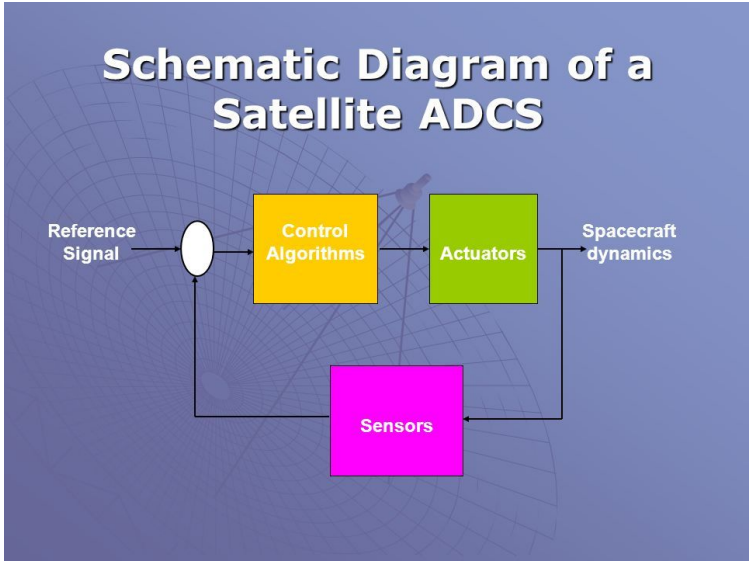
ADCS Documentation

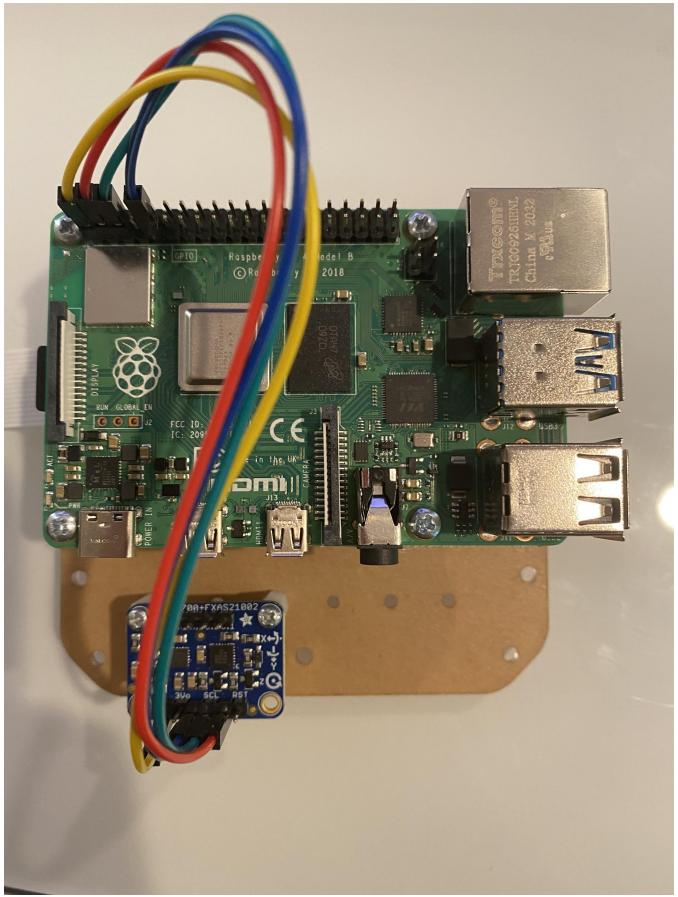
By Neil, Nevin, & Grant

What is ADCS?

Attitude Determination and Control System is used by satellites to determine acceleration, orientation, and position in space. If the satellite is drifting or tumbling, the ADCS controller will send instructions to the propulsion system in order to correct the attitude or spin.

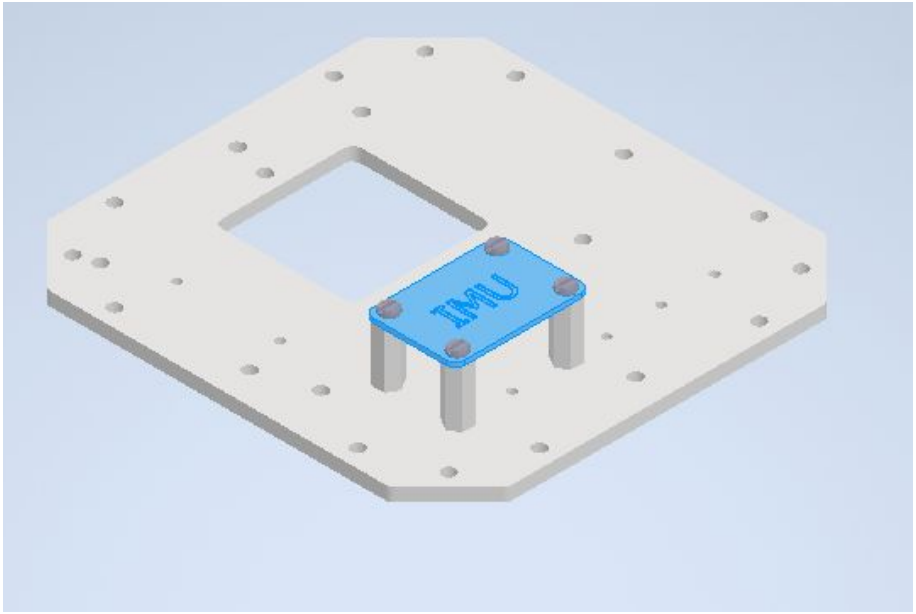


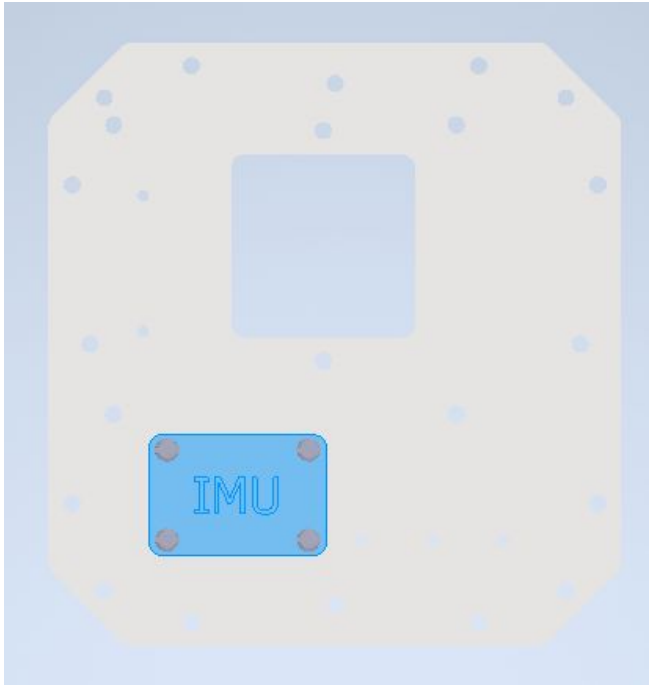
Our IMU + Raspberry Pi

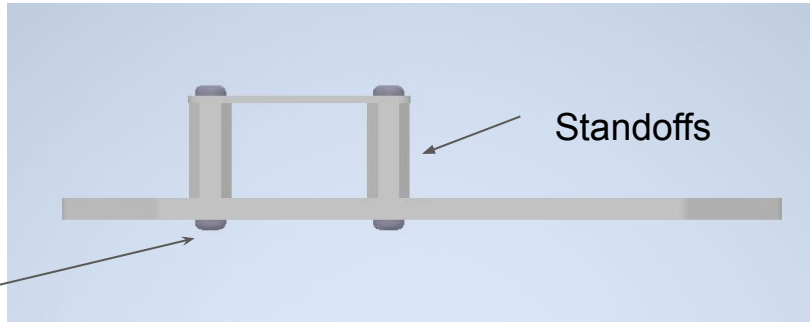


VIM to #1 3.3V DC Pin
IMU SCL to #5 SCL Pin
IMU SDA to #3 SDA Pin
IMU GND to #9 GND Pin

Mounting the IMU







Acceleration code

This code prints out the measured acceleration in m/s^2

```
import time
import os
import board
import busio
import adafruit_fxos8700

i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_fxos8700.FXOS8700(i2c)


def senseVals ():
    """This code prints out the acceleration values the IMU is reading
    while True:
        print("X: " + str(round(sensor.accelerometer[0],9)) + "m/s^2",
        "Y: " + str(round(sensor.accelerometer[1],9))+"m/s^2",
        "Z: " + str(round(sensor.accelerometer[2],9))+"m/s^2")
        time.sleep(1)

senseVals()
```

Sample Output

```
X: 1.411765334m/s^2 Y: 6.812365942m/s^2 Z: -0.906879765m/s^2
X: 0.289531535m/s^2 Y: -0.873380249m/s^2 Z: 10.214959679m/s^2
X: 0.3493521m/s^2 Y: -0.868594604m/s^2 Z: 10.229316615m/s^2
X: 0.30388847m/s^2 Y: -0.887737185m/s^2 Z: 10.219745325m/s^2
X: 0.325423874m/s^2 Y: -0.911665411m/s^2 Z: 10.219745325m/s^2
X: 0.311066938m/s^2 Y: -0.856630491m/s^2 Z: 10.217352502m/s^2
X: 0.330209519m/s^2 Y: -0.899701298m/s^2 Z: 10.236495083m/s^2
X: 0.279960244m/s^2 Y: -0.90209412m/s^2 Z: 10.236495083m/s^2
X: 0.366101858m/s^2 Y: -0.835095087m/s^2 Z: 10.217352502m/s^2
```

IMU Sensor Software

Adafruit_fxos8700

Accelerometer -> ax, ay, az = fxos.accelerometer

Magnetometer -> mx, my, mz = fxos.magnetometer

Adafruit_fxas21002c

Gyroscope -> gx, gy, gz = fxas.gyroscope

Calibration for Magnetometer, Gyroscope:

https://learn.adafruit.com/adafruit-sensorlab-magnetometer-calibration/calibration-with-raspberry-pi-using-blinka

Sources of Error

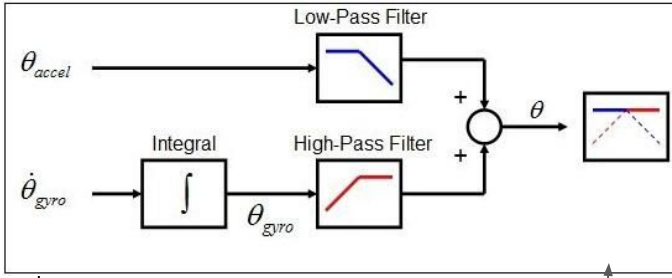
Random walk errors - Errors that grow over time because of random noise in the electronic systems

Gyroscope: Calibration error, gyroscope does not return to zero resulting in inaccurate angle measurement, Integration Errors

Accelerometer: Vibration Rectification Error, high levels of vibration can cause errors that add up over time

Magnetometer: Interference, nearby electronics as well as magnetic materials can cause inaccurate readings (Hard Iron, Soft Iron)

Calculating Orientation -> Flowchart



Accelerometer(Relative to Local Gravity)

Magnetometer (Relative to Magnetic North)

Gyroscope (Can't be used alone bc gyro drift)

Roll: arctan(accelY/accelZ)

Pitch: arctan(-accelX/sqrt(accelY^2 + accelZ^2))

Yaw: arctan((-My*cos(Roll) + Mz*sin(Roll)) / (Mx*cos(Pitch) + sin(Pitch) * [My * sin(Roll) + Mz * cos(Roll)]))

Roll: GyroRoll * dt

Pitch: GyroPitch * dt

Yaw: GyroYaw * dt

AccelRoll

AccelPitch

MagYaw

Filter(0.98*gyro + 0.02*otherSensor)

Pitch, Roll, Yaw

Calibrating the sensor

```
end = time.time_ns() + sampleLength*1000000000
reps = 0

while time.time_ns() < end:
    sumAccel += np.around(np.array(testAccel.accelerometer),9)
    sumGyro += np.around(np.array(testGyro.gyroscope),9)
    sumMagno += np.around(np.array(testAccel.magnetometer),1)
    reps += 1

sumAccel = sumAccel/reps
sumGyro = sumGyro/reps
sumMagno = sumMagno/reps
```