List of components

Arduino nano

Adxl345 gyroscope sensor

Servo motor

Arduino nano cable

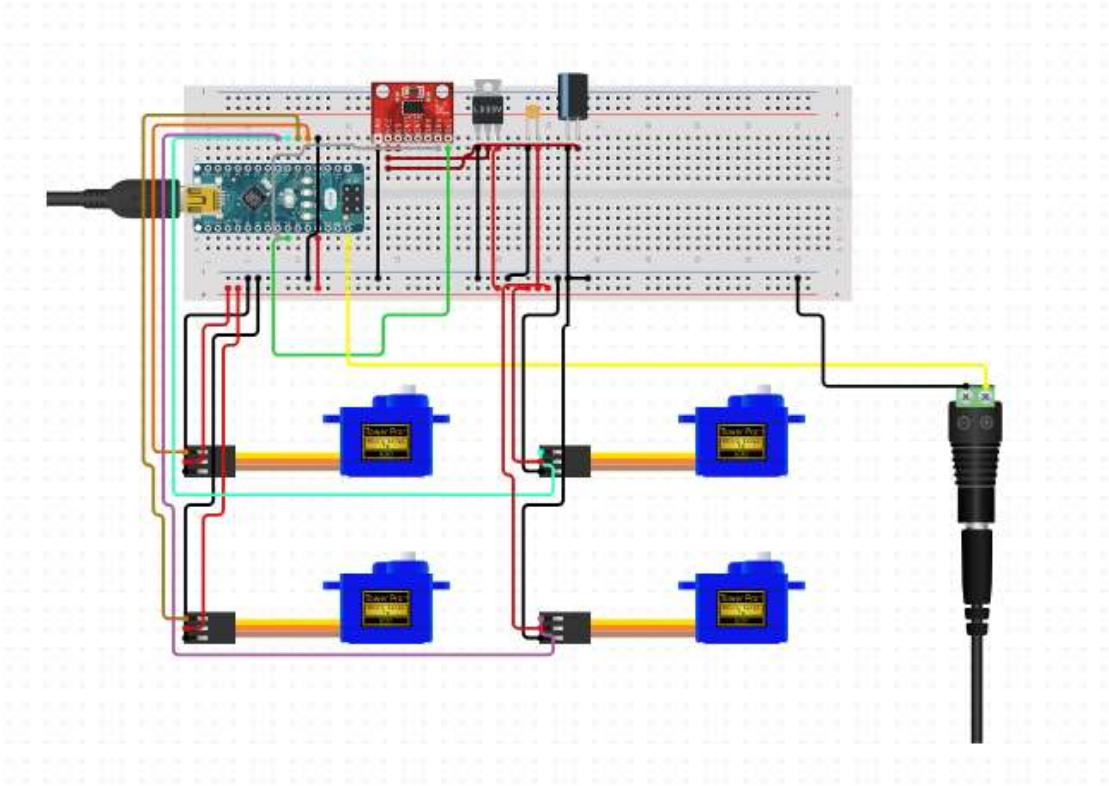
Jumper wires

Breadboard

14500 li-ion battery 600 mAh

14500 Battery charging board

Circuit diagram:



Here I didn’t use any capacitor and transistor I have only used a battery and to charge that I also used a charging board to make it light as possible I have used a AA sized li-ion battery

Code explanation :

1. Library Inclusion:

- `#include <Wire.h>`: This library allows communication over I2C.

- `#include <ADXL345.h>`: This library is specifically for the ADXL345 accelerometer.

- `#include <Servo.h>`: This library enables control of servo motors.

2. Global Variable Declaration:

- `Servo servo1`, `servo2`, `servo3`, `servo4`: These are servo motor objects to control the servo motors connected to pins 4, 5, 6, and 7 respectively on the Arduino Nano.

- `ADXL345 adxl`: This is an object representing the ADXL345 accelerometer.

- `int x, y, z`: These variables store the accelerometer readings for the X, Y, and Z axes respectively.

- `int mappedRawX, mappedRawY`: These variables store the mapped values of accelerometer readings for servo positions.

3. Setup Function (`setup`):

- `Serial.begin(9600)`: Initializes serial communication at a baud rate of 9600 for debugging purposes.

- `adxl.powerOn()`: Powers on the ADXL345 accelerometer.

- `servo1.attach(4)`, `servo2.attach(5)`, `servo3.attach(6)`, `servo4.attach(7)`: Attaches the servo motors to their respective pins on the Arduino Nano.

- `servo1.write(90)`, `servo2.write(90)`, `servo3.write(90)`, `servo4.write(90)`: Sets the initial positions of all servo motors to 90 degrees.

4. Loop Function (`loop`):

- `adxl.readAccel(&x, &y, &z)`: Reads the accelerometer values for the X, Y, and Z axes and stores them in variables `x`, `y`, and `z`.

- `mappedRawX = map(x, -1000, 1000, 0, 180)`, `mappedRawY = map(y, -1000, 1000, 0, 180)`: Maps the accelerometer readings to servo positions. The mapping function scales the input values from the accelerometer's range (-1000 to 1000) to the servo's range (0 to 180 degrees).

- `int servo1Pos = constrain(90 + mappedRawX, 0, 180)`, `int servo2Pos = constrain(90 - mappedRawX, 0, 180)`, `int servo3Pos = constrain(90 + mappedRawY, 0, 180)`, `int servo4Pos

= constrain(90 - mappedRawY, 0, 180)`: Calculates the servo positions based on the mapped values. The `constrain` function ensures that the servo positions stay within the valid range of 0 to 180 degrees.

- `servo1.write(servo1Pos)`, `servo2.write(servo2Pos)`, `servo3.write(servo3Pos)`, `servo4.write(servo4Pos)`: Sets the servo positions according to the calculated values.

- `Serial.print` and `Serial.println`: Outputs the servo positions to the serial monitor for debugging purposes.

Methodology:

1. Hardware Setup

- Connect ADXL345 Sensor: Connect the ADXL345 accelerometer sensor to the Arduino Nano. This typically involves connecting power (VCC and GND) and communication lines (SDA and SCL) between the sensor and the Arduino Nano.

- Connect Servo Motors: Connect the servo motors to the Arduino Nano. Each servo motor should have connections for power (usually connected to the 5V pin on the Arduino Nano), ground, and control signal (connected to individual digital pins such as D4, D5, D6, and D7 on the Arduino Nano).

2. Software Setup

- Install Libraries: Install the necessary libraries for the ADXL345 sensor (`Wire.h` and `ADXL345.h`) and the Servo library (`Servo.h`) in your Arduino IDE.

- Include Libraries: In your Arduino sketch, include the required libraries using `#include` directives at the beginning of your code.

3. Code Implementation

Initialization

- Declare Objects: Create instances of the ADXL345 (`adxl`) and Servo (`servo1`, `servo2`, `servo3`, `servo4`) classes.

- Setup Function: In the `setup` function, initialize serial communication (`Serial.begin(9600)`), power on the ADXL345 sensor (`adxl.powerOn()`), attach the servo motors to their respective pins (`servo1.attach(4)`, `servo2.attach(5)`, `servo3.attach(6)`, `servo4.attach(7)`), and set the

initial servo positions to 90 degrees (`servo1.write(90)`, `servo2.write(90)`, `servo3.write(90)`, `servo4.write(90)`).

Main Loop Execution

- Loop Function: In the `loop` function, continuously read accelerometer values (`adxl.readAccel(&x, &y, &z)`), map these values to servo positions for roll and pitch (`map` function), calculate servo positions (`servo1Pos`, `servo2Pos`, `servo3Pos`, `servo4Pos`), constrain servo positions within limits (`constrain` function), and write the servo positions to the servo motors (`servo1.write(servo1Pos)`, `servo2.write(servo2Pos)`, `servo3.write(servo3Pos)`, `servo4.write(servo4Pos)`).

- Serial Output: Optionally, print servo positions to the serial monitor for debugging (`Serial.print` and `Serial.println` statements).

4. Testing and Calibration

- Upload Code: Upload the code to the Arduino Nano.

- Monitor Output: Open the serial monitor in the Arduino IDE to monitor the servo positions and ensure they are adjusting correctly based on the accelerometer readings.

- Calibration: If needed, calibrate the system by adjusting mapping ranges, servo position limits, and sensor orientation to achieve desired control over the rocket's fins.

5. Integration and Operation

- Integration: Integrate the Arduino Nano, ADXL345 sensor, servo motors, and any other necessary components into the rocket system.

- Operation: Power on the system and observe how the servo motors adjust the rocket fins' positions based on the accelerometer readings, providing stabilization and control during flight.

6. Testing and Optimization

- Testing: Conduct thorough testing of the system to ensure proper functionality and stability during different conditions.

- Optimization: Fine-tune the code, sensor calibration, and servo motor settings to optimize performance and achieve precise control over the rocket's orientation and stability.