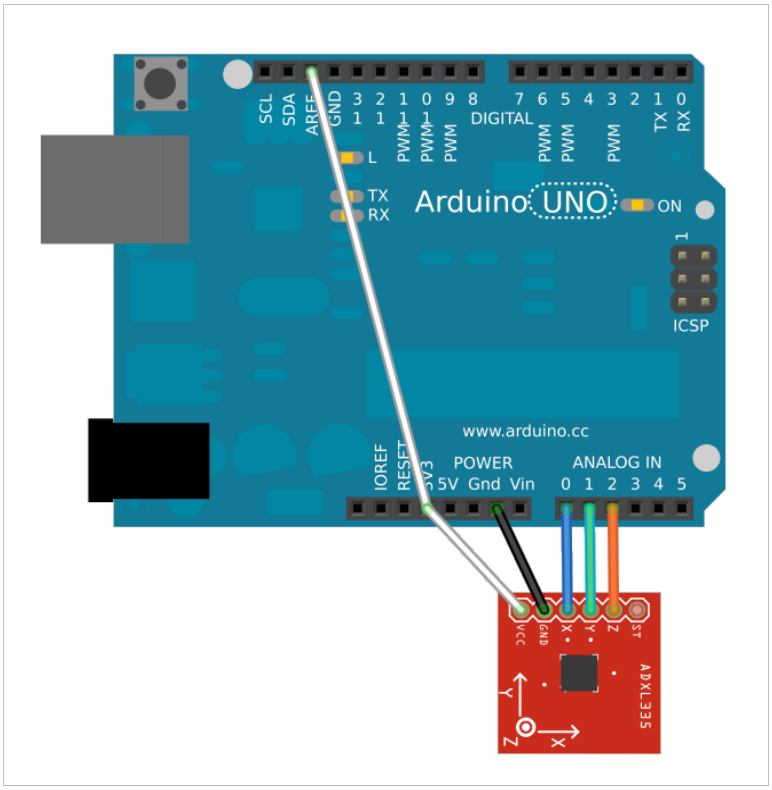
Mouse cu accelerometru si Arduino Leonardo

Acest proiect se bazeaza pe capacitatea placii Arduino Leonardo de a se comporta ca mouse. Prin combinarea unui Arduino Leonardo cu un accelerometru pe 3 axe, vom obtine un mouse pe care il vom controla prin miscarea mainii. Poti folosi orice fel de accelerometrul doresti. In cele ce urmeaza vom folosi un accelerometru ADXL335, pentru simplitate. Accelerometrul se conecteaza la Arduino in modul obisnuit, ca in schema mai jos.

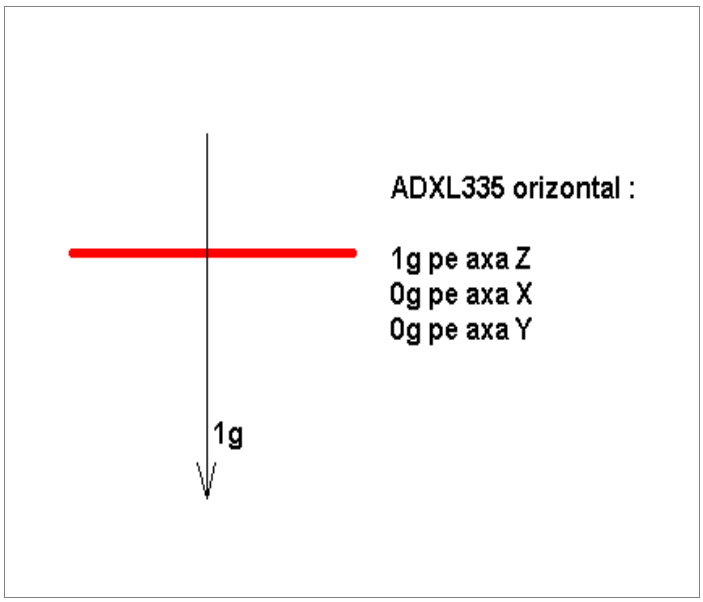
```
#define SENSIBILITATE_X 3
#define SENSIBILITATE_Y 3

void setup() {
 Serial.begin(9600);
 analogReference(EXTERNAL);
 Mouse.begin();

}

void loop() {
  float xAcc=readAcc(0);
  float yAcc=readAcc(1);
  float zAcc=readAcc(2);

  Serial.print("Acceleratie X: ");
  Serial.print(xAcc,DEC);
  Serial.print("Acceleratie Y: ");
  Serial.print(yAcc,DEC);
  Serial.println();
  delay(50);

  Mouse.move(xAcc * SENSIBILITATE_X, yAcc * SENSIBILITATE_Y, 0);

}

float readAcc(int port) {
 int value=analogRead(port);
 int miliVolts=map(value,0,1023,0,3300)-3300/2;
 float acc=(float)miliVolts/360;
 return acc;
}
```

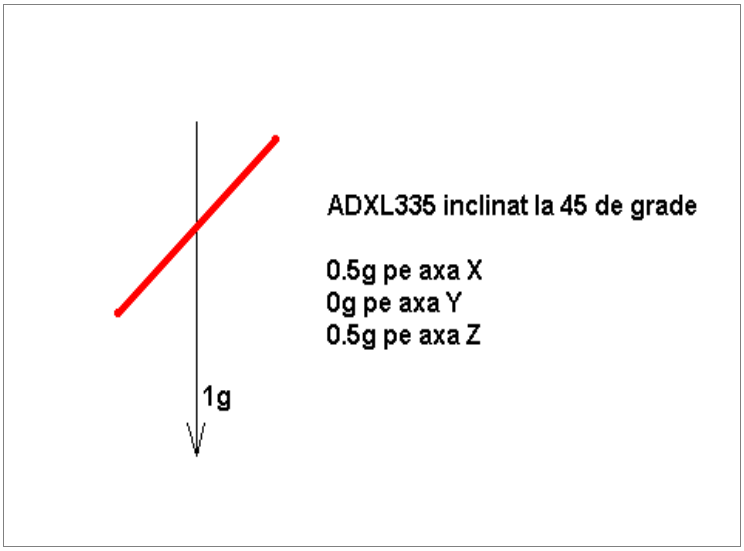


Arduino 3.3 V	ADXL335 VCC
Arduino GND	ADXL335 GND
Arduino Analog0	ADXL335 X
Arduino Analog1	ADXL335 Y
Arduino Analog2	ADXL335 Z
Arduino 3.3	Arduino AREF

Constantele SENSIBILITATE_X si SENSIBILITATE_Y definesc cat de mult se misca mouse-ul pe ecran atunci cand miscam accelerometrul. Cu cat valorile pentru aceste constante sunt mai mari, cu atat mouse-ul se va misca mai mult pe ecran atunci cand miscam accelerometrul. Esti liber sa te joci cu aceste constante pana cand miscarea pe ecran este exact asa cum vrei tu.

Principiul de functionare al acestui mouse se bazeaza pe faptul ca in orice locatie de pe Pamant avem o acceleratie gravitionala 1g, indreptata intotdeauna pe directia verticala. Cand accelerometrul ADXL335 este orientat pe directie orizontala, atunci pe axa X si pe axa Y acceleratiile citite sunt zero. Este o acceleratie de 1g citita pe axa Z. Atunci cand inclinam accelerometrul in plan, acceleratia de 1g care se manifesta doar pe axa Z cand accelerometrul este orizontal, incepe sa se manifeste si pe axele X si Y. Astfel daca inclinam accelerometrul la un unghi de 45 de grade pe axa X, vom avea o acceleratie de 0.5g pe axa X.


ADXL335 orizontal :
1g pe axa Z
0g pe axa X
0g pe axa Y
1g


ADXL335 inclinat la 45 de grade
0.5g pe axa X
0g pe axa Y
0.5g pe axa Z
1g

Acceleratiile citite de accelerometru sunt inmultite cu o valoare constanta (in cazul nostru cu 3, valoarea constantei SENSIBILITATE_X) si mouse-ul este miscat pe ecran corespunzator valorii obtinute. Poate vei spune ca o deplasare de 3 pixeli atunci cand accelerometrul este inclinat maximum este putin, dar adu-ti aminte ca functia *loop* se executa non-stop, astfel incat mouse-ul este miscat cu 3 pixeli la fiecare 50 de milisecunde (pentru ca in *loop* exista o instructiune *delay(50)*).

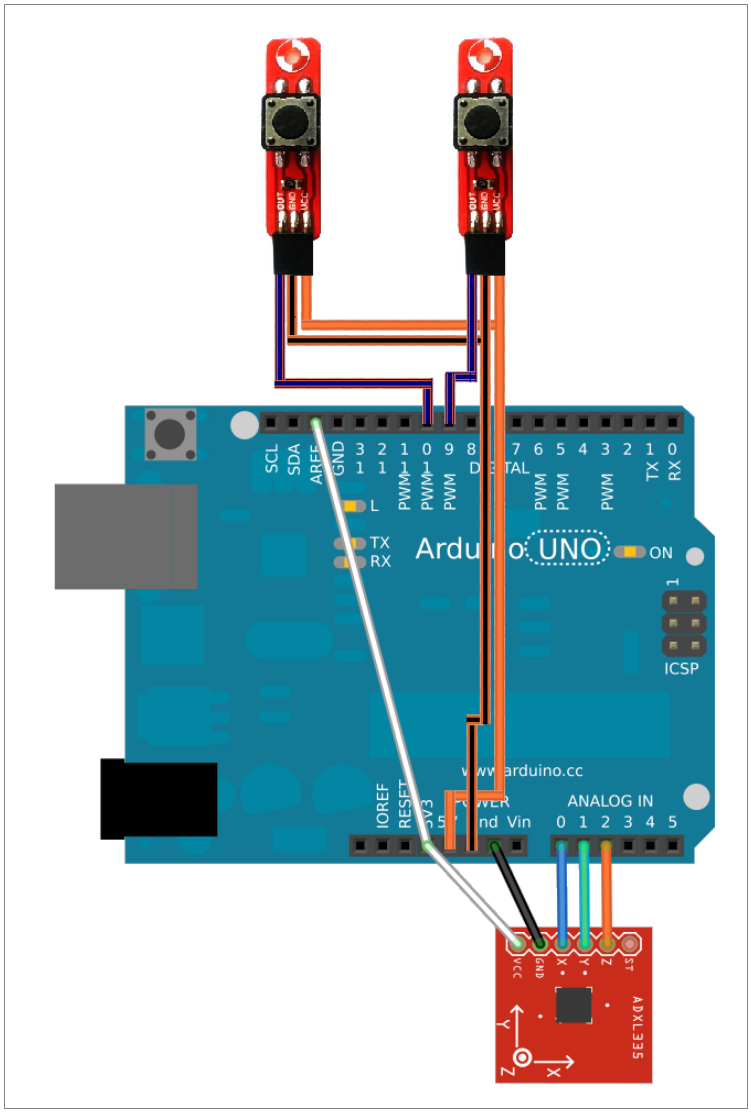
Daca ti se pare complicat sa tii mouse-ul fix pe ecran (pentru ca este destul de dificil sa tii ADXL335 perfect orizontal), atunci putem imbunatati programul de mai sus prin setarea unei zone in care mouse-ul nu se misca deloc pe ecran, ca mai jos (modificarile in **bold**) :

```
#define SENSIBILITATE_X 3
#define SENSIBILITATE_Y 3
#define ZONA_MOARTA_X 0.3
#define ZONA_MOARTA_Y 0.3

void setup() {
 Serial.begin(9600);
 analogReference(EXTERNAL);
 Mouse.begin();
}

void loop() {
  float xAcc=readAcc(0);
  float yAcc=readAcc(1);
  float zAcc=readAcc(2);

  Serial.print("Acceleratie X: ");
  Serial.print(xAcc,DEC);
  Serial.print("Acceleratie Y: ");
  Serial.print(yAcc,DEC);
  Serial.println();
  delay(50);

  int miscareX = 0;
  if (abs(xAcc) > ZONA_MOARTA_X) {
      miscareX = xAcc * SENSIBILITATE_X;
  }

  int miscareY = 0;
  if (abs(yAcc) > ZONA_MOARTA_Y) {
      miscareY = yAcc * SENSIBILITATE_Y;
  }

  Mouse.move(miscareX, miscareY, 0);

}

float readAcc(int port) {
 int value=analogRead(port);
 int miliVolts=map(value,0,1023,0,3300)-3300/2;
 float acc=(float)miliVolts/360;
 return acc;
}
```

Am introdus doua constante, ZONA_MOARTA_X si ZONA_MOARTA_Y, care definesc valorile acceleratiei sub care mouse-ul nu face nici o miscare. Astfel, daca tii accelerometrul intr-o pozitie aproape de orizontala, mouse-ul nu se misca deloc pe ecran.

In final, ca sa poti misca simplu accelerometrul, iti sugerez sa il prinzi pe o manusa de piele (cumparata de la standul cu elemente de protectie din Hornbach, BricoStore sau Practiker).

Urmatoarea extindere a acestui proiect este sa adaugi sistemului si doua butoane brick, obtinand astfel un mouse perfect functional. Schema de conectare este cea de mai jos. Butoanele le poti monta pe cate un deget al fiecarei manusi, astfel incat sa le poti apasa cu degetul mare (prin suprapunerea degetului mare peste unul dintre cele doua degete pe care sunt lipite butoanele).




```
#define SENSIBILITATE_X 3
#define SENSIBILITATE_Y 3
#define ZONA_MOARTA_X 0.3
#define ZONA_MOARTA_Y 0.3

#define BUTON1_PIN 9
#define BUTON1_PIN 10
#define DEBOUNCE_TIME 100

void setup() {
 Serial.begin(9600);
 analogReference(EXTERNAL);
 pinMode(BUTON1_PIN, INPUT);
 pinMode(BUTON2_PIN, INPUT);
 Mouse.begin();
}

long lastButtonPress1 = 0;
```

```
long lastButtonPress2 = 0;

void loop() {

  int buton1 = digitalRead(BUTON1_PIN);
  int buton2 = digitalRead(BUTON2_PIN);
  if (buton1 == 1) {
      if ((millis() - lastButtonPress1) > DEBOUNCE_TIME) {
            lastButtonPress1 = millis();
            Mouse.click(BUTTON_LEFT);
      }
  }
 if (buton2 == 1) {
      if ((millis() - lastButtonPress2) > DEBOUNCE_TIME) {
            lastButtonPress2 = millis();
            Mouse.click(BUTTON_RIGHT);
      }
  }

  float xAcc=readAcc(0);
  float yAcc=readAcc(1);
  float zAcc=readAcc(2);

  Serial.print("Acceleratie X: ");
  Serial.print(xAcc,DEC);
  Serial.print("Acceleratie Y: ");
  Serial.print(yAcc,DEC);
  Serial.println();
  delay(50);

  int miscareX = 0;
  if (abs(xAcc) > ZONA_MOARTA_X) {
      miscareX = xAcc * SENSIBILITATE_X;
  }

  int miscareY = 0;
  if (abs(yAcc) > ZONA_MOARTA_Y) {
      miscareY = yAcc * SENSIBILITATE_Y;
  }

  Mouse.move(miscareX, miscareY, 0);

}

float readAcc(int port) {
 int value=analogRead(port);
 int miliVolts=map(value,0,1023,0,3300)-3300/2;
 float acc=(float)miliVolts/360;
 return acc;
}
```

De remarcat in codul de mai sus este constanta DEBOUCE_TIME. Asa cum am povestit mai pe larg in cadrul sectiunii despre butoane brick, atunci cand se apasa un buton, semnalul receptionat de catre Arduino variaza de cateva ori intre 0 si 1 (nu trece instantaneu din 0 in 1). Acest lucru se intampla datorita faptului ca atunci cand apasam butonul, lamelele metalice care inchid contactul nu se ating perfect. Ca sa evitam ca Arduino sa citeasca mai multe apasari de buton, vreme de 100 de milisecunde

dupa prima apasare receptionata (valoarea lui DEBOUNCE_TIME) vom ignora orice alta apasare. Acest lucru inca permite conceptul de dublu click (doar ca diferenta intre doua click-uri succesive va fi exact DEBOUNCE_TIME milisecunde). Daca dublu click nu este sesizat corect de calculatorul tau, poti sa micsorezi valoarea acestei constante, sau sa modifici pe calculator durata intre cele doua click-uri succesive pentru a obtine un dublu click.