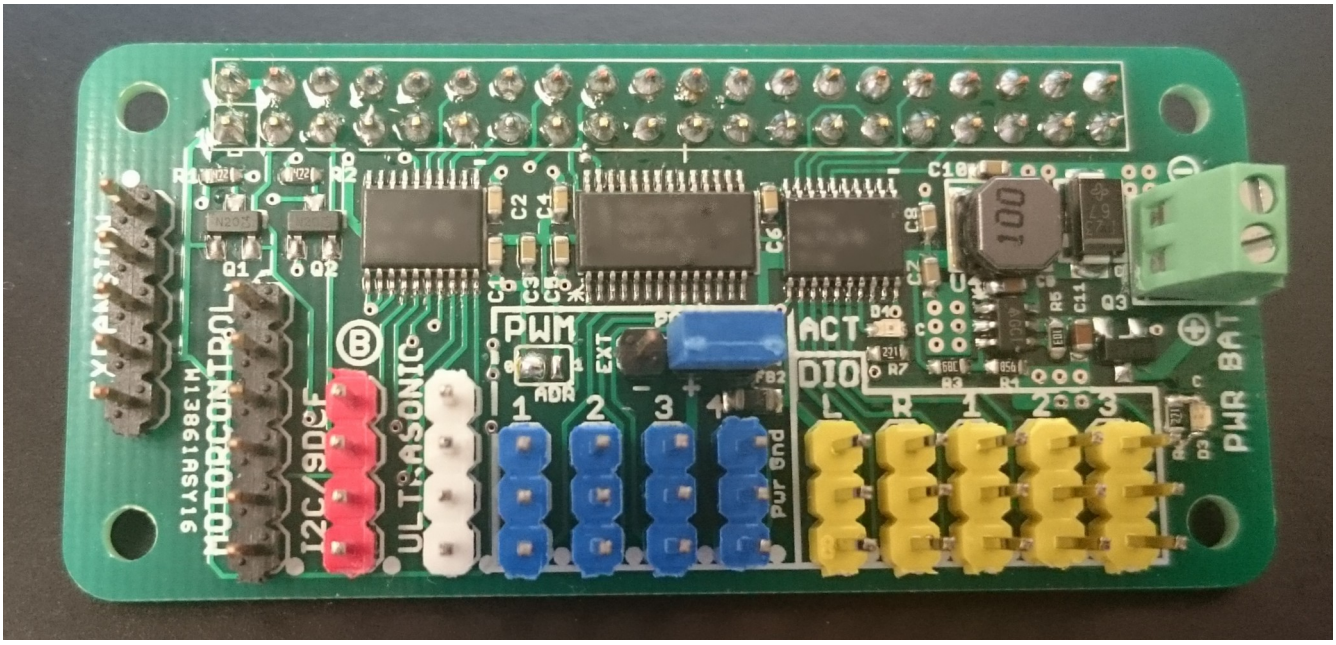
PiBot Zero
Technical Reference Manual

Table of Contents

Introduction

The Raspberry Pi is a powerful computing platform with a wide range of robotics applications, however it lacks the electrical interfaces needed to connect it with the sensors and controllers commonly used for robotics.



The PiBot Zero is a compact stacking daughter-card with the same footprint as the Raspberry Pi Zero. That can be used with the Raspberry Pi Zero, 2, or 3; it connects to the Pi GPIO Header to adapt the Raspberry Pi for robotics by providing:

- Power supply/voltage regulator: 6-20vdc input, 5v @ 1.5A output
- Dual DC motor controller interface
- 4x-6x DC/Servo motor control interfaces
- 5x-11x digital I/O interfaces (5v I/O)
- 1x Ultrasonic range-finder interface
- I2C/9DOF interfaces (3v3 and 5v devices supported)
- Activity LED
- Optional integrated 9DOF sensor

Each interface has its own connector with power and ground, designed for easy, straight-through wiring to sensors, servos, modules, etc.. These interfaces, together with the native camera and USB support offered by the Raspberry Pi serve as a complete robotics control platform for large and small robots.

Note: standard interface connectors are 0.100” pin headers. Pin 1 is marked with a dot.
Keyed and/or locking connectors are available as a factory option.

Please observe proper ESD handling procedures for electronic equipment

Power Supply

The Raspberry Pi and most robot peripherals require a well regulated +5vdc power supply. The PiBot Zero includes a high quality switching power supply that accepts the unregulated 6v-20vdc from most robot batteries and provides a regulated 5v output to power the Pi and its connected peripherals.

Power supply features include:

- Regulated +5vdc output at up to 1.5A (1500mA)
- 80-92% conversion efficiency
- Screw terminals for battery connection
- Reverse polarity protection
- Power indicator LED (green)

Power Budget

Raspberry Pi power consumption varies with the Pi model and the load on the microprocessor:

Pi Model	Idle	Heavy Load
Zero	<100mA	350mA
rPi2	260mA	420mA
rPi3	310mA	580mA

When designing a PiBot, it is important to calculate the worst-case power consumption and ensure that the it remains within the PiBot Zero's 1.5A limit. For example:

Device	Current (mA)	Qty	Total
Pi Zero	250	1	250
EW-7811UN WiFi adapter	100	1	100
Pi Camera	150	1	150
PiBot Zero + activity LED	12	1	12
L298N DC Motor Controller	1	1	1
HC-SR04 ultrasonic sensor	15	1	15
Tower Pro MG90S servo (each)	460	1	460
Indicator LED (each)	15	3	45
HC-020K wheel encoder (each)	20	2	40
LSM9DS1 IMU (9DOF)	5	1	5
Limit switch (each)	2	3	6
Total (must be <= 1500)			**1084**

Powering Motors

Motors consume a great deal of current that varies with the load on the motor. Although the PiBot Zero can power one or two small servo motors, it is not designed to directly power most motors and generally an external motor controllers should be used.

Most robots use two types of motors:

- Servo Motors: the PiBot Zero can directly power one or two small servo motors via the PWM outputs. To control many servo motors or more powerful servo motors, a separate power supply should be used to power the servo motors such as an inexpensive MP1584EN module. For details see the section on PWM/Servo Motor Control.

- DC Motors: The Pibot Zero is designed to drive external DC motor controllers. These controllers regulate power directly from the battery to the motors and draw their power directly from the battery (not from the PiBot Zero). The PiBot Zero uses PWM output signals to these motor controllers to adjust motor speed and direction.

    - For small robots, the L298N dual-channel motor controller can power 2 DC motors at up to 2A each. The L298N requires less than 1mA from the PiBot Zero and derives the rest of its power directly from the robot battery. The PiBot Zero Motor Control connector is designed to directly connect to an L298N module (see section on Motor Controller)

    - For larger robots, motor controllers such as the Victor SP, Spark, or Jaguar can power motors that draw 40A-100A. These motor controllers typically draw 20mA each from the PiBot Zero to power their opto-isolated inputs.

In addition to ensuring that the Pi and peripherals stay within the power budget for the PiBot Zero, it is important to ensure the overall robot power consumption (including motors and motor controllers) stays within the power budget of the battery and wiring. If the motors draw more current than the battery is able to supply, the battery voltage will drop and if it drops below the minimum required for the PiBot Zero, the Raspeberry Pi will reboot.

Note: You can manage power consumption by limiting the motors turned on at the same time.

Safety

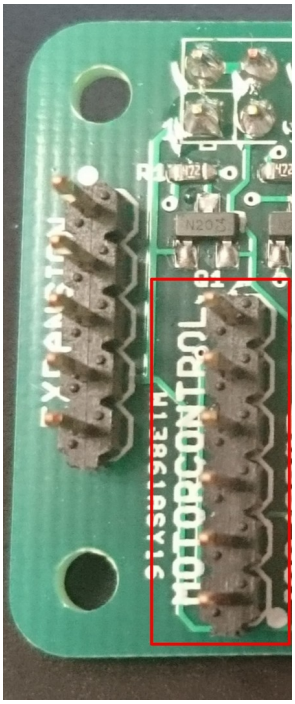
Robot batteries can supply large amounts of electric current that can be dangerous or even fatal. It is critical to follow appropriate electrical safety procedures when working with robot batteries.

Robots powered by large motors can be physically dangerous as well. Follow all proper safety procedures when working with robots to protect people and property.

Ensure proper gauge wire is used to prevent a possible fire. The PiBot Zero battery connection should be 26AWG or larger. Wires for large motors may need to be 10AWG or larger and battery wires may need to be 6AWG or larger. Wires should always be sized for worst case current requirements.

DC Motor Control

MOTORCONTROL Connector



The black 6-pin MOTORCONTROL connector is designed to interface with external DC motor controllers to control the speed and direction of two DC motors. High power PWM motor speed controllers and low-cost direct H-bridge motor controllers are supported.

The connector is pin-compatible with the L298N dual H-bridge DC motor controller which is is available for under $2 and provides variable speed and direction control for two brushed DC motors. The L298N has its own power supply and is connected directly to the robot battery; it supports voltages as high as 46v. Each DC motor can be driven with up to 2A continuous drive current and 3A peaks making it ideal for small, low cost robots.

To control larger motors, the 6-pin L298N connector can be used as two separate 3-pin PWM outputs to drive higher-power motor controllers such as the Victor SP or Spark. These motor controllers are suitable for high-current motors such as the CIM motors used in FRC robotics and can provide up to 60A continuous drive current.

The PiBot Zero MOTORCONTROL interface uses two PWM channels and four GPIO channels to provide variable speed and direction control for each motor channel. If DC motor control functions aren't needed, the MOTORCONTROL connections can serve as additional digital inputs and outputs (DIO) and PWM control outputs which can be used to control servo motors.

MOTORCONTROL Pin	Pi Header Pin	Wiring Pi Pin	Broadcom Function
1	N/A	N/A	PWM Channel 1
2	18	5	GPIO24 (BCM24)
3	15	3	GPIO22 (BCM22)
4	12	1	GPIO18 (BCM18)
5	11	0	GPIO17 (BCM17)
6	N/A	N/A	PWM Channel 0

PWM control signals are generated in hardware and do not depend on the Raspberry Pi microprocessor other than for configuration. Interface voltages are 0 to +5vdc.

Programming

The MOTORCONTROL connector supports two DC motor control channels. How the two channels implement motor speed control depends on the external motor controller being used.

Direct H-Bridge motor control (e.g. L298N): The PWM outputs control motor speed by varying the duty cycle from 0 to 100%; the GPIO outputs determine direction or braking.

L298N Channel A

Direction	**ENA (PWM1)**	**IN1 (GPIO24)**	**IN2 (GPIO22)**
Forward	0..100% duty=speed	High (1)	Low (0)
Reverse	0..100% duty=speed	Low (0)	High (1)
Brake		Low (0)	Low (0)
Brake		High (1)	High (1)

L298N Channel B

Direction	**ENB (PWM0)**	**IN3 (GPIO18)**	**IN4 (GPIO17)**
Forward	0..100% duty=speed	High (1)	Low (0)
Reverse	0..100% duty=speed	Low (0)	High (1)
Brake		Low (0)	Low (0)
Brake		High (1)	High (1)

PWM motor controller: For high current motor controllers such as the Victor SP or Spark, the PWM output determines both speed and direction using signals compatible with servo motor control. The GPIO outputs only provide a ground return path for the PWM signal. **NOTE**: A 560-ohm series resistor is recommended for the PWM control signal.

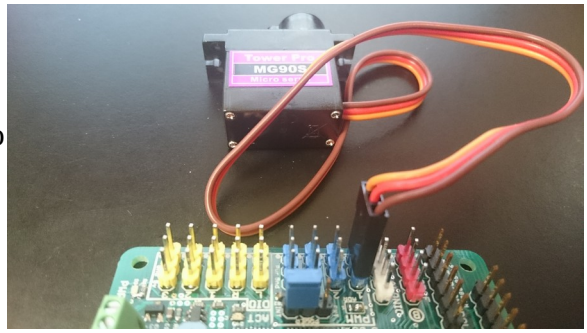
Motor Controller 1

Direction	**White (PWM1)**	**Red (GPIO24)**	**Black (GPIO22)**
Forward	1.54ms-2.00ms=speed	High (1)	Low (0)
Reverse	1.00ms-1.46ms=speed	High (1)	Low (0)
Brake	1.46-1.54 ms	High (1)	Low (0)

Motor Controller 2

Direction	**White (PWM0)**	**Red (GPIO17)**	**Black (GPIO18)**
Forward	1.54ms-2.00ms=speed	High (1)	Low (0)
Reverse	1.00ms-1.46ms=speed	High (1)	Low (0)
Brake	1.46-1.54 ms	High (1)	Low (0)

PWM Servo/DC Motor Control

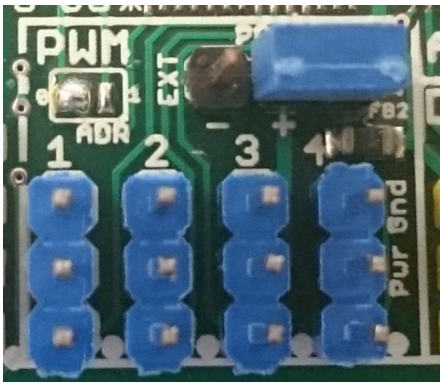


Blue 3-pin connectors PWM1-4 provide control outputs for up to 4 DC servo motors or additional external DC motor controllers. Each connector provides power, ground, and a PWM control signal and supports direct connection to a servo motor or DC motor controller.

PWM signals are generated by an integrated NXP PCA9685 which can produce up to 16 independent PWM control signals. Signals are generated in hardware for precisely controlled outputs that do not burden the Raspberry Pi microprocessor.

The pin functions on each of the 4 headers are:

PWM Pin	Function	Wiring
1	PWM Signal	White or orange
2	PWM Power	Red
3	Ground	Black or brown



PWM Power

Pin (2) of each connector provides power to the attached servo motor. The PWM power source is selected using the PWM Power header (black horizontal 3-pin header):

- Internal 5v power supply (default): For external DC motor controllers or one or two small servo motors, use the supplied blue jumper to connect the right pin marked INT to the center pin (PWM Power) as shown in photo inset.

- External 5v or 6v power configuration: For larger servo motors or several small servo motors, remove the blue jumper and connect external +5v or +6v to the center pin marked '+' and external ground to left pin marked EXT ('-'). An external MP1584EN module can provide up to 3A of current for servo motors. Remember to adjust the module output voltage to 5-6vdc using the trimmer potentiometer before connecting your servos.

IMPORTANT: Do not reverse the power connections for PWM power or you may damage your servo motors, the PiBot Zero, your Raspberry Pi, and any connected peripherals. Double check connections!

Programming

The NXP PCA9685 default address is 0x40, but address 0x41 may be selected by moving the ADR solder bridge from position 0 (default) to position 1. The address configuration may be specified as a factory option.

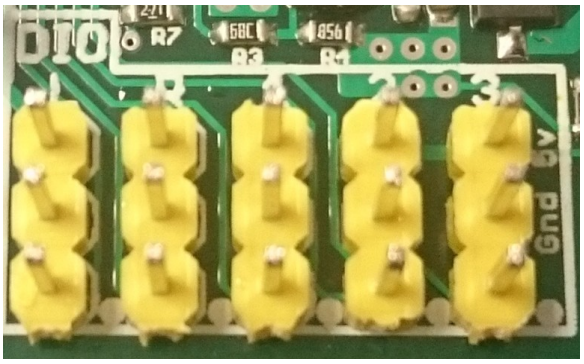
The PCA9685 uses an internal clock to generate the PWM frequency and the Raspberry Pi can configure the duty cycle for each channel using the I2C bus interface.

Connector	Pin	PWM Channel
Motor Controller	6	0
Motor Controller	1	1
PWM 1	1	2
PWM 2	1	3
PWM 3	1	4
PWM 4	1	5

DIO/Encoder Channels

Yellow 3-pin connectors (DIO L, R, 1-3) support connection to limit switches, LED indicators, rotary encoders, and other digital I/O. Each connector signal pin can be configured as an input or output.

Outputs swing between 0v and 5v, suitable for connection to most robot sensors. Inputs will recognize 3.25v to 5vdc as a high (1) logic level.



The pin functions on each of the DIO headers are:

DIO Pin	Function	Notes
1	Signal	Digital input or output
2	Ground	
3	+5vdc	

The L and R inputs are often connected to left and right wheel encoders such as the HC-020K to measure wheel speed, however they may be used for any digital I/O application.

When configured as outputs, the impedance of each signal pin is approximately 40 ohms. The output pins should **not** be used to drive high-current loads; **total** output current (source or sink) by all of the DIO pins should not exceed 100mA and no single pin should source or sink more than 50mA. If higher currents are needed, an external buffer must be used. When driving an LED or similar device, a series resistor must be used to limit current.

When configured as an input, DIO input impedance is 4K-ohms and should be driven by a signal that can source or sink more than 2mA. For example, when implementing a limit switch, it is desirable to use an SPDT switch connecting the signal line to either Ground or +5vdc. Note that using pull-up or pull-down resistors with the DIO inputs maybe challenging due to the relatively low input impedance which will form a voltage divider with the pull-up/down resistor.

Programming

DIO pin connections are as follows:

DIO	Pi Header Pin	Wiring Pi Pin	Broadcom Function
L	29	21	BCM5
R	32	26	BCM12 (PWM0)
1	31	22	BCM6
2	33	23	BCM13 (PWM1)
3	36	27	BCM16

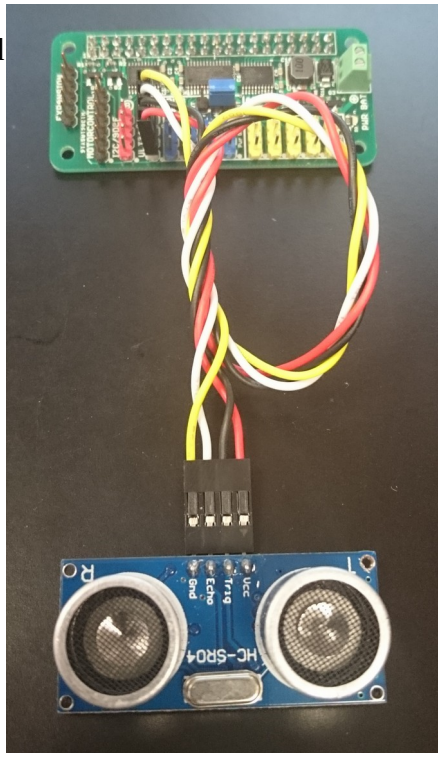
NOTE: DIO pins do not connect directly to the Pi Header; Raspberry Pi I/O is limited to 3v3 and would be damaged by the 5v levels required for many sensors.

Activity LED

An amber status LED on the PiBot Zero is controlled by the Raspberry Pi header pin 35 (BCM 19 / Wiring Pi pin 24). A 220-ohm series LED sets the On-current to approximately 11mA.

The LED allows the user to provide visual status by turning the LED on/off/blinking.

Ultrasonic/LIDAR Sensor Interface



The white 4-pin header provides an interface for the popular HC-SR04 ultrasonic rangefinder sensor. The HC-SR04 is available for under $1 and provides non-contact measurement of distance from 2cm to 400cm with good accuracy. The pin header connects to the HC-SR04 via a straight-through 4-pin cable and supplies 5v power for the module as well as the input and output control signals.

The TRIG output initiates a range measurement and the ECHO input receives a pulse whose duration is directly proportional to the distance to the target. The interface can also be used with higher accuracy ultrasonic sensors such as the Maxbotix HRLV series of sensors and with LIDAR range finders such as the LIDAR-Lite.

If an ultrasonic sensor is not used, the TRIG and ECHO pins may be used as general purpose digital inputs and outputs (DIO).

Programming

The Ultrasonic connector pin functions are:

ULTRASONIC Pin	HC-SR04 Pin	MaxBotix Pin	LIDAR Pin	Pi Header Pin	Wiring Pi Pin	Broadcom Function
1	+5V	6	1			
2	TRIG	4	2	16	4	BCM23
3	ECHO	2	3	13	2	BCM27
4	GND	7	6			

I2C/9DOF Interface

The red 4-pin header provides an interface for connection to 3.3v I2C peripherals such as 9-degree-of-freedom (9DOF) sensors. 9DOF sensor modules combine a digital compass/magnetometer, gyroscope, and accelerometer to provide sophisticated navigation capabilities for robotics. Modules may include discrete components such as the ITG3200, ADXL345, and HMC5883L or single chip high-integration sensors such as the MPU-9250 and LSM9DS1. 9DOF modules are often available for under $10. Digital compasses such as the HMC5883L are available for under $3.

The SCL1 and SDA1 signals on the I2C/9DOF interface operate at 3.3v levels and should **not** be connected to 5V I2C devices (see Expansion interface). Pull-up resistors are provided and should not be added on the external module. The connector pins are configured for a straight-through cable connection to most common low-cost 9DOF sensors.

If a long wire is used to connect the 9DOF sensor, it may be desirable to reduce noise on the power lines that will degrade accuracy. Add a decoupling capacitor between +3v3 and GND on the sensor module; values between 0.1uF and 10uF may be used; a non-polarized ceramic capacitor is recommended.

Programming

The I2C/9DOF connector pin functions are:

I2C/9DOF Pin	Function	Pi Header Pin	Wiring Pi Pin	Broadcom
1	+3.3v	1		
2	GND	6		
3	SCL1	5	9	BCM3 (SCL)
4	SDA1	3	8	BCM2 (SDA)

Each connected device must have its own unique address on the I2C bus. For example, the HMC583L digital compass uses I2C address 0x1E. The LCM9DS1 IMU (9DOF) uses I2C addresses 0xD4,0xD6 (accelerometer and gyroscope), and 0x38, 0x3C (magnetometer).

NOTE that pins 3 (SCL1) and 4 (SDA1) connect directly to the Raspberry Pi GPIO header pins and must not use voltages higher than 3.3vdc or the Raspberry Pi will be damaged. To use 5v I2C devices, see the Expansion interface below. These pins do not have the +/-15kV ESD immunity provided by the DIO pins and appropriate care should be taken.

Integrated 9DOF option

As a factory option, the PiBot Zero may be ordered with an integrated LSM9DS1 9DOF sensor. For information on programming the LSM9DS1, please refer to the datasheet.

Expansion Interface

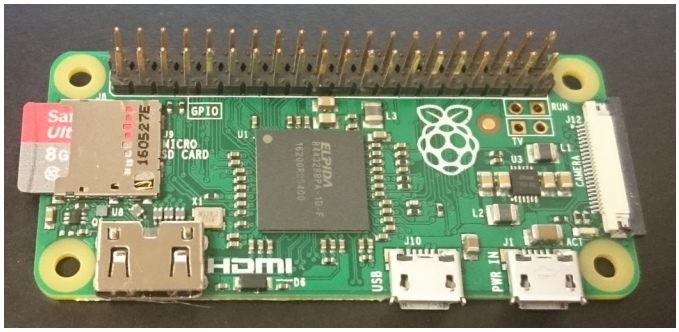
The black 5-pin expansion interface provides the ability to interface with common 5v devices using the I2C interface. Many devices such as the LIDAR-Lite can be connected to the same I2C bus allowing virtually unlimited expansion.

Expansion Connector Pin	Function	Pi Header Pin	Wiring Pi Pin	Broadcom Function
1	+5v	2		
2	SDA1	3	8	BCM2 (SDA)
3	SCL1	5	9	BCM3 (SCL)
4	+3.3v	1		
5	GND	6		

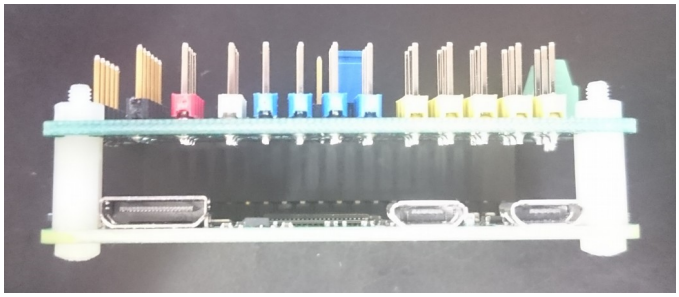
The Expansion interface provides the same I2C bus as the I2C/9DOF interface, but with bi-directional level shifters to allow connection of 5V devices. 3.3v devices should **not** be connected to the expansion interface I2C bus (connect these to the I2C/9DOF interface described above). **NOTE**: Pin 1 is at the TOP of the expansion connector.

Mounting

The PiBot Zero is designed as a daughtercard for a Raspberry Pi Zero; it will also work with the Raspberry Pi models 2 and 3. The Pi Zero GPIO header should be populated with a 2x20 pin header.

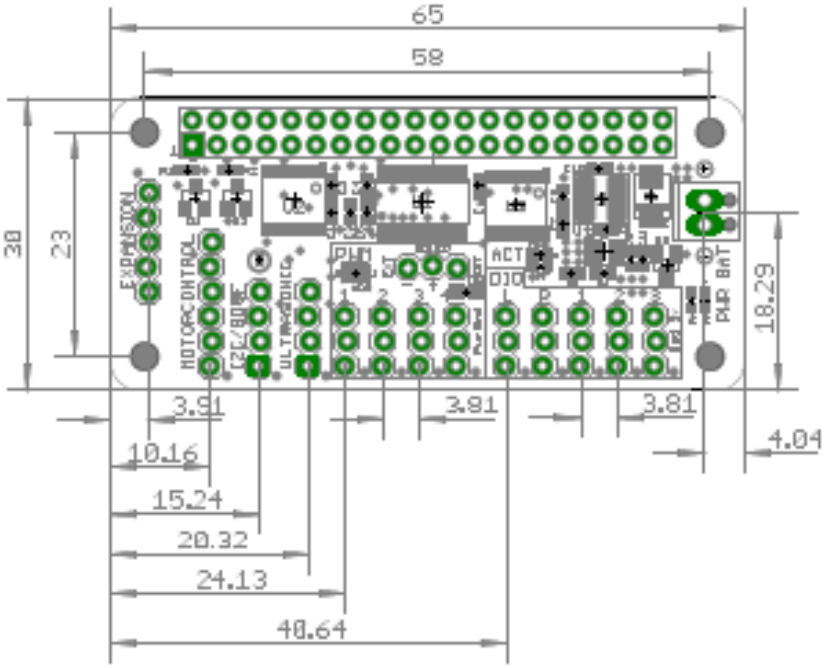


M2.5 x 11mm nylon standoffs may be used to securely mount the PiBot Zero to the Raspberry Pi Zero.



Dimensions

All dimensions are in mm:



Operating Specifications

Parameter	**Min**	**Max**
Operating Temperature	-40C	+85C
Battery Input Voltage	6vdc	20vdc
+5V DC output current (total)		1.5A
DIO input current	+/- 2mA	
DIO output current (any one pin)		+/-50mA
DIO output current (total incl/actvity LED)		+/-100mA
DIO ESD limits		+/-15kV
PWM output current		+10 / - 25mA
PWM ESD limits		+2kV HBM
+3v3 output current (supplied by Pi)		250mA

Software

The Raspberry Pi is a complete, modern computer capable of running modern general purpose operating systems such as linux and can be programmed using a wide variety of languages. The ability to program the Pi in Java is particularly appealing in educational settings since most high schools have standardized on Java as the first programming language. The Pi can also be programmed using C, C++, Python, and many other languages.

C, C++:

The popular WiringPi library provides excellent support for connecting with peripheral devices such as the PiBot Zero.

Java:

The Pi4J and Oracle DIO libraries provide excellent support for writing robotics applications in Java that must interface with peripheral devices such as the PiBot Zero.

Sample code for a digital input using Oracle DIO:

```
public class DigitalInput {

   // GPIO pin connected to the input
   private GPIOPin pin;

   /**
    * Create a digital input connected to the specified GPIO Pin
    * @param pinNumber e.g. 5 for GPIO5
    * @throws IOException
    */
   public DigitalInput(int pinNumber) throws IOException {
      // Configure pin as input
      GPIOPinConfig pinConfig = new GPIOPinConfig(DeviceConfig.DEFAULT,
          pinNumber,
          GPIOPinConfig.DIR_INPUT_ONLY,
          GPIOPinConfig.MODE_INPUT_PULL_UP,
          GPIOPinConfig.TRIGGER_BOTH_EDGES,
          false); // initial value ignored for inputs
      this.pin = (GPIOPin)DeviceManager.open(pinConfig);
   }

   /**
    * Get the current input status
    * @returns true if input is high (5v) or false if low (0v)
    */
   public boolean get() throws IOException {
      return pin.getValue();
   }

   /**
    * Sets a method to be called when the input status changes
```

```
     * @param listener method that takes a PinEvent parameter and gets
     *        called whenever the input status changes
     *        @code{
     *            void dioListener(jdk.dio.gpio.PinEvent e) {
     *                if (e.getValue()) ...
     *            }
     *        }
     */
    public void setInputListener(PinListener listener) throws IOException {
       pin.setInputListener(listener);
    }
}
```

Oracle offers the following sample code for interfacing with an Ultrasonic Sensor such as the HC-SR04:

```
public class HCSR04Device {

      private final int PULSE = 10000;        // #10 µs pulse = 10,000 ns
      private final int SPEEDOFSOUND = 34029; // Speed of sound = 34029 cm/s
      private GPIOPin trigger = null;
      private GPIOPin echo = null;

      public HCSR04Device(int _trigger, int _echo) {
      …
      trigger = (GPIOPin) DeviceManager.open(new GPIOPinConfig(
                  0, _trigger, GPIOPinConfig.DIR_OUTPUT_ONLY,
                  GPIOPinConfig.MODE_OUTPUT_PUSH_PULL,
                  GPIOPinConfig.TRIGGER_NONE, false));
            echo = (GPIOPin) DeviceManager.open(new GPIOPinConfig(
                        0, _echo, GPIOPinConfig.DIR_INPUT_ONLY,
                        GPIOPinConfig.MODE_INPUT_PULL_UP,
                        GPIOPinConfig.TRIGGER_NONE, false));
            I2CUtils.I2Cdelay(500);  //wait for 0.5 seconds
             ...
       }

      public double pulse() {
      long distance = 0;
      try {
            //Send a pulse trigger; must be 1 and 0 with a 10 µs wait
            trigger.setValue(true);
                  I2CUtils.I2CdelayNano(0, PULSE);// wait 10 µs
                  trigger.setValue(false);
                  long starttime = System.nanoTime(); //ns
                  long stop = starttime;
                  long start = starttime;
                  //echo will go 0 to 1 and need to save time for that.
                  //  2 seconds difference
                  while ((!echo.getValue()) &&
                        (start < starttime + 1000000000L * 2)) {
                      start = System.nanoTime();
                  }
                  while ((echo.getValue()) &&
                         (stop < starttime + 1000000000L * 2)) {
```

```
                    stop = System.nanoTime();
                }
                long delta = (stop – start);
                // echo from 0 to 1 depending on object distance
                distance = delta * SPEEDOFSOUND;
            } catch (IOException ex) {
                Logger.getGlobal().log(Level.WARNING,ex.getMessage());
            }
            return distance / 2.0 / (1000000000L); // cm/s
    }

    public void close() {

    ...
        if ((trigger!=null) && (echo!=null)){
            trigger.close();
            echo.close();;
        }
    ...
    }

}

public class TestSensors extends MIDlet {
    //Define HCSR04 Device object
    HCSR04Device hcsr04;
    private static final int TRIGGER_PIN = 23;
    private static final int ECHO_PIN = 27;

    //Define execution of read sensors thread
    private volatile boolean shouldRun = true;
    private ReadSensors sensorsTask;

    @Override
    public void startApp() {
    ...
        //Initialize Ultrasound sensor
        hcsr04=new HCSR04Device(TRIGGER_PIN, ECHO_PIN);
        //Start read sensors data thread
        sensorsTask=new ReadSensors();
        sensorsTask.start();
    }

    @Override
    public void destroyApp(boolean unconditional) {
        shouldRun=false;
    ...
        hcsr04.close();
    }

    // Thread to read distance each 5 seconds
    class ReadSensors extends Thread {
        private double distance=0.0;

        @Override
        public void run() {
```

```
            while (shouldRun){
                distance = hcsr04.pulse();
                if (distance>0)
                    System.out.println("Object detected at " +
                                       distance + " cm.");
                I2CUtils.I2Cdelay(5000);
                }
            }
        }

    }
```

Important Notices & Disclaimers

Regarding this Product
The terms and conditions of this sale, including the return policy, (hereinafter “terms”) are limited to those contained herein. Any additional or different terms in any forms delivered by Purchaser are hereby deemed to be material alterations and notice of objection to them and rejection of them is hereby given.

BY PURCHASING AND USING THIS PRODUCT, PURCHASER AGREES TO BE BOUND BY AND ACCEPTS THESE TERMS.
Tenetics, LLC, (hereinafter, “Tenetics” or “TENETICS” or “Company”) owns, and will continue to own any and all intellectual property and/or proprietary rights, including, but not limited to, copyrights, patents, trade secrets,. trademarks, and/or otherwise that are associated to, related to, and/or otherwise a part, element, and/or component of the products that you have purchased, which are protected by United States laws and any applicable international laws, treaties, and conventions concerning, but not limited to, patents, trademarks, copyrights, trade secrets and/or any other legal or equitable doctrines of law that grant the originator of works exclusive rights thereto (the “Intellectual Property”). The Intellectual Property is being non-exclusively licensed for its intended use and neither assigned, exclusive licensed, nor sold under this Agreement. All other rights, uses, and licenses that are contrary to the above are hereby expressly disclaimed. Tenetics reserves the right to make changes without further notice to any products to improve reliability, function or design.

Twelve (12) Month Limited Warranty; Disclaimers & Limitation of Liability

Tenetics warrants only to the purchaser of the Product from Tenetics (the "Customer") that this product purchased from Tenetics (the "Product") will be free from defects in materials and workmanship under the normal use and service for which the Product was designed for a period of twelve (12) months from the date of purchase of the Product by the Customer. Customer's exclusive remedy under this Limited Warranty shall be the repair or replacement, at Company's sole option, of the Product, or any part of the Product, determined by Tenetics to be defective. This Limited Warranty does not extend to any damage by reason of: (a) alteration, accident, abuse, neglect or misuse or improper or inadequate handling; (b) improper or inadequate wiring utilized or installed in connection with the Product; installation, operation or use of the Product not made in strict accordance with the specifications and written instructions provided by Tenetics; (c) use of the Product for any purpose other than those for which it was designed; (d) ordinary wear and tear; (e) disasters or Acts of God; (f) unauthorized attachments, alterations or modifications to the Product; (g) the misuse or failure of any item or equipment connected to the Product not supplied by Tenetics; and/or (h) improper maintenance or repair of the Product; or any other reason or event not caused by Tenetics.

This Limited Warranty is the sole warranty offered by Tenetics with respect to this Product. Tenetics does not assume any other liability in connection with the sale of the Product (see below). No representative of Tenetics is authorized to extend this Limited Warranty or to change it in any manner whatsoever. No warranty applies to any party other than the original Customer.

TENETICS HEREBY DISCLAIMS ALL OTHER WARRANTIES, WHETHER WRITTEN OR ORAL, EXPRESS OR IMPLIED BY LAW OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF FITNESS FOR A PARTICULAR USE OR PURPOSE OR AS TO MERCHANTABILITY OR NON-INFRINGEMENT. CUSTOMER'S SOLE REMEDY FOR ANY DEFECTIVE PRODUCT WILL BE AS STATED ABOVE, AND IN NO EVENT SHALL TENETICS OR ITS AFFILIATES BE LIABLE FOR ANY DIRECT, INDIRECT, PUNITIVE, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES THAT RESULT FROM PRODUCTS OR SERVICES BOUGHT BY PURCHASER, WHETHER THE ALLEGED LIABILITY IS BASED ON CONTRACT, TORT, NEGLIGENCE, STRICT LIABILITY, OR ANY OTHER BASIS, EVEN IF TENETICS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE. ***BY WAY OF EXAMPLE, AND WITHOUT LIMITATION, TENETICS SHALL NOT BE LIABLE FOR ANY DAMAGE THAT IS THE RESULT OF THE APPLICATION OR USE OF TENETIC’S PRODUCT LINE IN CONJUNCTION WITH ANY TYPE OF ELECTRONIC MANAGEMENT, HEALTH AND/OR SECURITY SYSTEM INCLUDING BUT NOT LIMITED TO SYSTEMS THAT MONITOR AND/OR CONTROL (1) PROPERTY ACCESS POINTS SUCH AS DOORS OR WINDOW, (2) HUMAN MEDICAL CONDITIONS, (3) ELECTRONICALLY OPERATED WIRELESS WATER VALVE DEVICES; (4) CLIMATE CONTROL DEVICES, (5) HOME APPLIANCES (5) ETC.*** ANY DAMAGES THAT TENETICS IS REQUIRED TO PAY FOR ANY AND ALL CAUSES, SHALL BE LIMITED IN AMOUNT TO THE PAYMENTS MADE BY THE PURCHASER TO TENETICS FOR THE SPECIFIC PRODUCTS OR SERVICE TO WHICH TENETIC’S LIABILITY RELATES. PURCHASER IS AWARE BOTH ORALLY AND IN WRITING HEREIN, AS TO THIS MATTER. TENETICS AND ITS DIRECTORS, OFFICERS, EMPLOYEES, SUBSIDIARIES AND AFFILIATES SHALL NOT BE LIABLE FOR ANY DAMAGES ARISING FROM ANY LOSS OF EQUIPMENT, LOSS OR DISTORTION OF DATA, LOSS OF TIME, LOSS OR DESTRUCTION OF SOFTWARE OR OTHER PROPERTY, LOSS OF PRODUCTION OR PROFITS, OVERHEAD COSTS, CLAIMS OF THIRD PARTIES, LABOR OR MATERIALS, PENALTIES OR LIQUIDATED DAMAGES OR PUNITIVE DAMAGES, WHATSOEVER, WHETHER BASED UPON BREACH OF WARRANTY, BREACH OF CONTRACT, NEGLIGENCE, STRICT LIABILITY OR ANY OTHER LEGAL THEORY, OR OTHER LOSSES OR EXPENSES INCURRED BY THE CUSTOMER OR ANY THIRD PARTY.

Return Policy

Any used or opened items, after thirty (30) days from purchase, cannot be returned for credit. Tenetics works hard to provide only quality products and to meet the needs and expectations of our clients. Therefore, we offer a **30-Day Money Back Guarantee** for this product. If for any reason you are not completely satisfied with the product or it just does not meet your expectations, simply return it to the retailer where the unit was purchased within 30 days from the date of the purchase for a full refund. Yet before asking for a refund, please consider contacting our technical department or visit our online forums for resolutions to most technical questions or difficulties. For speedy and full credit, please ensure that the returned product adheres to the following conditions: (1) it is returned within **30 days** from the invoice date, (2) item is in saleable and in original condition, (3) item is in original packaging, and carefully packed in a shipping box, and (4) a copy of the original invoice is included with the return.