**HX711 load cell:**

Refer to <https://makersportal.com/blog/2019/5/12/arduino-weighing-scale-with-load-cell-and-hx711>  for instructions on how to connect load cell to HX711 to Arduino. All 3 load cells can share a common power and ground connection through a breadboard.

Download to library at <https://github.com/olkal/HX711_ADC> for examples on calibrating and reading values from multiple load cells.

To calibrate load cell, connect Arduino to an individual load cell through HX711 and run “Calibration” example. Follow instructions on the serial monitor and record down “calibration value”.  Do this for every individual load cell.

Use HX711\_script3.ino to read all 3 load cells. Input “calibration values” of each respective load cell into the code. Weight reading is taken 10 times with 0.05sec intervals and average weight on each load cell is obtained. Calculations are then made with the average weights. Run the code and read values of weight and coordinates of CG from serial monitor.

Send “t” on serial monitor to tare load cells immediately.

Reading is in grams. CG units are in mm.

**BMI160 gyro:**

Refer to <https://github.com/hanyazou/BMI160-Arduino> for instructions to connect to gyro sensor. Download library at the same website. Use example “Gyro” example or bmi\_160\_gyro\_script.ino to read from sensor. Switch example code to I2C mode.

OR use <https://github.com/DFRobot/DFRobot_BMI160#installation> . Download library and use accelGyro example for both gyro and accelerometer reading.

Use axis on the right when through holes are at the bottom

Finding MOI: In bmi\_160\_gryo\_script3.ino, modified from bmi\_160\_gyro\_script.ino, angular velocity reading is taken every 0.05sec and acceleration is obtained by dividing the difference in velocity by the time difference. 10 instances of angular acceleration are obtained and average is calculated. Input torque of motor in Nm in code.

In bmi\_160\_gryo\_script2.ino, modified from accelGryo example on <https://github.com/DFRobot/DFRobot_BMI160#installation> , linear acceleration readings in the x, y, and z direction are taken every 0.01sec and average acceleration in x and y directions are obtained from 10 readings. Net linear acceleration is obtained through Pythagoras theorem. Units of acceleration is initially in g and is converted to m/s2. This method is used because the slot to fix the Gyro chip is not at the center of the top plate. Input torque of motor in Nm in code.

Gyro (angular velocity) reading is in °/s but must be changed to radian/s for calculations. Hence, angular acceleration reading is in radian/s2.  Acceleration readings is in g.

Correct MOI values with reference to coordinates of CG suing parallel axis theorem.

Helpful links:

http://www.arduinoprojects.net/sensor-projects/using-bmi160-sensor-arduino-uno.php

https://learn.sparkfun.com/tutorials/gyroscope/all

**Linear actuator with relay:**

SPDT relay to be used as overall switch and 3 DPDT relay to be used as switches for individual actuators and to be able to reverse polarity of each individual motor.

Information on DPDT relays: <http://www.learningaboutelectronics.com/Articles/How-to-connect-a-double-pole-double-throw-relay-in-a-circuit>

Information and wiring for SPDT relay: <https://www.circuitbasics.com/setting-up-a-5v-relay-on-the-arduino/>  and <https://lastminuteengineers.com/one-channel-relay-module-arduino-tutorial/>

Wiring for DPDT relay to be able to reverse polarity: <https://forum.digikey.com/t/polarity-reversal-using-a-dpdt-switch/626/4>

Wiring:

VCC and GND of all relays to be connected to 5V and GND of Arduino through breadboard.

IN of SPDT, DPDT 1, 2, and 3 to be connected to Arduino IO pin 6, 7, 8, and 9 respectively.

For each DPDT relay, NO2 and NC1 should be connected to the black wire of actuator while NO1 and NC2 should be connected to the red wire of actuator.

Use linear\_actuator\_script.ino to control the actuators through serial monitor. Time values can be changed by changing the argument in the delay() functions.

Send ‘1’ to power on. Send ‘2’ to power off.

Send ‘3’, ‘5’, or ‘7’ to move actuator 1, 2, or 3 up for 1 sec respectively.

Send ‘4’, ‘6’, or ‘8’ to move actuator 1, 2, or 3 down for 1 sec respectively.

Send ‘9’ to move all 3 actuators up completely.

Send ‘10’ to move all 3 actuators down completely.

Voltage to use: 12V

Current for 1 actuator: 0.57A - 0.6A

Current for 3 actuators: 1.71A - 1.8A

**LICHUAN LCDA257S stepper motor driver (Model: LC60H2102):**

See <https://dronebotworkshop.com/big-stepper-motors/> to get an overview on stepper drivers and motors

See <https://www.makerguides.com/tb6600-stepper-motor-driver-arduino-tutorial/> for code and wiring

AccelStepper library: <https://github.com/waspinator/AccelStepper>

For AccelStepper library, speed is in steps/sec, Acceleration is in steps/sec2 , position is in number of steps

Recommended voltage: 36V

Current output: 5A

Gearbox ratio is 1:20

2 choices of Arduino code to use. Stepper\_motor\_driver.ino (without accelstepper library) and Stepper\_motor\_driver2.ino (with accelstepper library).

General equation for Stepper\_motor\_driver.ino:

r => desired rev/min   P => pulse/rev setting

Time for half a pulse(T) = 3/2(rP)

Value to input(t) = Tx106  (t must be integer)

**Putting it all together (Gyro and Motor):**

Using FreeRTOS on Arduino, gyro and motor will run and read data simultaneously. GitHub library is at <https://github.com/feilipu/Arduino_FreeRTOS_Library> . Unfortunately, can’t use delayMicroseconds() function in RTOS, need to resort to other methods of controlling motor. Code is in moi.ino.

Alternatively, we can use pulse width modulation (pwm) to send pulses to stepper motor driver. analogWrite() function to output pwm signal does not block out void loop(). Hence, we can input gyro code immediately after analogWrite() and start collecting data. Go to <https://www.etechnophiles.com/how-to-change-pwm-frequency-of-arduino-mega/> for code to toggle pwm frequencies on Arduino meg.  Code to change frequency of pin 4 and pin 13 pwm to 62500Hz changes delay() function such that delay(1000000) now is approximately 15 sec. Another code is used that do not change delay () function. To account for the gyro not being absolutely flat, offset values of acceleration in X and Y axis are calculated before measuring acceleration when motor starts accelerating. Offset is then made by subtracting acceleration values by the offset values. Motor is programmed to switch directions every time it stops to conserve wire length required by gyro. Code is in moi2.ino.

However, gyro sensor experiences large amounts of noise and values collected have large standard deviations. To reduce the standard deviation, gyro code had to use library from <https://github.com/hanyazou/BMI160-Arduino> to include a digital low pass filter in the code.  Also, a simple Kalman Filter found at <https://github.com/denyssene/SimpleKalmanFilter> is added into the code which drastically decreased standard variation when top plate is stationary as well as when it is moving, the final value is put through the filter.

Using FreeRTOS in moi.ino, a graph of angular acceleration against time is plotted through the acceleration and deceleration of the stepper motor. It is observed that there is a very sharp peak acceleration when stepper motor accelerates and acceleration. Hence the gyro will measure acceleration values throughout the motor’s acceleration and deceleration process and the code will record down the peak acceleration value. Peak acceleration values are put in an array and after a certain number or recordings, mean and standard deviation are calculated from the peak values in array. Code is in moi3.ino.

Note: Changing baud rate of serial monitor affects reading of accelerometer maybe because it affects pulses sent to step motor, higher baud rate results in higher angular velocity and standard deviation. Higher angular velocity tends to result in higher standard deviation. For some reason, even though code only records peak value, peak value can slowly decrease or increase over several recordings, no big jumps of values but steady increase/decrease. May be due to tape?

**Putting it all together (Linear actuator and Load Cells):**

Refer to above instructions “Linear actuator with relay” for instructions on how to operate linear actuators. Once Arduino is on, load cells are already reading weight values and calculating CG. Linear actuators should be tested first by moving each actuator up and down. Linear actuators should be extended all the way to measure weight of satellite. Satellite should be placed on the top plate after actuators fully extend and taring load cells. Send “11” to tare load cells. Wait for “Tare load cell 1 / 2 / 3 complete” message.  Arduino mega is able to support all 3 load cells and the 4 relays hence all 7 components will share the same VCC and GND through breadboard. Code is in cg.ino.

Note: no perceivable difference in data made by changing baud rate. Load cells tend to start off consistently around a value then it starts to slowly veer off to other values with at an increasing rate, this is probably caused by load cell 1 because load cell one has the largest noise. If want to use for loop, cannot use serialPrintInterval, for loop will block void loop and so we can only move actuator or tare load cells in between for loop.