Wikisivu
http://wiki.helsinki.hacklab.fi/index.php/AirCore

DIY Air-Core/Air Coil/Magnetic Movement

Koodia, 3D-malleja jne keräillään jatkossa githubiin https://github.com/HelsinkiHacklab/aircores

Mike's Flight Deck is an introduction to home cockpit building:

- http://www.mikesflightdeck.com/oldnews/oldnews_2011.html
- http://www.mikesflightdeck.com/instruments/diy_aircore_instruments.html
- http://www.mikesflightdeck.com/mfdb/bsai/bsai.html

HI6SIM Aircoil motor in real instrument:

- http://www.youtube.com/watch?v=eXws-wsT8JQ

DIY closed-loop Galvanometer

- http://elm-chan.org/works/vlp/report_e.html

Magneetteja:

- http://www.kjmagnetics.com/categories.asp
  - Tilattu (rambo):
    - **Part No. | Unit Price|Quantity|Total**
    - RA2ADIA $7.08 | 2 | $14.16
    - R828DIA $3.87 | 5 | $19.35
    - R6036DIA $1.88 | 10 | $18.80
    - R424DIA $0.69 | 50 | $34.50
  - Mitatttu (suovula) - ulko / sisä x pituus:
    - 2 × R828DIA : ⌀ **15.90** / **3.18** × 15.90 mm, 22.6 g
    - 5 × R828DIA : ⌀ **12.70** / **3.18** × 12.70 mm, 11.3 g
    - 10 × R6036DIA : ⌀ **9.53** / **2.38** × 9.53 mm, 4.77 g
    - 50 × R424DIA : ⌀ **6.35** / **3.18** × 6.35 mm, 1.14 g
    - suovula etsiskelee parhaillaan akseleita, laakereita, conduittia jne.
      - Akselit
        - http://www.silshop.fi/shop/index.php?act=viewProd&productId=3400
        - http://www.silshop.fi/shop/index.php?act=viewProd&productId=3401
        - http://www.silshop.fi/shop/index.php?act=viewProd&productId=369
        - http://www.silshop.fi/shop/index.php?act=viewProd&productId=370
      - laakerit (Huom magneettien ulkomitat on [usein] isompia kuin laakereiden joten sitä coilformin muotoa saa miettiä)
        - 2mm
          - http://www.vxb.com/page/bearings/PROD/2mm/MR62ZZ
            - Jenkkifirma, toimitusajat ja postikulut täysin tuntemattomat
          - http://fi.rsdelivers.com/product/nmb/ddr-620zzha1p24ly121/metric-plain-bearing-2x6x3/6125723.aspx
            - RS:llä toimitusajat YE:hen tilattuna on yleensä pari-

kolme päivää
- http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=12008
  - HK-firma, postikulut halvat, toimitusajat pitkät
- http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=9234

- 3mm
  - http://www.vxb.com/page/bearings/PROD/3mm/MR63ZZ
  - http://fi.rsdelivers.com/product/rs/623-2z/single-row-shld-3mm-id/6189884.aspx <- Rambo tilannut muutaman näitä **3x10x4 mm**
  - http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=9247
  - http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=11523
  - http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=11726

- Liukulaakerit (tai voihan sitä tehdä keskelle palloillekin sopivan uran) voi tosiaan sorvata PTFE:stä (eli teflonista) jos vaan saa jotain sopivan mittatarkkaa ja ei-magneettista putkea jonka sisään koko paketti tungetaaan (jos on hätä niin sen putkenkin voi sorvata tarkaksi mutta se on vähän isompi operaatio (varsinkin kun pientä putkea on tosi paha sorvata sisäpuolelta yhtään pidempiä pätkiä kerralla)

- Multilayer Air Core Inductor Calculator: http://www.pronine.ca/multind.htm

- http://www.supermagnetman.net/index.php?cPath=37

AVR XMega:

- http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3808

Pohdintaa:

- 3D tulostettu coilform jonka sisään tungetaan joku mittatarkempi putki jossa akseli+laakerit
  - kannattaa mitoittaa siten että menee siististi jonkun vakiomittaisen kaupasta saatavan kupari-tms putken sisään.
    - tälle voi sitten esim laserleikkurilla tehdä riittävän tarkan kauluksen josta voi kiinnittää mihin haluaa.
- Pitääkö kela kiertää siten että joka toinen kierros menee eri puolelta akselia vai riittääkö puolet ja puolet (yksinkertaistaisi kelaamista merkittävästi)
  - mikäli on pakko kiertää vuorotellen niin sitten käytännössä taitaa olla pakko laittaa itse akseli jne paikalleen heti alkuun, kts http://www.mikesflightdeck.com/images/aircore2B.jpg
    - Ellei kehitä jotain mystistä tapaa kiertää sitä reikää josta putki jossa akseli jne on.
- Kuinka kiertää molemmat kelat yhtä-aikaa (lienee kannattaa luntata ompelukoneiden designejä, niissä on aika monimutkaista mekaniikkaa)
  - Tai toroidikäämien tekolaitteita: http://www.youtube.com/watch?v=6lvm3FGTHSI
  - Käämintäkone voisi noudatella tehdyn LEGO-mallin periaatteita, tulostetaan helical bevel gears tällä mallilla http://www.thingiverse.com/thing:3575
  - Näistä mallia http://www.simcoaftermarket.com/specialty-oem/micro-air-core/ näissä nähdäkseni on kelattu osa käämistä toiselle puolelle akselia ja osa toiselle

Concept Art:

Apollo Flight Director Attitude Indicator (FDAI) / "8-ball"

- http://history.nasa.gov/ap16fj/01popup_fdai.htm
- http://www.space1.com/Artifacts/Apollo_Artifacts/FDAI/fdai.html
- http://images2.bonhams.com/erez4/cache/Images_live_2010-02_22_8032945-77-2_jpg_tif_440d0d4ea2a91a5f.jpg
- Book: *Apollo Training - Guidance And Control Systems - Block II (1967-09-15)*
    - http://dl.dropbox.com/u/39575647/Temp/Apollo%20Training%20-%20Guidance%20And%20Control%20Systems%20-%20Block%20II%20%281967-09-15%29%20%5Bfav%5D.pdf
- Book: *The Apollo Guidance Computer - Architecture and Operation (2010)*
    - kysy suovulalta

Soyuz "Globus" IMP navigation instrument

- http://upload.wikimedia.org/wikipedia/commons/c/c7/Vostokpanel.JPG
- http://upload.wikimedia.org/wikipedia/commons/1/18/Voskhod_spacecraft_IMP_%27Globus%27_navigation_instrument%2C_front_view.jpg
- http://upload.wikimedia.org/wikipedia/commons/3/3d/Voskhod_spacecraft_IMP_%27Globus%27_navigation_instrument%2C_inside_view.jpg
- http://i240.photobucket.com/albums/ff132/francoisguay/GlobusFG/GlobusFG%20Publ/GlobusFGDocuphoto4123cSpub9226v02.jpg

// Aristan Sparkfunin 9DOF Stick - http://www.sparkfun.com/products/10724
// Arduino 1.0 ympäristölle

```
#include <Wire.h>


// Kiihtyvyys
int16_t acc_ddx = 0;
int16_t acc_ddy = 0;
int16_t acc_ddz = 0;
const uint8_t ACC = 0x53;          // ADSL345 3-Axis Accelerometer


// Kulmanopeus
int gyro_dx = 142;
int gyro_dy = 143;
int gyro_dz = 144;
const uint8_t GYRO = 0x68;         // ITG-3200 3-Axis MEMS Gyro


// Magneettikenttä
int mag_x = 242;
int mag_y = 243;
int mag_z = 243;
const uint8_t MAG = 0x1e;          // HMC5883L 3-Axis Magnetometer
```

```
// Kun joku menee pieleen, vilkutellaan lediä ja jäädään tähän
void HALT() {
  while (true) {
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(500);
  }
}


// Kirjoitetaan yksi tavu I2C-laitteelle
void writeTo(uint8_t device, uint8_t address, uint8_t value) {
  Wire.beginTransmission(device);
  Wire.write(address);
  Wire.write(value);
  Wire.endTransmission();
}


// Luetaan yksi tavu I2C-laitteelta
uint8_t readFrom(uint8_t device, uint8_t address) {
  uint8_t retval[1];
  readFrom(device, address, retval, 1);
  return *retval;
}


// Luetaan N-tavua I2C-laitteelta
void readFrom(uint8_t device, uint8_t address, uint8_t * buffer, uint8_t length) {
  Wire.beginTransmission(device);
  Wire.write(address);              // sends address to read from
  Wire.endTransmission();         // end transmission

  Wire.beginTransmission(device);
  Wire.requestFrom(device, length);    // request length bytes from device

  uint8_t i = 0;
  while(Wire.available() && i < length) {        // device may send less than requested (abnormal)
    buffer[i++] = Wire.read();    // receive a byte
  }

  if(i != length) HALT();

  Wire.endTransmission();         // end transmission
}


// Alustetaan kiihtyvyysanturi
void initAccelerometer() {
  // Turn on the ADXL345
  writeTo(ACC, 0x2d, 0x00);          // ADXL345_POWER_CTL = 0
  writeTo(ACC, 0x2d, 0x16);          // ADXL345_POWER_CTL = 16
```

```
  writeTo(ACC, 0x2d, 0x08);          // ADXL345_POWER_CTL = 8

  writeTo(ACC, 0x31, 0x08);          // ADXL345_DATA_FORMAT = 8
}


// Luetaan kiihtyvyysanturi
void readAccelerometer() {
  uint8_t  buffer[6];

  readFrom(ACC, 0x32, buffer, 6);        // ADXL345_DATAX0 = 0x32, ... ADXL345_DATAZ1 = 0x37, = 6 bytes

  // Each axis reading comes in 10 bit resolution, ie 2 bytes.  Least Significat Byte first!!
  // thus we are converting both bytes in to one int
  acc_ddx = ((((int)buffer[1]) << 8) | buffer[0]);
  acc_ddy = ((((int)buffer[3]) << 8) | buffer[2]);
  acc_ddz = ((((int)buffer[5]) << 8) | buffer[4]);
}


// Alustetaan gyroskooppi
void initGyro() {

  const uint8_t sampleRateDivider = 0;         // NOSRDIVIDER     = default    FsampleHz=SampleRateHz/(divider+1)
  const uint8_t scaleRange = 3;                // RANGE2000       = default
  const uint8_t filterAndSampleRate = 0;       // BW256_SR8       = default    256Khz BW and 8Khz SR
  const uint8_t clockSrc = 1;                  // PLL_XGYRO_REF   ?
  bool itgReady = true;
  bool rawReady = true;

  // Set sample rate
  writeTo(GYRO, 0x15, sampleRateDivider);                                    // SMPLRT_DIV = 0x15 RW   SETUP: Sample Rate Divider

  // Set Full scale range
  writeTo(GYRO, 0x16, (readFrom(GYRO, 0x16) & ~B00011000) | (scaleRange << 3));         // DLPF_FS = 0x16, DLPFFS_FS_SEL = 0x18

  // Set Filter and Sample rate
  writeTo(GYRO, 0x16, (readFrom(GYRO, 0x16) & ~B00000111) | filterAndSampleRate);       // DLPF_FS = 0x16, DLPFFS_DLPF_CFG = 0x07

  // Set Clock source
  writeTo(GYRO, 0x3e, (readFrom(GYRO, 0x3e) & ~B00000111) | clockSrc);                  // PWR_MGM = 0x3e, PWRMGM_CLK_SEL = 0x07

  // Set ITG Ready
  writeTo(GYRO, 0x17, (readFrom(GYRO, 0x17) & ~B00000100) | (itgReady << 2));          // INT_CFG = 0x17, INTCFG_ITG_RDY_EN = 0x04

  // Set Raw Ready
```

```
  writeTo(GYRO, 0x17, (readFrom(GYRO, 0x17) & ~B00000001) | rawReady);                // INT_CFG =
0x17, INTCFG_RAW_RDY_EN = 0x01

  // Startup delay
  delay(50);    // 50ms from gyro startup
  delay(20);    // 20ms register r/w startup

}


// Luetaan kiihtyvyysanturi
void readGyro() {
  uint8_t  buffer[6];

  readFrom(GYRO, 0x1d, buffer, 6);        // GYRO_XOUT0 = 0x1d, .. GYRO_ZOUT1 = 0x22, = 6 bytes

  gyro_dx = ((((int)buffer[0]) << 8) | buffer[1]);
  gyro_dy = ((((int)buffer[2]) << 8) | buffer[3]);
  gyro_dz = ((((int)buffer[4]) << 8) | buffer[5]);
}


// Alustetaan magnetometri
void initMagnetometer() {
  uint8_t   id[3];
  readFrom(MAG, 0x10, id, 3);               // IdentificationRegisterA = 0x10,  .. IdentificationRegisterC = 0x12,
= 3 bytes
//  if (id[0] != 'H' || id[1] != '4' || id[2] != '3') HALT();

  writeTo(MAG, 0x00, B00011100);           // ConfigurationRegisterA = 0x00, Samples per measureent = 1,
Data output rate = 75 Hz, No bias

  uint8_t reg;
  float gauss;
  float scale;

  reg = 0x00, gauss = 0.88, scale = 0.73;
  // reg = 0x01, gauss = 1.30, scale = 0.92;
  // reg = 0x02, gauss = 1.90, scale = 1.22;
  // reg = 0x03, gauss = 2.50, scale = 1.52;
  // reg = 0x04, gauss = 4.00, scale = 2.27;
  // reg = 0x05, gauss = 4.70, scale = 2.56;
  // reg = 0x06, gauss = 5.60, scale = 3.03;
  //reg = 0x07, gauss = 8.10, scale = 4.35;
  writeTo(MAG, 0x01, reg << 5);           // ConfigurationRegisterB = 0x01, setting is in the top 3 bits of the
register

  //writeTo(MAG, 0x02, B00000001);           // ModeRegister = 0x02, Single-measurement mode
  writeTo(MAG, 0x02, B00000000);           // ModeRegister = 0x02, Continuous-measurement mode

}
```

```
// Luetaan magnetometri
void readMagnetometer() {

  if (readFrom(MAG, 0x09) & B000001 != 1) {    // StatusRegister = 0x09
    mag_x = 0;
    mag_y = 0;
    mag_z = 0;
    // HALT();
    return;
  }

  uint8_t  buffer[6];

  readFrom(MAG, 0x03, buffer, 6);        // DataRegisterBegin = 0x03, = 6 bytes

  mag_x = ((((int)buffer[0]) << 8) | buffer[1]);
  mag_y = ((((int)buffer[2]) << 8) | buffer[3]);
  mag_z = ((((int)buffer[4]) << 8) | buffer[5]);
}


void setup() {
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW); // Ledi 13 pois päältä

  Wire.begin();              // Join I2C bus
  Serial.begin(115200);

  initAccelerometer();
  initGyro();
  initMagnetometer();

  // Odotetaan joku merkki ennen kuin aletaan lähettämään
  while (Serial.available() == 0) {
       // odotetaan
  }

}

void writeInt(int value) {
    Serial.write(value & 0xff);
    Serial.write((value >> 8) & 0xff);
}


void loop() {

  Serial.print("[");
  Serial.print(micros());
  Serial.print("]");

  readAccelerometer();
  readGyro();
```

```
  readMagnetometer();
/*
  Serial.print(" acc(");
  Serial.print(acc_ddx);
  Serial.print(",");
  Serial.print(acc_ddy);
  Serial.print(",");
  Serial.print(acc_ddz);
  Serial.print(") gyro(");
  Serial.print(gyro_dx);
  Serial.print(",");
  Serial.print(gyro_dy);
  Serial.print(",");
  Serial.print(gyro_dz);
  Serial.print(") mag(");
  Serial.print(mag_x);
  Serial.print(",");
  Serial.print(mag_y);
  Serial.print(",");
  Serial.print(mag_z);
  Serial.print(")");
  Serial.println();
  */
  writeInt(-1);
  writeInt(acc_ddx);
  writeInt(acc_ddy);
  writeInt(acc_ddy);


  delay(50);

}



//Wycliffe 0503517554 La 31.3. wraduma@gmail.com


 //Arduino

 // Aristan Sparkfunin 9DOF Stick - http://www.sparkfun.com/products/10724
// Arduino 1.0 ympäristölle


#include <Wire.h>


// Kiihtyvyys
int16_t acc_ddx = 0;
int16_t acc_ddy = 0;
int16_t acc_ddz = 0;
const uint8_t ACC = 0x53;          // ADSL345 3-Axis Accelerometer
```

```
// Kulmanopeus
int gyro_dx = 142;
int gyro_dy = 143;
int gyro_dz = 144;
const uint8_t GYRO = 0x68;        // ITG-3200 3-Axis MEMS Gyro


// Magneettikenttä
int mag_x = 242;
int mag_y = 243;
int mag_z = 243;
const uint8_t MAG = 0x1e;         // HMC5883L 3-Axis Magnetometer


// Kun joku menee pieleen, vilkutellaan lediä ja jäädään tähän
void HALT() {
  while (true) {
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(500);
  }
}


// Kirjoitetaan yksi tavu I2C-laitteelle
void writeTo(uint8_t device, uint8_t address, uint8_t value) {
  Wire.beginTransmission(device);
  Wire.write(address);
  Wire.write(value);
  Wire.endTransmission();
}


// Luetaan yksi tavu I2C-laitteelta
uint8_t readFrom(uint8_t device, uint8_t address) {
  uint8_t retval[1];
  readFrom(device, address, retval, 1);
  return *retval;
}


// Luetaan N-tavua I2C-laitteelta
void readFrom(uint8_t device, uint8_t address, uint8_t * buffer, uint8_t length) {
  Wire.beginTransmission(device);
  Wire.write(address);            // sends address to read from
  Wire.endTransmission();         // end transmission

  Wire.beginTransmission(device);
  Wire.requestFrom(device, length);    // request length bytes from device

  uint8_t i = 0;
```

```
  while(Wire.available() && i < length) {        // device may send less than requested (abnormal)
    buffer[i++] = Wire.read();    // receive a byte
  }

  if(i != length) HALT();

  Wire.endTransmission();         // end transmission
}


// Alustetaan kiihtyvyysanturi
void initAccelerometer() {
  // Turn on the ADXL345
  writeTo(ACC, 0x2d, 0x00);          // ADXL345_POWER_CTL = 0
  writeTo(ACC, 0x2d, 0x16);          // ADXL345_POWER_CTL = 16
  writeTo(ACC, 0x2d, 0x08);          // ADXL345_POWER_CTL = 8

  writeTo(ACC, 0x31, 0x08);          // ADXL345_DATA_FORMAT = 8
}


// Luetaan kiihtyvyysanturi
void readAccelerometer() {
  uint8_t  buffer[6];

  readFrom(ACC, 0x32, buffer, 6);        // ADXL345_DATAX0 = 0x32, ... ADXL345_DATAZ1 = 0x37, = 6
bytes

  // Each axis reading comes in 10 bit resolution, ie 2 bytes.  Least Significat Byte first!!
  // thus we are converting both bytes in to one int
  acc_ddx = ((((int)buffer[1]) << 8) | buffer[0]);
  acc_ddy = ((((int)buffer[3]) << 8) | buffer[2]);
  acc_ddz = ((((int)buffer[5]) << 8) | buffer[4]);
}


// Alustetaan gyroskooppi
void initGyro() {

  const uint8_t sampleRateDivider = 0;         // NOSRDIVIDER     = default    FsampleHz=SampleRateHz/
(divider+1)
  const uint8_t scaleRange = 3;                // RANGE2000       = default
  const uint8_t filterAndSampleRate = 0;       // BW256_SR8       = default    256Khz BW and 8Khz SR
  const uint8_t clockSrc = 1;                  // PLL_XGYRO_REF   ?
  bool itgReady = true;
  bool rawReady = true;

  // Set sample rate
  writeTo(GYRO, 0x15, sampleRateDivider);                                    // SMPLRT_DIV = 0x15
RW   SETUP: Sample Rate Divider

  // Set Full scale range
  writeTo(GYRO, 0x16, (readFrom(GYRO, 0x16) & ~B00011000) | (scaleRange << 3));        // DLPF_FS =
```

```
0x16, DLPFFS_FS_SEL = 0x18

  // Set Filter and Sample rate
  writeTo(GYRO, 0x16, (readFrom(GYRO, 0x16) & ~B00000111) | filterAndSampleRate);       // DLPF_FS
= 0x16, DLPFFS_DLPF_CFG = 0x07

  // Set Clock source
  writeTo(GYRO, 0x3e, (readFrom(GYRO, 0x3e) & ~B00000111) | clockSrc);                // PWR_MGM =
0x3e, PWRMGM_CLK_SEL = 0x07

  // Set ITG Ready
  writeTo(GYRO, 0x17, (readFrom(GYRO, 0x17) & ~B00000100) | (itgReady << 2));          // INT_CFG =
0x17, INTCFG_ITG_RDY_EN = 0x04

  // Set Raw Ready
  writeTo(GYRO, 0x17, (readFrom(GYRO, 0x17) & ~B00000001) | rawReady);                // INT_CFG =
0x17, INTCFG_RAW_RDY_EN = 0x01

  // Startup delay
  delay(50);    // 50ms from gyro startup
  delay(20);    // 20ms register r/w startup

}


// Luetaan kiihtyvyysanturi
void readGyro() {
  uint8_t  buffer[6];

  readFrom(GYRO, 0x1d, buffer, 6);        // GYRO_XOUT0 = 0x1d, .. GYRO_ZOUT1 = 0x22, = 6 bytes

  gyro_dx = ((((int)buffer[0]) << 8) | buffer[1]);
  gyro_dy = ((((int)buffer[2]) << 8) | buffer[3]);
  gyro_dz = ((((int)buffer[4]) << 8) | buffer[5]);
}


// Alustetaan magnetometri
void initMagnetometer() {
  uint8_t   id[3];
  readFrom(MAG, 0x10, id, 3);               // IdentificationRegisterA = 0x10,  .. IdentificationRegisterC = 0x12,
= 3 bytes
//  if (id[0] != 'H' || id[1] != '4' || id[2] != '3') HALT();

  writeTo(MAG, 0x00, B00011100);           // ConfigurationRegisterA = 0x00, Samples per measureent = 1,
Data output rate = 75 Hz, No bias

  uint8_t reg;
  float gauss;
  float scale;

  reg = 0x00, gauss = 0.88, scale = 0.73;
  // reg = 0x01, gauss = 1.30, scale = 0.92;
```

```
  // reg = 0x02, gauss = 1.90, scale = 1.22;
  // reg = 0x03, gauss = 2.50, scale = 1.52;
  // reg = 0x04, gauss = 4.00, scale = 2.27;
  // reg = 0x05, gauss = 4.70, scale = 2.56;
  // reg = 0x06, gauss = 5.60, scale = 3.03;
  //reg = 0x07, gauss = 8.10, scale = 4.35;
  writeTo(MAG, 0x01, reg << 5);            // ConfigurationRegisterB = 0x01, setting is in the top 3 bits of the
register

  //writeTo(MAG, 0x02, B00000001);           // ModeRegister = 0x02, Single-measurement mode
  writeTo(MAG, 0x02, B00000000);           // ModeRegister = 0x02, Continuous-measurement mode

}


// Luetaan magnetometri
void readMagnetometer() {

  if (readFrom(MAG, 0x09) & B000001 != 1) {    // StatusRegister = 0x09
    mag_x = 0;
    mag_y = 0;
    mag_z = 0;
    // HALT();
    return;
  }

  uint8_t  buffer[6];

  readFrom(MAG, 0x03, buffer, 6);        // DataRegisterBegin = 0x03, = 6 bytes

  mag_x = ((((int)buffer[0]) << 8) | buffer[1]);
  mag_y = ((((int)buffer[2]) << 8) | buffer[3]);
  mag_z = ((((int)buffer[4]) << 8) | buffer[5]);
}


void setup() {
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW); // Ledi 13 pois päältä

  Wire.begin();              // Join I2C bus
  Serial.begin(115200);

  initAccelerometer();
  initGyro();
  initMagnetometer();

  // Odotetaan joku merkki ennen kuin aletaan lähettämään


  //while (Serial.available() == 0) {
  // odotetaan
  //}
```

```
}


void writeInt(int value) {
   Serial.write(value & 0xff);
   Serial.write((value >> 8) & 0xff);
}


void loop() {
/*
  Serial.print("[");
  Serial.print(micros());
  Serial.print("]");
*/
  readAccelerometer();
  readGyro();
  readMagnetometer();
/*
  Serial.print(" acc(");
  Serial.print(acc_ddx);
  Serial.print(",");
  Serial.print(acc_ddy);
  Serial.print(",");
  Serial.print(acc_ddz);
  Serial.print(") gyro(");
  Serial.print(gyro_dx);
  Serial.print(",");
  Serial.print(gyro_dy);
  Serial.print(",");
  Serial.print(gyro_dz);
  Serial.print(") mag(");
  Serial.print(mag_x);
  Serial.print(",");
  Serial.print(mag_y);
  Serial.print(",");
  Serial.print(mag_z);
  Serial.print(")");
  Serial.println();
  */
  writeInt(-1);
  writeInt(acc_ddx);
  writeInt(acc_ddy);
  writeInt(acc_ddy);


  delay(50);

}
```

```
// Processing

/**
 * Rotate 1.
 *
 * Rotating simultaneously in the X and Y axis.
 * Transformation functions such as rotate() are additive.
 * Successively calling rotate(1.0) and rotate(2.0)
 * is equivalent to calling rotate(3.0).
 */

 // Kiihtyvyys
float acc_ddx = 0;
float acc_ddy = 0;
float acc_ddz = -1;



// Kulmanopeus
float gyro_dx = 0;
float gyro_dy = 0;
float gyro_dz = 0;



// Magneettikenttä
float mag_x = 0;
float mag_y = 0;
float mag_z = -1;


float a = 0.0;
float rSize;  // rectangle size

//Read Serial
import processing.serial.*;
Serial myPort;  // The serial port


void setup() {
  size(640, 360, P3D);
  rSize = width / 6;
  noStroke();
  fill(204, 204);
    // List all the available serial ports
  println(Serial.list());
  // I know that the first port in the serial list on my mac
  // is always my  Keyspan adaptor, so I open Serial.list()[0].
  // Open whatever port is the one you're using.
  myPort = new Serial(this, Serial.list()[5], 115200);
  //delay(2000);
```

```
  myPort.write ('9');
}

short readShort() {
  short val = 0;
  int foo = myPort.read() & 0xff;
  foo += (myPort.read() & 0xff) << 8;
  val = (short)foo;
 // short val = (short)myPort.read() & 0xff;
 // val += (short)((myPort.read() & 0xff) << 8);
  return val;
}

void draw() {
  if(myPort.available()>=8) {
     int val = readShort();
     if (val == -1) {
        acc_ddx = readShort();
        acc_ddy = readShort();
        acc_ddz = readShort();
     } else {
       print("erro val = ");
       print(val);
     }
  }
  background(0);
  a += 0.005;
  if(a > TWO_PI) {
    a = 0.0;
  }

  translate(width/2, height/2);

  rotateX(a);
  //
  rotateY(a * 2.0);
  rect(-rSize, -rSize, rSize*2, rSize*2);

  rotateX(a * acc_ddx/100);
  rotateY(a * 2.002);
  rect(-rSize, -rSize, rSize*2, rSize*2);

  while (myPort.available() > 0) {
    int inByte = myPort.read();
    println(inByte);
  }

}

//Wycliffe 0503517554 La 31.3. wraduma@gmail.com
```