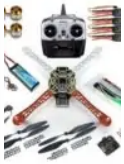
9 JULIO, 2018

Mando RC y receptor. Programación en Arduino (sketch)

Índice:

Páginas más vistas



DRONE CON ARDUINO
PASO A PASO
Arduino Drone | Material necesario y montaje de los componentes hardware

DRONE CON ARDUINO
PASO A PASO
Mando RC y receptor. Programación en Arduino (sketch)

DRONE CON ARDUINO
PASO A PASO
SENSORES
MS5611 módulo presión atmosférica. Resolución de 10 cm de altura | Arduino

DRONE CON ARDUINO
Probando el software completo

9. Software completo y esquema detallado *(código).*
10. Probando el Software completo antes de volar.
11. Como leer variables de *Arduino* en *Matlab (código).*
12. Los mejores *drones* de 2018 | Comparativa y guía de compra.

Mando radio control (RC)

Una de las maneras más fáciles y baratas de comunicarse con nuestro futuro *drone* es utilizar un mando *RC* y un receptor. Son fáciles de utilizar y se pueden conseguir por 30€ en *Ebay* en sus versiones más básicas (4 canales). Como todo en la electrónica, el mundo de los mandos RC es muy amplio pero para nuestro primer *drone* es suficiente. Dispondremos de un canal para controlar el *throttle*, otro para el *pitch*, uno más para el *roll* y el último para el *yaw*. He decidido **asociar los *sticks* con las siguientes funciones o movimientos del *drone*.** Recomiendo seguir el mismo criterio ya que es el mas extendido por ser el mas intuitivo a la hora de controlar el *drone.* Aquí una foto de mi mando:

Comprar mando RC en Amazon

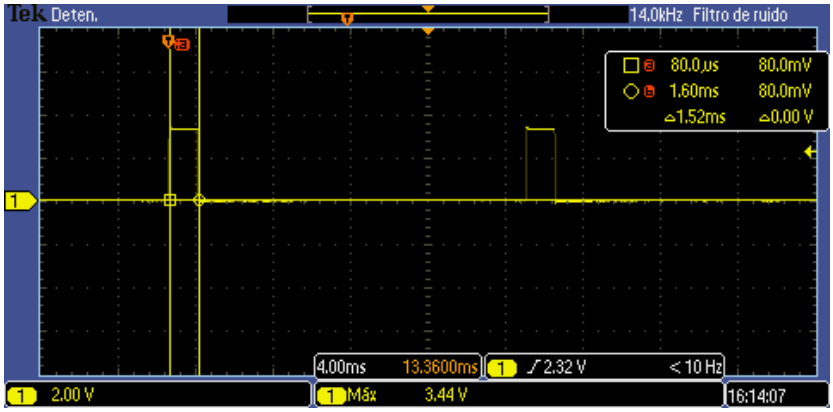
El funcionamiento de un mando *RC* es muy simple. **El movimiento de los *sticks* del mando es procesado y enviado por ondas de radio a nuestro receptor, que irá situado en el *frame* del**

DRONE CON ARDUINO
Calibración de hélices y motores Brushless para drones (sketch)

DRONE CON ARDUINO
PASO A PASO
Motores, ESC y su programación en Arduino (sketch)

DRONE CON ARDUINO
PASO A PASO
Leer señales PPM de radiocontrol con Arduino o STM32, paso a paso

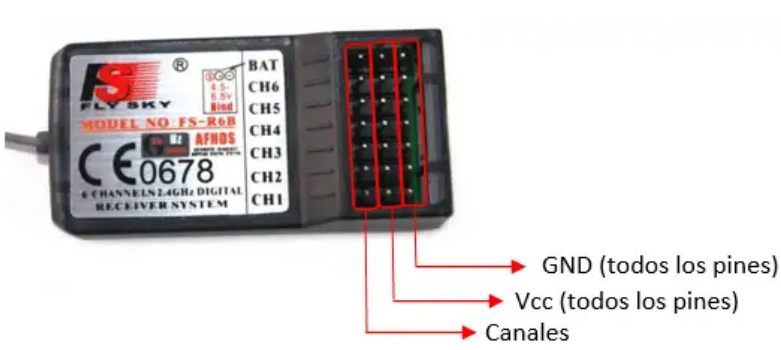
drone. Esta información se compone de señales *PWM* de 50*Hz* que varían en función de la posición de cada *stick*. Si accionamos al mínimo alguna de las palanca del mando, el ancho de pulso que recibamos en el receptor será de 1ms, y si la accionamos al máximo, el ancho del pulso será de 2ms (estos valores pueden ser diferentes en cada mando). Si dejamos la palanca en el punto central obtendremos pulsos de 1.5ms aproximadamente, siempre manteniendo una frecuencia de 50Hz. Esta información será enviada a la placa *Arduino* y se utilizara como referencia para mover el *drone*.



Como vemos en la imagen anterior, el mando no da exactamente pulsos de 1*ms* con el *stick* al mínimo, ni pulsos de exactamente 2*ms* subiéndolo a máximo. Estos pequeños errores de precisión del mando los corregiremos mas adelante al final de esta entrada.

Conexionado mando-*Arduino*

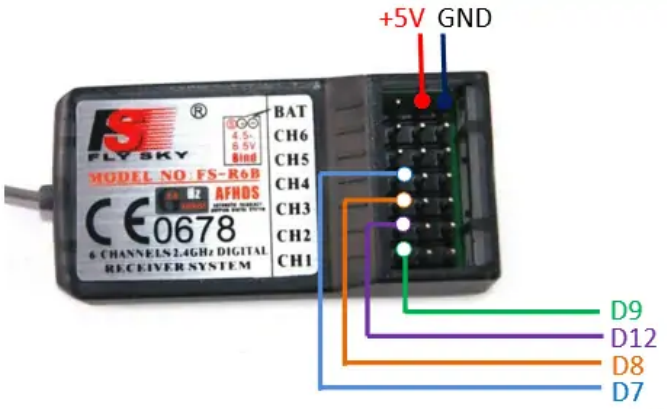
El receptor ira situado en el *frame* y es necesario alimentarlo y cablearlo de forma adecuada. Alimentaremos el receptor a +5V en cualquiera de los canales centrales, por ejemplo con la tensión de nuestra placa *Arduino*. Si hemos configurado bien la unión mando-receptor, al encender el mando debería encenderse un *led* rojo (en mi caso) que indica que funciona correctamente.



El conexionado necesario para ejecutar el código que os dejo al final de este artículo **se muestra a continuación**. Simplemente hay

que alimentar el receptor en cualquiera de los canales centrales y cablear las cuatro señales a las correspondientes entrada digitales, en mi caso:

- ***Pitch ⇒ CH2 ⇒ IN 12***
- ***Throttle ⇒ CH3 ⇒ IN 8***
- ***Roll ⇒ CH1 ⇒ IN 9***
- ***Yaw ⇒ CH4 ⇒ IN 7***



Antes de continuar me gustaría analizar un poco más en detalle el funcionamiento de mi receptor y la forma en la que genera los pulsos. La siguiente imagen muestra una captura sacada con osciloscopio donde visualizo las señales de *pitch, roll y yaw* (no he metido el canal de *throttle* por que mi osciloscopio solo tiene 3 canales). De esta imagen he sacado los siguientes datos:

- El receptor genera pulsos de 3.3V.
- **Los pulsos se generan de forma secuencial**, hasta que no termina un pulso no empieza el siguiente. **Nunca habrá más de un canal en estado HIGH cada vez.**

Algunos mandos de radiocontrol más avanzados pueden enviar la información de todos los canales utilizando una sola señal (*PPM*).

Tenéis todo lo necesario para leer señales PPM con Arduino en esta entrada.

Código Arduino *(Sketch)*

La forma más sencilla de leer este tipo de pulsos es utilizar la función *PulseIn* de *Arduino*. Lamentablemente esta función tan fácil de utilizar no sirve para esta aplicación (aunque sí para muchas otras). Pero, ¿Por qué? Como todos a estas alturas sabemos, el código de *Arduino* se ejecuta de manera secuencial, línea tras línea y de arriba abajo. *Arduino* no ejecuta una línea si no ha terminado de ejecutar la anterior. Cuando llegue a la línea *PulseIn()*, *Arduino* se quedará esperando en ella el tiempo que haga falta hasta recibir un pulso y medirlo. Como sabemos, el mando solo nos enviará un pulso por canal cada 20ms, por lo que esta función *PulseIn* estará esperando a recibir el pulso (y sin hacer nada mas) durante 20ms. Teniendo en cuenta que contamos con 4 canales, la demora se puede prolongar hasta 80ms. Este intervalo de tiempo que a priori puede parecer muy corto, para nuestros *drone* es toda una eternidad y hará que nunca sea estable (esto lo veremos en otro apartado más adelante).

Localización del receptor en el *frame*

¿Cómo podemos leer las señales recibidas desde el mando de una forma eficiente? **Utilizando interrupciones *hardware,* una por canal**. El funcionamiento de una interrupción *hardware* es muy simple: **ante un flanco positivo (cambio de estado de *LOW* a *HIGH*) o negativo (cambio de estado de *HIGH* a *LOW*) en alguna de las interrupciones (pins), *Arduino* detendrá la ejecución normal del programa y ejecutara la parte del código que hayamos asociado a esa interrupción en concreto**. Una vez ejecutado esta parte de código, *Arduino* regresará al programa principal y seguirá ejecutándolo donde lo dejó. A continuación os explicaré como leer los canales de vuestro mando *RC* de forma que la lectura solo lleve unos poco microsegundos (100 veces menos que con la función *PulseIn*). Para poder poner esto en marcha **necesitamos 4 interrupciones *hardware*, una por cada canal que queremos leer.**

La paca *Arduino Nano* cuenta únicamente con dos interrupciones *hardware*, pero existen **librerías como *EnableInterrupt.h*** que permiten convertir casi cualquier pin analógico o digital en interrupción. Esta librería es extremadamente simple de usar como veremos mas adelante.

Al final de este artículo podréis encontrar el *sketch* completo que utilizo para medir los pulsos con interrupciones *hardware*. **El *sketch* es muy simple como veréis a continuación.** Cuando se detecta un pulso (flanco positivo o negativo) en alguna de las cuatro interrupciones, se ejecuta la parte del código asociado a esa interrupción. El primer paso es identificar si hemos detectado un flanco positivo o uno negativo, para lo que bastará con leer el estado de la propia entrada digital. Si está en estado *HIGH* significará que hemos detectado un flanco positivo (comienzo del pulso) y si está en *LOW*, uno negativo (fin del pulso). **Cuando se detecta un flanco positivo se registra el tiempo (el instante)** en el que se ha dado la interrupción, para lo que utilizaremos la función *micros()* y lo guardamos en una variable. Una vez hecho esto *Arduino* sale de la interrupción y sigue ejecutando el código principal. Un tiempo **después se detecta un flanco negativo** (cambio de estado de *HIGH* a *LOW*), por lo que se vuelve a ejecutar la interrupción. Volvemos a leer el estado de la entrada que esta vez estará en estado *LOW*. **Finalmente detenemos el correspondiente contador y calculamos el tiempo que ha transcurrido desde que lo hemos activado, que será el tiempo que ha durado el pulso**. A continuación os dejo el código completo para que podáis leer los canales de vuestro mando:

La diferencia de mi *sketch* con la función *PulseIn* es que mi programa no está esperando (y sin hacer nada mas) hasta recibir un pulso para seguir ejecutando el *loop* principal. **El programa solo deja de ejecutarse mediante interrupciones para poner en**

marcha un contador y después detenerlo. Todo el tiempo restante continúa funcionando y ejecutando el control principal. Gracias a esto conseguimos leer los 4 canales del mando *RC* prácticamente sin consumir nada de tiempo. Os dejo el código completo para que podáis leer los canales de vuestro mando. El conexionado necesario es el mismo que hemos visto mas arriba:

- ***Roll ⇒ IN 9***
- ***Pitch ⇒ IN 12***
- ***Yaw ⇒ IN 7***
- ***Throttle ⇒ IN 8***

```
#include <EnableInterrupt.h>
long loop_timer, tiempo_ejecucion;

volatile long contPotenciaInit; // LEER MANDO RC POTENCI
volatile int PulsoPotencia;
void INTpotencia() {
  if (digitalRead(8) == HIGH) contPotenciaInit = micros(
  if (digitalRead(8) == LOW)PulsoPotencia = micros() - c
}

volatile long contPitchInit; // LEER MANDO RC PITCH
volatile int PulsoPitch;
void INTpitch() {
  if (digitalRead(12) == HIGH) contPitchInit = micros();
  if (digitalRead(12) == LOW) PulsoPitch = micros() - cc
}

volatile long contRollInit; // LEER MANDO RC ROLL
volatile int PulsoRoll;
void INTroll() {
  if (digitalRead(9) == HIGH) contRollInit = micros();
  if (digitalRead(9) == LOW) PulsoRoll = micros() - cont
}

volatile long contYawInit; // LEER MANDO RC YAW
volatile int PulsoYaw;
void INTyaw() {
  if (digitalRead(7) == HIGH) contYawInit = micros();
  if (digitalRead(7) == LOW) PulsoYaw = micros() - contY
}

void setup() {
  pinMode(13, OUTPUT);

  pinMode(7, INPUT_PULLUP);                    // YAW
  enableInterrupt(7, INTyaw, CHANGE);
```

```
  pinMode(8, INPUT_PULLUP);                  // POTENCIA
  enableInterrupt(8, INTpotencia, CHANGE);
  pinMode(12, INPUT_PULLUP);                 // PITCH
  enableInterrupt(12, INTpitch, CHANGE);
  pinMode(9, INPUT_PULLUP);                  // ROLL
  enableInterrupt(9, INTroll, CHANGE);

  Serial.begin(115200);
  delay(100);
}

void loop() {
  while (micros() - loop_timer &lt; 10000);

  tiempo_ejecucion = (micros() - loop_timer) / 1000;
  loop_timer = micros();

  Serial.print(PulsoPotencia);
  Serial.print("\t");
  Serial.print(PulsoPitch);
  Serial.print("\t");
  Serial.print(PulsoRoll);
  Serial.print("\t");
  Serial.println(PulsoYaw);
}
```

Para poder seguir es importante que hayas conseguido poner en marcha esta parte de forma adecuada. Os tenéis que asegurar que cada canal del mando corresponde con los movimientos del *drone* que hemos asignado. Moved la palanca asociada al *Pitch* y comprobad que es la variable *PulsoPitch* la que cambia y haced lo mismo con los demás canales. En caso de no estén bien asignados, podéis modificar el nombre de las variables (recomendado) o modificar el cableado, pero asegurad que moviendo el *stick* asociado al movimiento *Pitch* es la variable *PulsoPitch* la que cambia y no ninguna otra.

Los canales de mi mando esta configurados de la siguiente manera. Comprobad como están configurados los vuestros con el *sketch* de arriba. En caso de que alguno este invertido respecto a lo que pongo a continuación, habrá que ajustarlo mas adelante. En mi caso, el canal de *throttle* está invertido de fábrica (es cuestión de como esté diseñado el mando que compremos), ya que leo 2000us con el *stick* al mínimo y 1000us con el *stick* al máximo:

- ***Stick* de *Pitch* arriba**: 1000us aprox
- ***Stick* de *Pitch* abajo**: 2000us aprox

- ***Stick* de *Roll* derecha**: 2000us aprox
- ***Stick* de *Roll* izquierda**: 1000us aprox
- ***Stick* de *Yaw* derecha**: 1000us aprox
- ***Stick* de *Yaw* izquierda**: 2000us aprox
- **Stick throttle arriba**: 1000us aprox
- **Stick throttle abajo**: 2000us aprox

Como hemos visto mas arriba, el mando no da exactamente pulsos de 1*ms* con el *stick* al mínimo, ni pulsos de exactamente 2*ms* subiéndolo a máximo. Estos pequeños errores de precisión del mando han de ser corregidos, especialmente para el canal destinado al *throttle*. Para ello utilizaremos en siguiente código, que simplemente lee los cuatro canales y visualiza los valores máximo y mínimos mientras movemos los *stick* de un lado a otro. Anotaremos la duración de los cuatro pulsos en los extremos de cada canal, moviendo los cuatro *stick* del mando arriba/abajo e izquierda/derecha hasta conseguir los valores máximos y mínimos de pulso.

Estos valores después serán utilizados para escalar las lecturas del mando a nuestras necesidades para utilizarlos como consigna (referencia) para los controladores *PID.*

También podéis descargar el archivo en el siguiente enlace, archivos '*mandoCompletoINT*' y '*MaxMinArduino*'.

Ir al archivo

Estos números después se utilizarán para escalar el *throttle*. Para el *Pitch, Roll* y *Yaw* asumo que el mando es ideal y los pulsos varían exactamente entre 1ms y 2ms, ya que son menos críticos. **Es importante que en el código que os dejo mas adelante para poner en marcha y controlar los motores modifiquéis los**

siguientes parámetros en función de lo que haya arrojado el *software* de arriba:

```
// MANDO POTENCIA
const int PulsoMaxPotencia = 2000;
const int PulsoMinPotencia = 980;
const float tMaxPotencia = 1.72; // <--- - AQUÍ
const float tMinPotencia = 1.06; // <--- - AQUÍ
```

Ahora que ya sabemos cómo leer pulsos con nuestra placa *Arduino*, ¿qué hacemos con esta señal? ¿Cómo le decimos a nuestro *drone* que avance o que gire en una dirección determinada mediante un pulso en microsegundos?

Como ya sabemos, para poder girar o avanzar, el *drone* tiene que ser capaz de inclinarse manteniéndose estable en el aire. Para ello, **tenemos que indicar al *drone* cuantos grados queremos que se incline en una dirección determinada para poder desplazarse hacia donde se le ha indicado.** Por lo tanto, habrá que usar estas señales del mando *RC* como referencia para inclinar el *drone* en un eje o en otro. Al fin y al cabo, esa es la función final del mando: **indicar al *drone* cuantos grados queremos que se incline en un eje determinado para poder desplazarnos en una dirección concreta.** Para ello, es necesario '**procesar' la información en microsegundos que nos llegar del receptor y convertirla a una señal proporcional en grado de inclinación.**

Porgamos como ejemplo el *stick* asociado al eje *Pitch*. Tenemos que procesar la señal que recibimos del receptor (variable *PulsoPitch*), de forma que con el *stick* es su posición central (sin moverlo ni arriba ni abajo), el *drone* reciba una consigna de 0º de inclinación, es decir, que se mantenga estable y sin inclinarse.

PulsoPitch = 1500us (*stick* en posición central) ⇒ **0º de inclinación**

Que, si por el contrario movemos el *stick* hasta arriba, el *drone* reciba la orden de inclinarse unos -30º en ese eje para desplazarse en esa dirección.

PulsoPitch = 1000us (*stick* arriba) ⇒ **-30º de inclinación**

Y que, si movemos el *stick* hacia abajo, el *drone* reciba la orden de inclinarse unos +30º en ese eje para desplazarse en dirección contraria.

PulsoPitch = 2000us (*stick* abajo) ⇒ **+30º de inclinación**

Le estamos indicando al *drone* que se incline en el eje Pitch en un sentido o en otro (-30º/+30º) en función de la posición del *stick* del mando *RC*.

Para procesar las señales obtenidas del mando y transformarlas en consigna de inclinación, utilizaremos las siguientes ecuaciones. Hacen exactamente lo mismo que la función *map()* de *Arduino.* El simplemente la ecuación de una recta que pasa por dos puntos:

```
wPitch =   ((wMinPitch - wMaxPitch) / (tMax - tMin)) * (
wRoll =   ((wMaxRoll - wMinRoll) / (tMax - tMin)) * (Pul
wYaw =  ((wMinYaw - wMaxYaw) / (tMax - tMin)) * (PulsoYa
```

Os dejo el software completo con las ecuaciones de procesamiento ya metidas. Podéis también descargarlo en el siguiente enlace:

Ir al archivo

```
#include <EnableInterrupt.h>

long loop_timer, tiempo_ejecucion;
float wPitch, wRoll, wYaw, pulsoPotencia;

// MANDO POTENCIA
const int PulsoMaxPotencia = 2000;
const int PulsoMinPotencia = 1000;
const float tMaxPotencia = 1.83;  // &lt; - Si teneis la
const float tMinPotencia = 1.12;  // &lt; - por este y v

const float tMax = 2;
const float tMin = 1;

// MANDO PITCH
const int wMaxPitch = -30; // &lt; - Si teneis la entrad
const int wMinPitch = 30;  // &lt; - por este y viceversa

// MANDO ROLL
const int wMaxRoll = 30;  // &lt; - Si teneis la entrada
const int wMinRoll = -30; // &lt; - por este y viceversa
```

```
// MANDO YAW
const int wMaxYaw = 30;  // &lt; - Si teneis la entrada
const int wMinYaw = -30; // &lt; - por este y viceversa

volatile long contPotenciaInit; // LEER MANDO RC POTENCI
volatile int PulsoPotencia;
void INTpotencia() {
  if (digitalRead(8) == HIGH) contPotenciaInit = micros(
  if (digitalRead(8) == LOW)PulsoPotencia = micros() - c
}

volatile long contPitchInit; // LEER MANDO RC PITCH
volatile int PulsoPitch;
void INTpitch() {
  if (digitalRead(12) == HIGH) contPitchInit = micros();
  if (digitalRead(12) == LOW) PulsoPitch = micros() - cc
}

volatile long contRollInit; // LEER MANDO RC ROLL
volatile int PulsoRoll;
void INTroll() {
  if (digitalRead(9) == HIGH) contRollInit = micros();
  if (digitalRead(9) == LOW) PulsoRoll = micros() - cont
}

volatile long contYawInit; // LEER MANDO RC YAW
volatile int PulsoYaw;
void INTyaw() {
  if (digitalRead(7) == HIGH) contYawInit = micros();
  if (digitalRead(7) == LOW) PulsoYaw = micros() - contY
}

void setup() {
  pinMode(13, OUTPUT);

  pinMode(7, INPUT_PULLUP);                  // YAW
  enableInterrupt(7, INTyaw, CHANGE);
  pinMode(8, INPUT_PULLUP);                  // POTENCIA
  enableInterrupt(8, INTpotencia, CHANGE);
  pinMode(12, INPUT_PULLUP);                 // PITCH
  enableInterrupt(12, INTpitch, CHANGE);
  pinMode(9, INPUT_PULLUP);                  // ROLL
  enableInterrupt(9, INTroll, CHANGE);

  Serial.begin(115200);
  delay(200);
}

void loop() {
```

```
  while (micros() - loop_timer &lt; 10000);
  tiempo_ejecucion = (micros() - loop_timer) / 1000;
  loop_timer = micros();

  // =============== Ecuaciones de procesamiento
  wPitch =   ((wMinPitch - wMaxPitch) / (tMax - tMin)) *
  wRoll =   ((wMaxRoll - wMinRoll) / (tMax - tMin)) * ((
  wYaw =  ((wMinYaw - wMaxYaw) / (tMax - tMin)) * ((Puls
  pulsoPotencia =  ((PulsoMaxPotencia - PulsoMinPotencia
  // =============== Ecuaciones de procesamiento

  Serial.print(pulsoPotencia);
  Serial.print("\t");
  Serial.print(wPitch);
  Serial.print("\t");
  Serial.print(wRoll);
  Serial.print("\t");
  Serial.println(wYaw);
}
```

Lo mas importante es asegurar que una vez procesadas las señales, el canal de *throttle* no ha quedado invertido. Es decir, con el *stick* al mínimo tenemos que obtener una salida de 1000us aproximadamente, y con el *stick* al máximo una salida de 2000us aproximadamente. Si tenéis el canal invertido, corregirlo cambiando de orden los parámetros de entrada. De esto:

```
const float tMaxPotencia = 1.83;  // < - Si tenéis la er
const float tMinPotencia = 1.12;  // < - por este y vice
```

A esto:

```
const float tMaxPotencia = 1.12; // < - Si tenéis la ent
const float tMinPotencia = 1.83; // < - por este y vicev
```

Comprobar también los demás canales, y en caso de tener alguno invertido, seguid el mismo procedimiento que con el canal *throttle* hasta ajustarlos como se muestra a continuación:

- **Moviendo *stick* de *Pitch* hacia arriba, *wPitchConsigna* = -30º aprox.**
- **Moviendo *stick* de *Pitch* hacia abajo, *wPitchConsigna* = +30º aprox.**
- **Moviendo *stick* de *Roll* hacia la derecha, *wRollConsigna* = +30º aprox.**
- **Moviendo *stick* de *Roll* hacia la izquierda, *wRollConsigna* = -30º aprox.**

- **Moviendo *stick* de *Yaw* hacia la derecha, *wYawConsigna* = +30º aprox.**
- **Moviendo *stick* de *Yaw* hacia la izquierda, *wYawConsigna* = -30º aprox.**

Las implicaciones de no hacer esta modificación en los canales *Pitch, Roll* y *Yaw* no son tan graves, simplemente, cuando ordenáramos al *drone* 'avanzar', este retrocedería. Nada grave, pero conviene corregirlo.

Acordaos de cambiarlo también cuando descargues el *software* principal en la entrada número 9.

Continuar con la siguiente entrada:

1. Conceptos generales sobre *drones.*
2. Material necesario y montaje de los componentes hardware.
3. Mando RC y receptor. Programación en *Arduino (código).*
4. **→ MPU6050 y su programación en *Arduino (código).***
5. Batería LiPo *(código).*
6. Control de estabilidad y PID.
7. Motores, ESC y su programación en *Arduino (código).*
8. Calibración de hélices y motores *(código).*
9. Software completo y esquema detallado *(código).*
10. Probando el Software completo antes de volar.
11. Como leer variables de *Arduino* en *Matlab (código).*
12. Los mejores *drones* de 2018 | Comparativa y guía de compra.

5/5 - (4 votos)

Etiquetas: arduino, emisor, mando, pitch, PWM, receptor, roll, señal, sketch, stick, throttle, yaw

Related Posts

Control de estabilidad y PID para drones

Como organizar un proyecto Arduino en 4 pasos

Medir distancia con Arduino y sensor de ultrasonidos HC-SR04

Sobre el Autor

Drone Desdecero

56 Comentarios

kevin
caballero buenas tardes mi nombre es kevin de Colombia medellin, era para decir y felicitarlo por poner esta pagina o bloc es muy interesante porque este es un proyecto de hoy lo único que le doy como concejo para que hubiera quedado mas completo hubieras montado vídeos de como calibrar receptor, mpu 6050, y motores porque la verdad esta muy bien explicado pero uno como persona al menos yo, me gusta ver en practica para que quede mejor pero igual amigo felicidades.

27 marzo, 2019 Responder

ArduProject
Buenas Kevin!! Gracias por e comentario.
Tengo vídeos de como calibrar los motores y de como poner el marcha el drone una vez montado. Pensé que con eso y las explicaciones de bog seria suficiente. Me lo apunto como tarea pendiente.
Un saludo y animo!! ☺

28 marzo, 2019 Responder

Kevin Flores
Hola! felicitaciones, tu proyecto quedó genial! dónde puedo encontrar los videos que

comentaste que tienes? sería de gran ayuda, anticipadamente te agradezco.

22 junio, 2019 Responder

arduproject
Hola Kevin, los tienes en la propia entrada o en el canal de Youtube:
ArduProject Arduino
Un saludo!

27 junio, 2019 Responder

Kevin
A bueno caballero con mucho gusto lo del vídeo de calibración de motores no lo había visto esta excelente , tambien era para preguntarte estoy confundido con lo del censor UD pone lo del censor para configurarlo o muestra cómo está configurado. Y también compa lo de lo último que es par calibrar mando para que lo lea todo sobre lo último que ya es para que funcione del todo y no me da el código lo cargo y no me da osea me explico yo lo pongo en el programa de Arduino y lo verifico y no me da, en qué estoy fallando muchas gracias caballero, tambn si me puedes dar un número de wassap para contactarme más fácil y ya arreglamos me interesa terminarlo gracias

22 abril, 2019 Responder

Sergi
Primero de todo, muchas gracias por el post y toda la información!
Tengo un par de preguntas:
– Cuál es el alcance del mando?
– Cuál es el empuje de los motores? Es para saber el peso máximo que puede tener el drone y su podría llevar una cámara GoPro o similar.
Saludos

10 abril, 2019 Responder

arduproject
Buenas Sergi,
– El alcance del mando depende del mando que tengas… con el mio, si no hay obstáculos en el

camino, es de unos 300m.
– El empuje de los motores no sabría cuantificarlo ni medirlo. Una GOPRO de sobra…
Un saludo

3 mayo, 2019 Responder

Roberto Carlos
El mío no kiere dar😔😔

20 julio, 2019 Responder

arduproject
Buenas Roberto. Que significa que no quiere dar?

22 julio, 2019 Responder

Angel Hidalgo
Hola buenas noches, tengo una duda, estoy intentanto hacer este drone pero no puedo comprar el mando y estaba haciendo una app de android para manejarlo, pero no entiendo como funcina la parte de if (digitalRead(8) == HIGH) contPotenciaInit = micros();
if (digitalRead(8) == LOW)PulsoPotencia = micros() – contPotenciaInit;
y como la puedo adecuar a una barra de progreso en la app.
Que me recomendarias?

24 julio, 2019 Responder

Daniel Ramírez.gt
Que tal compañero, la verdad para un dron de esta talla no creo que controlarlo con una app Mobil sea lo mas eficiente, pero si es lo que esta a mano es completamente válido, si tienes una Tienda de electrónica en tu ciudad puedes comprar unos módulos de radiofrecuencia nrf24 junto con un atmega 328, o un arduino nano, podrías crear un mando con su receptor, y es muy econico, rondará talvez los €25

29 julio, 2019 Responder

THOMAS

Hola Buenas, construi el DRONE exactamente igual, lo unico no dispongo del mando RC, me podrias dar una mano para controlar el drone por bluetooth a traves de la app Universal RC Transmitter. Gracias

4 noviembre, 2019 Responder

arduproject
Buenas Thomas,
Lo siento, nunca he trabajado con módulos Bluetooth hasta ahora. Seguro que en internet encuentras infinidad de tutoriales y librerías.
Suerte!!

4 noviembre, 2019 Responder

THOMAS
Hola Como andas, mira estuve buscando en internet y encontre un pdf en el que se construye un drone con los mismos materiales de este proyecto, y se quiere controlarlo por bluetooth a traves de una APK, estoy siguiendo todos los pasos pero hay un error que no puedo solucionar. En el archivo principal dentro del setup se llama a una funcion inicializar_VariablesErrorPID(), que no existe en ninguna parte del codigo, suponemos que esta funcion es para estabilizar el drone, pero no existe. Nos podrias ayudar a reconstruir esa parte ?, el codigo esta al final del documento PDF.
Gracias
Link:
https://riunet.upv.es/bitstream/handle/10251%20Dise%C3%B1o%20y%20control%20de%20sequence=1

5 noviembre, 2019 Responder

arduproject
Buenas Thomas,
Tendrás que contactar con la persona que hizo ese trabajo, yo no te puedo ayudar. Si tienes cualquier duda sobre algo relacionado con este blog, no dudes en preguntar ☺
Suerte!

5 noviembre, 2019 Responder

Daniel
Buenas! Ante todo gracias por escribir el blog y por todos los códigos.
Con los valores obtenidos, el throttle al mínimo alcanza aproximadamente los 1000 (mi throttle está invertido) y al máximo 1500, es decir, no llega a los 2000. He incluido los valores que he calculado previamente pero no consigo llegar a ese valor. No sé a que puede deberse, espero pueda ayudarme a resolver la duda, gracias!

17 enero, 2020 Responder

arduproject
Buenas,
Que raro…. ¿obtienes la misma salida en todos los canales? de todas formas por software es muy sencillo hacer que llegue hasta 2000us.
Un saludo

20 enero, 2020 Responder

Xakko
Hola. Soy fanatico de Arduino y me estoy metiendo en el mundo de los drones (controlados por Arduino). Queria saber si hay alguna forma de unir el modulo receptor del wifi al arduino del dron sin pasar por un intermediario. Por mi parte jugare un poco a ver si me sale, pero desearia saber por que debe ser con un receptor externo y de ahi a pines y no el WiFi directo a la placa Arduino del dron.
Excelente todo!!! Gracias!!!

17 febrero, 2020 Responder

arduproject
Buenas Xakko,
Yo no he utilizado WIFI para controlar el drone.
Mando RC con receptor simplemente.
Un saludo!

17 febrero, 2020 Responder

alexander
a alguien le ha funcionado el dron con este tutorial? ya compre las todo y quiero hacerlo pero no veo que a alguien mas a parte de arduproyec le haya funcionado.

20 febrero, 2020 Responder

arduproject
Buenas Alexandre,
Hay gente que lo ha conseguido, y hay gente que no.
Este blog es una ayuda para construir un drone, donde además pongo el código que a mi me ha funcionado y en el que tanto he trabajado. No es un producto comercial.
Recuerda que es algo gratuito, que no gano nada por atraer a gente al blog.
Un saludo

21 febrero, 2020 Responder

Jose
Buenas,
sabes si existe alguna libreria para poder utilizar las interrupciones en arduino nano every??
esque tengo problemas con arduino nano cuando declaro esta libreria y utilizo sus funciones, lo que no me pasa en arduino uno, pero claro no es lo mismo montar un arduino uno que uno nano en un dron.
Un saludo.

7 marzo, 2020 Responder

arduproject
Buenas, no conozco esa placa. La que he utilizado yo es la librería #include
Prueba con esa.
Utilizar un Nano o un Uno es indiferente, funcionará con ambas.
Un saludo

10 marzo, 2020 Responder

Nemesis
Enhorabuena por tu proyecto , y por el tiempo que le has dedicado a detallarlo paso a paso , y de esta forma compartirlo con todos.
Enhorabuena y gracias.

7 abril, 2020 Responder

arduproject
Mil gracias!! ☺

9 abril, 2020 Responder

michael
Hola he estado mirando tu código y me pregunto si funcionará bien, corrígeme si me equivoco, pero según veo , asocias al duración del pulso con la salida de potencia, pero aunque varíe en el tiempo siempre se le esta suministrando la potencia máxima la motor, entonces no estas controlando la potencia suministrada al motor como con un potenciometro, si no que envías un flanco positivo cada vez que se activa el receptor.
lo mismo ocurriría con la inclinación, a la hora de mover el roll se inclinara 30º según veo, pero siempre sera esta inclinación, por lo tanto no se están cambiado los grados de inclinación sino que siempre son los mismo 30º.

19 abril, 2020 Responder

arduproject
Buenas,
La potencia se controla exactamente igual que con un potenciómetro. No es fácil explicar esto de forma resumida en un comentario, para eso está la entrada.
Cuando cargues el código y lo veas funcionar, lo entenderás mejor ☺

21 abril, 2020 Responder

jordi
hola
que significa los canales de la radio control y si tiene mas canales que pasaria es mejor?

22 mayo, 2020 Responder

arduproject
Tendrás mas grados de libertad con el drone. Un mando con mas canales siempre es mejor a largo plazo, por si quieres añadir nueva funcionalidades al drone.
Un saludo

23 junio, 2020 Responder

Davinson Martinez

arduproyect, Saludos desde Colombia. Felicitarlo por tan inmenso y arduo trabajo. Estoy en la face terminal del proyecto haciendo pruebas de vuelo, me surge un inconveniente a la hora de alzar vuelo en el despegue, y revisando el código más a profundidad en el sketch de prueba de mando RC. (Mandocompletoproces) me di cuenta de que mi control no está dando los valores en 0 ni tan cercanos a 0 estando todas las entradas en posición media, ni en el eje pitch, ni en el roll, ni en el yaw, y en punto máximo y mínimo tampoco se aproximan a (30) ni (-30). Creo que de allí se deriva de que al levantar vuelo a 5cm del piso empieza a inclinarse hacia uno de sus ejes sin antes mandar ninguna orden de inclinación desde el mando estando aún todos las entradas en punto media que debería ser 0° grados. Un dato adicional mi control RC no está muy bien calibrado de fábrica ya que al leer los valores en bruto (microsegundos) por el puerto serial con serial.print me da los siguientes datos, para el trotl en estado mínimo (1.884)u media (1.336)u máxima(880)u. Para el eje pitch mínimo (1.332)u media(1.580)u máxima(1.832)u. Para el eje roll mínimo(1.284)u media (1.524)u máxima(1.788)u. Para el eje yaw mínimo(1.320)u media (1.576)u máxima(1820)u. Como puede ver ninguna se aproxima a 1000 en estado mínimo ni 1500 en estado media ni 2000 en máxima… Para mi entender no es falla del trotl ya que aunque no tenga los balores sercanos a lo establecido este es capas de dar su potencia en campo para alzar el vuelo y también es capaz de disminuir con gran precisión al aterrizar hasta apagar motores por completo si volver a ponerlos en marcha si damos potencia (trotl). Por favor deme un consejo que devo corregir o implementar. Muchísimas gracias por su ayuda y por el proyecto, le deseo lo mejor.

15 junio, 2020 Responder

arduproject

Buenas Davinson! perdona por la demora, he estado ausente un par de meses por temas personales, y no he podido entrar en ese tiempo.

Es un problema bastante común. Es imprescindible que con el mando esté bien ajustado antes de hacer pruebas de vuelo. ¿Has logrado corregir el problema? sin tocar las palancas del mando, la consigna tiene que ser 0 para pitch, roll y yaw. La clave podría estar en estas lineas de código:
// Si las lecturas son cercanas a 0, las forzamos a 0 para evitar inclinar el drone
// por error
if (wPitchFilt < 9 && wPitchFilt > -9)wPitchConsigna = 0; // '9' por un error de fabrica de mi mando
if (wRollFilt < 3 && wRollFilt > -3)wRollConsigna = 0;
if (wYawlFilt < 3 && wYawlFilt > -3)wYawConsigna = 0;

23 junio, 2020 Responder

Davinson Martinez
A mi me ha funcionado hay que dedicarle tiempo a comprender el código por ende aprender el lenguaje de programaciones si es que quieres implementar, si lo que quieres es solo armarlo sigue los pasos tal cual y te funcionará a la perfección

15 junio, 2020 Responder

---

arduproject
Buenas Davinson! Me alegra mucho que te haya ido bien… como bien dices, hay que tomar tiempo para entender el código y solucionar los problemas que surjan.
Un saludo

23 junio, 2020 Responder

Guillermo
Buenas tardes, estoy realizando este proyecto, pero cuando cargo el primer código de esta entrada del blog el monitor serial solo imprime el signo de interrogación seguido de los valores que se deberían obtener en 0, alguna idea de cual es el problema?
Dejo aquí exactamente el output para que lo veais:
0 0 0 0

0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
⸮⸮⸮

26 junio, 2020 Responder

---

arduproject
Buenas,
Has configurado bien el baudrate del canal serie de Arduino? tiene que coincidir con lo que pongas en esta línea de código:
Serial.begin(115200);
Revisa también el cableado, los pines que he utilizado yo y los que has usado tu son los mismos seguro?
Un saludo

30 junio, 2020 Responder

Guillermo
Una pregunta, serviria el mando FlySky T-6 con el transmisor que has utilizado, el R6B?
Y habría que cambiar parte del código si se utiliza un controlador distinto al que has utilizado?
Muchas gracias.

27 junio, 2020 Responder

Guillermo
Perdón, quería decir FlySky i6, FS-i6. Pero en general el código no tiene que cambiar verdad? Si es así, podrías decirme por qué; para que yo pueda deducir el código entendiendo el problema o la diferencia general entre los sketch.

28 junio, 2020 Responder

Eduardo
Hola, yo estoy utilizando el F6-i6 y lo unico que veo por el puerto serie es un simbolo cuadrado continuamente.
Muchas gracias

28 junio, 2020 Responder

arduproject
Buenas Eduardo,
Has configurado bien el baudrate del canal serie de Arduino? tiene que coincidir con lo que pongas en esta línea de código:
Serial.begin(115200);
Si solucionas el problema ponlo en los comentarios ☺
Un saludo

30 junio, 2020 Responder

Guillermo
El receptor que has utilizado ¿enciende algún LED cuando funciona correctamente?

28 junio, 2020 Responder

arduproject
Tiene un led para saber el estado de carga de las pilas, pero no si funciona correctamente.
Un saludo

30 junio, 2020 Responder

Eduardo
Hola,
Efectivamente no me di cuenta de que yo tengo que poner el serial.begin(9600)
Una vez hecho esto, subí el programa y la potencia iba de 1205 a 15014
// MANDO POTENCIA
const int PulsoMaxPotencia = 2000; » lo sustituyo a qui» 15014
const int PulsoMinPotencia = 980; » lo sustituyo a qui » 1205
const float tMaxPotencia = 1.72; // <— Este valor de donde sale, No consigo verlo en la explicación
const float tMinPotencia = 1.06; // <— Este valor de donde sale
Cuando cargo el software completo me da tiempo excedido (LED 13)
Gracias.

1 julio, 2020 Responder

Eduardo
Hola de nuevo, rectifico el valor de máxima es 1514, se me colo un «0»
Perdon

1 julio, 2020 Responder

Eduardo
Hola, ya me voy aclarando, mis lecturas son:
PulsoMaxPotencia = 1516
PulsoMinPotencia = 1012
tMaxPotencia = 1.50
tMinPotencia = 1.01
Pich max +50
Pich min -50
Yaw max +50
Yaw min -50
Roll max +50
Roll min -50
Todavia no consigo hacer funcionar los motores. Voy a poner el display haber si así veo donde me falla.
Gracias

7 julio, 2020 Responder

Eduardo
Hola, yo de nuevo, siento ser pesado pero espero que esto sirva para los que tengan algún problemilla.
He conectado el display y parece que va todo bien hasta que llego aqui
lcd.print(«Calibrar motores»);
// Calibrar motores –> Mover stick de Roll a la derecha para continuar. Simplemente mandamos pulsos de 1000us (pulso minimo) a los motores.
// Hasta no hacerlo no salimos de este bucle. Con esto conseguimos entrar al loop principal con los motores listos para girar.
// Se puede comentar si se quiere, no es imprescindible.
while (wRoll < 20) {
pulsoPotencia = ((PulsoMaxPotencia – PulsoMinPotencia) / (tMinPotencia – tMaxPotencia)) * ((PulsoPotencia) / 1000.00 – tMaxPotencia) + PulsoMinPotencia;
wRoll = ((wMaxRoll – wMinRoll) / (tMax – tMin)) * ((PulsoRoll – calRoll) / 1000.00 – tMin) + wMinRoll;

Muevo el Roll y me salta el led 13 de ciclo sobrepasado y no se mueven los motores

7 julio, 2020 Responder

Eduardo

Hola de nuevo, he conseguido que haga las calibraciónes y que cuando muevo el roll ya no me salta el led 13, me parpadea el led azul y los motores no se mueven.
Gracias

10 julio, 2020 Responder

Adrián

Hola Buenas! He tenido un problema, he conectado todos los canales con sus respectivos pines en la placa arduino entonces al comprobar la longitud de pulso de cada canal me falla una, yaw, pitch y throttle funcionan bien (adquieren valores entre 1000 y 2000) pero el problema viene al mover la palanca respectiva al roll, el valor no cambia y permanece constante en 1472. ¿Algun consejo?

14 julio, 2020 Responder

Adrián

He probado a cambiar de puerto todas las conexiones, incluso he montado el circuito en una tarjeta arduino uno y sigue sin detectar el movimiento hacia la derecha y izquierda del joystick derecho. Antes de intentar montar este controlador de vuelo usaba otro distinto en el cual si funcionaba el roll así que no creo que sea problema de la emisora.

14 julio, 2020 Responder

arduproject

Buenas,
Me parece raro que funcionen todos los canales menos unos, cuando el código es igual para todos. Por ejemplo, el pitch te funciona bien. ¿A que canal lo tienes conectado? ¿has probado a conectar el Roll a ese canal (que ya sabemos que funciona bien)?
Un saludo

20 julio, 2020 Responder

Francisco javier de la
Claro que si Alexander a mi si me funciono. Lo hice exactamente como el proyecto esta indicado. solo tuve algunos contratiempos por que soy novato en Arduino pero si me funciono. Me encanto.

12 septiembre, 2020 Responder

arduproject
☺

12 septiembre, 2020 Responder

Leonardo
Fascinante el detalle del proyecto. Sin duda un punto de partida inminente para montar mi primer drone. Queria pedirte si tienes algun mail para poder contactarme directamente contigo sin tener que pasar por el blog para poder evacuar dudas y otros detalles que vayan surgiendo. Un saludo desde Argentina

28 septiembre, 2020 Responder

arduproject
Buenas Leonardo, gracias ☺
Tiene mi email debajo de 'Páginas más vistas' en la página principal, pero intenta poner las dudas en los comentarios, de esta forma todo el mundo puede verlas.

30 septiembre, 2020 Responder

Giulianoc75
Hola buenas tardes, tengo el dron que ya anda volando, solo que se me estuvo complicando mantenerlo estable, hasta que empeze a jugar con los trimmers y logre estabilizarlo en pitch y roll al despegar, el unico problema es que en yaw no logro hacerlo. El dron gira para la la izquierda sobre si mismo, y poniendo el trimmer al maximo hacia la derecha logro reducirlo, pero sigue girando y se complica direccionarlo. Supongo que tendre que toquetear el wMaxyaw y wMinyaw, pero queria consultarte si es lo correcto o si debo chequear el problema por otro lado. Las helices ya las calibre

con tu programa. Aclaro por si es que suma, que en mi caso estoy armando un hexacopetro (solo le agrege 2 motores mas al codigo de PWM). Y de estos hay 3 motores que rotann en senntido horario y otros 3 en antihorario, cada uno enfrentado con el opuesto. Muchas gracias y que tengas un buen dia.

6 enero, 2021 Responder

Cristián

Cordial saludo tengo una duda y creo que por eso se daño mi emisora MICROZONE MC7RB tu conectas 5v y gnd en el Bat mi emisora no dice bat si M.bus se supone que conecto hay los 5v y gnd .. o conecto cada canal con su respectivo 5v y gnd al arduino

Agradezco su pronta respuesta

9 enero, 2021 Responder

Giulianoc75

Buenas tardes, quería comentar que ya encontré la solución al problema. El tema estaba en la señal de yaw que se le asigna a cada motor. En este caso, yo le asignaba a los motores de giro inverso que el yaw se reste, por lo cual el PID cuando quería contrarrestar, el dron aumentaba el error, y eso generaba un círculo vicioso que hacía que el mismo girase sobre si a lo loco. Una cosa sencilla que no tuve en cuenta al armar el hexadron. Espero que esto le sirva a alguno que llegase a caer en el mismo error. Muchas gracias

11 enero, 2021 Responder

Añadir un comentario

Comentario:

Nombre:

Dirección de correo electrónico:

Web:

Añadir comentario

About

Política de cookies

Política de privacidad

Volver arriba ↑