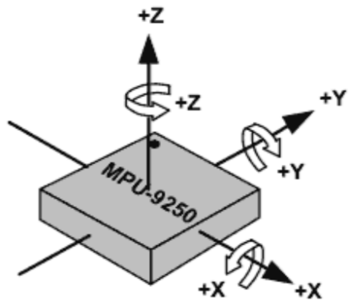
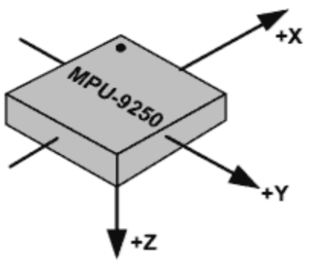
ENGR 16100
Supplementary Document: Inertial Measurement Unit

One method for measuring velocity and acceleration in real time is by using an inertial measurement unit (IMU). The IMU is a single board containing three sensors: an accelerometer (measuring linear acceleration), a rate gyroscope (measuring angular velocity), and a magnetometer (measuring magnetic field). The IMU system is a nine degree of freedom device, which means all three sensors measure along three dimensions, generating nine total readings.

However, the true signals of the real-life phenomena can be obscured by noise caused by random or systematic error. To clean the data generated, several filters have been applied to the signal, including window filtering and Kalman filtering. If you are curious about how these techniques work, more information may be found from a variety of Internet sources.



Orientation of Axes of Sensitivity and Polarity of Rotation for Accelerometer and Gyroscope



Orientation of Axes of Sensitivity for Compass

Figure 1: Example Coordinate Axes of IMU

IMU Setup
To interact with the IMU, it must be plugged into one of the GrovePi I2C ports. There are three of these ports located on the opposite end of the board than the pins.

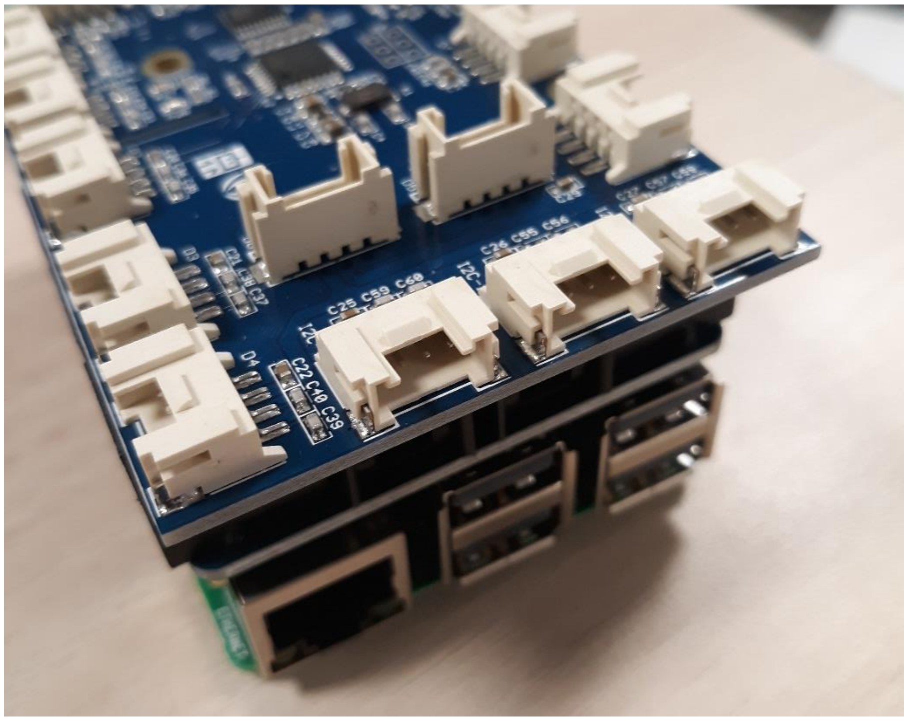


Figure 2: The 3 GrovePi I2C Ports

One the IMU is plugged in, it can be interacted with via the functions in the `MPU9250` library.

IMU Functions

To access the functions that correspond with the IMU, import the `MPU9250` library which can be accessed on Blackboard. Please note that for any code that uses the IMU, the program `MPU9250.py` **must** be in the same folder as the main program. Refer to the example programs posted on Blackboard under the same directory as this document for more information.

Once the library is initialized as an object:

```
myIMU = MPU9250()
```

The Main Functions in the MPU9250 library can be utilized:

- `myIMU.readAccel()` returns a list of three floats corresponding to the x, y, and z linear accelerations in g-force
- `myIMU.readGyro()` returns a list of three floats corresponding to the x, y, and z angular velocities in degrees per second
- `myIMU.readMagnet()` returns a list of three floats corresponding to the x, y, and z components of a vector pointing in the direction of the local magnetic field

Filter Functions

As hinted at above, the data from an IMU can tend to be fairly noisy, which can make it harder to gauge the reliability of the data. To aid in improving the clarity and reliability of the system's data, functions have been provided to aid in filtering the data. They are found in the library `IMUFilters.py` in the same directory as this document. A data logging program `filterLogcsv.py` is also included to show an implementation of these functions.

Functions in the `IMUFilters` library:

- `AvgCali(mpu9250, depth, dly)` returns the biases seen in each of the accelerometer and gyro axes as a list, in the format `[accelx, accely, accelz, gyrox, gyroy, gyroz]`. It takes the inputs of the `mpu9250` data object, the number of samples to take to find the bias (`depth`), and the delay between sample draws in seconds (`dly`).

- `genWindow(width, start)` returns a window structure used for filtering incoming data. It returns a list of lists. Element zero of this list is the filtered output. Element one is the width, the number of data points to include in the filter. Element two is the current index in the window. Element three is the window itself made up of one width's worth of previous data. The function assumes that the system starts the given starting value (`start`), meaning that the window will be full of the same starting value and the initial output will read as that starting value. This can be any given value but is usually the initial bias or zero.

- `WindowFilterDyn(window, dly, pull)` is a dynamic window filter that returns an updated window. Its inputs are the `window`, the delay between samplings in seconds (`dly`), and the new input to the window (`pull`), which should correspond to the new data for that particular window (a float). This filter averages over the whole window of data to give a filtered output. To adjust the filter, modify the `dly` and `window` widths. You can set the window width in the `genWindow` function when the window is created.

- `KalmanFilter(mpu9250, state, flter, dly)` is a very basic Kalman Filter. It assumes that each axis is independent of the others and that the noise factors are broken down into process and system noise. This function takes as its inputs an `mpu9250` data object, the previous state of all the accel and gyro axes (`state`, an array in the same order as what `AvgCali` returns), the filter values for the process and system noise (`flter`), and the delay between samples (`dly`). It returns the new filtered state that will eventually have to be fed back in as an input. To adjust the filter, adjust the system and process noise factors in the `flter` input for each axis. You can also adjust the delay between samples.

- `FindSTD(biases, mpu9250, dly)` is a function that finds the standard deviations of the noise of each of the accel and gyro axes. It takes as an input: the known `biases` of each axis of accel and gyro data (what `AvgCali` returns), the `mpu9250` data object, and the delay between samples in seconds (`dly`). It returns the standard deviations of the noise of each of the axes in the same order as they are input in `biases`.

- `InvGaussFilter(adv, value, bias, std, count)` filters any single axis. It takes as inputs a boolean (`adv`), the new `value` to be filtered, the `bias` for that axis, the standard deviation for the noise of that axis (`std`), and the `count` of standard deviations that will be counted as noise. The function operates by looking at the value and seeing how close it is to the original bias. If it is within `count` standard deviations `std` away, the value returned is modified. If `adv` is true, the value is set to the bias for that axis. If `adv` is false, the value is set to zero. The choice of `adv` is dependent on whether or not the user wishes to filter out the original gravity vector.

References

- http://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_Ch15.pdf
- https://ocw.mit.edu/resources/res-6-007-signals-and-systems-spring-2011/video-lectures/lecture-12-filtering/
- https://stanford.edu/class/ee363/lectures/kf.pdf
- http://wiki.seeed.cc/Grove-IMU_10DOF/