# CAN Protocol for Wheele

As suggested by Mike, we have implemented a CAN architecture on Wheele.

For now, every data number is simply stored as 2 bytes: 0x0000 to 0xFFFF, which represents -32768 to 32767

In the code, the raw data is stored as int16. It gets manually converted to uint16 by adding 32768 before sent on CAN.

So when reading a CAN msg, data\_number = decimal(0xhhhh) - 32768

Currently, these are the hardware nodes:

* The top arduino board that replaces part of  the bot board
  + **Transmits CAN ID 0x101: raw RC** speed and steer commands from right joystick (left joystick to be added)  
    Data: Speed\_High\_Byte  Speed\_Low\_Byte  Steer\_HB  Steer\_LB  Left\_JS\_Hrz\_HB LB Left\_JS\_Vert\_HB LB  
               A1 B1  A2 B2  A3 B3  A4 B4
  + **Transmits CAN ID 0x131: gyro data** (+/-32767 **centi-deg/sec**)  
    GX\_H GX\_L GY\_H GY\_L GZ\_H GZ\_L 00 00
  + **Transmits CAN ID 0x132: accel data** (+/-32767 mm/sec^2 ?)  
    aX\_H aX\_L aY\_H aY\_L aZ\_H aZ\_L 00 00
  + **Transmits CAN ID 0x133: bno055 heading** (+/- 32767 centi-deg) and mag data (uTx100)  
    yawH yawL magxH magxL magyH magyL magzH magzL
  + **Receives no CAN IDs**

AN EXAMPLE candump can0 from the top and bottom arduinos, broadcast:  
  can0  101   [8]  85 4D 85 60 00 00 00 00  
  can0  105   [8]  2E 7E 35 2F 00 00 00 00  
  can0  121   [8]  80 00 7F FF 00 00 00 00

* The main pi controller with ROS
  + **Transmits CAN ID 0x301: command velocity**
    - Bytes 0-1, velocity in mm/sec
    - Bytes 2-3, yaw rate in deci degrees (1/10 deg)
    - Bytes 4-7, 0x00
  + **Receives CAN IDs** 0x101, 0x131, 0x132, 0x133, 0x105, 0x121, and 0x140
* Bottom ESC/Servo controller micro that replaces the Micro Maestro and part of the bot board
  + **Transmits CAN ID 0x105: encoder counts**  
    Data: Left\_Enc\_High\_Byte Low\_Byte    Right\_Enc\_High\_Byte Low\_Byte  
              A1 B1  A2 B2  00 00 00 00  
    The encoder counts are stored as int16 and will change from 32767 to -32768 or vice versa  
    The pi will handle this when it sees a sign change in the raw encoder counts
  + **Transmits CAN ID 0x121: bumper switch (probably add close range proximity (sonar?) sensors)**  
    Data: Bump\_HB LB  
              80 01  80 00 00 00 00 00 (not hit)  
              80 00  80 00 00 00 00 00 (bumper hit)
  + **Transmits CAN ID 0x140: Battery Power**.  The battery voltage and current statistics will be continuously gathered, and returned periodically.  The mean, min, and max values are the statistics since the last Battery Power packet.
    - Bytes 0-1, battery voltage (mean) in milliVolts
    - Bytes 2-3, battery current (mean) in milliAmps
    - Bytes 4-5, battery current (min) in milliAmps
    - Bytes 6-7, battery current (max) in milliAmps
  + **Receives CAN IDs** 0x101 and 0x301