Aim

To assemble UAV (quadcopter) using electronic components and control it with Arduino Uno, and add gimbal for stable photography and videography on drone using servo motors.

Components

4 BLDC motors, 4 ESCs, 3300 mAh 3 cell LiPo battery, RC transmitter, 3D printed frame of quadcopter, Arduino Uno (flight controller), MPU6050 gyro and accelerometer sensor

Theory

Components required

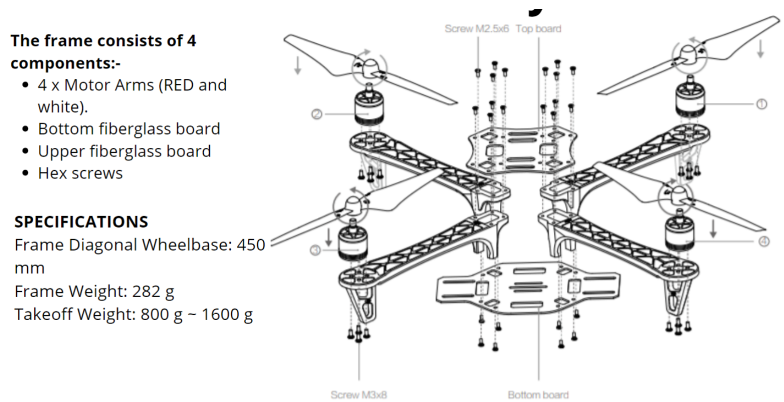
1. Brushless DC motor:
    - BLDC doesn't have brushes
    - Uses **electronic commutation** (eliminates mechanically torn brushes)
    - Two parts: motor and stator
2. Electronic speed controllers (ESCs):
    - Devices that switch power between the three combinations of two of the three poles of a BLDC
    - They take power directly from the Battery and supply it the power in accordance with the **PWM signal** provided.
    - Most ESCs need to be calibrated so that they know the minimum and maximum PWM values that the flight controller will send.
3. Transmitter and reciever:
    - RC transmitter has 4-6 channels (for pilot to use in changing direction, throttle, etc.)
    - Receiver is placed on the drone and decodes the data sent from the transmitter
4. Li-polymer battery
5. Flight controller (FC):
    - Circuit boards that have particular sensors such as **gyroscopes** and **accelerometers** and several other insignificant but useful sensors such as barometer,compass, etc.

PID controller

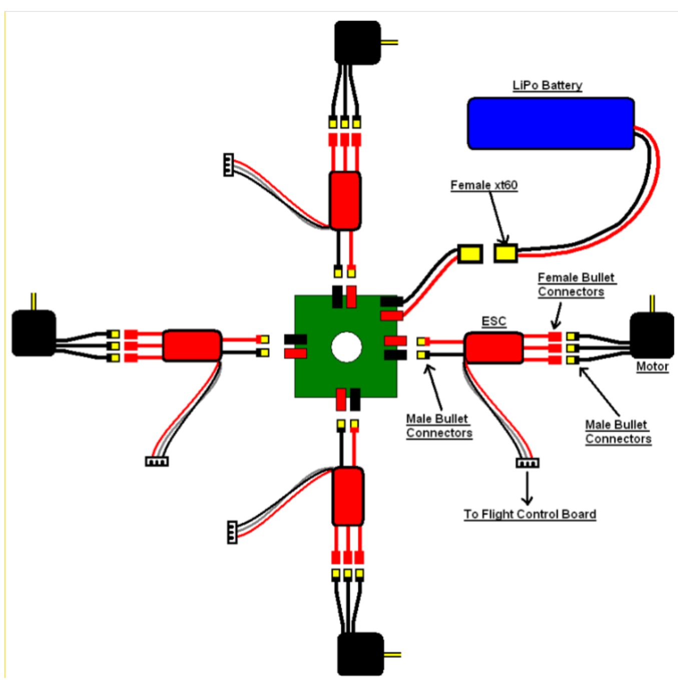
Control loop mechanism which continuously calculates the error value so that the drone can achieve desired position.

1. Proportional (P): Proportional controller involves correcting the state of the plant proportional to the error generated. Kp is the gain constant for increasing or decreasing the aggressiveness of correction.
2. Integral (I): Effectively cumulates the error result from the "P" action to increase the correction factor. Ki is the gain constant for increasing or decreasing the aggressiveness of correction.
3. Derivative (D): Attempts to minimize the overshoot by slowing the correction factor applied as the target is approached.

Frame Assembly

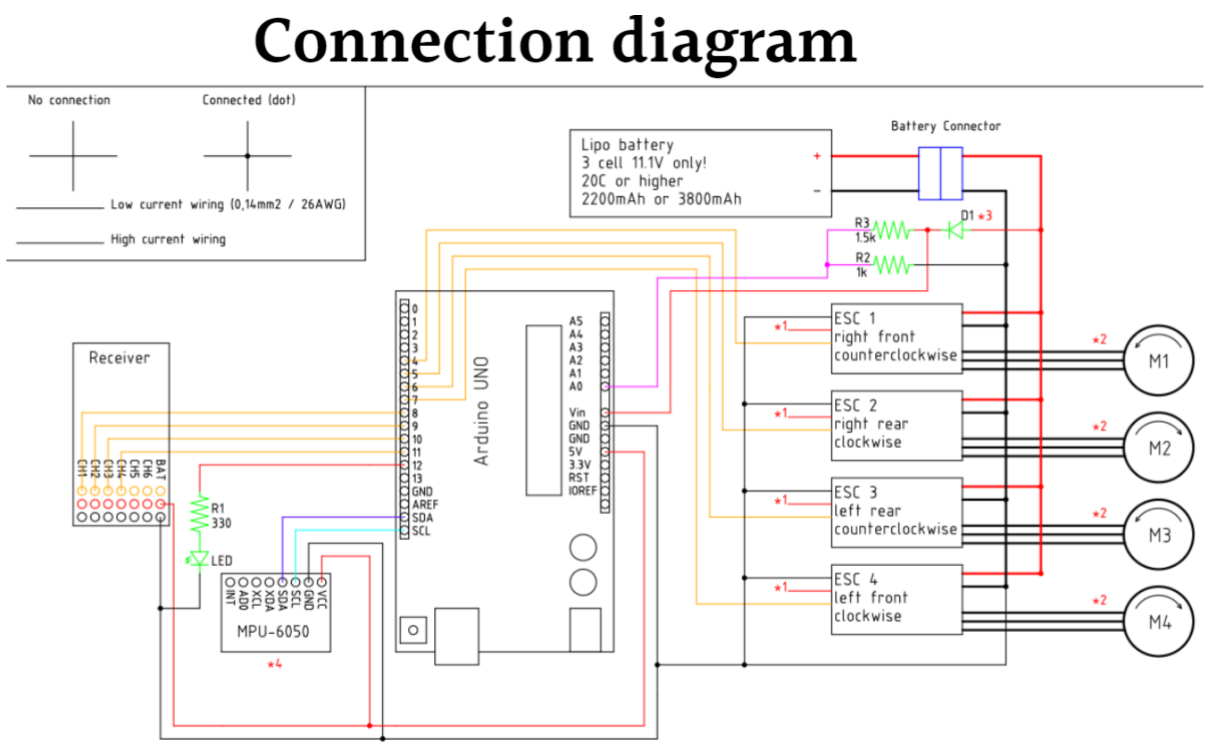


ESC and Motor Wiring



Flight controller Using Arduino Uno

Connections of components with Arduino



Basic layout of the code

1. Reading gyro angular rate data
2. Read receiver signals
3. Angle conversion of PWM signals from IMU
4. Calculating PID corrections
5. Calculate pulse for each ESC
6. Send PWM signals to each ESC
7. Repeat

(This loop takes 4 microseconds to run each time)

MPU6050

- MPU6050 is a 6 axis imu sensor with 3 axis gyroscope and3 axis accelerometer onboard
- It works on **i^2 c communication protocol**
- Gyroscope values are provided as a 16 bit digital value for the angular rate of each axis.
- Accelerometer values are also provided as a 16 bit digital value for the acceleration experienced on each axis.

I^2C on Arduino

- Add library to communicate with I^2C using:

```
#include <Wire.h>
```

- Pins on Arduino Uno: A4 (SDA), A5 (SCL)

Registers

It is important to know which registers need to be manipulated in order to receive a desired sensor value from the IMU, some of them are discussed here:

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	SELF_TEST_X	R/W	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	SELF_TEST_Z	R/W	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	SELF_TEST_A	R/W	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							

Calculations for Flight Controller

Gyro angle= gyro input *Delta t

$$\text{Total acceleration vector} = \sqrt{(ACC\_x)^2 + (ACC\_y)^2 + (ACC\_z)^2}$$

$$\text{Sin(Accelerometer pitch angle)} = \frac{ACC\_Y}{\sqrt{\text{total acceleration vector}}}$$

$$\text{Sin(Accelerometer roll angle)} = \frac{ACC\_X}{\sqrt{\text{total acceleration vector}}}$$

Total angle output = gyro angle*(96%-98%) +Accelerometer angle *(2%-4%)

Self leveling

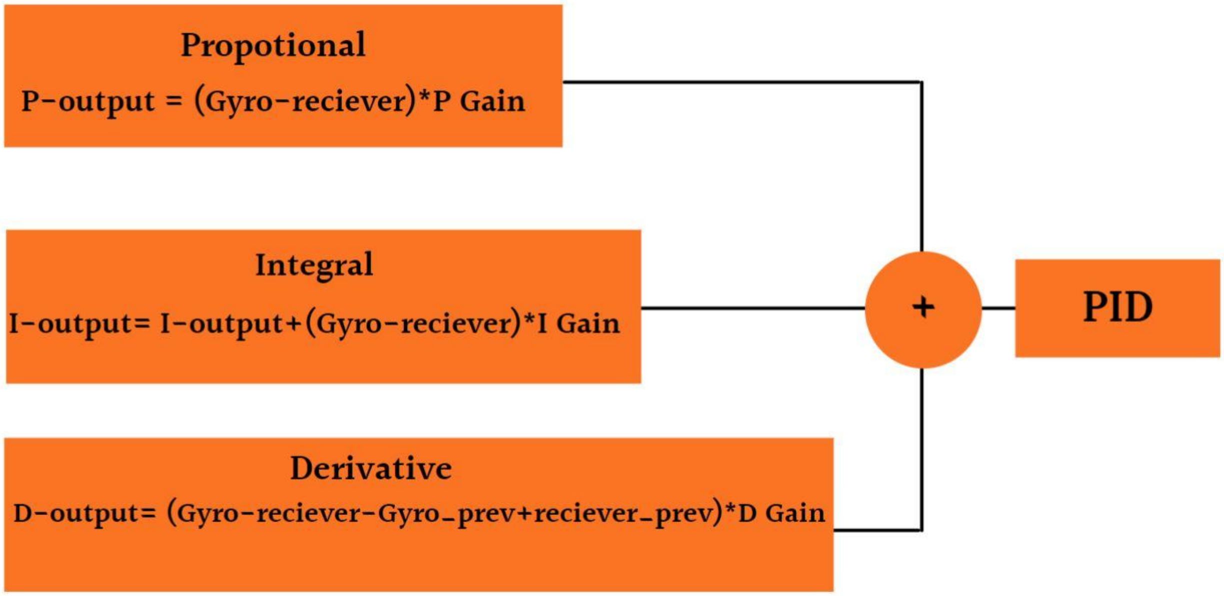
pitch level adjust = angle pitch * 15
roll level adjust = angle roll * 15

pid_roll_setpoint = 0

pid_roll_setpoint = receiver input channel 1 - 1500

pid_roll_setpoint = pid roll setpoint - roll level adjust
pid_roll_setpoint /= 3.0

PID loop for Gyro

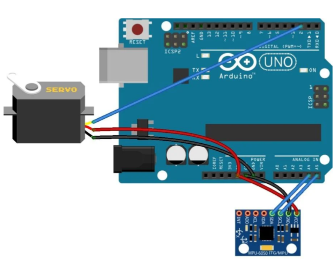


Coding on Arduino

```
pid_error_temp = gyro_roll_input - pid_roll_setpoint;

pid_i_mem_roll += pid_i_gain_roll * pid_error_temp;

pid_output_roll = pid_p_gain_roll * pid_error_temp + pid_i_mem_roll + pid_d_gain_roll *
(pid_error_temp - pid_last_roll_d_error);

pid_last_roll_d_error = pid_error_temp;
```

Controlling servo for Gimbal Using MPU6050

Connections of Components with Arduino



Coding on Arduino

Three libraries were used as:

```
#include <Wire.h>
#include <MPU6050.h>
#include <Servo.h>
```

Code:

```
#include <Wire.h>
#include <MPU6050.h>
#include <Servo.h>

Servo serv_motor;
int servo_pin = 2;

MPU6050 sensor ;
int16_t ax, ay, az ;
int16_t gx, gy, gz ;

void setup ( ){
        serv_motor.attach ( servo_pin );
        Wire.begin ( );
        Serial.begin  (9600);
}

void loop ( ) {
        sensor.getMotion6 (&ax, &ay, &az, &gx, &gy, &gz);
        ax = map (ax, -17000, 17000, 0, 180) ;
        Serial.println (ax);
        serv_motor.write (ax);
        delay (200);
}
```

Working

Quadcopter

1. Radio signals are sent by transmitter to the receiver

2. Receiver then transmits PWM signals to the flight controller (Arduino)
3. Arduino then runs a PID loop with correct calculations to attain the position as desired by the pilot
4. While doing this, the drone is stabilised by the gyro and accelerometer sensor (MPU 6050) in the flight controller
5. Arduino further sends PWM signals to the ESCs to wary the speeds of the BLDC motors and move the quadcopter accordingly

Result

The quadcopter was built successfully with the use of Arduino and a gimbal to mount camera on for stability in capturing better videos and photos.

References

1. Study material pdfs
2. Arduino project hub
3. Wikipedia
4. https://maker.pro/arduino/tutorial/how-to-control-a-servo-with-an-arduino-and-mpu6050