2B, LOW);
NoRepeat = 0;

```



7	Submitted to Liverpool John Moores University Student Paper	1%
----------	---	-----------

8	Swagat Chutia, Lakhyajit Gohain, Nayan M. Kakoty, Dhanapati Dea. "Robotic Algae	1%
----------	--	-----------

Harvester: A Novel Method for Efficient Algae Collection", Aquacultural Engineering, 2022
Publication

9	hozx.pressureappliedcustoms.info Internet Source	1%
----------	--	-----------

10	www.labelektronika.com Internet Source	1%
-----------	--	-----------

11	Submitted to Universiti Teknikal Malaysia Melaka Student Paper	1%
-----------	--	-----------

12	Submitted to University of Balamand Student Paper	1%
-----------	---	-----------

13	Submitted to Birzeit University Main Library Student Paper	1%
14	Submitted to University of Huddersfield Student Paper	1%
15	Submitted to Durban University of Technology Student Paper	<1%
16	Submitted to Kingston University Student Paper	<1%
17	Submitted to University of Bolton Student Paper	<1%
18	www.seeedstudio.com Internet Source	<1%

Exclude quotes Off

Exclude bibliography On

Exclude assignment template On

Exclude matches Off