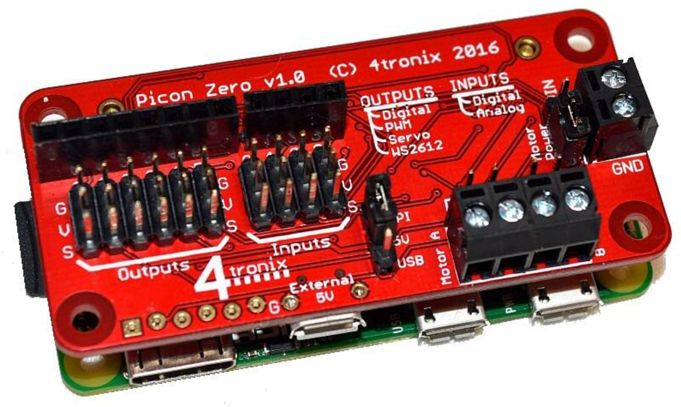
25 Mar 2016 Picon Zero – Intelligent Robotics for Raspberry Pi

Posted By admin In Robotics for Raspberry Pi Comments Off

Picon Zero – Intelligent Robotic Controller for Raspberry Pi



Purchase Picon Zero here

Worksheets now available – see the bottom of this page

Description

The Picon Zero is an add-on PCB for the Raspberry Pi. It is physically the same size as a Raspberry Pi Zero and so is ideal as a pseudo-Hat (pHat) for the Pi Zero. However, it can of course be used on any Raspberry Pi with a 40-pin GPIO connector. Currently that includes the A+, B+, 2B and 3B.

As well as 2 full H-Bridge motor drivers, the Picon Zero has a number of Input and Output pins that can be configured in a variety of ways, allowing you to easily add analog inputs or neopixel outputs to your Raspberry Pi without any complicated software and kernel specific drivers. It also provides an interface for an HC-SR04 ultrasonic distance sensor and opens up 5 GPIO pins from the Raspberry Pi for you to use as you see fit.

The Picon Zero is shipped with all components, including the headers and screw terminals, fully soldered. This saves you doing any soldering and allows you to use it directly from the box

Specification

- Pseudo-Hat format PCB, 65 x 30 mm with gold plated pads and a ready-fitted 40-pin female header (if using a Pi-Zero, ensure you have soldered a male header in place)
- 2 full H-Bridge motor drivers. Can drive up to 1.5A continuously (2A peak) per channel, at from 3 – 11V
- Each motor output has both a 2-pin screw terminal and a 2-pin male header. You choose which to use, depending what connectors you have on the motors
- The power for the motors can be taken from an external power source (3V to 11V) or from the Picon Zero's own 5V
- The Picon Zero's 5V can be selected to be from the Raspberry Pi's 5V line, or from a USB connector on the Picon Zero. This allows you to use 2 USB battery banks: one to power the Pi and the other to power the servos and motors on the Picon Zero
- 4 Inputs that can accept up to 5V. These inputs can be configured as:
  - Digital inputs
  - Analog inputs (10 bits, 0..1023)
  - DS18B20
  - DHT11 (TBD)
- 6 Outputs that can drive 5V and be configured as:
  - Digital Output (High or Low)

    - PWM Output (Pulse Width Modulation from 0% to 100%)
    - Servo (drives a standard servo through 180 degrees)
    - Neopixel WS2812 (Can drive up to 64 neopixels in a chain. Only available on output 5)

- All Inputs and Outputs use GVS (Ground, Volts, Signal) 3-pin male headers. This is a common standard for 3-pin sensors and servos and allows direct connection without additional wires
- 4-pin female header that can connect directly to an HC-SR04 ultrasonic distance sensor. You can plug the sensor directly into this header for an ultra-compact, object avoiding robot
- 8-pin female header for Ground, 3.3V, 5V and 5 GPIO signals allowing users to add their own additional features

Hardware Configuration

Picon Zero has 2 jumpers for setting the hardware configuration. Make sure you have them set in the correct positions for your requirements

1. JP1 – **Board 5V Selector**. This jumper selects where to get the 5V power from for the Picon Zero Outputs. The options are:
    1. Jumper at the top between RPI and 5V. The 5V power for the board is taken from the Raspberry Pi 5V pins on the GPIO connector. With low power output devices and low power 5V motors, this allows you to power all devices with a single 5V power input to the Raspberry Pi
    2. Jumper at the bottom between USB and 5V. The 5V power is taken from the micro-USB connector on the Picon Zero. For higher power out devices, you can provide extra power through the micro-USB connector on the board
2. JP2 – **Motor Power Selector**. This jumper selects where to get the power for the motors from. The two options are:
    1. Jumper at the top between Vin and MotorPower. The motors are driven from the 2-pin screw terminal. This voltage can be anything from 3V to 11V. Useful for motors that require a voltage different from 5V, or that require more current than is available on either of the USB input connectors
    2. Jumper at the bottom between 5V and MotorPower. The motors are driven from the board's 5V power signal. Remember that this in turn can be selected to be either the Raspberry Pi or the micro-USB connector on the Picon Zero.

Programming

Raspberry Pi Setup

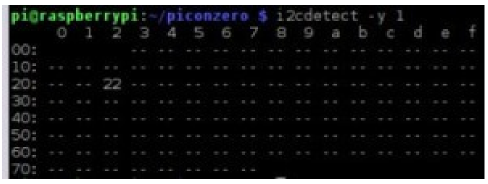
The Picon Zero is an I2C device, so you must ensure that your Raspberry Pi is setup to use I2C and smbus correctly:

- sudo apt-get install python-smbus python3-smbus python-dev python3-dev
- sudo nano /boot/config.txt
    - and add the following 2 lines to the end of the file:
    - dtparam=i2c1=on
    - dtparam=i2c_arm=on
    - Press <ctrl-x> and accept the default prompts to save the file
- sudo reboot

Plug the Picon Zero onto the Pi and run

- i2cdetect -y 1

If all is well, you will see the Picon Zero showing up as address 22 as below:



```
pi@raspberrypi:~/piconzero $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- 22 -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Python Example Installation

The Picon Zero is provided with a python library which allows very simple access to all the features in an intuitive and consistent manner. A separate small library handles the ultrasonic which is on a GPIO pin and not part of the onboard micro-controller.

Download the Picon Zero library itself from here

You can also download directly both libraries and example files by typing the following into the terminal window. Open LXTerminal and type:

- wget http://4tronix.co.uk/piconz.sh -O piconz.sh

- bash piconz.sh

That will create a **piconzero** folder in your home folder, with the libraries and example files:

- **piconzero.py** library module for Picon Zero
- **hcsr04.py** library module for ultrasonic sensor
- **version.py** prints the board type and firmware version
- **motorTest.py** tests the motor functions using the arrow keys on the keyboard
- **ioTest.py** demonstrates the use of an analog input (potentiometer on Input 0) used to control various forms of output
- **sonarTest.py** demonstrates taking distance readings if you have an HC-SR04 ultrasonic sensor attached
- **tempTest.py** demonstrates reading the temperature from an attached DS18B20 sensor
- **10linesTest.py** shows a simple 10 (or 11) line program that reads an analog input and outputs to various devices (same as **ioTest.py** but without some bells and whistles). Simply to show how easy it can be to use the Picon Zero
- **pixelTest.py** flashes all the attached neopixels from White to Off and back again
- **servoTest.py** uses the arrow keys to move 2 servos (pan and tilt) and the G, H keys to move a third servo (grabber claw)

Basics of Programming Picon Zero

A python program should start by importing the library, eg.

```
import piconzero as pz
```

Notice that we have imported the whole library and then called it 'pz' for short. Now whenever we use any of the library functions we need to put the 'pz.' in front so python knows where to get the functions from.

Then you should call the initialising function to clear any previous settings that may have been left from a previous program closing untidily

```
pz.init( )
```

Now the library is set up and ready to go. Ensure you set the configuration required for each pin before using it (unless you are using the default configurations, when it is not necessary)

At the end of your program you should clean up the system and leave it in a nice state for the next program to use

```
pz.cleanup( )
```

The classified complete list of library functions follows

General Functions

- init ( ): This clears all the Inputs and Outputs back to their default configurations, stops the motors, disables any servos and switches off neopixels. It also sets a few internal variables that allows the Picon Zero to keep track of what it is doing
- cleanup ( ): This is pretty much the same as init() but is used at the end of the program to clean up all the outputs for the next program
- getRevision ( ): This returns a list with board revision and firmware revision. It allows you to check whether the firmware is capable of any new functions (assuming new functions will be added later)

Motor Functions

- forward (speed): Moves both motors forward at selected speed. Speed can vary from 0 to 100. (it can also be negative, which would cause the robot to reverse)
- reverse (speed): Moves both motors in reverse ar selected speed. Speed can vary from 0 to 100 (can also be negative)
- spinLeft (speed): Spins left on the spot at selected speed. Speed can vary from 0 to 100 (can also be negative)
- spinRight (speed): Spins right on the spot at selected speed. Speed can vary from 0 to 100 (can also be negative)
- stop ( ): Stops both motors
- setMotor (motor, speed): This is the primitive motor function, called by all the others. Motor can be either 0 (Motor A) or 1 (Motor B). Speed can vary from -100 (full reverse) to +100 (full forwards). The speed number is a percentage.

Input Functions

- setInputConfig (channel, config): This sets the configuration of the selected Input channel. There are 4 Input channels (0 to 3). The config parameter determines if the channel is:
  - 0: Digital (0 or 1) – this is the Default input configuration [there is a 3rd parameter for version 08 firmware and later: Pullup, set to False by default, but can be set to True which will provide a 10K internal pullup resistor on the selected channel

  - 1: Analog (0 to 1023) - read analog values from selected channel. 0V -> 0; 5V -> 1023
  - 2: DS18B20 [Available from firmware revision 06]. Reads a 18B20 temperature sensor

- readInput (channel): Gets the value of the selected input channel. This will return:
  - o (False) or 1 (True) if the channel is Digital
  - A value from 0 to 1023 if the channel is analog
  - A temperature in centigrade if the channel is a DS18B20

Output Functions

- setOutputConfig (channel, config): This sets the configuration of the selected Output channel. There are 6 Output channels (0 to 5) and these can be set as follows:
  - 0: Digital (Low or High) – this is the Default output configuration
  - 1: PWM (0 to 100% duty cycle)
  - 2: Servo (0 to 180 degrees)
  - 3: Neopixel WS2812B (individually address any pixel and set values of 0..255 for each colour). Only Output channel 5 can be set to this configuration value

- setOutput (channel, data): Sets the output channel with the data entered – Digital, PWM and Servo data only.
- setPixel (pixel, red, green, blue): Sets the selected pixel (0 to 63) with the selected red, green and blue values (0 to 255).
- setPixel(pixel, red, green, blue, update): The optional parameter update (defaults to True), determines if the updated pixel data is updated immediately or not. Updating the pixel strip strip takes time, so if you are changing a number of pixels at once, it is best to switch off updates until they have all been changed, then use the updatePixels ( ) function
- setAllPixels (red, green, blue): Sets all pixels with the selected red, green and blue values (0 to 255) [Available from firmware revision 07]
- setAllPixels (red, green, blue, update): The optional parameter update (defaults to True), determines if the updated pixel data is updated immediately or not. Updating the pixel strip strip takes time [Available from firmware revision 07]
- updatePixels ( ): Causes an immediate update of all the pixels from the latest data
- setBrightness (brightness): Sets the overall brightness (0 to 255) of the neopixel chain. All RGB values are scaled to fit into this max value. Default is 40

Worksheets

These downloadable worksheets take you through using various features of the Picon Zero in simple, self-contained, stages.

1. Worksheet 01 – Setting up your Raspberry Pi for Picon Zero
2. Worksheet 02 – Configuring and controlling motors
3. Worksheet 03 – Getting Input from Buttons and Switches
4. Worksheet 04 – First steps with analog inputs
5. Worksheet 05 - Controlling servos
6. Worksheet 06 - Controlling the brightness of LEDs (PWM)
7. Worksheet 07 - Reading temperatures from a DS18B20
8. Worksheet 08 - Driving WS2812B RGB LEDs (aka NeoPixels)

← Previous Post Next Post →

Comments are closed.