ONDOKUZ MAYIS ÜNİVERSİTESİ

ELEKTRİK - ELEKTRONİK MÜHENDİSLİĞİ

Algılayıcılar ve Dönüştürücüler

Konu:MPU6050 Sensörü, İvme ve Gyro Verilerinin Okunması

Hazırlayanlar:

BATUHAN TOPAL / 17060071

Caner ADSOY / 17060032

MUHAMMET MERT ÇAKIR /17060101

ÖMER FARUK BOZ / 17060057

Ders Sorumlusu : Dr. Öğretim Üyesi İLYAS EMİNOĞLU

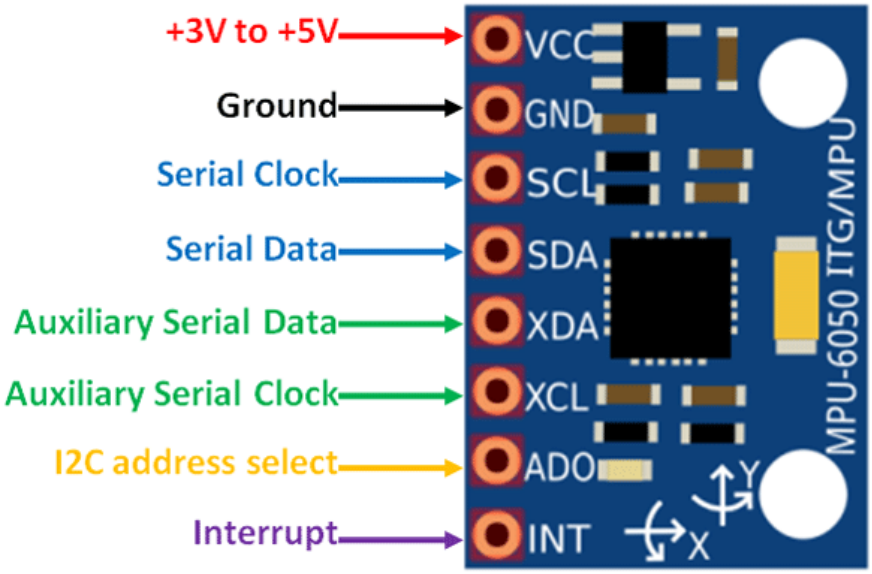
Samsun

2021

MPU6050 Sensörü

MPU6050 3 eksenli gyro ve 3 eksen açısal ivme ölçer olan IMU sensörüdür. Toplam 6 çıkış veren bu sensör iletişim için I2C protokolünü kullanır.I2C'de veri iletişimi için SCL ve SDA hatları bulunur. MPU 6050 gibi IMU sensörleri elektronik cihazların çoğunda kullanılır. Akıllı saatler, Fitbit bantları, Robotlar, Drone ,Gimball ve akıllı telefonlarda kullanılmaktadır.

MPU6050 Sensör Pinleri

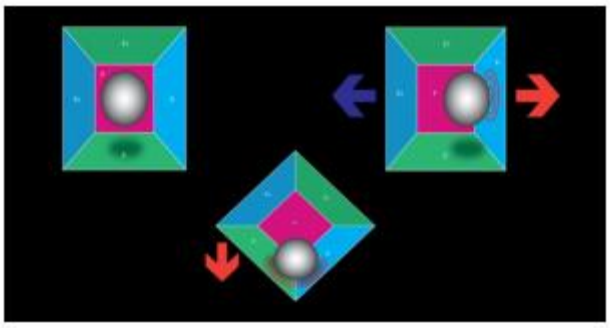


1.İvme Ölçer

A-Özellikleri

- Programlanabilir tam ölçek aralığı ± 2g, ± 4g, ± 8g ve ± 16g olan dijital çıkışlı üç eksenli ivmeölçer
- Entegre 16 bit ADC'ler, herhangi bir harici çoklayıcı gerektirmeden ivmeölçerlerin eşzamanlı örneklemesini sağlar
- İvmeölçer normal çalışma akımı: 500µA
- Düşük güç ivmeölçer modu akımı: 1.25Hz'de 10µA, 5Hz'de 20µA, 20Hz'de 60µA, 40Hz'de 110µA
- Yön tespiti ve sinyal verme
- Dokunma algılama
- Kullanıcı tarafından programlanabilen kesintiler(interrupt)

İvme ölçer Piezoelektrik etkisi ile çalışmaktadır. Piezoelektrik bazı malzemelerin üzerine mekanik olarak bir kuvvet uygulanması sonucunda ortaya çıkan elektriktir. Sensör kuvvet olarak yer çekimini kullanarak değer verir.



B-İvme Ölçer Kullanım Alanları

Akselometre (ivmeölçer) akıllı telefonlarımızda bulunur ve pusula uygulamasının doğru çalışmasını sağlar. Yüksek hassasiyete sahip ivme ölçerler gemi, uçak ,denizaltılarda kullanılır. Teknolojik ciharlarda titreşimi ölçer ve belirli aralıkta kalmasını sağlar ayrıca kameralar da kullanım alanındandır.

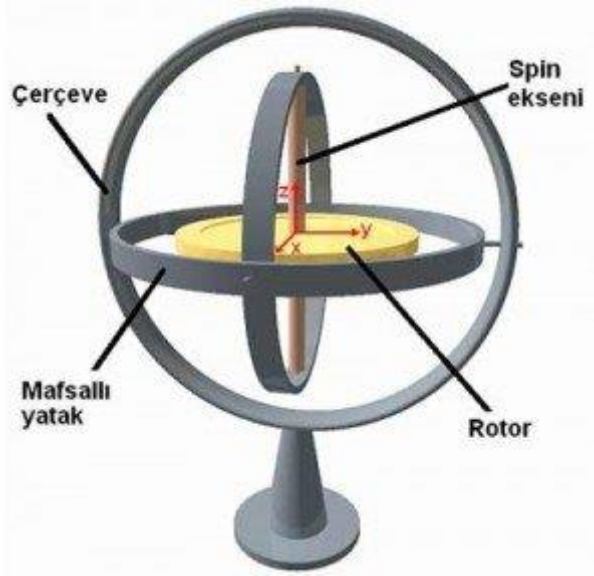
2.Jiroskop

A-Özellikleri

- ± 250, ± 500, ± 1000 ve ± 2000 ° / sn'lik kullanıcı tarafından programlanabilir tam ölçek aralığına sahip dijital çıkışlı X, Y ve Z Ekseni açısal oran sensörleri (jiroskoplar)
- FSYNC pinine bağlı harici senkronizasyon sinyali; görüntü, video ve GPS senkronizasyonunu destekler

- Entegre 16 bit ADC'ler, jiroskopların aynı anda örneklenmesini sağlar
- Gelişmiş önyargı ve hassasiyet sıcaklık kararlılığı, kullanıcı kalibrasyonu ihtiyacını azaltır
- Geliştirilmiş düşük frekanslı gürültü performansı
- Dijital olarak programlanabilir düşük geçişli filtre
- Jiroskop çalışma akımı: 3.6mA
- Bekleme akımı: 5µA
- Fabrikada kalibre edilmiş hassasiyet ölçek faktörü

Jiroskop, dönen bir disktir, çark olarak da bilinir. Merkezkaç kuvveti ile çalışır. Bir jiroskop, dönen bir çark, rotor ve eksenden oluşur. Eksen, rotor içinde dönebileceği bir çembere yataklanmış şekildedir. Çember de dik açı ile başka bir çembere bağlanmıştır. İç ve dış çemberle dik açı yapan bir çerçeveye kenetlenmiş şekilde bir de dış çemberi bulunur. Dönüş ekseni her hangi bir yönde olabilir. Jiroskopların dışındaki çerçeve dengeleme çemberi ile desteklenmiştir



B-Jiroskop Kullanım Alanları

Jiroskop açısal hızı ölçmek veya korumak için kullanılan bir cihazdır. Elektronik cihazlarda bulunan mikroçip paketlenmiş MEMS jiroskopları, katı hal halka lazerleri, fiber optik jiroskoplar ve son derece hassas kuantum jiroskopu gibi diğer çalışma prensiplerine dayalı jiroskoplar da mevcuttur. Jiroskop uygulamaları Hubble Teleskopu gibi veya suyun altındaki bir denizaltının çelik gövdesinin içindeki navigasyon sistemlerini içerir. Hassasiyetlerinden dolayı jiroskoplar, jeotheodolitlerde, tünel madenciliğinde yönünü korumak için de kullanılır. Jiroskopların, manyetik pusulaları gemilerde, uçaklarda ve uzay gemilerinde, genel olarak taşıtlarda stabiliteyi sağlamak için kullanılır.

Uçak Hareket Eksenleri

Uçaklar 3 eksen üzerinde hareket ederler. Bunlar Longitudinal Axis (Boylamsal Eksen), Vertical Axis (Dikey Eksen) ve Lateral Axis (Yatay Eksen) dir.

Boylamsal Eksen: Bir uçağın ağırlık merkezinden geçen burnundan kuyruğuna uzanan eksendir. Uçağın bu eksen etrafında yaptığı harekete (Roll) yatış hareketi denir.

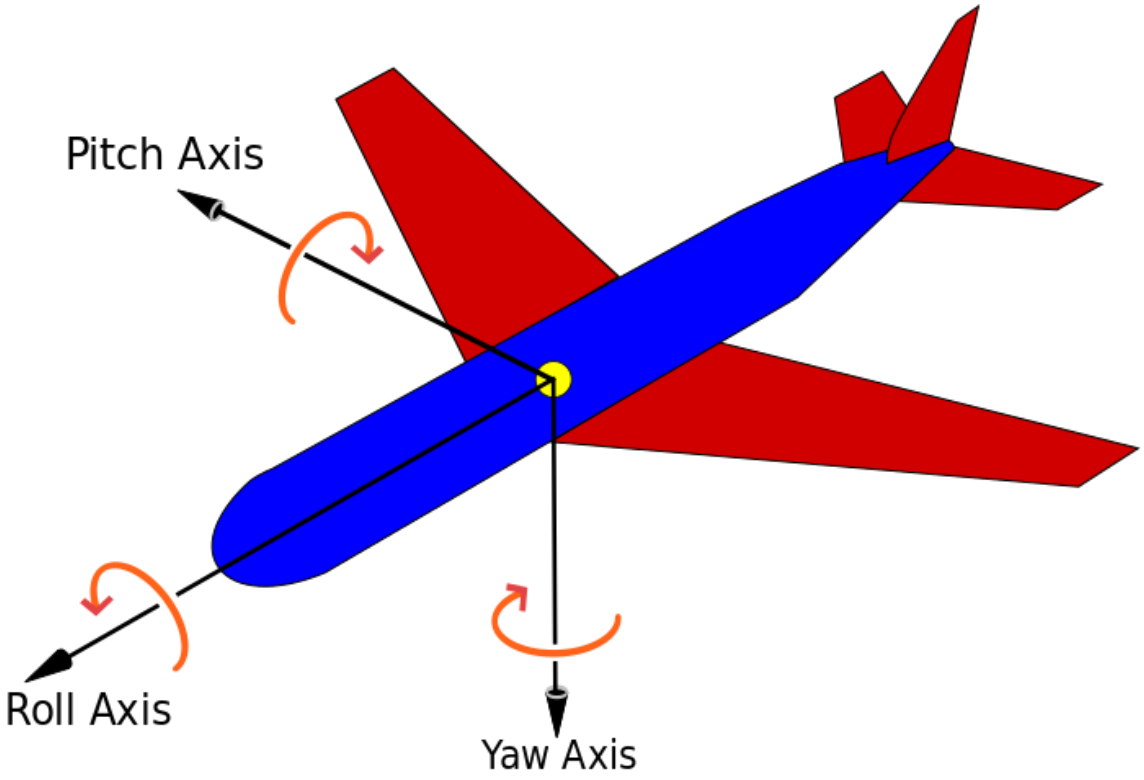
Uçağın bu eksen etrafında yaptığı hareketler aileron, elevon veya spoiler ile kontrol edilir.

Yatay Eksen: Uçağın ağırlık merkezinden geçerek bir kanat ucundan diğer kanat ucuna doğru uzanan eksendir. Bir uçağın bu eksen etrafında yaptığı harekete (Pitch) yunuslama denir.

Uçağın bu eksen etrafında yaptığı yunuslama hareketi, irtifa dümeni (elevator), hareketli yatay stabilize (stabilizator) ve bazı uçaklarda elevonlar tarafından kontrol edilir.

Dikey Eksen: Uçağın ağırlık merkezinden geçerek gövde üst kısmından gövde alt kısmına uzanan eksendir. Bir uçağın dikey eksen etrafında yaptığı harekete (Yaw) Dönme hareketi denir.

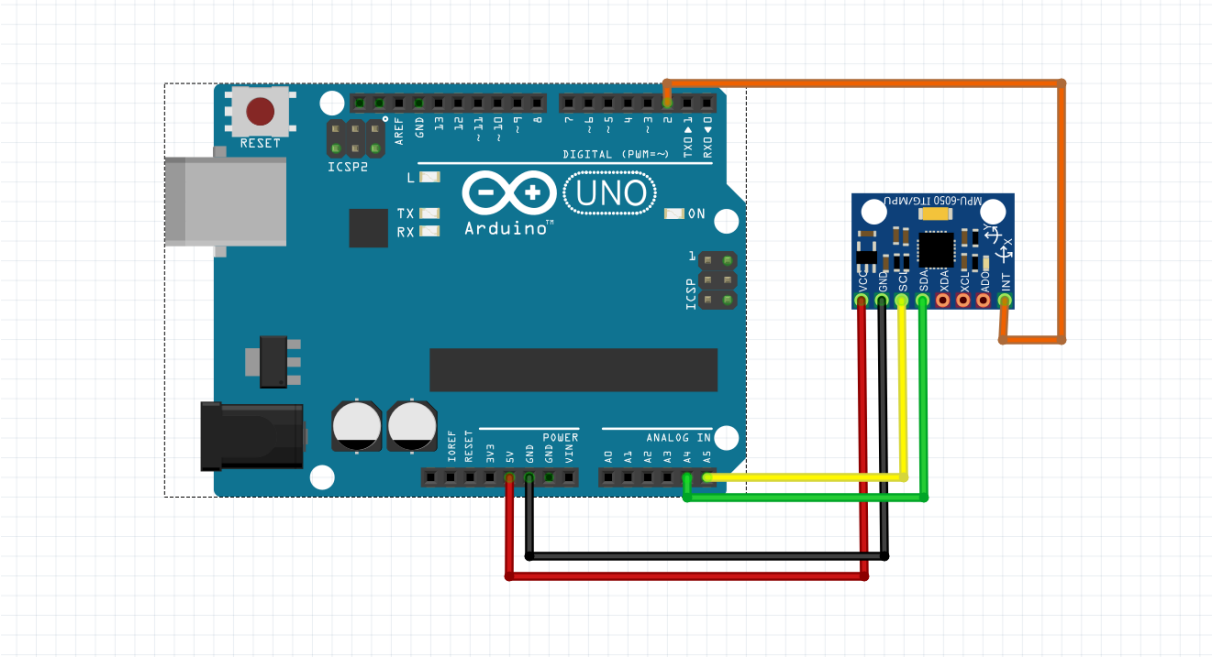
Uçağın dikey eksen etrafındaki hareketi (rudder) istikamet dümeni tarafından sağlanır.



Arduino MPU6050 Sensörü Kullanımı

1.Uygulama: Sensörden Gelen Veriler ile Nesnenin Hareketinin Sağlanması

Bağlantı Seması



Arduino Kodları

MPU6050_DMP6 | Arduino 1.8.13

Dosya Düzenle Taslak Araçlar Yardım

MPU6050_DMP6 §

```
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif
```

Kullandığımız kütüphaneleri dahil ettik.

```
MPU6050 mpu;
#define OUTPUT_READABLE_YAWPITCHROLL
#define OUTPUT_TEAPOT

#define INTERRUPT_PIN 2  // use pin 2 on Arduino Uno & most boards
#define LED_PIN 3 // (Arduino is 13, Teensy is 11, Teensy++ is 6)
bool blinkState = false;
```

MPU6050 Sensörüne bir isim verdik.
Yaw,Pitch ve Roll hareketlerinin tanımladık.
Ledin ilk durumunu False ayarladık

```
// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer
uint16_t aci_y ;
```

MPU'nun veri okunması için gerekli değişkenleri tanımladık

```
// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector
```

Sensörden gelen verilerin içine kaydedileceği dizileri tanımladık

```
// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high
void dmpDataReady() {
    mpuInterrupt = true;
}
```

```
void setup() {
    // join I2C bus (I2Cdev library doesn't do this automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
        Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation difficulties
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif
```

İ2C Haberleşmesini başlattık

```
    Serial.begin(9600);
    while (!Serial); // wait for Leonardo enumeration, others continue immediately
```

Verilerin Ekranda görülmesi için Serial Haberleşmesi Başlattık

```
    // initialize device
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();
    pinMode(INTERRUPT_PIN, INPUT);
```

MPU6050 'yi başlattık.
İnterrupt(Kesme) Pinini INPUT olarak belirledik

```
    // verify connection
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));
```

```
    // wait for ready
    Serial.println(F("\nSend any character to begin DMP programming and demo: "));
    while (Serial.available() && Serial.read()); // empty buffer
    while (!Serial.available());                 // wait for data
    while (Serial.available() && Serial.read()); // empty buffer again

    // load and configure the DMP
    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();
```

Verilerin okunmadan önceki son hazırlıklarını yaptık
Buffer sıfırlandı(Verilerin içine yazılacağı diziler)

```
    // supply your own gyro offsets here, scaled for min sensitivity
    mpu.setXGyroOffset(220);
    mpu.setYGyroOffset(76);
    mpu.setZGyroOffset(-85);
    mpu.setZAccelOffset(1788); // 1688 factory default for my test chip
```

MPU Offset degerleri ayarlandı.

```
// make sure it worked (returns 0 if so)
if (devStatus == 0) {
    // Calibration Time: generate offsets and calibrate our MPU6050
    mpu.CalibrateAccel(6);
    mpu.CalibrateGyro(6);
    mpu.PrintActiveOffsets();
    // turn on the DMP, now that it's ready
    Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);

    // enable Arduino interrupt detection
    Serial.print(F("Enabling interrupt detection (Arduino external interrupt "));
    Serial.print(digitalPinToInterrupt(INTERRUPT_PIN));
    Serial.println(F(")..."));
    attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();

    // set our DMP Ready flag so the main loop() function knows it's okay to use it
    Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;

    // get expected DMP packet size for later comparison
    packetSize = mpu.dmpGetFIFOPacketSize();
} else {
    // ERROR!
    // 1 = initial memory load failed
    // 2 = DMP configuration updates failed
    // (if it's going to break, usually the code will be 1)
    Serial.print(F("DMP Initialization failed (code "));
    Serial.print(devStatus);
```

Sensörün çalışır durumda olduğu kontrol edilerek döngünün içine girilir.
Burada kalibrasyon ayarları yapılır. Ardından Kesme için kullanıcıdan bir harf girilmesi istenir.

Eğer ki sensörün bağlantısında bir sorun olursa Kullanıcıya hata mesajını bildirir

```
    // configure LED for output
    pinMode(LED_PIN, OUTPUT);
}
```

Led ,Çıkış pini olarak tanımlandı.

```
void loop() {
    // if programming failed, don't try to do anything
    if (!dmpReady) return;
    // read a packet from FIFO
    if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) { // Get the Latest packet
        #ifdef OUTPUT_READABLE_QUATERNION
            // display quaternion values in easy matrix form: w x y z
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            Serial.print("quat\t");
            Serial.print(q.w);
            Serial.print("\t");
            Serial.print(q.x);
            Serial.print("\t");
            Serial.print(q.y);
            Serial.print("\t");
            Serial.println(q.z);
        #endif

        #ifdef OUTPUT_READABLE_EULER
            // display Euler angles in degrees
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetEuler(euler, &q);
            Serial.print("euler\t");
            Serial.print(euler[0] * 180/M_PI);
            Serial.print("\t");
            Serial.print(euler[1] * 180/M_PI);
```

Sonsuz döngünün içine girildi.

Eğer program çalışmazsa programı sonlandıran kod en başa yazıldı

```
#ifdef OUTPUT_READABLE_YAWPITCHROLL
    // display Euler angles in degrees
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
    Serial.print("ypr\t");
    Serial.print(ypr[0] * 180/M_PI);
    Serial.print("\t");
    Serial.print(ypr[1] * 180/M_PI);
    Serial.print("\t");
    Serial.println(ypr[2] * 180/M_PI);
#endif
```

Yaw,Pitch,Roll Hareketlerinden elde edilen veriler gerekli formüller kullanılarak hesaplandı ve ekrana yazdırıldı

```
#ifdef OUTPUT_READABLE_REALACCEL
    // display real acceleration, adjusted to remove gravity
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetAccel(&aa, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
    Serial.print("areal\t");
    Serial.print(aaReal.x);
    Serial.print("\t");
    Serial.print(aaReal.y);
    Serial.print("\t");
    Serial.println(aaReal.z);
#endif
```

MPU6050 Sensöründen gelen açı değerlerini ham olarak ekrana yazdırılması istenirse kodun başına tanımlanmalıdır.

```
#ifdef OUTPUT_READABLE_WORLDACCEL
    // display initial world-frame acceleration, adjusted to remove gravity
    // and rotated based on known orientation from quaternion
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetAccel(&aa, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
    mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);
    Serial.print("aworld\t");
    Serial.print(aaWorld.x);
    Serial.print("\t");
    Serial.print(aaWorld.y);
    Serial.print("\t");
    Serial.println(aaWorld.z);
#endif

#ifdef OUTPUT_TEAPOT
    // display quaternion values in InvenSense Teapot demo format:
    teapotPacket[2] = fifoBuffer[0];
    teapotPacket[3] = fifoBuffer[1];
    teapotPacket[4] = fifoBuffer[4];
    teapotPacket[5] = fifoBuffer[5];
    teapotPacket[6] = fifoBuffer[8];
    teapotPacket[7] = fifoBuffer[9];
    teapotPacket[8] = fifoBuffer[12];
    teapotPacket[9] = fifoBuffer[13];
    Serial.write(teapotPacket, 14);
    teapotPacket[11]++; // packetCount, loops at 0xFF on purpose
#endif
```

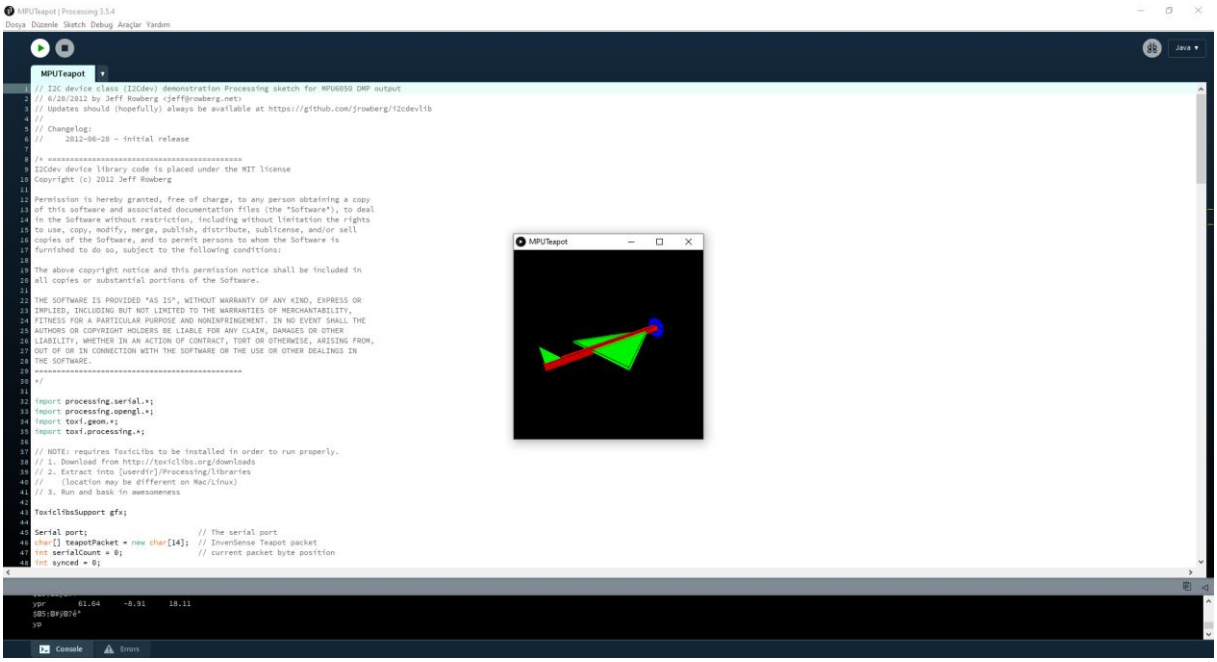
```
    // blink LED to indicate activity
    blinkState = !blinkState;
    digitalWrite(LED_PIN, blinkState);
```

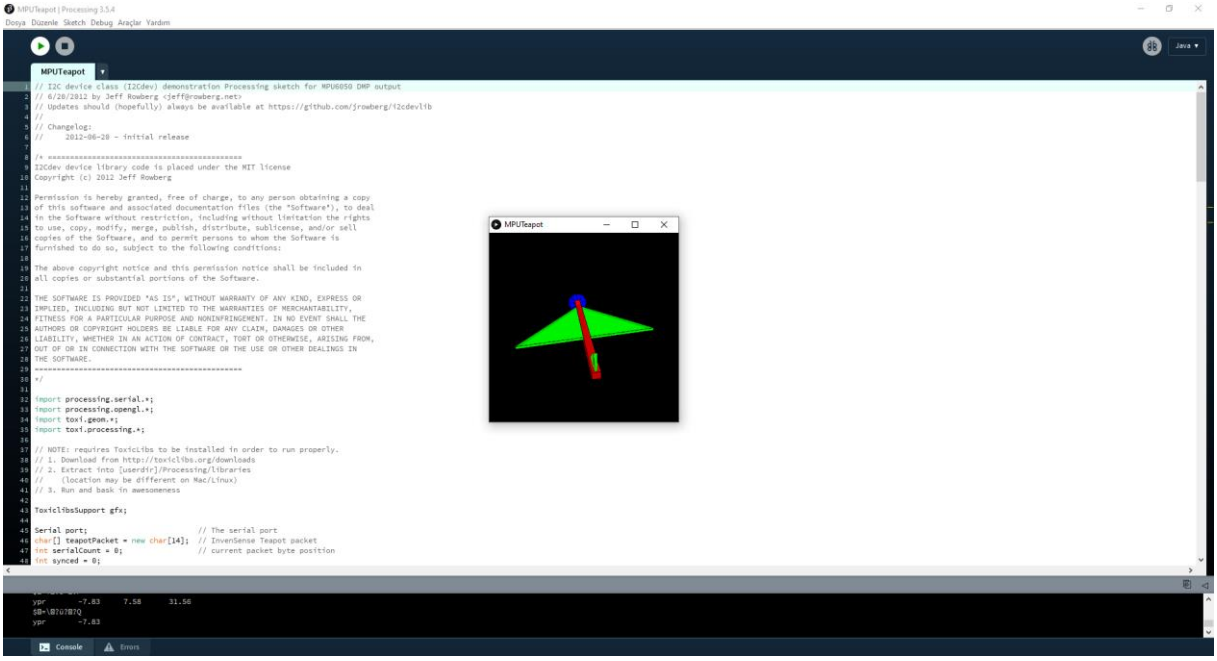
Verilerin geldiğini anlamak için devreye bağlanan led ,her veri geldiğinde yanıp söner

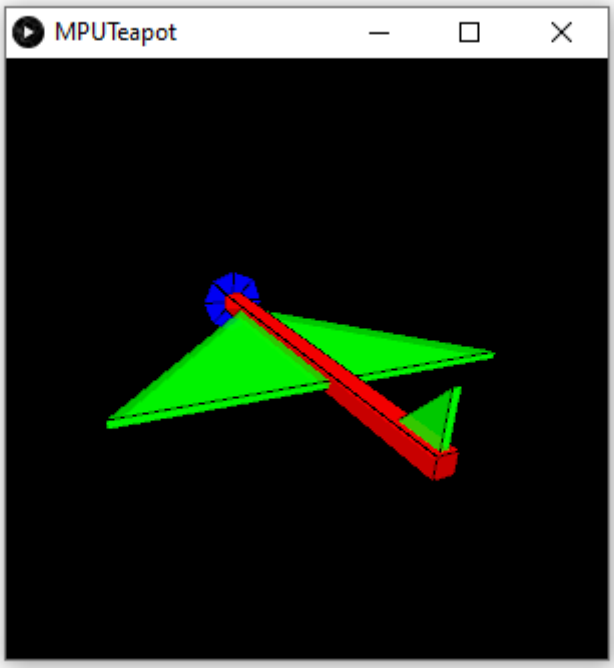
Kodlarımızı Arduino Uno kartımıza yüklüyoruz…

Processing Uygulamasına yüklediğimiz kodları çalıştırdıktan bir kaç saniye sonra veriler Arduino'dan Ekrana(Processing) aktarılır.

Sensörü x,y,z ekselerinde hareket ettirdiğimizde ekrandaki model uçağın aynı eksenlerde hareket ettiğini görebiliriz.







2.Uygulama: Sensörden Gelen İşlenmemiş İvme ve Gyro Verilerinin Ekranda Görülmesi

Arduino Kodları

mpu6050deneme§

```
#include <I2Cdev.h>

#include <helper_3dmath.h>
#include <MPU6050.h>
```

İ2C ve MPU6050
Kütüphaneleri dahil edildi.

```
MPU6050 accelgyro; // Mpu6050 sensör tanımlama
int16_t ax, ay, az; //ivme tanımlama
int16_t gx, gy, gz; //gyro tanımlama
```

Sensör tanımlandı.
İvme ve gyro verileri tanımlandı

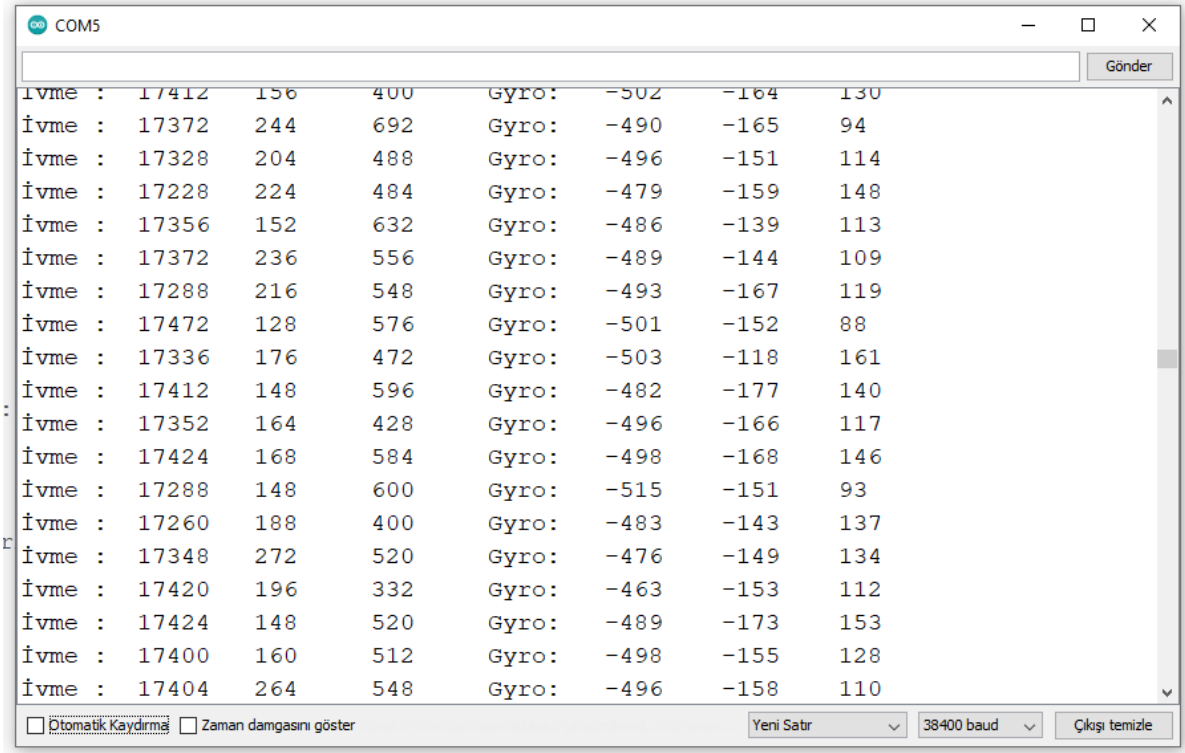
```
void setup() {
Wire.begin();
Serial.begin(38400);
Serial.println("I2C cihazlar başlatılıyor...");
accelgyro.initialize();
Serial.println("Test cihazı bağlantıları...");
Serial.println(accelgyro.testConnection() ? "MPU6050 bağlantı başarılı" : "MPU6050 bağlantısı başarısız");
}
```

İ2C Haberleşmesi başlatıldı.
Sensör bağlantısı kontrol edildi.
Durum raporu uygun bir mesajla ekrana yazdırıldı

```
void loop() {
accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); // ivme ve gyro değerlerini okuma
 //açısal ivmeleri ve gyro değerlerini ekrana yazdırma
Serial.print("İvme :\t");
Serial.print(ax); Serial.print("\t");
Serial.print(ay); Serial.print("\t");
Serial.print(az); Serial.print("\t");
Serial.print("Gyro:"); Serial.print("\t");
Serial.print(gx); Serial.print("\t");
Serial.print(gy); Serial.print("\t");
Serial.println(gz);
}
```

Sensörden gelen ivme ve gyro değerleri değişkenlerin içine aktarıldı.

İvme ve Gyro Değerleri Ekrana yazdırılır.


COM5
Gönder
İvme : 17412 156 400 Gyro: -502 -164 130
İvme : 17372 244 692 Gyro: -490 -165 94
İvme : 17328 204 488 Gyro: -496 -151 114
İvme : 17228 224 484 Gyro: -479 -159 148
İvme : 17356 152 632 Gyro: -486 -139 113
İvme : 17372 236 556 Gyro: -489 -144 109
İvme : 17288 216 548 Gyro: -493 -167 119
İvme : 17472 128 576 Gyro: -501 -152 88
İvme : 17336 176 472 Gyro: -503 -118 161
İvme : 17412 148 596 Gyro: -482 -177 140
İvme : 17352 164 428 Gyro: -496 -166 117
İvme : 17424 168 584 Gyro: -498 -168 146
İvme : 17288 148 600 Gyro: -515 -151 93
İvme : 17260 188 400 Gyro: -483 -143 137
İvme : 17348 272 520 Gyro: -476 -149 134
İvme : 17420 196 332 Gyro: -463 -153 112
İvme : 17424 148 520 Gyro: -489 -173 153
İvme : 17400 160 512 Gyro: -498 -155 128
İvme : 17404 264 548 Gyro: -496 -158 110
Otomatik Kaydırma
Zaman damgasını göster
Yeni Satır
38400 baud
Çıkışı temizle