Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tema: Reporte del proyecto

Materia: Servomecanismos

Maestro: Ing. Víctor De Jesús Medrano Zarazúa

Omar Antonio Niño Canizales - 1556714
Miguel Ángel Flores Rangel - 1791727
Miguel Ángel Portillo - 1525341
Julián Castor Zúñiga - 1513140

Salón: 3303

Grupo: 006

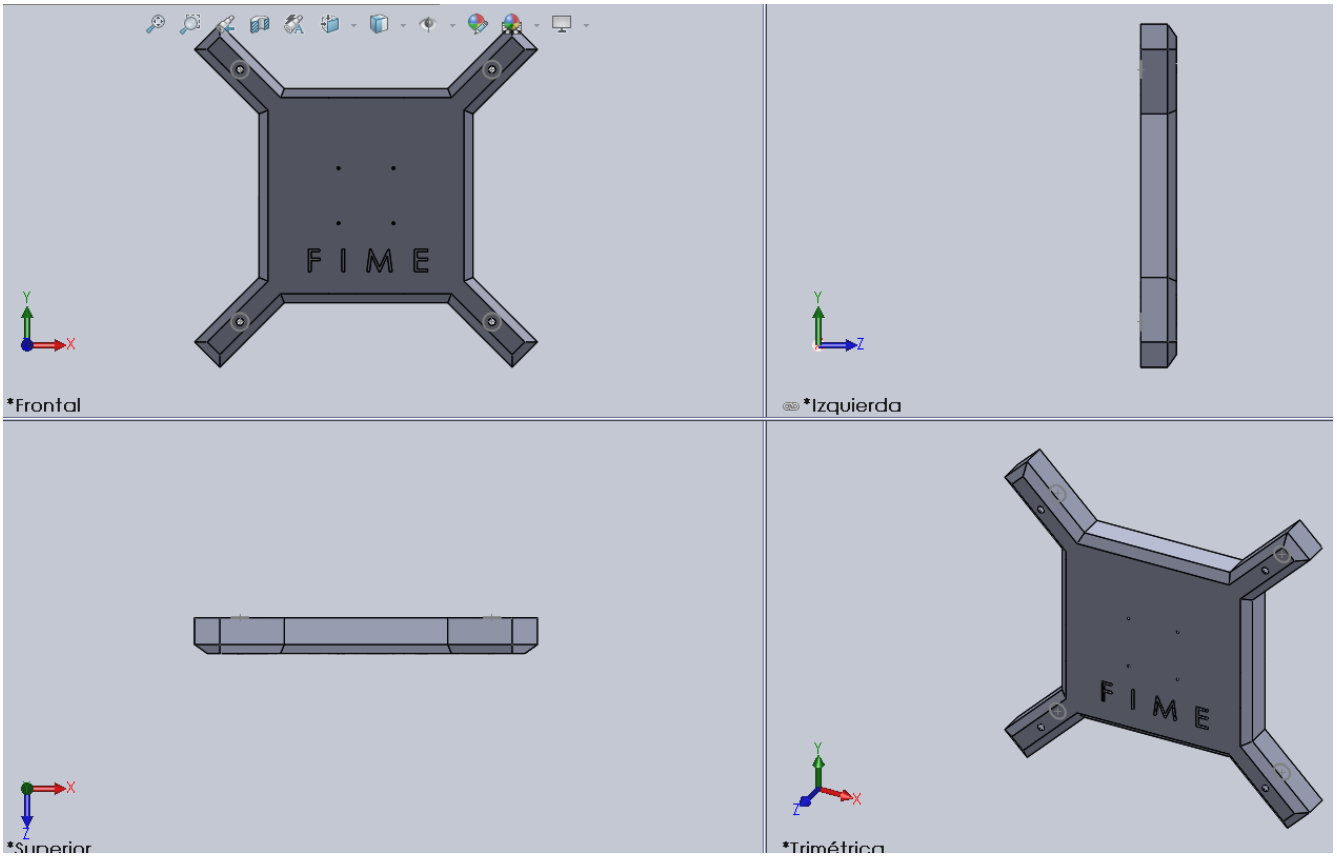
Días: L-M-V

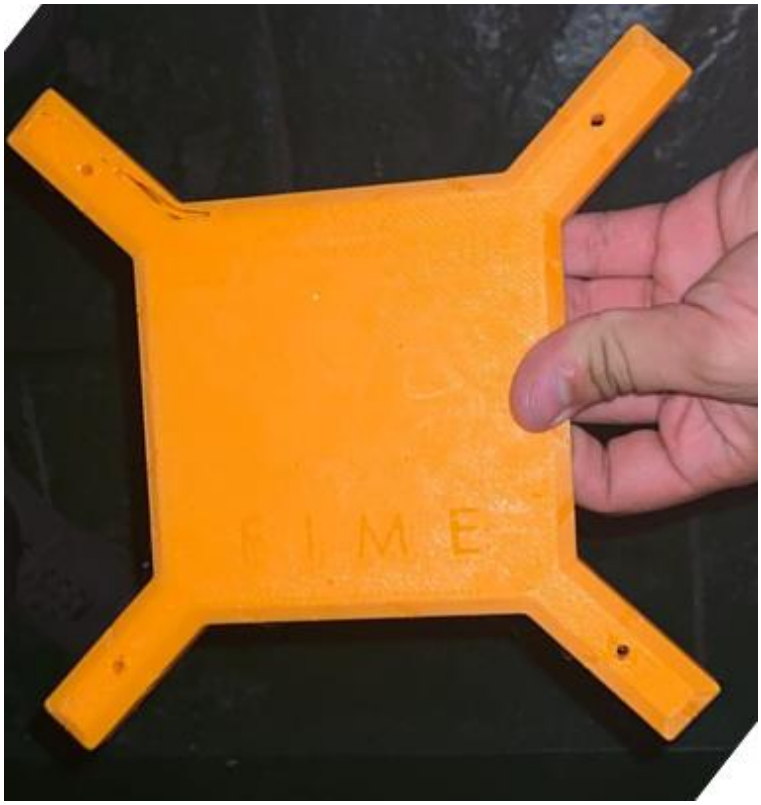
Hora: N3

Semestre Enero-Junio 2019

Reporte del proyecto

Estructura



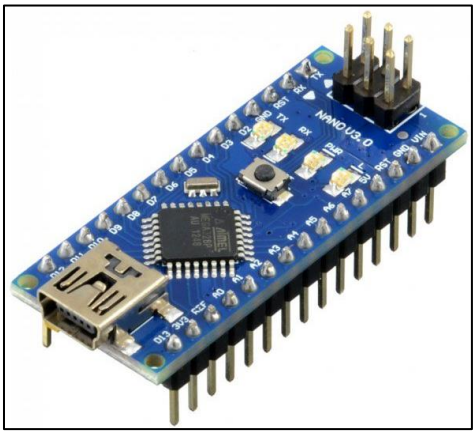


Microcontrolador/Microprocesador

Arduino Nano V3 – ATmega328

Descripción, ¿Qué es y para qué?:

El Arduino Nano es una pequeña y completa placa basada en el ATmega328 -Arduino Nano 3.0- o el ATmega168 en sus versiones anteriores que se usa conectándola a una protoboard. Tiene más o menos la misma funcionalidad que el Arduino Duemilanove, pero con una presentación diferente. No posee conector para alimentación externa, y funciona con un cable USB Mini-B.



Características:

- Microcontrolador: Atmel ATmega328
- Tensión de Operación (nivel lógico): 5 V
- Tensión de Entrada (recomendado): 7-12 V
- Tensión de Entrada (límites): 6-20 V
- Pines E/S Digitales: 14 (de los cuales 6 proveen de salida PWM
- Entradas Analógicas: 8 Corriente máx. por cada PIN de E/S: 40 mA
- Memoria Flash: 32 KB (ATmega328) de los cuales 2KB son usados por el bootloader (16 KB – ATmega168)
- SRAM: 2 KB (ATmega328) (1 KB ATmega168)
- EEPROM: 1 KB (ATmega328) (512 bytes – ATmega168)
- Frecuencia de reloj: 16 MHz
- Dimensiones: 18,5mm x 43,2mm

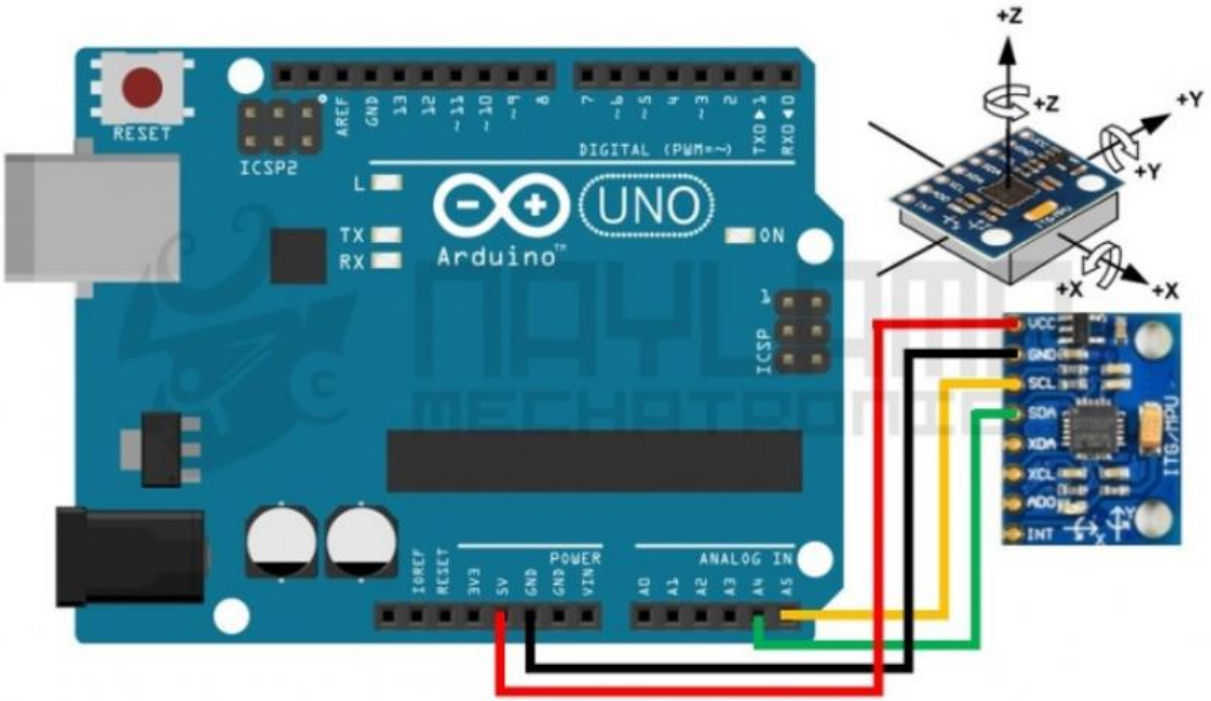
Energía:

El Arduino Nano posee selección automática de la fuente de alimentación y puede ser alimentado a través de:

- Una conexión Mini-B USB.
- Una fuente de alimentación no regulada de 6-20V (pin 30).
- Una fuente de alimentación regulada de 5V (pin 27)

Al alimentar el Arduino a través del Mini USB, el CH340 proporciona una salida de 3.3V en el pin 16 de la placa. Por ende, cuando se conecta a una fuente externa (no USB), los 3.3V no se encuentran disponibles.

Sensores



EL MPU6050 es una unidad de medición inercial o IMU (Inertial Measurment Units) de 6 grados de libertad (DoF) pues combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes. Este sensor es muy utilizado en navegación, goniometría, estabilización, etc.

- Aceleración y acelerómetros

La aceleración es la variación de la velocidad por unidad de tiempo es decir razón de cambio en la velocidad respecto al tiempo:

$a=dV/dt$

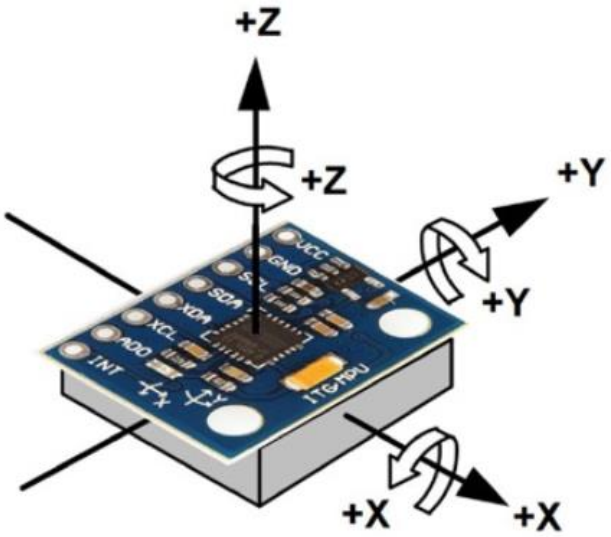
Así mismo la segunda ley de Newton indica que en un cuerpo con masa constante, la aceleración del cuerpo es proporcional a la fuerza que actúa sobre él mismo:

$a=F/m$

Este segundo concepto es utilizado por los acelerómetros para medir la aceleración. Los acelerómetros internamente tienen un MEMS (MicroElectroMechanical Systems) que de forma similar a un sistema masa resorte permite medir la aceleración.

El módulo Acelerómetro MPU tiene un giroscopio de tres ejes con el que podemos medir velocidad angular y un acelerómetro también de 3 ejes con el que medimos los componentes X, Y y Z de la aceleración.

La dirección de los ejes está indicado en el módulo el cual hay que tener en cuenta para no equivocarnos en el signo de las aceleraciones.



La comunicación del módulo es por I2C, esto le permite trabajar con la mayoría de microcontroladores. Los pines SCL y SDA tienen una resistencia pull-up en placa para una conexión directa al microcontrolador o Arduino.

Motores

Diámetro del Motor : 7mm

● Longitud del Motor: 20mm

● Diámetro del eje: 1.0mm

● Longitud del Cable : 60mm

● Peso del motor: 4g

● Opciones del motor: Motor CW (con cable rojo y azul), Motor CWW (con cable negro y blanco)

● Cantidad: 4 pcs (2 CW y 2 CCW)

Hélices

● Diámetro: 75mm

● Diámetro del eje de la hélice: 0.97mm

● Material: ABS

● Color : rojo, negro y amarillo (Opcional)

● Cantidad: 2 juegos (1 juego con 4 hélices, 2 CW y CCW), un total de 8 hélices

● Peso de hélice: 30g

Batería

Batería lipo de 500maH 1s a 3.7v.

Módulo bluetooth hc-06

El módulo HC-06 es prácticamente idéntico a simple vista con los demás módulos que existen en el mercado.

Una simple diferencia es que el módulo HC-06 funciona como Slave y el HC-05 como Master y Slave (lo que podría confundir a algunos).

Físicamente se diferencian por el número de pines. En el HC-06 tiene un conector de 4 pines mientras que el HC-05 trae uno de 6 pines.

El HC-06 tiene 4 pines:

- Vcc, Voltaje positivo de alimentación, aquí hay tener cuidado porque hay módulos que solo soportan voltajes de 3.3V, pero en su mayoría ya vienen acondicionados para q trabajen en el rango de 3.3V a 6V pero es bueno revisar los dato técnicos de nuestro modulo antes de hacer las conexiones

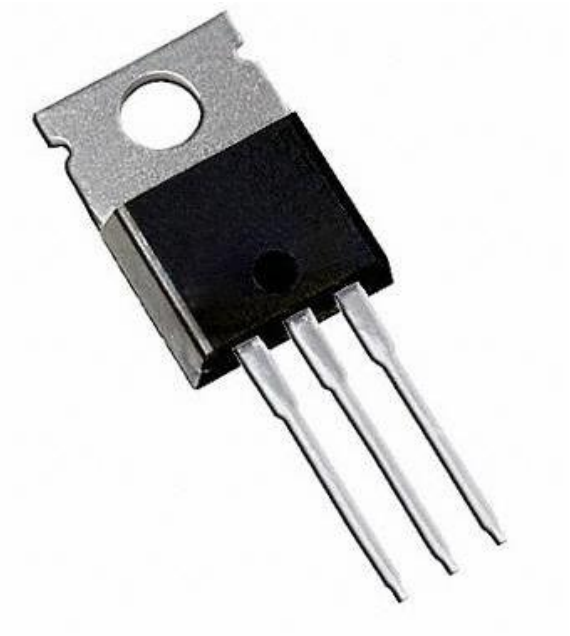
- GND, Voltaje negativo de alimentación, se tienen que conectar al GND del Arduino o al GND de la placa que se esté usando.

- TX, Pin de Transmisión de datos, por este pin el HC-06 transmite los datos que le llegan desde la PC o Móvil mediante bluetooth, este pin debe ir conectado al pin RX del Arduino

- RX, pin de Recepción, a través de este pin el HC-06 recibirá los datos del Arduino los cuales se transmitirán por Bluetooth, este pin va conectado al Pin TX del Arduino

Transistores

4x transistores MOSFET y elegir uno puede ser complicado. En primer lugar, deben soportar la potencia y la corriente de los motores y, en segundo lugar, la tensión de umbral debe ser baja; de lo contrario, Arduino no podrá encenderlos por completo (en el caso de MOSFET de tipo N). En mi caso, la corriente máxima es de 2.75 A con un voltaje de 3.7 V. Esto significa que necesito un MOSFET, que al menos resistirá alrededor de 4 - 5 A por si acaso (también se calentará menos).



MOSFET IRF840 canal N 500V, 8A

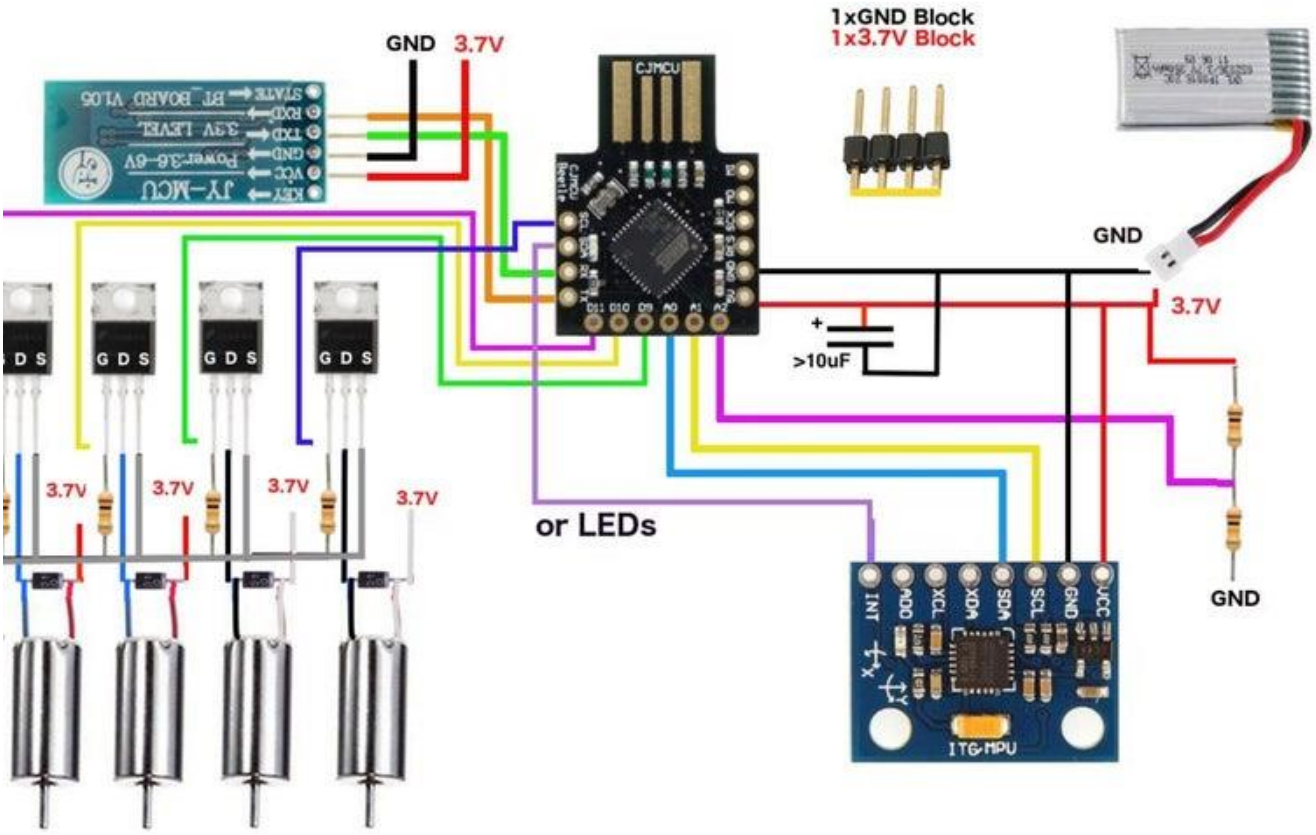
Ref: IRF840

Transistor mosfet de potencia canal N 500V, 8Amperes.

Otros componentes

Se utilizaron resistencias y diodos para la parte del circuito de conexión del motor; también utilizamos un cargador de pilas genérico para lipo’s.

Diagrama eléctrico



Programación

Código para leer el voltaje de la batería:

```
#define ALPHA 0.1

#define MULTIPLIER 6.67

float motorBattery;


void setup () {

  pinMode(A0, INPUT);

  analogReference(INTERNAL); // Set internal reference source of 1.1V
```

```
  // float tmp = analogRead(A0) / 1023.0 * MULTIPLIER; // make units Voltage
  motorBattery = 5.0;
}
void loop () {
  motorBattery = smoothBattery(motorBattery, analogRead(A0) / 1023.0 * MULTIPLIER, ALPHA);
}

float smoothBattery (float prevEntry, float newEntry, float alpha) {
  return (1-alpha) * prevEntry + alpha * newEntry;
}
```

Controlador PID

```
#define FL_MOTOR 3
#define FR_MOTOR 9
#define BR_MOTOR 10
#define BL_MOTOR 11
//---------------------------------PID-------------------------------------
//Define Variables we'll be connecting to
double rollSetpoint, rollInput, rollOutput;
double pitchSetpoint, pitchInput, pitchOutput;

//Define the aggressive and conservative Tuning Parameters<br>double consKp = 1, consKi = 0.05, consKd = 0.25;
PID pitchPID(&rollInput, &rollOutput, &rollSetpoint, consKp, consKi, consKd, DIRECT);
PID rollPID(&pitchInput, &pitchOutput, &pitchSetpoint, consKp, consKi, consKd, DIRECT);

void setup() {
//-----------------------------PID---------------------------------
  pitchInput = 0.0;
  rollInput = 0.0;
```

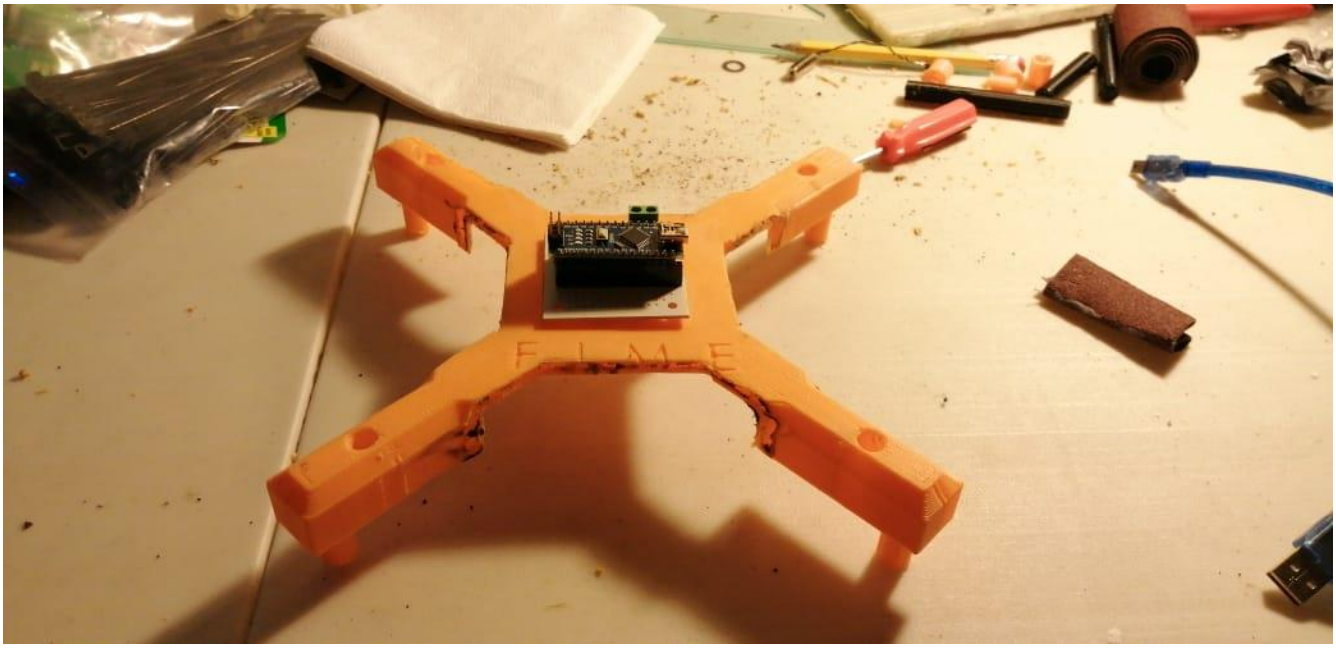
```
  pitchSetpoint = 0.0;
  rollSetpoint = 0.0;
  //turn the PID on
  pitchPID.SetMode(AUTOMATIC);
  rollPID.SetMode(AUTOMATIC);
  pitchPID.SetOutputLimits(-20, 20);
  rollPID.SetOutputLimits(-20, 20);
  //-----------------------------------------------------------------
  for (int i = 0; i < 4; i++) {
    targetSpeed[i] = 0;
  }

  pinMode(FL_MOTOR, OUTPUT);
  pinMode(FR_MOTOR, OUTPUT);
  pinMode(BR_MOTOR, OUTPUT);
  pinMode(BL_MOTOR, OUTPUT);

void loop() {
  pitchPID.Compute();
  rollPID.Compute();
  int actSpeed[4];
  stabilise (targetSpeed, actSpeed, rollOutput, pitchOutput);
  //    targetSpeed = actSpeed; // should this be here or not
}
void stabilise (int* currSpeed, int* actSpeed, float rollDiff, float pitchDiff) {
  //actual Speed is calculated as follows +- half rollDiff +- half pitchDiff
  actSpeed[0] = (int) currSpeed[0] + (rollDiff / 2) - (pitchDiff / 2);
  actSpeed[1] = (int) currSpeed[1] + (rollDiff / 2) + (pitchDiff / 2);
  actSpeed[2] = (int) currSpeed[2] - (rollDiff / 2) + (pitchDiff / 2);
  actSpeed[3] = (int) currSpeed[3] - (rollDiff / 2) - (pitchDiff / 2);
```

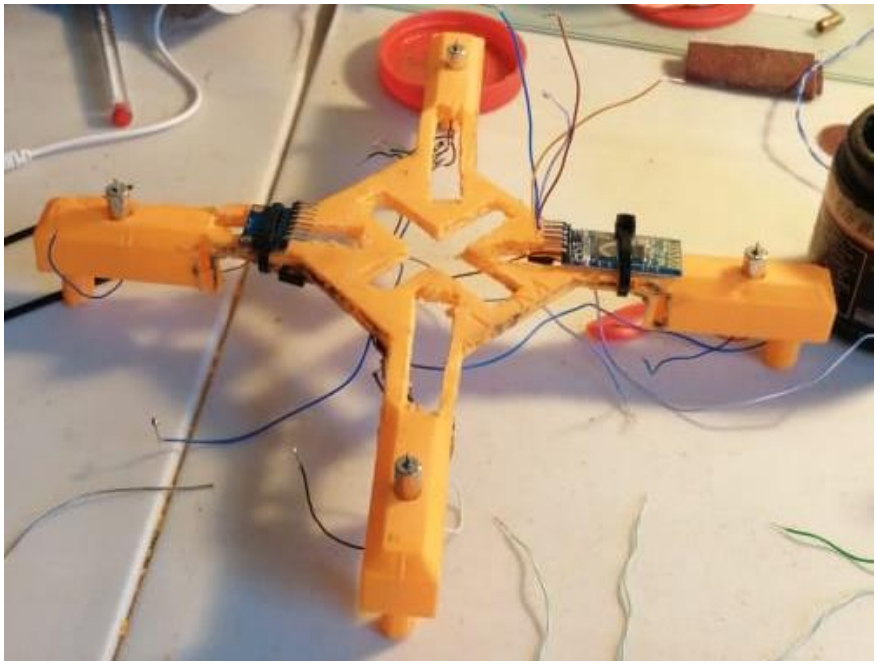
```
  for (int i = 0; i < 4; i ++) {
    if (actSpeed[i] < 0) actSpeed[i] = 0;
  }
}
void runIndividual (int* actSpeed) {
  analogWrite(FL_MOTOR, actSpeed[0]);
  analogWrite(FR_MOTOR, actSpeed[1]);
  analogWrite(BR_MOTOR, actSpeed[2]);
  analogWrite(BL_MOTOR, actSpeed[3]);
}
```

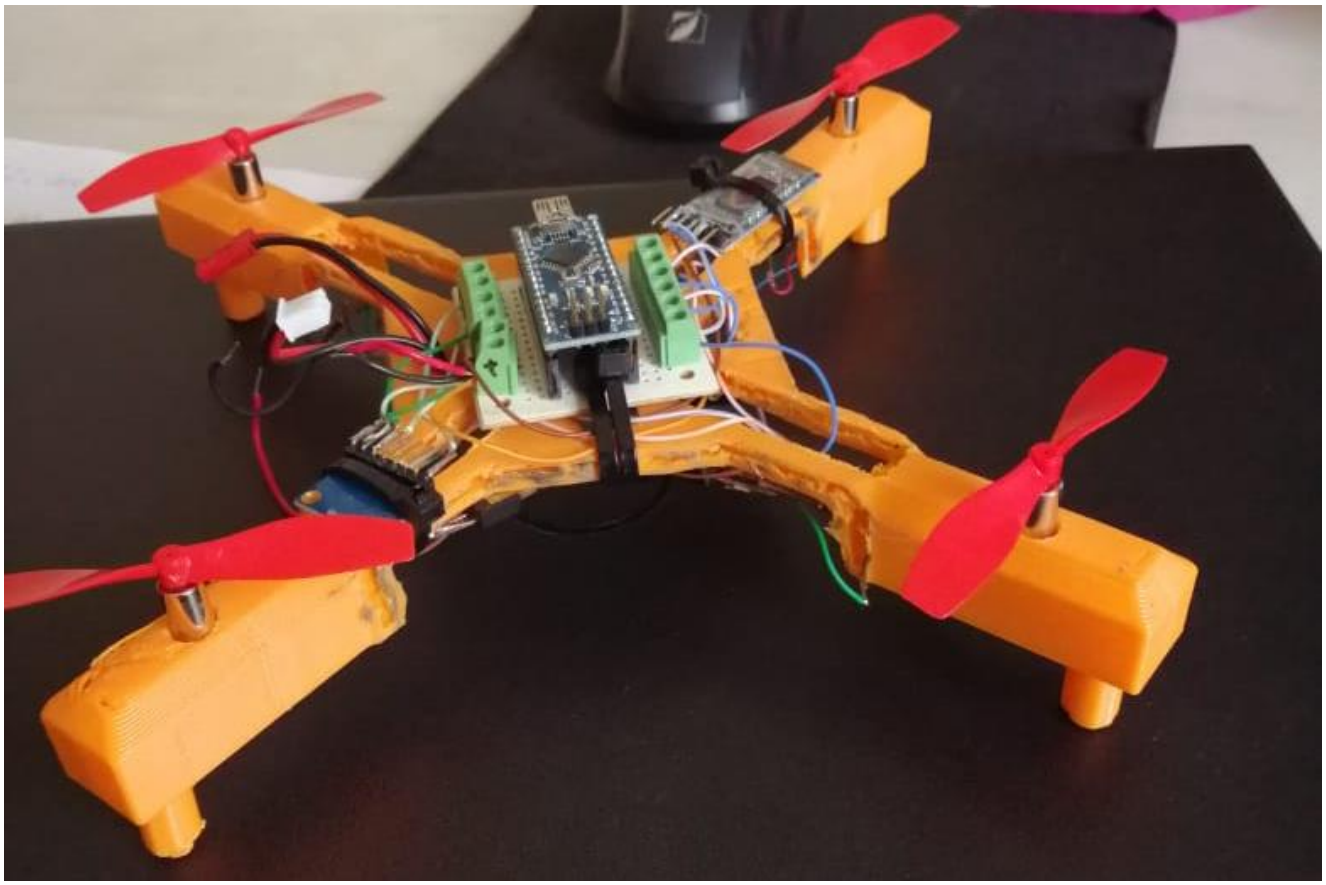
Código Bluetooth

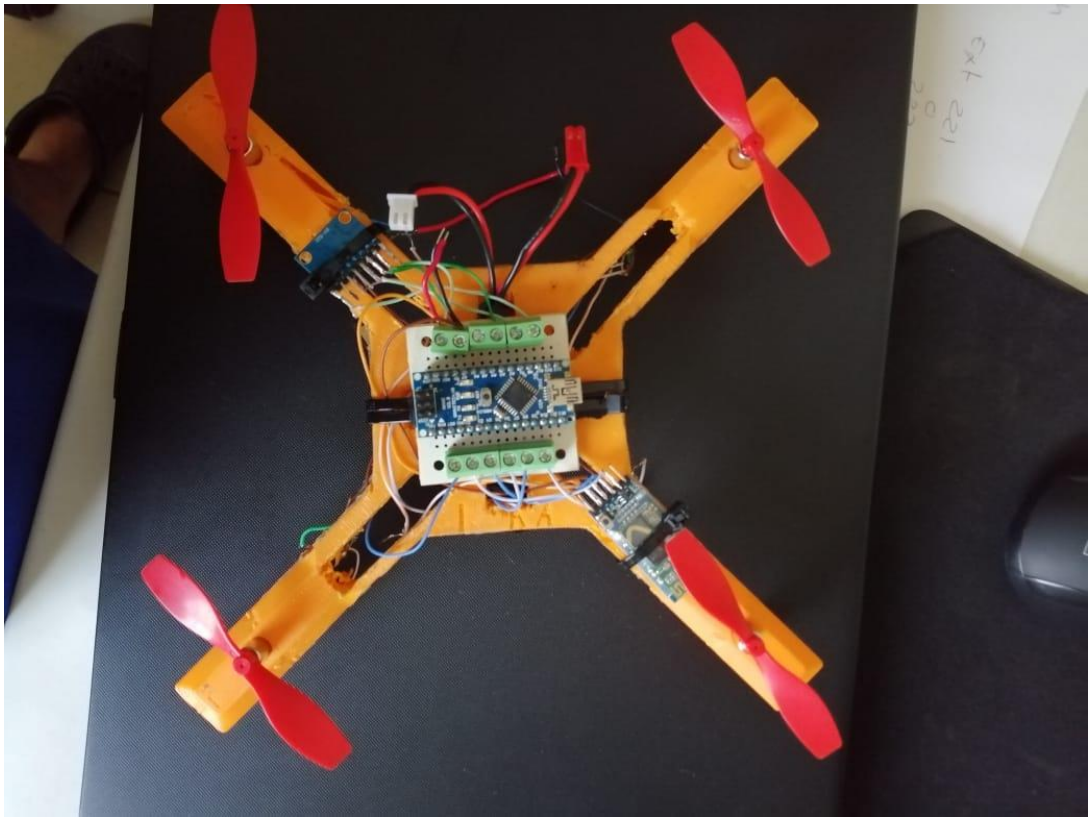
```
SoftwareSerial mySerial (7, 8); //RX, TX
void setup() {
  mySerial.begin(9600);
}
void loop () {
  if (mySerial.available()) {
  myReading = mySerial.parseInt();
    for (int i = 0; i < 4; i++) {
      targetSpeed[i] = myReading;
    }
    //flushing anything that wasn't read
    while (mySerial.available())
      mySerial.read();
  }
}
```

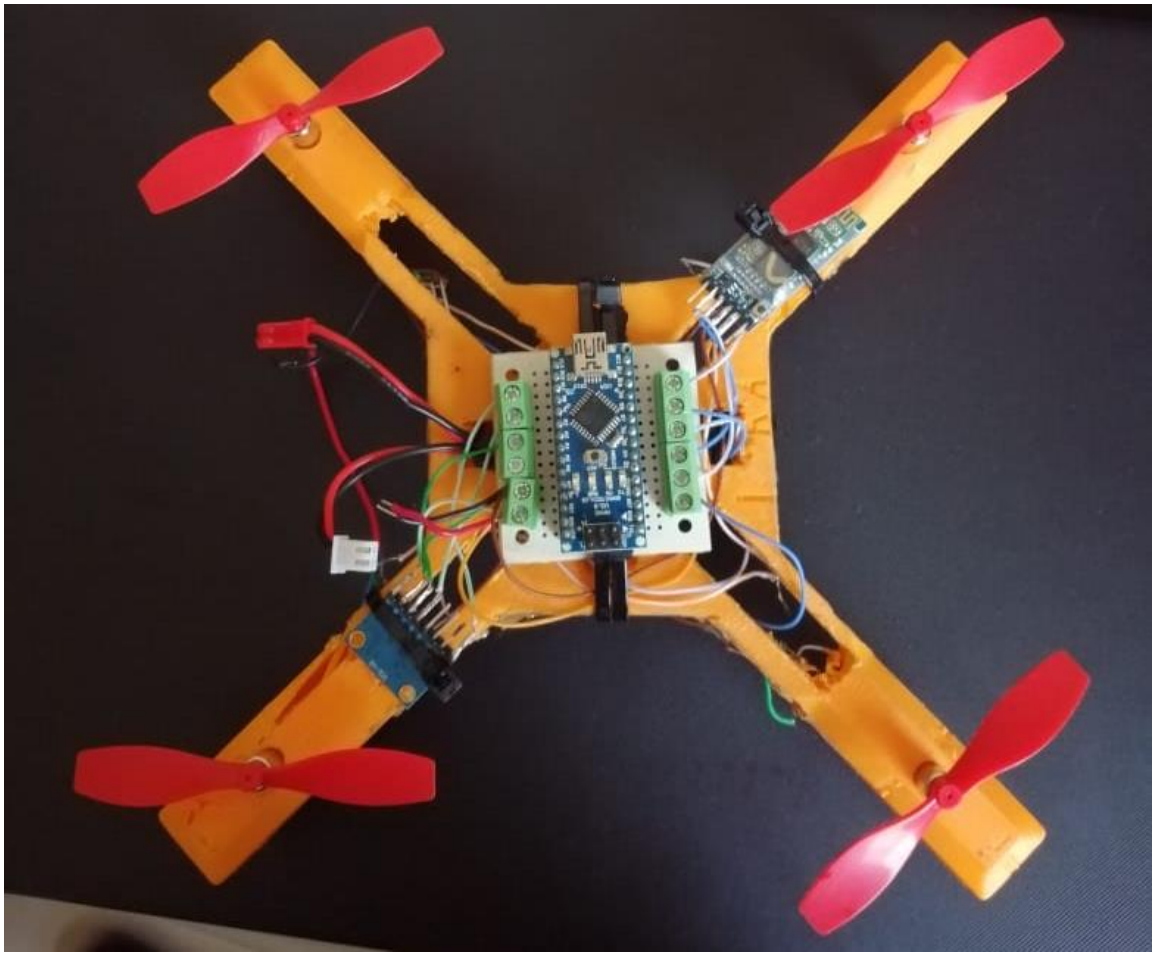
Armado del drone











Conclusiones

En este proyecto vimos lo complicado que es hacer un drone sin utilizar tarjetas controladoras de vuelo. Lo más difícil y el motivo por el cual no nos funcionó fue la parte mas tediosa de muchos proyectos: la programación. A pesar de tener un código con control pid, no pudimos hacerlo funcionar debido a que estuvimos batallando para entablar la comunicación entre el microcontrolador y la tarjeta de bluetooth.

Otra de las cosas que también nos llevo a tener muy poco tiempo para la programación y atrasarnos en el proyecto fue que el primer proveedor nos dio un producto que no le habíamos pedido. Contactamos a unas personas que imprimían en 3D y le mandamos un modelo que ya existía de un drone para COTIZARLO, esto era solo para darnos una idea de en cuanto nos iba a salir el drone que nosotros íbamos a diseñar después (esto lo hicimos porque tenemos muy poca experiencia imprimiendo en 3D y pues queríamos tener una referencia para ir ahorrando y ver más o menos cuanto nos gastaríamos); cuando ya teníamos el diseño en Solidworks ahora si les mandamos el archivo que queríamos que nos imprimieran en 3D, pero las personas nos imprimieron el primer documento que solamente queríamos cotizar. Sin contar que tuvieron un pésimo servicio, y nos dijeron que un día nos lo iban a tener y a la mera hora que no lo habían ni empezado, y luego cuando ya nos lo entregaron, estaba con una calidad pésima con un acabado malísimo.

Referencias

https://www.agelectronica.com/

https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html

https://www.naylampmechatronics.com/blog/12_Tutorial-B%C3%A1sico-de-Uso-del-M%C3%B3dulo-Bluetooth-H.html

https://www.didacticaselectronicas.com/index.php/semiconductores/transistor-mosfet-canal-n-500v,-8a-irf840-detail

https://www.instructables.com/id/Arduino-micro-Quadcopter/