Cratemonitor.py Documentation

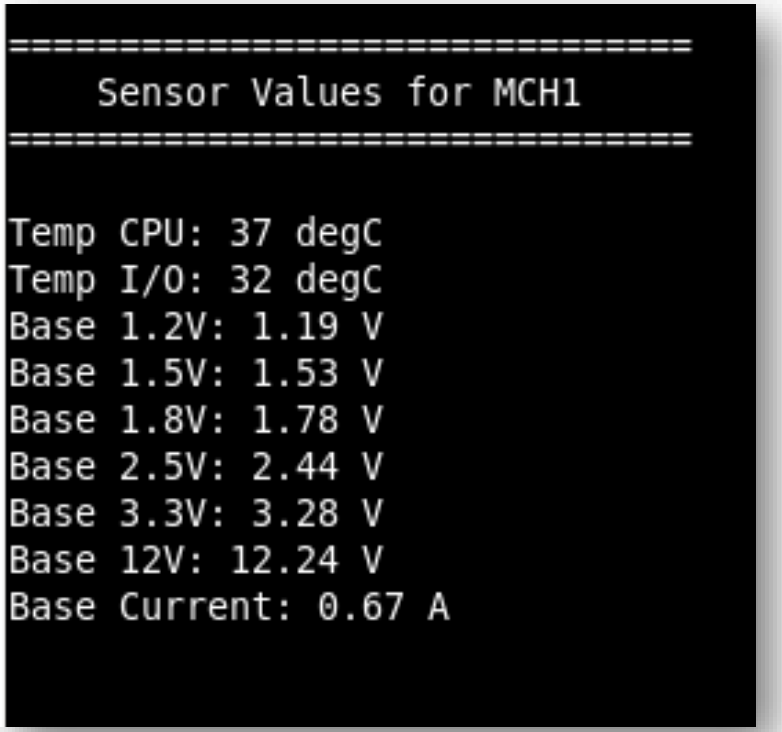
- Birk Engegaard, birk.engegaard@cern.ch

Quick facts

- Python2 script, uses ipmitool
- Crate general, I.e. it can be called or hardcoded with mch hostname.
- The monitorable objects are python classes with sensor values as attributes.
- Upon instantiation the class will check type, I.e. is it an ICL-CERN mmc or an FC7 R2 mmc, and query the card for its sensor values and store them in the variables. If the card/board/mch/CU/PM is of an unkown flavor it will store 'None' values for the sensors.
- The classes are easily modified to include different flavors of FC7s, CUs etc, and brand new classes can also be added. The developer does not need to know everything in the script to modify it to work with the crate or card of choice.
- The classes have getData(), setHostname() and printSensorValues() functions.
- For now empty variables are of python 'None' type

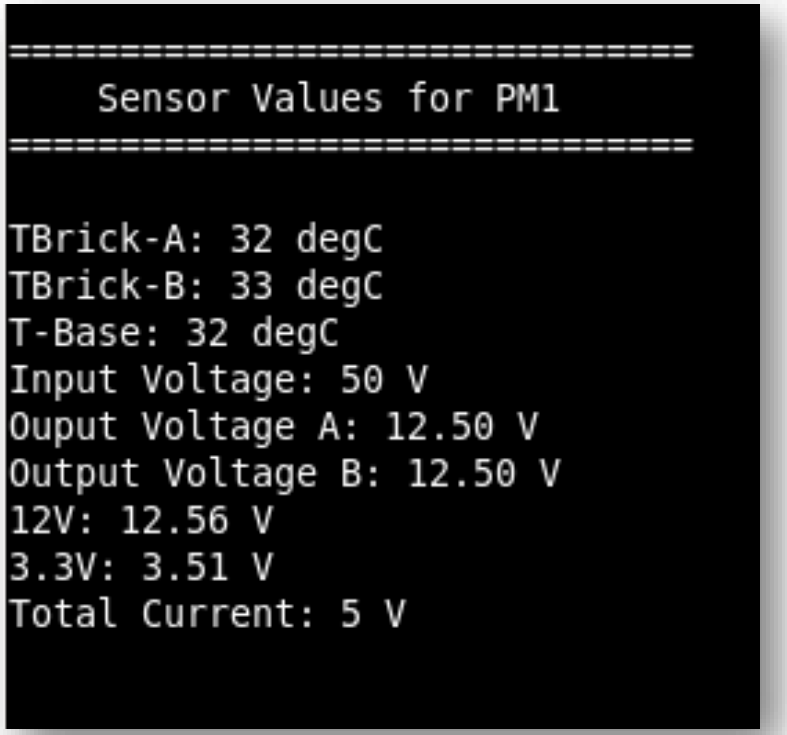
The MCH object

- Instantiation: mchObject = MCH(), print values: mchObject.printSensorValues() or print mchObject.current
- Supported flavors: NAT-MCH-MCMC
- Class functions: setHostname(), getData(), printSensorValues()
- getData() gives the sensor variables a value and returns a list of all the values
- __init__(self. MCHIndex = 1)
  - self.MCHIndex = MCHIndex
  - self.entity = "194.{0}".format(str(96 + self.MCHIndex)) # converting MCH index to ipmi entity
  - self.hostname = HOSTNAME # Global variable
  - self.flavor = None # The card type
  - self.tempCPU = None # Temperature of the CPU
  - self.tempIO = None # Another temperature value
  - self.volt1V5 = None # 1.5V measurement
  - self.volt1V8 = None # 1.8V measurement
  - self.volt2V5 = None # 2.5V measurement
  - self.volt3V3 = None # 3.3V measurement
  - self.volt12V = None # 12V measurement
  - self.current = None # Base current consumption
  - self.sensorValueList = self.getData() # Fill sensor data upon instantiation



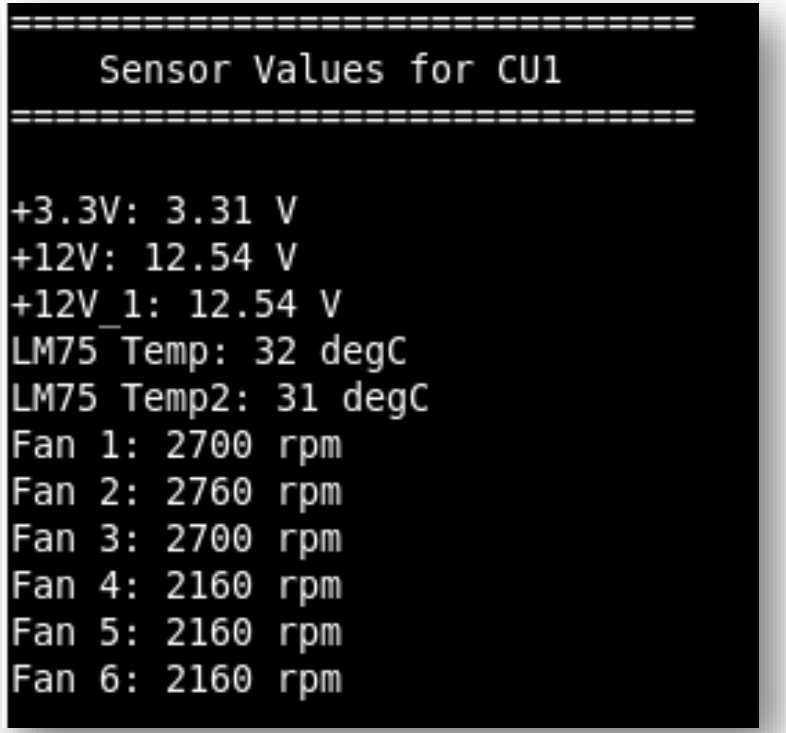
The PM object

- Takes in the power module index. Turns out PM1A and PM4 in the crate corresponds to PM index 1 and 2 in IPMItool
- Instantation: PM1 = PM(1), PM2 = PM(2). Print values: PM1.printSensorValues() or print PM2.currentSum
- Supported flavors: NAT-PM-DC840
- Class functions: setHostname(), getData(), printSensorValues()
- getData() gives the sensor variables a value and returns a list of all the values
- __init__(self, PMIndex = 1)
  - self.PMIndex = PMIndex # PM index in crate
  - self.entity = "10.{0}".format(str(96 + self.PMIndex)) # converting PM index to ipmi entity
  - self.hostname = HOSTNAME # Global Variable
  - self.tempA = None # Temperature value [degC]
  - self.tempB = None # Temperature value [degC]
  - self.tempBase = None # Base Temperature [degC]
  - self.VIN = None # Input voltage [V]
  - self.VOutA = None # Output voltage A [V]
  - self.VOutB = None # Output voltage B [V]
  - self.volt12V = None 12V measurement [V]
  - self.volt3V3 = None 3.3V measurement [V}
  - self.currentSum = None # Sum of output current [A]
  - self.flavor = None # Power module type
  - self.sensorValueList = self.getData() # Fill sensor data upon instantation



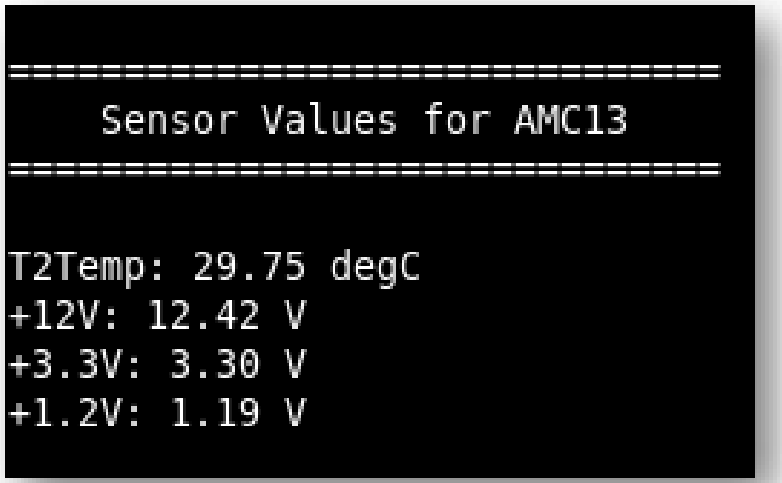
The CU object

- Takes in the cooling unit index.
- Instantation: CU1 = CU(1), CU2 = CU(2). Print values: CU1.printSensorValues() or print CU2.fan1
- Supported flavors: Schroff uTCA CU
- Class functions: setHostname(), getData(), printSensorValues() and checkFlavor()
- getData() gives the sensor variables a value and returns a list of all the values
- __init__(self, CUindex)
  - self.hostname = HOSTNAME # Global variable
  - self.CUIndex = CUIndex # CU index
  - self.entity = "30.{0}".format(96 + CUIndex) # Converting index to entity
  - if self.CUIndex == 1: # Converting index to CU target number
    self.target = "0xa8"
    else:
    self.target = "0xaa"
  - self.flavor = None # CU type
  - self.CU3V3 = None # 3.3V measurement
  - self.CU12V = None # 12V measurement
  - self.CU12V_1 = None # Another 12V measurement
  - self.LM75Temp = None # LM75 Temperature
  - self.LM75Temp2 = None # LM75 temperature 2
  - self.fan1 = None # Fan speeds [rpm]
  - self.fan2 = None
  - self.fan3 = None
  - self.fan4 = None
  - self.fan5 = None
  - self.fan6 = None
  - self.sensorValueList = self.getData() # Fill sensor data upon instantiation



The AMC13 object

- Instantation: amc13object = AMC13(1). Print values: amc13object.printSensorValues() or print amc13object.T2Temp
- Supported flavors: BU AMC13
- Class functions: setHostname(), getData() and printSensorValues()
- getData() gives the sensor variables a value and returns a list of all the values
- __init__(self)
  - self.hostname = HOSTNAME # Global variable
  - self.flavor = None # amc13 type
  - self.T2Temp = None # T2 Temperature
  - self.volt12V = None # 12V measurement
  - self.volt3V3 = None # 3.3V measurement
  - self.volt1V2 = None # 1.2 measurement
  - self.sensorValueList = self.getData() # Fill sensor data upon instantiation



The FC7 object

- Instantation: amc1 = FC7(1). Print values: amc1.printSensorValues() or print amc1.humidity
- Supported flavors: No flavors supported yet. ICL-CERN FC7 not supported. R2 FC7 to be supported
- Class functions: setHostname(), getData(), checkFlavor(), printSensorValues()
- getData() gives the sensor variables a value and returns a list of all the values
- __init__(self, FC7Index)
    - self.hostname = HOSTNAME # Global variable
    - self.FC7Index = FC7Index # Location in crate
    - self.entity = "193.{0}".format(96 + FC7Index) # converting index to entity
    - self.flavor = None # MMC type
    - self.humidity = None # humidity measurement
    - self.temperature = None # temperature measurement
    - self.volt3V3 = None # 3.3V
    - self.current3V3 = None # 3.3V current
    - self.volt5V = None # 5V
    - self.current5V = None # 5V current
    - self.l12_VADJ = None # ADJ voltage?
    - self.l12_IADJ = None # ADJ current?
    - self.volt2V5 = None # 2.5V
    - self.current2V5 = None # 2.5V current
    - self.volt1V8 = None # 1.8V
    - self.current1V8 = None # 1.8V current
    - self.voltMP3V3 = None # MP 3.3V
    - self.currentMP3V3 = None # MP 3.3V current
    - self.volt12V = None # 12V
    - self.current12V = None # 12V current
    - self.volt1V5 = None # 1.5V
    - self.current1V5 = None # 1.5V current
    - self.volt1V = None # 1V
    - self.current1V = None # 1V current
    - self.volt1V8_GTX = None # GTC 1.8V
    - self.current1V8_GTX = None # GTX 1.8V current
    - self.volt1V_GTX = None # GTX 1V
    - self.current1V_GTX = None # GTX 1V current
    - self.volt1V2_GTX = None # GTX 1.2V
    - self.current1V2_GTX = None # GTX 1.2V current
    - self.l8_VADJ = None # ADJ Voltage
    - self.l8_IADJ = None # ADJ current
    - self.sensorValueList = self.getData() # Fill sensor data upon instantiation

Improvements to be made

- Now queries the MCH for sensor data for every instantiation. Would be more time efficient to call the MCH for all it’s data to begin with, and then have the different classes get data from this list.
  - For now this can't be done since all the sensors doesn't work with the full sensor list. Some of them has to be called individually with entity or target index.

- Compatibility with the FC7.