Multipurpose Robot

IOT Mini Project

Srushti Narvekar (IT16)
Akshay (IT17)

Contents

Introduction

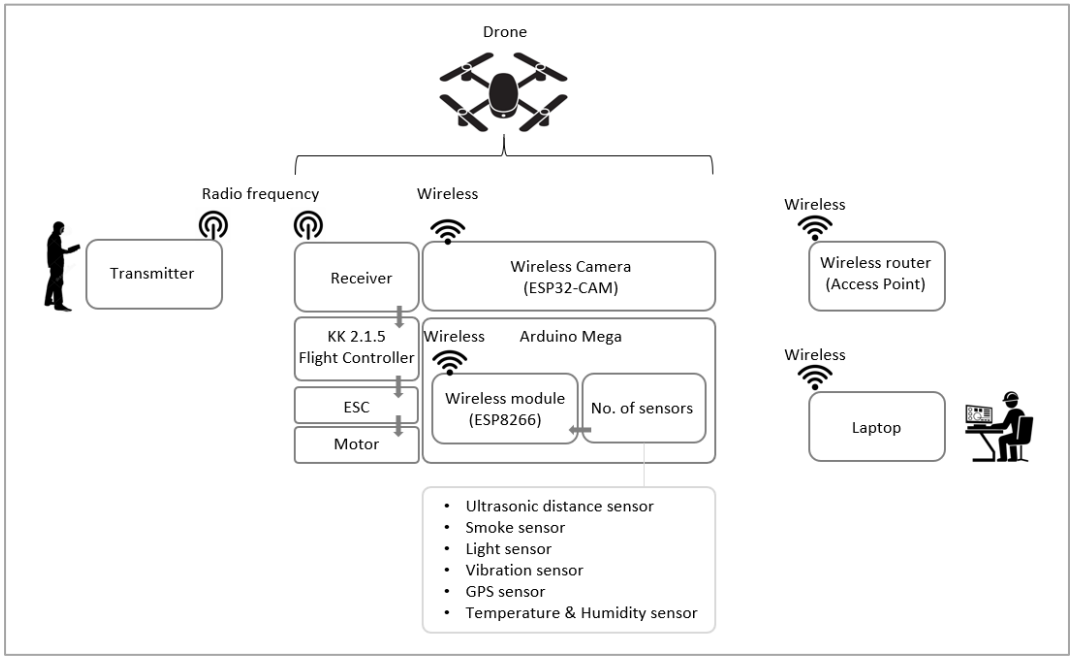
A Multipurpose robot (MPR) using drone has the potential for performing many tasks where humans cannot enter, for example, high temperature and high-altitude surveillance in many industries, rescue missions.

A Drone has four propellers with motors that generate, the thrust for lifting the aircraft. A drone is also called as the Quadcopter. The basic principle behind the quadcopter is, the two motors will rotate in the clockwise direction the other two will rotate in an anticlockwise direction allowing the aircraft to vertically ascend.

While taking the flight with the help a camera we can have live streaming and capture images. Using ultrasonic sensor, MPR can identify presence of obstacle in front of him. MPR can sense temperature & humidity, smoke, presence of light, vibration using DHT11, MQ2, light sensor & vibration sensor respectively. With the help of the GPS module we can have global position of the MPR.

Overview

Block diagram



Bill of material

Sr.No.	Part Description	Qty.	Cost
1	Arduino Mega 2560	1	₹ 678.00
2	Wi-Fi Module (ESP8266)	1	₹ 127.00
3	Wi-Fi Camera Module (ESP32-CAM)	1	₹ 850.00
4	Vibration sensor module (SW-420)	1	₹ 42.00
5	Light Sensor	1	₹ 190.00
6	Ultrasonic distance sensor	1	₹ 157.00
7	DHT11 sensor	1	₹ 200.00
8	MQ2 Smoke sensor	1	₹ 220.00
9	U-BLOX NEO-6M GPS Positioning Module	1	₹ 789.00
10	Light Emitting Diodes (LED)	10	₹ 10.00
11	Buzzer	1	₹ 100.00
12	Cable jumpers	3	₹ 150.00
13	USB to Serial adapter	1	₹ 268.00
14	TP-Link TL-MR3020 Mini Pocket 3G/4G Wireless Router	1	₹ 1,099.00
15	F450 Quadcopter frame	1	₹ 636.00
16	1045 Quadcopter propeller pair	2	₹ 118.00
17	Electronic Speed Controller (ESC 30A)	4	₹ 1,220.00
18	Brushless DC Motor 1000KV	4	₹ 1,220.00
19	Flight Controller board KK 2.1.5	1	₹ 1,499.00
20	Transmitter and Receiver for Quadcopter	1	₹ 3,100.00
21	AA Battery	8	₹ 240.00
22	Lipo Battery 3S (2200Mah 11.1V 25/30C)	1	₹ 1,399.00
23	Lipo Battery Charger	1	₹ 458.00
			₹ 14,770.00

Software

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board.

Hardware

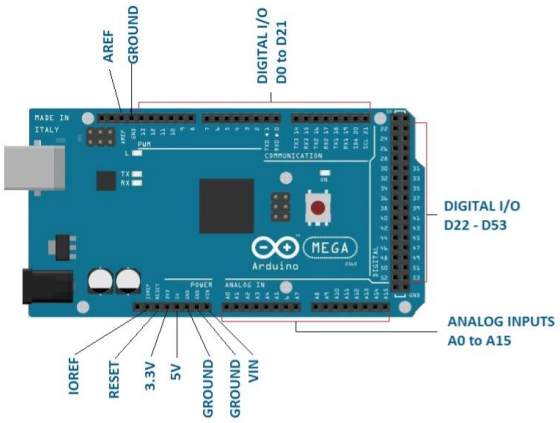
Part Name: Arduino Mega 2560

Part Image:

Part Description:

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno

Part wiring sample:

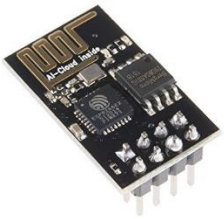


Part sample code:

```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

Part Name: ESP8266

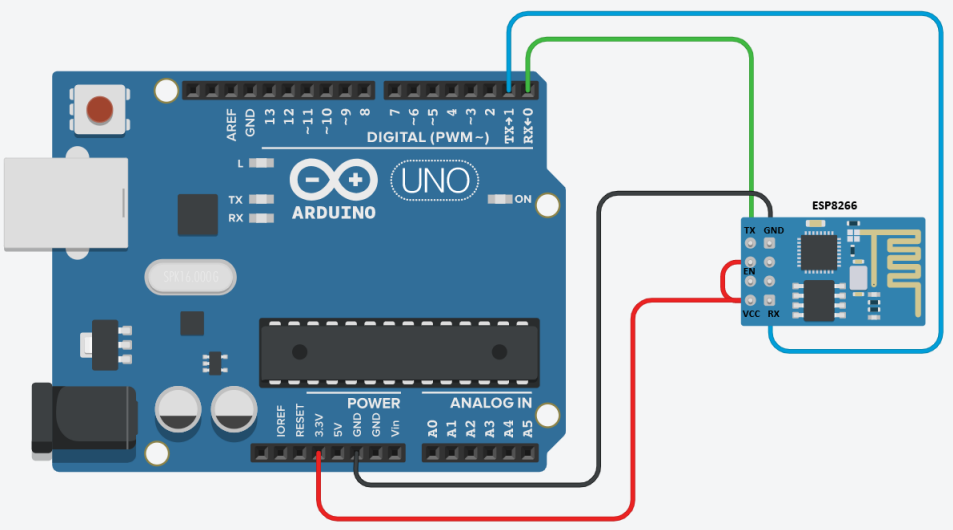
Part Image:



Part Description:

The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network

Part Wiring:



Part Sample Code:

```
//------------------------------------------------------------------------WiFi module (ESP8266) supporting declaration

String server=""; //variable for sending data to webpage
boolean No_IP=false; //variable to check for ip Address
String IP=""; //variable to store ip Address
char temp1='0';

int a=0;
int b=0;

//-------------------------------------------------------------------------------------------------------------------
```

```
void loop()
{
  //-------------------------------------------
  b=0;
  Serial.println("Refresh Page");
  while(b<1000)
  {
    b++;
    while(Serial3.available())
    {
      if(Serial3.find("0,CONNECT"))
      {
        Serial.println("Starting");
        sendToServer();
        Serial.println("Finished");
        delay(1000);
      }
    }
    delay(1);
  }
  //-------------------------------------------
  digitalWrite(LED_Green, HIGH);
  DHT_Reading();
  MQ2_Reading();
  Vibration_Reading();
  Light_Reading();
  Ultrasonic_Reading();
  GPS_Reading();
  //-------------------------------------------
  delay(1000);
}
```

```
void findIp(int time1) //check for the availability of IP Address
{
  int time2=millis();
  while(time2+time1>millis())
  {
    while(Serial3.available()>0)
    {
      if(Serial3.find("IP has been read"))
      {
        No_IP=true;
      }
    }
  }
}
```

```
void showIP()//Display the IP Address
{
  IP="";
  char ch=0;
  while(1)
  {
    Serial3.println("AT+CIFSR");
    while(Serial3.available()>0)
    {
      if(Serial3.find("STAIP,"))
      {
        delay(1000);
        Serial.print("IP Address:");
        while(Serial3.available()>0)
        {
          ch=Serial3.read();
          if(ch=='+')
          break;
          IP+=ch;
        }
      }
      if(ch=='+')
      break;
    }
    if(ch=='+')
    break;
    delay(1000);
  }
  Serial.print(IP);
  Serial.print("Port:");
  Serial.println(80);
}
```

```
void establishConnection(String command, int timeOut) //Define the process for sending AT commands to module
{
  int q=0;
  while(1)
  {
    Serial.println(command);
    Serial3.println(command);
    while(Serial3.available())
    {
      if(Serial3.find("OK"))
      q=8;
    }
    delay(timeOut);
    if(q>5)
    break;
    q++;
  }
  if(q==8)
  Serial.println("OK");
  else
  Serial.println("Error");
}
```

```
void wifi_init() //send AT commands to module
{
  establishConnection("AT",100);
  delay(1000);
  establishConnection("AT+CWMODE=3",100);
  delay(1000);
  establishConnection("AT+CWQAP",100);
  delay(1000);
  establishConnection("AT+RST",5000);
  delay(1000);
  findIp(5000);
  if(!No_IP)
  {
    Serial.println("Connecting Wifi....");
    establishConnection("AT+CWJAP=\"MPR\",\"mpr@1234\"",7000); //provide your WiFi username and password here
  }
  else
  {

  }
  Serial.println("Wifi Connected");
  showIP();
  establishConnection("AT+CIPMUX=1",100);
  establishConnection("AT+CIPSERVER=1,80",100);
}
```

```
void sendData(String server1)//send data to module
{
  int p=0;
  while(1)
  {
    unsigned int l=server1.length();
    Serial.print("AT+CIPSEND=0,");
    Serial3.print("AT+CIPSEND=0,");
    Serial.println(l+2);
    Serial3.println(l+2);
    delay(100);
    Serial.println(server1);
    Serial3.println(server1);
    while(Serial3.available())
    {
      //Serial.print(Serial.read());
      if(Serial3.find("OK"))
      {
        p=11;
        break;
      }
    }
    if(p==11)
    break;
    delay(100);
  }
}
```

```
void sendToServer()//send data to webpage
{
  server = "<h1>Multipurpose Robot</h1>";
  server+="<p>=========================================================================================</p>";
  server+="<p>Temperature (Deg.C): " + String(DHT_Temp)+ "<p>";
  server+="<p>Humidity (%): " + String(DHT_Humidity)+ "<p>";
  server+="<p>Smoke reading: " + String(Smoke_Reading)+ "<p>";
  server+="<p>Vibration status: " + String(Vibration_Status)+ "<p>";
  server+="<p>Light status: " + String(Light_Status)+ "<p>";
  server+="<p>Obstacle distance (cm): " + String(Obstacle_Distance)+ "<p>";
  server+="<p>GPS (Latitude,Longitude): " + String(Latitude_Status)+","+ String(Longitude_Status)+"<p>";
  server+="<p>=========================================================================================</p>";
  sendData(server);
  delay(1000);
  Serial3.println("AT+CIPCLOSE=0");
}
```

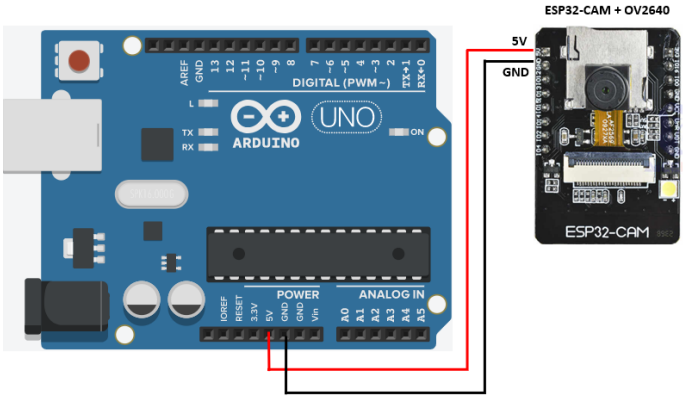
Part Name: ESP32-CAM

Part Image:

Part Description:

The ESP32-CAM is a very small camera module with the ESP32-S chip. Besides the OV2640 camera, and several GPIOs to connect peripherals, it also features a microSD card slot that can be useful to store images taken with the camera or to store files to serve to clients

Part Wiring:



Part Sample Code:

```
#include "esp_camera.h"
#include <WiFi.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"

#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"

const char* ssid = "MPR";
const char* password = "mpr@1234";

void startCameraServer();
```

```
void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  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_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
```

```
  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

#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
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);//flip it back
    s->set_brightness(s, 1);//up the blightness just a bit
    s->set_saturation(s, -2);//lower the saturation
  }
  //drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);
```

```
#if defined(CAMERA_MODEL_M5STACK_WIDE)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

  WiFi.begin(ssid, password);

  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() {
  // put your main code here, to run repeatedly:
  delay(10000);
}
```

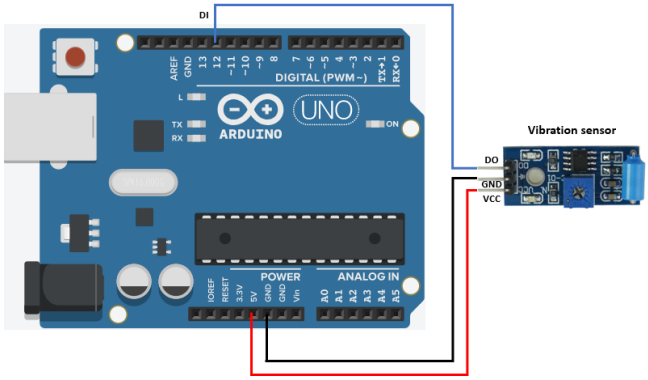
Part Name: Vibration sensor module (SW-420)

Part Image:

Part Description:

The Vibration module based on the vibration sensor SW-420 and Comparator LM393 to detect if there is any vibration that beyond the threshold. The threshold can be adjusted by the on-board potentiometer. When this no vibration, this module output logic LOW the signal indicate LED light, and vice versa.

Part Wiring:



Part Sample Code:

```
void Vibration_Reading()
{

  Vibration_Status=digitalRead(Vibration_Pin);
  if(Vibration_Status==1)
  {
    digitalWrite(LED_Yellow, HIGH);
  }
  else
  {
    digitalWrite(LED_Yellow, LOW);
  }

}
```

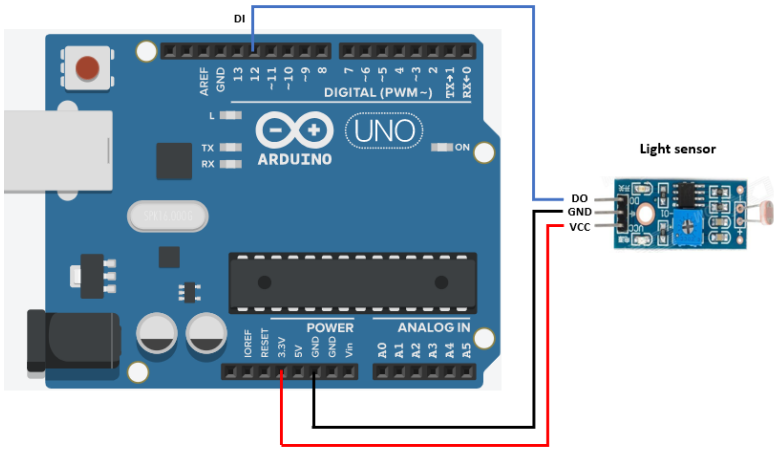
Part Name: Light sensor

Part Image:

Part Description:

It is a photosensitive resistor module, suitable to detect environmental light intensity and ambient brightness. Its sensitivity can be tuned with an on board potentiometer, where turning it clockwise will increase the sensitivity and increase the detection range.

Part Wiring:



Part Sample Code:

```
void Light_Reading()
{
  Light_Status = digitalRead(Light_Sensor_Pin);

  if(Light_Status == HIGH )
  {
    digitalWrite(LED_White, HIGH);
  }
  else
  {
    digitalWrite(LED_White, LOW);
  }

}
```

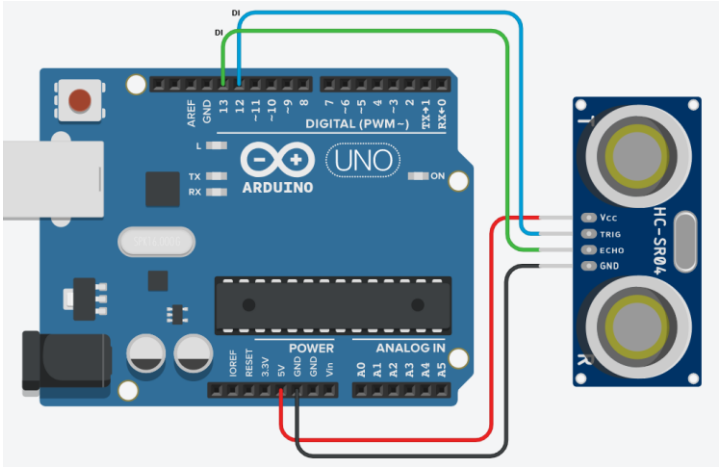
Part Name: Ultrasonic distance sensor

Part Image:

Part Description:

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module

Part Wiring:



Part Sample Code:

```
void Ultrasonic_Reading()
{
  digitalWrite(Ultrasonic_Trigger_Pin, LOW);
  delayMicroseconds(2);
  digitalWrite(Ultrasonic_Trigger_Pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(Ultrasonic_Trigger_Pin, LOW);

  Ultrasonic_Wave_Travel_Duration = pulseIn(Ultrasonic_Echo_Pin, HIGH);
  Obstacle_Distance = (Ultrasonic_Wave_Travel_Duration*.0343)/2;

  if(Obstacle_Distance < 30 )
  {
    digitalWrite(LED_Green, LOW);
    delay(100);
    digitalWrite(LED_Green, HIGH);
    delay(100);
  }

}
```

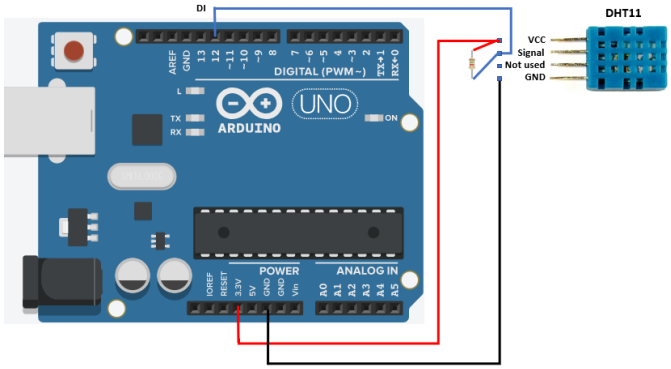
Part Name: Temperature & Humidity sensor (DHT11)

Part Image:

Part Description:

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor to measure humidity and a thermistor to measure the temperature of surrounding air

Part Wiring:



Part Sample Code:

```
void DHT_Reading()
{

  int chk = DHT11.read(DHT11_Pin);

  DHT_Temp=DHT11.temperature;
  DHT_Humidity=DHT11.humidity;

}
```

Part Name: Smoke sensor (MQ2)

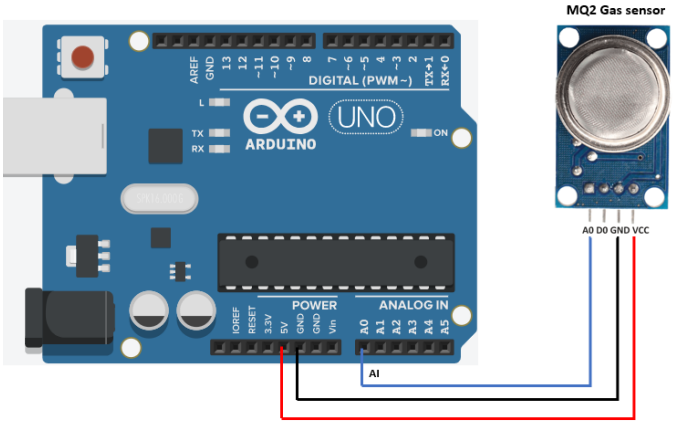
Part Image:

Part Description:

The MQ-2 smoke sensor is sensitive to smoke and to the flammable gases like LPG, Butane, Propane, Methane, Alcohol, Hydrogen

The resistance of the sensor is different depending on the type of the gas. The smoke sensor has a built-in potentiometer that allows you to adjust the sensor sensitivity according to how accurate you want to detect gas.

Part Wiring:



Part Sample Code:

```
void MQ2_Reading()
{

  Smoke_Reading = analogRead(MQ2_Pin);

  if (Smoke_Reading > Smoke_Alarm_Setpoint)
  {
    tone(Buzzer_Pin, 1500, 1000);
    digitalWrite(LED_Red, HIGH);
  }
  else
  {
    noTone(Buzzer_Pin);
    digitalWrite(LED_Red, LOW);
  }

}
```

Part Name: GPS Positioning Module (U-BLOX NEO-6M)

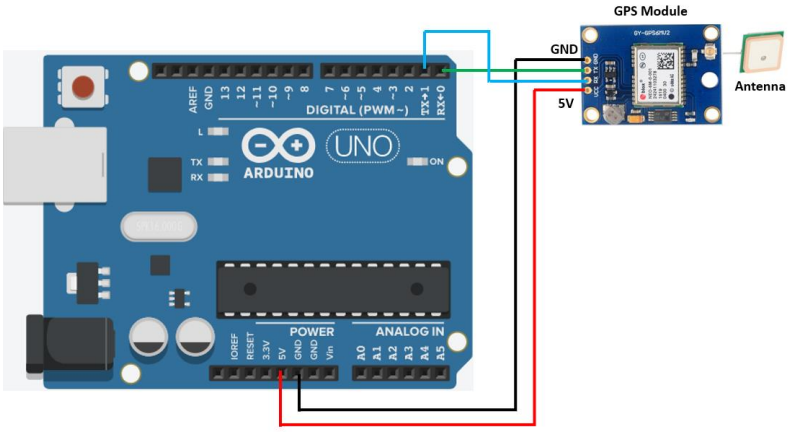
Part Image:

Part Description:

GPS satellites transmit at least 2 low-power radio signals. The signals travel by line of sight, meaning they will pass through clouds, glass and plastic but will not go through most solid objects, such as buildings and mountains. However, modern receivers are more sensitive and can usually track through houses.

These GPS satellites transmit information signal over radio frequency (1.1 to 1.5 GHz) to the receiver. With the help of this received information, a ground station or GPS module can compute its position and time.

Part Wiring:



Part Sample Code:

```
void GPS_Reading()
{
  while (Serial2.available() > 0)
  {
    gps.encode(Serial2.read());
    if (gps.location.isUpdated())
    {
      Latitude_Status=(gps.location.lat(), 6);
      Longitude_Status=(gps.location.lng(), 6);
      Serial.print(gps.location.lat(), 6);
      Serial.println(gps.location.lng(), 6);
    }

  }
}
```

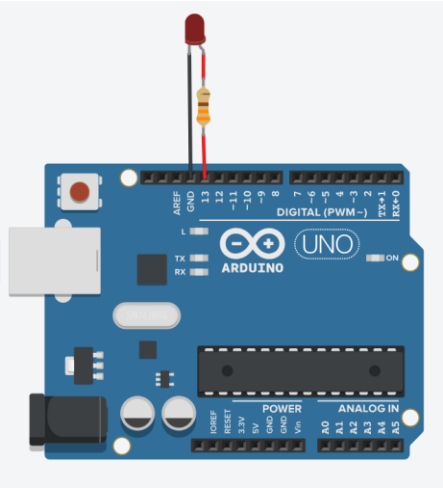
Part Name: LED's

Part Image:

Part Description:

A light-emitting diode is a two-lead semiconductor light source. It is a p–n junction diode that emits light when activated.

Part Wiring:



Part Sample Code:

```
if(Vibration_Status==1)
{
  digitalWrite(LED_Yellow, HIGH);
}
else
{
  digitalWrite(LED_Yellow, LOW);
}
```

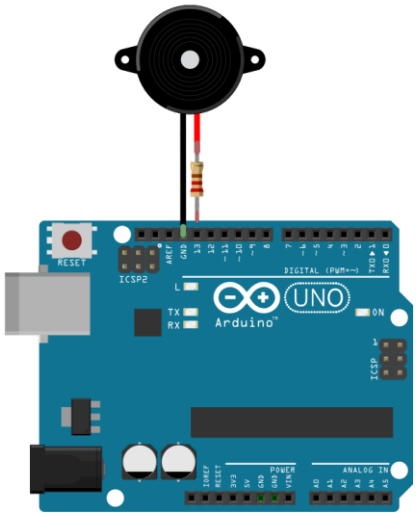
Part Name: Buzzer

Part Image:

Part Description:

Piezo buzzers are simple devices that are commonly used to produce beeps and sounds in many electronic gadgets, like alarm clocks, toys, pc boards, etc. They consume very little current and have high impedance, which means that you can safely connect them directly to a micro-controller pin.

Part Wiring:



Part Sample Code:

```
if (Smoke_Reading > Smoke_Alarm_Setpoint)
{
  tone(Buzzer_Pin, 1500, 1000);
  digitalWrite(LED_Red, HIGH);
}
else
{
  noTone(Buzzer_Pin);
  digitalWrite(LED_Red, LOW);
}
```

Part Name: Cable jumpers

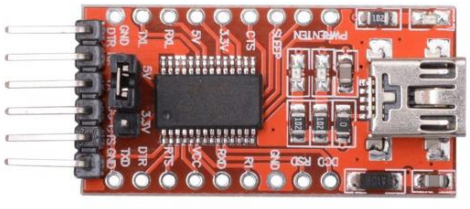
Part Image:

Part Description:

A jump wire (also known as jumper wire, or jumper) is an electrical wire, or group of them in a cable, with a connector or pin at each end, which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

Part Name: USB to Serial adapter

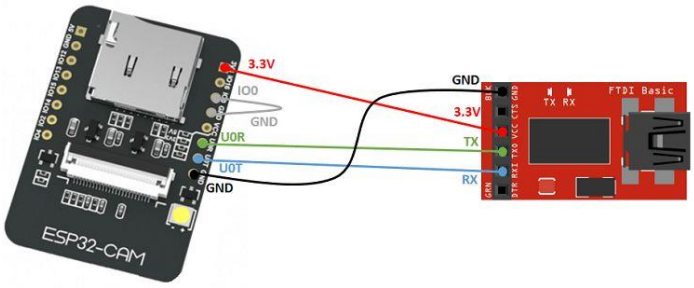
Part Image:



Part Description:

The USB TTL Serial cables are a range of USB to serial converter cables which provide connectivity between USB and serial UART interfaces. A range of cables are available offering connectivity at 5V, 3.3V or user specified signal levels with various connector interfaces

Part Wiring: Sample wiring with ESP32-CAM for programming



Part Name: Mini wireless router

Part Image:

Part Description:

Mini wireless router has the wireless speed and range to power a complex set of networking applications to create a highly efficient mobile office or entertainment network in no time. Small enough to fit in the average pocket and with no cables to plug in, save a WAN cable, the router is uniquely suited to providing robust wireless networking to students, or anyone else for work or play.

Part Name: F450 quadcopter frame

Part Image:

Part Description:

The F450 frames are built from very strong materials, the arms are made from the ultra-strong PA66+30GF material which provides better resistance to damage on hard landings, while the main frame plates use a high strength compound PCB material, which makes wiring of ESCs and battery easy and safe on the lower of the two frame plates which is also the power distribution board.

Part Name: 1045 Propeller pair

Part Image:

Part Description:

A set of 2 x 1045 Propellers. The set includes one clockwise rotating and one Counter-clockwise rotating propeller.

Diameter: 10" (25.4 cm), Pitch: 4.5" (11.43 cm)

Part Name: Electronic Speed Controller (ESC 30A)

Part Image:

Part Description:

Electronic Speed Controller is like a nerve that delivers the movement information from the brain (flight controller) to the arm or leg muscles (motors). It regulates how much power the motors get, which determines the speed and direction changes of the quad.

Part Name: Brushless DC Motor 1000KV

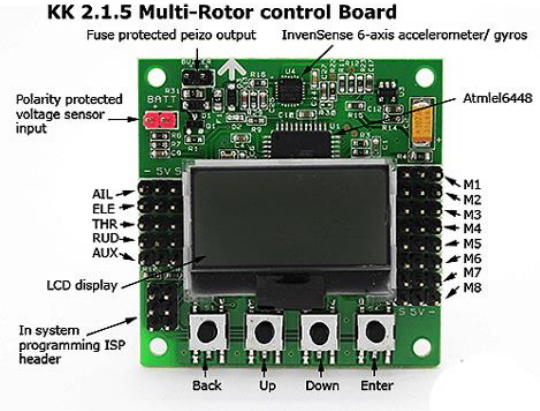
Part Image:

Part Description:

The thrust that allows the Quadcopter to get airborne is provided by Brushless DC motors and each of them is separately controlled by an electronic speed controller or ESC.

Part Name: Flight Controller board KK 2.1.5

Part Image:



Part Description:

KK2.1 Multi-Rotor controller manages the flight of (mostly) multirotor Aircraft (Tri copters, Quadcopters, Hex copters etc.). Its purpose is to stabilize the aircraft during flight and to do this, it takes signals from on-board gyroscopes (roll, pitch and yaw) and passes these signals to the Atmega324PA processor, which in-turn processes signals according the users selected firmware (e.g. Quadcopter) and passes the control signals to the installed Electronic Speed Controllers (ESCs) and the combination of these signals instructs the ESCs to make fine adjustments to the motors rotational speeds which in-turn stabilizes the craft. The KK2.1 Multi-Rotor control board also uses signals from your radio system via a receiver (Rx) and passes these signals together with stabilization signals to the Atmega324PA IC via the aileron; elevator; throttle and rudder user demand inputs. Once processed, this information is sent to the ESCs which in turn adjust the rotational speed of each motor to control flight orientation (up, down, backwards, forwards, left, right, yaw)

Part Name: Transmitter and Receiver for Quadcoopter

Part Image:

Part Description:

The transmitter enables control through radio waves and the receiver receives the signals to perform drone movement through flight controller board.

Part Name: Lipo battery (Rechargeable)

Part Image:

Part Description:

Lithium-polymer batteries are lighter and more flexible than other kinds of lithium-ion batteries because of their soft shells, allowing them to be used in mobile and other electronic devices, as well as in remote control vehicles. Voltage: 11.1V & power ratingd:2200mah.

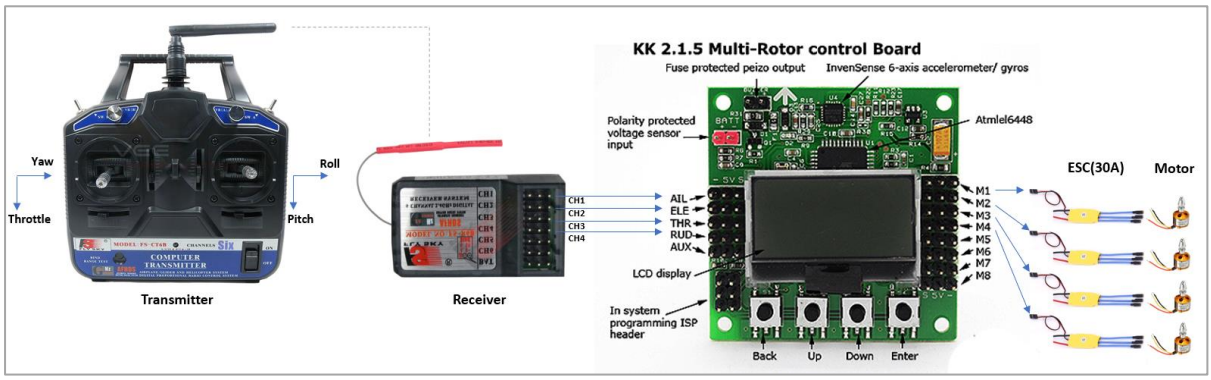
Part Name: Lipo charger

Part Image:

Part Description:

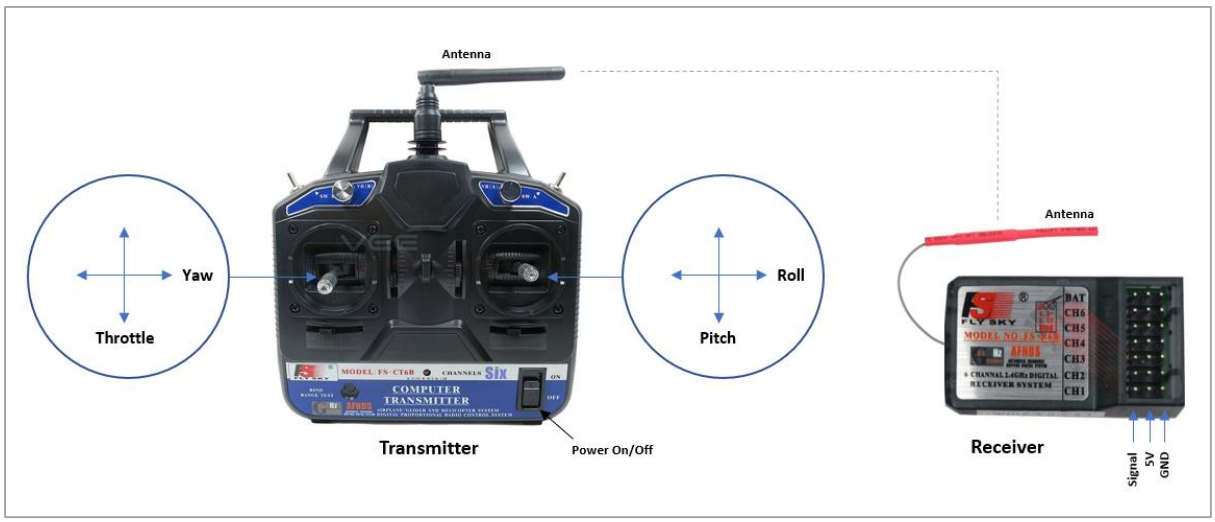
This is a simple and compact LiPoly balance charger for 2~3s batteries. It features built in JST-XH balance plug ports and 3 LEDs to indicate charge status. Comes with Inbuilt Power Supply. So you can straightaway plug it into the wall outlet. It's an ideal charger to charge your LiPo Packs. Simple, reliable and safe to use Built in power supply 100~240V.

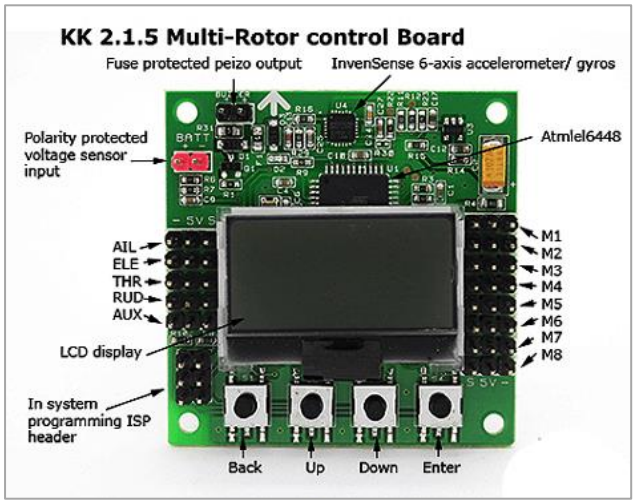
Flight control flow diagram



A radio transmitter is a device that allows the user to control the drone wirelessly. The signal/commands are then received by a radio receiver which is connected to a flight controller. Flight controller send the signals to motor through ESC and drone moves accordingly.

Close look





Flight controller setup

Note: Propeller should be disconnected

Step-1: Factory Reset	Menu → Factory Reset
Step-2: ACC Calibration	Menu → ACC Calibration
	Make sure the quad is on a flat and level surface.
Step-3: Load motor layout	Menu → Load Motor Layout → Quadcopter X mode
Step-4: Receiver Test	Menu → Receiver Test (Tune all positions to zero)
Step-5: PI Editor	Menu → PI Editor
	Axis: Roll (Aileron) = P Gain:75, P Limit:50, I Gain:40, I Limit:20
	Axis: Pitch (Elevator) = P Gain:75, P Limit:50, I Gain:40, I Limit:20
	Axis: Yaw (Rudder) = P Gain:75, P Limit:20, I Gain:30, I Limit:10
Step-6: Self level setting	Menu → Self-level Settings (P Gain: 70, P Limit:20)

Step7: ESC throttle calibration

- Switch off flight controller board & Transmitter
- Switch on transmitter and keep its throttle in full position
- Now press & hold 1st & 4th button together and power on flight controller board.
- Now after few seconds put the throttle in zero position
- Wait for short beep and release the both buttons
- Now spin the motors using throttle movement. All of them should start at same time.

Step8: Motor direction

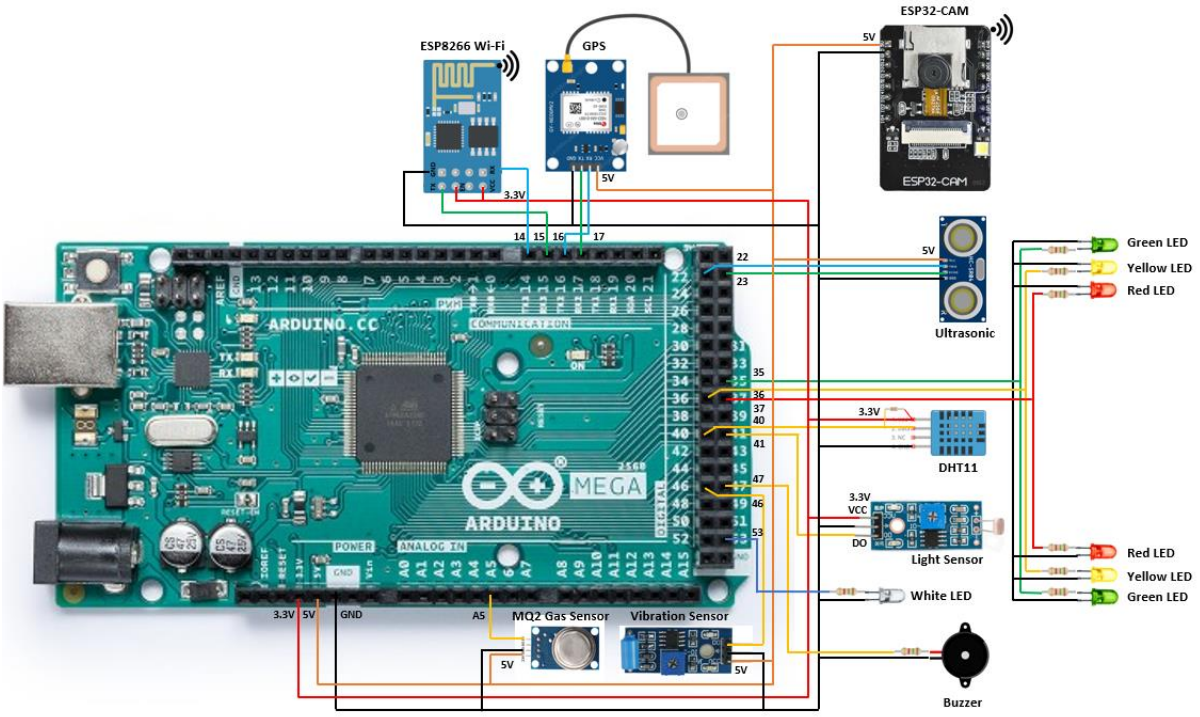
- Check all motor directions as desired i.e.
- Motor1: CW, Motor2: CCW, Motor3: CW, Motor4: CCW
- CW means Clockwise, CCW means Counter-clockwise
- If motor spins in wrong direction, then switch any two wires connected between motor & ESC

Step9: Flight control

ARMED/START: Put throttle to zero position & move to full left

SAFE/STOP: Put throttle to zero position & move to full right

Wiring Scheme



Functional details

Sr.No.	Part Description	Function
1	Arduino Mega 2560	Read inputs, process it as per program & gives output
2	Wi-Fi Module (ESP8266)	Broadcast Arduino parameters on Wi-Fi
3	Wi-Fi Camera Module (ESP32-CAM)	Live video stream
4	Vibration sensor module (SW-420)	Sense vibration
5	Light Sensor	Sense light presence around drone
6	Ultrasonic distance sensor	Sense obstacle presence in front of drone
7	DHT11 sensor	Sense temperature & humidity surround the drone
8	MQ2 Smoke sensor	Sense smoke presence surround the drone
9	U-BLOX NEO-6M GPS Positioning Module	Get Global position of drone
10	Light Emitting Diodes (LED)	Get visual indication
11	Buzzer	Buzz the sound for any alarming condition
12	TP-Link TL-MR3020 Mini Pocket 3G/4G Wireless Router	Work as access point for Wi-Fi network
13	Flight Controller board KK 2.1.5	Control the flight