Víctor Fernando Pérez García

Blimp Technical Report

Project overview

The project consists of controlling a blimp base by switching from a remote controller to a ground base system such a computer or other controller.

Equipment

- **Futaba R136F Receiver**
- **Quantum Reverse Electronic Speed Controller (ESC).**
- **Veloci Compso ESC**
- **Hitec HS-422 Servo Motor**
- **3 * DC motors**
- **3200 mAh LIPo Battery**

Code

```
#include <Servo.h>

// create servo object to control a servo
Servo servo1;
Servo servo2;
Servo servo3;

// Time condition of switch in miliseconds
int timec = 2000;

// Time setting
unsigned long previousMillis=0;

  // Receiver Outputs to Servos
int Rc1 = 2;  // switch signal
int Rc2 = 3;  // RPM (9)
int Rc3 = 4;  // Horizontal Servo (10)
int Rc4 = 5;  // Tlt motor (11)

//SoftwareSerial mySerial(7, 8); // RX, TX
```

```
void setup()
{
  Serial.begin(115200);

// Servos Outputs
  servo1.attach(9);
  servo2.attach(10);
  servo3.attach(11);

  // Setting the Input and Output in the ports
  pinMode(Rc1, INPUT);
  pinMode(Rc2, INPUT);
  pinMode(Rc3, INPUT);
  pinMode(Rc4, INPUT);

 // Now set up two tasks to run independently.
}
// Case value
int a;

volatile int s1 ;
volatile int s2 ;
volatile int s3 ;
volatile int s4 ;

int data[15];
 int k1;
   int k2;
   int k3;
   int k4;
   int r1;
   int r2;
   int r3;
   int r4;

   int i;
void loop(){
volatile int currentMillis= millis();
 s1 = pulseIn(Rc1,HIGH); // switch 1100<s1<900
    s2 = pulseIn(Rc2,HIGH); // switch 1100<s1<900
    s3 = pulseIn(Rc3,HIGH); // switch 1100<s1<900
    s4 = pulseIn(Rc4,HIGH); // switch 1100<s1<900

if (s1<=1200){
   if ((currentMillis - previousMillis) >= timec) {
// Reset timing
a=1;
  }}
   else if (s1>=1800){
   if ((currentMillis - previousMillis) >= timec) {
a=2;
}}
else {  Serial.println(a); previousMillis=currentMillis;}


  switch (a) {
//Remote controller
```

```
  case 1:
    Serial.print("principal motor power "); Serial.print(( s2-1000)*.1);
Serial.print(" %, Servo Angle "); Serial.print( (s3-1000)*.1); Serial.print("
% , Lateral motor power "); Serial.print((s4-1000)*.1);  Serial.print(" % ");
 servo1.write(s2);  // micros for rpm 1000-2000
 servo2.write(s3*2-1500);    // micros for rpm 1000-2000
 servo3.write(s4);    // milli for servo 0-180 (1000-2000)
 break;

//Workstation, another controller
  case 2:
if (Serial.available() > 3)
{
    for (int i=0; i < 9 ; i++)
    {
        data[i] = Serial.read();
    }
    Serial.flush();
}

String string1= String(data[0]-48);
String string2= String(data[1]-48);
String string3;
  String string4= String(data[3]-48);
String string5= String(data[4]-48);
String string6;
  String string7= String(data[6]-48);
String string8= String(data[7]-48);
String string9;

string3=string1+string2;
  string6=string4+string5;
  string9=string7+string8;


   r1= string3.toInt();
     r2= string6.toInt();
     r3= string9.toInt();

if (r1>=0 ){
   k1= r1;}
  if( r2>=0){
     k2= r2;}
  if( r3>=0){
     k3= r3;}

Serial.print(k1); Serial.print("% "); Serial.print(k2); Serial.print("% ");
   Serial.print(k3); Serial.print("% ");
   servo1.write(1000*k1/100+1000);  // micros for rpm 1000-2000
 servo2.write(1000*k2/100+1000);    // micros for rpm 1000-2000
 servo3.write(1000*k3/100+1000);    // milli for servo 1000-2000
    }}
```

Assembling Scheme

Víctor Fernando Pérez García

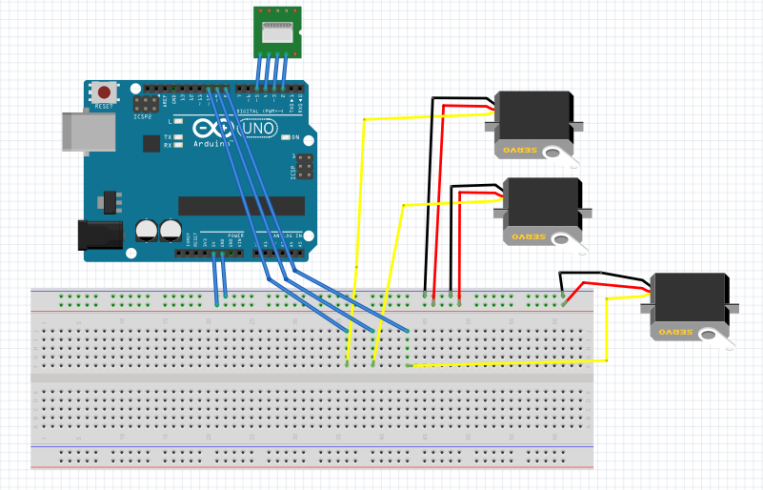

Fig. 1. Breadboard design.

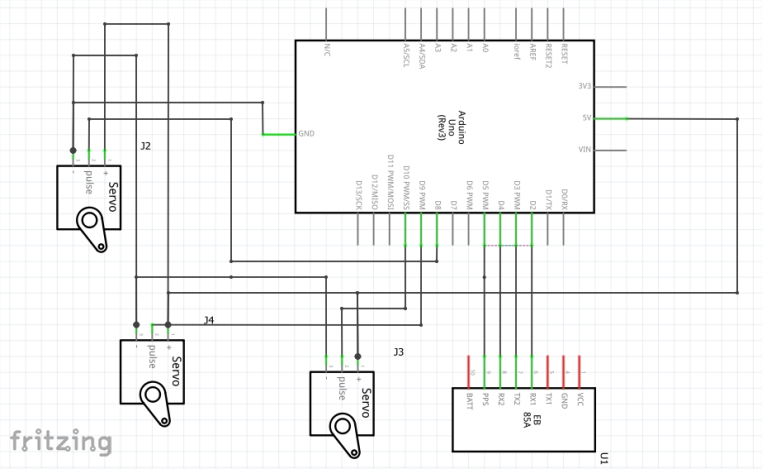

Fig. 2. Schematic design.

Where two servos are visual representations of the ESC's.

The visual representation means the following physical equalities, Rx1=Rx1, Tx2=Rx2, Rx2=Rx3, and PPS=Rx4.

Results

1. At first instance the output had time delay due to the pulse inputs of the remote controlled were stored in the stack as global variables, but after changing them to "volatile int" format they were stored at the heap i.e. ram memory as local variables making the process feasible and efficient.
2. The Time difference between the remote controller instructions and the workstation its of .15 seconds.

Víctor Fernando Pérez García

Instructions

1. After turning on the remote controller the battery its connected to the device.
2. There are 2 modes, mode 1 consists in receiving the instructions directly from the RC Controller and mode 2 consists in the control by a workstation or second controller UART based Serial communication.
3. To activate mode 1 the right horizontal controller joystick most be moved to the right for two seconds.

Fig. 3. Mode 1 activation.

4. To activate mode 2 the right horizontal controller joystick most be moved to the left for two seconds.

Fig. 4. Mode 2 activation.

5. In mode 1, to control the principal motor, the left vertical joystick from the middle position till the top it's the control range for this motor (Fig. 5.).
   For the angle of rotation, the joystick is vertical in the right side (Fig. 6).
   And the Tail motor is the left horizontal joystick range, (Fig.7.).

Fig. 5. Principal motors control.

Fig. 6. Tail motor control.

Fig. 7. Servo motor angle control.

6. Mode 2, after activating it, 3 digits will spawn in the Serial interface of Arduino, “00% 00% 00%”, the first digit is the principal motor total percentage of capability, the second correspond to the angle percentage of rotation, and the third represent the tail motor percentage of capability.
   The principal motor range of operation it’s from %50 to 00%, where 00% it’s the maximum.
   For the angle of rotation, 50% correspond to 90°.
   And for the tail motor, for positive direction the range is from 50% increasing till 100%, and for negative direction, is from 50% increasing till 00%.

Useful sites

http://arduino.cc/