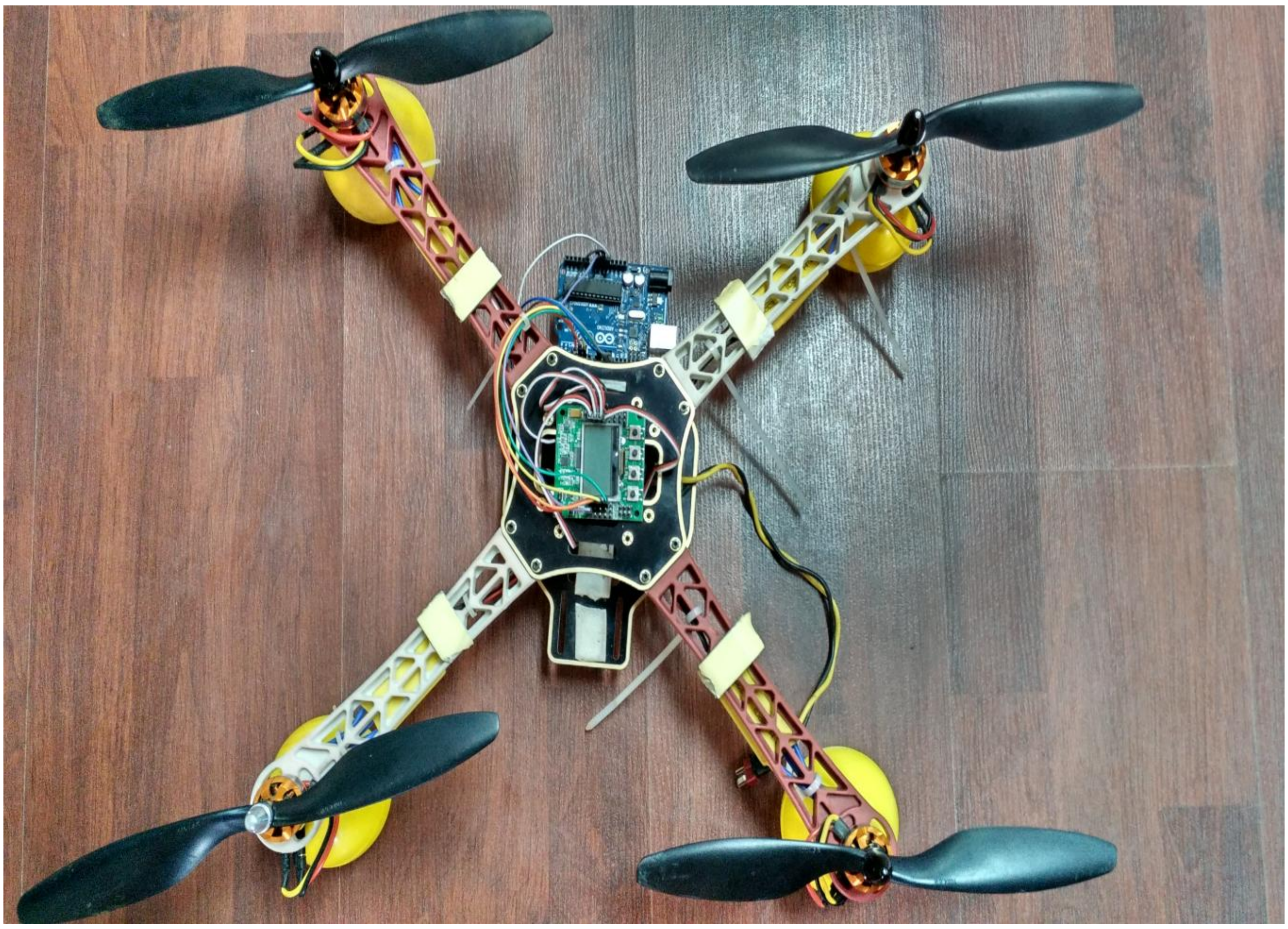
ABSTRACT:

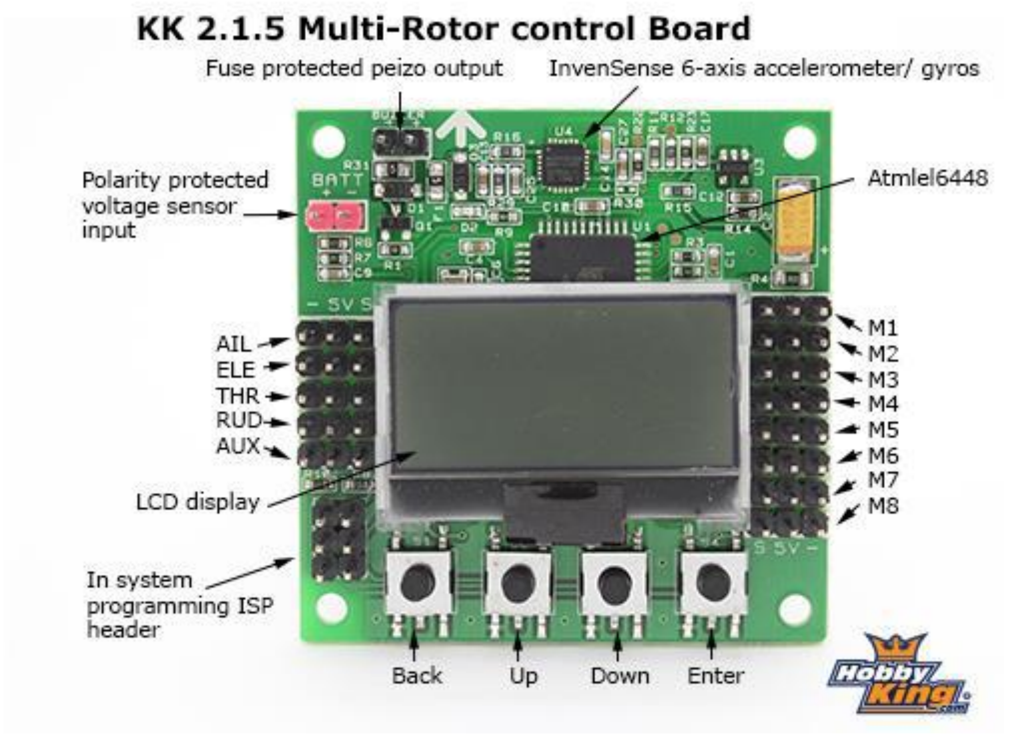
The objective of this project is to design an autonomous landing vehicle which has the capability to land safely on a flat surface by auto rpm control and hovering based on the data from the ultrasonic height sensor.
The lander has the basic design of a quad rotor. The chassis of the lander has an X design with all four arms attached to the centre cuboidal region. The ultrasonic height sensor is interfaced to the Arduino Uno microcontroller board based on whose data the kk2.1 board is to be operated. The motors are attached to the ends of the four arms which are being controlled by electronic speed controllers which in turn are controlled by the kk2.1 multirotor controller board.
The flight of the lander is based on its motion along the three axes namely:
1. Roll
2. Pitch
3. Yaw
Three signals namely Aileron, Elevator and Rudder are used to control the motion of the lander along the above mentioned axes.



COMPONENTS:

- KK 2.1 MULTIROTOR FLIGHT CONTROLLER BOARD
- ARDUINO UNO
- ULTRASONIC DISTANCE SENSOR
- LITHIUM POLYMER BATTERY
- BRUSHLESS DC MOTOR
- ELECTRONIC SPEED CONTROLLERS
- FRAME AND PROPELLERS
- JUMPER WIRES

KK2.1 MULTIROTOR FLIGHT CONTROLLER BOARD:


KK 2.1.5 Multi-Rotor control Board
Fuse protected peizo output
InvenSense 6-axis accelerometer/ gyros
Polarity protected voltage sensor input
Atmlel6448
AIL
ELE
THR
RUD
AUX
M1
M2
M3
M4
M5
M6
M7
M8
LCD display
In system programming ISP header
Back
Up
Down
Enter

Specifications:

Size:	50.5mm x 50.5mm x 12mm
Weight:	21 gram (Inc Piezo buzzer)
IC:	Atmega644 PA
Gyro/Acc:	6050MPU InvenSense Inc.
Auto-level:	Yes
Input Voltage:	4.8-6.0V
AVR interface:	standard 6 pin.
Signal from Receiver:	1520us (5 channels)
Signal to ESC:	1520us

Description:

The Hobby King KK2.1.5 Multi-Rotor controller is a flight control board for multi-rotor aircraft (Tricopters, Quadcopters, Hexcoptersetc). Its purpose is to stabilize the aircraft during flight. In order to accomplish this it takes the signal from the 6050MPU gyro/acc (roll, pitch and yaw) then passes the signal to the Atmega644PA IC. The Atmega644PA IC unit then processes these signals according the users selected firmware and passes control signals to the installed Electronic Speed Controllers (ESCs). These signals instruct the ESCs to make fine adjustments to the motors rotational speed which in turn stabilizes multi-rotor craft.

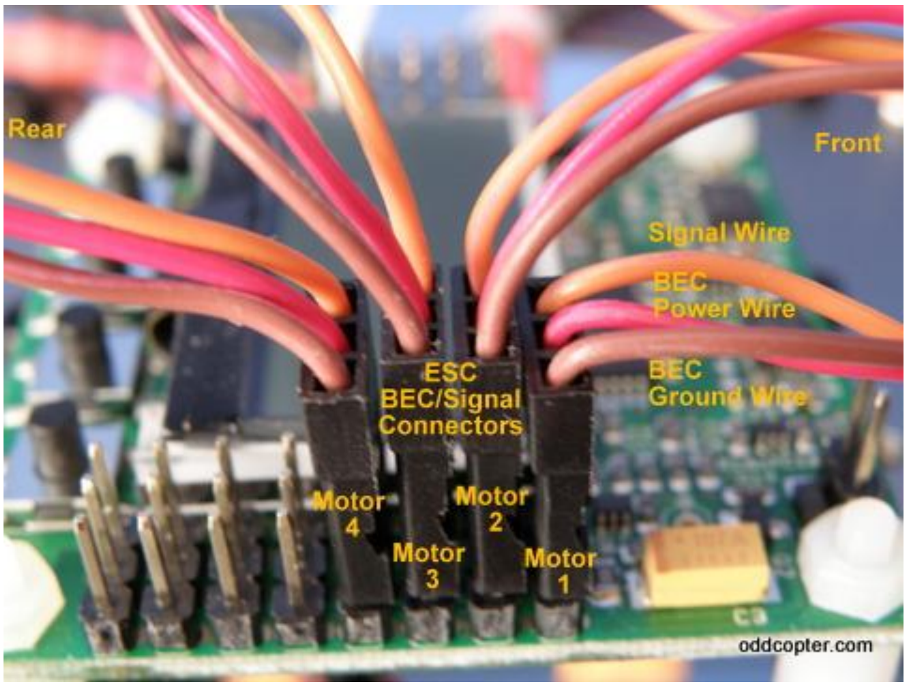
The HobbyKing KK2.1.5 Multi-Rotor control board also uses signals from radio systems receiver (Rx) and passes these signals to the Atmega644PA IC via the aileron, elevator, throttle and rudder inputs. Once this information has been processed the IC will send varying signals to the ESCs which in turn adjust the rotational speed of each motor to induce controlled flight (up, down, backwards, forwards, left, right, yaw).

The Flight Controller Board must always have a source of +5v from an ESC, either one of the motors ESC or from a separate unit feeding the Receiver. If each ESC has a BEC (normal unless OPTO types) then it may be necessary to remove the power feed from the other ESC, usually by cutting the power line (RED) Cable on the other ESC.

The models which are supported by the KK 2.1 board are as follows:

Dualcopter, Tricopter, Y6, Quadcopter +, Quadcopter X, Hexcopter +, Hexcopter X, Octocopter +, Octocopter X, X8 +, X8 X, H8, H6, V8, V6, Aero 1S Aileron, Aero 2S Aileron, Flying Wing, Singlecopter 2M 2S, Singlecopter 1M 4S.

The model which we are using is the **Quadcopter X** model.



ARDUINO UNO:

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, 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 a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

Specifications:

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage	7-12V
Input Voltage	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

The power pins are as follows:

• VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

• 5V. The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

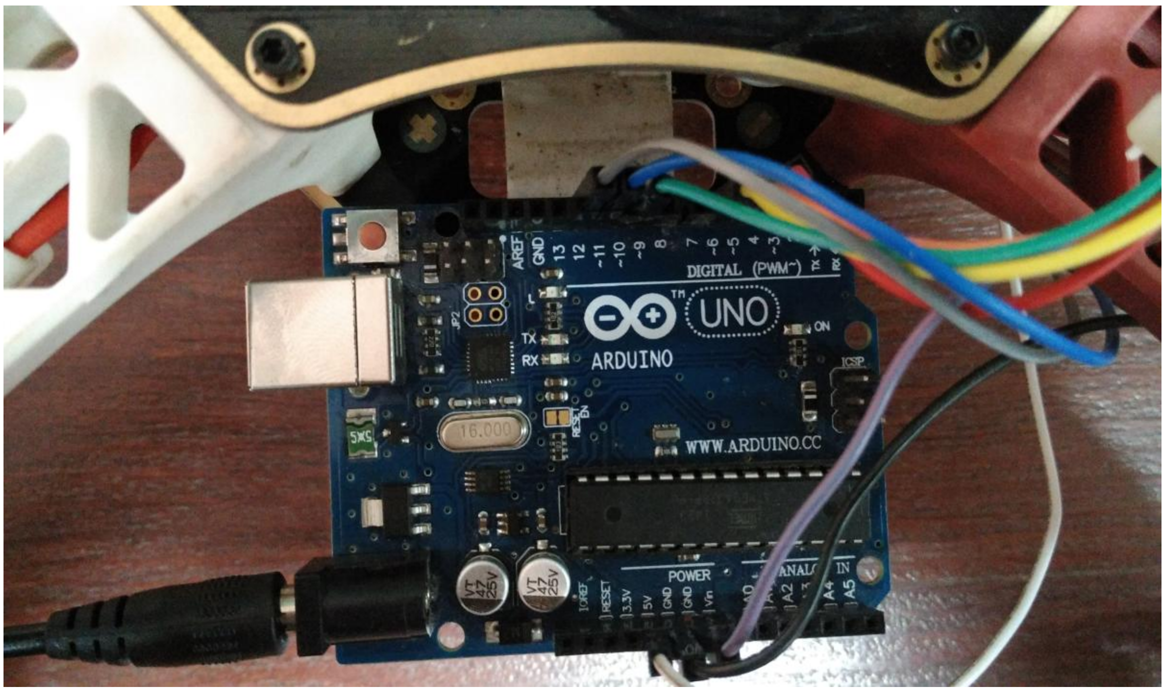
• 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

• GND. Ground pins.

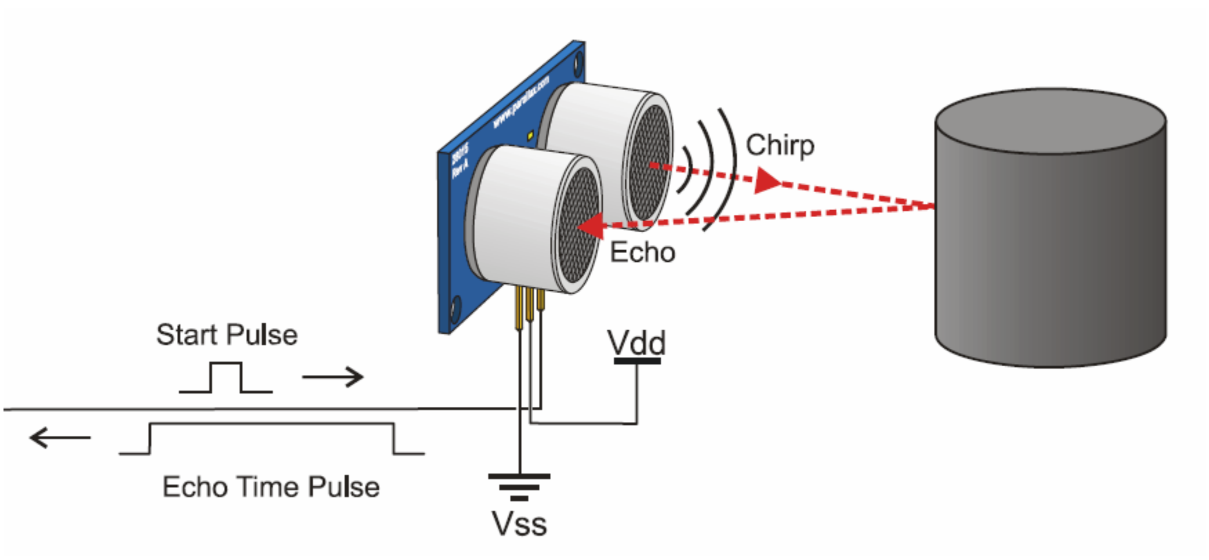
The pins which we have used are as follows:

PWM pins 3, 5, 6 and 9 for the four signals aileron, elevator, throttle and rudder.

Pins 11 and 12 for ECHO and TRIG pins of the ultrasonic distance sensor.



ULTRASONIC DISTANCE SENSOR:

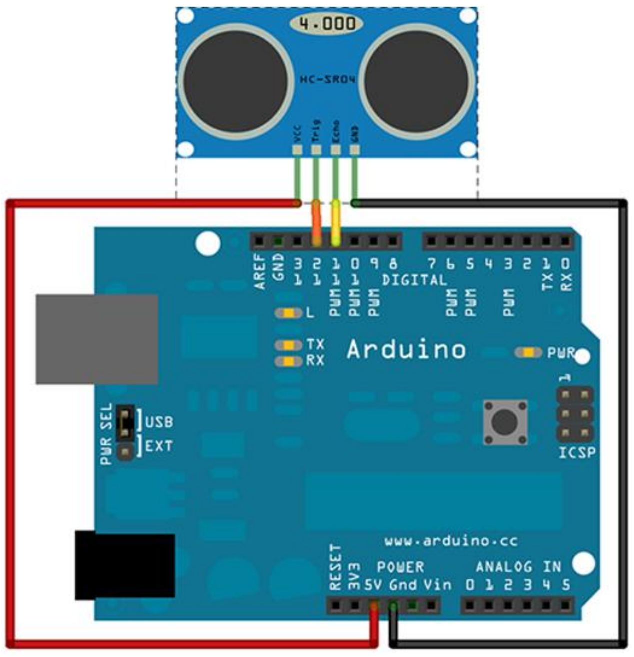


Ultrasonic sensors have an acoustic transducer which is vibrating at ultrasonic frequencies. The pulses are emitted in a cone-shaped beam and aimed at a target object. Pulses reflected by the target to the sensor are detected as echoes. The device measures the time delay between each emitted and echo pulse to accurately determine the sensor-to-target distance. HC SR04 Ultrasonic Rangefinder provides very short to long-range detection and ranging, in an incredibly small package. It can detect objects from 0-inches to 80-inches.

Specifications:

 Resolution of 1 inch
 20Hz reading rate
 42 kHz Ultrasonic sensor measures distance to objects
 RoHS Compliant
 Read from all 3 sensor outputs: Analog Voltage, RS232 Serial, Pulse Width
 Virtually no sensor dead zone, objects closer than 6 inches range as 6 inches
 Maximum Range of 254 inches (645 cm)
 Operates from 2.5-5.5V
 Low 2.0mA average current requirement
 Small, lightweight module.

INTERFACE WITH ARDUINO:



The pins 11 and 12 are connected to trig and echo pins of the HC SR04 sensor respectively.

Servo library:

Servos have to receive high-pulse control signals at regular intervals to keep turning. If the signal stops, so does the servo. Once the sketch uses the Servo library to set up the signal, it can move on to other code, like delays, checking sensors, etc. Meanwhile, the servo keeps turning because the Servo library keeps running in the background. It regularly interrupts the execution of other code to initiate those high pulses, doing it so quickly that it’s practically unnoticeable.

The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega.

writeMicroseconds:

Writes a value in microseconds () to the servo, controlling the shaft accordingly. On a standard servo, this will set the angle of the shaft. On standard servos a parameter value of 1000 is fully counter-clockwise, 2000 is fully clockwise, and 1500 is in the middle.

Syntax:

servo.writeMicroseconds()

Parameters:

servo: a variable of type Servo

LITHIUM POLYMER BATTERY:

Specifications:

Minimum Capacity:2200 mAh

Configuration:11.1 V/ 3 cell

Constant discharge:25C

Peak discharge:35C

Weight:188g

Size:105x33x24 mm

A lithium-ion polymer battery (abbreviated variously as LiPo, LIP, Li-poly) is a rechargeable battery of lithium-ion technology in a pouch format. LiPos come in a soft package or pouch, which makes them lighter but also less rigid. They offer high discharge rates and a high energy/storage weight ratio. From a single cell 100 mAh to 6 cells 22000 mAh battery packs are found.

We have used a 3 cell 2200 mAh lithium polymer battery.

BRUSHLESS DC MOTORS:

Specifications:

Battery:	2~4 cell/7.4~14.8V
RPM**:**	1100 kv
Max current**:**	18A
No load current**:**	1A
Max power:	336W
Internal resistance:	0.107 ohm
Weight:	70g (including connectors**)**
Diameter of shaft:	4mm
Dimensions:	28x36m
Prop size**:**	7.4V/11x7 14.85V/7x3
Max thrust**:**	1130g

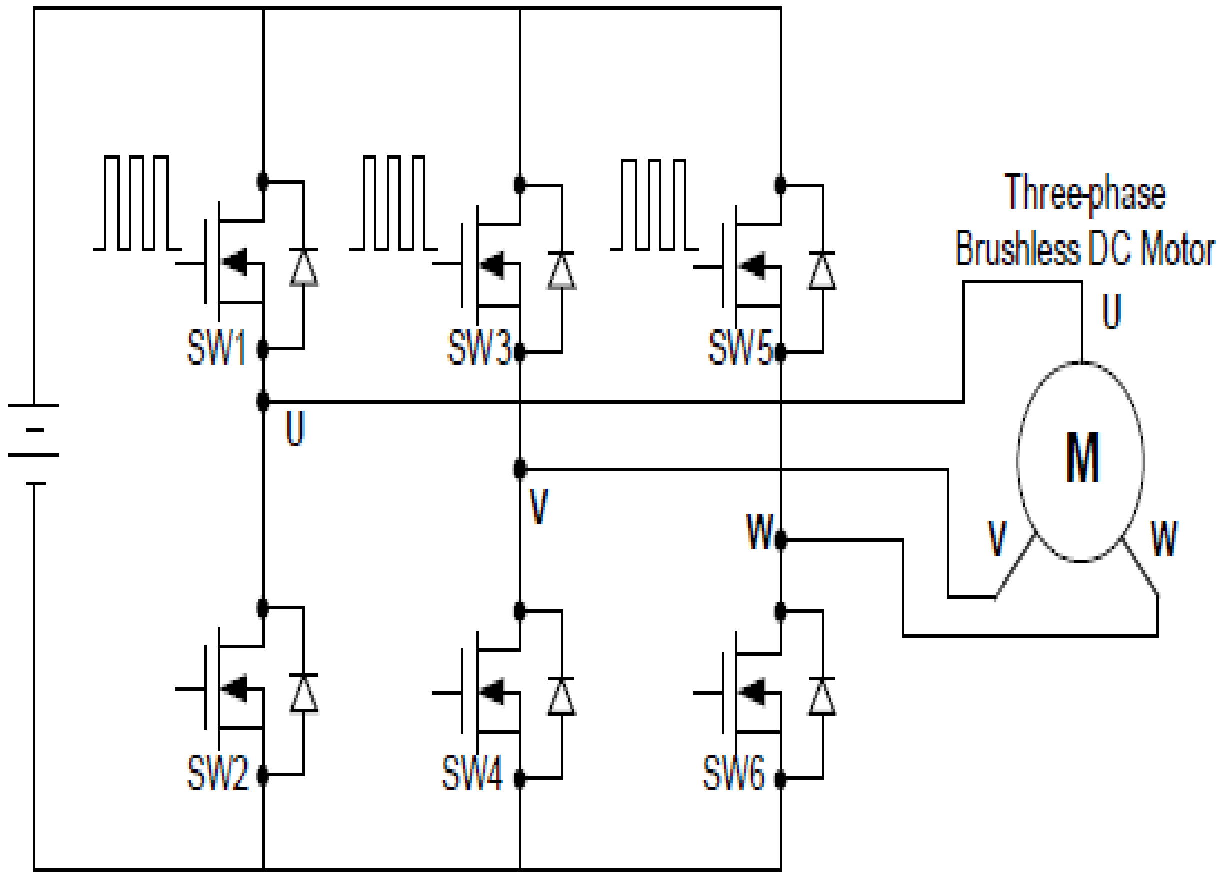
Brushless DC electric motors (BLDC motors, BL motors) are also known as electronically commutated motors (ECMs, EC motors). They are synchronous motorsthat are powered by a DC electric source via an integrated inverter/switching power supply, which produces an AC electric signal to drive the motor. They basically have a cylindrical shell of magnets that rotates on precision bearings around a core of tightly and neatly coiled wire. They have minimal friction.

The KV rating on a brushless dc motor is equal to RPM per volt applied to the motor. So a motor with a KV rating of 1000 KV will spin at 1000 RPM when 1 volt is applied.

We have used four brushless dc motors of 1100 kV. They have been placed along the corners of the frame.

WORKING PRINCIPLE:

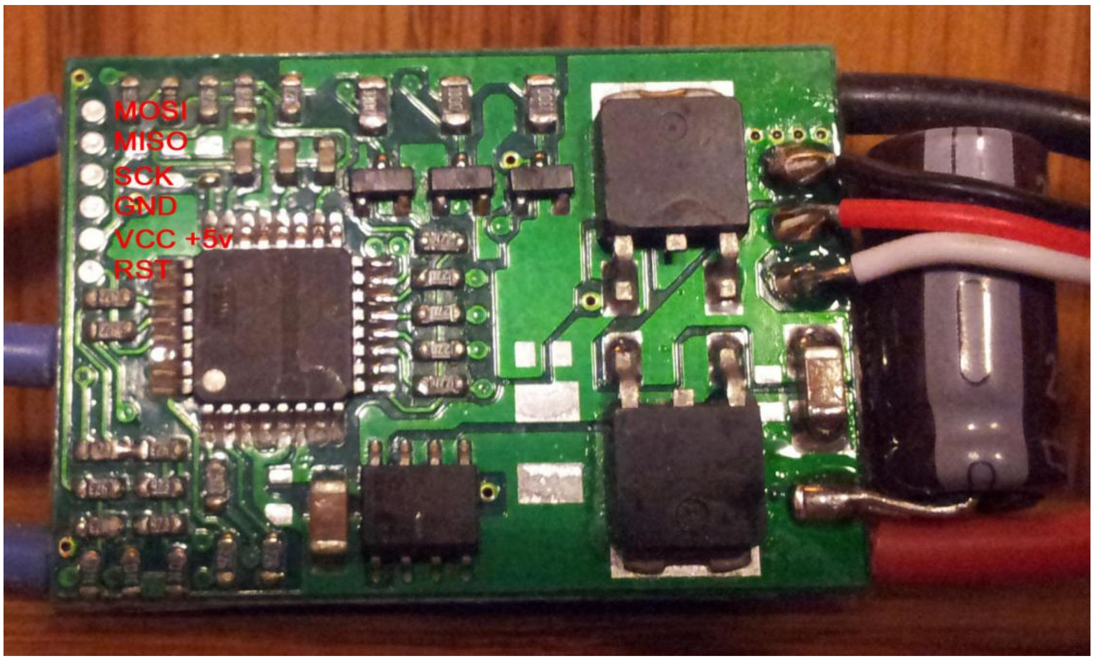
Brushless DC motors use electric switches to realize current commutation, and thus continuously rotate the motor. These electric switches are usually connected in an H-bridge structure for a single-phase BLDC motor, and a three-phase bridge structure for a three-phase BLDC motor shown in. Usually the high-side switches are controlled using pulse-width modulation (PWM), which converts a DC voltage into a modulated voltage, which easily and efficiently limits the start-up current, control speed and torque. Generally, raising the switching frequency increases PWM losses, though lowering the switching frequency limits the system’s bandwidth and can raise the ripple current pulses to the points where they become destructive or shut down the BLDC motor driver.



Comparison between Brushed DC and Brushless DC motors:

FEATURE	BLDC MOTOR	BRUSHED DC MOTOR	ADVANTAGE OF BLDC OVER BRUSHED DC
Commutation	Electronic commutation	Mechanical commutation	Electronic switches replaces mechanical devices
Efficiency	High	Moderate	Voltage drop on electronic device is smaller than that of brushes
Maintenance	Little	Periodic	No brushes/commutator maintenance
Output power	High	Low	Modern permanent magnet and no rotor losses
Speed/torque	Flat	Moderately flat	No brush friction to reduce torque
Dynamic response	Fast	Slow	Lower rotor inertia because of permanent magnets
Electric noise	Low	High	No arcs from brushes to generate noise causing EMI problems

ELECTRONIC SPEED CONTROLLERS:



An electronic speed control or ESC is an electronic circuit with the purpose to vary an electric motor's speed, its direction and possibly also to act as a dynamic brake. ESCs are often used on electrically powered radio controlled models, with the variety most often used for brushless motors essentially providing an electronically generated three-phase electric power low voltage source of energy for the motor.

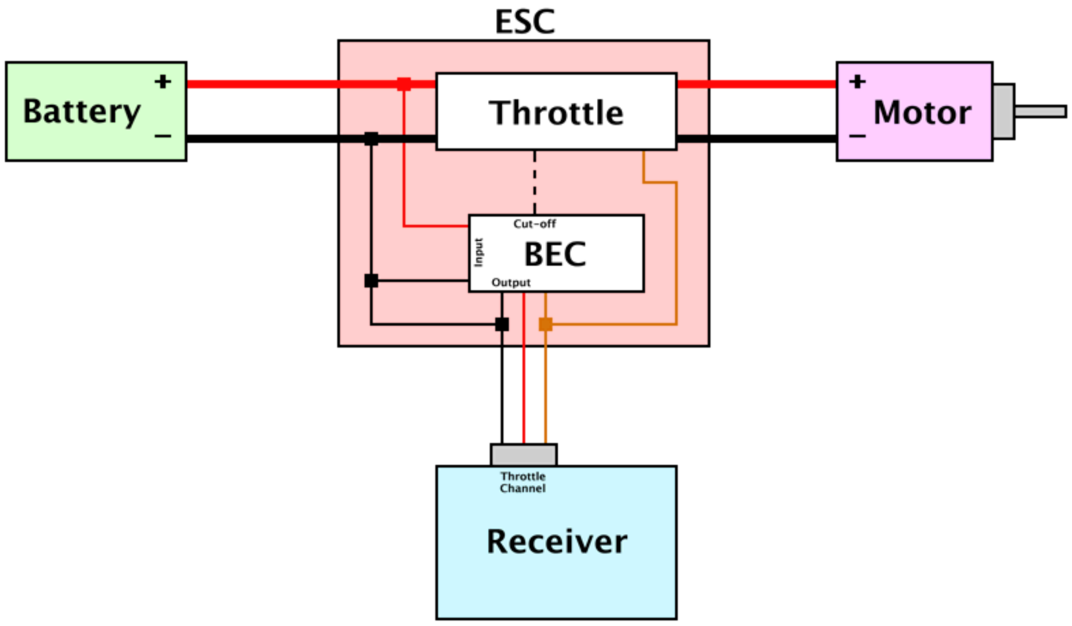
FUNCTIONING:

Regardless of the type used, an ESC interprets control information not as mechanical motion as would be the case of a servo, but rather in a way that varies the switching rate of a network of field effect transistors, or FETs. The rapid switching of the transistors is what causes the motor itself to emit its characteristic high-pitched whine, especially noticeable at lower speeds. It also allows much smoother and more precise variation of motor speed in a far more efficient manner than the mechanical type with a resistive coil and moving arm once in common use.

Most modern ESCs incorporate a battery eliminator circuit (or BEC) to regulate voltage for the receiver, removing the need for separate receiver batteries. BECs are usually either linear or switched mode voltage regulators.

DC ESCs in the broader sense are PWM controllers for electric motors. The ESC generally accepts a nominal 50 Hz PWM servo input signal whose pulse width varies from 1 ms to 2 ms.When supplied with a 1 ms width pulse at 50 Hz, the ESC responds by turning off the DC motor attached to its output. A 1.5 ms pulse-width input signal drives the motor at approximately half-speed. When presented with 2.0 ms input signal, the motor runs at full speed.

.BLOCK DIAGRAM OF AN ESC:



SOME OF THE IMPORTANT SUB-BLOCKS OF AN ESC:

- Battery Elimination Circuit

- Pulse Width Modulation Module

- 16 and 8 bit Timers

- Analog to Digital converters

- Watch Dog Timers

Digital control of BLDC motor:

The BLDC motor is driven by rectangular voltage strokes coupled with the given rotor position. The generated stator flux interacts with the rotor flux, which is generated by a rotor magnet, defines the torque and thus the speed of the motor. The voltage strokes must be properly applied to the two phases of the three-phase winding system so that the angle between the stator flux and the rotor flux is kept close to 90°, to get the maximum generated torque. Due to this fact, the motor requires electronic control based on actual rotor angle position for proper operation.

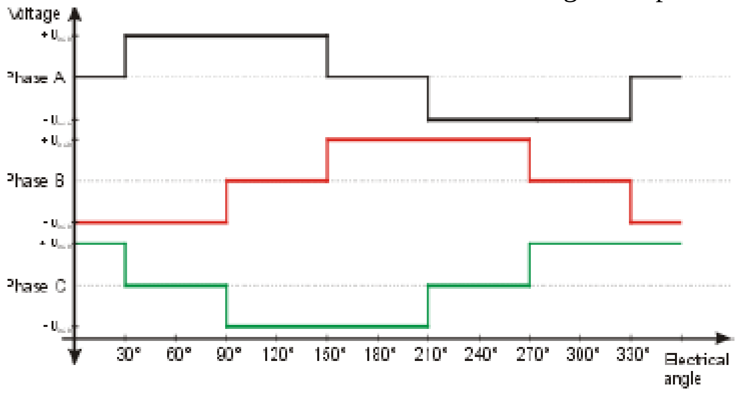


Figure 2. Voltage strokes applied to the three-phase BLDC motor

FRAME:

Every quadcopter or other multirotor aircraft needs a frame to house all the other components. Things to consider here are weight, size, and materials. They're strong, light, and have a sensible configuration including a built-in power distribution board (PDB) that allows for a clean and easy build.

PROPELLERS:

A quadcopter has four propellers, two "normal" propellers that spin counter-clockwise, and two "pusher" propellers that spin clockwise. The pusher propellers will usually be labelled with an 'R' after the size. We are using 10 x 4.8 propellers.

WORKING MECHANISM:

- The four motors are connected to their respective speed controllers.

- The speed controllers are further connected to the receiver pins on the KK flight controller board.

- The propellers are connected to the four motors in such a way that one pair of diagonal propellers rotate in clockwise direction and other diagonal pair of propellers rotate in counter clockwise direction which creates the necessary thrust to lift the lander above the ground level.

-In order to start the motors the flight controller board has to be initiated by sending respective pwm values to receiver pins.

-Once the board is initiated different PWM values are sent for the lift off from the Arduino board.

-The lander is made to reach a specific height and then the descent phase starts which is controlled by the distance sensor interfaced to the Arduino.

- Finally it is ensured that the lander lands safely on the ground by giving the necessary PWM values.

ALGORITHM:

1. Four objects of class Servo which is predefined in Arduino are created for four different receiver pins namely Aileron(3), Rudder(9), Elevator(5) and Throttle(6).
2. One more object of class UltraSense is created for the pinsTrig(12) and Echo(11) on the ultrasonic distance sensor.
3. Arming the KK board is done by sending the respective pwm values
   Aileron – 1500us
   Elevator – 1500us
   Throttle – 1000us
   Rudder – 1000us
4. The Lander is made to lift off by giving increasing values of pwm to the Throttle pin as follows:
   1000us – 5 seconds
   1200us – 6 seconds
   1400us – 5 seconds
   1600us – 6 seconds (Lift off takes place at this value)
   1570us – 3 seconds (To bring the lander to a height within the sensor range)
5. Once the lander is brought back to the range of the sensor, its functioning begins.

6. For different height ranges different pwm values are given to the Throttle pin as follows:

Height Range (Inches)	PWM Values for Throttle Pin(In Microseconds)
<= 80	1570 us
<= 60	1560 us
<= 40	1550 us
<= 20	1555 us
<= 10	1565 us
<= 5	1000 us(Switch the motors Off)

ARDUINO CODE:

- ARMING

```
#include<Servo.h>

Servo myservo1;//SERVO OBJECT AILERON
Servo myservo2;//SERVO OBJECT ELEVATOR
Servo myservo3;//SERVO OBJECT THROTTLE
Servo myservo4;// SERVO OBJECT RUDDER

void setup()
 {

myservo2.attach(5);//elevator
myservo1.attach(3);//aileron
myservo3.attach(6);//throttle
myservo4.attach(9);//rudder

}

void loop()

 {

myservo1.writeMicroseconds(1500);
myservo2.writeMicroseconds(1500);
myservo3.writeMicroseconds(1000);
myservo4.writeMicroseconds(1000);
}
```

➢ DISTANCE SENSING AND LANDING

```
#include<Servo.h>
#include <ultra_sense.h>
int a=1,b=1;
inti=1;
int dis;

Trans t;// Object to set the ultrasonic sensor parameters

Servo myservo1;//SERVO OBJECT AILERON
Servo myservo2;//SERVO OBJECT ELEVATOR
Servo myservo3;//SERVO OBJECT TH  ROTTLE
Servo myservo4;// SERVO OBJECT RUDDER


void setup()
{

t.Trig=12; //Set Trig pin as digital pin 12, can be varied
t.Echo=11; //Set Echo pin as digital pin 11, can be varied
Serial.begin(9600);

myservo2.attach(5);//elevator
myservo1.attach(3);//aileron
myservo3.attach(6);//throttle
myservo4.attach(9);//rudder

}
void loop()
{
if(i==1)
{



//=====================INITIALISING======================
===========================================

if(a==1)
  {
myservo1.writeMicroseconds(1500);
myservo2.writeMicroseconds(1500);
myservo3.writeMicroseconds(1000);
myservo4.writeMicroseconds(1500);
delay(6000);
myservo1.writeMicroseconds(1500);
```

```
myservo2.writeMicroseconds(1500);
myservo3.writeMicroseconds(1200);
myservo4.writeMicroseconds(1500);
delay(6000);

myservo1.writeMicroseconds(1500);
myservo2.writeMicroseconds(1500);
myservo3.writeMicroseconds(1400);
myservo4.writeMicroseconds(1500);
delay(6000);
myservo1.writeMicroseconds(1500);
myservo2.writeMicroseconds(1500);
myservo3.writeMicroseconds(1600);
myservo4.writeMicroseconds(1500);
delay(6000);

myservo1.writeMicroseconds(1500);
myservo2.writeMicroseconds(1500);
myservo3.writeMicroseconds(1570);
myservo4.writeMicroseconds(1500);
delay(3000);


a=a+1;
}
// ==================SENSING THE HEIGHT ============

t.begin();
  //t.print("inches");
delay(100);  dis=t.IN_INCHES();

Serial.println(dis);



   // ========FOR VARIOUS RANGES OF THE DISTANCES ===



if(dis<80)
    {
myservo1.writeMicroseconds(1500);
myservo2.writeMicroseconds(1500);
myservo3.writeMicroseconds(1570);
myservo4.writeMicroseconds(1500);
    }
if(dis<60)
    {
```

```
myservo1.writeMicroseconds(1500);
myservo2.writeMicroseconds(1500);
myservo3.writeMicroseconds(1560);
myservo4.writeMicroseconds(1500);
   }

if(dis<40)
 {
myservo1.writeMicroseconds(1500);
myservo2.writeMicroseconds(1500);
myservo3.writeMicroseconds(1550);
myservo4.writeMicroseconds(1500);
 }

if(dis<20)
 {
myservo1.writeMicroseconds(1500);
myservo2.writeMicroseconds(1500);
myservo3.writeMicroseconds(1555);
myservo4.writeMicroseconds(1500);
 }
if(dis<10)
 {
myservo1.writeMicroseconds(1500);
myservo2.writeMicroseconds(1500);
myservo3.writeMicroseconds(1565);
myservo4.writeMicroseconds(1500);
 }
if(dis<5&&dis!=0)
 {
myservo1.writeMicroseconds(1500);
myservo2.writeMicroseconds(1500);
myservo3.writeMicroseconds(1000);
myservo4.writeMicroseconds(1500);
i=i+1;
 }


  }

}
```

BIBLIOGRAPHY:

- Programming Arduino Getting Started with Sketches by Prof. Simon Monk
- Quad copters and Drones: A Beginner's Guide to Successfully Flying and choosing the Right Drone Kindle Edition by Prof.Mark Smith
- www.hobbyking.com
- www.turnigy.com/motors
- www.instructables.com
- www.google.com
- www.wikipedia.org

DATASHEETS