

In the next release of the image I'll be adding in analog support so I thought it was worth doing a quick write up on how this will be done. The traditional option for analog inputs, driving cabs with pots, spinners, trackballs etc has been to buy an APac board from Ultimarc. They are very similar to the IPac and have dedicated ports to take connections for up to four analog devices. They are presented via USB and appear as two separate game controllers with X and Y axis which can then be configured in mame as inputs. Dead simple but the boards are over £30 delivered, you need to use up a USB port and there isn't really room on the nice tidy DIY Pi2JPAC board so you need to mount the APac separately. Analog inputs on the Pi are not natively supported via the digital only GPIO pins so an add-on is needed to translate the signals to a digital value. A very cheap solution is a little add-on board called the Analog Zero which uses an analog to digital conversion chip called the MCP3008. It comes in a build it yourself kit from RaspIO and can be purchased for around £10 from many electronics and Pi suppliers. You solder the parts together and then mount the board onto the GPIO pins on the Pi. You then have a choice of how to cable your analog inputs into the board, either using header pins and JST XH/NH style connectors or the provided Dupont breadboard style connector pins. The board provides up to 8 channels of analog inputs, each potentiometer will take up one input so you could support up to four analog joysticks if you really needed to. Each analog input translates as a single joystick axis so a driving cab for example would need three: x axis for steering, y axis for gas and z axis for brake. To wire up your inputs you need to connect the outer pins of the potentiometer to 3.3v and ground, and the centre pin called the wiper to the analog input. Voltage is measured between the outer pins and compared to the voltage between wiper pin and ground so as the pot is turned, resistance goes up and the difference between the wiper and the reference voltage is translated to a numerical value. The board connects to the Pi via an interface called SPI (Serial Peripheral Interface) which is kind of an updated serial port allowing for much higher clock rates. Thankfully the communication is all handled seamlessly using the system provided libraries which makes writing the code to get the inputs to work much easier! I've put together some python code to communicate with the board and create input devices in the operating system that can then be configured in the various versions of mame in the image. Given the various types of input available I've come up with a little configuration menu that can be launched via the main service menu, a similar set up to the wifi config menu. At the moment it supports three setups, a driving cab setup with three axis, a single joystick setup with two axis and a two joystick setup with two axis each. You can switch between the operating modes via the menu, change the analog input pins you want to use and adjust a couple of settings for noise and deadzone. Noise is the sensitivity setting, if the inputs seem jerky or not smooth you can increase this setting to reduce that, you might be using pots of a different value for example. The other setting is to create a deadzone for a joystick, if the inputs are not set to zero when centered on an analog stick it can cause you control issues in game so this can be increased or reduced if need be. The defaults are shown in the config menu and have been calibrated against 5k linear pots. Configuration within the emulators is different for each one, in advance mame for example, analog inputs have to be hard coded within the config files themselves and only tweaked in the menu so they will be added based on my testing with different analog games. The other versions of mame can generally be configured via the menus, tab in mame and F1 in FBA, I can't guarantee a 100% success rate so you please let me know of any issues you get and I'll try to resolve them quickly.