ZIO, Motherboard

User Manual

1.4, Jul 2013

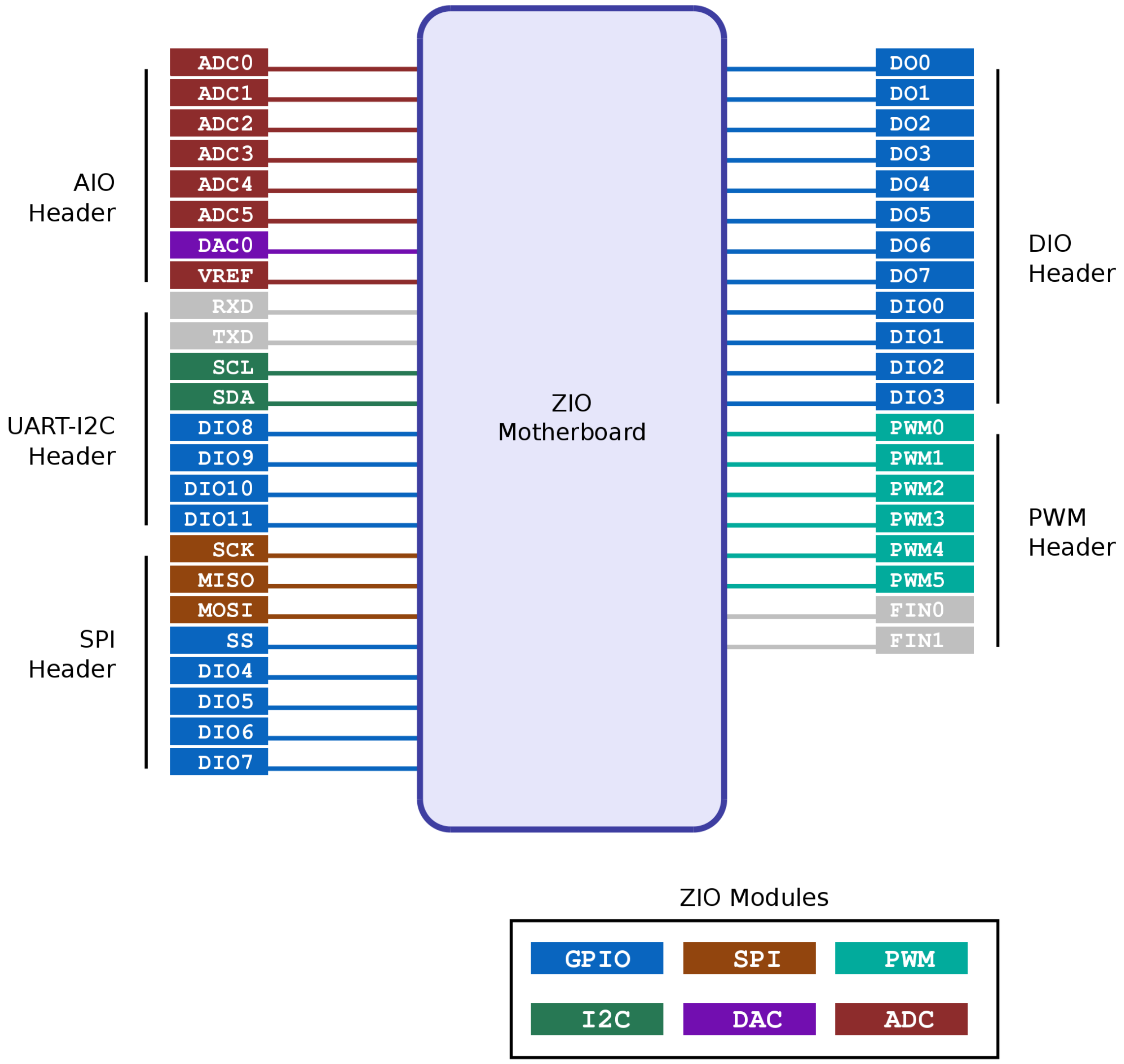
Table of Contents

Chapter 1. Introduction

1. Philosophy

- Move development from micro-controllers to PC
- Use high level languages like Python and Java.
- Extend the IO capabilities of the PC.
- Rapid prototype development.

Figure 1.1. Block Diagram



2. Product Features

- Connects to PC through USB
- Interfaces - GPIO, PWM, ADC, DAC, SPI, I²C
- Host-side API for programming the ports
- APIs available for Java and Python
- API documentation for easy reference
- Port interfacing guidelines for common scenarios

- GUI based Control Panel to explore the board
- On-field firmware upgrade through USB

Chapter 2. Connecting to ZIO

In this chapter we will describe the connector used for the ZIO ports and the pins found on each of the ports. The ZIO has 5 FRC connectors.

1. DIO
2. AIO
3. PWM
4. UART-I²C
5. SPI

1. SPI Pinmap

The `SPI` header is terminated with serial peripheral interface (SPI) bus, 4 general purpose IO and power supply. Add-on boards with SPI interface and general purpose IOs like MMC/SD card,EEPROM etc., can be connected through this header.

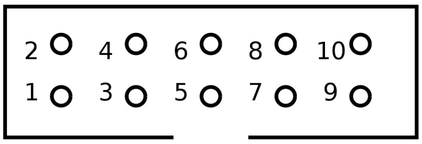


Table 2.1. SPI Header

Pin #	Header Signal	Signal Type
1	`VCC`	+5V
2	`SCK`	TTL Out
3	`MISO`	TTL In [1]
4	`MOSI`	TTL Out
5	`SS`	TTL Out
6	`DIO0`	OC [2]
7	`DIO1`	OC [2]
8	`DIO2`	OC [2]
9	`DIO3`	OC [2]
10	`GND`	Ground

[1] 5V tolerant Input

[2] Open collector, with 5V pull-up

`VCC` (Pin 1)
: This is the +5V power supply for the external devices. The supply has a total current limit of 200mA when powered through USB.

`SCK` (Pin 2)
: This is Serial Clock signal.

`MISO` (Pin 3)
: This is the Master Input, Slave Output signal.

`MOSI` (Pin 4)
: This is the Master Output, Slave Input signal.

`SS` (Pin 5)
: This is the SPI chip select signal.

`DIO` (Pin 6-9)
: These are digital input/output signals. These lines can be used to interface any extra signals required for a SPI devices like SD Card, etc., or can be used as chip selects for four other devices. The signals are pulled up to 5V, through a 10K resistor.

GND (Pin 10) — This is the ground signal. All other signals are referenced to the this signal.

2. UART-I2C Pinmap

The UART-I2C header is terminated with serial communication signals, I²C signals and power supply. Add-on boards, with different functionalities, can be connected through this header.

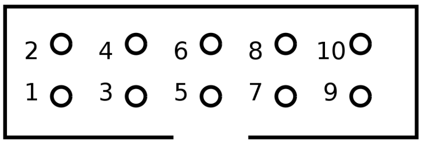


Table 2.2. UART-I2C Header

Pin #	Header Signal	Signal Type
1	VCC	+5V
2	RXD	TTL In [1]
3	TXD	TTL Out
4	SCL	OC [2]
5	SDA	OC [2]
6	DIO0	OC [2]
7	DIO1	OC [2]
8	DIO2	OC [2]
9	DIO3	OC [2]
10	GND	Ground

[1] 5V tolerant input

[2] Open collector, with 5V pull-up

VCC (Pin 1) — This is the +5V power supply for the external devices. The supply has a total current limit of 200mA when powered through USB.

RXD (Pin 2) — This is receive line of serial IO.

TXD (Pin 3) — This is transmit line of serial IO.

SCL, SDA (Pin 4, 5) — These are I²C bus signals(clock, data), and can be used to connect I²C devices. The signals are pulled up to 5V, through a 4.7K resistor.

DIO (Pin 6-9) — These are digital input/output signals. These pins can be used for hand-shake and flow control signals like DTR, RTS, CTS, etc. The signals are pulled up to 5V, through a 10K resistor.

GND (Pin 10) — This is the ground signal. All other signals are referenced to this signal.

3. DIO Pinmap

The DIO header is terminated with GPIO signals, along with power supply. Add-on boards, with different functionalities, can be connected through this header.

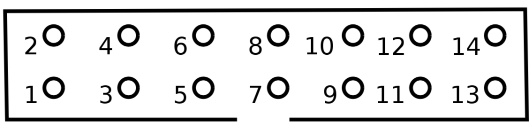


Table 2.3. DIO Header

Pin #	Header Signal	Signal Type
1	VCC	+5V
2	DO0	TTL Out
3	DO1	TTL Out
4	DO2	TTL Out
5	DO3	TTL Out
6	DO4	TTL Out
7	DO5	TTL Out
8	DO6	TTL Out
9	DO7	TTL Out
10	DIO8	OC [2]
11	DIO9	OC [2]
12	DIO10	OC [2]
13	DIO11	OC [2]
14	GND	Ground

[1] 5V tolerant input

[2] Open collector, with 5V pull-up

`VCC` (Pin 1)
: This is the +5V power supply for the external devices. The supply has a total current limit of 200mA when powered through USB.

`DO` (Pin 2-9)
: These are digital output signals. The signal is a 5V logic signal, but the output can drive a 5V device or 3.3V device with 5V tolerance.

`DIO` (Pin 10-13)
: These are digital input/output signals. The signal is a 5V logic signal, but the output can drive a 5V device or 3.3V device with 5V tolerance. These signals can be used as control and hand-shake signals. The signals are pulled up to 5V, through a 10K resistor.

`GND` (Pin 14)
: This is the ground signal. All other signals are referenced to this signal.

4. PWM Pinmap

The PWM header is terminated with 5 pulse width modulation signals and power supply. Add-on boards like LED control, motor control can be connected through this header.

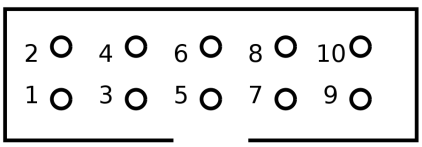


Table 2.4. PWM Header

Pin #	Signal Name
1	VCC
2	PWM 0
3	PWM 1
4	PWM 2
5	PWM 3

Pin #	Signal Name
6	`PWM 4`
7	`PWM 5`
8	`Freq-In 0`
9	`Freq-In 1`
10	`GND`

`VCC` (Pin 1)
: This is the +5V power supply for the external add-on boards. The supply has a total current limit of 200mA when powered through USB.

`PWM` (Pin 2 - 7)
: These are PWM output signals. The PWM signal when active produces a stream of pulses whose width can be controlled through software. An important parameter of a PWM signal is the **duty cycle**. The duty cycle is defined as the ratio between the pulse duration and pulse period of a rectangular waveform.

The PWM signal can be used to control the power delivered to a load, by controlling the duty cycle of the PWM signal. PWM signals are generally used for Motor speed control, LED brightness control, power supplies and wave form generation.

The PWM signal is a 5V CMOS/TTL output.

`Freq-In` (Pin 8, 9)
: These are input signals, used for event counting and frequency measurement. These signals are 5V tolerant CMOS/TTL inputs.

5. `AIO` Pinmap

The `AIO` header is terminated with 6 ADC channels, 1 DAC and power supply. Sensors can be connected to this header.

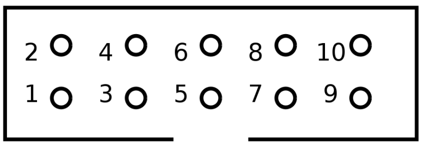


Table 2.5. `AIO` Header

Pin #	Signal Name
1	`VCC`
2	`ADC 0`
3	`ADC 1`
4	`ADC 2`
5	`ADC 3`
6	`ADC 4`
7	`ADC 5`
8	`DAC 0`
9	`VREF-OUT`
10	`GND`

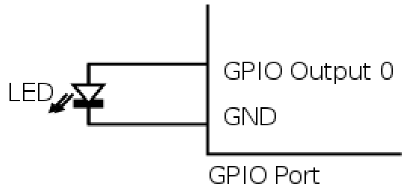
`VCC` (Pin 1)
: This is the +5V power supply for the external add-on boards. The supply has a total current limit of 200mA when powered through USB.

`ADC` (Pin 2-5)	These are analog input signals connected to a 10-bit Analog-to-Digital Converter. The maximum analog input voltage is 3.0V.
`DAC` (Pin 8)	This is analog output signal connected to a 10-bit Digital-to-Analog Converter. The voltage level can vary from 0V to 5V.
`VREF-OUT` (Pin 9)	This is the ADC's reference voltage.
`GND` (Pin 10)	This is the ground signal. All other signals are referenced to this signal.

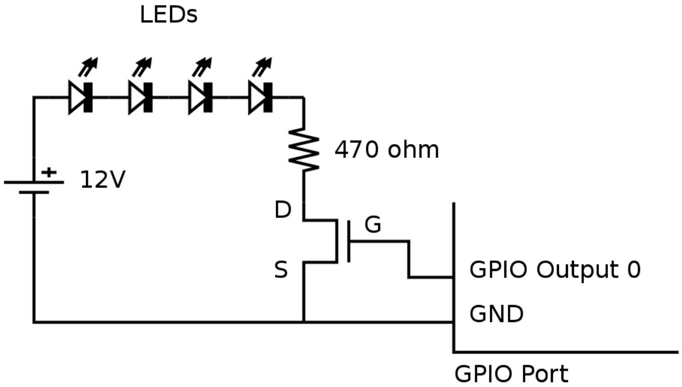
Chapter 3. ZIO Recipes

1. GPIO Port

Connecting LEDs. Connect the anode of the LED to an `Output` signal, and the cathode to GND. The built-in series resistor is sufficient to limit the current.



Connecting series of LEDs. Since the `Output` signal can not provide sufficient power for more than one LED, and external power source is to be used. And the power supply can be controlled using a MOSFET switch.



The circuit diagram for connecting a series of LEDs is shown above. The following formula can be used to calculate the resistance for the current limiting resistor. (The voltage drop across the MOSFET is considered to be negligible.)

$R = (V_{cc} - NV_d) / I_d$

Where,

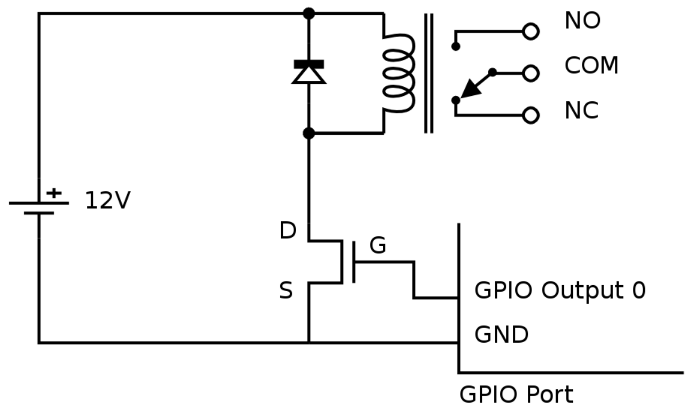
V_d	Voltage Drop Across LED
N	No. of LEDs
I_d	Current for the required brightness
V_{cc}	LED supply voltage
R	Current Limiting Resistor

As an example, for the following parameters,

- V_{cc} = 12V
- I_d = 11mA
- N = 4

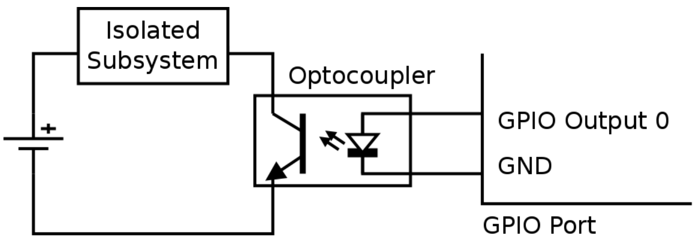
the calculated current limiting resistance is 470 ohms.

Connecting relays. Relays are used to control a high-voltage/high-current circuit with a low-voltage/ low-current signal. A relay can be connected to the ZIO through a MOSFET as shown in the following circuit diagram.

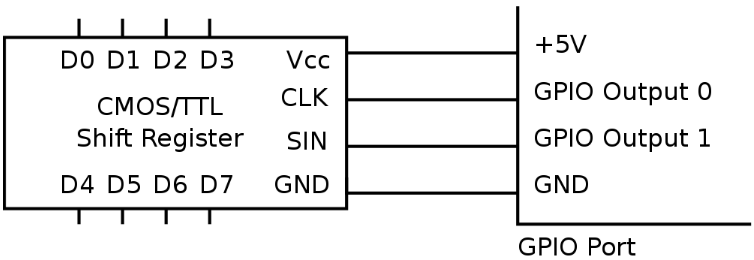


Isolating outputs using opto-coupler. There are situations in which signals from one subsystem need to be electrically isolated from another subsystem in an electrical equipment. For example, a microcontroller operating at 5V, controls the power to a load operating at 230V AC. In such situations, the microcontroller needs to be electrically isolated from the high voltage section, using a opto-coupler.

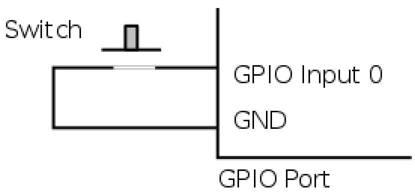
Note that, though relays can also be used for this purpose, they are generally bulky, slow, unreliable, and power hungry.



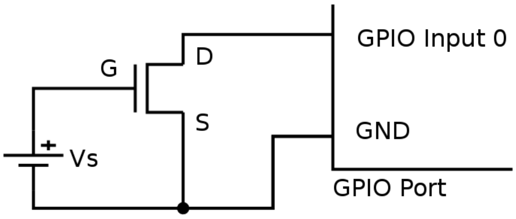
Connecting to CMOS/TTL inputs. CMOS/TTL inputs can be directly connected to the `Output` signal. An example of shift register connected to the `Output` signals is shown in the following circuit diagram.



Connecting Switches. Switches can be directly connected between the `Input` and `GND`. When the switch is pressed the `Input` signal will be low, and when the switch is released the `Input` signal will be become high due to the built-in in pull-up resistor.

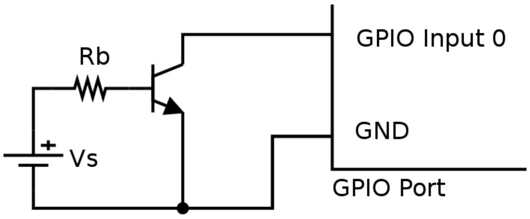


Detecting External Voltage. Any external voltage input can be connected to the ZIO `Input` signal through a MOSFET or a BJT. An example circuit using a MOSFET is shown below.



If the input voltage (V_S) is greater than the threshold voltage of the MOSFET, the `Input` signal will be low, or else it will be high.

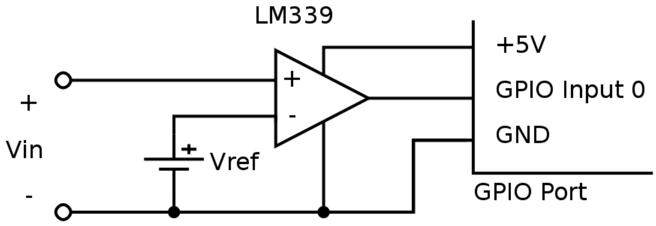
An example circuit using a BJT is shown below.



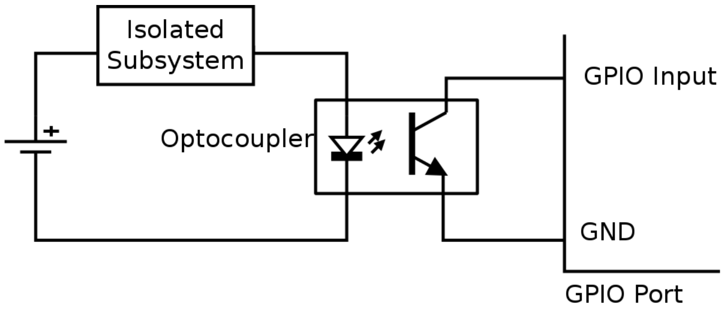
If the input current (I_S) is greater than (I_t = 0.5mA / h_{FE}), the `Input` signal will be low, or else it will be high. For all practical purposes, a (I_t = 1mA) input current is sufficient to make the `Input` signal go low. The base resistance (R_b) has to be chosen to make the `Input` signal low, when the required input voltage is driven.

$R_b = (V_s - V_{be}) / I_t$

Connecting an Analog Comparator. An analog comparator can be used to identify if the input voltage is larger than a specified reference voltage. Any operational amplifier can be used as a comparator, but a dedicated comparators like LM339 which provide open collector CMOS/TTL outputs are suitable for interfacing with logic circuits. An example circuit is shown in the following diagram.

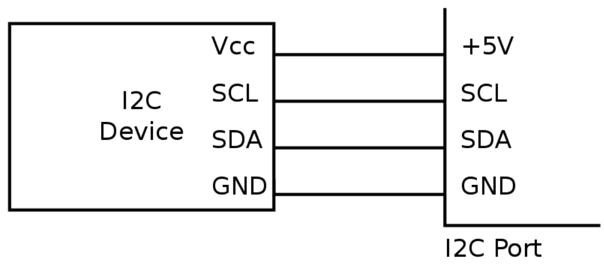


Isolating inputs using opto-coupler. As in the case of outputs, inputs can also be electrically isolated using opto-couplers.

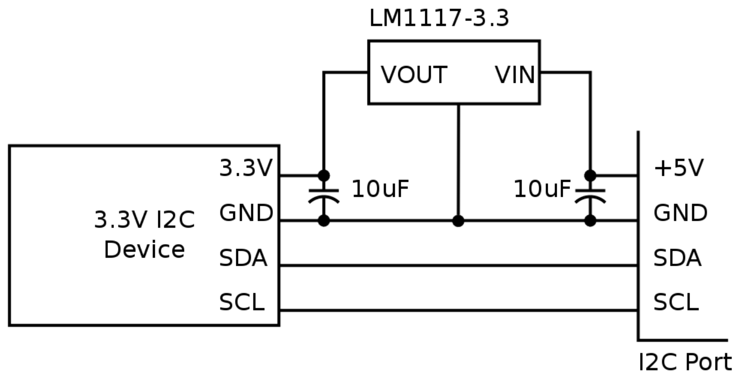


2. I²C Port

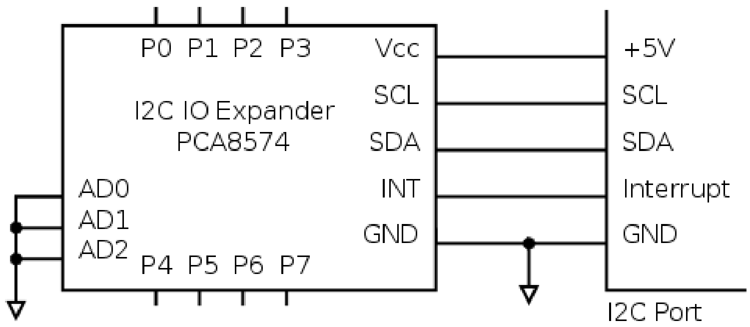
Connecting 5V I²C devices. Since the I²C signal are pulled up to 5V, 5V I²C devices can be directly connected to the I²C port.



Connecting 3.3V I²C devices with 5V tolerance. Any 3.3V I²C device with 5V tolerance can be directly connected to the I²C port. The device can be powered from an external 3.3V supply, or the 3.3V supply can be generated from the `+5V Power` using a regulator. An example circuit with the commonly available LM1117-3.3 regulator is shown below.

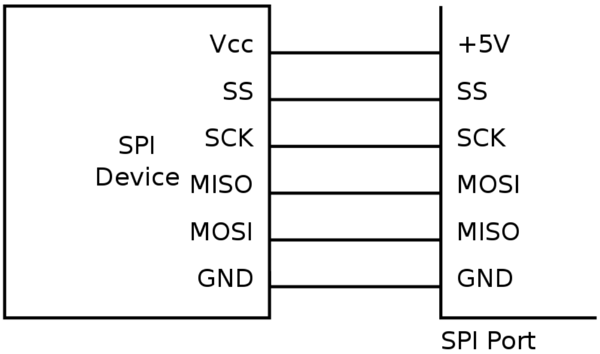


IO Expander. Additional digital inputs and outputs, if required, can be obtained using a I²C IO expander. The PCA8574 provides 8 digital I/O lines, and PCA8578 provides 16 digital I/O lines. An example circuit using the PCA8574, with I²C device address set to `0x20`, is shown below.

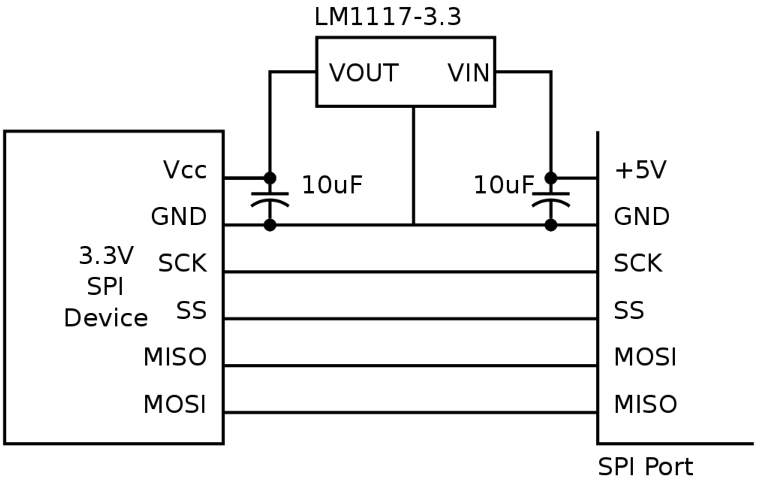


3. SPI Port

Connecting 5V SPI devices. Since the SPI signal are 5V TTL/CMOS signals, 5V SPI devices can be directly connected to the SPI port.



Connecting 3.3V SPI devices with 5V tolerance. Any 3.3V SPI device with 5V tolerance can be directly connected to the SPI port. The device can be powered from an external 3.3V supply, or the 3.3V supply can be generated from the +5V `Power` using a regulator. An example circuit with the commonly available LM1117-3.3 regulator is shown below.



4. Sensor Port

4.1. Resistive Sensors

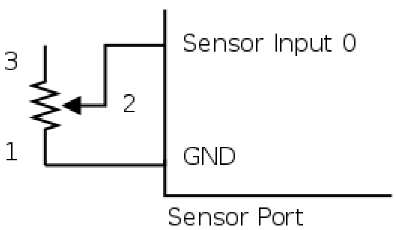
Connecting a Potentiometer. The position of potentiometer can be sensed by connecting the potentiometer to the sensor input as shown in the figure below. When the centre pin 2 of the potentiometer is moved from pin 1 to pin 3, the raw value varies from 0 to N_{max}. Where N_{max} is given by the following formula.

$N_{max} = (0xFFFF \times R_{max}) / (R_{max} + 10K)$

Here,

- R_{max} is the maximum resistance of the potentiometer
- 10K is the internal pull up resistor on the Sensor signal. For more details refer ???.

For a 10K potentiometer, $N_{max} = (0xFFFF \times 10K) / (10K + 10K) = 0x7FFF$



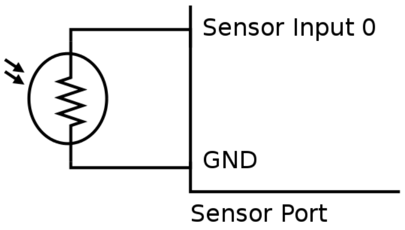
Connecting a Resistive Sensor. Sensors whose resistance varies with the parameter being measured are called resistive sensors. Examples of resistive sensors are Light Dependent Resistor (LDR), thermistor, etc. These sensors can be directly connected between the `Sensor` signal and `GND`. As the parameter being measured varies, the resistance varies accordingly, and the raw value (N) produced is given by the following formula.

N = (0xFFFF x R) / (R + 10K)

Here,

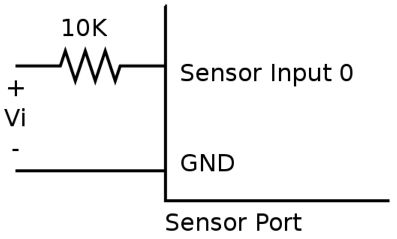
- R is the resistance of the sensor
- 10K is the internal pull up resistor on the Sensor signal. For more details refer ???.

An example circuit, using the LDR, is shown below.

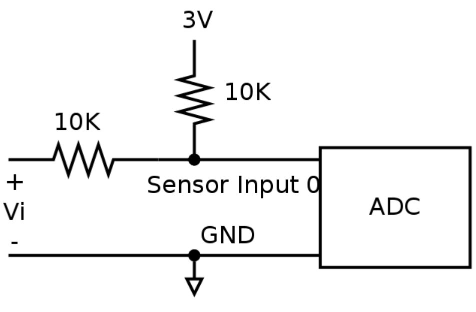


4.2. Voltage Sensors

Voltage measurement, -3V to +3V. Though the ADC input range is 0 to 3V, it is possible to measure voltages between -3V and +3V using a simple circuit. The circuit diagram is shown in the figure below.



To better understand the operation of the circuit, the circuit is shown with the internal pull-up resistor on the `Sensor` signal, in the following diagram.



Using superposition, the voltage at `Sensor 0` is given by the following formula.

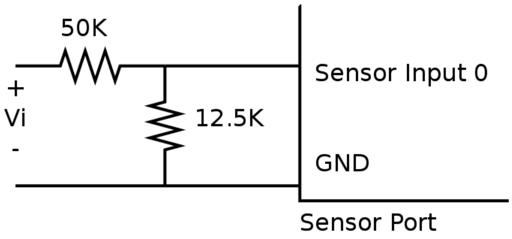
Voltage at `Sensor 0` = 1.5V + V_i / 2

As V_i decreases from 3V to -3V, the voltage at the `Sensor 0` decreases linearly from 3V to 0V, and the raw value from 0xFFFF to 0.

V_i (V)	Voltage at `Sensor 0` (V)	Raw Value
3	3	0xFFFF

V_i (V)	Voltage at `Sensor 0` (V)	Raw Value
0	1.5	0x7FFF
-3	0	0

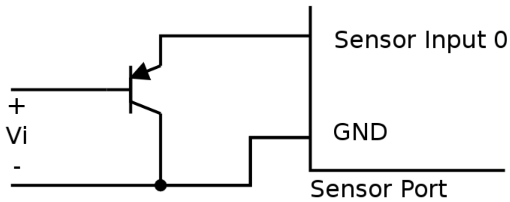
Voltage measurement, -15V to +15V. The following circuit can be used to measure voltages in the range -15V to +15V. The input voltages and the corresponding raw values is shown in the table below.



V_i (V)	Voltage at `Sensor 0` (V)	Raw Value
15	3.0	0xFFFF
0	1.5	0x7FFF
-15	0.0	0

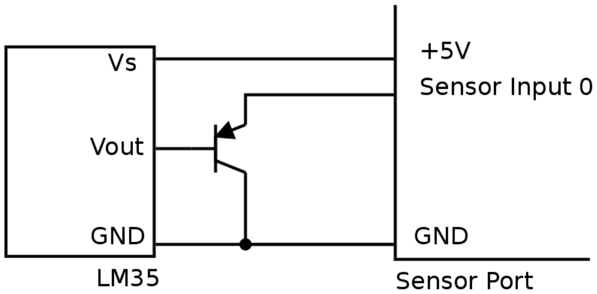
4.3. Non-resistive Sensors

Transistor Buffer. Non-resistive sensors usually generate a voltage signal that varies with the parameter being measured. Such sensors cannot be directly connected to the `Sensor N` signal, due the signal being pulled-up to 3V using a 10K resistor. A transistor buffer can be used to overcome this problem. The transistor isolates the sensor from the pull-up. A transistor buffer circuit is shown below.



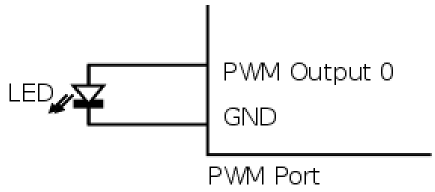
This is a PNP emitter follower, where the emitter voltage is almost equal to the base voltage. For a V_i range of 0 to 4.4V, the voltage at `Sensor 0` is (V_i + 0.6). To compensate for the added 0.6V, subtract 0.6 to the obtained voltage.

Temperature Sensor. The LM35 is an example of an non-resistive sensor. The LM35 produces a voltage that is proportional to the temperature. The voltage output by the LM35, increases by 10mV for every degree Celsius rise in temperature. As the temperature changes from 2^{o}C to 150^{o}C, the voltage rises from 0V to 1.5V. The LM35 can be connected to the Sensor port using the transistor buffer and is shown in the following circuit.

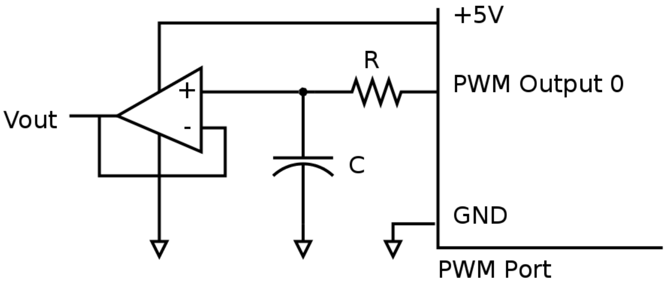


5. PWM Port

LED Brightness Control. An LED can be connected between the `PWM N` signal and `GND` as shown in the following diagram. When the duty cycle is varied the LED brightness varies accordingly.



One Bit DAC. An analog output can be generated from the PWM signal, using a low pass filter circuit. The low pass filter circuit with an op-amp buffer is shown in the following diagram.

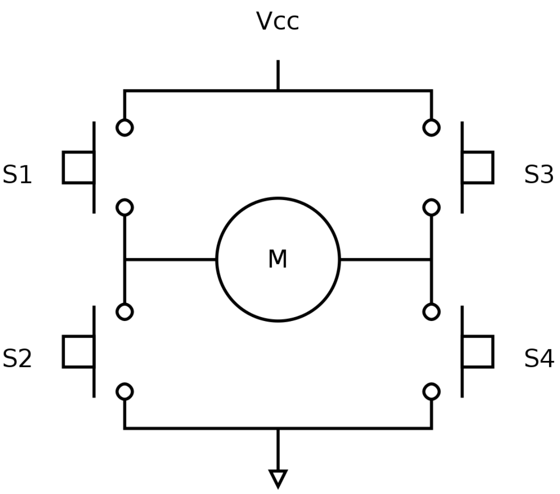


If the analog output has a frequency of F, the PWM frequency should be much higher than F. The values of R and C are given by the following formula.

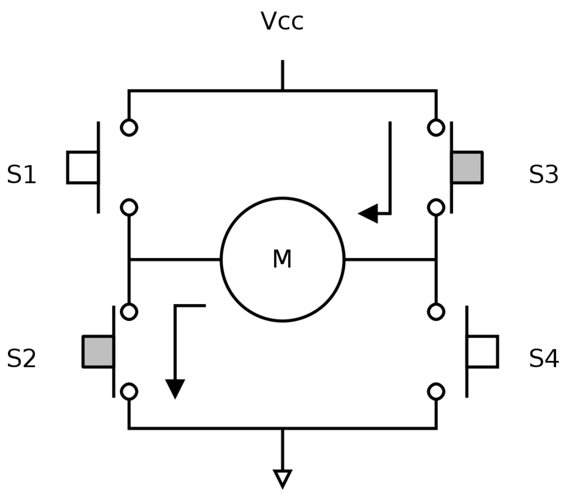
RC = 1 / (2 π F)

For an output frequency of 1kHz, choosing R = 4kohm, C = 0.04uF.

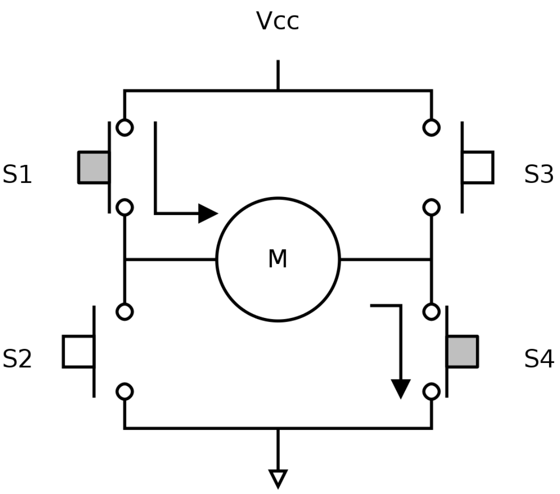
DC Motor Control. A DC motor's speed and direction of rotation can be controlled using the PWM port. The DC motor has to be interfaced through a circuit called the H-Bridge. A simple H-Bridge constructed using switches is shown in the following diagram. By controlling, the switches the motor can be made to rotate forward, reverse, brake, and free run. The various switch states and their effect on the motor is shown in the following table.



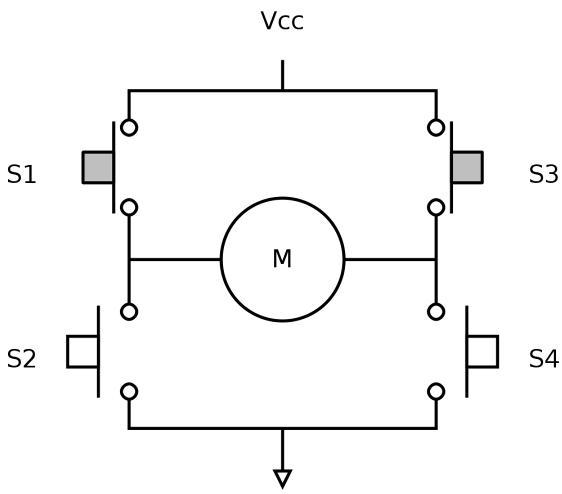
(a) Free Running State



(b) Reverse State



(c) Forward State

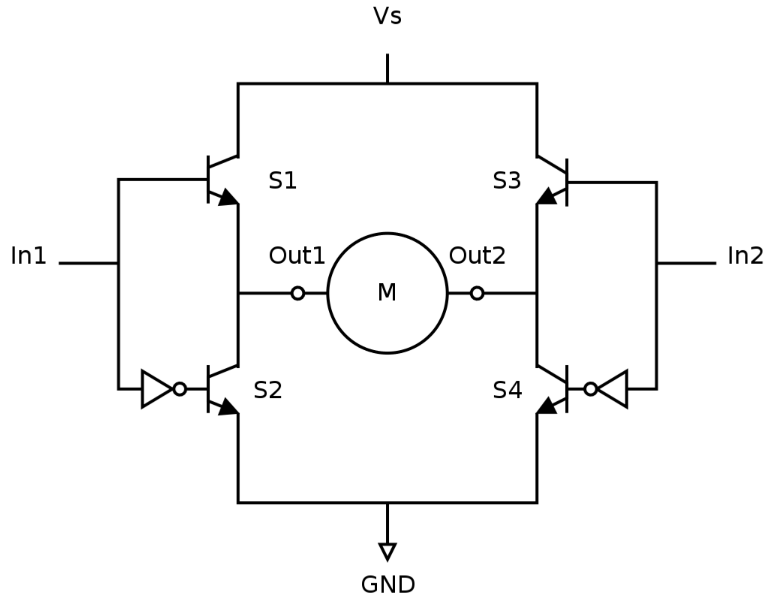


(d) Brake State

S1	S2	S3	S4	Function
0	0	0	0	Free-run
0	1	1	0	Reverse
1	0	0	1	Forward
0	1	0	1	Brake
1	0	1	0	Brake

Forward	The current to flows in one direction through the motor.
Reverse	The current flows in the opposite direction through the motor.
Brake	Applying same voltage to both the terminals, counters the back EMF produced by the motor, and causes it to come to a sudden stop.
Free-run	Power is cut-off from the motor, and the motor free-runs and eventually stops.

To control the motor through digital signals, the switches are replaced by transistors / MOSFETs. Driver ICs like the L298, that implement the H-Bridge can also be used for motor control applications. The block diagram of one half of a L298 is shown in the following diagram.



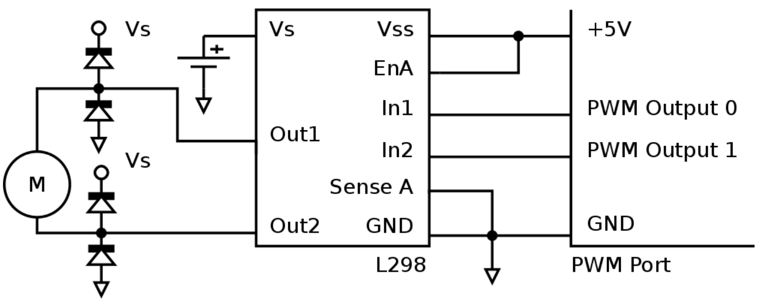
By controlling the inputs, various functions can be selected, as shown in the table below.

In1	In2	Function
0	0	Brake
0	1	Reverse
1	0	Forward
1	1	Brake

When in Forward state or Reverse state, the speed of the motor can be controlled by driving the inputs with a PWM signal

In1 (Duty Cycle)	In2 (Duty Cycle)	Function
0%	0%	Brake
100%	100%	Brake
0%	100%	Reverse, full speed
100%	0%	Forward, full speed
0%	X%	Reverse, speed proportional to duty cycle
X%	0%	Forward, speed proportional to duty cycle

A circuit for interfacing a DC motor to the PWM port using the L298, is shown in the following diagram.



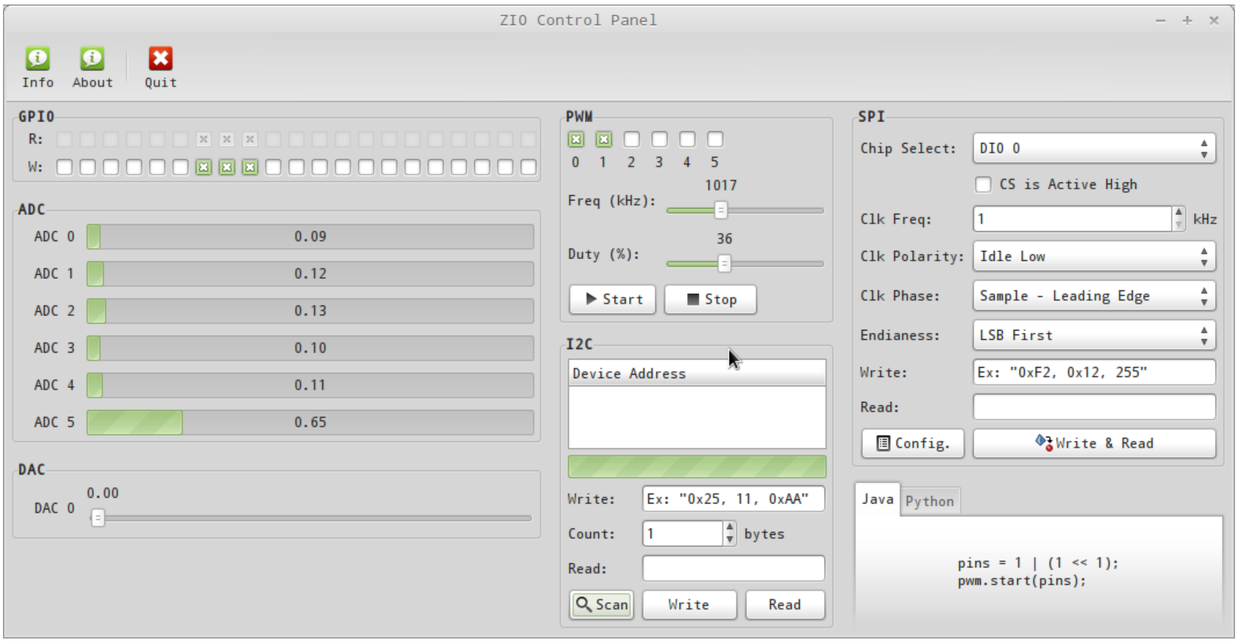
Chapter 4. ZIO Control Panel

The ZIO Control Panel is a GUI application that allows most features of ZIO to be tested without writing code.

When the control panel is started, the application prompts for the serial device name of the ZIO motherboard, as show in Figure 4.1, “Serial Device Input”. Select the serial device and click on `OK`. The control panel window as shown in Figure 4.2, “Control Panel Screenshot” is displayed. The control panel has multiple sub-panels, one for each module.

Figure 4.1. Serial Device Input

Figure 4.2. Control Panel Screenshot



Controlling GPIO Outputs. The GPIO outputs can be controlled by toggling the check box on the `GPIO Out` panel.

Reading GPIO Inputs. The GPIO inputs can be read by inspecting the check box on the `GPIO In` panel.

Reading Sensor Inputs. The Sensor inputs can be read by inspecting the progress bar on the `Sensor` panel.

Controlling PWM Outputs. PWM signals can generated using the controls in the `PWM` panel,

1. Select the PWM channels, by toggling the checkboxes.
2. Set the PWM frequency, in the frequency slider.
3. Set the PWM duty cycle, in the duty cycle slider.
4. Click `Start` to start generating PWM signal.
5. Click `Stop` to stop generating PWM signal.

Controlling I²C Devices. I²C devices can be accessed using the controls in `I2C` panel. To list devices present on the bus,

1. Click on the `Scan` button.
2. Addresses of devices present on the bus is displayed on the list box.

To write to a device,

1. Select the device address.
2. Enter the data bytes to be written in hex, separated by commas, in the `Write` text box.
3. Click on the `Write` button.

To read from a device,

1. Select the device address.
2. Select the no. of bytes to read.
3. Click on the `Read` button.

Controlling SPI Devices. SPI devices can be accessed using the controls in `SPI` panel.

To configure the device,

1. Specify the GPIO output that is to be used as chip select, in the `Chip Select` combo box.
2. If the chip select is active high, select the `CS is Active High` check box.
3. Specify the clock polarity in the `Clk Polarity` combo box.
4. Specify the clock phase in the `Clk Phase` combo box.
5. Specify the endianess in the `Endianess` combo box.
6. Click on `Config.` to select the configuration specified. The has to be done every time the configuration is changed.

To write and read from the device.

1. Specify the list of bytes to be written in the `Write` text box.
2. Click on `Write & Read` to write the specified byte and read an equal no. of bytes.

Equivalent Code. The equivalent code for the currently performed operation is indicated in the Java and Python tabs. This is an easy way to learn the Java and Python API.

Chapter 5. Legal Information

1. Copying

2. Disclaimers

NO WARRANTY. ZILOGIC SYSTEMS' DEVELOPMENT KITS (AND TECHNICAL SUPPORT, IF ANY) ARE PROVIDED "AS IS" AND WITHOUT ANY WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAWS, ZILOGIC SYSTEMS EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT. ZILOGIC SYSTEMS DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN ZILOGIC SYSTEMS' DEVELOPMENT KITS WILL MEET YOUR REQUIREMENTS, OR THAT THE OPERATION WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT DEFECTS IN ZILOGIC SYSTEMS' DEVELOPMENT KITS WILL BE CORRECTED. FURTHERMORE, ZILOGIC SYSTEMS DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE ZILOGIC SYSTEMS' DEVELOPMENT KITS IN TERMS OF THEIR CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY OR MAY BE LIMITED.

Limitation of Liability. Zilogic Systems' development kits are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, not in applications where failure or malfunction of a Zilogic Systems product can resonably be expected to result in personal injury, death or severe property or environmental damage. Zilogic Systems accepts no liability for inclusion and/or use of Zilogic Systems' development kits in such equipment or applications and therefore such inclusion and/or use is at the customer’s own risk.