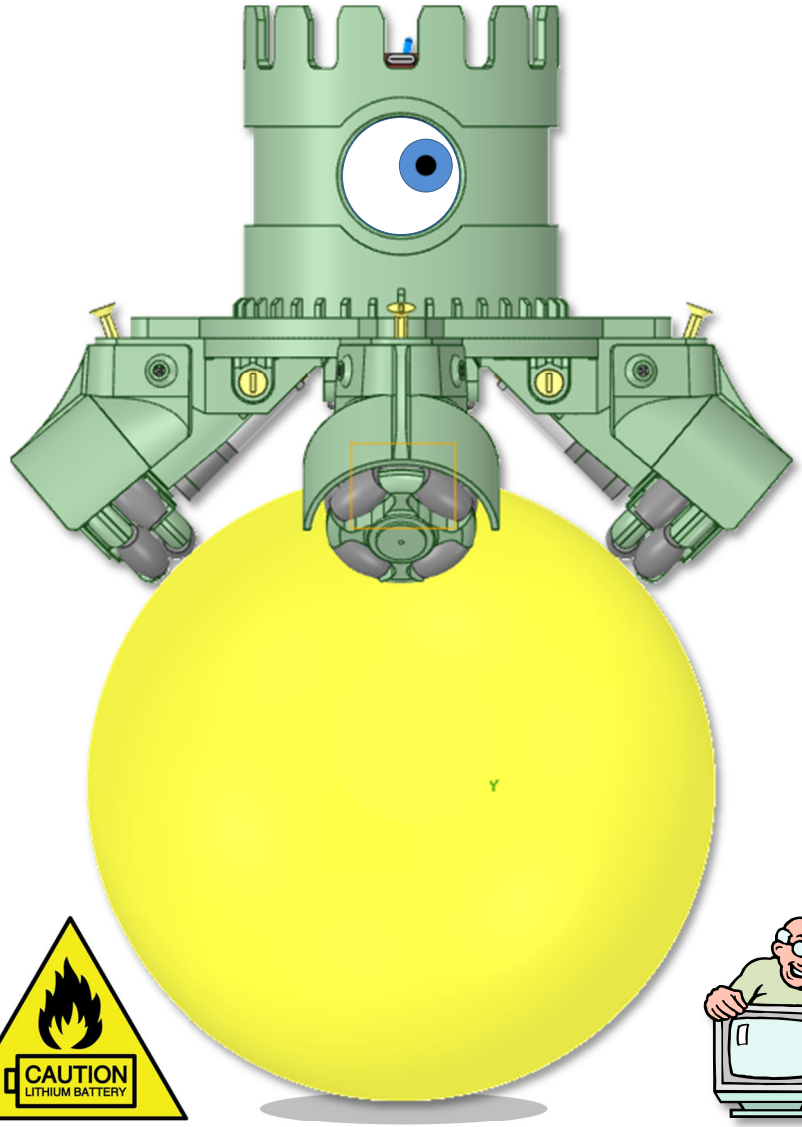
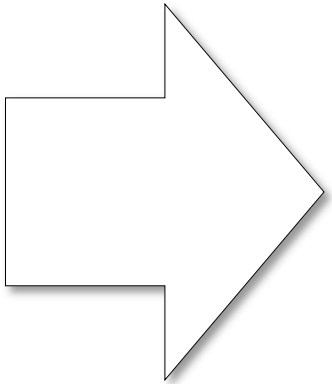
BallBot 4x4 (ESP32)

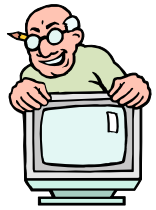
Calibration





An important process, that will govern your robot's performance.

CAUTION



Lithium batteries can be extremely dangerous, if not handled and cared for properly. This design does not include any form of current limiting circuit, like a fuse. So, care must be taken to ensure that the wiring guidelines are followed accurately, that checks are made for short-circuits, and that battery polarities are marked, and they are inserted the correct way round. Failure to do so, could result in an explosive fire.

Charging Practices: Always remove batteries from your project to charge them. Use a charger, designed for the battery used, and from a trusted supplier. Choose a flat, non-flammable surface to charge on, away from flammable materials. Never leave unattended when charging. Don't charge overnight. Monitor charging to ensure charge characteristics are as expected. Only pair batteries with similar characteristics. Do not overcharge, or leave charging for prolonged periods. This increases the risk of damage and fire.

Battery care & maintenance: Stop using a battery if it is swollen, damaged, dented or leaking. Never charge a damaged battery. Never allow a Lithium battery to discharge below 3.2 volts, as cell damage will occur. Avoid extreme temperatures. Do not charge or store batteries in very hot or cold environments. Don't cover batteries whilst charging, as this can trap heat, causing overheating.

In case of fire: Get out and stay out. If a fire starts, leave immediately, and call the fire brigade. For low voltage Lithium batteries, water is a safe extinguisher.

Built-in Monitoring: Most of my project designs include code, and circuitry, to monitor battery voltage, whilst in use. This code then seeks to alert the operator, when the battery has reached a critical low voltage, before shutting down power consuming circuitry; including the micro. Time should therefore be spent on calibrating this feature, as a precaution, for good battery management and maintenance.

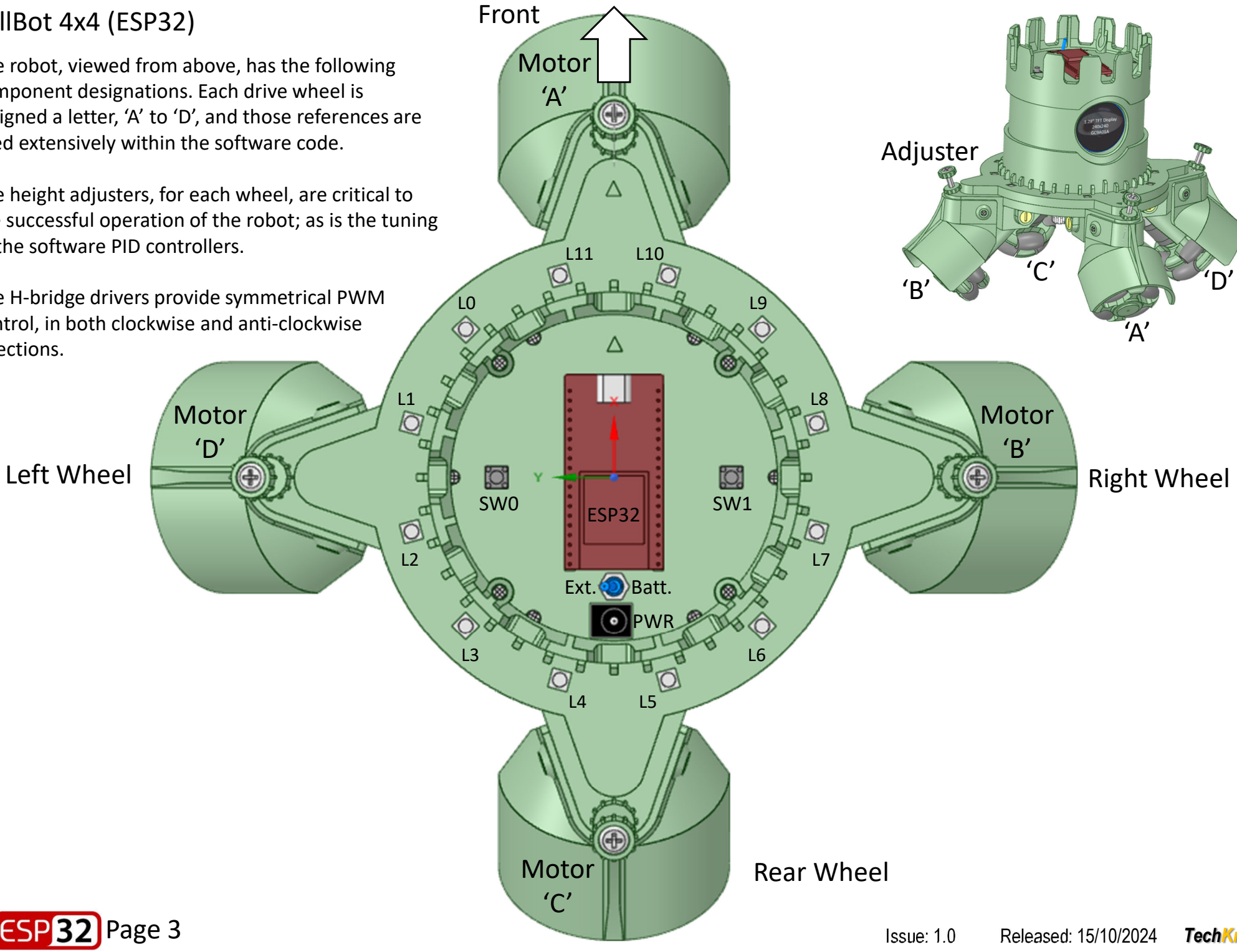
Carefully dispose of batteries that have been discharged below their critical voltage.

BallBot 4x4 (ESP32)

The robot, viewed from above, has the following component designations. Each drive wheel is assigned a letter, ‘A’ to ‘D’, and those references are used extensively within the software code.

The height adjusters, for each wheel, are critical to the successful operation of the robot; as is the tuning of the software PID controllers.

The H-bridge drivers provide symmetrical PWM control, in both clockwise and anti-clockwise directions.

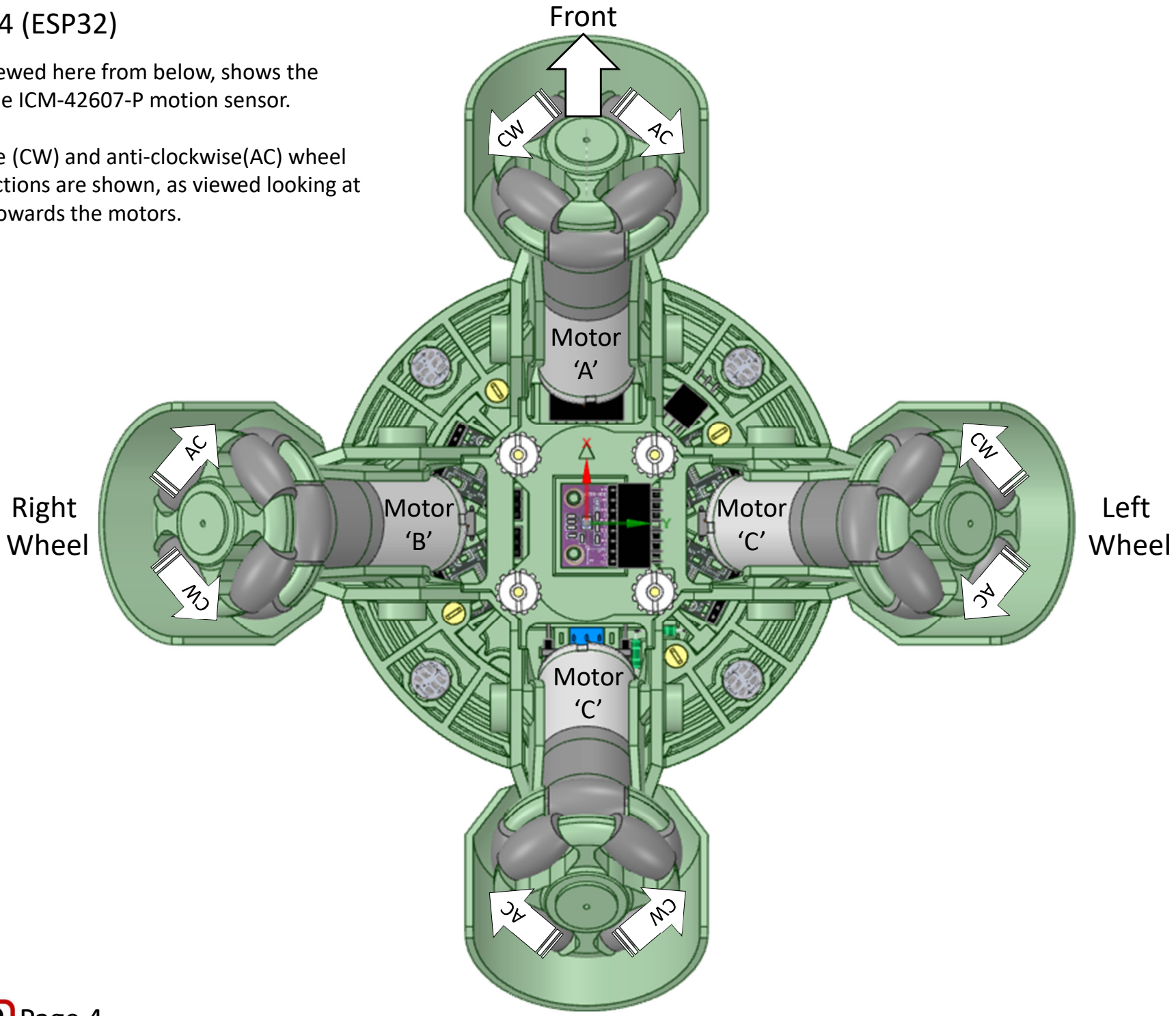


Issue: 1.0 Released: 15/10/2024 TechKnowTone

BallBot 4x4 (ESP32)

The robot, viewed here from below, shows the position of the ICM-42607-P motion sensor.

The clockwise (CW) and anti-clockwise(AC) wheel rotation directions are shown, as viewed looking at each wheel towards the motors.



Issue: 1.0 Released: 15/10/2024 TechKnowTone

BallBot 4x4 (ESP32) Wheel Adjustment

The robot's balancing performance relies heavily on the accurate setting of the omni wheels. These special wheels are made up from two wheels, positioned next to each other, each containing four rollers. The wheels are offset, such that the rollers aren't opposite each other, but the length is such that they overlap, and effectively form a continuous rubber traction wheel. The BallBot weights in at 1.01kg and was designed for a ball of 20.5cm diameter. But the motor adjusters do allow for some variation in ball size.

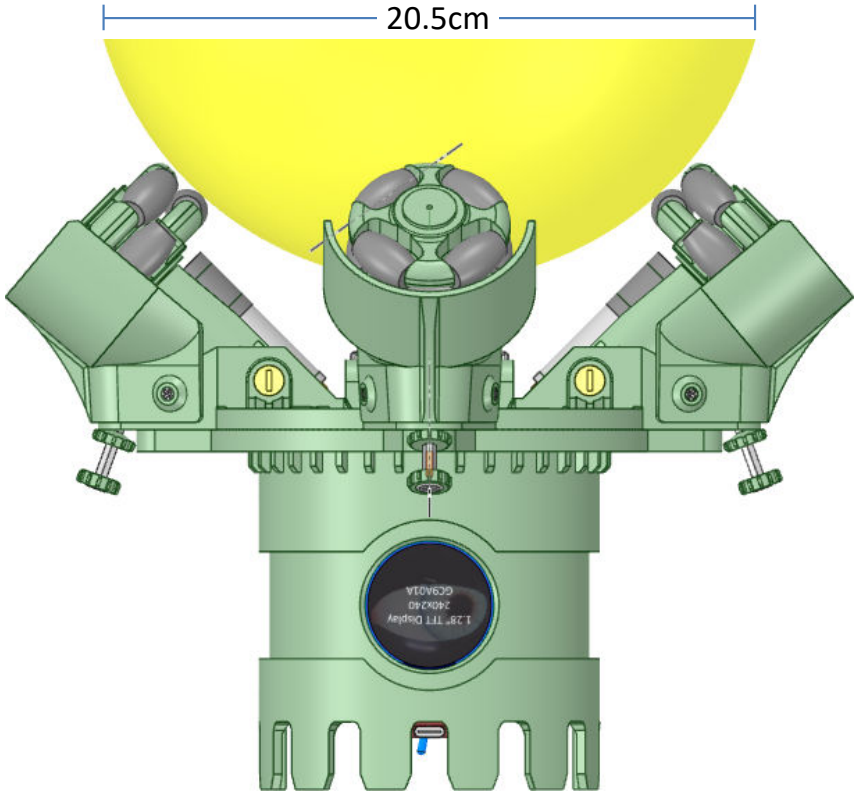
These wheels must be at right angles to the surface they touch; in this case the ball. The adjusting screws allow you to set the angle of each motor, relative to its pivot point. Start by adjusting opposite wheel angles, to achieve the right-angled touch, leaving a gap where the is no roller.

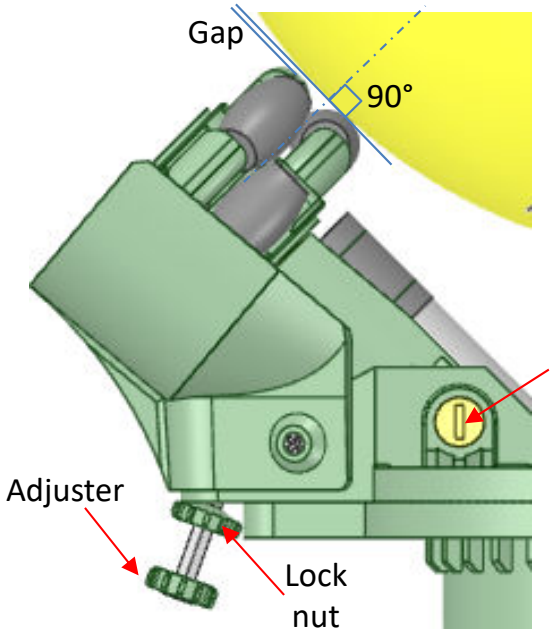
Check that the ball is firmly touching all four wheels, and that it does not rock between them. If it does, make further adjustments to remove the gap that is causing the rocking effect.

Once satisfied, you can turn the robot upside down, sitting on its crown, and run the auto-mated motor drive demonstration. This will give you a good idea how good your adjustments are.

Bear in mind that the ball is likely to be much lighter than the robot, and may not be perfectly cylindrical, when pumped up hard. So, when it is on the motor test there may still be some slippage.

It is important that you set these four wheels, and lock off the adjusters, before attempting to tune the PID controllers. If a wheel slips or drags, the robot will fall off the ball and crash.





Rubber roller touches, but omni wheel gap does not.

Issue: 1.0 Released: 15/10/2024 TechKnowTone

BallBot 4x4 ICM-42607-P Orientation

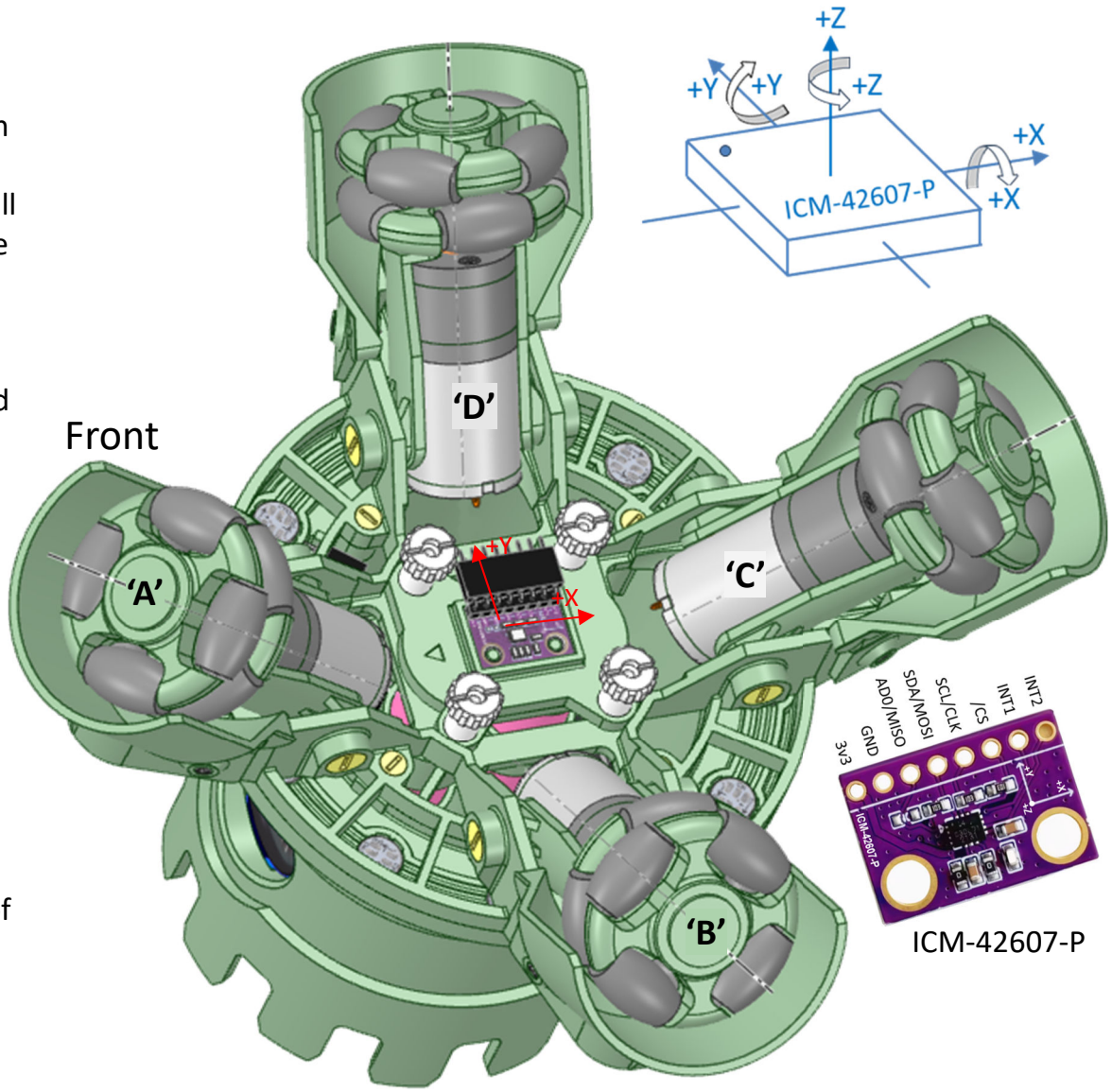
Given the way the ICM device is mounted, at the base of the BallBot, we can deduce the following relationships:

X-axis is horizontal, feeding front to back, through the robot. When the robot tilts forwards, the X-accelerometer pitch will read positive, due to gravity. When it tilts backwards the value will become negative. The Y-gyro pitch rate value will be positive when the robot tilts forwards, and negative backwards.

Y-axis is horizontal, feeding right to left, through the robot. When the robot rolls to the right, the Y-accelerometer roll will read positive, and negative when rolling towards the left. The X-gyro roll rate value will be negative when the robot rolls to the right, and positive when rolling to the left.

Z-axis is vertical, through the centre of the robot. When the robot is upright, the Z accelerometer will read negative, 1g due to gravity, and positive when the robot is upside down. When turning to the right, the Z-gyro yaw rate value will be positive, and negative when turning to the left.

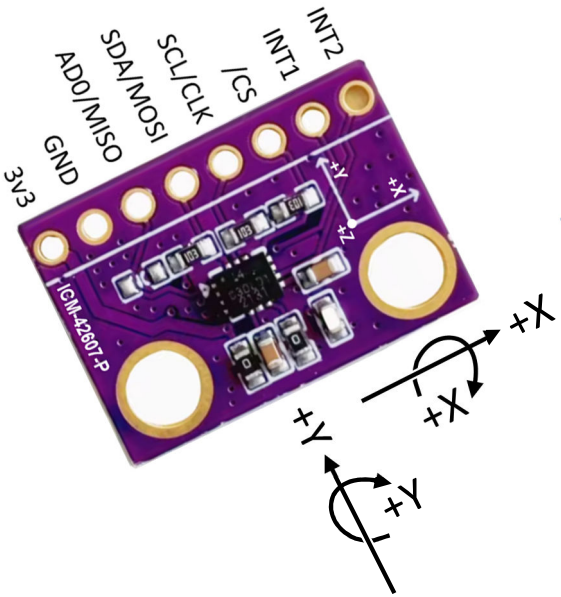
Note that the ICM-42607-P is quite accurate, but it will have small accelerometer offsets, due to manufacturing tolerances and the way in which it is mounted, and the gyros will also have a small amount of rate drift when stationary. Where possible, these errors are determined manually and removed in the code definitions, to improve the accuracy of the control system.

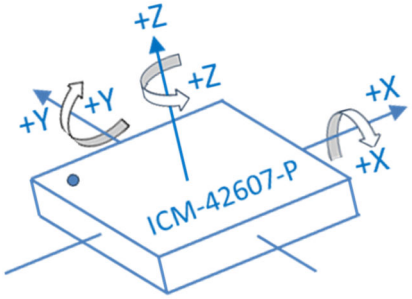


Robot viewed from below

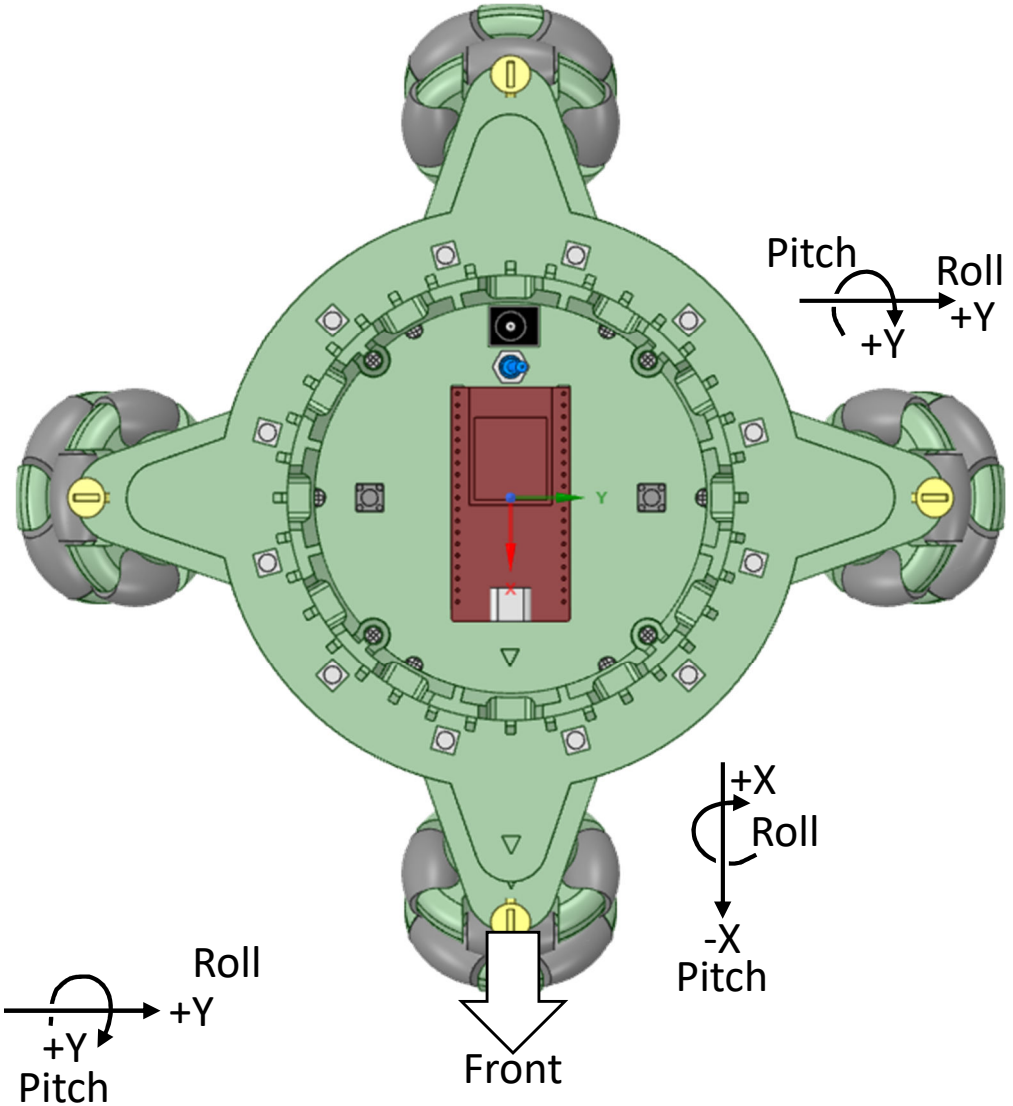
For example both the X and Y accelerometers should read zero when the robot is stood on its stand, however the surface upon which it is stood, could be at a small angle. You should therefore take readings at intervals, whilst progressively rotating the robot. Then take their mean values, as the true offset of the sensor.

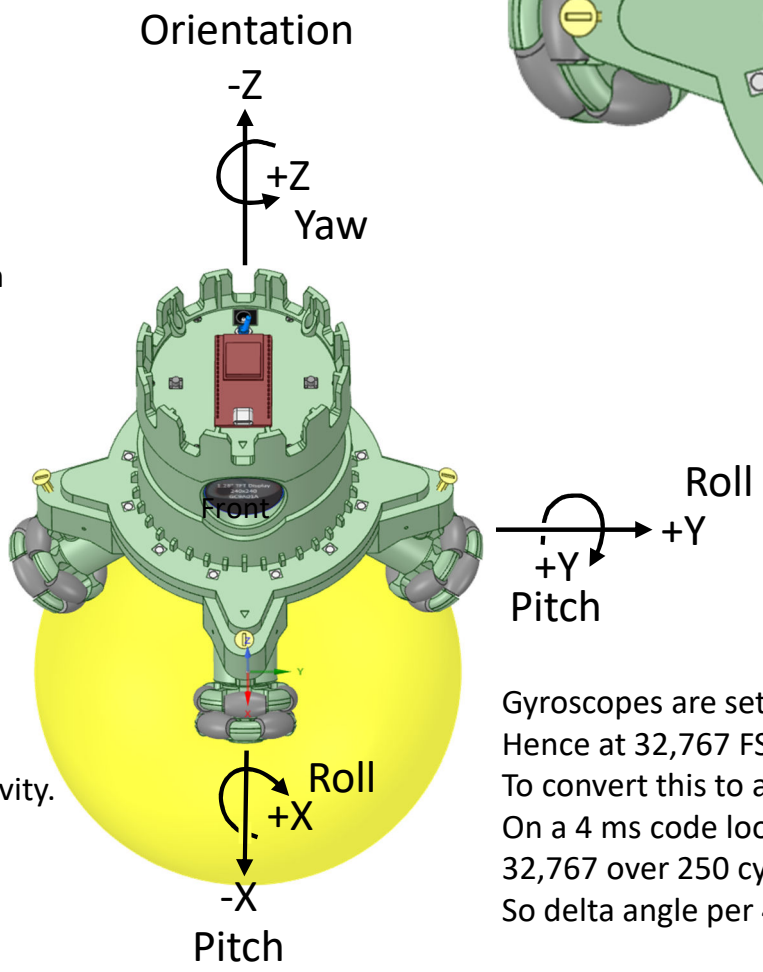
ICM-42607-P Sensor Orientation

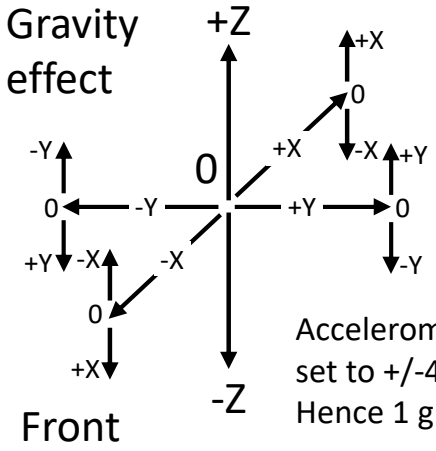




Note: BallBot sensor is mounted upside down



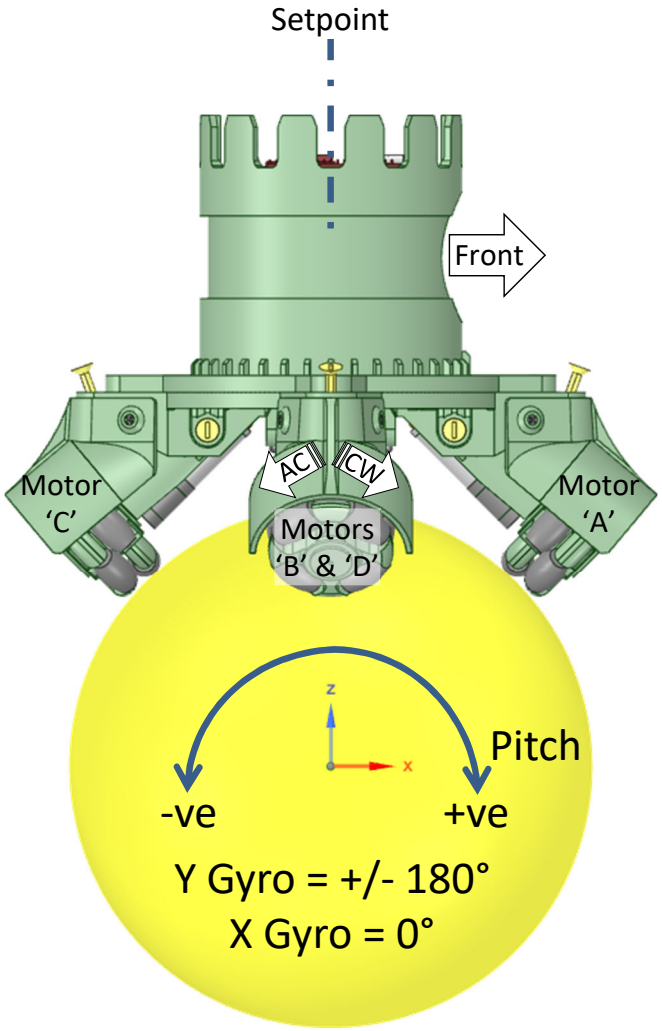


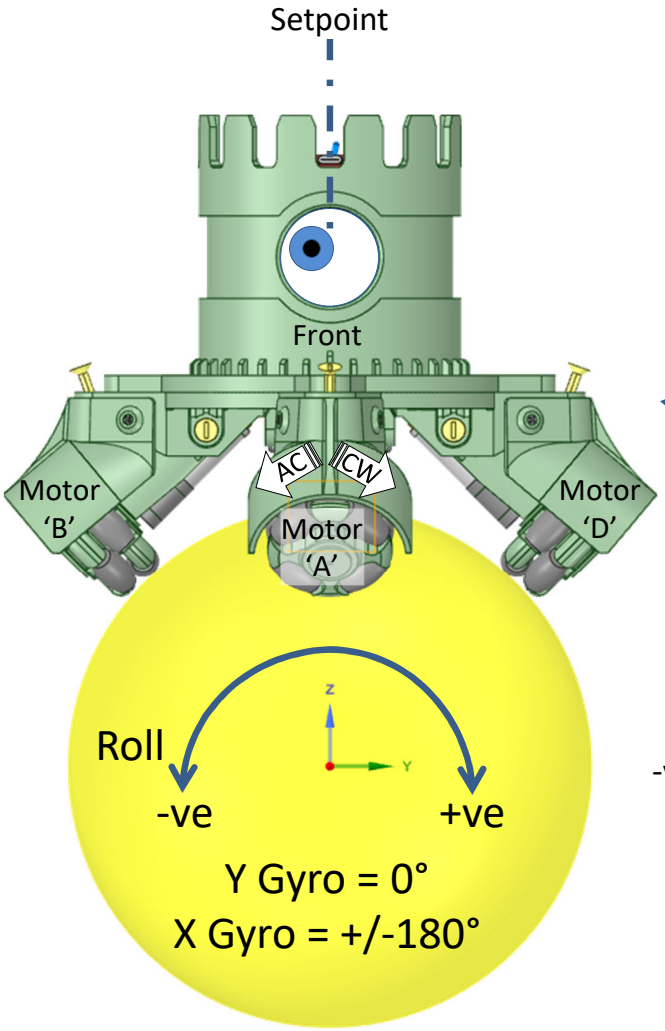


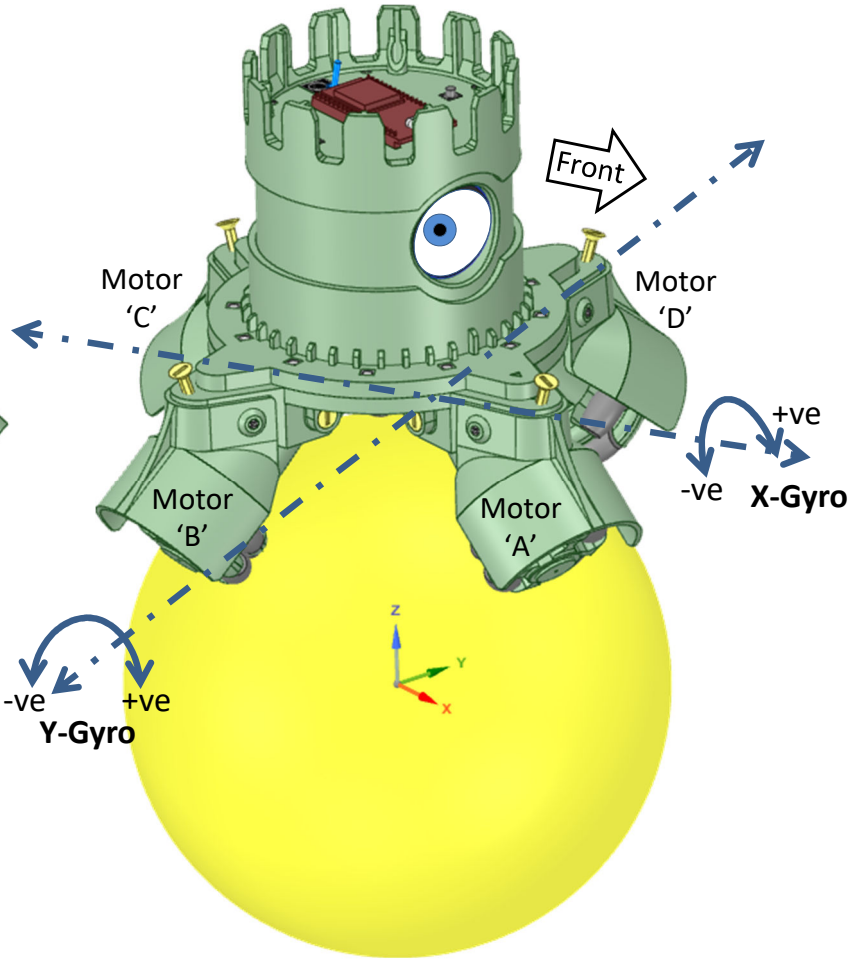
Accelerometers
set to +/-4g fsd sensitivity.
Hence 1 g == 8192

Gyroscopes are set at +/-250 °/sec FSD.
Hence at 32,767 FSD; rotation of 1 °/sec = 131.068
To convert this to a gyro angle we use the time between readings.
On a 4 ms code loop cycle, we would accumulate a count of
32,767 over 250 cycles when rotating at 1 °/sec.
So delta angle per 4 ms cycle = gyro rate * 0.0000305

Issue: 1.0 Released: 15/10/2024 TechKnowTone







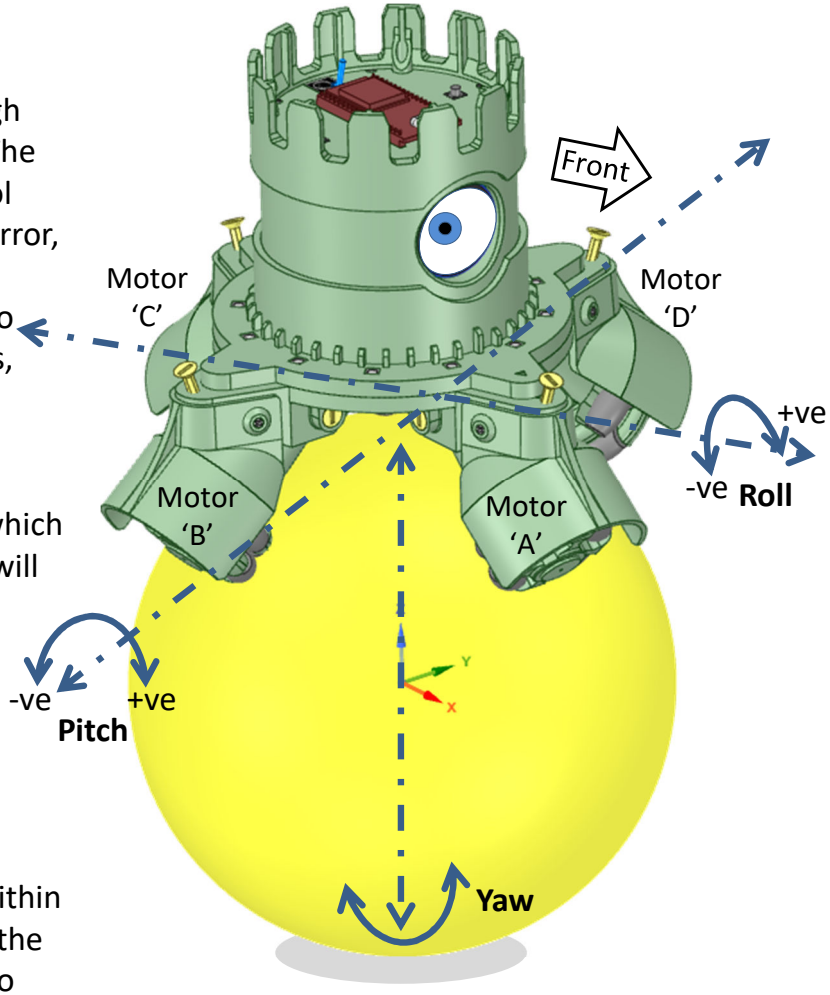
Gyro X & Y angles to motor clockwise drive relationships:

Motor A^{CW} = X Gyro (Roll)
Motor B^{CW} = Y Gyro (Pitch)
Motor C^{CW} = -X Gyro (Roll)
Motor D^{CW} = -Y Gyro (Pitch)

Note: here we are calling the Pitch gyro the Y Gyro, and Roll gyro the X Gyro, to be consistent with the accelerometer values, which are used in the code for gyro drift correction.

PID Controller Tuning

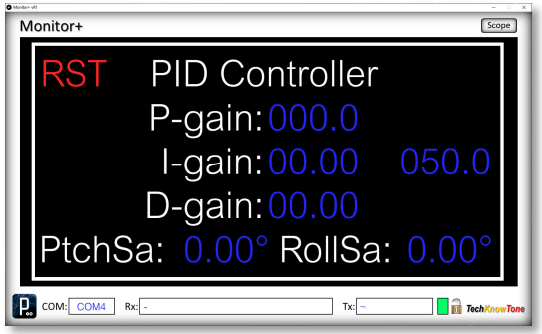
The ICM-42607-P motion sensor enables the BallBot to self-balance by providing data, at a high rate (250Hz), which is converted into a vertical angle and compared with a setpoint variable. The resulting difference (error) is used by the PID controllers to drive the four wheels. If the control system senses that the robot is falling forwards (Pitch), it will drive forwards to minimise the error, and similarly backwards, if it senses that it is falling backwards. The same principal applies to sideways (Roll) movements. There are in effect two identical PID controllers, which both aim to maintain equilibrium about the setpoint angle. Both PID controllers use the same gain settings, but the balancing setpoints are likely to be different, due to physical effects and variations.



Each PID controller applies three gain settings to the angular Pitch and Roll error signals. A **P**roportional gain, which simply multiplies the error signals by a gain factor. An **I**ntegral gain, which effectively accumulates small errors, such that if the error is small near the setpoint target, it will become larger quite quickly over time. A **D**erivative gain, which in effect provides a response braking function, to avoid overshoot and instability, resulting from the other two gains.

To tune the BallBot's PID controller, run the Monitor+ app with your PC connected to the Wii transceiver serial port. This enables the PID controller app to send control messages directly to your BallBot over Wi-Fi in a hands-off fashion. Start with all PID gain variables set to zero.

[1] Initiate the balance state by placing the robot on top of the ball, in a vertical state. If it is within +/-5° when you press SW1, the LEDs will turn yellow, entering the SafeMode 1 state. Then tilt the robot out of the 5° vertical, to find the spirit level SafeMode 2 state. Then use the LED lamps to find the vertical position, and the ACTIVE state, SafeMode 4 will be reached. With all PID variables at zero, the motors should not respond. [2] now slowly increase the P-gain value, whilst physically moving the robot forwards and backwards. You should start to feel drive from the motors assisting your movements. [3] increasing the P-gain will take you to a point where the robot almost self-balances, but more gain beyond that point makes the whole system go unstable. [4] repeat, and note the highest P-gain value possible, just before instability kicks in. Then back off the P-gain by about 33%. [5] now with P-gain set, slowly increase the I-gain value. You should find a point at which self-balance occurs; but again, increasing I-gain further will cause instability. Note the best I-gain value for self balance. The D-gain adjustment is to prevent overshoot and instability.



Monitor+ app

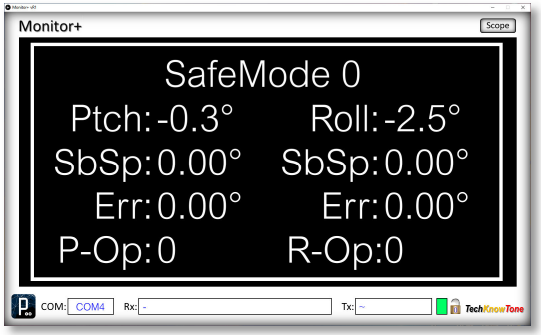
Issue: 1.0 Released: 15/10/2024 TechKnowTone

PID Controller Tuning Continued

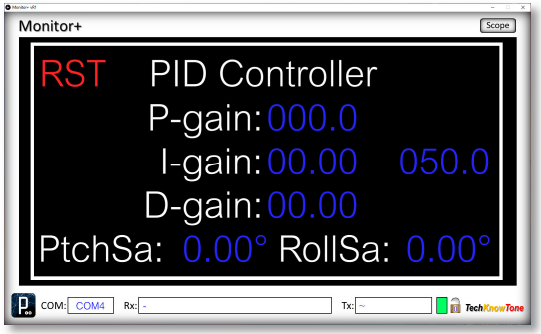
[6] you can now bring the D-gain variable into play. Increasing D-gain should allow you to have higher values of I-gain and P-gain, before instability occurs. Higher values should mean that the robot effectively stiffens up about the setpoint angle, and doesn't wander or vibrate at that point.

The PID gain values are applied equally to both the Pitch and Roll controllers, as they are symmetrical systems; albeit operating on different axis. This tuning method is empirical, as all robots will have different setpoints, and there is no steadfast way of arriving at the PID gain values. Once you have determined the three gain numbers, they can then be entered into the C++ code in the table of definitions.

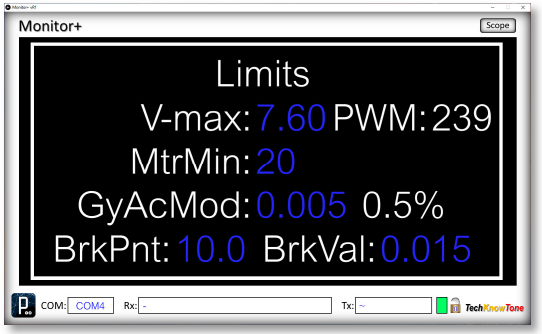
The Monitor+ SafeMode display shows the Pitch and Roll angles, their respective self-balancing setpoints their angular error values, and their PID outputs which are applied to the motors. Note that the self-balancing setpoints are derived automatically whilst the robot is moving, to ensure balance. Once you get your robot balancing in a reliable fashion, you can look at the self-balancing setpoints. You use these two values as start angles for your robot, as this will make it much easier to initiate balance, and avoid the robot from lurching at the start.



The Monitor+ display screens that have coloured text, indicates that the text fields can be clicked to invoke an action. Clicking on the **RST** field will reset all of the values to the defaults defined in the code. Whilst clicking on **blue** fields will change their values, with immediate effect. Each digit can be adjusted via the left/right mouse button. The field to the right of the I-gain value is the Imax limit. Initially set to **050.0**, to make the robot less lively; and later set it to **255.0** to improve overall stability, particularly when driving along.



The Limits display provides the ability to adjust V-max, which is the nominal battery voltage. If the battery voltage exceeds this limit, the motor PWM values are reduced proportionately, so that the motor does not provide more power than intended. Due to motor gearbox friction, there will be PWM values below which the motor shafts will not turn. This dead spot can be removed by setting MtrMin to a value around 20. Avoid setting this too high. The GyAcMod value is the amount of accelerometer angle, which is used to correct the gyro angles; which has no absolute value. It needs to be enough to correct any gyro drift, but too high a value will inject vibration noise into the PID error signals. The BrkPnt and BrkVal brake values, prevent the PID controllers from being wound up, when the robot is pushed along on a slope, or by hand.



Battery Voltage Calibration

See Lithium discharge curve obtained from the internet below. In this analysis the lipo battery consists of two identical batteries connected in series.
Assume fully charged 8.2v battery max voltage is V_{BM} >= 8.4v max (charging)
Set battery warning point Bat7v2 at V_B = 7.2v (2 x 3.6v)
Set battery critical point Bat6v6 at V_{BC} = 6.6v (2 x 3.3v)

The ESP32 is powered via a 3.3v voltage regulator, connected to the 3v3 pin, but the 6k8 supply sampling resistor is connected to source V_{Batt}.
For ESP32 V_{ADC} == 4095 fsd on 12-bit converter (4095 max).
If we use a 6k8 resistor feeding A0 and a 3k3 resistor to GND, we get a conversion factor of 10.1v == 4095, or 2.47mV/bit, or 405.4 bit/v
Using a Multimeter and a variable DC supply, I determined the following V_{ADC} values for corresponding threshold voltages:

MAX: (100%) 3190 V_{ADC} = 8.2v, gave A0 = 3032 on V_{ADC} (2 x 4.1v)

HIGH: (80%) Bat7v6 V_{ADC} = 7.6v, gave A0 = 2906 on V_{ADC} (2 x 3.8v)

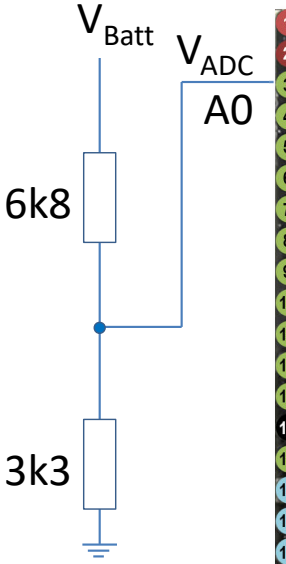
WARNING: (20%) Bat7v2 V_{ADC} = 7.2v, gives A0 = 2734 on V_{ADC} (2 x 3.6v)

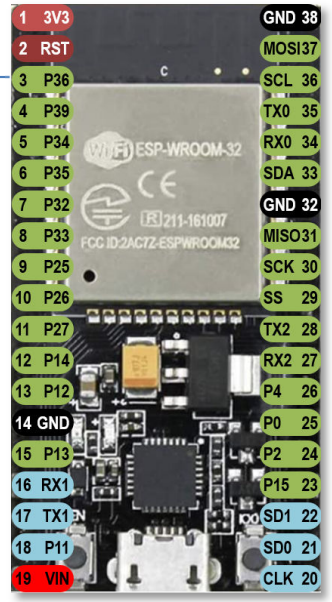
CRITICAL: (0%) Bat6v6 V_{ADC} = 6.6.0v, gives A0 = 2482 on V_{ADC} (2 x 3.3v)

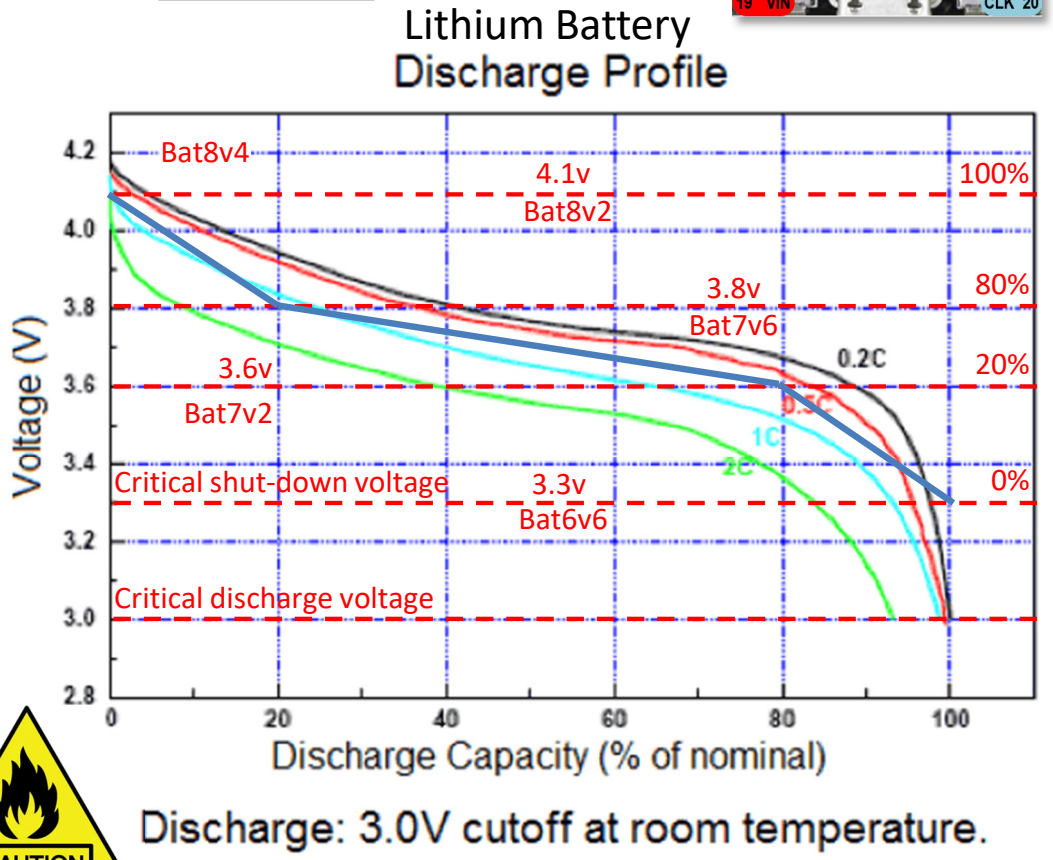
Run the code in TEST mode. The code will sample the battery voltage on power-up to ensure it is sufficient, then at every 8ms interval, calculating an average (1/50) to remove noise. It also detects no battery as being USB mode.

Note: If connected to USB port with internal battery switched OFF the ADC will read a value of A0 = 1500, or less. So, if the micro starts with such a low reading it assumes that it is on USB power, and this will limit its behaviour.

It also detects being raised above USB mode, if for example an external supply is connected to the DC power socket.







Discharge: 3.0V cutoff at room temperature.

Issue: 1.0 Released: 15/10/2024 TechKnowTone