Contents

[Autonomous System 2](#_Toc340304971)

[Hardware 2](#_Toc340304972)

[Sensors 4](#_Toc340304973)

[IMU (Gyroscope + Accelerometer) [MPU-6000] 4](#_Toc340304974)

[Magnetometer [HMC5883L] 5](#_Toc340304975)

[GPS [MT3329] 6](#_Toc340304976)

[Airspeed Sensor [MPXV7002DP] 7](#_Toc340304977)

[Barometric Pressure Sensor [MS5611] 8](#_Toc340304978)

[Software 9](#_Toc340304979)

[Overview 9](#_Toc340304980)

[Sensors & Signal Filtering 9](#_Toc340304981)

[State Estimation 14](#_Toc340304982)

[State Transformation 17](#_Toc340304983)

[References 21](#_Toc340304984)

# Autonomous System

## Hardware

The autopilot system is at the core of the project and thus a lot of time has gone into studying and designing the system. The required functions of the autopilot are auto takeoff from a flat runway, waypoint navigation and finally landing autonomously at the point of takeoff. There are several different platforms available for accomplishing this task, such as the OpenPilot project, the PIXHAWK and the Ardupilot Mega.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Platform | uC | Sensors | Software | Popularity | Cost | Score |
| OpenPilot | 1 | 1 | 1 | 1 | 2 | 6 |
| Ardupilot | 2 | 3 | 3 | 3 | 3 | **14** |
| PIXHAWK | 3 | 2 | 2 | 2 | 1 | 10 |

Figure Autopilot Decision Matrix

The platform chosen is the Ardupilot Mega because of a careful study of its individual components and having the most robust design. It is powered by an Atmel AVR ATMega2560 microcontroller which is fast and reliable. There are several advantages of this platform, one such advantage is that the AVR microcontroller is very popular and has a lot of support online. This can be critical in developing the code that would run on this board, as the more reference material is available the easier there is to find solutions to problems. Another advantage is that the package is completely assembled out-of-the-box; there is no need to get any more electronics and no soldering is required. Furthermore, the autopilot code for the Ardupilot is open-source and well documented thus modifying it would be easier than rewriting the whole stack. Lastly, unlike other autopilot platforms this system incorporates the MPU-6000 IMU rather than a separate accelerometer and gyroscope. The core advantage is that an IMU has a built-in compensator for the accelerometer during banking/pitching. In any system that incorporates an accelerometer, there needs to be a method to keep track of the direction of gravity. Primarily because accelerometers report real values, as in if an accelerometer is completely stationary and on a flat surface then a value of 1**G** is register in the zed direction. And it is critical for the calculations of the autopilot to compensate for this, since erroneous acceleration values would make the system fail catastrophically.

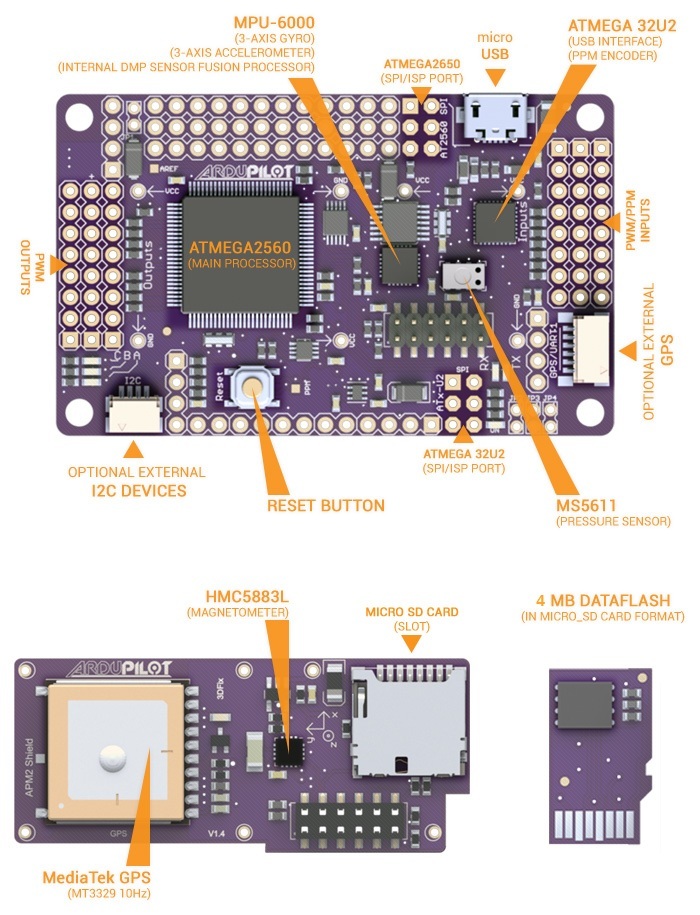


Figure 2 Ardupilot Platform [1]

## Sensors

### IMU (Gyroscope + Accelerometer) [MPU-6000]

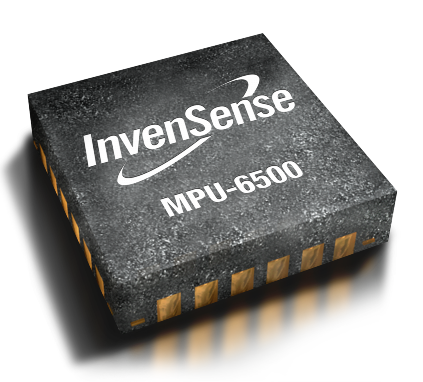


Figure MPU-6000 IMU Module [2]

Table 1 IMU Specifications [7]

|  |  |  |  |
| --- | --- | --- | --- |
| **Gyro Full Scale Range** | **Gyro Sensitivity** | **Accel. Full Scale Range** | **Accel. Sensitivity** |
| (°/sec) | (LSB/°/sec) | (g) | (LSB/g) |
| ±250 ±500 ±1000 ±2000 | 131 65.5 32.8 16.4 | ±2 ±4 ±8 ±16 | 16384 8192 4096 2048 |

The MPU-6000 includes both a gyroscope and accelerometer in a single package. Also the package includes a small microcontroller that includes several motion sensor algorithms that would allow the implementation of motion detection quickly and efficiently. However, this IMU allows for direct retrieval of raw accelerometer and gyroscope data. This is particularly beneficial for the implementation of more advanced algorithms for motion detection, because airplanes require different algorithms compared to tablets. A gyroscopes primary function is to measure the angular momentum of the airplane during its flight and maneuverings. From the angular momentum it would be possible to integrate the signal and obtain the change in angle. The accelerometer is also critical in navigation because it reports the linear acceleration of all three axes. Again using the values of the accelerometer is essential to obtain the velocity and position of the aircraft.

### Magnetometer [HMC5883L]

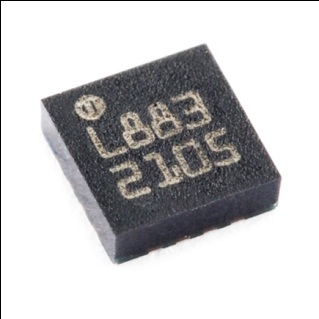


Figure 4 HMC5883L [3]

The job of a magnetometer is to give the magnitude and direction of any magnetic field. In this case the magnetometer is used detect the magnetic field of the earth, thus the direction of magnetic north. With having magnetic north defined, the magnetometer can be used as a digital compass. Having the heading is critical for designing the auto take-off and auto landing systems. It is due to the need to choose the direction of the runway for the auto landing procedure. Another advantage is the values retrieved during operation are linear, this fact will allow for easy calibration and easy conversion of values for use in the autopilot algorithms.

**Features**

* 1 Degree Accuracy
* I2C Interface
* 160Hz Output Rate
* Low Power (100 uA)

### GPS [MT3329]



Figure 5 MT3329 GPS Module [4]

The GPS is the most critical sensor for the autonomous flight capabilities of the aircraft, because an accurate state estimation is critical for this system. Without the GPS sensor then there will be no way for the aircraft to find the values of longitude and latitude. Even through the GPS delivers several different data values, such as altitude, velocity and heading. None of the values will be used other than the longitude and latitude because the GPS updates at most at about 10Hz, which is not sufficient for the performance requirement of the aircraft. This GPS also has a built in ARM processor that takes a lot of the processing tasks of the primary controller, thus just delivering for the autopilot a data stream without any overhead saving development time.

**Features**

* 10Hz Update rate
* 3-meter Position Accuracy
* 35 second Cold start acquisition speed

### Airspeed Sensor [MPXV7002DP]

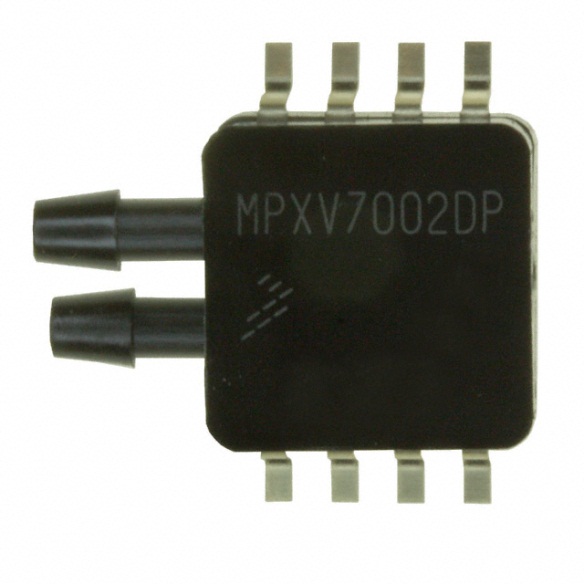


Figure MPXV7000DP Dynamic Pressure Sensor [5]

For finding the airspeed over the wings of the aircraft, there will be a pitot tube attached to a dynamic pressure sensor. The dynamic pressure sensor uses a piezoelectric transducer in the package of the sensor to measure the difference between the static and dynamic pressure, and return the value to the microcontroller. The airspeed sensor is critical for insuring that there is continuous lift over the wing of the aircraft. If there is at any point where the aircraft approaches the airspeed where there is little lift then the aircraft will stall and make recovery difficult. In the autopilot system there are checks and balances that make sure that does not happen.

**Features**

* 2.5% Error over 10C to 60C
* Accuracy of 1.0 kPa
* 1.0 ms Response time

### Barometric Pressure Sensor [MS5611]

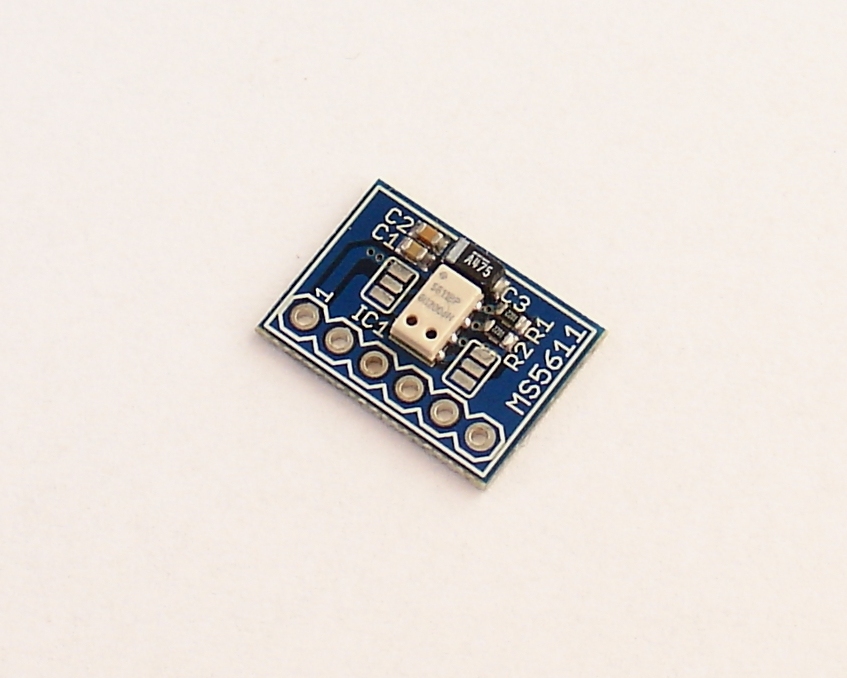


Figure MS5611 Barometric Pressure Sensor [6]

Lastly there is the barometric pressure sensor, which will be used to measure the altitude of the aircraft. This is accomplished by measuring the atmospheric pressure and converting the value using a fixed set algorithm to altitude. This module has a resolution of 10cm, which is quite impressive and is adequate for the systems purpose. Prior to reading the data sheet for this module, there were plans to use a sonar module. There is also an advantage to using this module, with respect to expected noise from the sensor. The characteristic noise of the sensor is about +/- 2.5 mbar which will allow the system to use a less computationally heavy signal filter.

A barometric pressure sensor is built similar to the airspeed sensor, which also measures pressure. A piezoelectric element placed in a plastic enclosure for protection of the elements from impact and debris.

## Software

### Overview

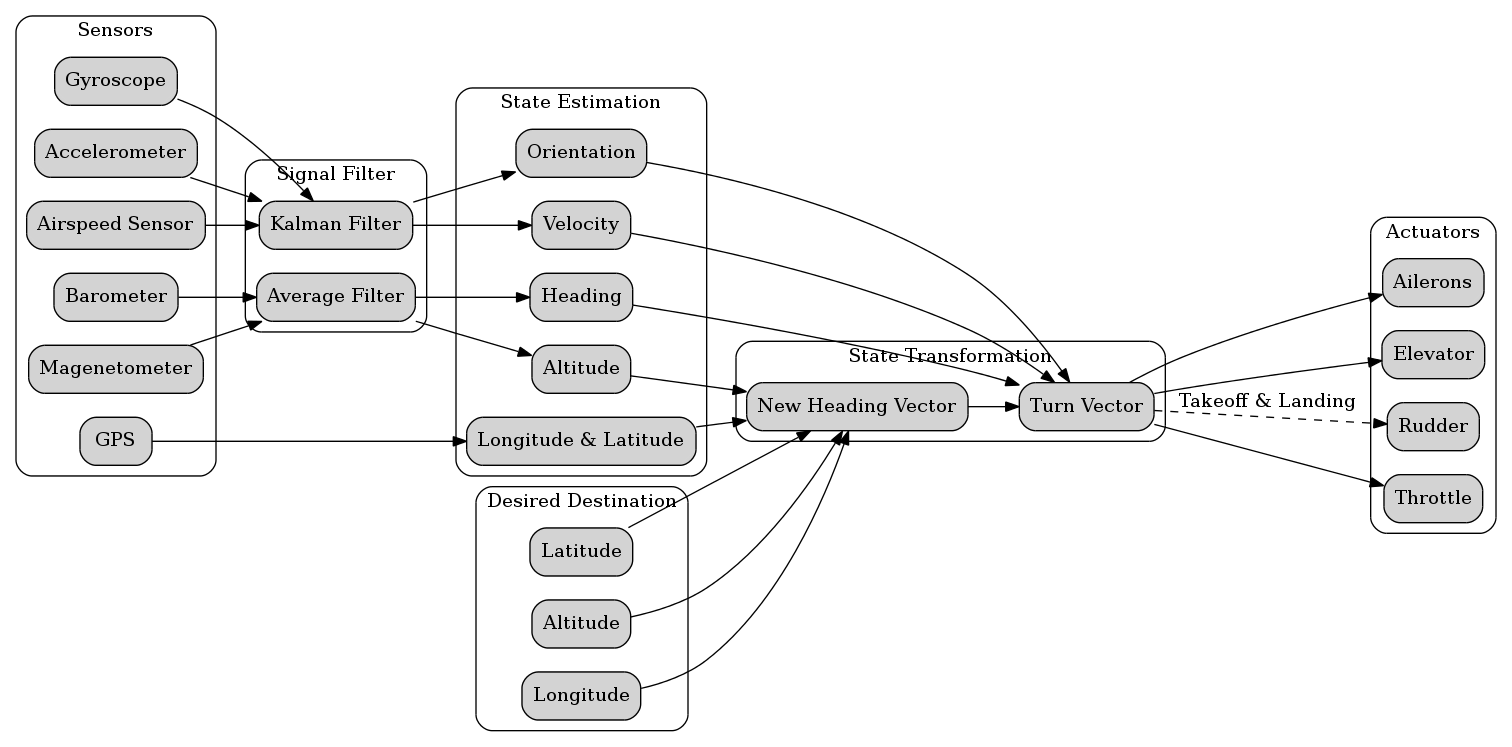


Figure 8 System Breakdown

There are many stages to the autopilot system, these can be split in to two distinct groups estimation and transformation. Estimation is the process by which the system processes sensory input to determine the state of the system. For the aircraft these states are longitude, latitude, altitude, velocity, orientation and heading. The second group of processes is the transformation specific functions which take the estimates derived previously and apply desired transformation to them. From the desired transformation the system can manipulate the actuators to achieve the desired outcome. This all occurs with a feedback from the state estimators to make sure there is minimal error.

### Sensors & Signal Filtering

There are several parts to the process of state estimation. Firstly the sensor input is critical to have a good state estimate. The error value of each sensor is extremely important when designing the state estimation functions, because it is critical during the next stage of processing. Thus researching the sensors are important when designing the electronics platform of the system. In the system designed for autopilot, there was the advantage of having a system prebuilt from the beginning with these factors in mind. All these factors were discussed previously in the hardware section of the report.

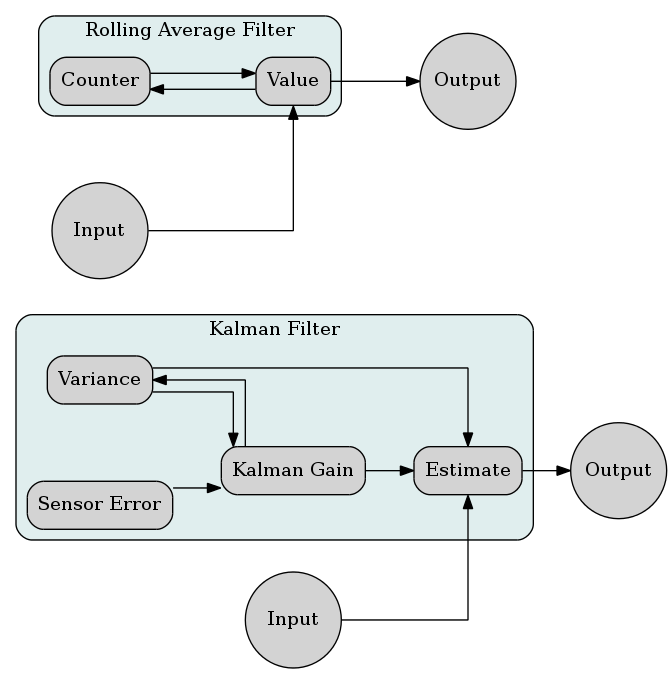


Figure Signal Filtering Logic Diagram

The next stage of the state estimation is the signal filtering for the sensors to achieve a usable reading. Typically a sensor has noise in the signal from several sources such as disturbances in the atmosphere. The process of signal filtering is a balancing act between accuracy and response. The more processing that takes place on a signal then there would be a larger delay in response. There are many signal filtering methods to choose from, each with their own advantages and disadvantages. The key factors in choosing a filtering method are processing restrictions, accuracy, and state behavior. The microcontroller’s raw computational strength is by far the largest of these factors. Since any lag in the response could be catastrophic to the system or to external entities. State behavior is defined as how common a change in a system is expected. For example in a civil airplane the majority of the time the state is a stable cruise on a fixed heading. This is not true for military aircraft where there is a lot of turning and dog fighting thus signal states would have a large variation in values. After research and design considerations two filters were chosen to be implemented in the autopilot. The two filters chosen were a primary kalman filter and a secondary rolling average filter for less critical signals.

#### Rolling Average Filter

The Rolling Average filter uses a simple algorithm to calculate the state value.

Looking at the algorithm it is clear that it is simple to implement, indeed it can be done in handful lines of code. Using this method will reduce noise but will not lead to a converged value. This is useful for sensors that are not critical and do not need to give real-time values, such as the GPS locations and the barometric pressure sensor.

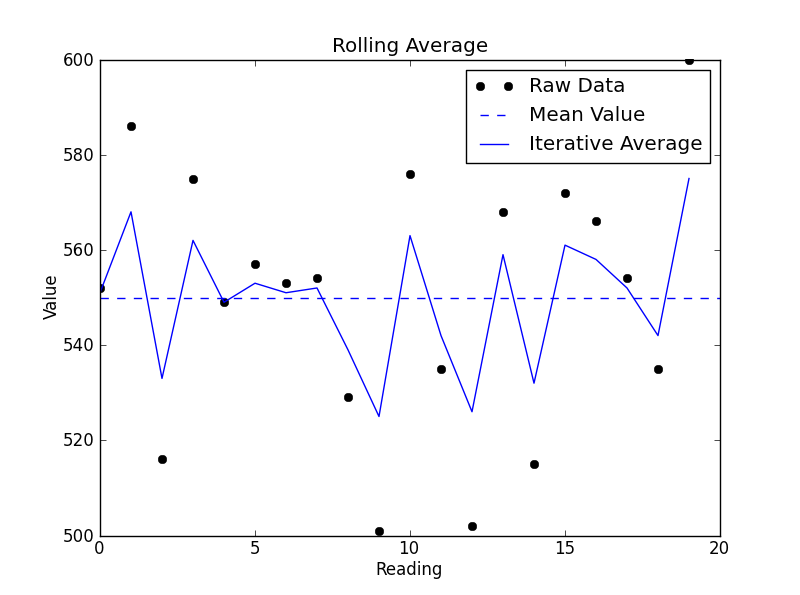


Figure Rolling Average Plot

Figure 8 shows that a rolling average is great if you intend on using about 20 readings to achieve an accurate value. Considering that some sensors can only achieve 10Hz then this is great if there is only a need for a 2 second delay. In cases of sensors like the GPS or the barometric pressure sensor during cruise a rolling filter is more than adequate. However for sensors that require fast accuracy there will be a need to sacrifice some computation cycles and implement a Kalman filter.

#### Kalman Filter

The Kalman filter at the time of writing of this report is the darling of the systems industry. It has been used in many sectors of industry from aerospace to manufacturing to military and others. The primary advantage of using the kalman filter is that the value quickly converges to an accurate value. The primary disadvantage of the kalman filter is that it is a multistep process which uses a more complicated algorithm. The algorithm used in the kalman filter is a bit more complicated and requires a more powerful processor to crunch the numbers.

#### Steps in a Kalman Filter:

**Step One: Calculate the Kalman Gain**

Variance in this equation is can be simply found from monitoring the signal and choosing and approximate value. Luckily the kalman filter converges fast thus it is possible to choose a random value and it would still return good value. Also the variance is also iterated on thus reducing the error significantly.

**Step Two: Update the Estimate**

Now the state can be calculated from using the previous estimate. The sensor reading is simply the raw sensor value.

**Step Three: Update Variance**

Now the variance is updated to reflect the new signal Variance.

**Step Four: Repeat Step One**

Return to step one, a new value for the kalman gain is found to reflect the new estimate and help in converging the estimate.

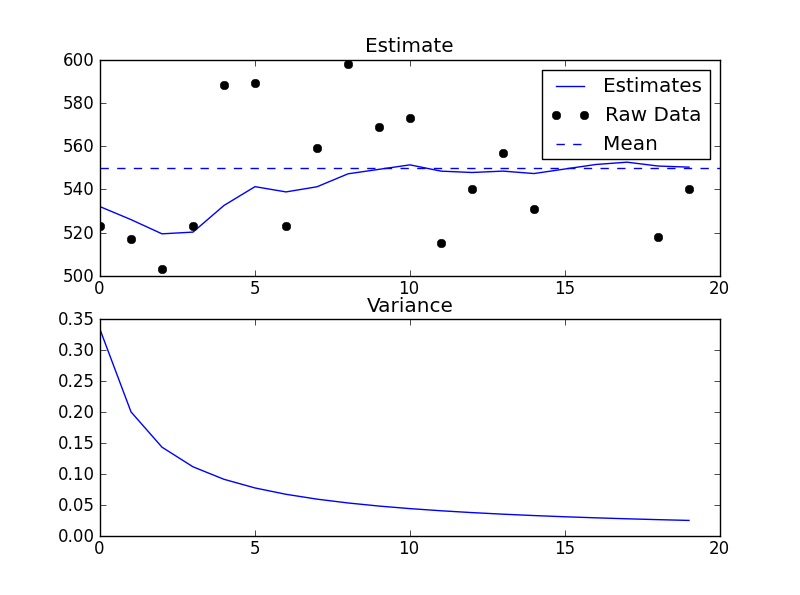


Figure Kalman Filter Example

Above is a sample of the kalman filter in use, the data quickly converges after only 10 readings. Thus using a kalman filter is almost twice as effective as using a rolling average filter. Thus for signals that require high accuracy quickly there is only on obvious choice with respect to the filter to be used.

### State Estimation

After completing the signal filtering stage of the system the data is then interpreted to define the current state. The important states of an aircraft are orientation, velocity, and spatial location (longitude, latitude and altitude). With this information the autopilot will be able to effectively make decisions on the state transformation of the system.

#### Orientation

The process of finding the orientation of the aircraft is a multistage process using multiple sensors. The first step is to use the accelerometer to find the initial orientation. There are key assumptions done at this stage, most relevant of which are that the aircraft is at a level state and is not in motion. The reason for these requirements is because the system will be able to determine the direction of the ground, and thus have a horizon set as reference for all transformation.

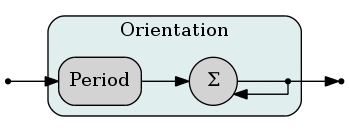


Figure Orientation Logic Diagram

The accelerometer returns 3 voltages for each Euclidean axis (X, Y, Z) and from these values the system can convert it to each of the angles.

The down vector calculated above is the unit vector of gravity. This vector will be used to ensure the aircraft levels correctly during flight. The Horizon Surface is not necessary for any calculation; however it is done primarily for visualization.

With the down vector defined the system will use the gyroscope to keep track of it. This is accomplished through simple mathematical formulas.

The formula is a simple kinematic equation used to return the angles of rotation about each axis. Next the system will apply these rotations using quaternion transformation.

Every vector in memory will have to go through this transformation, down vector and heading vector.

#### Velocity Vector

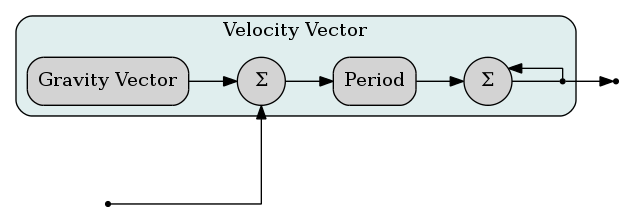


Figure Velocity Vector Logic Diagram

To calculate the velocity vector is slightly more complex than the orientation, primarily because the accelerometer readings need to be modified before being used in any calculation. The modification required is to remove effect of the gravitational acceleration on the aircraft. As mentioned earlier the accelerometer reads absolute acceleration, i.e. includes the effect of gravity. Thus to accurately use an accelerometer for the purpose of motion detection it must undergo this operation.

From the above equation it becomes simple to use the readings to find the acceleration of the system. Afterwards the system can integrate the values once to get the velocity and then integrate the reading again to find the position.

Note that the position is not used in any way in the system; it is simply an odometer to keep track of the flight distance.

#### Spatial Location

The location of the aircraft has three components, longitude, latitude and altitude. Longitude and latitude is quite simply read from the GPS module and no further processing is done on the reading. The reason for this is because the GPS module has all its processing done on its own processor. Many GPS algorithms are proprietary and cannot be used openly. Sadly GPS being a black box as it is means that we are at the mercy of it accuracy. The error expected from a GPS module could range from 3~9 meters which is fine for a large aircraft but for a drone it is not adequate landing on 2 meter wide runway.

Altitude is also not very difficult to find since it is a scalar and does not require vector mathematics like the orientation and velocity. The altitude can be simply derived from the direct measurement of the barometric pressure sensor. The only compensation that would be useful for the system would be to measure the ground level air pressure. Using the value as a reference point to compare to the air pressure during flight to prevent the aircraft from getting too low and impacting the ground.

### State Transformation

The process of computing state transformation is relatively simple compared to state estimation, primarily due to the lack of intermediary steps in the process. As seen previously state estimation requires a lot of pre-processing of data before the signal data is usable. In state transformation no pre-processing is required only some basic mathematics.

The process by which the autopilot works is simply done by following three steps. First, interpret the state estimate and do any conversions necessary to simplify process by which data is calculated. For example, when handling any vector math the system automatically converts all vectors to quaternion form. Second, compute the new heading vector. And lastly, compute the turn vector which is critical in finding the required values of the servos to achieve the desired pitch and yaw.

#### New Heading Vector

The process of calculating the new heading vector is quite simple mathematically. The GPS delivers the current locations longitude and latitude while the barometer will provide the current altitude. With these pieces of information the system will be able to generate a 3-axis point as the origin of the desired heading vector. The head of the heading vector is an input of the system, set using the ground station software provided by the Ardupilot Mega.

This heading will continually be updated by a feedback loop from the state estimator phase of the program. With this method there will continually be an updated heading vector ready for the next phase of the state transformation process. Take note that the above formula is only for reference purposes; the software internally converts all vectors to quaternions. The reasoning for the internal use of quaternions is for the ease of referencing of data, and since this a highly dynamic system it would benefit from this optimization.

#### Turn Vector

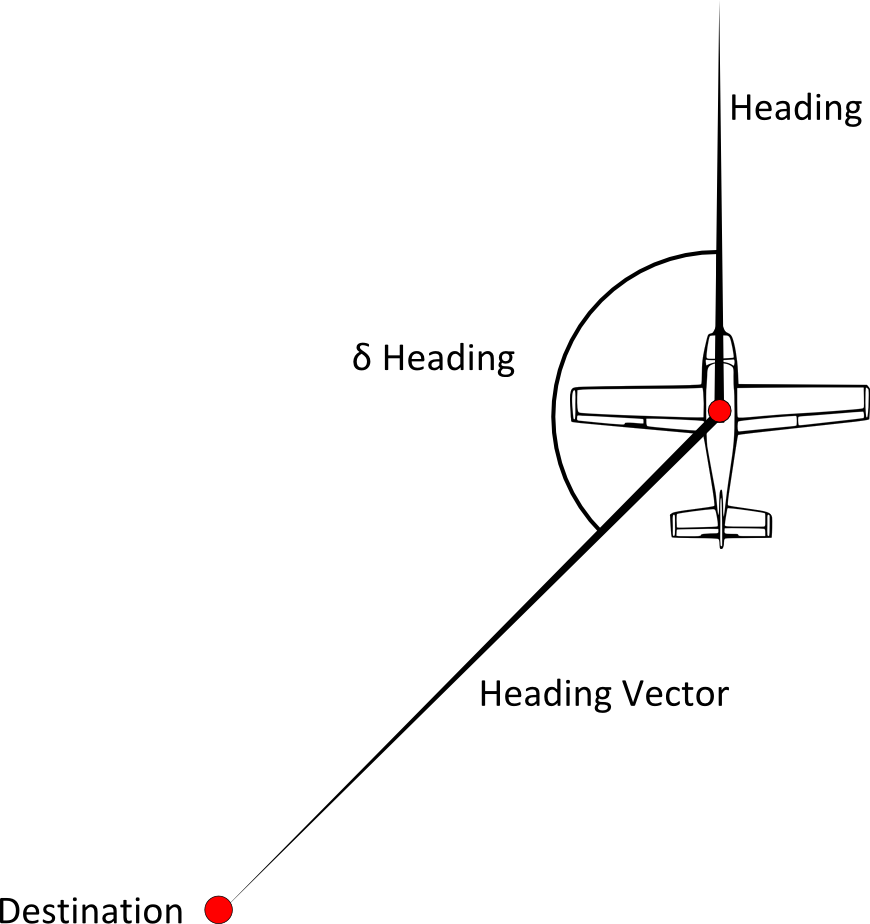


Figure 14 Plane Transformation

The turn vector is the system by which the aircraft changes its heading. Typically an airplane changes it’s heading by actuating the ailerons and rolling to an angle to accomplish the turn in the appropriate heading. However, if the turning rate is still too slow it can be increased by actuating the elevators to increase the rate of turn.

The method the autopilot behaves is by rolling the airplane to a fixed angle, and since the airfoil has lift it will slowly change the heading. Once the heading approaches the desired heading vector, the autopilot will roll back to level. This is accomplished using a PID controller which is set to monitor and modify the aileron angle to prevent any overshoot. Having a large overshoot is not optimal for the design of the autopilot, because the system needs to settle quickly. Also the system should not oscillate primarily because that would lead to instability and could lead to a catastrophic and an unrecoverable situation. The only method that can be used to prevent this is by using a trial and error tests, because the aircraft is not complete and cannot be modeled yet. Once complete a model may be possible if time permits however trial and error will be the primary PID tuning method that will be used.

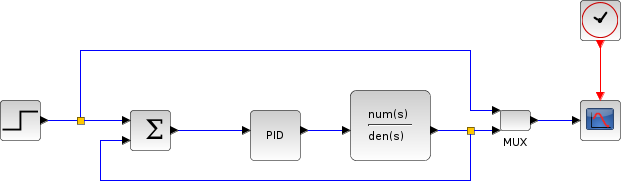


Figure PID Controller Example

# References

[1] “APM2 Ardupilot-mega Official Arduplane Repository”. Internet: <http://code.google.com/p/ardupilot-mega/wiki/APM2board>, Oct 10,2012 [Oct 23,2012]

[2] “HMC5883L”. Internet: <http://www.mindkits.co.nz/images/products/sparkfun/10494-01b-435.jpg>, Nov 5,2012 [Nov 5,2012].

[3] “MPU-6000”. Internet: <http://invensense.com/mems/gyro/mpu6000.html>, Nov 5,2012 [Nov 5,2012].

[4] “MT3329”. Internet: <http://store.diydrones.com/MediaTek_MT3329_GPS_10Hz_Adapter_Basic_p/mt3329-02.htm>, Nov 5,2012 [Nov 5,2012].

[5] “MPXV7002DP”. Internet: <http://www.digikey.com/product-detail/en/MPXV7002DP/MPXV7002DP-ND/1168436>, Nov 5,2012 [Nov 5,2012].

[6] “MS5611”. Internet: <http://www.rcgroups.com/forums/attachment.php?attachmentid=4031004>, Nov 5,2012 [Nov 5,2012].

[7] “MEMS Gyro-Accel”. Internet: <http://www.invensense.com/mems/gyro/mpu6050.html>, Nov 5,2012 [Nov 5,2012].