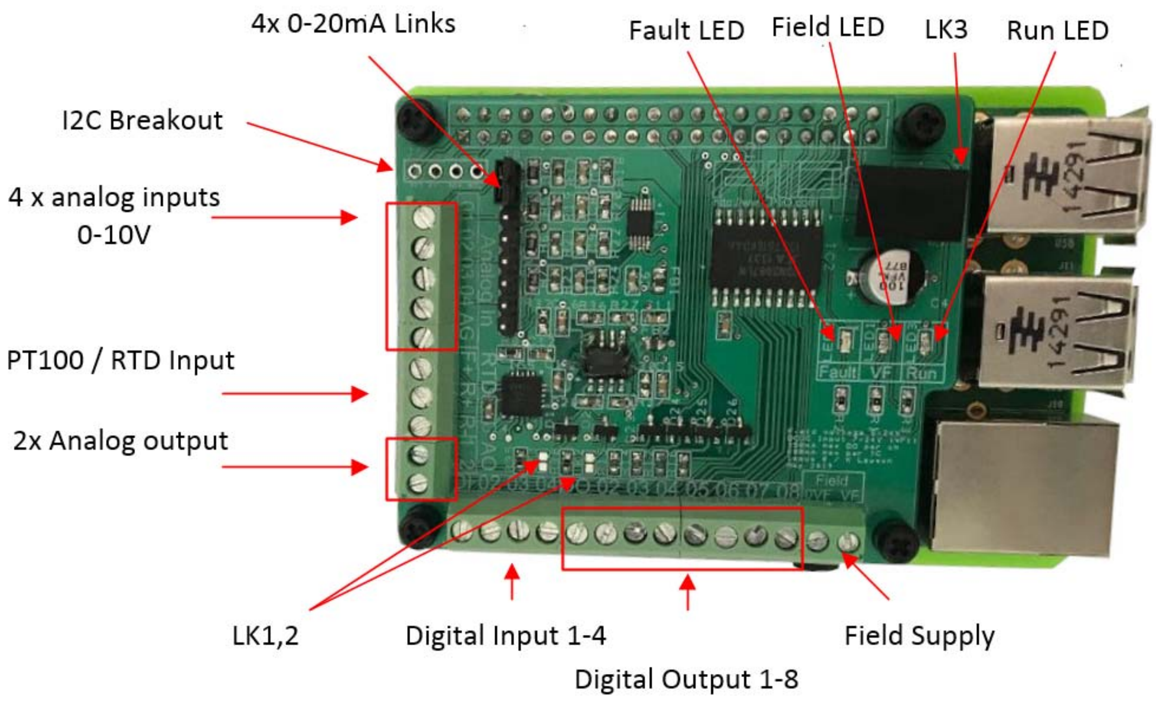
PIO ADIO PCB Manual

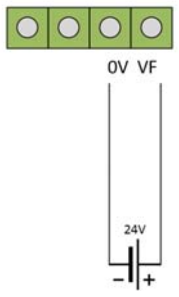
Description

The PiIO ADIO PCB sits on top of a raspberry PI PCB and can be used to interface it to light industrial and test / measurement applications. The board features 8 high side outputs and 4 digital inputs. Additionally there are 4 0-10V / 0-20mA inputs, a 3 wire PT100 input and two 0-10V analog outputs.



Powering the board

The board is powered via the terminal blocks, A supply of 12-24V should be used. Where a DCDC is fitted to the ADIO unit it is not required to provide a 5V micro USB power.



DCDC Not fitted

If you have not chosen to not have the DCDC fitted then then you will need to power the pi using a micro USB. The field supply will be required on the PiIO unit still though.

DCDC fitted

If you have chosen to have the DCDC fitted then it is not required to power the Pi using a micro USB and everything is powered from the field supply.

Field supplies

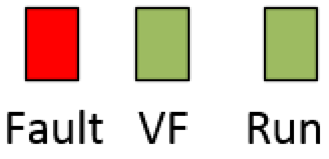
The field supply input is used to power the analog and digital outputs and the optionally fitted on-board DCDC for the PI.

On-board DCDC

A 1A or 2A DCDC Power supply is optionally fitted to the board to power the PCB. This has a maximum input voltage of 28V which limits the maximum fieldbus voltage used.

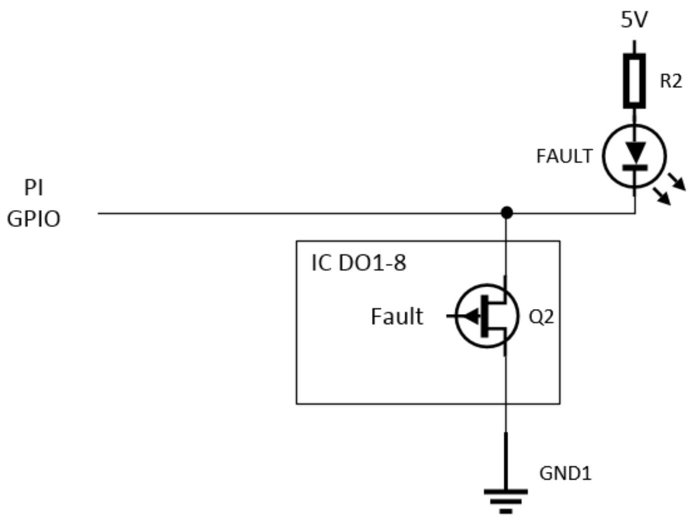
LEDs

There are three LEDs on the board.



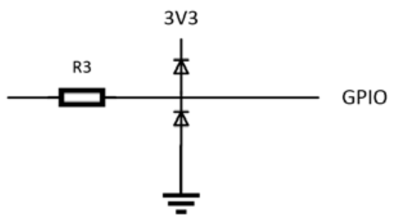
- **Fault** – Indicates a fault with one or more of the output high side driver ICs. This can be a short or over temp.
- **VF** – Indicated Field supply VF1 is powered.
- **Run** – software controlled to a GPIO Output, generally set to pulsing to indicate the program is running.

Note- The fault output of the driver ICs is connected to a Pi GPIO Pin, this allows fault status of the board back to the software.



Digital inputs

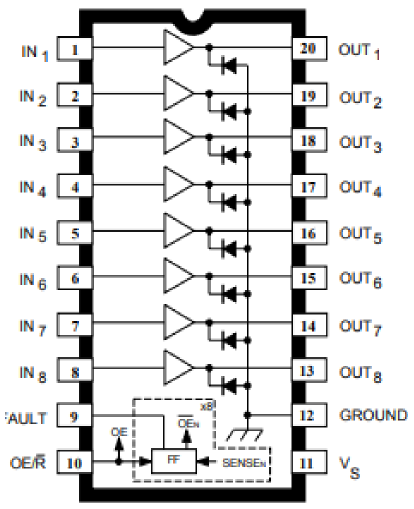
A simple clamp circuit allows digital inputs to be interfaced:



The inputs are designated as DI1-4 and are located at the bottom of the board. Input high is any voltage over 1V up to 35V.

Digital outputs

The digital outputs are controlled via a UDN2987 High side driver IC. This contains Darlington driver arrays and operator up to 35V.

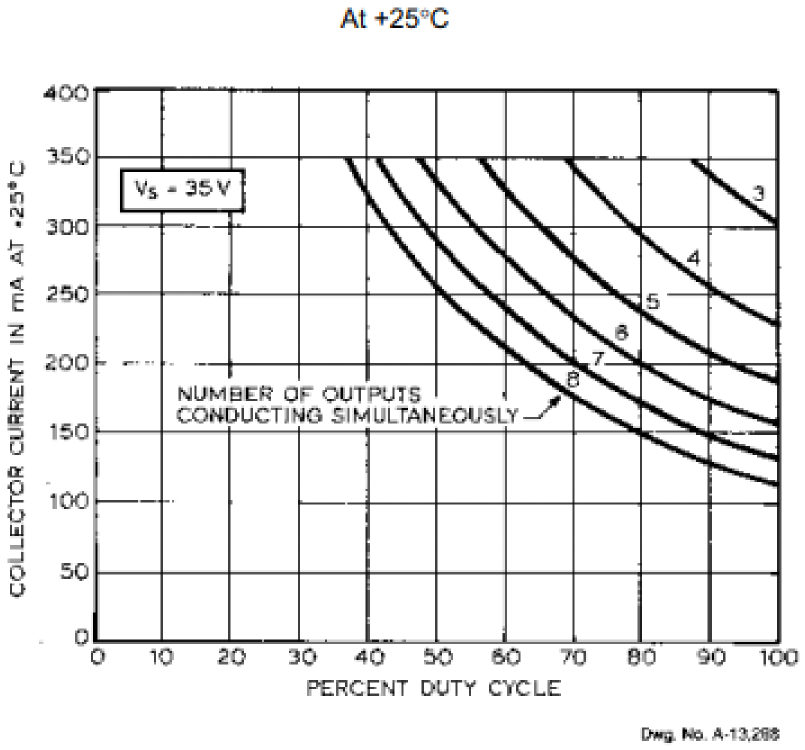


The drivers feature the following functionality:

- 35V V Max (VF1/2 – limited by DCDC if fitted on VF1 which has 28V max)
- Output enable input
- 8x 350mA Overcurrent protected High side Darlington drivers
- Internal Ground clamp diodes
- Thermal shutdown
- Output faults disable that channel and can be reset by toggling the OE Pin.

Application thermal considerations

The IC can pass 100mA on all 8 channels at a temperature of 25 Degrees Centigrade. The following data sheet describes how this can be uprated depending on channel duty cycle. Generally speaking though should be given to how much current you are asking the device to deliver and across how many pins.



Analog outputs

There are two analog outputs AO1 and AO2, which are generated by feeding a PWM pulse into an on board 1.5Hz filter, which then is fed into a LM358 op amp with a gain of 3. The output can be driven between roughly 0 and 10V and the default 100Hz PWM frequency is sufficient to do this. In order to drive 10V the field voltage must be at least 12V since the op amp can only get within 2V of the positive supply rail.

The LM358 can supply roughly 20mA, a 1K ohm resistor is placed in series to provide some protection however this can be shorted with the two links LK1 and LK2 if higher currents are required, it should be noted though driving more than 20mA risks damaging the op-amp.

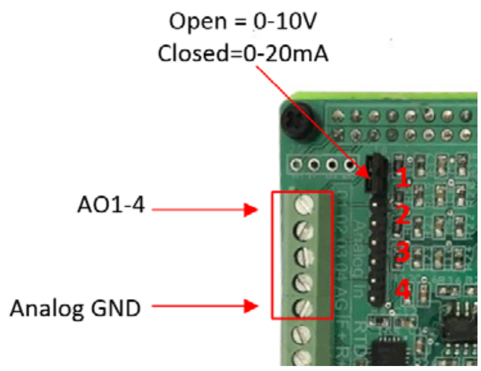
The following table describes the volt drop over the protective 1K resistor at varying loads, when shorted though the load resistor limits the short circuit current to 10mA which is safe for the op amp.

Vout	Load resistance (AOn)	Volt drop over protective resistor
10V	470K	0.02V
10V	100K	0.09V
10V	10K	0.9V

Analog inputs

The unit features 4x 0-10V analog inputs which can be reconfigures as 0-20mA inputs using jumpers. A Texas instruments ADS1015 12bit resolution programmable gain ADC is used for this purpose.

Placing a jumper on a channels header applies a 500R load to the input of that channel, this converts a 20mA current input to 10V enabling the unit to measure either current or voltage on each channel.



ti

Figure 1 Analog input configuration

By default the gain of the ADC is set to 2, a potential divider and filter on the PCB reduces the input voltage to the ADC my a factor of 5. The following table illustrates how various gains can be used to alter the input range of the ADC.

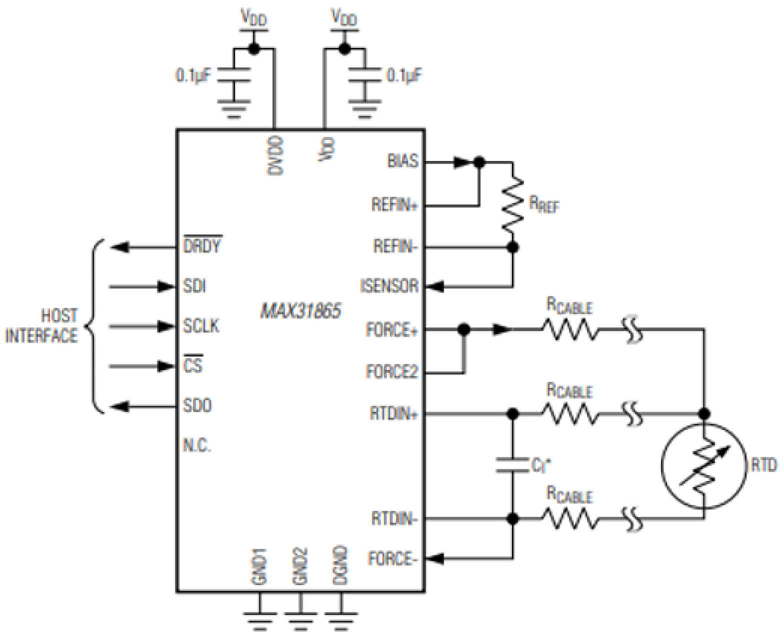
ADC Gain	**ADC range**	**Voltage range at terminal (Vmax)**	**ADC Register value at (Vmax)**
2/3	+/-6.144V	30.72	2048
1	+/-4.096V	20.48	2048
2	+/-2.048V	10.24	2048
4	+/-1.024V	5.12	2048
8	+/-0.512V	2.56	2048
16	+/-0.256V	1.28	2048

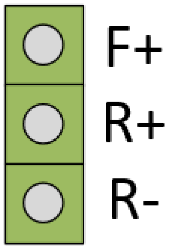
A fifth terminal is provided for the analog ground reference, this is filtered from the main 0V on-board this PCB.

The input resistance at the terminals is 400K + 6M (gain 2) in voltage mode and 500 Ohms in current mode.

Temperature input

A MAX31865ATP RTD sensor is provided to interface to a PT100 sensor. The unit is configured in 3 wire mode.

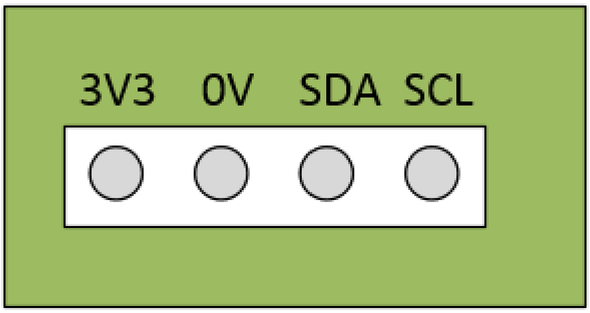




The above diagram shows how to connect a 3 wire PT100, all 3 wires are required for the transducer to operate correctly. 2 Wire sensors can be used by wiring in an additional wire for the Force terminal, this should be as close to the sensor as possible as not to affect accuracy. The sensor interfaces to the Pi over a SPI interface.

I2C Interface

A 4 way 2.54mm header is provided to interface to additional I2C devices. This already connects the on-board ADC but can be used to attach additional devices.



The Pi already has bus termination resistors for the I2C bus, any additional devices should have as short a connection as possible to minimise distortion on the I2C bus.

Software

The software library is provided at [https://github.com/lawsonkeith/PiIO]. This is a python3 library and is designed to work on linux based systems such as Raspbian.

You can clone this repository and manually install it’s dependencies as described in the following video:

github repo board page

A video on how to install the library and set up your Pi can be found here.

basic config

Various example projects are documented here:

node red control

node red and python

python only control

You will need to perform a number of tasks before your system is ready to use:

1. Update OS
2. Edit nano config file (if you’re using nano as an editor)
3. Enable SPI / I2C and SSH in raspi-config
4. Clone the github repository
5. Install required Linux packages
6. Install required python packages
7. Test node red by importing a json flow into it.

Software structure

The repository is structured as follows:

- **PiIO** – Fundamental drivers written in python 3
- **Docs** – Markdown documentation
- **Examples** – Contains python3 examples
- **Images** – Contains pictures used in the repository
- **Manuals** – Contains all PDF manuals including this one
- **Install_packages**.sh – installs required linux packages
- **Install_py_packages**.sh – installs required python packages
- **Setup**.py – used to install the PiIO library

Basic_functs example

This example does not require any hardware but just shows the operation of some of the PiIO utility API.

- Alarm function
- Exponential moving average function
- Scale function
- Rising edge function

- Falling edge function
- Timed pulse function
- Timed on function
- Timed off function

You can run the program [python3 ./basic_fincts.py] and the program will step through and test each of these utility functions.

Definitions of these functions can be found in PiIO/PiIO.py.

Basic_ADIO example

This program cycles through all the basic functionality of the board.

hydro_ADIO example

This program demonstrates an application used to control beer fermentation as well as an irrigation system. A PT100 controls a beet heater whilst the program checks the local weather to tee if my strawberries need watering, if so a solenoid is energised that waters them. A user interface is provided using node red.

Concurrent_DO example

This program show the use of python threads.

Additional examples for the DIO board can easily be modified for usage on this board…

Fault_DIO example

This program shows how the fault status of the output drivers can be read back in software.

Nodered_DIO example

This demonstrates how Node red can be used to provide a user interface via a web browser that can be used to scan inputs and write to outputs on the board. Again proportional control is used to drive the outputs.

Nodered_direct_DIO example

In this example the GPIO nodes of node red are used so that the node red user interface can control the PI GPIO but without having to run a python program on the PI. This provides an easier implementation but reduced functionality over how the IO can be controlled.