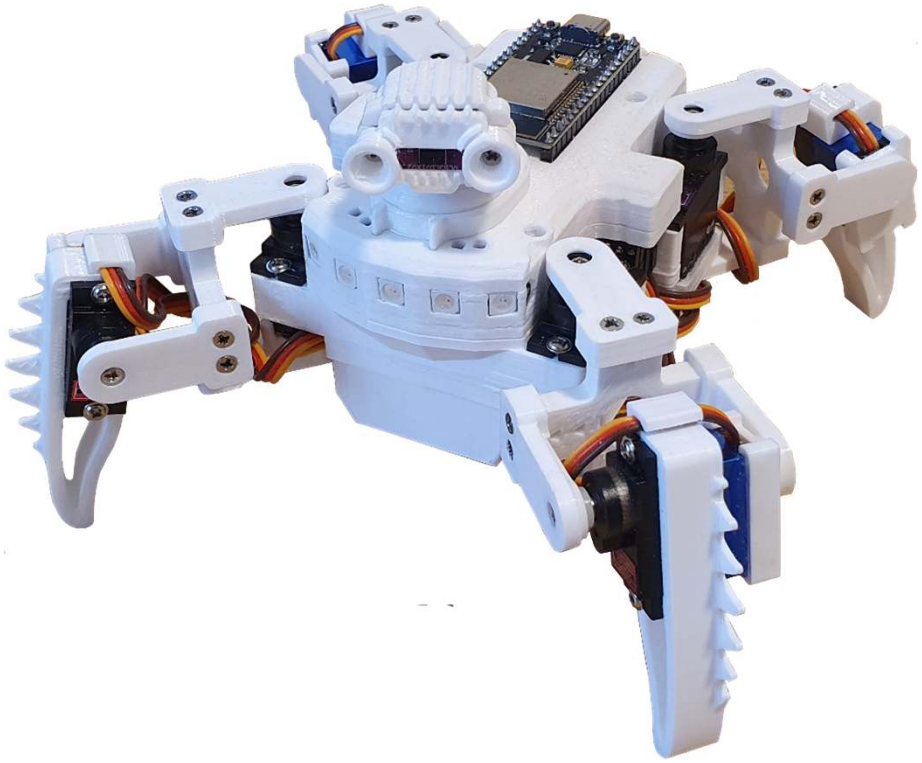
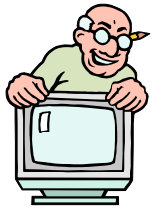
QuadAuto MKII (ESP32)

Project Overview Slides

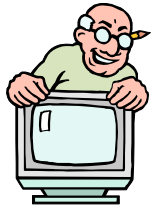




Slides that aim to explain key features of this project.

Issue: 1.5 Released: 24/03/2026 TechKnowTone

Take Care



It's important to respect the use of this technology.

This design does not include any form of circuit protection, so sloppy wiring, resulting in short circuits, can pose a major hazard.

It does offer good practise, and instructions that should be adhered to. Like the use of a multimeter to check for wiring faults, and the use of a good quality battery charger.

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.

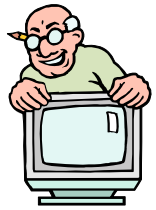
Battery care & maintenance: Stop using a battery if it is swollen, damaged, dented or leaking. Never charge a damaged battery. Never allow a Lithium battery to discharge below 3.2 volts, as cell damage will occur. Avoid extreme temperatures. Do not charge or store batteries in very hot or cold environments. 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 damaged, or discharged below the critical voltage.

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.

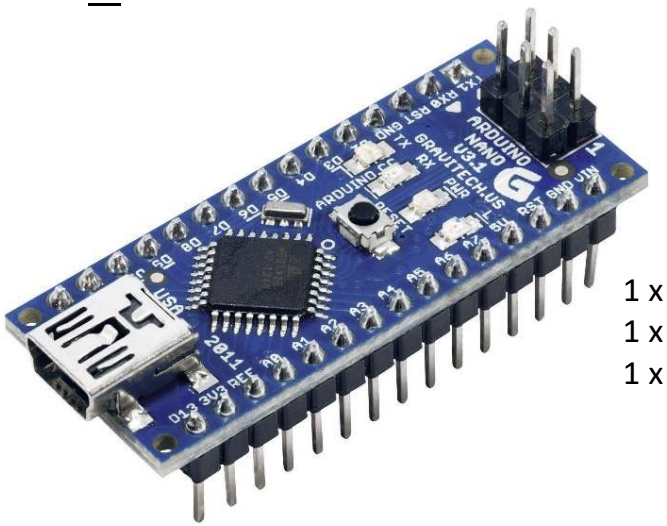
Battery care & maintenance: Stop using a battery if it is swollen, damaged, dented or leaking. Never charge a damaged battery. Never allow a Lithium battery to discharge below 3.2 volts, as cell damage will occur. Avoid extreme temperatures. Do not charge or store batteries in very hot or cold environments. 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.

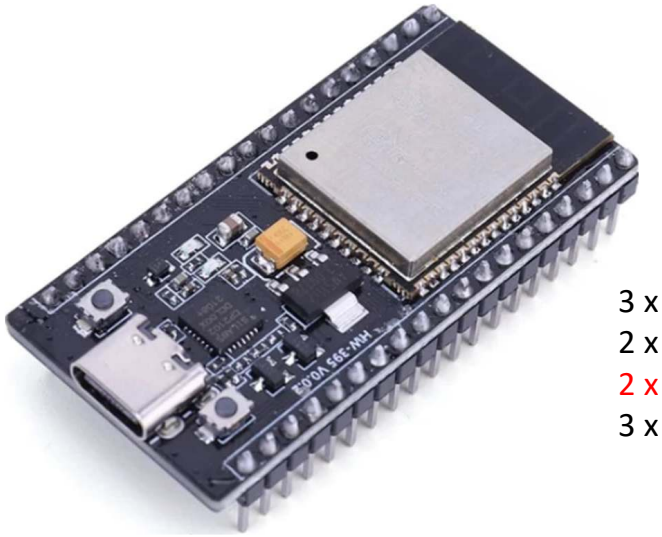
Built-in Monitoring: Most of my project designs include code, and circuitry, to monitor battery voltage, whilst in use. This code then seeks to alert the operator, when the battery has reached a critical low voltage, before shutting down power consuming circuitry; including the micro. Time should therefore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

Carefully dispose of batteries that damaged, or discharged below the critical voltage.

ARDUINO NANO v ESP32-WROOM-32



1 x UART
1 x I2C
1 x SPI

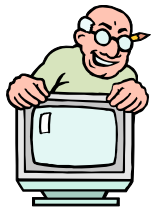


3 x UART
2 x I2C
2 x I2S
3 x SPI

Microcontroller:	ATmega328P (8-bit)	
Clock Speed:	16 MHz	
Operating Voltage:	5.0 v	
VIN:	7 – 12v	
Digital I/O:	14 (6 PWM)	
Analog Inputs:	8 (10-bit ADC)	
Flash Memory:	32 KB (2KB bootloader)	
RAM:	2 KB	
EEPROM:	1 KB	
WiFi:	None	

Microcontroller:	Dual core LX6 (32-bit)	**2 x 4 x**
Clock Speed:	240 MHz	**15 x**
Operating Voltage:	2.2 - 3.6 v	
VIN:	5v	
Digital I/O:	22 (16 PWM) + 4 input only	
Analog Inputs:	18 (12-bit, 2 x ADC), 10 touch	**2 x**
Flash Memory:	4 MB (inc. bootloader)	**128 x**
RAM:	520 KB	**260 x**
EEPROM:	512 B	**0.5 x**
WiFi:	802.11 b/g/n + Bluetooth 4.2 BR/EDR/BLE	

Battery Monitoring



Batteries are based on good chemistry, which is temperature sensitive.

Therefore, good battery monitoring systems measure temperature, voltage and current.

This design takes a more simplistic approach, only relying on voltage measurement, to give an indication of battery health. We calibrate the ESP32's ADC at key points in a normalised discharge curve.

This gives us a more accurate reading of voltage, and a percentage of capacity based on the discharge curve.

Battery Voltage Calibration

See Lithium discharge curve obtained from the internet. 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 at V_{BW} = 7.2v (2 x 3.6v)
Set battery critical point at V_{BC} = 6.6v (2 x3.3v), don't go below this!

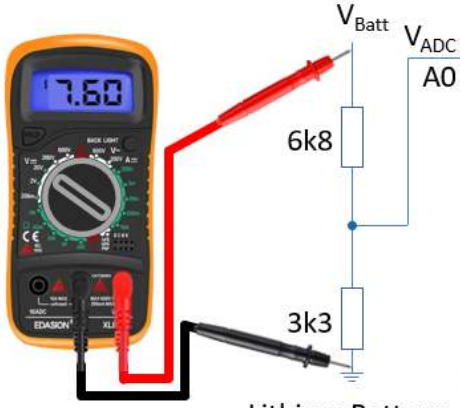
The ESP32 is powered via a 3v3 voltage regulator, connected to the 3v3 pin. But the 6k8 supply sampling resistor is connected to source V_{Batt} or Ext. supply.
For ESP32 V_{ADC} == 4095 on 12-bit converter (4095 max).
If we use a 6k8 resistor feeding A0 and a 3k3 resistor to GND, we get a conversion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

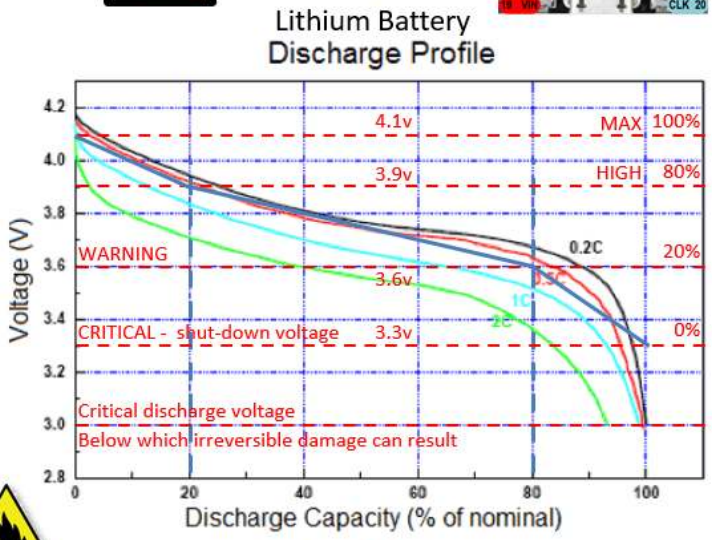
MAX. O.C	V_{OC} = 8.4v, gave A0 = 3295 On V_{ADC} (2 x 4.2v)
MAX: (100%)	V_M = 8.2v, gave A0 = 3200 on V_{ADC} (2 x 4.1v)
HIGH: (80%)	V_H = 7.8v, gave A0 = 2997 on V_{ADC} (2 x 3.9v)
WARNING: (20%)	V_{BW} = 7.2v, gives A0 = 2762 on V_{ADC} (2 x 3.6v)
CRITICAL: (0%)	V_{BC} = 6.6v, gives A0 = 2513 on V_{ADC} (2 x 3.3v)

The code will sample the battery voltage on power-up to ensure it is sufficient, then at every 40ms interval, calculating an average (1/50) to remove noise. Then converts ADC values to voltage in the getBatV() function.

In the code I have assumed a discharge curve ranging from 8.2v (100%) to 6.6v (0%) capacity, using the blue overlay line shown. The voltage is monitored and used to predict the remaining capacity of the battery in use.

Note: If connected to USB port with internal battery switched OFF the ADC will read a value 5 volts (A0 = 1919) or less. So, if the micro starts with such a low reading it knows that it is on USB power, which limits functions available.

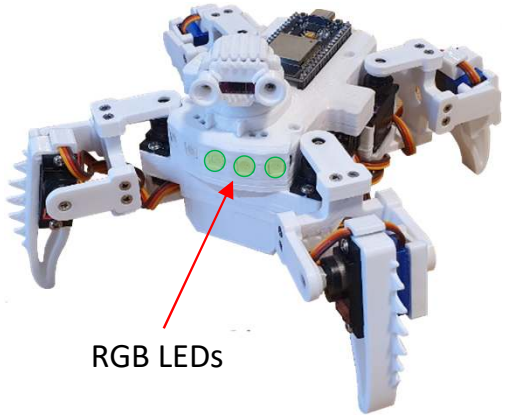




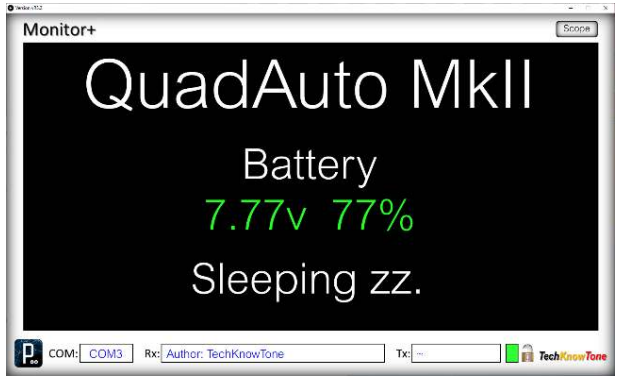
Discharge: 3.0V cutoff at room temperature.

CAUTION LITHIUM BATTERY

Issue: 1.3 Released: 13/02/2026 TechKnowTone



When powered up the Quadruped indicates battery health, in both its RGB LEDs (colour and number), and through the Monitor+ app, when connected. The Calibration document explains how to measure and enter values in your code.



Issue: 1.5 Released: 24/03/2026 TechKnowTone

Battery Voltage Calibration

See Lithium discharge curve obtained from the internet. 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 at V_{BW} = 7.2v (2 x 3.6v)
Set battery critical point at V_{BC} = 6.6v (2 x3.3v), don't go below this!

The ESP32 is powered via a 3v3 voltage regulator, connected to the 3v3 pin. But the 6k8 supply sampling resistor is connected to source V_{Batt} or Ext. supply.
For ESP32 V_{ADC} == 4095 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
Place the droid in TEST mode. Using a Multimeter and a variable DC supply, determine the following V_{ADC} values for corresponding threshold voltages:

MAX. O.C V_{OC} = 8.4v, gave A0 = 3295 On V_{ADC} (2 x 4.2v)

MAX: (100%) V_M = 8.2v, gave A0 = 3200 on V_{ADC} (2 x 4.1v)

HIGH: (80%) V_H = 7.8v, gave A0 = 2997 on V_{ADC} (2 x 3.9v)

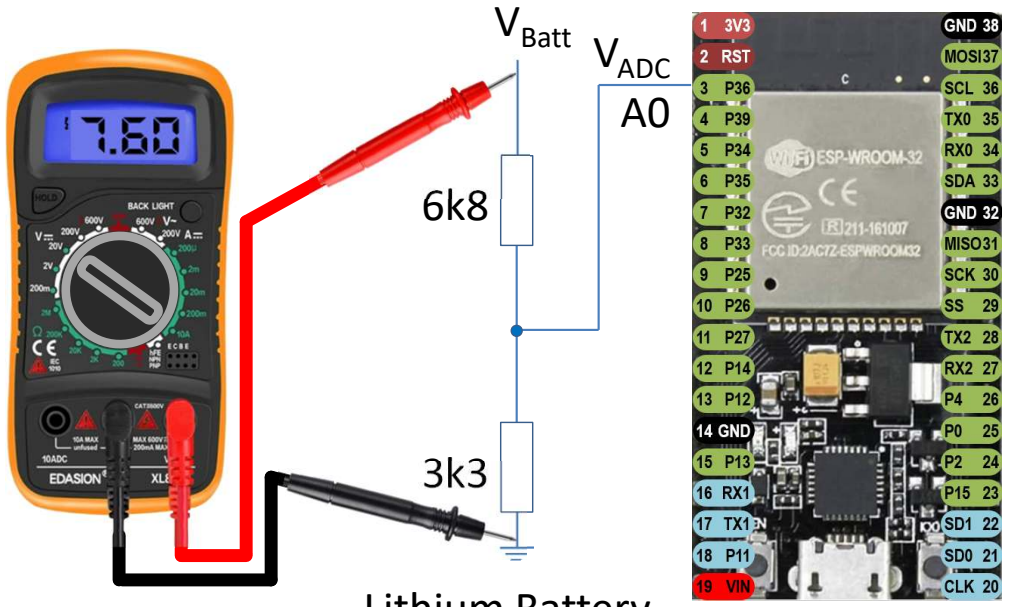
WARNING: (20%) V_{BW} = 7.2v, gives A0 = 2762 on V_{ADC} (2 x 3.6v)

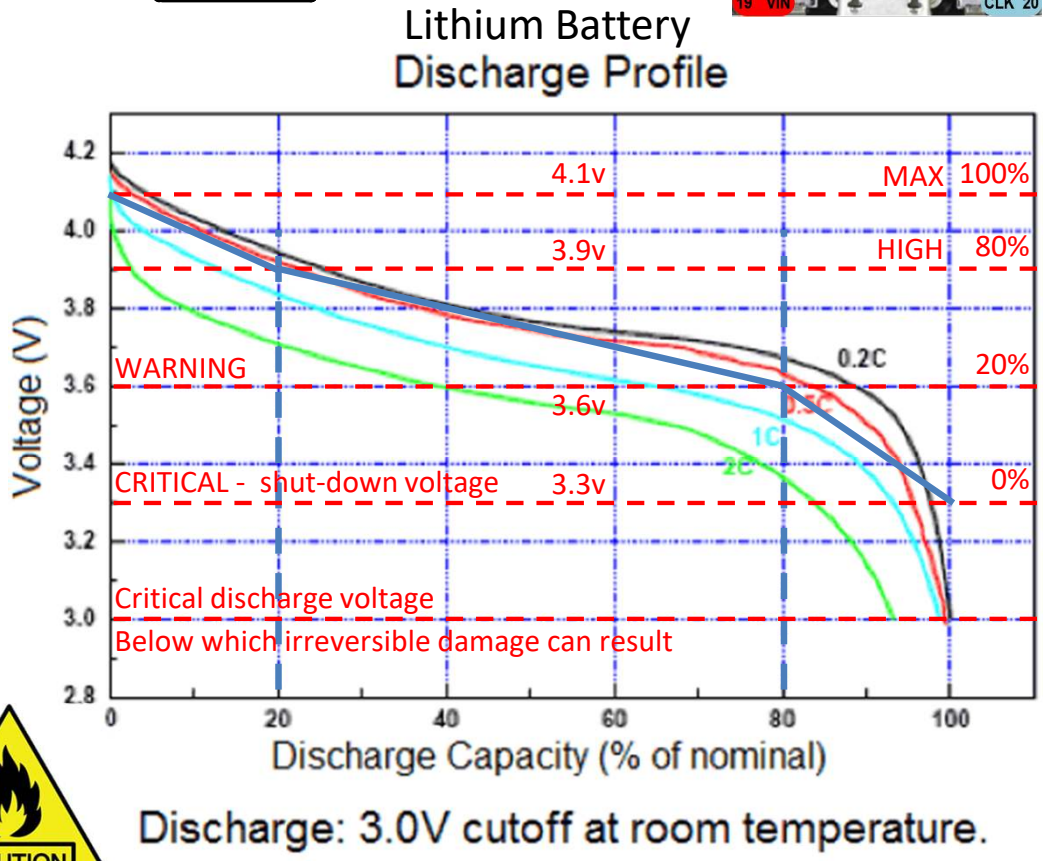
CRITICAL: (0%) V_{BC} = 6.6v, gives A0 = 2513 on V_{ADC} (2 x 3.3v)

The code will sample the battery voltage on power-up to ensure it is sufficient, then at every 40ms interval, calculating an average (1/50) to remove noise. Then converts ADC values to voltage in the getBatV() function.

In the code I have assumed a discharge curve ranging from 8.2v (100%) to 6.6v (0%) capacity, using the blue overlay line shown. The voltage is monitored and used to predict the remaining capacity of the battery in use.

Note: If connected to USB port with internal battery switched OFF the ADC will read a value 5 volts (A0 = 1919) or less. So, if the micro starts with such a low reading it knows that it is on USB power, which limits functions available.

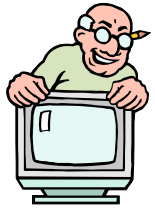




Discharge: 3.0V cutoff at room temperature.

Issue: 1.5 Released: 24/03/2026 TechKnowTone

Leg Lift Angles

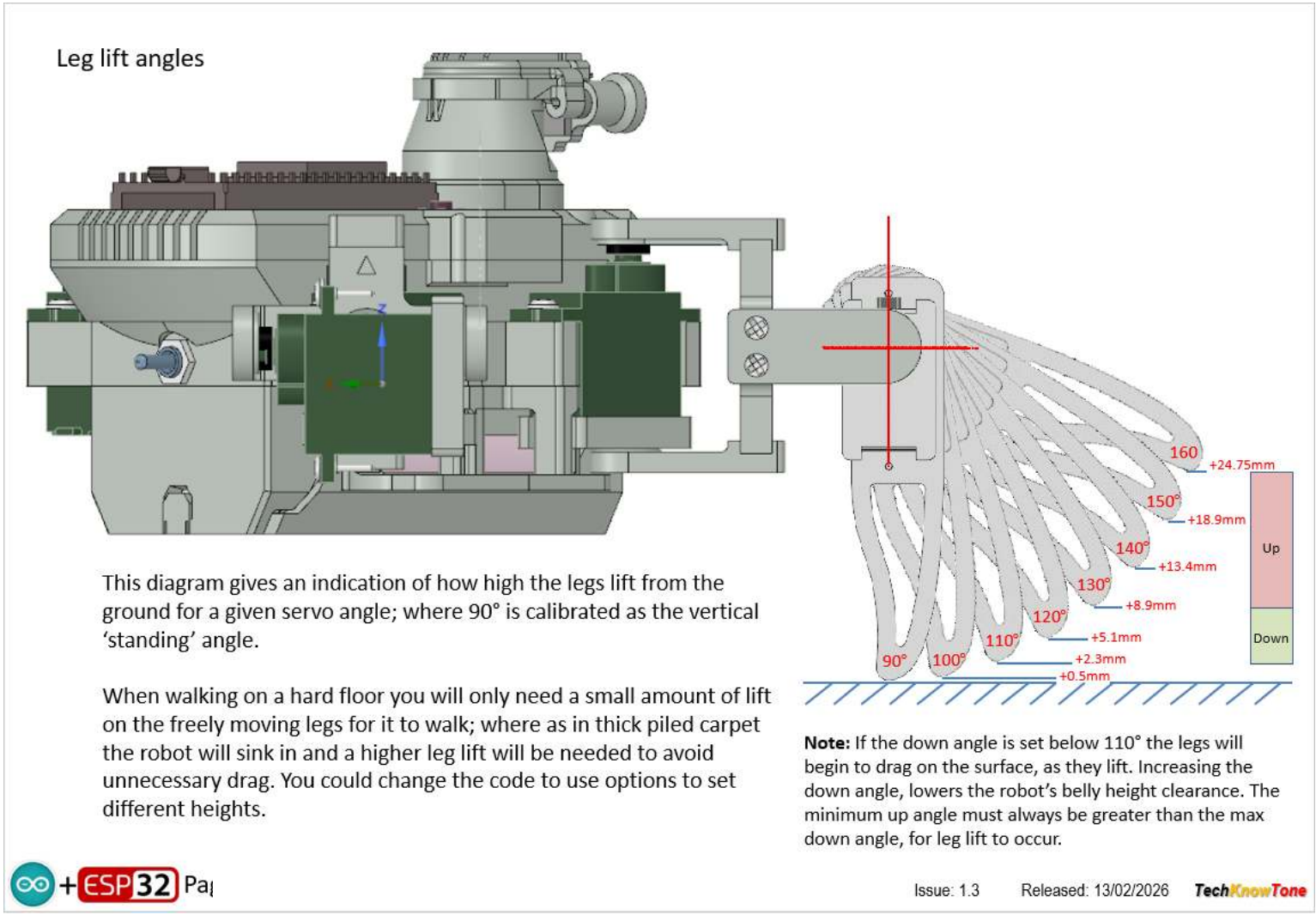


As the Quadruped only uses two servo motors in each leg, these effectively act as a shoulder joint. A leg can be rotated around the body, and swung out. But it can not be lifted, like a normal leg.

In order to walk, the robot needs to lower its body height, by swinging the legs out slightly. Then it can effectively raise them, by swinging them upwards.

On a flat surface this works fine, and there is no need to raise them too heigh when walking. But on a carpet, the legs need to be raised more. Code options, depending on the Wii controller used, allow you to raise the legs when needed, at low speed.

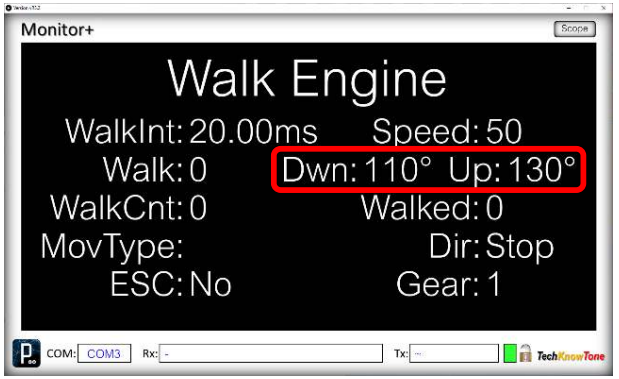
When using the Monitor+ app you can view the angles being used for leg down and leg up. For flat surfaces these are 110° and 130° respectively.



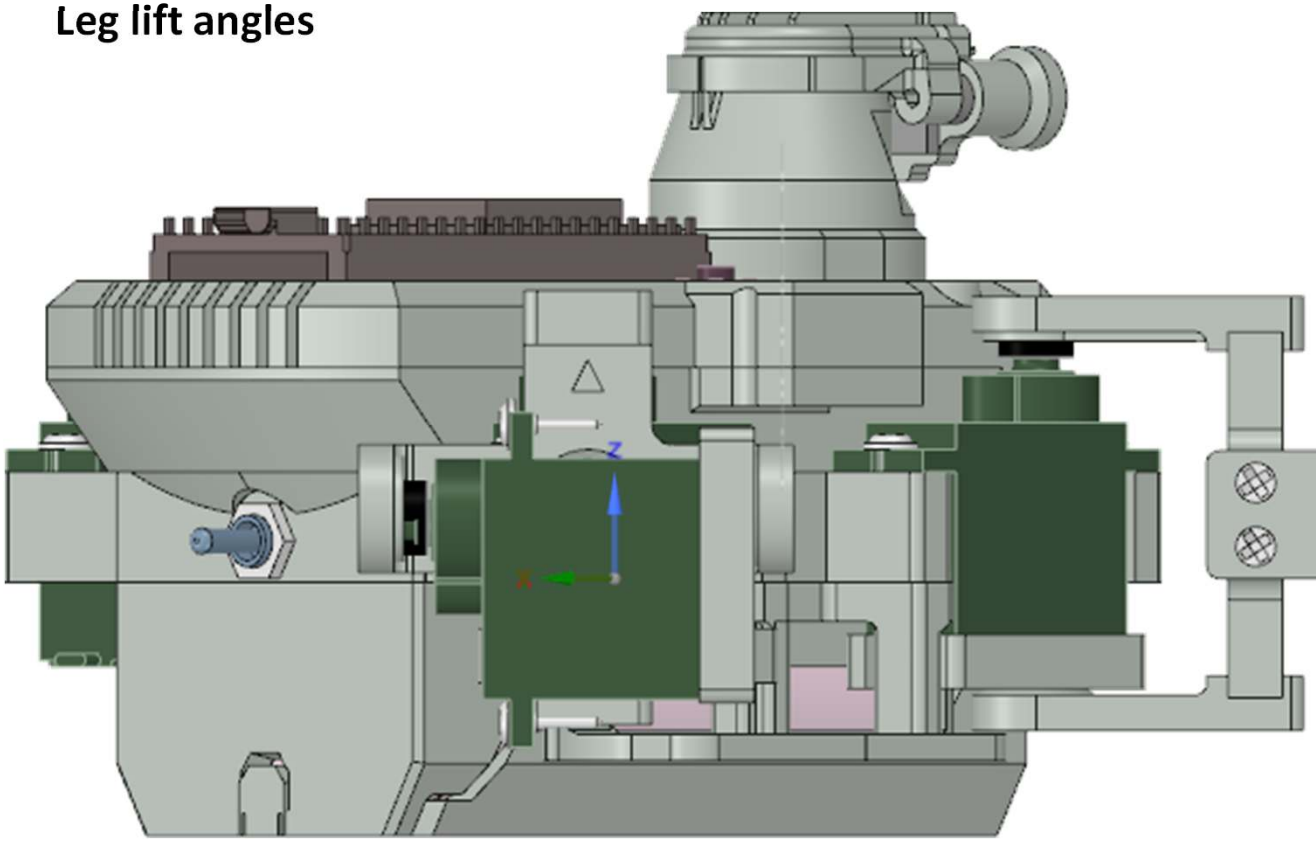
This diagram gives an indication of how high the legs lift from the ground for a given servo angle; where 90° is calibrated as the vertical 'standing' angle.

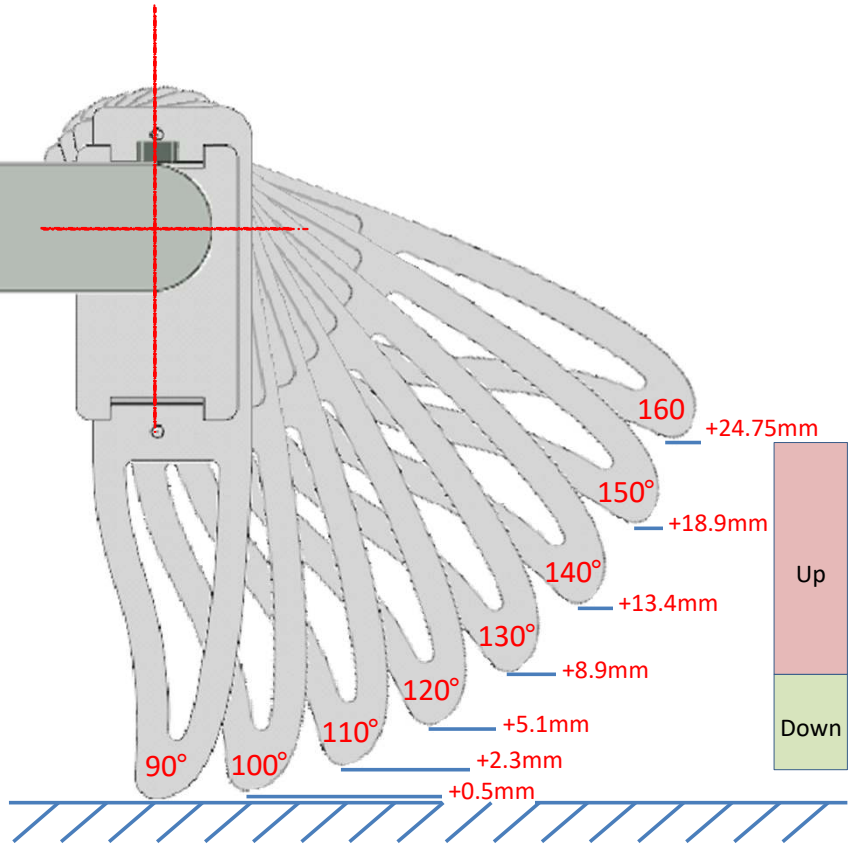
When walking on a hard floor you will only need a small amount of lift on the freely moving legs for it to walk; where as in thick piled carpet the robot will sink in and a higher leg lift will be needed to avoid unnecessary drag. You could change the code to use options to set different heights.

Note: If the down angle is set below 110° the legs will begin to drag on the surface, as they lift. Increasing the down angle, lowers the robot's belly height clearance. The minimum up angle must always be greater than the max down angle, for leg lift to occur.



Leg lift angles





This diagram gives an indication of how high the legs lift from the ground for a given servo angle; where 90° is calibrated as the vertical ‘standing’ angle.

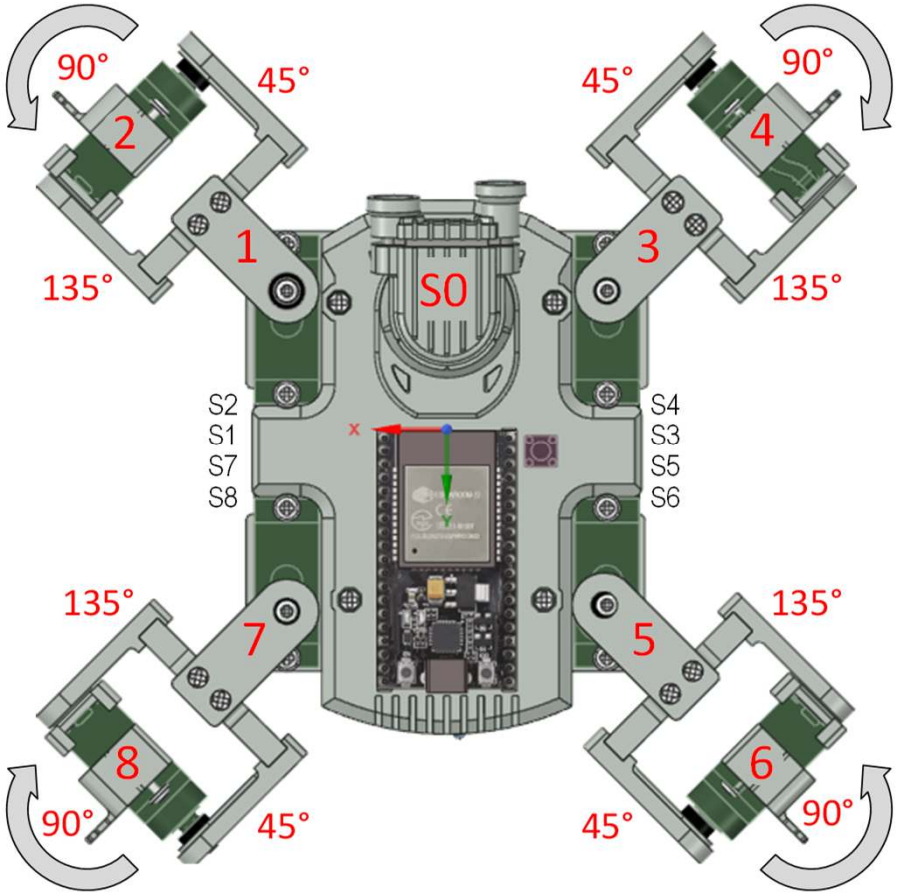
When walking on a hard floor you will only need a small amount of lift on the freely moving legs for it to walk; where as in thick piled carpet the robot will sink in and a higher leg lift will be needed to avoid unnecessary drag. You could change the code to use options to set different heights.

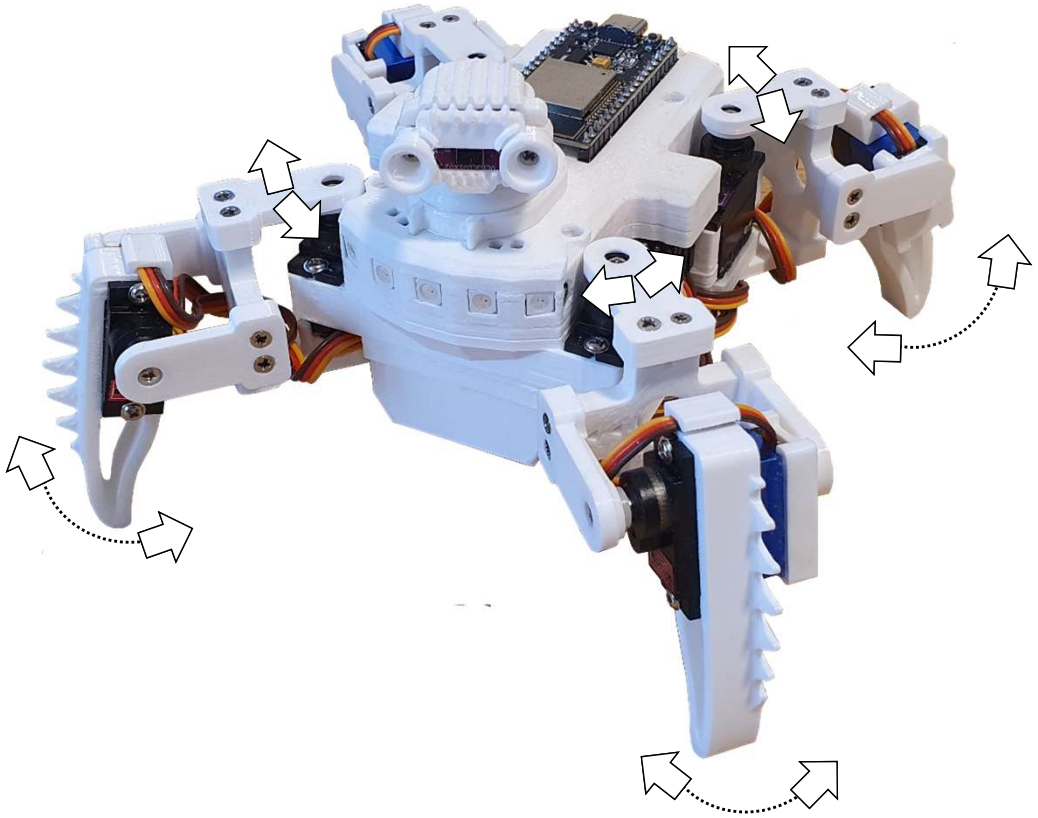
Note: If the down angle is set below 110° the legs will begin to drag on the surface, as they lift. Increasing the down angle, lowers the robot’s belly height clearance. The minimum up angle must always be greater than the max down angle, for leg lift to occur.

Walking Basics - Rules

Rules

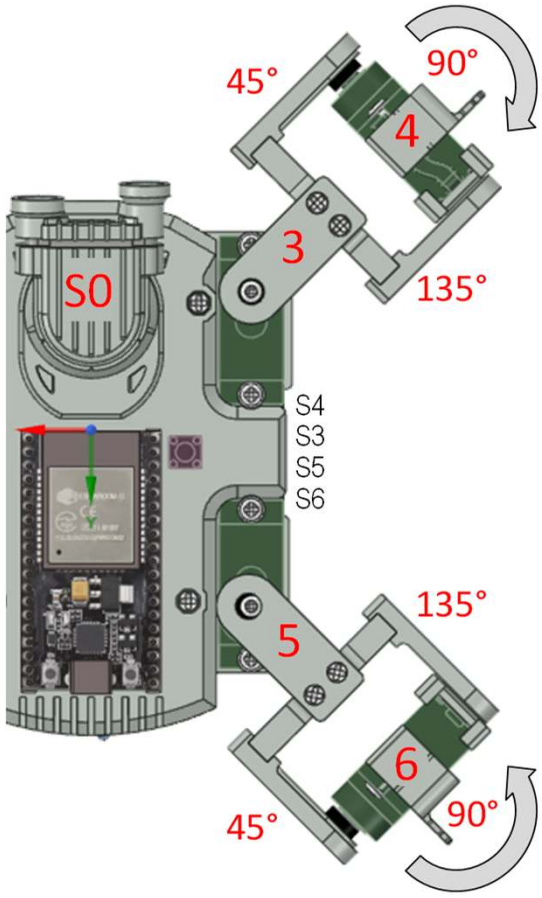
- When stationary all 4 legs are on the ground, and vertical to save power.
- Legs are splayed outwards, so that they can in effect be splayed further to be lifted them off the ground.
- When walking, at least 2 opposite legs are on the ground, pushing backwards, and 2 opposite legs are raised, moving forwards.
- Leg movements are constrained to avoid collisions with other legs, or the robots body.
- To turn whilst walking, the angular range (gate) is reduced on the side to which the robot is turning.
- Leg movements are based on a simple count sequence, so that walking speed can simply be changed by count frequency.
- To simplify changes in walking directions, the legs are moved quickly to predefined start points.

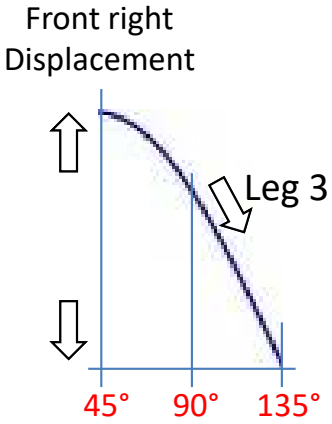


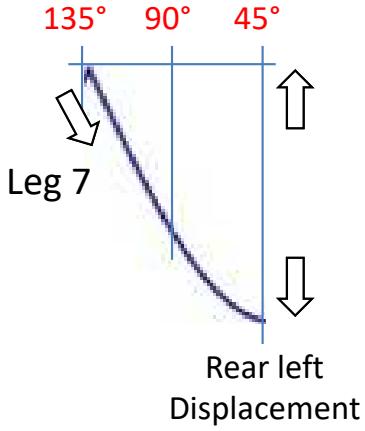


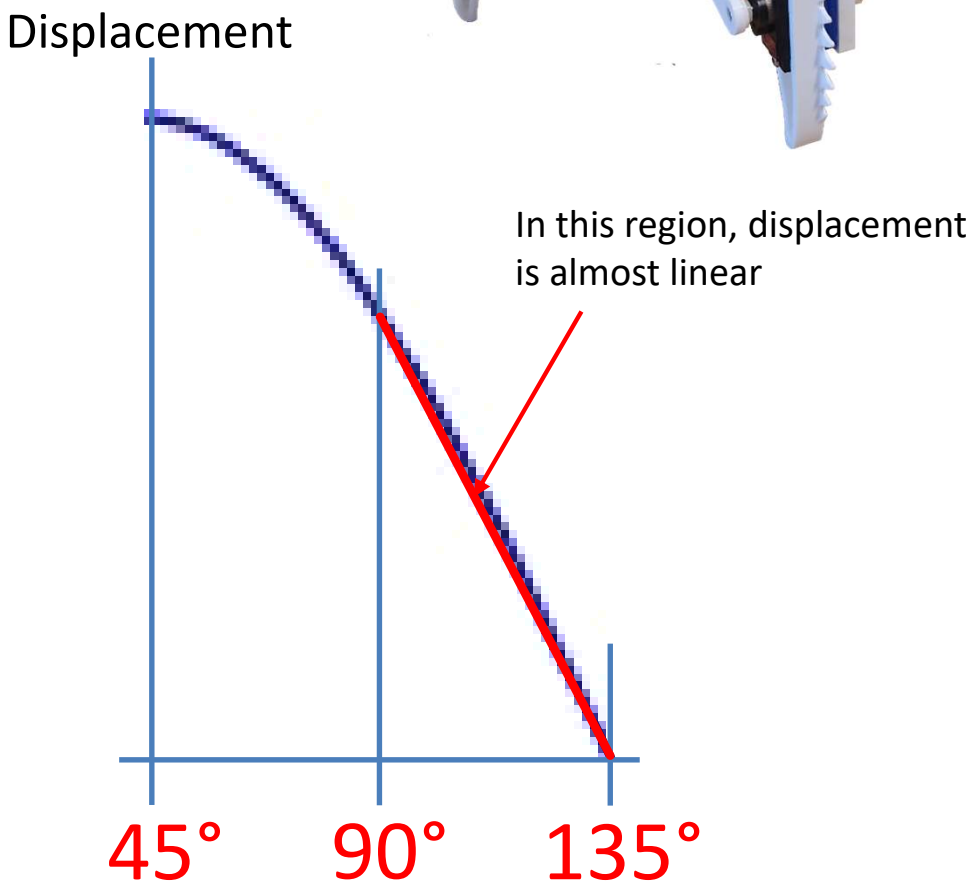
Walking Basics - Angles

- Changes in leg angle result in a sinusoidal displacement forwards/backwards.
- The displacement is smallest in the 45° region, and greatest in the 135° region.
- It is therefore best to use the angular range of 90° to 135° for all legs.
- Diagonally opposite legs (say 3 and 7) will be on opposite ends of the displacement curve.
- Mathematical adjustments in the code aims to minimise these displacement errors.









Walking Basics - Count

- The leg walking sequence can be broken down into 88 steps.
- With leg angles going from 90° up to 135°, and then back down to 90° again.
- With a 'Walk Drive' factor ranging from 0 – 128, to adjust the angle for steering.

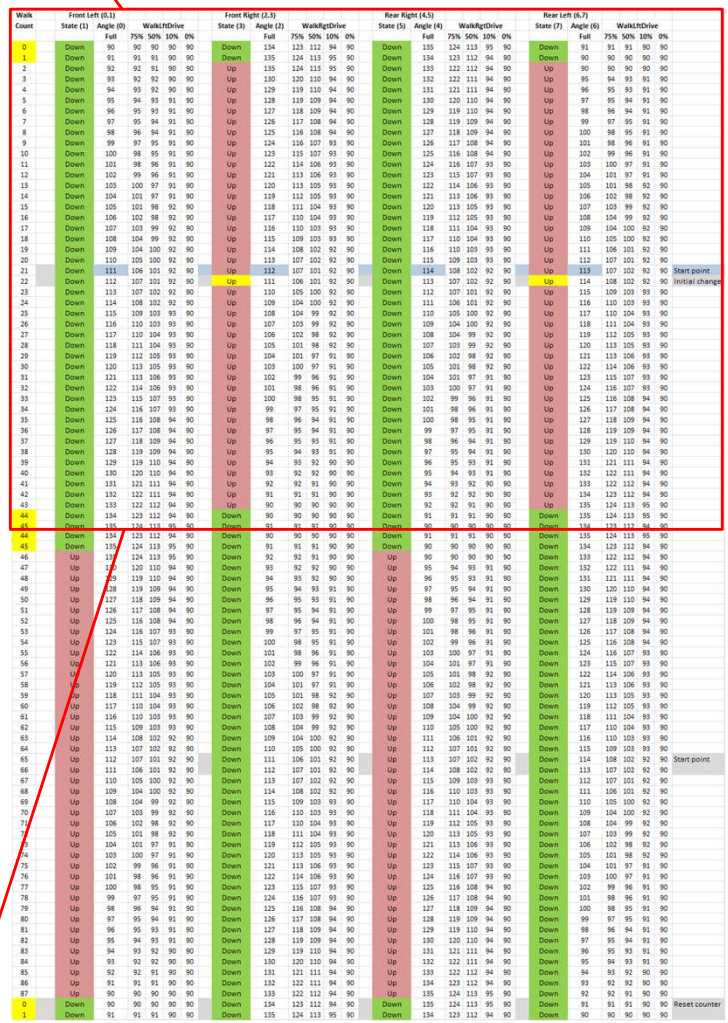
When:

GREEN - a leg is down

PINK - a leg is raised up

Walk	Front Left (0,1)						Front Right (2,3)						Rear Right (4,5)						Rear Left (6,7)						
Count	State (1)	Angle (0)	WalkLftDrive				State (3)	Angle (2)	WalkRgtDrive				State (5)	Angle (4)	WalkRgtDrive				State (7)	Angle (6)	WalkLftDrive				
		Full	75%	50%	10%	0%		Full	75%	50%	10%	0%		Full	75%	50%	10%	0%		Full	75%	50%	10%	0%	
0	Down	90	90	90	90	90	Down	134	123	112	94	90	Down	135	124	113	95	90	Down	91	91	91	90	90	
1	Down	91	91	91	90	90	Down	135	124	113	95	90	Down	134	123	112	94	90	Down	90	90	90	90	90	
2	Down	92	92	91	90	90	Up	135	124	113	95	90	Down	133	122	112	94	90	Up	90	90	90	90	90	
3	Down	93	92	92	90	90	Up	130	120	110	94	90	Down	132	122	111	94	90	Up	95	94	93	91	90	
4	Down	94	93	92	90	90	Up	129	119	110	94	90	Down	131	121	111	94	90	Up	96	95	93	91	90	
5	Down	95	94	93	91	90	Up	128	119	109	94	90	Down	130	120	110	94	90	Up	97	95	94	91	90	
6	Down	96	95	93	91	90	Up	127	118	109	94	90	Down	129	119	110	94	90	Up	98	96	94	91	90	
7	Down	97	95	94	91	90	Up	126	117	108	94	90	Down	128	119	109	94	90	Up	99	97	95	91	90	
8	Down	98	96	94	91	90	Up	125	116	108	94	90	Down	127	118	109	94	90	Up	100	98	95	91	90	
9	Down	99	97	95	91	90	Up	124	116	107	93	90	Down	126	117	108	94	90	Up	101	98	96	91	90	
10	Down	100	98	95	91	90	Up	123	115	107	93	90	Down	125	116	108	94	90	Up	102	99	96	91	90	
11	Down	101	98	96	91	90	Up	122	114	106	93	90	Down	124	116	107	93	90	Up	103	100	97	91	90	
12	Down	102	99	96	91	90	Up	121	113	106	93	90	Down	123	115	107	93	90	Up	104	101	97	91	90	
13	Down	103	100	97	91	90	Up	120	113	105	93	90	Down	122	114	106	93	90	Up	105	101	98	92	90	
14	Down	104	101	97	91	90	Up	119	112	105	93	90	Down	121	113	106	93	90	Up	106	102	98	92	90	
15	Down	105	101	98	92	90	Up	118	111	104	93	90	Down	120	113	105	93	90	Up	107	103	99	92	90	
16	Down	106	102	98	92	90	Up	117	110	104	93	90	Down	119	112	105	93	90	Up	108	104	99	92	90	
17	Down	107	103	99	92	90	Up	116	110	103	93	90	Down	118	111	104	93	90	Up	109	104	100	92	90	
18	Down	108	104	99	92	90	Up	115	109	103	93	90	Down	117	110	104	93	90	Up	110	105	100	92	90	
19	Down	109	104	100	92	90	Up	114	108	102	92	90	Down	116	110	103	93	90	Up	111	106	101	92	90	
20	Down	110	105	100	92	90	Up	113	107	102	92	90	Down	115	109	103	93	90	Up	112	107	101	92	90	
21	Down	111	106	101	92	90	Up	112	107	101	92	90	Down	114	108	102	92	90	Up	113	107	102	92	90	Start point
22	Down	112	107	101	92	90	Up	111	106	101	92	90	Down	113	107	102	92	90	Up	114	108	102	92	90	Initial change
23	Down	113	107	102	92	90	Up	110	105	100	92	90	Down	112	107	101	92	90	Up	115	109	103	93	90	
24	Down	114	108	102	92	90	Up	109	104	100	92	90	Down	111	106	101	92	90	Up	116	110	103	93	90	
25	Down	115	109	103	93	90	Up	108	104	99	92	90	Down	110	105	100	92	90	Up	117	110	104	93	90	
26	Down	116	110	103	93	90	Up	107	103	99	92	90	Down	109	104	100	92	90	Up	118	111	104	93	90	
27	Down	117	110	104	93	90	Up	106	102	98	92	90	Down	108	104	99	92	90	Up	119	112	105	93	90	
28	Down	118	111	104	93	90	Up	105	101	98	92	90	Down	107	103	99	92	90	Up	120	113	105	93	90	
29	Down	119	112	105	93	90	Up	104	101	97	91	90	Down	106	102	98	92	90	Up	121	113	106	93	90	
30	Down	120	113	105	93	90	Up	103	100	97	91	90	Down	105	101	98	92	90	Up	122	114	106	93	90	
31	Down	121	113	106	93	90	Up	102	99	96	91	90	Down	104	101	97	91	90	Up	123	115	107	93	90	
32	Down	122	114	106	93	90	Up	101	98	96	91	90	Down	103	100	97	91	90	Up	124	116	107	93	90	
33	Down	123	115	107	93	90	Up	100	98	95	91	90	Down	102	99	96	91	90	Up	125	116	108	94	90	
34	Down	124	116	107	93	90	Up	99	97	95	91	90	Down	101	98	96	91	90	Up	126	117	108	94	90	
35	Down	125	116	108	94	90	Up	98	96	94	91	90	Down	100	98	95	91	90	Up	127	118	109	94	90	
36	Down	126	117	108	94	90	Up	97	95	94	91	90	Down	99	97	95	91	90	Up	128	119	109	94	90	
37	Down	127	118	109	94	90	Up	96	95	93	91	90	Down	98	96	94	91	90	Up	129	119	110	94	90	
38	Down	128	119	109	94	90	Up	95	94	93	91	90	Down	97	95	94	91	90	Up	130	120	110	94	90	
39	Down	129	119	110	94	90	Up	94	93	92	90	90	Down	96	95	93	91	90	Up	131	121	111	94	90	
40	Down	130	120	110	94	90	Up	93	92	92	90	90	Down	95	94	93	91	90	Up	132	122	111	94	90	
41	Down	131	121	111	94	90	Up	92	92	91	90	90	Down	94	93	92	90	90	Up	133	122	112	94	90	
42	Down	132	122	111	94	90	Up	91	91	91	90	90	Down	93	92	92	90	90	Up	134	123	112	94	90	
43	Down	133	122	112	94	90	Up	90	90	90	90	90	Down	92	92	91	90	90	Up	135	124	113	95	90	
44	Down	134	123	112	94	90	Down	90	90	90	90	90	Down	91	91	91	90	90	Down	135	124	113	95	90	
45	Down	135	124	113	95	90	Down	91	91	91	90	90	Down	90	90	90	90	90	Down	134	123	112	94	90	

Full sequence.



Issue: 1.5 Released: 24/03/2026

Walking Basics - Code

- The ‘Walk_Engine’ code function is called repeatedly to perform incremental steps in the walk count.
- The frequency at which it is called, sets the pace at which the robot walks.

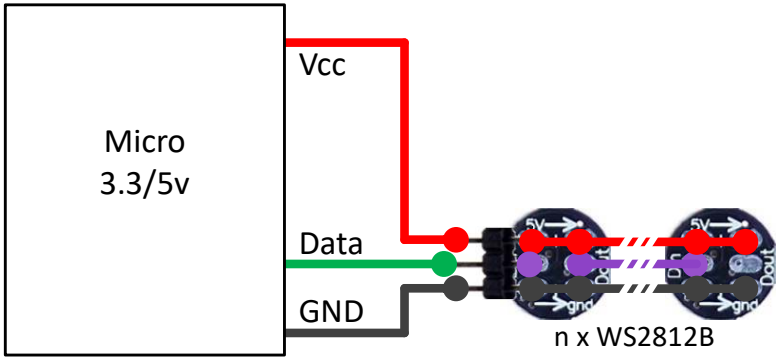
```
1930  void Walk_Engine() {
1931    // called from the main loop on a timer
1932    // Walk values:
1933    // 0  no walk action. But leg moves may be needed.
1934    // 1  walking forwards
1935    // 2  walking towards the right, sideways
1936    // 3  walking backwards
1937    // 4  walking towards the left, sideways
1938    // 5  neutral turning towards the left
1939    // 6  neutral turning towards the right
1940    if (ESC) {return;}
1941
1942    if (Walk > 0) {
1943      WalkCnt++; if (WalkCnt > 87) {WalkCnt = 0;} // increment and restart
1944      // Serial.println(WalkCnt);
1945      switch (Walk) {
1946        case 1: // walking forwards
1947          // front left leg &  rear right leg
1948          Walked++; // increase how far walked counter
1949          if (WalkCnt < 1) {SetAngle(2, LegDn); SetAngle(6, LegDn);}
1950          else if (WalkCnt == 22) {SetAngle(4,LegUp); SetAngle(8,LegUp);} // Up start condition
1951          else if (WalkCnt < 46) {SetAngle90To135(1,90+((WalkCnt * WalkLftDrive)/128));
1952            SetAngle90To135(5,90+(((45-WalkCnt)*WalkRgtDrive)/128));}
1953          else if (WalkCnt == 46) {SetAngle(2,LegUp); SetAngle(6,LegUp);}
1954          else {SetAngle90To135(1,90+(((87-WalkCnt)*WalkLftDrive)/128));
1955            SetAngle90To135(5,90-(((42-WalkCnt)*WalkRgtDrive)/128));}
1956          // front right leg &  rear left leg
1957          if (WalkCnt > 43) {SetAngle(4,LegDn); SetAngle(8,LegDn);
1958            SetAngle90To135(3,90+(((WalkCnt-44)*WalkRgtDrive)/128));
1959            SetAngle90To135(7,90+(((89-WalkCnt)*WalkLftDrive)/128));}
1960          else if (WalkCnt > 2) {SetAngle90To135(3,90+(((43-WalkCnt)*WalkRgtDrive)/128));
1961            SetAngle90To135(7,90+(((WalkCnt+2)*WalkLftDrive)/128));}
1962          else if (WalkCnt > 1) {SetAngle(4,LegUp); SetAngle(8,LegUp);}
1963          else {SetAngle(4,LegDn); SetAngle(8,LegDn);
1964            SetAngle90To135(3,90+(((WalkCnt+44)*WalkRgtDrive)/128));
1965            SetAngle90To135(7,90+(((1-WalkCnt)*WalkLftDrive)/128));} break;
1966
```

The variable ‘Walk’ determines the logic applied to the count sequence, and therefore the direction in which it walks. There are 6 different sequences to consider.

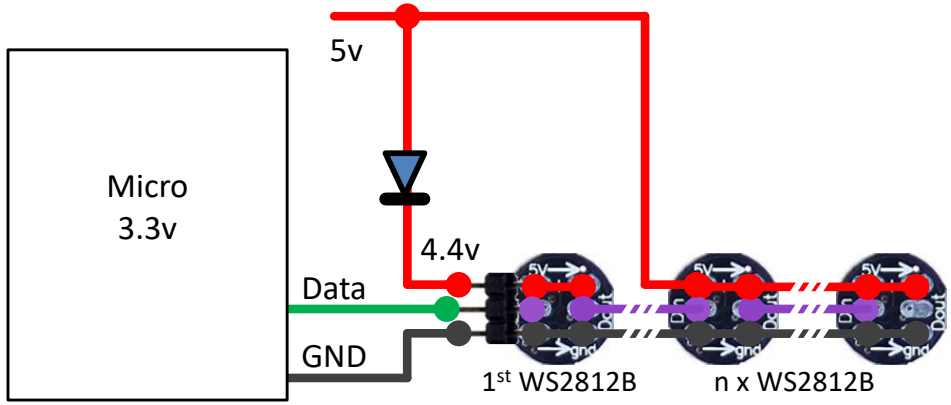
The ‘Walk Drive’ variables determines the extent to which the angular steering motion is applied. At 128 there is full leg movement, reducing to none, at 0.

WS2812B RGB LEDs

- A WS2812B device, is one of the most common forms of programmable RGB LEDs.
- It contains three LEDs, Red, Green and Blue, controlled by 24-bit (3 x 8-bit) values. When combined equally, they produce white light.
- They are designed for 5v operations, but they will work down to 3.3v at low power levels, making them ideal for microcontrollers.
- An internal invertor, pumps up the voltage feeding the three LEDs, to 12v, to give consistent colour/brightness output levels.
- A CMOS device. If run at 5v, for output power, but driven from a 3.3v micro, a level-shifter is required on the first devices input.
- Devices are connected in a 3-wire daisy chain fashion; limited by power distribution, and data timing considerations.
- Serial data is clocked into the device at 800kbps, and timing is quite critical. All LEDs in a chain change synchronously, at the end of the data stream.
- It takes 28.8µs to clock in 24-bits of data, to configure one LED. At a frame rate of 25Hz you could drive a chain of 1389 LEDs.
- The first data block (24-bits) is taken by the first LED, which then passes on the rest down the chain. Excess data will fall off the end.
- Coders use libraries to configure and drive a micros GPIO pins. Adafruit Neopixel or FastLED, are commonly used C++ libraries.



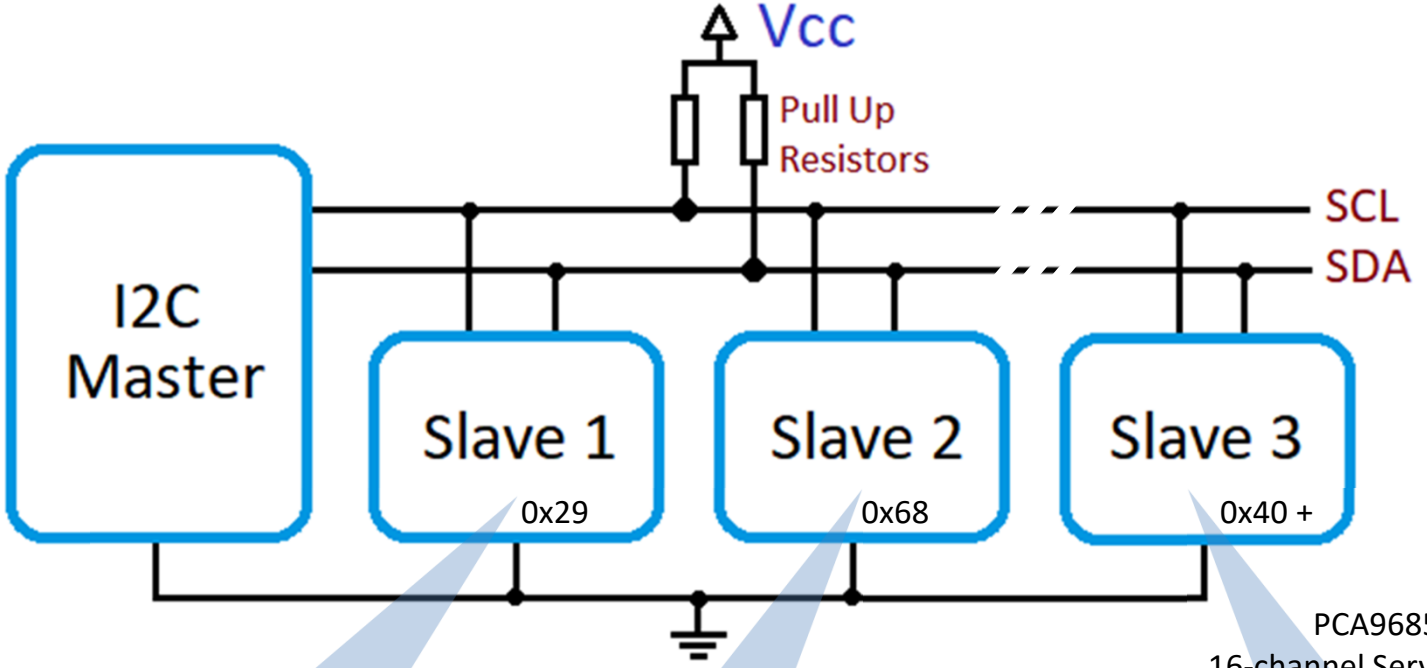
Common supply configuration



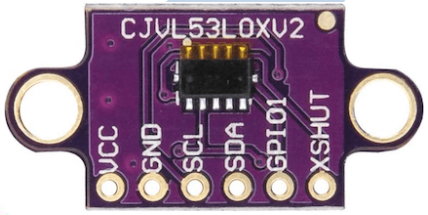
Level-shifter configuration

I2C Data Bus

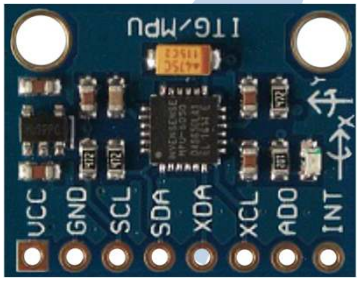
- This is a common hardware option, built into most microcontrollers, like Arduino, ESP32, etc.
- The I2C (Inter-Integrated Circuit) 4-wire bus, is a synchronous, low-speed serial data bus, for connecting peripherals to a host microcontroller.
- The open-collector 2-wire bus uses a Serial Data (SDA), and a Serial Clock (SCL) connection, with two pull-up resistors, over short distances.
- The Master/slave relationship is controlled by the micro, and all peripherals have a unique 7-bit or 10-bit address.
- Peripheral devices will have a range of 8-bit register, used to configure and exchange data.
- Bus speeds vary from Standard (100 kHz), Fast Mode (400 kHz), to Fast Mode Plus (1 MHz)
- Coders use libraries to communicate with peripherals; or parts of library code, if the library code has limitations.



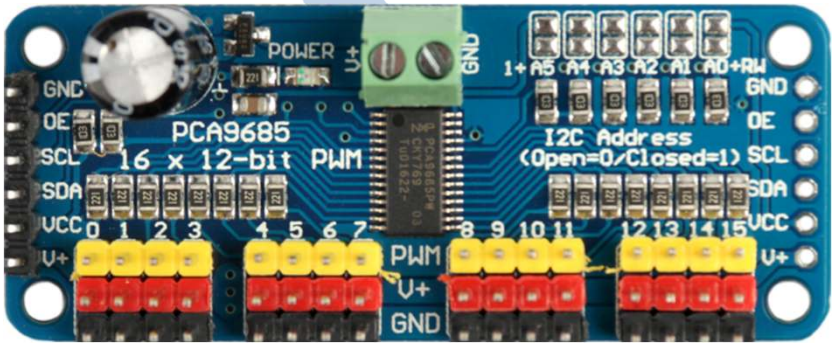
VL53L0X
Laser Range Finder

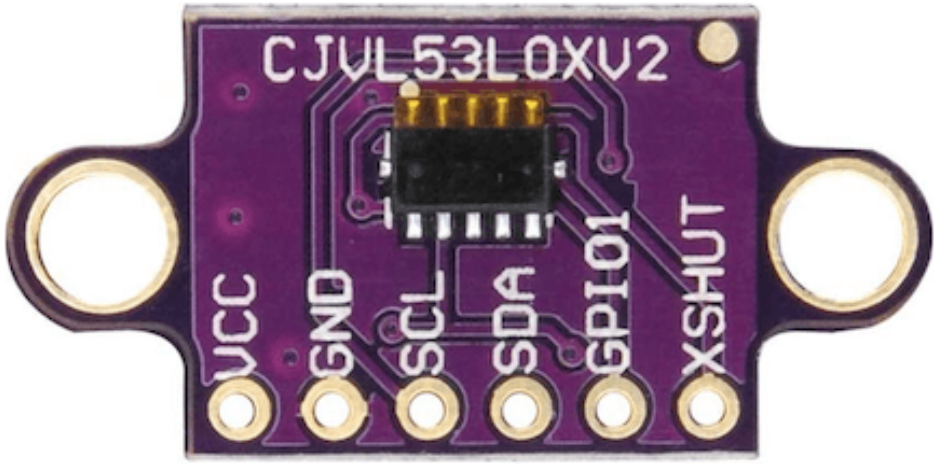


MPU6050
3-axis Motion Sensor



PCA9685
16-channel Servo Driver

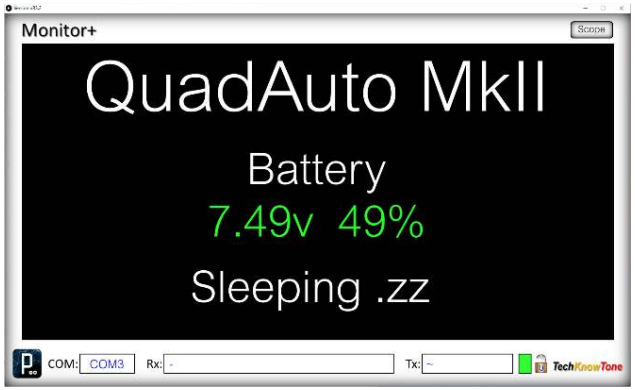


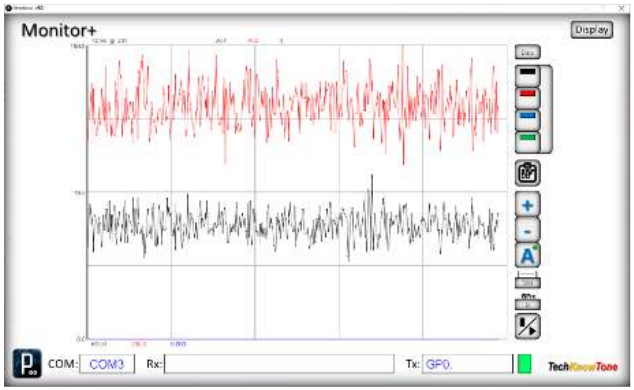

CJVL53L0XV2
VCC
GND
SCL
SDA
GPIO1
XSHUT

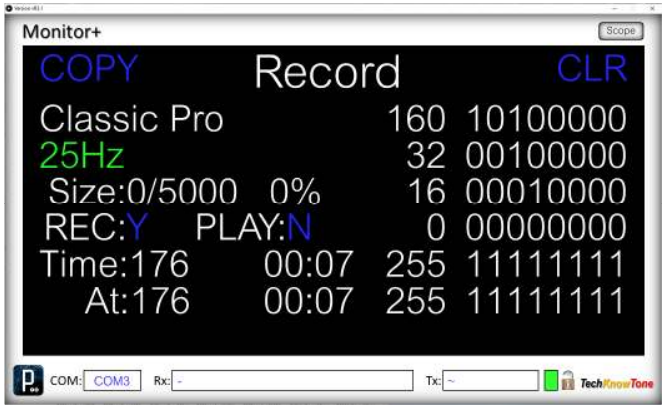
Wii Controllers – I2C

ESPNOW 2.4 GHz WiFi

Monitor+ App


Monitor+
Scope
QuadAuto MkII
Battery
7.49v 49%
Sleeping .zz
COM:
COM3
Rx:
Tx:
TechKnowTone


Monitor+
Display
COM:
COM3
Rx:
Tx:
GP0.
TechKnowTone


Monitor+
Scope
COPY
Record
CLR
Classic Pro
25Hz
Size:0/5000 0%
REC:Y PLAY:N
Time:176 00:07
At:176 00:07
160 10100000
32 00100000
16 00010000
0 00000000
255 11111111
255 11111111
COM:
COM3
Rx:
Tx:
TechKnowTone