Pivot: A Motion Based User Interface

6.111 Final Project Proposal

Andrew Kurtz and Nestor Franco

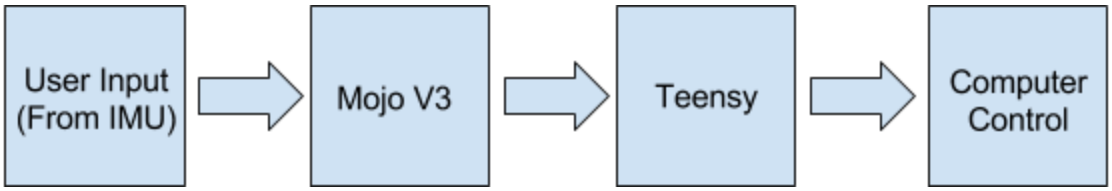
1 Overview (NF)

With the advent of online video streaming sites such as YouTube and Netflix, traditional cable and satellite TV subscriptions are declining. As such, more and more people are connecting a computer to a TV screen not just to stream video, but also to play games and browse websites. It can be inconvenient to use a full sized wireless keyboard and mouse while sitting on the couch - a better alternative would be to have some kind of compact, handheld controller that contains both keyboard key input and mouse control.

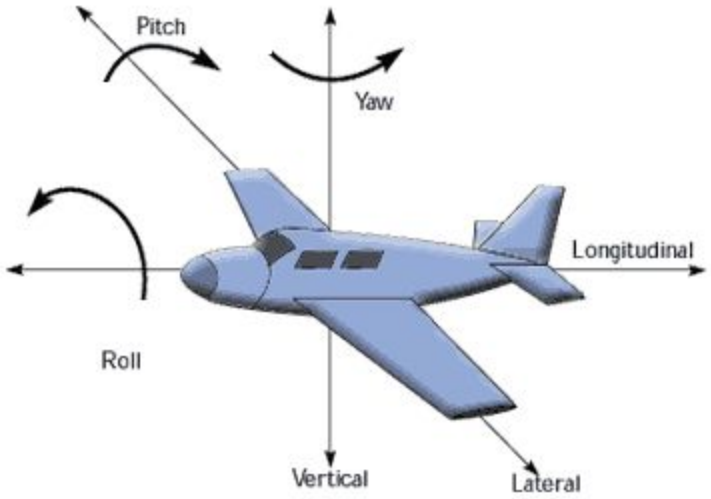
Our final project will focus on developing the mouse control portion of this controller using a combination of angular rate, and acceleration sensors. We will be using an IMU to collect these measurements, and these signals would be directly fed into a small FPGA board. The FPGA will run these signals through filters and algorithms, which will give us a more accurate measurement of the orientation and movement of the controller, and then the FPGA will translate that into mouse movement and function (e.g. mouse clicks and scrolling). The mouse signals would then be communicated to a computer through USB. The basic functionality will consist of this, and further improvements could include making it wireless and allowing user customization (e.g. adjustment of mouse sensitivity levels, or setting user-defined commands).

2 Design

2.1 High Level Diagram (AK)



2.2 Definition of User Input (AK)



User inputs can fall into 4 categories: translations, rotations, taps, and gestures.

- Translations are defined as planar motion in either the lateral/longitudinal plane or the lateral/vertical plane.
- Rotations are defined as either pitch, roll, or yaw motions.
- Taps are defined by an event of large acceleration (as recorded by the IMU) happening over a short time period (on the order of 10ms).
- Gestures are defined as a planar motion which traces out a distinct path. For example if the user were to quickly trace a circle with the controller this would be considered a gesture.

Our device will, at minimum be able to recognize translations, rotations, and taps. We hope to add gestures but are unsure if this will be possible during the time allotted.

2.3 Hardware Details (AK)

Upon completion, the hardware for this project will be consolidated into a handheld, 3D printed enclosure approximately 3 inches by 4 inches in size. The enclosure will allow access for a USB cord as well as, at minimum, a physical reset button. The enclosure will also be made as ergonomic as possible to increase the user experience.

2.4 Stretch Goals (AK)

On top of our basic goal of having our device serve the same functions as a standard computer mouse we hope to add functionality by implementing some type of gesture learning and recognition.

Essentially we want the user to be able to record a motion and map that motion to a keyboard command, for example a counterclockwise circle could be mapped to the page back command and a clockwise circle could be mapped to the page forward command.

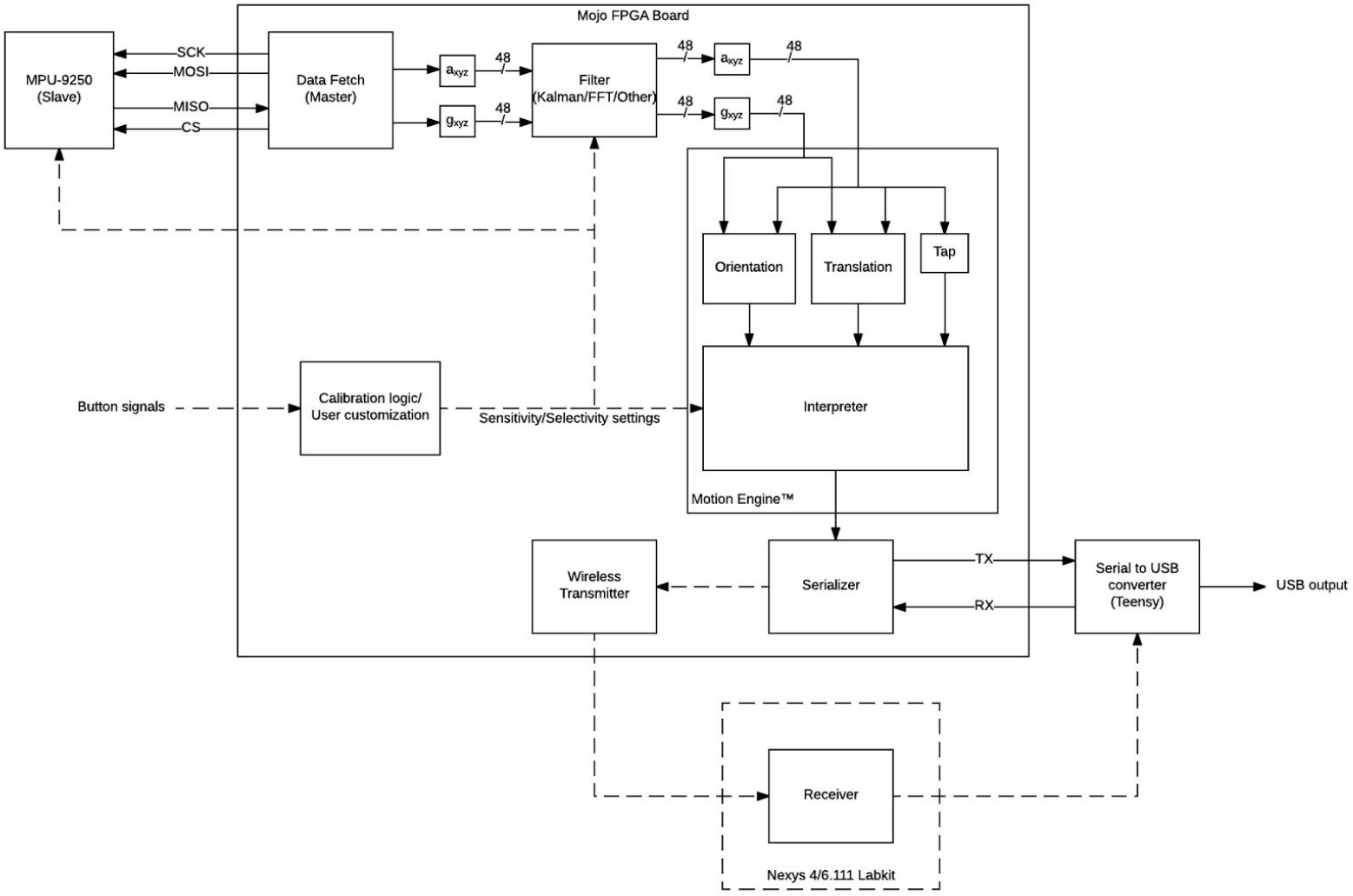
The basic functionality for gesture learning would consist of hard coding different possibilities, i.e. page forward (stored as "option 0"), page back ("option 1"), page up ("option 2"), page down ("option 3"), etc. The user would use a bank of dip switches to select an option, then while pressing a "learn" button perform the desired motion.

Time pending we would hope to have a system by which a full keyboard and mouse could be hooked up to our device and a user would be able to record any combination of keystrokes or mouse inputs and map these to a gesture.

Other stretch goals include the desire to add wireless functionality via a serial to bluetooth chip (purchasable as an all in one solution from Sparkfun for $25 each) as well as physical calibration (to set the zero reference for the coordinate system) and sensitivity adjustment buttons.

3 Implementation

3.1 Block Diagram (NF)



3.2 SPI Interpreter (NF)

Decodes the data from the SPI bus of the IMU and supplies it to the filter module.

3.3 Filter (NF)

This module will take in the noisy acceleration, and angle measurements and output a more accurate estimation of controller movement. This may be achieved through several methods like Kalman filtering, and signal averaging. We expect this module to be the most time consuming to create.

3.4 Motion Engine™ (NF + AK)

Our Motion Engine™ will translate acceleration, angle, and potentially magnetic measurements into mouse movement using predefined gains, and will also detect certain gestures or conditions that would trigger a mouse command, like right click.

Use of the magnetometer will be evaluated upon initial tests but our focus will be trying to use the accelerometer and gyroscope for position feedback. Without the magnetometer (and a large, local, magnetic field) it will be impossible to obtain absolute position data for the controller. We believe relative position, angle, and acceleration will be more than enough for accurate control of a computer cursor.

Gesture recognition (one of our stretch goals) may require more accurate absolute positioning, however, so pending our progress we will have to reevaluate this.

Tap sensing can be done with a simple state machine looking at accelerations above a certain magnitude (the magnitude of acceleration of a tap will be much larger than any motion based acceleration). This can happen before sensor fusion and filtering.

To create an easy to use device it is important that the device be able to recognize the difference between different user inputs and isolating motion, for example, when a user taps the device the cursor on the computer should not move. The same should be true for gestures.

3.5 Serializer (NF)

The Serializer will then take the desired mouse behavior and output it using a serial standard.

3.6 Serial to USB (NF + AK)

This module will not be created using the FPGA, but instead we will be using a separate microcontroller (a Teensy 3.2 to be exact). The microcontroller will be programmed using the arduino environment to convert serial commands received from the FPGA to follow the Human Interface Device protocol (HID) and transmit those commands over the onboard USB port. The HID protocol is what most keyboards and mice use to interface with computers. Luckily the Teensy comes with premade libraries and example code for using the HID protocol, as such implementation of this module will be extremely straight forward.

3.7 Description of IMU hardware (AK)

We will be using the MPU-9250 integrated IMU from Invensense. This is a dual die chip containing an accelerometer, gyroscope and magnetometer. This chip outputs raw sensor data via either an I2C or SPI bus. We have chosen this chip because it is one of the most widely used in the market (lots of user data available). We are only planning on using the gyroscope and accelerometer at this time but figured having the magnetometer available was a nice feature.

4 Challenges/Risk (AK)

Our project is risky for a number of reasons:

- We plan to use the Mojo dev board, which the course staff does not have experience with.
- We are attempting to apply rather sophisticated filtering methods to the IMU data which could prove to be a time sync.
- Murphy's law: any complex project comes with problems which are difficult to identify till you are deeper into the weeds.

We are willing to accept the risk associated with using the Mojo dev board since there is a plethora of documentation and tutorials online for using it.

Luckily, if after some work we find it too difficult to apply adequate filtering to our raw sensor data we have the option to de-scope by buying an off the shelf integrated IMU. These devices automatically take raw accelerometer, and gyroscope data, filter and combine them to output an estimated state over a serial interface. If we decide to take this path we would focus our efforts on developing the gesture recognition software described above.

5 Schedule (AK + NF)

Week 1 (10/30 - 11/05): Research communication protocols and filtering methods, define precise functionality desired by controller.

Week 2 (11/06 - 11/12): Design and simulation of the SPI Interpreter (Nestor). Teensy HID development and design of serializer module (Andrew).

Week 3 (11/13 - 11/19): Design and simulation of the filter module (Andrew). Design and simulation of the Motion Engine™ (Nestor and Andrew).

Week 4 (11/20 - 11/26): Continue work on Motion Engine™ and integration (Nestor and Andrew).

Week 5 (11/27 - 12/03): Combine all the parts and hope for a working system. Design and 3D print sexy ergonomic housing (Andrew).

Week 6 (12/04 - 12/10): Bug fixing and additional functionality implementation.

6 Testing (NF)

6.1 SPI Interpreter

We can test this module by having another FPGA board output a known SPI signal, feed it into our Mojo board, and then view the decoded data on a logic analyzer.

The SPI protocol is susceptible to clock skew if the wires used are too long. We will be closely monitoring our SPI lines to ensure these are not going to be the source of errors down the line.

6.2 Filter

This filter can be tested after the SPI Interpreter is completed by generating signals with the IMU, passing it through the interpreter, and then passing that into the filter. We then can compare the pre and post filtered data using a logic analyzer.

6.3 Motion Engine™

To test the Motion Engine™, we can use another FPGA board to send gyroscopic and acceleration values into the Mojo board, and see if the Mojo outputs numbers for delta_x, delta_y, right_click, left_click, etc.

To make our lives easier at the start we may decide to use an arduino to output and display raw sensor data. We will use this data to explore the possibility of using the magnetometer in conjunction with a large rare earth magnet for fine positioning.

6.4 Serializer

Finally, we can test the serializer by giving it a few values for delta_x, delta_y, etc., and see if the serial waveform it produces matches what we would expect.

Validation of this section should be straightforward.

7 Resources (AK)

Our project requires us to have the MPU-9250 IMU chip, the Mojo V3 FPGA development board, a small prototyping board for the Mojo V3, and a Teensy 3.2 development board.

The Mojo, prototyping board, Teensy, and IMU will be loaned to us by our mentor Joe Steinmeyer.

In addition to the above, our team will be utilizing a 3D printer in the Aero Astro department to create an ergonomic enclosure for our device.

Should we choose to pursue the bluetooth communication stretch goal we have identified modules sold by Sparkfun electronics for $25 each. The lead time on these is on the order of days.

If we go wireless, we will want to add a battery to our project. This can most easily be accomplished by using a small cell phone external battery (thanks to the career fair, these are free and plentiful around MIT).

8 Conclusion (AK)

The ways in which humans interact with computers has not changed much in the last 20+ years. This project seeks to implement a motion based user input device based on an FPGA which can be recognized by a computer as a standard keyboard/mouse interface. We think this device will provide a more intuitive, and easier to use interface to decrease the barrier between humans and computers.

We believe that the use of an FPGA to handle sensor filtering, position estimation, and command processing will allow us to decrease response lag and increase input accuracy beyond what can normally be obtained with a microcontroller.