**ARDUINO BASED GESTURE CONTROLLED ROVER**

**A Project Report Submitted by**

|  |  |
| --- | --- |
| **RAVI SHANKAR YADAV**  **(122CS0058)** | **KRUTI KHEMRAJ LENGURE**  **(122EC0001)** |
| **ASHUTOSH MISHRA**  **(122CS0032)** | **HITESH BHOYAR**  **(122EC0021)** |
| **AYUSH MISHRA**  **(122CS0072)** | **ANJAN KUMAR SHARMA**  **(122EC0041)** |
| **ADITYA SHANKAR**  **(122CS0069)** | **ADITYA KUMAR RAI**  **(122ME0028)** |
| **RUBHAM SINGH**  **(122ME0024)** | **HARSHVARDHAN GIRI**  **(122ME0031)** |
| **SHUBHANKAR BHATTACHARYA**  **(122CS0047)** |  |

**In partial fulfillment of the requirement for the course of**

**ME105 -Design Realization**



**Department of Mechanical Engineering**

**Indian Institute of Information Technology Design and Manufacturing**

**Kurnool-518008, Andhra Pradesh (India)**

(An Institute of National Importance Funded by MoE, Govt. of India)

**June 2023**

**PROJECT APPROVAL CERTIFICATE**

This project entitled “**ARDUINO BASED GESTURE CONTROLLED ROVER**

” By Roll No’s. **122CS0072, 122CS0032, 122CS0058, 122CS0069, 122CS0047, 122EC0001, 122EC0021, 122EC0041, 122ME0024, 122ME0028, 122ME0031 of First Year Students of 2023 Batch, Indian Institute of Information Technology Design and Manufacturing, Kurnool is approved for the course of ME105 -Design Realization** for the degree in “**Bachelor of Technology**”.

|  |
| --- |
| Faculty In Charge  Dr. Vipindas K.  Assistant Professor, Department of Mechanical Engineering  Indian Institute of Information Technology Design and Manufacturing, Kurnool |
| Faculty In Charge  Dr. E. Hemachandran  Assistant Professor, Department of Mechanical Engineering  Indian Institute of Information Technology Design and Manufacturing, Kurnool |
| Faculty In Charge  Dr. Din Bandhu  Assistant Professor, Department of Mechanical Engineering  Indian Institute of Information Technology Design and Manufacturing, Kurnool |

Date: 3 June, 2023

Place: Indian Institute of Information Technology Design and Manufacturing, Kurnool

|  |
| --- |
| **ACKNOWLEDGEMENT**  We would like to express our profound gratitude and thanks to our project guide Dr. Vipindas, Assistant Professor, Head of Department of Mechanical Engineering, Indian Institute of Information Technology Design and Manufacturing, Kurnool for introducing the present topic and intellectual support, inspiring guidance and his invaluable encouragement, suggestions and cooperation helps a lot to completion of our project work successfully.  We also thankful to Dr. Hemachandran, Assistant Professor and Head of Department of Mechanical Engineering, Indian Institute of Information Technology Design and Manufacturing, Kurnool, for providing necessary information and guidance regarding the project.  Finally, we would like to thank IIITDM Kurnool for providing us this wonderful opportunity and all our friends for their kind co-operation which helps us to complete our project |
|  |

**ABSTRACT**

As we are living in a society where surveillance is needed every time e.g.: in the case of disasters, for the places where human cannot go and for spying. So, for this we came up with the idea of making a prototype of rover which could be controlled by a transmitter and would send us the live footage of the place where it is present. This is a wirelessly controlled rover which would be able to do surveillance. The system consists of two main components: a transmitter equipped with onboard gyroscope to recognize gestures and a rover integrated with an Arduino Mega Pro.

The rover is equipped with an Arduino Mega Pro, motor drivers, and ultrasonic sensors for obstacle avoidance and camera for surveillance. The rover's motors are controlled to move maneuvers omni-directionally based on either joystick or recognized gestures.

This project is to demonstrate the successful implementation of the Arduino-based rover control system with gesture recognition. The rover accurately responds to the recognized gestures, showcasing its reliability and effectiveness.

**CONTENTS**

|  |  |
| --- | --- |
| **Items** | **Page** |
|  |  |
| ***Title Sheet*** | *i* |
| ***Project Approval Certificate*** | *ii* |
| ***Abstract*** | *iii* |
| ***Contents*** | *iv* |
| ***List of Tables*** | *v* |
| ***List of Figures*** | *vi* |
| ***List of Abbreviations*** | *vii* |
|  |  |
| **CHAPTER 1: Introduction** | 09 |
| **CHAPTER 2: Hardware** |  |
| 1. Arduino Nano 2. Arduino Mega Pro | 10  13 |
| 1. Motor Driver | 14 |
| 2.4 Gyroscope Sensor  2.5 Adapter Board  2.6 Dupont Cables  2.7 OLED Display Module  2.8 Ultrasonic Sensor  2.9 GPS Module With EPROM  2.10 Charger Protection Module  2.11 Mecanum Wheel  2.12 Joystick Module  2.13 Acrylic | 15  16  17  17  18  19  20  20  20  21 |
| **CHAPTER 3: Software** |  |
| 3.1 Receiver Rover code  3.2 Transmitter Rover Code  3.3 ESP32 Camera Code | 24  30  36 |
|  |  |
| **CHAPTER 4: Design and Working** |  |
| 4.1 Circuit Design  4.2 Rover Body Making  4.3 How it Works?  **CHAPTER 5: Summary, Contribution and future scope** | 40  40  41 |
| 5.1 Summary  5.2 Future Scope | 44  44 |
|  |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Caption** | **Page** |
|  | ANALOG PINS | 13 |
|  | DIGITAL PINS | 14 |
|  | ATmega328 | 14 |
|  | BO MOTORS | 17 |
|  | GYROSCOPE SENSOR | 17 |
|  | ADAPTER BOARD | 18 |
|  | OLED DISPLAY | 18 |
|  | GPS MODULE | 19 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF FIGURES**

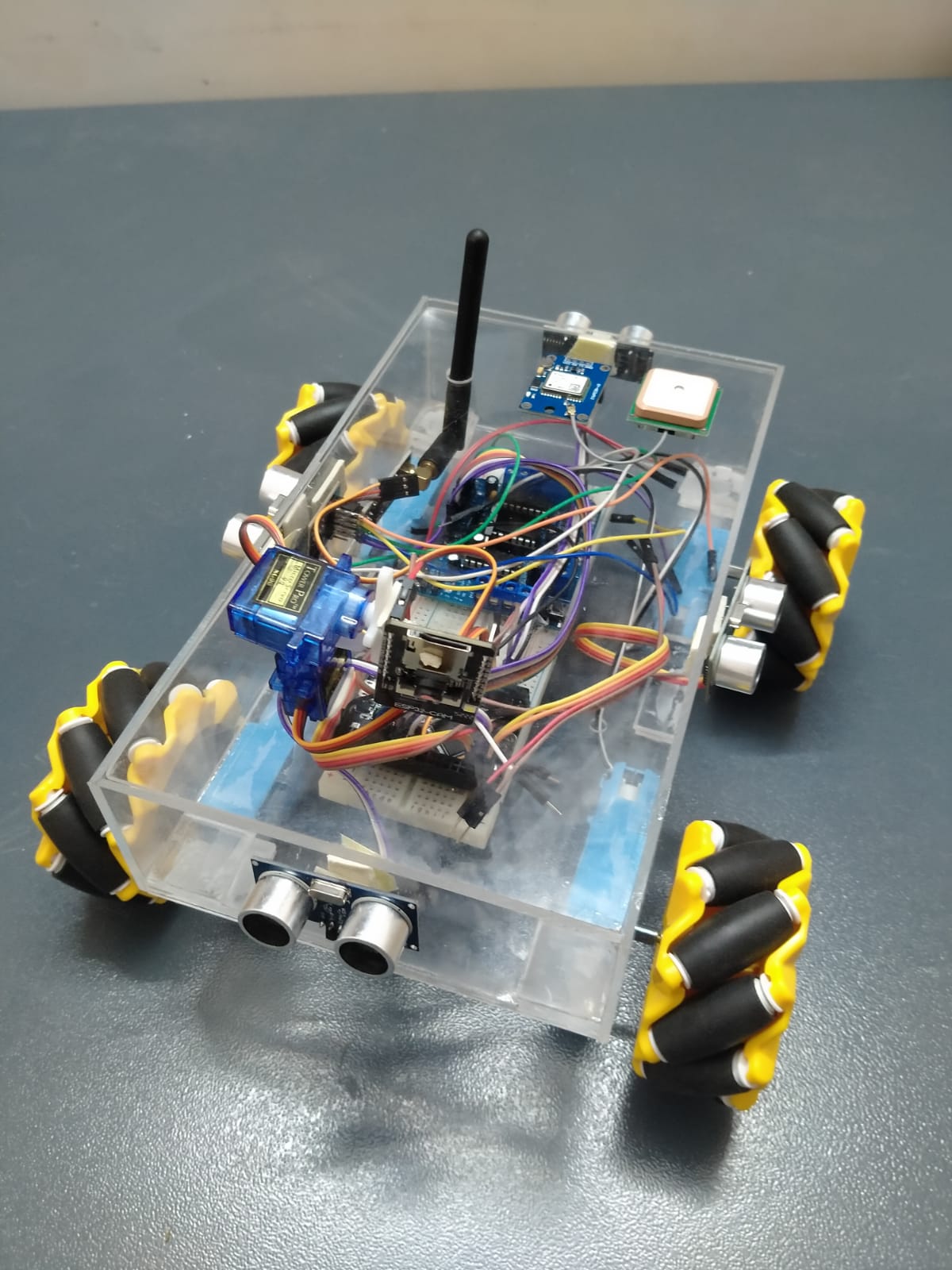
|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Caption** | **Page** |
|  | Block diagram of Arduino nano | 11 |
|  | Power tree of Arduino | 12 |
|  | Arduino mega pro | 14 |
|  | Motor driver | 15 |
|  | Servo motor | 15 |
|  | BO motor | 16 |
|  | Ultra sonic sensor | 19 |
|  | Mecanum Wheel | 20 |
|  | Joystick module | 21 |
|  | Acrylic | 22 |
|  | Rover receiver circuit diagram | 27 |
|  | Transmitter circuit diagram | 27 |
| 13 | Electric cutter | 28 |
| 14 | Electric drill machines | 29 |
| 15 | Flowchart | 31 |

**LIST OF ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
| BO | **:** | Battery operated |
| MIPS | **:** | Million Instruction per second |
| SRAM | **:** | Static Random Access Memory |
| EEPROM | **:** | Electrically Erasable Programmable Read Only Memory |
| PWM | **:** | Pulse Width Modulation |
| USART | **:** | Universal Synchronous/Asynchronous Receiver/Transmitter |
| SPI | **:** | Serial Peripheral Interfacee |
| USB | **:** | Universal Serial Bus |
| I/O | **:** | Input/Output |
| IDE | **:** | Integrated Development Environment |
| GPIO | **:** | General Purpose Input/Output |
| GND | **:** | Ground |
| VIN | **:** | Voltage Input |

**CHAPTER: - 1 INTRODUCTION**

Step into the world of robotic wonder with the Arduino-based Gesture Control Rover. Here mere hand gestures by the remote, controls a fascinating blend of technology and mobility. This state-of-the-art product allows you to easily control every rover movement through intuitive gestures and remote controls. Masterminded by an Arduino microcontroller and equipped with an array of sensors and actuators, this probe pushes the boundaries of traditional robotics and takes users into the realm of interactive exploration and engineering. This immersive rover combines gesture recognition technology with motor-controlled precision to deliver an exciting combination of innovation and play that lets your imagination run wild. Get ready for an adventure where the synergy of the Arduino's versatility, sensor module's sharp perception, and motor driver's rock-solid precision combine to create a fascinating and compelling robotic project like no other. Brace yourself for an inspiring journey into the future of interactive technology. There at your hands, you have the power to realize the extraordinary.



**CHAPTER: - 2 HARDWARE**

**2.1 Arduino Nano: -**

Arduino Nano is an intelligent development board designed for building faster prototypes with the smallest dimension. Arduino Nano being the oldest member of the Nano family, provides enough interfaces for your breadboard-friendly applications. At the heart of the board is ATmega328 microcontroller clocked at a frequency of 16 MHz featuring the same functionalities as the Arduino Duemilanove. The board offers 22 digital input/output pins, 8 analog pins, and a mini-USB port.

**Features:**

**ATmega328 Microcontroller**

* High-performance low-power 8-bit processor
* Achieve up to 16 MIPS for 16 MHz clock frequency
* 32 kB of which 2 KB used by bootloader
* 2 kB internal SRAM
* 1 kB EEPROM
* 32 x 8 General Purpose Working Registers
* Real Time Counter with Separate Oscillator
* Six PWM Channels
* Programmable Serial USART
* Master/Slave SPI Serial Interface

**Power**

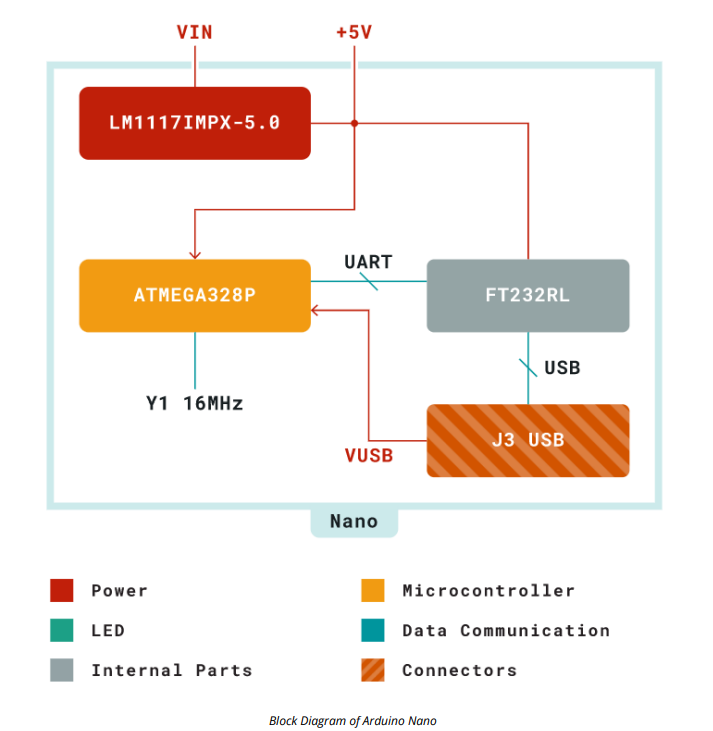
* Mini-B USB connection
* 6-20V unregulated external power supply (pin 30)
* 5V regulated external power supply (pin 27)

**I/O**

* 22 Digital
* 8 Analog
* 6 PWM Output

**Functional Overview**

* **Block diagram:**



**Board Operation**

**Getting Started – IDE**

If you want to program your Arduino Nano while offline you need to install the Arduino Desktop IDE. To connect the Arduino Uno to your computer, you’ll need a Micro-B USB cable. This also provides power to the board, as indicated by the LED.

**Getting Started** - Arduino Web Editor

All Arduino boards, including this one, work out-of-the-box on the Arduino Web Editor, by just installing a simple plugin. The Arduino Web Editor is hosted online, therefore it will always be up-to-date with the latest features and support for all boards. Follow to start coding on the browser and upload your sketches onto your board.

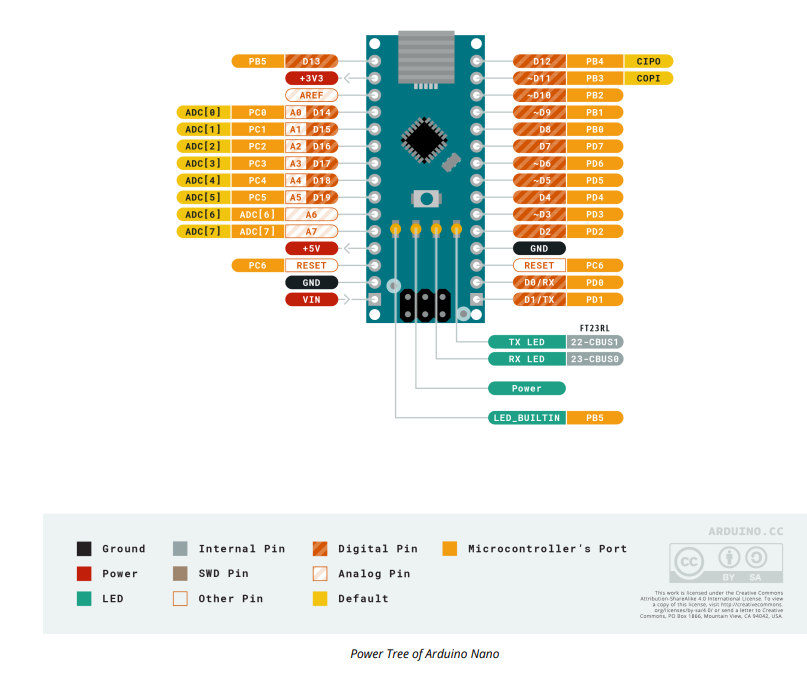
**Sample Sketches** -

Sample sketches for the Arduino can be found either in the “Examples” menu in the Arduino IDE or in the “Documentation” section of the Arduino website.

**Online Resources** -

Now that you have gone through the basics of what you can do with the board you can explore the endless possibilities it provides by checking exciting projects on Project Hub , the Arduino Library Reference and the online store where you will be able to complement your board with sensors, actuators and more.

* **Connector Pinouts**



**Analog Pins:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin** | **Function** | **Type** | **Description** |
| 1 | +3V3 | Power | 5V USB Power |
| 2 | A0 | Analog | Analog input 0 /GPIO |
| 3 | A1 | Analog | Analog input 1 /GPIO |
| 4 | A2 | Analog | Analog input 2 /GPIO |
| 5 | A3 | Analog | Analog input 3 /GPIO |
| 6 | A4 | Analog | Analog input 4 /GPIO |
| 7 | A5 | Analog | Analog input 5 /GPIO |
| 8 | A6 | Analog | Analog input 6 /GPIO |
| 9 | A7 | Analog | Analog input 7 /GPIO |
| 10 | +5V | Power | +5 V Power Rail |
| 11 | Reset | Reset | Reset |
| 12 | GND | Power | Ground |
| 12 | VIN | Power | Voltage Input |

**DIGITAL PINS**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin** | **Function** | **Type** | **Description** |
| 1 | D1/TX1 | Digital | Digital Input 1/GPIO |
| 2 | D0/RX0 | Digital | Digital Input 0/GPIO |
| 3 | D2 | Digital | Digital Input 2/GPIO |
| 4 | D3 | Digital | Digital Input 3/GPIO |
| 5 | D4 | Digital | Digital Input 4/GPIO |
| 6 | D5 | Digital | Digital Input 5/GPIO |
| 7 | D6 | Digital | Digital Input 6/GPIO |
| 8 | D7 | Digital | Digital Input 7/GPIO |
| 9 | D8 | Digital | Digital Input 8/GPIO |
| 10 | D9 | Digital | Digital Input 9/GPIO |
| 11 | D10 | Digital | Digital Input 10/GPIO |
| 12 | D11 | Digital | Digital Input 11/GPIO |
| 13 | D12 | Digital | Digital Input 12/GPIO |
| 14 | D13 | Digital | Digital Input 13/GPIO |
| 15 | Reset | Reset | Reset |
| 16 | GND | Power | Power |

**ATmega328:**

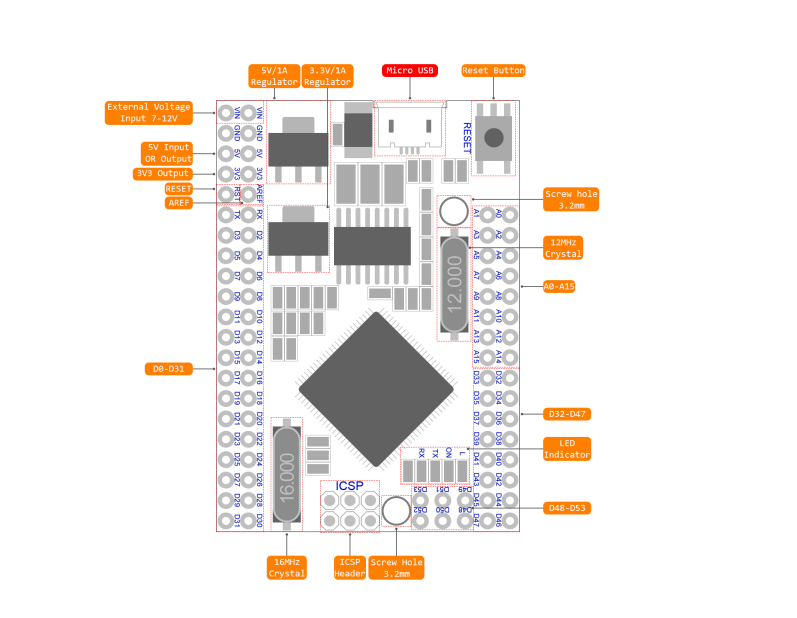
|  |  |  |  |
| --- | --- | --- | --- |
| **Pin** | **Function** | **Type** | **Description** |
| 1 | PB0 | Internal | Serial Wire Debug |
| 2 | PB1 | Internal | Serial Wire Debug |
| 3 | PB2 | Internal | Serial Wire Debug |
| 4 | PB3 | Internal | Serial Wire Debug |
| 5 | PB4 | Internal | Serial Wire Debug |
| 6 | PB5 | Internal | Serial Wire Debug |

**2.2 Arduino mega pro: -**

The Mega Pro Embed CH340G / ATmega2560 board is based on the ATmega2560 microcontroller. Built on the Atmel ATmega2560 microcontroller and USB-UART interface chip CH340G. This is a great solution to get your final project on the welding prototype board.

The Mega Pro Embed functionalities are identical to the Mega 2560. Even though it is an Embed board, it is solid as stable as the Mega Board. It uses the original chip, and 16 MHz high-quality quartz resonators are present on the board. The board uses the chip CH340G as the converter UART-USB. It provides a stable data exchange result when you work at a frequency of 12Mhz (requires the driver to be installed on the computer).

We can power the board or power the pins through the MicroUSB connector. The voltage regulator can handle input voltages from 6V to 9V (peak 18V) DC. The voltage regulator allows to use up to 18V as an input value, however, we do not recommend exceeding the recommended values, since this can lead to excessive heat and damage the device.

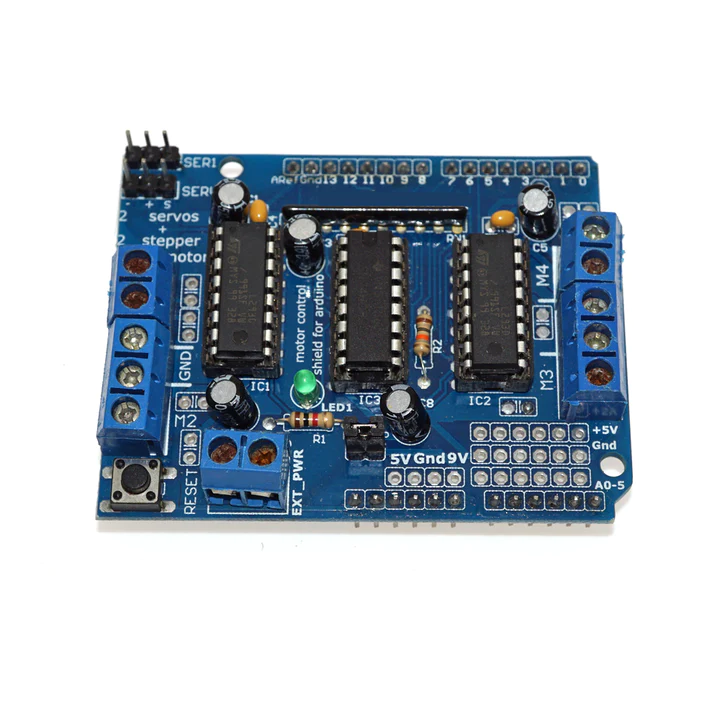


**2.3 Motor Driver: -**

The L293D Motor Driver/Servo Shield for Arduino is probably one of the most versatile on the market and features 2 servo and 4 motor connectors for DC or stepper motors. That makes it a great shield for any robotic project.

This Arduino compatible motor Driver shield is a full-featured product that it can be used to drive 4 DC motor or two 4-wire steppers and two 5v servos. It drives the DC motor and stepper with the L293D, and it drives the servo with Arduino pin9 and pin10.

The shield contains two L293D motor drivers and one 74HC595 shift register. The shift register expands 3 pins of the Arduino to 8 pins to control the direction of the motor drivers. The output enables the L293D is directly connected to the PWM outputs of the Arduino. The two types of motors used in this project are: -



**2.3.1 Servo Motor: -**

A **servo motor** is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. If you want to rotate an object at some specific angles or distance, then you use a servo motor. It is just made up of a simple motor which runs through a **servo mechanism**. If motor is powered by a DC power supply, then it is called DC servo motor, and if it is AC-powered motor then it is called AC servo motor. For this tutorial, we will be discussing only about the **DC servo motor working**. Apart from these major classifications, there are many other types of servo motors based on the type of gear arrangement and operating characteristics. A servo motor usually comes with a gear arrangement that allows us to get a very high torque servo motor in small and lightweight packages



**2.3.2 BO Motor: -**

It is a BO Series 1 100RPM DC Motor Plastic Gear Motor. The BO series straight motor gives good torque and rpm at lower operating voltages, which is the biggest advantage of these motors. A small shaft with matching wheels gives an optimized design for your application or robot. Mounting holes on the body & lightweight makes it suitable for in-circuit placement. This motor can be used with 69mm Diameter Wheel for Plastic Gear Motors and 87mm Diameter Multipurpose Wheel for Plastic Gear Motors.

|  |  |
| --- | --- |
| **Operating Voltage** | 3 ~ 12 |
| **Shaft Length (mm)** | 8.5 |
| **Shaft Diameter (mm)** | (Double D-type) 5.5 |
| **No Load Current (mA)** | 40-180mA. |
| **Rated Speed After Reduction (RPM)** | 100 |
| **Rated Torque (Kg-Cm)** | 1 |
| **Weight (gm)** | 30 |
| **Dimensions in mm (LxWxH)** | 64 x 22 x 18 |
| **Shipment Weight** | 0.033 kg |



**2.4 Gyroscope Sensor: -**

The **MPU-605**0 3-Axis Accelerometer and Gyro Sensor module use **MPU-6050** which is a little piece of motion processing tech. The **MPU6050** devices combine a 3-axis gyroscope and a 3-axis accelerometer on the same silicon together with an onboard Digital Motion Processor (DMP) capable of processing complex 9-axis MotionFusion algorithms.

|  |  |
| --- | --- |
| **Driver IC** | MPU-6050 |
| **Operating Voltage** | 3 ~ 5 |
| **Communication** | I2C Protocol |
| **Gyro range(Â°/s)** | Â± 250, 500, 1000, 2000 |
| **Acceleration range(g)** | Â± 2 Â± 4 Â± 8 Â± 16 |
| **Length (mm)** | 20 |
| **Width (mm)** | 16 |
|  |  |

**2.5 Adapter Board: -**

3.3V VCC Adapter Board for NRF24L01 Wireless Transceiver Module enables you to use NRF24l01 type transceivers on 5V systems like Arduino. It regulates the 3.3V input to 1.9~3.6V DC and incorporates bypass capacitors for reliable operation. The NRF24L01 utilizes Enhanced ShockBurst (ESB) protocol to support two-way data packet communication with packet buffering, packet acknowledgement and automatic re-transmission of lost packets.

|  |  |
| --- | --- |
| **On-board chip** | AMS1117-3.3 |
| **Input Voltage** | 5V DC |
| **Output Voltage** | 1.9~3.6 DC |
| **Operating Current** | 12 mA |

**2.6 OLED Display: -**

0.96” I2C OLED Display is an OLED monochrome 128×64 dot matrix display module with I2C Interface. It is perfect when you need an ultra-small display. Comparing to LCD, OLED screens are way more competitive, which has several advantages such as high brightness, self-emission, high contrast ratio, slim outline, wide viewing angle, wide temperature range, and low power consumption. It is compatible with any 3.3V-5V microcontroller, such as Arduino.

|  |  |
| --- | --- |
| **Resolution** | 128 x 64 Pixels |
| **Display Area** | 21.74 x 10.86mm |
| **Driving Voltage** | 3.3-5V |
| **Operating Temperature** | -40º~70º Celsius |
| **Interface Type** | IIC |
| **Pixel Color** | White |

**2.7 Ultrasonic Sensor: -**

HC-SR04-Ultrasonic Range Finder is a very popular sensor that is found in many applications where it requires measuring distance and detecting objects.

The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The HC-SR04 ultrasonic sensor uses sonar to determine the distance to an object like bats or dolphins do.

This Ultrasonic Sensor module is a transmitter, a receiver, and a control circuit in one single pack!! It has very handy and compact construction. It offers excellent range accuracy and stable readings in an easy-to-use package. Its operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect).

The Trigger and the Echo pins are the I/O pins of this module and hence they can be connected to the I/O pins of the microcontroller/Arduino. When the receiver detects the return wave the Echo pin goes high for a particular amount of time which will be equal to the time taken for the wave to return to the sensor.

Ultrasonic Ranging Module HC-SR04 provides 2cm-400cm non-contact distance sensing capabilities, Ranging accuracy up to 3mm.

This Ultrasonic Sensor can be attached to your project using a mounting bracket, so buy it now at Robu.in we have a very good quality Acrylic Mounting Bracket for this HC-SR04 Ultrasonic Module.

[[1]](#footnote-2)

**2.8 GPS Module with EPROM: -**

GPS module is based on the NEO 6M GPS. This unit uses the latest technology to give the best possible positioning information and includes a larger built-in 25 x 25mm active GPS antenna with a UART TTL socket. A battery is also included so that you can obtain a GPS lock faster. This is an updated GPS module that can be used with Ardupilot mega v2. This GPS module gives the best possible position information, allowing for better performance with your Ardupilot or other Multirotor control platform. The GPS module has serial TTL output, it has four pins: TX, RX, VCC, and GND.

|  |  |
| --- | --- |
| **Model** | Ublox NEO-6M |
| **Receiver Type** | 50 Channels GPS L1 frequency, C/A Code SBAS: WAAS, EGNOS, MSAS |
| **Input Supply Voltage** | 2.7 ~ 6 |
| **Main Chip** | NEO-6 |
| **Sensitivity (dBm)** | -160 156 Cold Start (without aiding): -147 dBm Tracking & Navigation: -161 dBm |
| **Navigation Update Rate** | 5Hz |
| **Position Accuracy (Meter)** | 2 |

**2.9 Charger Protection Module: -**

TP4056 1A Li-Ion Battery Charging Board Type C with Current Protection is a tiny module, perfect for charging single cell 3.7V 1 Ah or higher lithium-ion (Li-Ion) cells such as 16550s that don’t have their own protection circuit. Based on the TP4056 charger IC and DW01 battery protection IC this module will offer 1A charge current then cut off when finished. Furthermore, when the battery voltage drops below 2.4V the protection IC will switch the load off to protect the cell from running at too low of a voltage – and it protects against over-voltage and reverse polarity connection (it will usually destroy itself instead of the battery)

**2.10 Mecanum Wheel: -**

The **mecanum wheel** is an omnidirectional wheel design for a land-based vehicle to move in any direction. The mecanum wheel is a form of tireless wheel, with a series of rubberized external rollers obliquely attached to the whole circumference of its rim. These rollers typically each have an axis of rotation at 45° to the wheel plane and at 45° to the axle line. Each Mecanum wheel is an independent non-steering drive wheel with its own powertrain, and when spinning generates a propelling force perpendicular to the roller axle, which can be vectored into a longitudinal and a transverse component in relation to the vehicle.



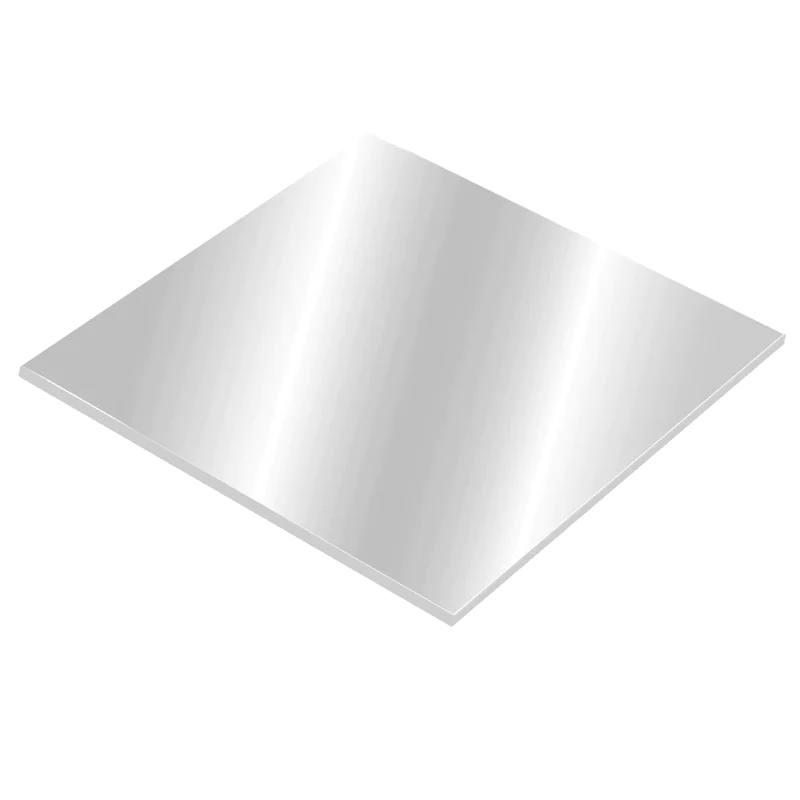
**2.12 Joystick Module: -**

Joystick Module PS2 Breakout Sensor very similar to the ‘analog’ joysticks on PS2 (PlayStation 2) controllers. Directional movements are simply two potentiometers – one for each axis. Pots are ~10k each. This joystick also has a select button that is actuated when the joystick is press down. With the help of this Joystick Module, you can measure position coordinates on the X and Y axis by moving the “hat”. It also contains a switch that is press-able by pushing the “hat". It also contains a switch that is press-able by pushing the “hat” down. Similar to the XBOX controller. The X and Y axes are two 10k potentiometers which control 2D movement by generating analog signals. When the module is in working mode, it will output two analog values, representing two directions. This module uses the 5V power supply, and value, when reading through analog input, would be about 2.5V, a value will increase with joystick movement and will go up till maximum 5V; the value will decrease when the joystick is moved in other direction till 0V.



**2.13 Acrylic sheet: -**

Acrylic is a plastic material with outstanding strength, stiffness, and optical clarity. Acrylic sheet is easy to fabricate, bond well with adhesives and solvents, and is easy to thermoform. It has superior weathering properties compared to many other transparent plastics. Cast acrylic sheet is much more flexible than glass or many other building materials. When using large sheets for windows, it is important that rabbets or channels be deep enough to provide support against high winds. Colorless Cast acrylic sheet has a light transmittance of 92%. It is clearer than window glass and will not turn yellow. Cast acrylic sheet is also available in a large variety of transparent and translucent colors. Shatter-resistant, earthquake-safe, and burglar-resistant. Increase safety with windows glazed of acrylic. Acrylic, also known as Plexiglass, is a versatile plastic material with a variety of purposes and benefits, available in a spectrum of colors and opacities. High impact resistance High optical clarity Innate weather ability and UV resistance



**CHAPTER 3 : - SOFTWARE**

**3.1 Receiver Rover Code: -**

#include <SPI.h> // FOR SPI COMMUNICATION OF NRF24L01 WITH ARDUINO

#include <nRF24L01.h> // FOR NRF24L01

#include <RF24.h> // FOR NRF24L01

#include <AFMotor.h> // FOR MOTORS

#include <TinyGPS++.h> // FOR GPS

#include <Servo.h> // FOR SERVO

#include <NewPing.h> // FOR ULTRASONIC SENSORS

RF24 radio(48, 49); // nRF24L01 (CE, CSN)

const byte addresses[][6] = {"00001", "00002"}; //PIPE ADDRESS

unsigned long time0 = 0;

unsigned long time1 = 0;

#define toggle 16

int flag\_front=1 , flag\_back=1 , flag\_left=1 , flag\_right=1;

AF\_DCMotor motor1(1); // FOR MOTORS

AF\_DCMotor motor2(2);

AF\_DCMotor motor3(3);

AF\_DCMotor motor4(4);

Servo servoX; // CREATE SERVO USING THIS FUNCTION

Servo servoY;

int angleX = 180;

int angleY = 180;

#define neogps Serial1 // SET SERIAL PORT1 AS GPS RECEIVING PORT

TinyGPSPlus gps; // TINYGPS set gps as function

#define maxdist 200 // FOR ULTRASONIC SENSORS

NewPing sonar\_front (22 , 23 , maxdist); // Ultrasonic sensor trig, echo, maxdist

NewPing sonar\_back (24 , 25 , maxdist);

NewPing sonar\_left (26 , 27 , maxdist);

NewPing sonar\_right (28 , 29 , maxdist);

struct data\_package // STRUCTURE FOR CONTROLLER

{

byte joy1X;

byte joy1Y;

byte j1button;

byte joy2X;

byte joy2Y;

byte j2button;

byte toggle1;

};

data\_package data; // VARIABLE IS data

struct gps\_package // STRUCTURE FOR GPS DATA

{

float latitude;

float longitude;

float altitude;

float speed;

byte satellite;

byte abc;

};

gps\_package gpsdata; // VARIABLE IS gpsdata

void setup()

{

Serial.begin(9600);

neogps.begin(9600); // BEGIN SERIAL1 COMMUNICATION WITH Neo6mGPS

gpsdata.abc=0;

radio.begin(); // SET RADIO

radio.openWritingPipe( addresses[0]); // 00001

radio.openReadingPipe(1, addresses[1]); // 00002

radio.setAutoAck(false);

radio.setDataRate(RF24\_250KBPS);

radio.setPALevel(RF24\_PA\_LOW);

pinMode(toggle, INPUT\_PULLUP);

motor1.setSpeed(255); // SET MOTOR SPEED

motor2.setSpeed(255);

motor3.setSpeed(255);

motor4.setSpeed(255);

motor1.run(RELEASE); // SET MOTOR TO RELEASE(STOP)

motor2.run(RELEASE);

motor3.run(RELEASE);

motor4.run(RELEASE);

servoX.attach(9); // ATTACH SERVO TO PINS

servoY.attach(10);

resetdata();

}

void loop()

{

radio.startListening(); // SET AS RECEIVER

if (radio.available())

{

radio.read(&data, sizeof(data\_package));

time0 = millis();

}

time1 = millis();

if ( time1 - time0 > 1000 )

{ resetdata(); }

serial\_print\_data(); // SERIAL PRINT RECEIVED DATA

int toggle\_switch = digitalRead(toggle);

if( toggle\_switch == 0 )

{

if( sonar\_front.ping\_cm() < 40 )

flag\_front = 0;

if( sonar\_back.ping\_cm () < 40 )

flag\_back = 0;

if( sonar\_left.ping\_cm () < 40 )

flag\_left = 0;

if( sonar\_right.ping\_cm() < 40 )

flag\_right = 0;

}

// MOVEMENTS OF THE ROVER

if (data.joy1Y > 150 && flag\_front == 1 ) // FORWARD

{

motor1.run(FORWARD);

motor2.run(FORWARD);

motor3.run(FORWARD);

motor4.run(FORWARD);

}

else if (data.joy1Y < 104 && flag\_back == 1 ) // BACKWARD

{

motor1.run(BACKWARD);

motor2.run(BACKWARD);

motor3.run(BACKWARD);

motor4.run(BACKWARD);

}

else if (data.joy1X > 150 && flag\_left == 1 ) // LEFT

{

motor1.run(BACKWARD);

motor2.run(FORWARD);

motor3.run(FORWARD);

motor4.run(BACKWARD);

}

else if (data.joy1X < 104 && flag\_right == 1 ) // RIGHT

{

motor1.run(FORWARD);

motor2.run(BACKWARD);

motor3.run(BACKWARD);

motor4.run(FORWARD);

}

else // STOP MOVING

release();

if ( data.joy1Y > 150 && data.joy1X > 150 && flag\_front == 1 && flag\_left == 1 ) // FORWARD LEFT

{

motor1.run(RELEASE);

motor2.run(FORWARD);

motor3.run(FORWARD);

motor4.run(RELEASE);

}

else if (data.joy1Y > 150 && data.joy1X < 104 && flag\_front == 1 && flag\_right == 1 ) // FORWARD RIGHT

{

motor1.run(FORWARD);

motor2.run(RELEASE);

motor3.run(RELEASE);

motor4.run(FORWARD);

}

else if (data.joy1Y < 104 && data.joy1X > 150 && flag\_back == 1 && flag\_left == 1 ) // BACKWARD LEFT

{

motor1.run(BACKWARD);

motor2.run(RELEASE);

motor3.run(RELEASE);

motor4.run(BACKWARD);

}

else if (data.joy1Y < 104 && data.joy1X < 104 && flag\_back == 1 && flag\_right == 1 ) // BACKWARD RIGHT

{

motor1.run(RELEASE);

motor2.run(BACKWARD);

motor3.run(BACKWARD);

motor4.run(RELEASE);

}

else //STOP MOVING

release();

if (data.j1button == 1) // ROTATE LEFT OR ANTI-CLOCKWISE

{

motor1.run(BACKWARD);

motor2.run(FORWARD);

motor3.run(BACKWARD);

motor4.run(FORWARD);

}

else if (data.j2button == 1) // ROTATE RIGHT OR CLOCKWISE

{

motor1.run(FORWARD);

motor2.run(BACKWARD);

motor3.run(FORWARD);

motor4.run(BACKWARD);

}

else //STOP MOVING

release();

flag\_front = 1;

flag\_back = 1;

flag\_left = 1;

flag\_right = 1;

//

// MOVEMENTS OF THE SERVO // SERVO MOVE UNTIL THE JOYSTICK IS MOVED THEN STOPS WHEN JOYSTICK IS RELEASED

if (data.joy2X > 150 && angleX < 360) // SERVO X AXIS

{

angleX = angleX + 1;

servoX.write(angleX);

}

if(data.joy2X < 104 && angleX > 0 )

{

angleX = angleX - 1;

servoX.write(angleX);

}

if(data.joy2Y > 150 && angleY < 360) // SERVO Y AXIS

{

angleY = angleY + 1;

servoY.write(angleY);

}

if(data.joy2Y < 104 && angleY > 0)

{

angleY = angleY - 1;

servoY.write(angleY);

}

//

delay(5);

radio.stopListening(); // SET AS TRANSMITTER

while (neogps.available() > 0) // GET GPS DATA

if (gps.encode(neogps.read()))

{

if ( gps.location.isValid() == 1 )

{

gpsdata.latitude = gps.location.lat();

gpsdata.longitude = gps.location.lng();

gpsdata.altitude = gps.altitude.meters();

gpsdata.speed = gps.speed.kmph();

gpsdata.satellite = gps.satellites.value();

}

//else

//{ gpsdata.abc = 1; }

}

gpsdata.abc = 1;

radio.write( &gpsdata, sizeof(gps\_package) );

delay(5);

}

void release()

{

motor1.run(RELEASE);

motor2.run(RELEASE);

motor3.run(RELEASE);

motor4.run(RELEASE);

}

void serial\_print\_data()

{

Serial.print("joy1 = ");

Serial.print(data.joy1X);

Serial.print(" | ");

Serial.print(data.joy1Y);

Serial.print(" | ");

Serial.print(data.j1button);

Serial.print(" | ");

Serial.print("joy2 = ");

Serial.print(data.joy2X);

Serial.print(" | ");

Serial.print(data.joy2Y);

Serial.print(" | ");

Serial.print(data.j2button);

Serial.print(" | ");

Serial.print("toggle = ");

Serial.println(data.toggle1);

}

void resetdata() // RESET DATA PACAKAGE OF TRANSMITTER

{

data.joy1X = 127;

data.joy1Y = 127;

data.j1button = 1;

data.joy2X = 127;

data.joy2X = 127;

data.j2button = 1;

data.toggle1 = 1;

}

**3.2 Transmitter Rover Code: -**

/\*

TRANSMITTER CODE

\*/

#include <SPI.h> // FOR SPI COMMUNICATION OF NRF24L01 WITH ARDUINO

#include <nRF24L01.h> // FOR NRF24L01

#include <RF24.h> // FOR NRF24L01

#include <Wire.h> // FOR I2C COMMUNICATION with OLED AND ACCELEROMETER

#include <Adafruit\_SH1106.h> // FOR OLED

#include <Adafruit\_GFX.h> // FOR OLED

RF24 radio(9, 10); // nRF24L01 (CE, CSN)

const byte addresses[][6] = {"00001", "00002"}; //PIPE ADDRESS

#define j1 4 // JOYSTICK BUTTON

#define j2 5

#define t1 7 // TOGGLE SWITCH

const int MPU = 0x68; // MPU6050 I2C ADDRESS

float AccX, AccY, AccZ;

float GyroX, GyroY, GyroZ;

float accAngleX, accAngleY, gyroAngleX, gyroAngleY;

float angleX, angleY;

float elapsedTime, currentTime, previousTime;

int c = 0;

#define OLED\_RESET -1 // ADAFRUIT OLED

Adafruit\_SH1106 display(OLED\_RESET);

unsigned long time0 = 0;

unsigned long time1 = 0;

struct data\_package // MAX SIZE OF THIS STRUCK IS 32 BYTES - NRF24L01 BUFFER LIMIT

{

byte joy1X;

byte joy1Y;

byte j1button;

byte joy2X;

byte joy2Y;

byte j2button;

byte toggle1;

};

data\_package data; // VARIABLE IS data

struct gps\_package // STRUCTURE FOR GPS DATA MAX 32 BYTES

{

float latitude;

float longitude;

float altitude;

float speed;

byte satellite;

byte abc;

};

gps\_package gpsdata; // VARIABLE IS gpsdata

void setup()

{

Serial.begin(9600);

initialize\_MPU6050(); // Initialize interface to the MPU6050

radio.begin(); // SET RADIO

radio.openWritingPipe(addresses[1]); // 00002

radio.openReadingPipe(1, addresses[0]); // 00001

radio.setAutoAck(false);

radio.setDataRate(RF24\_250KBPS);

radio.setPALevel(RF24\_PA\_LOW);

pinMode(j1, INPUT\_PULLUP); // Activate the Arduino internal pull-up resistors

pinMode(j2, INPUT\_PULLUP);

pinMode(t1, INPUT\_PULLUP);

set\_default(); // SET INITIAL DEFAULT VATUES OF INPUTS

display.begin(SH1106\_SWITCHCAPVCC, 0x3C); // INITIALIZE DISPLAY WITH ADDRESS

display.clearDisplay();

display.setTextColor(WHITE);

display.setTextSize(1);

display.setCursor(0,0);

display.println("hello");

display.println("world");

display.display();

}

void loop()

{

radio.stopListening(); // SET AS TRANSMITTER

// READ ANALOG INPUTS

data.joy1X = map(analogRead(A0), 1023, 0, 0, 255); // Convert the analog read value from 0 to 1023 into a BYTE value from 0 to 255

data.joy1Y = map(analogRead(A1), 1023, 0, 0, 255);

data.joy2X = map(analogRead(A2), 0, 1023, 0, 255);

data.joy2Y = map(analogRead(A3), 0, 1023, 0, 255);

// READ DIGITAL INPUTS

data.j1button = digitalRead(j1);

data.j2button = digitalRead(j2);

data.toggle1 = digitalRead(t1);

if (digitalRead(t1) == 0)

{ read\_IMU(); }

radio.write( &data, sizeof(data\_package) ); // Send the data from the structure to the receiver

delay(5);

radio.startListening(); // SET AS RECEIVER FOR OLED

if ( radio.available() )

{

radio.read( &gpsdata, sizeof(gps\_package) );

time0 = millis();

}

time1 = millis();

Serial.print(data.joy1X);

Serial.print(" | ");

Serial.print(data.joy1Y);

Serial.print(" | ");

Serial.print(data.joy2X);

Serial.print(" | ");

Serial.println(data.joy2Y);

/\*

// SERIAL PRINT GPS RECEIVED DATA

Serial.print("LATI: ");

Serial.print( gpsdata.latitude );

Serial.print(" | ");

Serial.print("LONG: ");

Serial.print( gpsdata.longitude );

Serial.print(" | ");

Serial.print("ALTI: ");

Serial.print( gpsdata.altitude );

Serial.print(" | ");

Serial.print("Speed: ");

Serial.print( gpsdata.speed );

Serial.print("Kmph");

Serial.print(" | ");

Serial.print("SAT: ");

Serial.print( gpsdata.satellite );

Serial.print(" | ");

Serial.print("receiver data: ");

Serial.println( gpsdata.abc );

//

\*/

if ( time1 - time0 < 30000 )

{

display.clearDisplay(); // DISPLAY CO-ORDINATES

display.setTextSize(1);

display.setCursor(0, 0);

display.println("GPS LOCATION");

display.print("LATI: "); // LATITUDE

display.println(gpsdata.latitude, 6 );

display.print("LONG: "); // LONGITUDE

display.println(gpsdata.longitude, 6 );

display.print("ALTI:"); // ALTITUDE

display.println(gpsdata.altitude, 3 );

display.print("Speed:"); // SPEED KMPH

display.print(gpsdata.speed, 3 );

display.println(" Kmph");

display.print("SAT : "); // NUM OF SATELLITE TRACKING

display.println(gpsdata.satellite );

display.display();

}

else

{

display.clearDisplay();

display.setTextSize(2);

display.setCursor(0,0);

display.println("GPS");

display.println("CONNECTION");

display.println("LOST");

display.display();

}

/\*

if( gpsdata.abc==1 ) // IF GPS MODULE IS NOT GETTING DATA

{

display.clearDisplay();

display.setCursor(0, 0);

display.setTextSize(1);

display.println("GPS");

display.println("INITIALIZING");

display.display();

}

gpsdata.abc=0;

\*/

gpsdata.abc = 0; // receiver data check

delay(5);

}

void set\_default()

{

data.joy1X = 127;

data.joy1Y = 127;

data.j1button = 1;

data.joy2X = 127;

data.joy2Y = 127;

data.j2button = 1;

data.toggle1 = 1;

}

void initialize\_MPU6050()

{

Wire.begin(); // Initialize comunication

Wire.beginTransmission(MPU); // Start communication with MPU6050 // MPU=0x68

Wire.write(0x6B); // Talk to the register 6B

Wire.write(0x00); // Make reset - place a 0 into the 6B register

Wire.endTransmission(true); //end the transmission

// Configure Accelerometer

Wire.beginTransmission(MPU);

Wire.write(0x1C); //Talk to the ACCEL\_CONFIG register

Wire.write(0x10); //Set the register bits as 00010000 (+/- 8g full scale range)

Wire.endTransmission(true);

// Configure Gyro

Wire.beginTransmission(MPU);

Wire.write(0x1B); // Talk to the GYRO\_CONFIG register (1B hex)

Wire.write(0x10); // Set the register bits as 00010000 (1000dps full scale)

Wire.endTransmission(true);

}

void read\_IMU()

{

// === Read acceleromter data === //

Wire.beginTransmission(MPU);

Wire.write(0x3B); // Start with register 0x3B (ACCEL\_XOUT\_H)

Wire.endTransmission(false);

Wire.requestFrom(MPU, 6, true); // Read 6 registers total, each axis value is stored in 2 registers

//For a range of +-8g, we need to divide the raw values by 4096, according to the datasheet

AccX = (Wire.read() << 8 | Wire.read()) / 4096.0; // X-axis value

AccY = (Wire.read() << 8 | Wire.read()) / 4096.0; // Y-axis value

AccZ = (Wire.read() << 8 | Wire.read()) / 4096.0; // Z-axis value

// Calculating angle values using

accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) \* 180 / PI) + 1.15; // AccErrorX ~(-1.15) See the calculate\_IMU\_error()custom function for more details

accAngleY = (atan(-1 \* AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) \* 180 / PI) - 0.52; // AccErrorX ~(0.5)

// === Read gyro data === //

previousTime = currentTime; // Previous time is stored before the actual time read

currentTime = millis(); // Current time actual time read

elapsedTime = (currentTime - previousTime) / 1000; // Divide by 1000 to get seconds

Wire.beginTransmission(MPU);

Wire.write(0x43); // Gyro data first register address 0x43

Wire.endTransmission(false);

Wire.requestFrom(MPU, 4, true); // Read 4 registers total, each axis value is stored in 2 registers

GyroX = (Wire.read() << 8 | Wire.read()) / 32.8; // For a 1000dps range we have to divide first the raw value by 32.8, according to the datasheet

GyroY = (Wire.read() << 8 | Wire.read()) / 32.8;

GyroX = GyroX + 1.85; //// GyroErrorX ~(-1.85)

GyroY = GyroY - 0.15; // GyroErrorY ~(0.15)

// Currently the raw values are in degrees per seconds, deg/s, so we need to multiply by sendonds (s) to get the angle in degrees

gyroAngleX = GyroX \* elapsedTime;

gyroAngleY = GyroY \* elapsedTime;

// Complementary filter - combine acceleromter and gyro angle values

angleX = 0.98 \* (angleX + gyroAngleX) + 0.02 \* accAngleX;

angleY = 0.98 \* (angleY + gyroAngleY) + 0.02 \* accAngleY;

// Map the angle values from -90deg to +90 deg into values from 0 to 255, like the values we are getting from the Joystick

data.joy1X = map(angleX, -90, +90, 255, 0);

data.joy1Y = map(angleY, -90, +90, 255, 0);

}

**3.3 ESP32 Camera Code: -**

#include "esp\_camera.h"

#include <WiFi.h>

#define CAMERA\_MODEL\_AI\_THINKER // Has PSRAM // JUST USE THE CAMERAWEBSERVER EXAMPLE

#define PWDN\_GPIO\_NUM 32

#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

// ===========================

// Enter your WiFi credentials

// ===========================

const char\* ssid = "\*\*\*\*\*\*\*\*\*\*";

const char\* password = "\*\*\*\*\*\*\*\*\*\*";

void startCameraServer();

void setupLedFlash(int pin);

void setup() {

Serial.begin(115200);

Serial.setDebugOutput(true);

Serial.println();

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\_sccb\_sda = SIOD\_GPIO\_NUM;

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

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

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

config.xclk\_freq\_hz = 20000000;

config.frame\_size = FRAMESIZE\_UXGA;

config.pixel\_format = PIXFORMAT\_JPEG; // for streaming

//config.pixel\_format = PIXFORMAT\_RGB565; // for face detection/recognition

config.grab\_mode = CAMERA\_GRAB\_WHEN\_EMPTY;

config.fb\_location = CAMERA\_FB\_IN\_PSRAM;

config.jpeg\_quality = 12;

config.fb\_count = 1;

// if PSRAM IC present, init with UXGA resolution and higher JPEG quality

// for larger pre-allocated frame buffer.

if(config.pixel\_format == PIXFORMAT\_JPEG){

if(psramFound()){

config.jpeg\_quality = 10;

config.fb\_count = 2;

config.grab\_mode = CAMERA\_GRAB\_LATEST;

} else {

// Limit the frame size when PSRAM is not available

config.frame\_size = FRAMESIZE\_SVGA;

config.fb\_location = CAMERA\_FB\_IN\_DRAM;

}

} else {

// Best option for face detection/recognition

config.frame\_size = FRAMESIZE\_240X240;

#if CONFIG\_IDF\_TARGET\_ESP32S3

config.fb\_count = 2;

#endif

}

#if defined(CAMERA\_MODEL\_ESP\_EYE)

pinMode(13, INPUT\_PULLUP);

pinMode(14, INPUT\_PULLUP);

#endif

// camera init

esp\_err\_t err = esp\_camera\_init(&config);

if (err != ESP\_OK) {

Serial.printf("Camera init failed with error 0x%x", err);

return;

}

sensor\_t \* s = esp\_camera\_sensor\_get();

// initial sensors are flipped vertically and colors are a bit saturated

// drop down frame size for higher initial frame rate

if(config.pixel\_format == PIXFORMAT\_JPEG){

s->set\_framesize(s, FRAMESIZE\_QVGA);

}

// Setup LED FLash if LED pin is defined in camera\_pins.h

#if defined(LED\_GPIO\_NUM)

setupLedFlash(LED\_GPIO\_NUM);

#endif

WiFi.begin(ssid, password);

WiFi.setSleep(false);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

startCameraServer();

Serial.print("Camera Ready! Use 'http://");

Serial.print(WiFi.localIP());

Serial.println("' to connect");

}

void loop() {

// Do nothing. Everything is done in another task by the web server

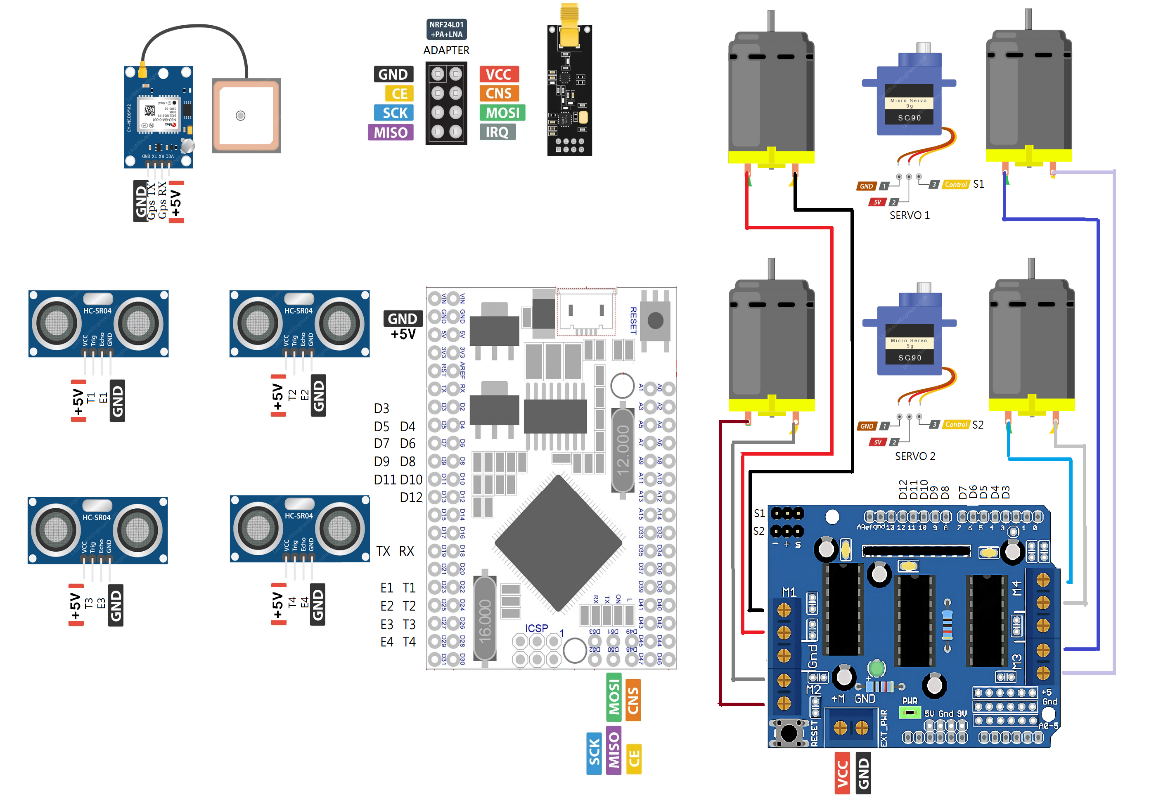
delay(10000);

}

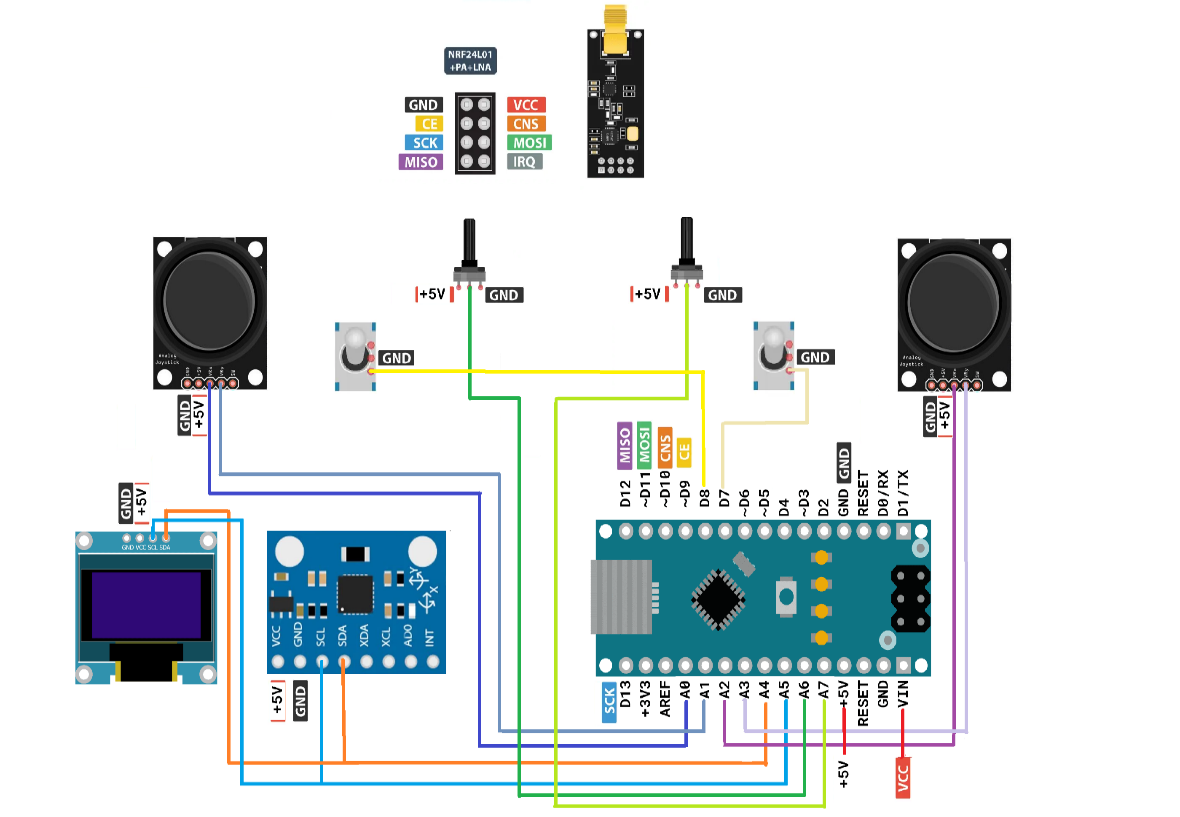
**CHAPTER 4: - DESIGN AND WORKING**

**4.1 Circuit Design: -**

**1.ROVER RECIEVER CIRCUIT DIAGRAM:**



**2.TRANSMITTER CIRCUIT DIAGRAM**



**4.2 Rover Body Making: -**

The body is fully made up of acrylic sheets which is joined together with adhesive. The acrylic is cut using the jigsaw. It is also drilled where ever required using the drilling machine. After the cutting of acrylic sheet, the surfaces are smoothened with the help of regmark paper.

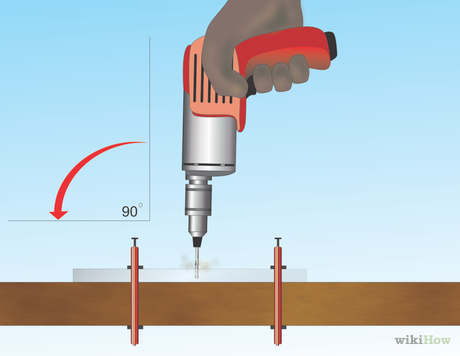
**PRECAUTIONS REQUIRED WHILE CUTTING THE ACRYLIC SHEET:**

* Make sure you’re cutting on a level surface and that the acrylic sheet is secured in a clamp or is otherwise in a stable position
* Wear eye protection and gloves, and make sure those around you wear eye protection as well
* Acrylic contains respiratory irritants, so wear a painter’s mask or a similar sealed ventilator during the cutting process
* Other individuals in the area should also wear masks, and exhaust fans should be running throughout the cutting process
* Cut carefully and slowly, and disconnect your saw from power when done



**PRECAUTIONS REQUIRED WHILE DRILLING THE ACRYLIC SHEET:**

* Drilling must be done with a high-speed drilling machine.
* The drilling should be done in slow pace and very less jerk should be applied.
* If the drilling is done by a hand drilling machine it should be kept straight



**4.3 How It Works: -**

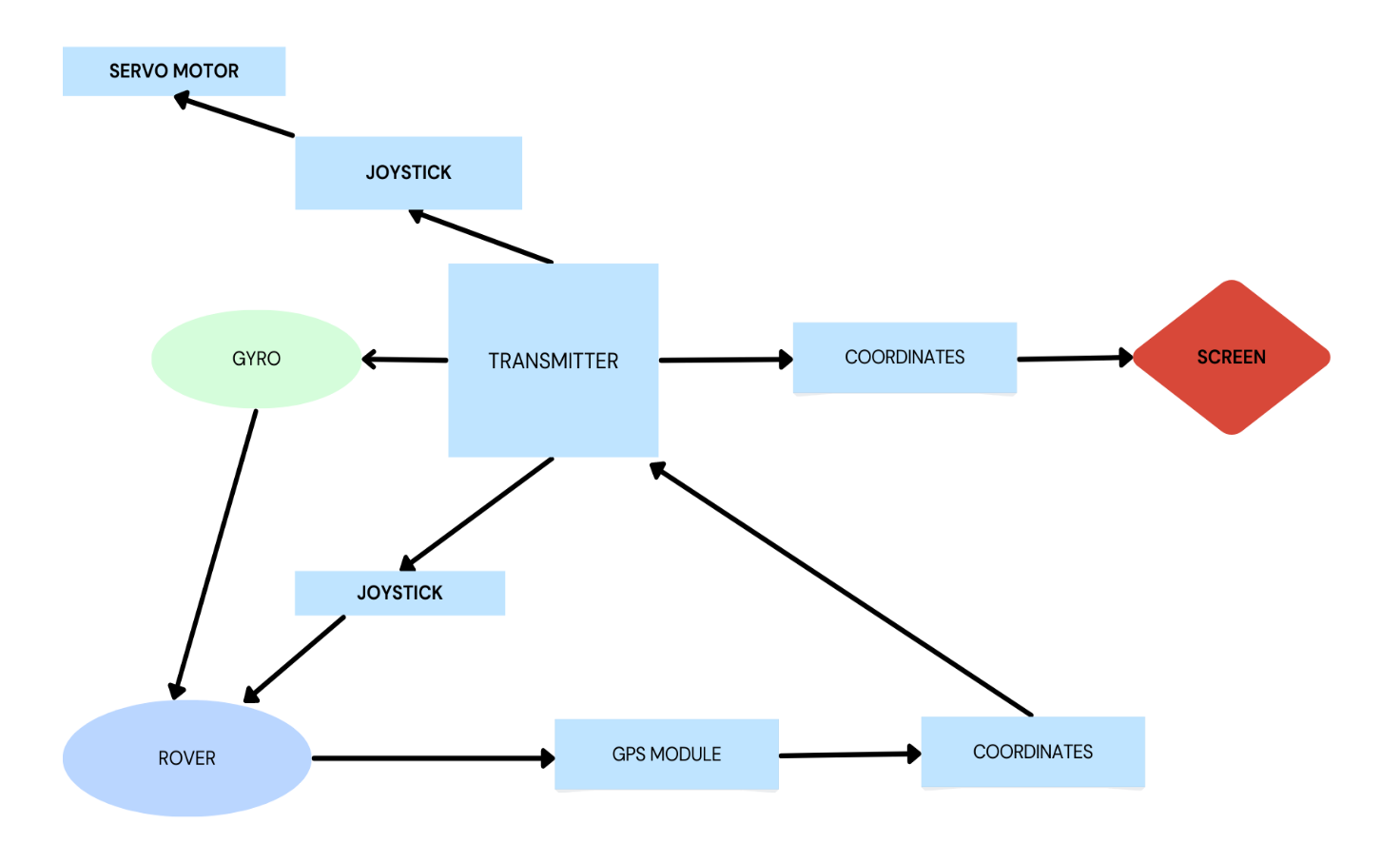
The project generally consists of two components i.e.:

1. Transmitter
2. Receiver rover

**1. Transmitter**: The transmitter consists of a gyroscope sensor which detects the motion of the transmitter and transmit it to the receiver's end which gives the command to the rover to move in the required direction. The other way to control the rover is using the joystick module available in the transmitter. The modes can be changed using the toggle switch. The joystick module also enables the rover to rotate in both anticlockwise and clockwise direction. There is also an OLED screen present in the transmitter which shows the coordinates of the rover where it is present. The other joystick module controls the moment of servo motors.

**2. Receiver Rover:** It takes command from the transmitter using the receiver module and performs the required actions. A GPS module is present which detects the coordinates of the rover and sends it to the transmitter which displays it on the led screen. The ultrasonic sensors present in the rover helps the rover to avoid collisions.

The ESP32 camera module is used for surveillance which is mounted over the servo motor which allows it to rotates in different directions. The live footage of the surveillance could be seen on the laptop.



**CHAPTER 5 : -SUMMARY AND FUTURE SCOPE**

**5.1 Summary: -**

The Arduino-based Gesture Control Rover is a robotic vehicle that can be controlled using hand inputs and remote. This includes Arduino microcontroller, which acts as the rover's brain, as well as various sensors and actuators that enable movement and gesture recognition.  
  
The rover typically consists of a chassis with wheels for locomotion, motor drives that control the movement of the wheels, and a power supply that powers the system. It also includes a gesture recognition engine, such as an accelerometer and gyroscope, that recognizes and interprets remote movements.  
  
To control the rover, the user performs predefined action recognized by the sensor module. The Arduino microcontroller processes sensor data and triggers appropriate actions such as: Move forward, backward, turn left or right, or stop the rover. These commands are sent to the motor drivers, which control the motors attached to the wheels to enable the desired movement.  
  
The Arduino platform provides an easy-to-use environment for programming gesture recognition and motor control logic. The Arduino programming language allows users to define gestures and corresponding actions to customize rover behavior and integrate additional features such as obstacle avoidance and wireless communication.  
  
Overall, the Arduino-based gesture-controlled rover manipulates robotic vehicles by allowing users to control the robotic vehicle's movements. It provides an interactive and intuitive way to Combine the capabilities of Arduino microcontrollers, sensor modules, and motor drivers to create engaging and engineering project.

**5.2 Future scope: -**

It could find uses in the fields such as Environment Monitoring where it could monitor and collect data like air quality, temperature, humidity and pollution levels. Next its primary objective could be like Search and Rescue where it could assist individuals in emergency situations. Later, it could also be used in the vast field of Agriculture where it could monitor soil moisture, detect diseases and precise spraying of fertilizers or pesticides. Then it could also find its application in the Industial applications such as to increase efficiency and productivity and further more applications down the line.

1. [↑](#footnote-ref-2)