Project name: Kinesis : An intuitive mouse interface

Author: Apostol Andrei Constantin

Purpose: Computer peripherals have mainly consisted of mouse, keyboard and joysticks. However, with VR and AR swiftly emerging on the market, it is paramount to explore new designs that can offer a more immersive experience and allow us to fully benefit from these technologies. Currently, most VR headsets do not include a controller, and those that do are either prohibitively expensive or offer an unidimensional experience.
The Kinesis Project proposes an affordable and intuitive mouse interface, which is designed as a wearable glove that can emulate all mouse functions (clicking, moving, hold, scroll etc.) using only the user's hand gestures. Since some variation is involved, such as hand size, finger size and how the user wears the device, the glove is designed to calibrate and adapt to each specific user.

Requirements:

- 1x Arduino Pro Micro (or any ATMEGA32U4 compatible board)
- 1x MPU-6050 Gyroscope & Accelerometer Module
- 1x Button
- 1x Breadboard 60p
- 3x 2.2 Inch Flex Sensor
- 1x standard glove

Optional:

- Protoboard/PCB

Cost: Aprox. 50 USD. (eBay)

We have only identified one similar product : the Nintendo PowerGlove, which is a controller designed for the Nintendo NES platform, which is a retired system. Price range for a pre-owned PowerGlove is 100+$.
As of now, no similar product exists on the market.

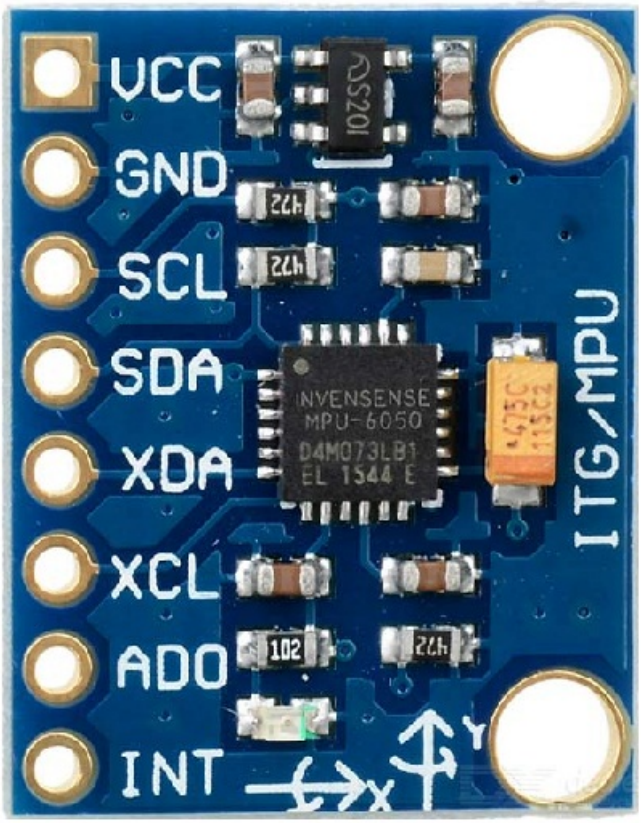
Project description:
The glove contains flex sensors sewn to the index, middle and ring finger. As the sensor is flexed, the resistance across the sensor increases, which helps us detect user gestures. Values can be read by constructing a simple voltage divider. Index finger is used for left-click, middle-finger for right click and the ring finger is used for scrolling.
The user should equip the Kinesis Glove and connect it to a PC. After connection, the RX and TX leds will blink, signaling the calibration phase is about to begin. During this period, the hand should be held in a comfortable position. Once the leds stop blinking, the board reads the initial hand position and the calibration phase begins. During this phase (lasts aprox. 2 seconds) the user should clench and release his fist, as to determine the

flexing range. The program then computes thresholds for left and right click, relative to the flexing range.

Let min_i denote the minimum value read from sensor i,
max_i denote the maximum value read from sensor i,
where $i \in 1,2$
then, the formula used to calculate the threshold is
$min_i + s * (max_i - min_i)$
where $s \in [0,1]$ is a variable sensitivity factor.

Low values of s determine low sensitivity, which means the user should make very ample gestures in order for actions to be triggered. Similarly, high values of s determine high sensitivity, meaning the user is only required to make "tap"-like gestures for clicks to be effected.
Currently, s is set to 0.7 for both. This value has been determined experimentally and has been chosen as a compromise between comfort and control. Feel free to tweak this value to your liking.

The scroll sensor (mounted on ring finger) is activated only relative to the initial position determined when calibration phase begins. Current threshold settings are:
$init - 40$ for scrolling down
$init + 15$ for scrolling up,
where $init$ is the initial position.



Mouse movement is done via the values read from the MPU-6050 module. The MPU-6050 sensor contains a MEMS accelerometer and a MEMS gyro in a single chip. It is very accurate, as it contains 16-bit ADC hardware for each channel. Therefore, it's able to capture the x,y and z channels at the same time. The sensor uses the I2C-bus to communicate with the Arduino. In order to read raw values,we first have to disable sleep mode, and then the registers for the accelerometer and gyro can be read.

At each step, the mouse is moved relative to its current position. The formula used to determine the mouse movement is the following:

$$dist_x = -(g_x + 300)/200$$

$$dist_y = (g_z - 100)/200$$

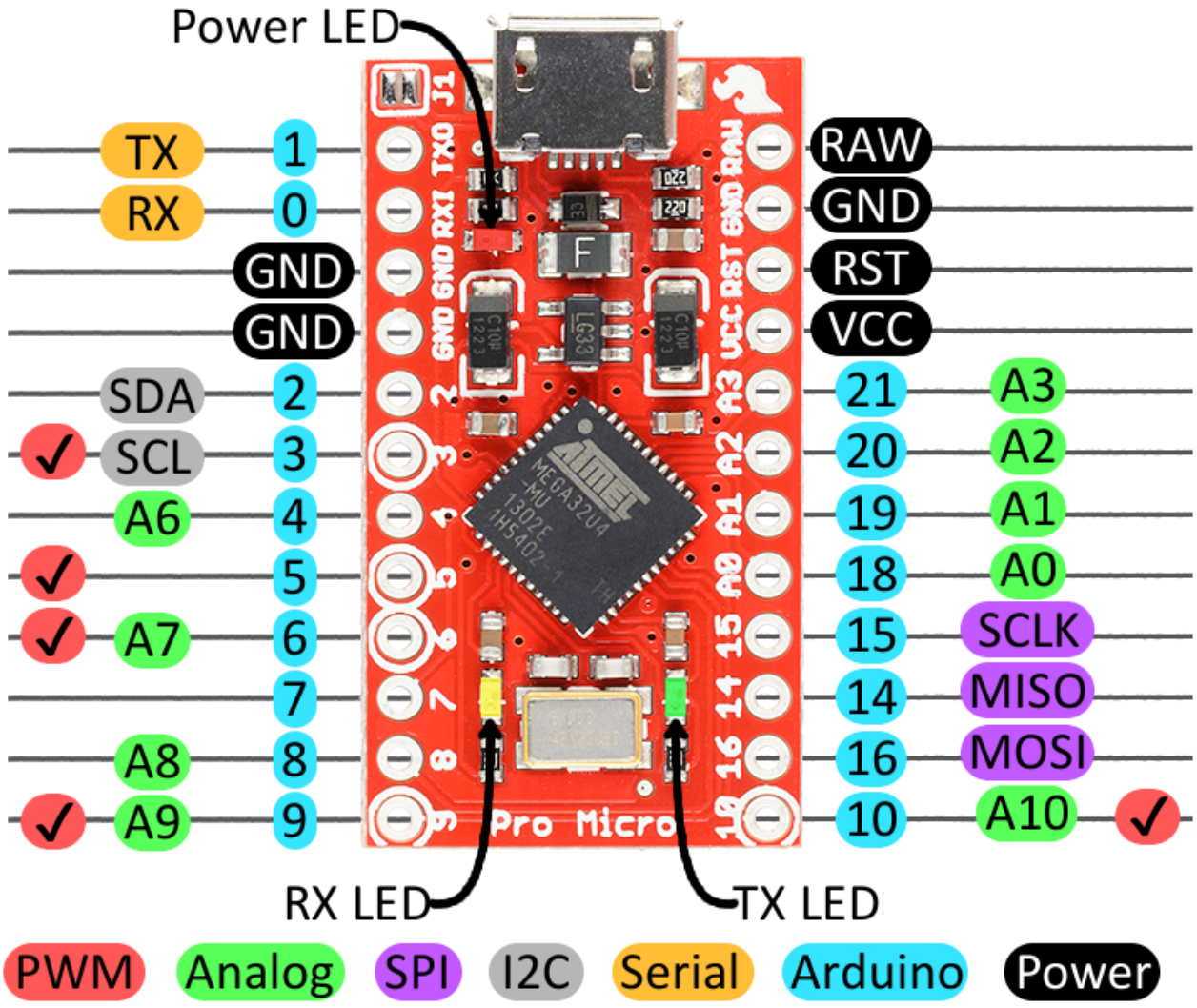
where $dist_x$ and $dist_y$ are the distance the mouse moves on the X and Y axis respectively, and g_x and g_z values read from the gyroscope's X and Z axis.

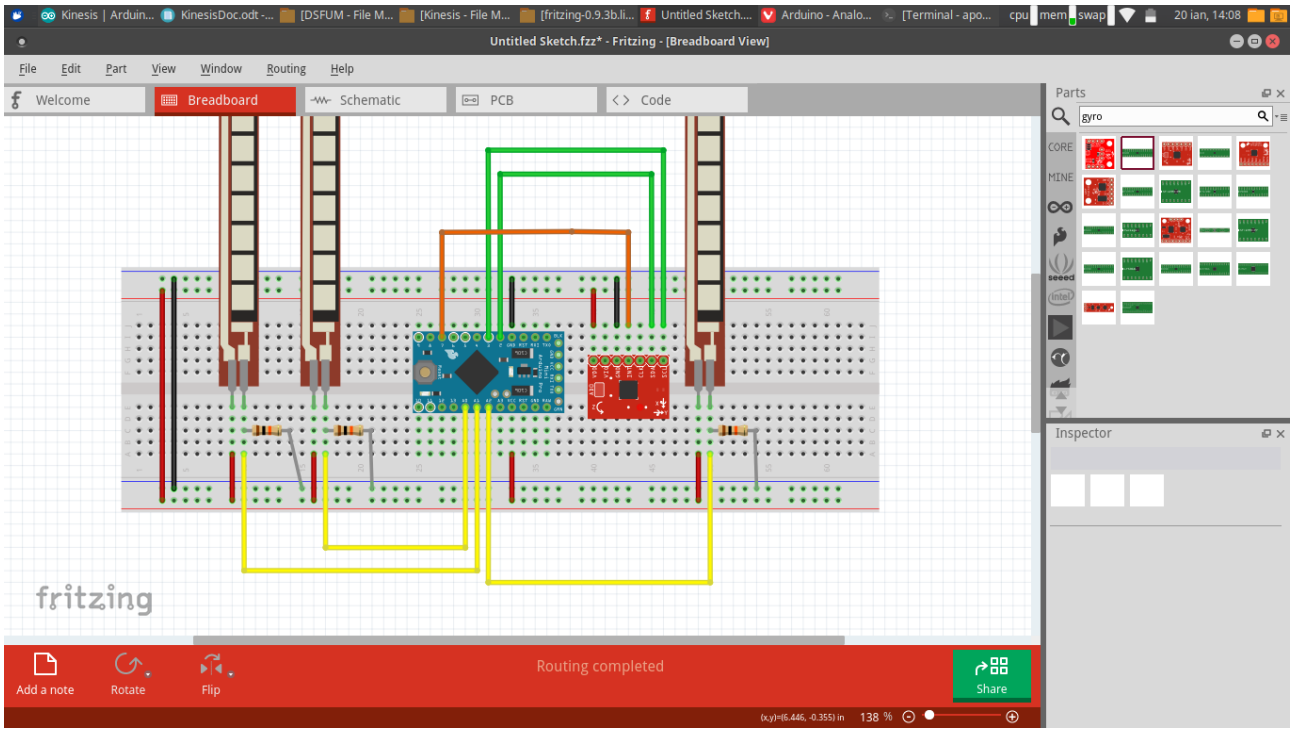
Note that due to the module's orientation on the breadboard, there is no 1:1 correspondence between screen and gyroscope axis. The added corrections are done to map the values correctly. To increase or decrease mouse sensitivity, simply tweak the value by which it is divided (currently 200).

If calibration has not been done properly or the user wants to remove the glove without losing control of the mouse, the user can press the Reset button. Resetting will make the ProMicro enter Bootloader Mode, meaning it may take up to 8 seconds for it to start up.

Connections: To establish I2C communication, we need to connect the SCL and SDA pins of the Arduino Pro Micro and the MPU-6050. Flex sensor connections are just as easy.
In order to read values from the flex sensors, we have used voltage dividers.

To enable I2C communication between the MPU-6050 and the Pro Micro, we have to wire together the SCL and SDA pins. ProMicro's SDA and SCL pins are 2 and 3, respectively. Also, the MPU's INT pin has to be connected to an interrupt-enabled pin on the ProMicro. For this project, we have chosen pin 7.
Caution! If you are using a different 32u4 board, please consult its schematic as the Serial communication pins may differ.





Arduino ProMicro

VCC → +5V
GND → Common GND
Add a button between GND and RST pins.

MPU6050

VCC → +5V
GND → Common GND
SDA → ProMicro 2
SCL → ProMicro 3
INT → ProMicro 7

Flex sensor → Arduino

VCC → +5V
GND → Common GND
OUT → ProMicro AnalogPin (left→A0, mid→A1, right→A2)

Source code: You cand find the entire commented source code as well as the Fritzing project and the documentation in this repo: https://github.com/AndreiXYZ/ProiectArduino

Resource analysis:
The memory footprint is fairly lightweight: 8kb of program storage space and only 435 bytes of dynamic memory.
As for speed, during testing it has proven to be very responsive and doesn't have any noticeable delay between gestures and mouse action.

Improvements & future considerations: Although presenting good potential, the Kinesis Glove is still in its early stages of development and could use further improvements.
Here are some future considerations for this project:

-Wireless communication (IR/BLE)
-Potentiometer to fine-tune sensitivity on the fly
-Add encasing