Outrun Multi Project I'd been planning to add in analog support for the DIY image for s while, the APAc is plug and play and seemed ideal and it was really only configuration file changes needed to get that in the image. Chris P invited me to the meet and then made a passing joke about Flinnster and I having a Pi image ready for his Powerdrift cab the following Saturday. I like a challenge so said I might be able to get something running by then which left 5 days to get a build ready. The first problem was that in any of the emulators in the image, Power Drift didn't run perfectly so needed tweaking, ramped up the overclock on the pi and dialled down the sound quality in advance mame which got it running smoothly with no sound slow down or clipping issues. I ordered an APAc for the inputs and then had to tackle the shifter input. In a real Outrun or Powerdrift cab the gear shifter is a single switch, if the switch is pressed and held down you are in high gear and if the switch is open you are in low gear. I thought I might need to build an external circuit to handle the logic but after some testing I wrote some python code that creates a virtual joystick with a single button and then creates an interrupt on the state of a single GPIO pin. I tied the switch to the pin and ground so when the pin changes state either lo-hi or hi-lo the virtual button is pressed and toggles the gearshift. In later versions of mame this is just a config file entry so much simpler to get working! The wiring was relatively straightforward, I grabbed the pinout from a great post on UKVAC by Smarty when he made up an Outrun to Powerdrift loom and used a hacked jamma loom and a jpac to test. We hooked everything up and had an issue with the video not syncing correctly, then Flinnster spotted that my ground and sync wires were reversed! At some point Sega switched using blue for sync and white for ground! Once that was working we got the gear switch working and then hooked up the APac to test the steering which behaved like a digital input, at about a half turn the analog inputs quickly went from centered to full left or right. That was a config file problem, advance mame needs the joystick number and axis assigned to the paddle controller in the config file so once that was done it worked fine. I'd taken a PC as a backup and we switched to using that when we had the steering issues and Windows mame had no problems configuring everything via the GUI. After some expert tweaking by Flinnster we got it running great and Eric Banana gave it the thumbs up. Since then I've been working on how to get this to a standard solution that you can use in any driving cab to get mame working and while using a JPac and jamma as a standard interface would work well, you then need to buy a board to give you analog inputs and then something that can do outputs if you want to use lights and force feedback and the price goes up. I did a bit of research into analog to digital conversion on the pi and came across a couple of solutions that might fit the bill - some firmware for Arduinos that gives you analog and digital controllers just like an APAc and a LEDWiz compatible device for outputs, and an addon board for the pi called the Analog Zero which is a little hat with a chip called an MCP3008 analog to digital convertor that gives you 8 analog inputs. It gives you 10-bit resolution so values from 0-1024 which is more than enough for pot based controllers, you can add a compatible 12-bit chip into the same socket to give you 0-4096 if you really need it! The analog zero board is perfect for adding to the DIY solution as you can just plug it straight onto the GPIO so no physical mods needed to the board and I've created some python code to read the inputs that will be included in the next image and you'll be able to enable it via the service menu. It's the same form factor as a Pi Zero, costs £10 and you need to solder it together yourself. To physically attach the analog inputs you wire up 3.3v and ground from the Analog Zero board to the outer pins on the pot and the middle wiper pin goes to one of the analog inputs A0-A7. I've assigned each one to a joystick axis by default, two 3-axis sticks with either buttons via the remaining GPIO pins on the hat or via the JPac as normal. The Arduino solution is a bit more involved, the firmware goes back to 2012 and has been used as a basis for a controller used in virtual pinball called a Pinscape controller. It was developed to give virtual plunger, nudge, buttons and lighting outputs in a cheap

microcontroller. When used with an Arduino you lose the gyro function but it's not critical for a mame driving cab. I'd not worked with Arduinos before so took a bit of getting up to speed with the terminology and working out how to compile the firmware so it can been uploaded to the board. I'd also not appreciated that Arduinos have onboard analog to digital conversion which the Pi lacks. I picked up a cheapo knock off Leonardo for £7 from Amazon, you can get them cheaper if you don't mind waiting for delivery from China but I was keen to get cracking! I spent an afternoon going through the code on github and figured out there are two files that need to be modified to change the configuration of the board. One controls the hardware functions that are presented via the USB interface and the other one controls the pinouts of the board. I built a test config, compiled it and uploaded to the Arduino and initial testing went badly! I had a single pot attached and the input was being read on all pins - I rotated the pot to the left and all of the x and y inputs went up! After a lot of head scratching I worked out that all configured inputs need to be attached for the voltages to be read correctly. Once I got the hang of the config I created a bunch of different versions which all seemed to work great and I settled on a 3 axis controller setup, x for steering, y for gas and z for brake. I plan to release these as compiled firmware files ready to go so people don't have to installl the development tools and feck about compiling code. Flashing the Arduino is dead easy, you press the reset button on the Arduino twice to make it ready to flash then it's a single command line to push the file. The last challenge to overcome is the issue of outputs from mame on the Pi - advance mame only supports a simple scripting system that can do different things based on events happening in the game. Frustratingly the official mame output system was released just after the .106 version advance mame is based on and it just doesn't support what I'm trying to do out of the box. Outputs in mame are trigger based events that can be captured and actions taken such as lighting up a start button lamp when a credit is entered, vibrating a gun when fired or providing force feedback via a motor in a driving cab. The Outrun and Powerdrift cabs both have a feedback motor attached to the steering wheel which shakes when you bump a car or crash off the track. It's activted via a 5v signal sent to a relay in the cab which then turns the motor on and off. Pretty each to hook into either via GPIO from the Pi or the Arduino controller using a LEDWiz style output and a little 5v relay board. I wouldn't want to hook the pins directly to the cab relay switch just in case it blows the board, the pins on the Pi and the Arduino are very sensitive and relay board costs about 50p. I've come to a bit of a dead end with advance mame now, there is no standard way to capture output in the same way as mainstream mame which is a real shame as it was almost there, if you don't mind not having force feedback the solution works perfectly! I have started testing with other versions of mame in the hope that I can get outputs, the games just don't run well enough in the newest versions so I've been working backwards compiling them from source on the latest Raspbian Stretch image and SDL with hardware acceleration. Mame 206, 174, 172 were a no go and then I tried lr-mame2010 which is a libretro core based on mame 139. Performance is really good, the games appear to run at full speed but I haven't worked out whether outputs are included or not, being a libretro core you don't get the same access to the configuration so I'll check out the source code on github and try to suss it. On the upside the PC version is pretty much ready to go, you can drop in a mame PC based on my image, use the Arduino controller and get the full Outrun or Powerdrift experience with a custom loom. I've added a breakout board with screw terminals to the Arduino just to make connections easier to hook up but you can go direct to the board with header pins if you prefer. The PC solution can obviously give you better performance than the Pi depending on the spec you run with so you can run many more games but the using the Pi will mean you can build a board that will be truly plug and play, powered from the cab directly which is still the ultimate aim.