```
#include <Wire.h>

#include <I2Cdev.h>

//Cogemos las librerias necesarias para ejecutar el LED infrarrojo
#include <boarddefs.h>
#include <ir_Lego_PF_BitStreamEncoder.h>
#include <IRremote.h>
#include <IRremoteInt.h>

#include <I2Cdev.h>

// I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files
// for both classes must be in the include path of your project
#include "I2Cdev.h"
#include "MPU6050.h"

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE
implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for InvenSense evaluation board)
// AD0 high = 0x69
MPU6050 accelgyro;
//MPU6050 accelgyro(0x69); // <-- use for AD0 high
// delaramos las variables normales y las viejas
int16_t ax, ay, az;
int16_t gx, gy, gz;
int16_t ax_old, ay_old, az_old;
int16_t gx_old, gy_old, gz_old;
 const unsigned int S_pwr[68] = {4600, 4350, 700, 1550, 650, 1550, 650, 1600,
650, 450, 650, 450, 650, 450, 650, 450, 700, 400, 700, 1550, 650, 1550, 650,
1600, 650, 450, 650, 450, 650, 450, 700, 450, 650, 450, 650, 450, 650, 1550, 700,
450, 650, 450, 650, 450, 650, 450, 650, 450, 700, 400, 650, 1600, 650, 450, 650,
1550, 650, 1600, 650, 1550, 650, 1550, 700, 1550, 650, 1550, 650};
const unsigned int S_mute[68] = {4650, 4350, 650, 1550, 650, 1550, 700, 1550,
700, 400, 700, 400, 700, 400, 700, 450, 650, 450, 650, 1550, 700, 1500, 700,
1550, 700, 400, 700, 450, 650, 400, 700, 450, 700, 400, 700, 1500, 700, 1550,
650, 1550, 700, 1500, 700, 450, 700, 400, 700, 400, 700, 400, 700, 400, 700, 450,
650, 450, 700, 400, 700, 1500, 700, 1550, 650, 1550, 700, 1500, 700};
const unsigned int S_pup[68] = {4600, 4350, 700, 1500, 700, 1500, 700, 1550,
700, 450, 650, 400, 700, 450, 650, 450, 700, 400, 700, 1500, 700, 1550, 650,
1550, 700, 450, 650, 450, 700, 400, 700, 400, 700, 400, 700, 400, 700, 1550, 700,
400, 700, 400, 700, 1550, 650, 450, 700, 400, 700, 400, 700, 1550, 650, 450, 650,
1600, 650, 1550, 650, 450, 700, 1500, 700, 1500, 700, 1550, 650};
const unsigned int S_pdown[68] = {4650, 4300, 700, 1550, 700, 1500, 700, 1550,
700, 400, 700, 400, 700, 400, 700, 450, 650, 450, 650, 1550, 700, 1500, 700,
1550, 700, 400, 700, 400, 700, 400, 700, 450, 700, 400, 700, 400, 700, 400, 700,
450, 650, 450, 650, 1550, 700, 400, 700, 450, 650, 400, 700, 1550, 700, 1500,
700, 1550, 700, 1500, 700, 400, 700, 1550, 650, 1550, 700, 1500, 700};
const unsigned int S_vup[68] = {4600, 4350, 650, 1550, 700, 1500, 700, 1550,
700, 400, 700, 400, 700, 450, 650, 450, 700, 400, 700, 1500, 700, 1550, 650,
1550, 700, 400, 700, 400, 700, 450, 650, 450, 700, 400, 700, 1500, 700, 1550,
650, 1550, 700, 400, 700, 450, 700, 400, 700, 400, 700, 400, 700, 450, 650, 450,
650, 450, 650, 1550, 700, 1500, 700, 1550, 700, 1500, 700, 1550, 650};
const unsigned int S_vdown[68] = {4600, 4350, 700, 1550, 650, 1550, 700, 1500,
700, 450, 650, 450, 700, 400, 700, 400, 700, 400, 700, 1550, 700, 1500, 700,
1550, 700, 400, 700, 400, 700, 400, 700, 450, 650, 450, 650, 1550, 700, 1500,
700, 450, 650, 1550, 700, 400, 700, 400, 700, 450, 700, 400, 700, 400, 700, 400,
700, 1550, 700, 400, 700, 1500, 700, 1500, 700, 1550, 700, 1500, 700};
const unsigned int S_guide[68] = {4600, 4350, 700, 1500, 700, 1550, 700, 1500,
700, 450, 650, 450, 700, 400, 700, 400, 700, 400, 700, 1550, 650, 1550, 700,
1500, 700, 450, 650, 450, 700, 400, 700, 400, 700, 400, 700, 1550, 700, 1500,
700, 1550, 650, 1550, 700, 400, 700, 400, 700, 1550, 700, 400, 700, 400, 700,
400, 700, 450, 700, 400, 650, 1550, 700, 1550, 650, 450, 700, 1500, 700};
const unsigned int S_exit[68] = {4650, 4300, 700, 1550, 650, 1550, 700, 1550,
700, 400, 700, 400, 700, 450, 650, 450, 650, 450, 650, 1550, 700, 1500, 700,
1550, 700, 450, 650, 400, 700, 450, 650, 450, 700, 400, 700, 1500, 700, 400, 700,
1550, 700, 1500, 700, 400, 700, 1550, 700, 450, 650, 400, 700, 450, 650, 1550,
700, 400, 700, 400, 700, 1550, 650, 450, 650, 1550, 700, 1500, 700};
const unsigned int S_tv[68] = {4600, 4350, 650, 1550, 700, 1500, 700, 1550, 700,
400, 700, 400, 700, 400, 700, 450, 700, 400, 700, 1500, 700, 1500, 700, 1550,
700, 400, 700, 400, 700, 450, 650, 450, 700, 400, 700, 1500, 700, 1550, 700, 400,
700, 400, 700, 400, 700, 400, 700, 1550, 700, 400, 700, 400, 700, 400, 700, 1550,
700, 1500, 700, 1550, 650, 1550, 700, 400, 700, 1500, 700};

#define LED_PIN 13

void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif

  // initialize serial communication
  // (38400 chosen because it works as well at 8MHz as it does at 16MHz, but
  // it's really up to you depending on your project)
  Serial.begin(38400);

  // initialize device
  Serial.println("Initializing I2C devices...");
  accelgyro.initialize();

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

  // use the code below to change accel/gyro offset values
  /*
    Serial.println("Updating internal sensor offsets...");
    // -76          -2359  1688    0          0          0
    Serial.print(accelgyro.getXAccelOffset()); Serial.print("\t"); // -76
    Serial.print(accelgyro.getYAccelOffset()); Serial.print("\t"); // -2359
    Serial.print(accelgyro.getZAccelOffset()); Serial.print("\t"); // 1688
    Serial.print(accelgyro.getXGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getYGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getZGyroOffset()); Serial.print("\t"); // 0
    Serial.print("\n");
    accelgyro.setXGyroOffset(220);
    accelgyro.setYGyroOffset(76);
    accelgyro.setZGyroOffset(-85);
    Serial.print(accelgyro.getXAccelOffset()); Serial.print("\t"); // -76
    Serial.print(accelgyro.getYAccelOffset()); Serial.print("\t"); // -2359
    Serial.print(accelgyro.getZAccelOffset()); Serial.print("\t"); // 1688
    Serial.print(accelgyro.getXGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getYGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getZGyroOffset()); Serial.print("\t"); // 0
    Serial.print("\n");
   */

  // configure Arduino LED for
  pinMode(LED_PIN, OUTPUT);
  Serial.println("Setup completed");

}

void loop() {

  // read raw accel/gyro measurements from device
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  // these methods (and a few others) are also available
  //accelgyro.getAcceleration(&ax, &ay, &az);
  //accelgyro.getRotation(&gx, &gy, &gz);


  // display tab-separated accel/gyro x/y/z values
  /* Serial.print("a/g:\t");
    delay (250);
    Serial.print(ax); Serial.print("\t");
    delay (250);
    Serial.print(ay); Serial.print("\t");
    delay (250);
    Serial.print(az); Serial.print("\t");
    delay (250);
    Serial.print(gx); Serial.print("\t");
    delay (250);
    Serial.print(gy); Serial.print("\t");d
    delay (250);
    Serial.println(gz);*/
  if (gx > gx_old + 10000) {
    Serial.println ("mute,");
    digitalWrite(LED_PIN, S_mute[68]);
  }
  if (gx < gx_old - 10000) {
    Serial.println ("mute,");
    digitalWrite(LED_PIN, S_mute[68]);
  }

  if (ax > ax_old + 10000) {
    Serial.println ("power,");
    //función para que el led envie los impulsos al receptor
    digitalWrite(LED_PIN, S_pwr[68]);
  }
  if (ax < ax_old - 10000) {
    Serial.println ("power,");
    digitalWrite(LED_PIN, S_pwr[68]);
  }
  if (ay > ay_old + 10000) {
    Serial.println ("hdmi,");
    digitalWrite(LED_PIN, S_tv[68]);
  }
  if (ay < ay_old - 10000) {
    Serial.println ("hdmi,");
    digitalWrite(LED_PIN, S_tv[68]);
  }
  if (az > az_old + 10000) {
    Serial.println ("exit,");
    digitalWrite(LED_PIN, S_exit[68]);
  }
  if (az < az_old - 10000) {
    Serial.println ("guide,");
    digitalWrite(LED_PIN, S_guide[68]);
  }
  if (gy > gy_old + 10000) {
    Serial.println ("canalup,");
    digitalWrite(LED_PIN, S_pup[68]);
  }
  if (gy < gy_old - 10000) {
    Serial.println ("canaldown,");
    digitalWrite(LED_PIN, S_pdown[68]);
  }
  if (gz > gz_old + 10000) {
    Serial.println ("subir volumen,");
    digitalWrite(LED_PIN, S_vup[68]);
  }
  if (gz < gz_old - 1000) {
    Serial.println ("bajar volumen,");
    digitalWrite(LED_PIN, S_vdown[68]);
  }
  // reseteo de variables, de nuevas a viejas
  ax_old = ax;
  ay_old = ay;
  az_old = az;
  gx_old = gx;
  gy_old = gy;
  gz_old = gz;



#ifdef OUTPUT_BINARY_ACCELGYRO
  Serial.write((uint8_t)(ax >> 8)); Serial.write((uint8_t)(ax & 0xFF));
  Serial.write((uint8_t)(ay >> 8)); Serial.write((uint8_t)(ay & 0xFF));
  Serial.write((uint8_t)(az >> 8)); Serial.write((uint8_t)(az & 0xFF));
  Serial.write((uint8_t)(gx >> 8)); Serial.write((uint8_t)(gx & 0xFF));
  Serial.write((uint8_t)(gy >> 8)); Serial.write((uint8_t)(gy & 0xFF));
  Serial.write((uint8_t)(fzgz >> 8)); Serial.write((uint8_t)(gz & 0xFF));
#endif


}
```