```
/* ESTA FUNCIÓN DEVUELVE LOS DATOS QUE SE QUIEREN OBTENER DEL MPU6050 (PITCH, YAW Y ROLL)*/
void IMU() {
  fifoCount = mpu.getFIFOCount(); //Se obtiene el número de bytes del DMP
  if (fifoCount >= 42) { //Si el número de datos en el DMP es mayor a 42 bytes (mayor que el packetSize)
    mpu.resetFIFO(); //Se resetea el FIFO
    fifoCount = mpu.getFIFOCount(); //Se vuelven a obtener el número de bytes en el FIFO del DMP
  }
  while (fifoCount < packetSize) { //Mientras el número de datos es menor al paquete
    fifoCount = mpu.getFIFOCount(); //Se vuelven a obtener el número de bytes en el FIFO del DMP
    delay(1); //El delay es necesario para controlar el fifoCount. No se ha podido ver exactamente como afecta porque al crear un println el problema se resuelve por el incremento de tiempo.
    if (fifoCount > packetSize) { //Si el número de datos en el DMP es mayor a 42 bytes (mayor que el packetSize)
      mpu.resetFIFO(); //Se resetea el FIFO
      fifoCount = mpu.getFIFOCount(); //Se vuelven a obtener el número de bytes en el FIFO del DMP
    }
  }
  fifoCount = fifoCount - packetSize; //Se resta al fifoCount el packetSize (el resultado debería ser siempre 0)
  mpu.getFIFOBytes(fifoBuffer, packetSize); //Se obtienen los valores del DMP en forma de bytes
  mpu.dmpGetQuaternion(&q, fifoBuffer); //Se obtiene el cuaternión de actitud mediante el DMP
  QuatToEuler(q, yaw, pitch, roll); //Se hace el paso del cuaternión de actitud a los ángulos de Euler


  if (inici == true) {

    I2Cread;

    // ---  Lectura acelerometro y giroscopio ---
    uint8_t Buf[14];
    I2Cread(MPU9250_ADDRESS, 0x3B, 14, Buf);

    // Convertir registros acelerometro
    ax = -(Buf[0] << 8 | Buf[1]);
    ay = -(Buf[2] << 8 | Buf[3]);
    az = Buf[4] << 8 | Buf[5];

    // Convertir registros giroscopio
    gx = -(Buf[8] << 8 | Buf[9]);
    gy = -(Buf[10] << 8 | Buf[11]);
    gz = Buf[12] << 8 | Buf[13];
    //Conversion datos de raw a m/s2
    ax_m_s2 = ax * (9.81 / 16384.0);
    ay_m_s2 = ay * (9.81 / 16384.0);
    az_m_s2 = az * (9.81 / 16384.0);
    //float gx_deg_s = gx * (250.0/32768.0);
    //float gy_deg_s = gy * (250.0/32768.0);
    //float gz_deg_s = gz * (250.0/32768.0);

    Wire.begin();

    // Yaw, Pitch, Roll
    Serial.print( yaw  );
    Serial.print("\t");
    Serial.print(pitch );
    Serial.print("\t");
    Serial.print( roll);
    Serial.print("\t");
```

```
      // Quaternion
      Serial.print(q.x);
      Serial.print("\t");
      Serial.print(q.y);
      Serial.print("\t");
      Serial.print(q.z);
      Serial.print("\t");
      Serial.print(q.w);
      Serial.print("\t");

      // Acelerometro
      Serial.print(ax);
      Serial.print("\t");
      Serial.print(ay);
      Serial.print("\t");
      Serial.print(az);
      Serial.print("\t");

      // Giroscopio
      Serial.print(gx);
      Serial.print("\t");
      Serial.print(gy);
      Serial.print("\t");
      Serial.print(gz);
      Serial.print("\t");
      mostra = mostra + 1;
      Serial.print(mostra);
      Serial.print("\t");

      Serial.println("");

      while (millis() < espera) {
      }

      espera = espera + 50;
    }
    else {
      inici = digitalRead(boto);
    }
  }
  //Lectura de los valores del Acelerometro
  void I2Cread(uint8_t Address, uint8_t Register, uint8_t Nbytes, uint8_t* Data)
  {
    Wire.beginTransmission(Address);
    Wire.write(Register);
    Wire.endTransmission();
    Wire.requestFrom(Address, Nbytes);
    uint8_t index = 0;
    while (Wire.available())
      Data[index++] = Wire.read();
  }
```