UPPSALA
UNIVERSITET

PROJECT CURSOR GLOVE REPORT

By Asif Mohamed, Vishnu Ullas & Paul Sujeet

CONTENTS

1. ABSTRACT

Our main goal is to build a wearable-device with feasible and compact hardware modules that can be used to control a mouse pointer on any PC. To achieve this goal, we need to implement few other goals. To control the movement of the mouse pointer using a glove, we need to visualize a 3D orientation from a single 6 or 9 DOF IMU (Inertial Measurement Unit). To produce the most accurate and reliable orientation estimations, we need to research and design the best filter for our application and overall, we need to create a better user-environment interaction.

2. MOTIVATION

During our master's degree, we had implemented two projects that involve automation. The first project was called "Sweet Shot" and it comprised of a small automated cannon that would sense if a person has entered a room and calculate his/her height via an Ultrasonic sensor placed on the frame of the door. Then the cannon would automatically calibrate itself to aim for the individual's waist and shoot candy at them. The cannon was made from recyclable material and only had one DOF (just moves up and down) as we implemented only using an accelerometer.

The next project we made was called "May the candy be with you". It was an extension of the previous project and our first look into using a wireless glove like device for automation purposes. In this project we gave the cannon 2 DOF (both up & down and left & right) and the movement of the cannon and the firing was controlled by the glove the user wears.

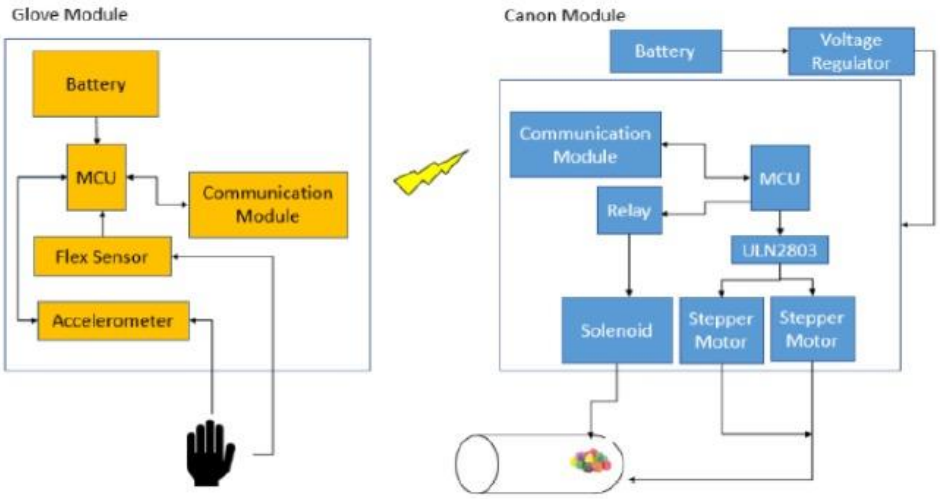

Figure 1. Block Schematics of 'May the Candy be with you'.

After the implementation of this project, we decided to focus more on the application of the glove and how we can make it more automated and use it for other applications. Hence, the Cursor Glove. While researching further into this idea, we came across various projects and related papers that provided us with further understanding to implement this project.

Two students from Cornell University (1) implemented a wireless computer pointing device with glove-based controls and user preference select ability. The idea is to

control a cursor through different hand orientations and finger presses using just an accelerometer. Users can operate their computers with their hands in midair without the hassle of desks surfaces or wires. The results were promising but there was a lot of noise that was not taken into consideration. Clearly, just using one inertial sensor to track hand motion wasn't enough. Thus, we started researching on orientation estimation based on IMUs.

A paper (2) from 2017 was referred to us by our supervisor, Frederik, and it focused on the signal processing aspects of position and orientation estimation using inertial sensors. It also discusses about different modeling choices and algorithms that include optimized-based smoothing and filtering as well as less computational extended Kalman filter and complementary filter implementations. Studying this journal paper gave us quite a vast number of choices of estimation algorithms to jump into.

In 2009 Sebastian Madgwick developed an IMU and AHRS sensor fusion algorithm (3) as part of his Ph.D. research at the University of Bristol. This paper (4) gave a deep overview on the Madgwick filter, which is more of an optimization technique to fuse sensor data and gave us a better understanding on its implementation.

In the next chapter, we explain the different features of inertial sensors and algorithms that use these features for estimation orientation.

3. INERTIAL SENSORS

The term inertial sensor is used to denote the combination of a three-axis accelerometer and a three axis gyroscope. Devices containing these sensors are usually referred to as inertial measurement units (IMUs). A gyroscope measures the sensor's angular velocity, i.e. the rate of change of the sensor's orientation. An accelerometer measures the external specific force acting on the sensor. The specific force consists of both the sensor's acceleration and the earth's gravity. Nowadays, many gyroscopes and accelerometers are based on microelectromechanical system (MEMS) technology. MEMS components are small, light, inexpensive, have low power consumption, short start-up times and their accuracy has significantly increased over the years.

Inertial sensors can be used to provide information about the pose of any object that they are rigidly attached to. It is also possible to combine multiple inertial sensors (e.g. magnetometer) to obtain information about the pose of separate connected objects. Hence, inertial sensors can be used to track human motion as well.

There exists a large amount of literature (5) (4) (6) on the use of inertial sensors for position and orientation estimation. The reason for this is not only the large number of application areas. Important reasons are also that the estimation problems are nonlinear and that different parametrizations of the orientation need to be considered (2), each with its own specific properties.

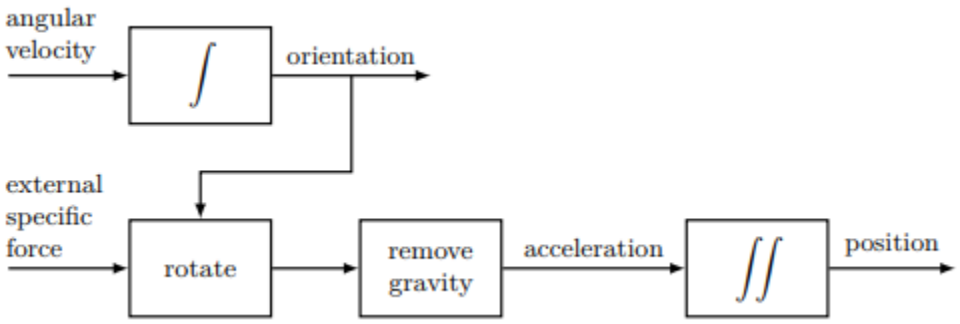

Figure 2.Schematic illustration of dead-reckoning, where the accelerometer measurements (external specific force) and the gyroscope measurements (angular velocity) are integrated to position and orientation

Inertial sensors are frequently used for navigation purposes where the position and the orientation of a device are of interest. Integration of the gyroscope measurements provides information about the orientation of the sensor. After

subtraction of the earth's gravity, double integration of the accelerometer measurements provides information about the sensor's position. To be able to subtract the earth's gravity, the orientation of the sensor needs to be known. Hence, estimation of the sensor's position and orientation are inherently linked when it comes to inertial sensors. The process of integrating the measurements from inertial sensors to obtain position and orientation information, often called dead-reckoning.

In practice, however, the inertial measurements are noisy thus the integration steps from angular velocity and from acceleration to position introduce integration drift. Errors in the measurements have a large impact on the quality of the estimated position and orientation using inertial sensors only. This is particularly the case for position, which relies both on double integration of the acceleration and on accurate orientation estimates to subtract the earth's gravity. Because of this, inertial sensors need to be supplemented with other sensors and other models to obtain accurate position and orientation estimates. For orientation estimation, they are often used in combination with magnetometers, which measure the direction of the magnetic field.

In this chapter we focus on the signal processing orientation estimation using inertial sensors, discussing different modeling choices and several important algorithms.

THE COORDINATE FRAMES

In order to discuss the quantities measured by IMUs in more detail, a few coordinate frames (2) need to be introduced:

The body frame b is the coordinate frame of the moving IMU. Its origin is in the center of the accelerometer triad and it is aligned to the casing. All the inertial measurements are resolved in this frame.

The navigation frame n is a local geographic frame in which we want to navigate. In other words, we are interested in the position and orientation of the b-frame with respect to this frame. For most applications it is defined stationary with respect to the earth. However, in cases when the sensor is expected to move over large distances, it is customary to move and rotate the n-frame along the surface of the earth.

The inertial frame i is a stationary frame. The IMU measures linear acceleration and angular velocity with respect to this frame. Its origin is located at the center of the earth and its axes are aligned with respect to the stars.

The earth frame e coincides with the i-frame but rotates with the earth. That is, it has its origin at the center of the earth and axes which are fixed with respect to the earth.

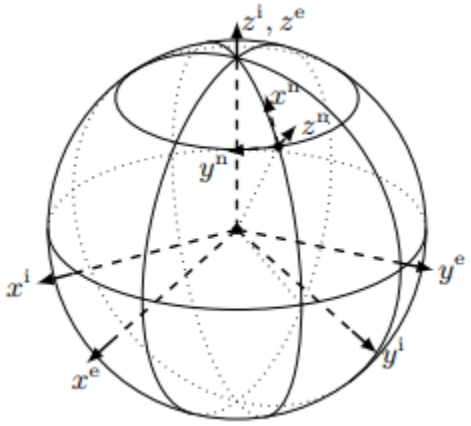

Figure 3. An illustration of three of the coordinate frames: the n-frame at a certain location on the earth, the e-frame rotating with the earth and the i-frame.

PARAMETERIZING ORIENTATIONS

In this section we introduce four different ways of parametrizing orientations (2). Note that these describe the same quantity and can hence be used interchangeably. There are differences in for instance the number of parameters used in the representation, the singularities and the uniqueness.

Rotational Matrices

A basic rotation (also called elemental rotation) is a rotation about one of the axes of a coordinate system. The following three basic rotation matrices rotate vectors by an angle θ about the *x*-, *y*-, or z-axis, in three dimensions, using the right-hand rule—which codifies their alternating signs. (The same matrices can also represent a clockwise rotation of the axes).

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Other rotation matrices can be obtained from these three using matrix multiplication. For example, the product,

$$R = R_z(\alpha)\, R_y(\beta)\, R_x(\gamma)$$

represents a rotation whose yaw, pitch, and roll angles (Euler angles) are α, β and γ, respectively.

Euler Angles

Rotation can also be defined as a consecutive rotation around three axes in terms of so-called Euler angles. We use the convention (z, y, x) which first rotates an angle ψ around the z-axis, subsequently an angle θ around the y-axis and finally an angle φ around the x-axis.

The ψ, θ, φ angles are also often referred to as yaw (or heading), pitch and roll, respectively. Furthermore, roll and pitch together are often referred to as inclination. Like the rotation vector, Euler angles parametrize orientation as a three-dimensional vector.

Euler angle representations are not unique descriptions of a rotation for two reasons. First, due to wrapping of the Euler angles, the rotation *(0, 0, 0)* is for instance equal to *(0, 0, 2πk)* for any integer. Furthermore, setting $\theta = \pi/2$, the only rotations that can be observed is φ and ψ. Because of this, for example the rotations $(\pi/2\ ,\ \pi/2\ ,\ 0)$, $(0,\ \pi/2\ ,\ -\pi/2\ )$, $(\pi,\ \pi/2\ ,\ \pi/2\ )$ are all three equivalents. This is called gimbal lock.

Unit Quaternions

A commonly used parametrization of orientation is that of unit quaternions. Quaternions (7) are an appealing tool for describing rotations in 3D as they do not suffer from gimbal lock, that impairs methods based on Euler angle. They were first introduced by Hamilton and are widely used in orientation estimation algorithms. A unit quaternion uses a 4-dimensional representation of the orientation according to

$$q = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \end{pmatrix}^\mathsf{T} = \begin{pmatrix} q_0 \\ q_v \end{pmatrix}, \qquad q \in \mathbb{R}^4, \qquad \|q\|_2 = 1.$$

A unit quaternion is not a unique description of an orientation. The reason for this is that if q represents a certain orientation, then −q describes the same orientation.

MEASUREMENT MODELS

Gyroscope Model

The gyroscope measures the angular velocity ω at each time instance t. However, its measurements are corrupted by a slowly time varying bias and noise. Hence the measurement model (6) is given by,

$$y_{\omega,t} = \omega^{\mathrm{b}}_{\mathrm{ib},t} + \delta^{\mathrm{b}}_{\omega,t} + e^{\mathrm{b}}_{\omega,t}.$$

Where ω is the angular velocity from inertial frame to body frame, δ is the gyroscope bias and e is the process noise.

The gyroscope measurement noise is quite Gaussian. If the sensor is properly calibrated, the measurement of the gyroscope axes is independent. It can be assumed that,

$$\Sigma_\omega = \begin{pmatrix} \sigma^2_{\omega,x} & 0 & 0 \\ 0 & \sigma^2_{\omega,y} & 0 \\ 0 & 0 & \sigma^2_{\omega,z} \end{pmatrix}$$

Accelerometer Model

The accelerometer (6) measures the specific force f^{b}_t at each time instance. The accelerometer measurements are typically assumed to be corrupted by a bias δ. and noise e as

$$y_{\mathrm{a},t} = f_t^{\mathrm{b}} + \delta_{\mathrm{a},t}^{\mathrm{b}} + e_{\mathrm{a},t}^{\mathrm{b}}.$$

The accelerometer bias is slowly time-varying. Like the gyroscope bias, the accelerometer bias can either be modeled as a constant parameter, or as part of the time-varying state. Since the accelerometer measures both the local gravity vector and the linear acceleration of the sensor, it provides information both about the change in position and about the inclination of the sensor. Neglecting the Coriolis acceleration, the measurement model can be simplified into

$$y_{\mathrm{a},t} = R_t^{\mathrm{bn}}(a_{\mathrm{nn}}^{\mathrm{n}} - g^{\mathrm{n}}) + \delta_{\mathrm{a},t}^{\mathrm{b}} + e_{\mathrm{a},t}^{\mathrm{b}}$$

Where, $a_{\mathrm{nn}}^{\mathrm{n}}$ is the linear acceleration and g^{n} is the gravity vector.

Naturally, the model is almost never completely true. However, it can often be used as a sufficiently good approximation of reality.

Magnetometer Model

Magnetometers measure the local magnetic field, consisting of both the earth magnetic field and the magnetic field due to the presence of magnetic material. The (local) earth magnetic field is denoted m^{n} and it is illustrated in Figure.

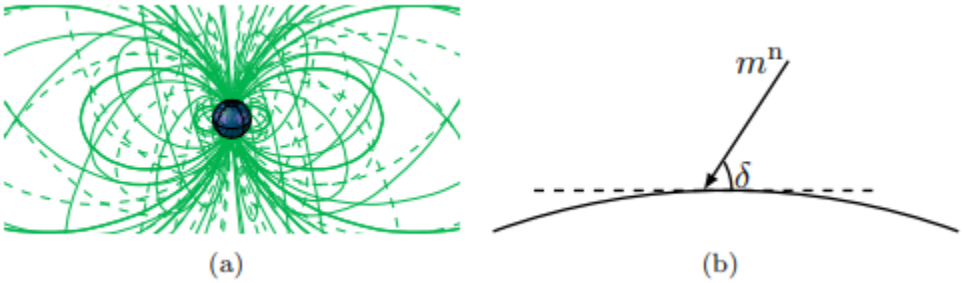

Figure 4. (a) Schematic of the earth magnetic field lines (green) around the earth (blue). (b) Schematic of a part of the earth where the local earth magnetic field m^{n} makes an angle δ with the horizontal plane. This angle is called the dip angle.

Its horizontal component points towards the earth's magnetic north pole. The ratio between the horizontal and vertical component depends on the location on the earth and can be expressed in terms of the so-called dip angle δ. The dip angle and the magnitude of the earth magnetic field are accurately known from geophysical studies.

If the sensor does not travel over significant distances as compared to the size of the earth, the local earth magnetic field can be modelled as being constant. In case no magnetic material is present in the vicinity of the sensor, orientation information can be deduced from the magnetometer. More specifically, magnetometers are typically used to complement accelerometers to provide information about the sensor heading, i.e. about the orientation around the gravity vector which cannot be determined from the accelerometer measurements. Magnetometers provide information about the heading in all locations on the earth except on the magnetic poles, where the local magnetic field m^{n} is vertical. Orientation can be estimated based on the direction of the magnetic field. The magnitude of the field is irrelevant. Because of this, without loss of generality we model (6) the earth magnetic field as

$$m^{\mathrm{n}} = \begin{pmatrix} \cos\delta & 0 & \sin\delta \end{pmatrix}^{\mathsf{T}}$$

If the magnetometer only measures the local magnetic field, its measurements $y_{\mathrm{m},t}$ can be modeled as

$$y_{\mathrm{m},t} = R_t^{\mathrm{bn}} m^{\mathrm{n}} + e_{\mathrm{m},t}.$$

The noise $e_{\mathrm{m},t}$ represents the magnetometer measurement noise as well as the model uncertainty.

4. FILTERS FOR ORIENTATION ESTIMATION

In this chapter, we will be discussing the various algorithms pertaining orientation estimation and optimization of IMU data.

STATE-SPACE MODEL REPRESENTATION

Before we can jump to the filters, we need to get familiar with how we can represent the entire state-space model. It describes the behavior of a system. The most general form of a linear discrete time invariant system is described by the two following equations,

$$x(k+1) = F\,x(k) + G\,u(k) + v(k)$$
$$y(k) = H\,x(k) + D\,u(k) + w(k)$$

Where x(k) is the state vector (e.g. Orientation parameters), at discrete time k

u(k) is the system input at discrete time k

y(k) is the system output (e.g. measurements) at discrete time k

F - State matrix

G – Input matrix

H – Output matrix

D – Feedthrough matrix

v(k) is the process noise

w(k) is the measurement noise

KALMAN FILTER

As the name mentions, Kalman filter (2) is not your traditional frequency filter but a recursive state estimator. Based on the system state and state covariance, it can describe how reliable a state estimate is and how much state variables change all together.

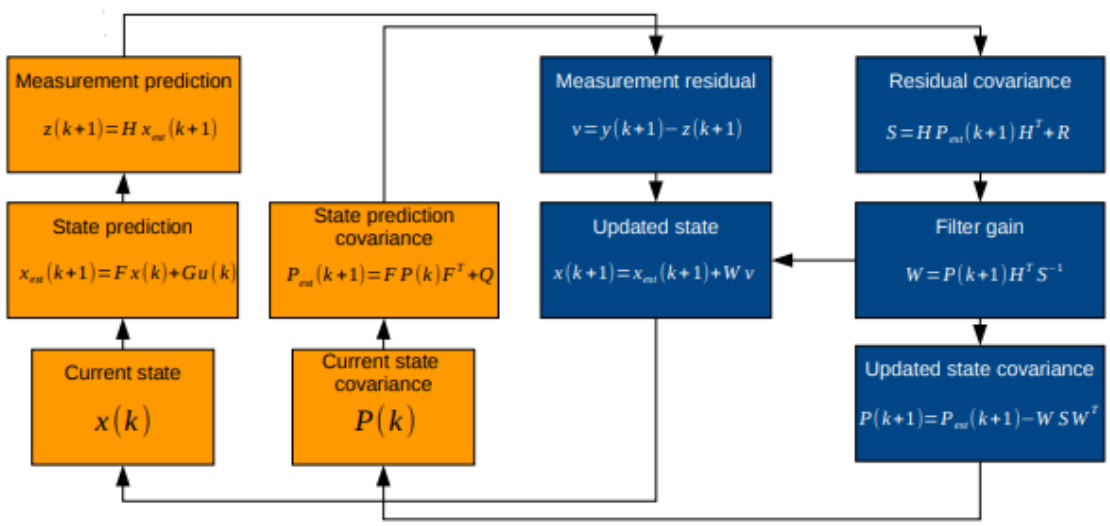

Figure 5. Block diagram of Kalman Filter

Where Q is the process noise covariance matrix and R is the measurement noise covariance matrix

Prediction – The next estimated states, measurements and state covariance are predicted.

Update – The residual state covariance and measurements are updated along with the states using filter gain.

The Kalman Filter is permanently weighing the residual based on the relation of process and measurement noise. The weighted residual is then used to update the system state. Measurement and process noise can vary with time. Different information sources can be weighted dynamically depending on a given situation.

COMPLEMENTARY FILTER

The complementary filter (4) gives us a "best of both worlds" kind of deal but uses only 6DOF IMU data. On the short term, we use the data from the gyroscope, because it is very precise and not susceptible to external forces. On the long term, we use the data from the accelerometer, as it does not drift. In its most simple form, the filter looks as follows:

$$angle = 0.98 * (angle + gyrData * dt) + 0.02 * (accData)$$

The gyroscope data is integrated every timestep with the current angle value. After this it is combined with the low-pass data from the accelerometer (already processed with atan2). The constants (0.98 and 0.02) must add up to 1 but can of course be changed to tune the filter properly.

Every iteration the pitch and roll angle values are updated with the new gyroscope values by means of integration over time. The filter then checks if the magnitude of the force seen by the accelerometer has a reasonable value that could be the real g-force vector. If the value is too small or too big, we know for sure that it is a disturbance we don't need to consider. Afterwards, it will update the pitch and roll angles with the accelerometer data by taking 98% of the current value and adding 2% of the angle calculated by the accelerometer. This will ensure that the measurement won't drift, but that it will be very accurate on the short term.

MADGWICK FILTER

Madgwick (4) has presented an interesting approach, which is based on Improving the orientation estimate by fusing accelerometer and gyroscope data as a minimization problem and solving it with a gradient technique.

Madgwick uses a quaternion approach to represent the attitude, which immediately poses the problem of how to convert the measured acceleration vector into a quaternion. Madgwick has described the problem in clear detail: A body's attitude (quaternion) cannot be unambiguously represented by a direction (vector) since any rotation of the body around that direction gives the same vector but a different quaternion. The solution manifold is a "line" and not a "point". Or plainly: The body's yaw angle is totally undetermined.

To tackle this problem, he suggested to determine that rotation, which brings the gravity vector in the earth frame in coincidence with the measured acceleration in the body frame, that is to find the quaternion respectively. Converting the measured vector to a quaternion is very desirable since then data fusing could be done directly on quaternions. In order to determine this rotation computationally, Madgwick suggested to formulate it as minimization problem and to solve it iteratively by the method of steepest decent.

This approach has two problems. The exact solution is not unique but there are infinitely many, and not the exact solution is calculated. One can hence expect that the yaw angle in the computed orientation is not only arbitrary but determined by the noise introduced by the incomplete steepest decent. The yaw angle fluctuates.

EXTENDED KALMAN FILTER

The Kalman Filter (8) (2) assumes a system of linear functions. The linear transformation of a normal distributed variable is normal distributed again. This is not valid for non-linear systems. In real world, almost every system is non-linear. In order to handle non-linear systems, the Kalman Filter is "extended" by approximating non-linearities with a Taylor Series Expansion. (9)A 1st order expansion is often enough. The 1st order Taylor Series Expansion is equivalent to the evaluation of the Jacobians of the functions F and H at a certain point x.

$$\boldsymbol{F}(k)=\left.\frac{\partial f[k,x(k),u(k)]}{\partial x}\right|_{x=x(k)} \qquad \boldsymbol{H}(k+1)=\left.\frac{\partial h[k+1,x(k+1)]}{\partial x}\right|_{x=x_{est}(k+1)}$$

To estimate and update the state:

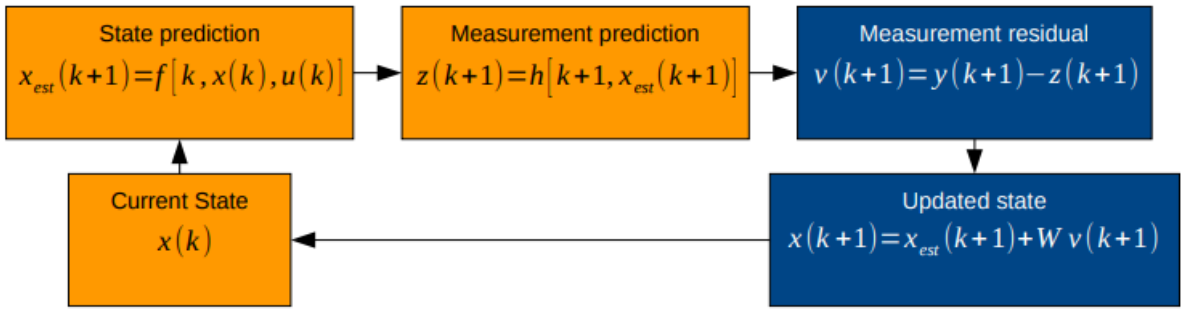

Figure 6. Block Diagram of Predict Step in EKF

And to estimate and update the state covariance:

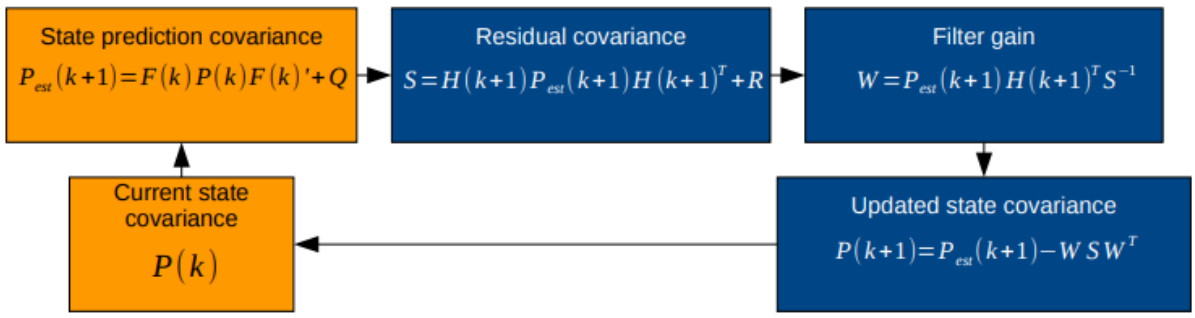

Figure 7. Block Diagram of Update Step of EKF

OBSERVATION & EXPERIMENTATION

In order to analyze the discussed filters, we were required to use an app called the Sensor Fusion app (10) that was developed by Linkoping University and referred to us by our supervisor. This app can record raw measurements from any sensor that is available on the respective smartphone used. This can be recorded or streamed live via WLAN to MathWorks MATLAB. Here, we have used a Samsung Galaxy S8 smartphone to read raw data from the accelerometer, gyroscope and magnetometer and recorded the following observations of the Euler angles after filtering them with the Madgwick filter and QEKF filter in MATLAB scripts.

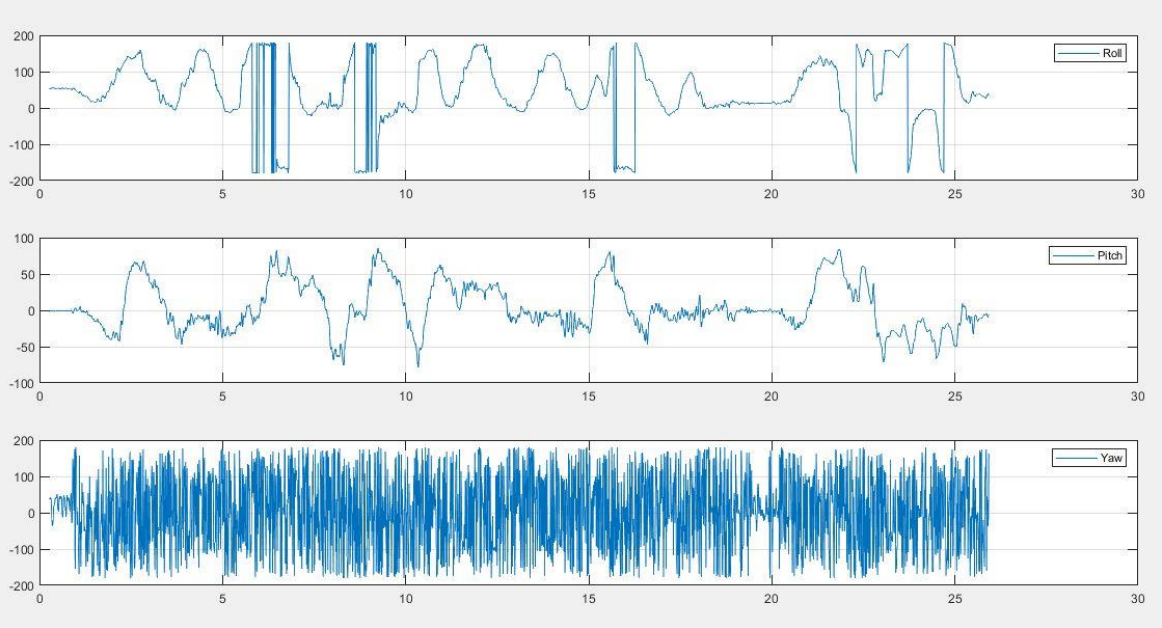

Figure 8. Raw Euler Angles from IMUs without Filtering/Optimization

The above fig. shows the Raw Euler angles without undergoing any filtration or optimization. As we can see, the amount of noise affecting the Euler angles are very high and the data we retrieve from the sensors are not accurate at all. Now, we implement the Madgwick filter onto these raw data and optimize in quaternion representations rather than Euler angles.

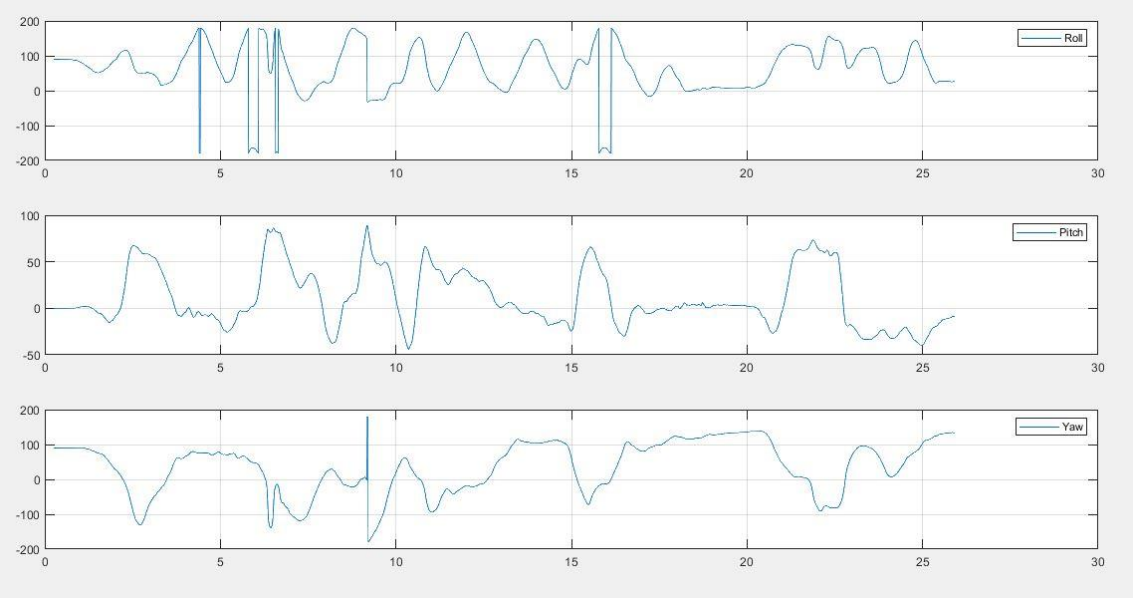

Figure 9. Optimized Euler angles using Madgwick Filter

This fig. shows the Euler angles after filtering them through the Madgwick filter. As we can see, most of the noise affecting the Euler angles have been filtered out and the Euler angle values seem optimized. However, the accuracy of the Yaw values is still way not optimal, and the Roll angles are still taking time to optimize. Still, this accuracy is better than Kalman filter and complementary filter themselves in practice.

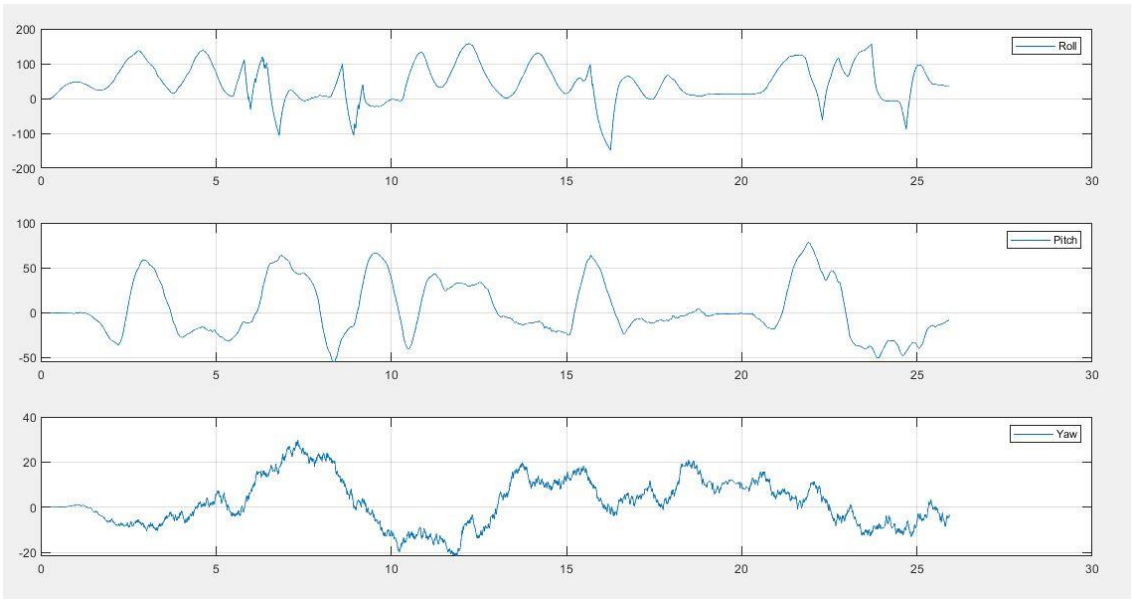

Figure 10. Estimated Euler Angles with Quaternion Extended Kalman Filter.

After Madgwick filter, we wanted to test Extended Kalman Filter since it was more promising than an optimizing filter. The above fig. shows the observations of the Euler angles after filtering them through the Quaternion based EKF (QEKF). We observed that using this filter we get the most accurate and optimized results. Even thou, the Yaw values are slightly noisy compared to Madgwick filtering, they are more accurate in real time. The Roll and Pitch angles on the other hand are more than perfect. Using these two filters, we want to build the application we had discussed to see how well they perform in a non-linear system like a glove-like device.

5. IMPLEMENTATION

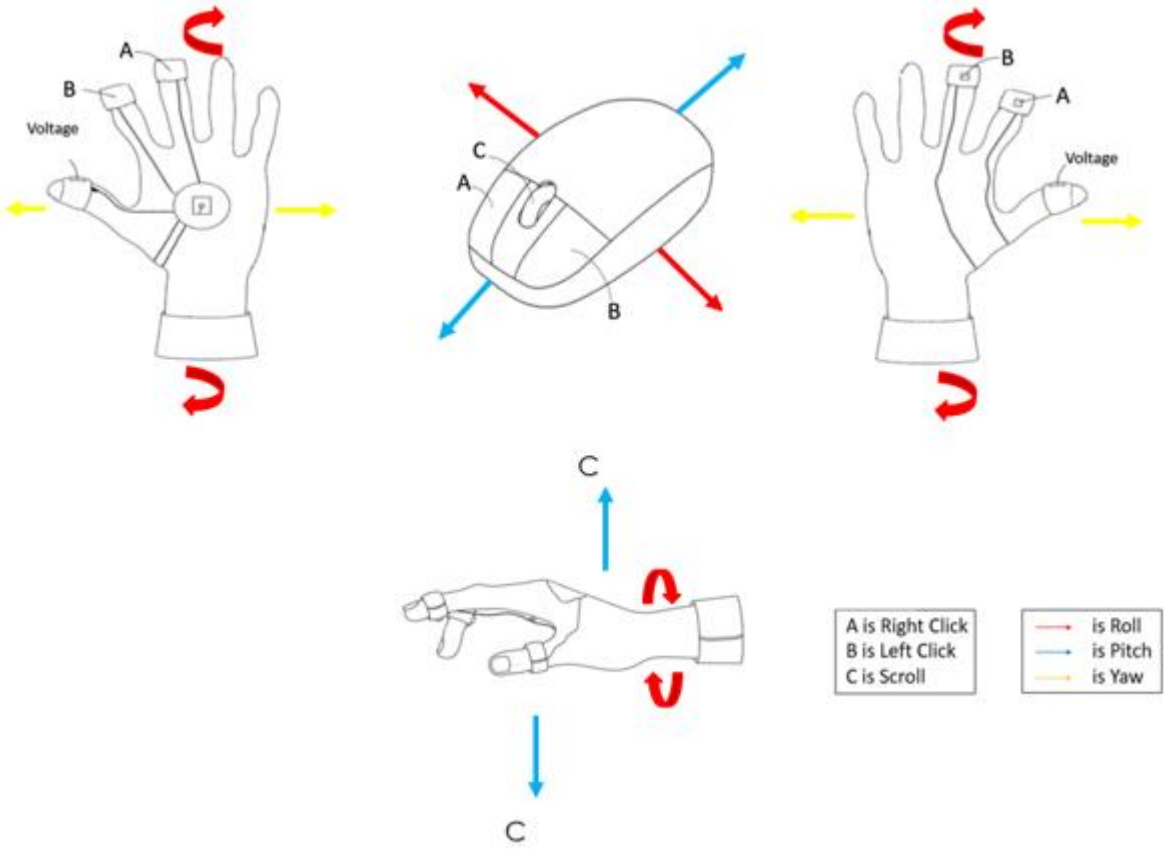

Figure 11. General orientation of how the glove is going to be performed.

Thus, using the observations that we have seen before, we want to implement these techniques onto an application. The glove used in our project is just a normal glove. The PCB containing the Arduino Nano, the MPU-9250 and the HC-05 transceiver is placed on the top of the glove. The contact pads were made for 3 fingers (ring finger, middle finger, index finger) which are connected to their individual 550 Ohm resistors. The contact pad on the thumb is connected directly to the pull-up digital pin on the Nano.

As you can see from the above figure, the red line describes the Roll, the blue line describes the Pitch and the yellow line describes the Yaw of the IMU. The index finger contact is programmed for left-click and the middle finger contact for right-click. The ring finger contact is programmed as an interrupt to pause the transmission of data from the Glove to the Base-Station. The thumb contact is used to provide voltage. Thus, whenever the thumb contacts one of the 3 fingers, it enables them to do their preprogrammed function. In theory, the thumb and fingers behave like a switch.

The Roll, Pitch and Yaw values, that is estimated/filtered from the raw data, are used to move the mouse pointer. The Roll moves the mouse pointer through the X-axis while the Pitch moves the mouse pointer through the Y-axis. The Yaw is used for the scroll function.

HARDWARE REQUIREMENTS

The hardware overall contains two units, the Glove and the Base-Station.

Glove Unit

Figure 12. Picture of our Glove product

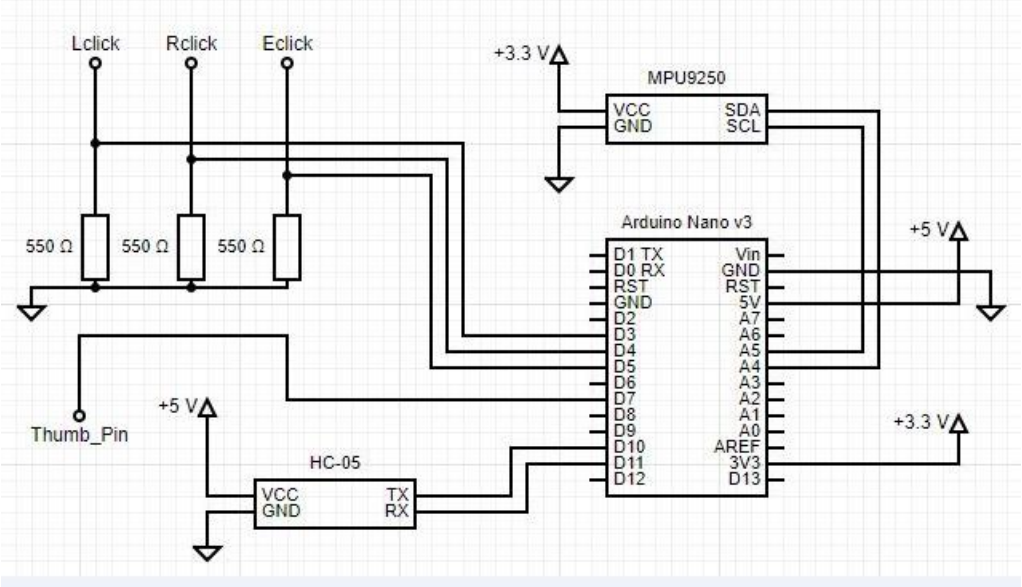

Figure 13. Circuit Schematics of the Glove Unit

The Glove comprises of the following:

Arduino Nano

The Arduino Nano (11) is a small, complete, and breadboard-friendly board based on the ATmega328P. The Arduino Nano is programmed using the Arduino Software (IDE) and runs both online and offline. Its tiny size makes it perfect for compact applications even though the functionally is the same as their larger counterparts.

Figure 14. Arduino Nano

MPU-9250 IMU chip

The MPU-9250 (12) is the latest 9-axis MEMS sensor from InvenSense®. The MPU-9250 has lowered power consumption and decreased size by 44% compared to the MPU-9150. It has claimed to have Gyro noise performance 3x better and a compass

full scale range over 4x better than competitive offerings. The MPU-9250 uses 16-bit analog-to-digital converters (ADCs) for digitizing all 9 axes.

It establishes serial communication between the Arduino Nano via the I2C communication protocol.

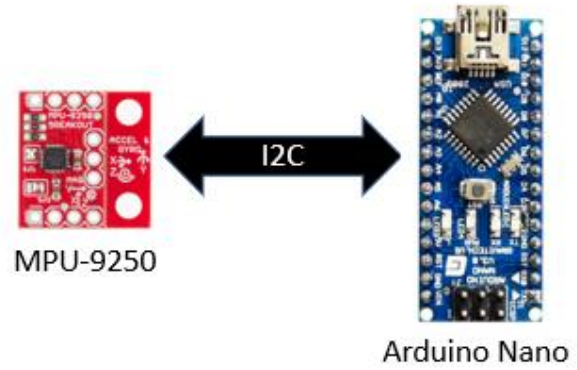

Figure 15. Connection between Arduino Nano & MPU9250

Bluetooth Transceiver HC-05 (Master)

HC-05 Bluetooth Module (13) is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Its communication is via serial communication which makes an easy way to interface with controller or PC. HC-05 Bluetooth module provides switching mode between master and slave mode which means it able to use neither receiving nor transmitting data.

For our project, we have programmed this HC-05 module in Master mode as all it needs to do is transmit data from the Glove to the Base-Station. The HC-05 establishes serial communication with the Arduino Nano via UART communication protocol.

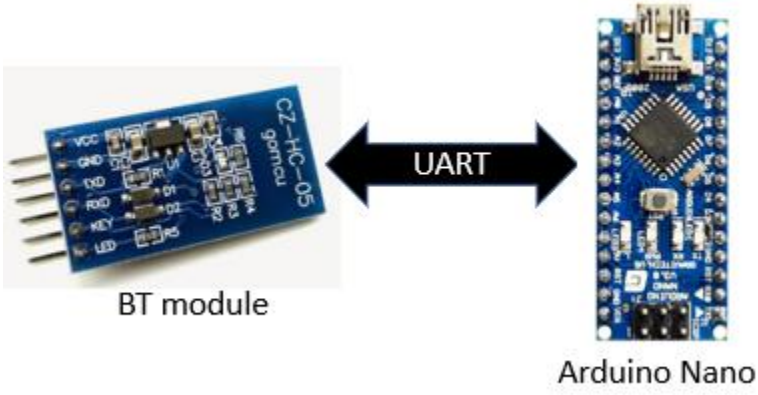

Figure 16. Connection between Arduino Nano & HC-05

Contact Pads

The contact pads are made using normal household aluminum foil paper and insulated tape. The 3 finger contact pads are connected to a 550 ohm resistor each. The thumb contact pad is connected to the pulled up digital pin of the Arduino.

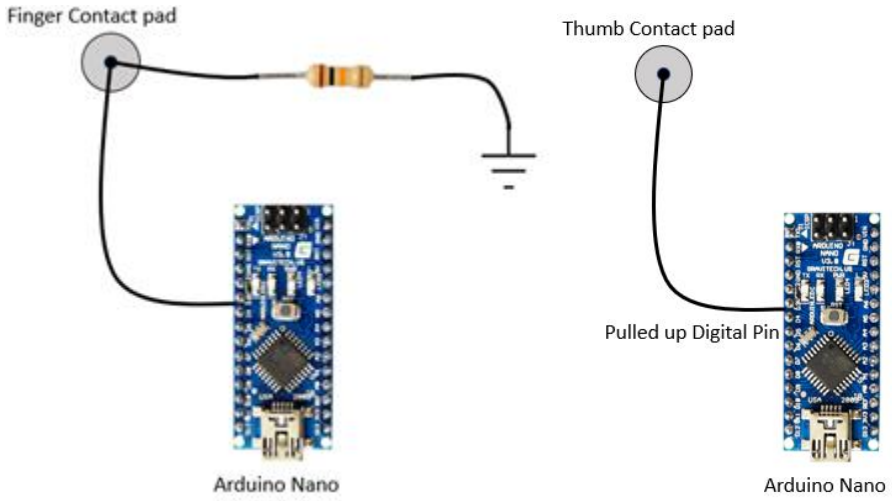

Figure 17. General Schematic of Contact Pads.

the Base-Station Unit

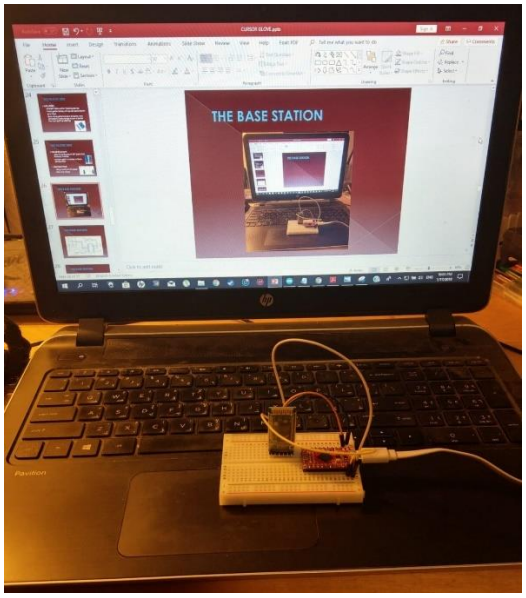

Figure 18. Base-Station Unit

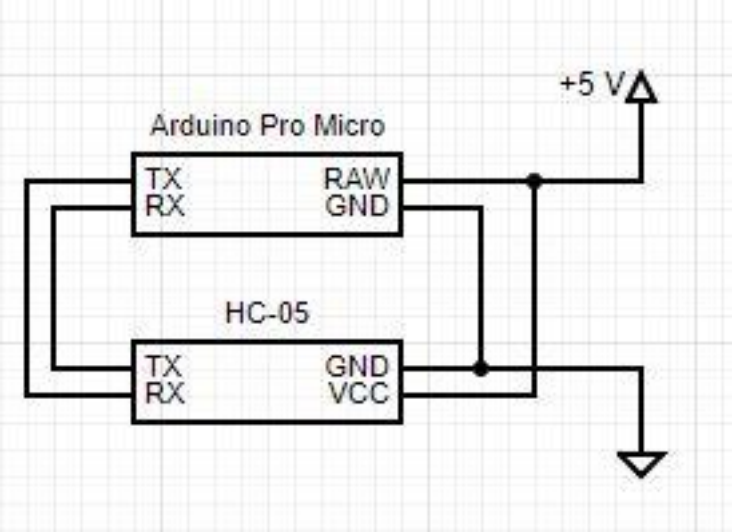

Figure 19. Circuit Schematic of Base-Station

The Base-Station only comprises of the following:

Arduino Pro Micro (3.3V/8MHz)

The Pro Micro (14) contains an ATmega32U4 on board. The USB transceiver inside the 32U4 allows us to add USB connectivity on-board and do away with bulky external USB interface.

This tiny little board does all the Arduino tricks that we're familiar with: 9 channels of 10-bit ADC, 5 PWM pins, 12 DIOs as well as hardware serial connections Rx and Tx. There is a voltage regulator on board so it can accept voltage up to 12VDC.

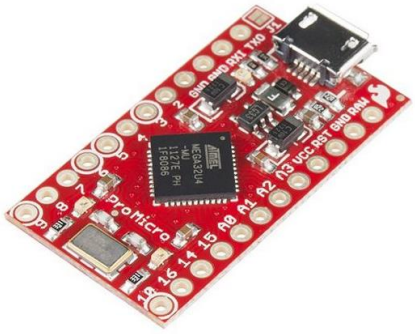

Figure 20. Arduino Pro-Micro

Bluetooth Transceiver HC-05 (SLAVE)

The same BT transceiver is used as in the glove unit. This HC-05 (13) we configured it in the Slave mode as all it needs to do is receive the data sent from the HC-05 at the Base-Station in the same format it has been sent to.

SOFTWARE IMPLEMENTATION

Each hardware component in the Glove and the Base-Station have their own software requirements which are explained below.

GLOVE Unit

MPU-9250

The Arduino library we used to program the MPU-9250 was taken from GitHub based on the work of Kris Winer (15).

The MPU consists of 16-bit Analog-to-Digital converters (ADCs) for digitizing all the 9 axes. The MPU establishes serial communication with the Arduino Nano via I2C communication protocol (16). The Inter-integrated Circuit (I2C) Protocol is a protocol intended to allow multiple "slave" digital integrated circuits ("chips") to communicate with one or more "master" chips. Like the Serial Peripheral Interface (SPI), it is only intended for short distance communications within a single device. Like Asynchronous Serial Interfaces (such as RS-232 or UARTs), it only requires two signal wires to exchange information. The digital motion processor (DMP) in the IMU stores the axes. It just requires simple calibration. The data is being sampled at 100hz.

Bluetooth transceiver HC-05 (Master)

The Bluetooth transceiver first begins UART (17) Serial communication with the Arduino Nano. A universal asynchronous receiver-transmitter (UART) is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. The electric signaling levels and methods are handled by a driver circuit external to the UART. A UART is usually an individual (or part of an) integrated circuit (IC) used for serial communications over a computer or peripheral device serial port.

This transceiver is configured to be in Master mode using the AT command mode (18).

If the Euler data and the respective contact states are valid, it sends data of type string with a certain format to the Slave Bluetooth transceiver in the Base-Station.

The format of the data being sent is represented this way:

<Roll, Pitch, Yaw, L-click, R-click>.

Arduino Nano

Using the Arduino IDE, we can program our MCU quite easily in Arduino environment. The Nano first checks if both the Bluetooth modules have successfully paired with each other. If they have established a successful communication, then the Arduino starts reading the DMP data to retrieve the respective axes. The Euler angles obtained from this data are updated instantaneously with the use of Madgwick filter. The Nano also reads the contact pads designed for left-click, right-click and for e-click (to pause the data transmission).

The Base-Station Unit

Bluetooth transceiver HC-05 (Slave)

This Bluetooth transceiver, using the AT mode (18), is configured to be in Slave mode and to automatically pair with the Master HC-05 in the Glove. The transceiver first begins UART Serial communication with the Arduino Pro-Micro. Once it pairs with the Master HC-05 at the Base-Station, it starts to receive the data that's being sent.

Arduino Pro-Micro

As discussed previously about the Arduino IDE, the Pro-Micro extracts the data received by the HC-05 and splits the information into individual components. This Arduino has an existing library in its forum that uses functions to directly manipulate the cursor on the display. The received Roll and Pitch values are extracted then mapped for the movement of the cursor through the Arduino mouse library (19). Two button states (left-state & right-state) are configured to implement the mouse button clicks (left-click & right-click). When the e-click is high, the mouse functions are suspended until clicked again since all data transmission is halted.

6. CONCLUSIONS AND FUTURE WORKS

The goal of this project was not to give a complete overview of all algorithms that can be used for position and orientation estimation. Instead, our aim was to implement these algorithms into an application that we can observe their reliability and robustness.

After implementing the whole application, we faced a couple of problems that we could have worked on if given more time period.

- Extended Kalman filter

  As the Arduino Nano is still quite a small MCU with limited memory, there wasn't enough dynamic memory to hold the EKF library we had created for Arduino. It had consumed almost 201% more SRAM than Arduino Nano comprised of. We wish to work on this further by optimizing the code to decrease the computational size or by using another MCU instead of the Nano e.g. The Arduino Mega 2560 which can hold.

- Battery powered

  The Glove works now by receiving power directly from the PC via the USB power cable. This limits the movement of the user and we would like to focus on making it battery operated so that it wouldn't affect the freedom of movement of the user.

- Eliminate Base-Station

  We would also like to eliminate the Base-Station and directly connect the Glove to the PC via Bluetooth. This way all the user needs to do is wear the glove and power it on and have full control of the mouse movement once the Bluetooth connection has been established. This can be done by building an application that reads the Bluetooth data.

- Hand gestures

  Apart from just mouse manipulation, we would like to incorporate other features like hand gestures into the Glove. This would make the user-environment interaction way friendlier and easier to use. The user will be able to configure the gestures and make the Glove their own.

- Integrate with other applications

  The Glove can be used in a variety of applications and we would like to broaden this environment and focus on using the Glove for other applications like gaming or Virtual reality.

- Adding more inertial sensors

  The Glove can be more robust and accurate by adding more inertial sensors into our state system e.g. Global Navigation Sensor / Global Positioning System.

7. BIBLOGRAPHY

1. **Adam Shih, Hyodong Lee.** Cornell University. [Online] 2012. http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2012/as986_hl525/as986_hl525/index.htm.

2. **Manon Kok, Jeroen D. Hol and Thomas B. Schon.** Using Inertial Sensors for Position and Orientation Estimation, Foundations and Trends in Signal Processing: Vol. 11: No. 1-2, pp 1-153. [Online] 2017. http://dx.doi.org/10.1561/2000000094.

3. **X-IO Technologies.** Open source IMU and AHRS algorithms. [Online] July 31, 2012. http://x-io.co.uk/open-source-imu-and-ahrs-algorithms/.

4. **Madgwick, Sebastian O.H.** An efficient orientation filter for inertial and inertial/magnetic sensor arrays. [Online] April 30, 2010. http://x-io.co.uk/res/doc/madgwick_internal_report.pdf.

5. **Kaiqiang Feng, Jie Li et al.** A New Quaternion-Based Kalman Filter for Real-Time Attitude Estimation Using the Two-Step Geometrically-Intuitive Correction Algorithm. [Online] https://www.mdpi.com/1424-8220/17/9/2146.

6. **Roberto G. Valenti, Ivan Dryanovski and Jizhong Xiao.** Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs. [Online] https://www.mdpi.com/1424-8220/15/8/19302.

7. **Robotics, CH.** Understanding Quaternions . [Online] http://www.chrobotics.com/docs/AN-1006-UnderstandingQuaternions.pdf.

8. **Strohmeier, Michael.** Quaternion based Extended Kalman Filter. [Online] https://wuecampus2.uni-wuerzburg.de/moodle/pluginfile.php/1109745/mod_resource/content/1/QEKF_Floatsat_WS16.pdf.

9. **Morrell, Darryl.** Extended Kalmna Filter Lecture Notes. [Online] 1997. https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/kalman/ekf_lecture_notes.pdf.

10. **Gustafsson, Fredrik.** Sensor Fusion app. [Online] http://users.isy.liu.se/en/rt/fredrik/app/.

11. **Arduino.** Getting Started with the Arduino Nano. [Online] https://www.arduino.cc/en/Guide/ArduinoNano.

12. **sparkfun.** MPU-9250 Hookup Guide. [Online] https://learn.sparkfun.com/tutorials/mpu-9250-hookup-guide?_ga=2.225372122.1124681611.1513648154-1486562402.1510876149.

13. **Electronic, GM.** HC-05 Bluetooth Module User's Manual V1.0. [Online] https://www.gme.cz/data/attachments/dsh.772-148.1.pdf.

14. **sparkfun.** Pro Micro - 3.3V/8MHz. [Online] https://www.sparkfun.com/products/12587.

15. **Winer, Kris.** Arduino sketches for MPU9250 9DoF with AHRS sensor fusion. *GitHub*. [Online] https://github.com/kriswiner/MPU9250.

16. **sparkfun.** I2C. [Online] https://learn.sparkfun.com/tutorials/i2c.

17. **Wikipedia.** Universal asynchronous receiver-transmitter. [Online] https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter.

18. **Dejan.** How To Configure and Pair Two HC-05 Bluetooth Modules as Master and Slave | AT Commands. *How To Mechatronics.* [Online] https://howtomechatronics.com/tutorials/arduino/how-to-configure-pair-two-hc-05-bluetooth-module-master-slave-commands/.

19. **Arduino.** Mouse. [Online] https://www.arduino.cc/reference/en/language/functions/usb/mouse/.