GoldCorp Datalogger

# Hardware

Below is a list of all the hardware that is connected/needs to be connected, including configuration instructions.

## Raspberry Pi

Works, and is configured.

## CAN Board – SK Pang

Works fine plugged into the Raspberry Pi. No modifications were needed.

In the GitHub repo, its schematic can be found at the following location, CAN-Bus-Datalogger\Raspberry Pi CAN\pican\_rev\_B.pdf

## RTC DS1307 – SparkFun (BOB-12708)

Modified SparkFun RTC. I cleared the jumper that connects SCL and SDA via resistors to 5V, as per [this guide](https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi/wiring-the-rtc) to hooking up an RTC to a Raspberry Pi.

Schematic can be found in the GitHub repo under CAN-Bus-Datalogger\Raspberry Pi CAN\RTC-Module.sch

On boot up, the Raspberry Pi syncs it’s time with the RTCs time.

If you’d like to test the time, *sudo hwclock –r* reads the time from the RTC; *sudo* *hwclock –w* writes it.

## Soft Shutdown Hardware

The Raspberry Pi likes to be asked to shutdown nicely, instead of abruptly losing power as this may damage it. For this, we are using a 12V battery pack, and an ignition switch.

### Ignition Switch

[2A Car Ignition Switch](http://mausberry-circuits.myshopify.com/collections/car-power-supply-switches/products/2a-car-supply-switch) is used. Hook up instructions can be found [here](http://mausberrycircuits.com/pages/car-setup), but all of the packages are already installed & configured. Should be plug & play.

The vehicle will provide the battery with 12V, and the battery will supply the switch with constant 12V power. The 12V from the vehicle will be wired to the Ignition of the switch, such that if power from the vehicle cuts out, the switch will know to safely shutdown using power from the battery.

### Battery Pack

Lee ordered a battery pack. Use it.

# Software

A total of 5 cron jobs, and 2 python scripts control the entire Datalogger. They are outlined below.

## Directories

All files pertaining to the Datalogger are found in /data/

* /data/scripts – all scripts that are to be run
* /data/RAW – where RAW canbus files are stored before they are parsed
* /data/dailylogs – where parsed logs sit
* /data/summary – where summary.csv sits

Only the /data/dailylogs and /data/summary directories will be accessible via samba share by the user, and the directories are set to read-only. At the moment, the entire /data/ directory is also shared with full read/write access, but this is to be revoked before delivery.

## Cron Jobs

### start\_can

Sends the commands to get the canbus line running:

*ip link set can0 type can bitrate 500000 listen-only on  
ip link set can0 up*

### ensurecanstarted

This script will look at the list of processes running, and if *mcp251x\_wq* isn’t running, it will run the /data/scripts/start\_can script to get it running again.

I have no idea what can shut down the canbus, but this is here just in case.

### startlogging

Kills any running candump, and creates a new one (with a bunch of filters so it only records relevant messages).

*cd /data/RAW  
sudo killall candump  
(candump -l can0,477:7ff,478:7ff,475:7ff,270:7ff,294:7ff,306:7ff)&*

### parse\_logs

Waits 5 seconds to ensure that the previous log has completed. Runs parse\_file\_Pi.py which will parse all files (candump\*.log) that haven’t been modified in the last minute. After parsing them, it will delete them (regardless of whether parsing was successful).

### createsummary

Runs update\_summary.py which will take the summarized line of data at the top of a daily log and move it into the summary.csv document.

*(python /data/scripts/update\_summary.py `find /data/dailylogs -type f -name "\*.csv" -mmin +4`;)&*

### deletefiles

Checks to see if the SD card is getting full; if so, deletes files in the /data/dailylogs directory. The script will keep deleting the oldest file until the SD card’s capacity is at 95% or less.

## Cron Schedule

The cron schedule is currently set to the lowest acceptable time interval. Logs happen every minute.

\* \* \* \* \* bash /data/scripts/ensure\_can\_logging #Every minute

\* \* \* \* \* bash /data/scripts/startlogging #Every minute  
\* \* \* \* \* bash /data/scripts/parse\_logs #Every minute (but has a delay of 5 seconds)  
0 5 \* \* \* bash /data/scripts/update\_summary #00:05 every day  
0 6 \* \* \* bash /data/scripts/deletefiles #00:06 every day

It would be good to change the logging and parsing to be every x minutes (a longer time period), as there is a bit of data lost or repeated between logs.