McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

HET Telemetry Database: Details, Tools and Examples

September 26, 2018

McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

Topics

- Events, messages, and their payloads
- Capturing events
- Database structure
- Tools
- Examples
- Future

Events, messages and payloads

McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

- TCS subsystems broadcast events
- Events carry time stamped metrics from the various subsystem components, sensors and actuators
- Events may be emitted on a cycle, upon state change or other subsystem stimuli, such as receipt of an event


```
{ "time": "2018-09-20T09:02:58.824",
"tcs.DMI.measurement": {
        "setpoint":     12820.075000,
        "valid":        "true",
        "validity_error":       "None",
        "delta.w":      1.818989e-12,
        "w":    12820.075000,
        "correction.w": 1.818989e-12,
        "ignored_for_offset":   "false",
        "active":       "false",
        "__system":     "tcs",
        "__source":     "DMI",
        "__key":        "measurement",
        "__data_time":  "1537434178.824349853",
        "__wire_time":  "1537434178.824930080",
        "__data":       "false",
        "__pid":        6373

} }
```


Thursday, September 20, 2018 4:02:58 AM GMT-05:00 DST

Events, messages and payloads

McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

- Messages are sent and received between subsystems and clients
- The syscmd shell script wraps python that wraps sending a message from the user to the subsystem via C++ API
- For example, the cal script sends messages to the PFIP subsystem and CAMRA subsystem to actuate lamps, shutters, and the readout hardware
- Messages may be exposed as events using the wire tapping feature for debugging purposes

```
#!/bin/bash
syscmd -l 'expose( seconds=300.0, … )'
```

```
{
  "__handler": "expose",
  "__data_time": "1537435780.812028469"
  "seconds": 300.0,
  …
}
```

Events, messages and payloads

McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

- The payload is the portion of the event or message within the innermost {} to the right
- Payloads are time stamped with the time the corresponding object was created, and placed on the wire
- Tapping events also contain the time at which the tapped event was received
- Metadata are marked with a __ prefix. Meta-metadata in tap events are doubly prefixed
- Beyond time stamps, event and message metadata diverge


```
{ "time": "2018-09-20T08:50:56.851",
"tracker.tap.recv": {
        "id":   279,
        "t":    "1537433466.836263681",
        "x":    -1696.661083,
        "y":    35.761726,
        "z":    108.779186,
        "rho":  -0.542967,
        "theta":        -0.137300,
        "phi":  -7.263098,
        "____effective_id":     127,
        "____original_id":      127,
        "____origin":   "tracker.localhost.localdomain.915
        "____handler":  "setTraj",
        "____async":    "false",
        "____done":     "false",
        "____ack":      "false",
        "____error":    "false",
        "__data_time":  "1537433456.851523910",
        "__wire_time":  "1537433456.851525537",
        "____data":     "false",
        "____pid":      915,
        "__data":       "false",
        "tap_time":     "1537433456.851839386",
        "__system":     "tracker",
        "__source":     "tap",
        "__key":        "recv"
} }
```

Capturing events

McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

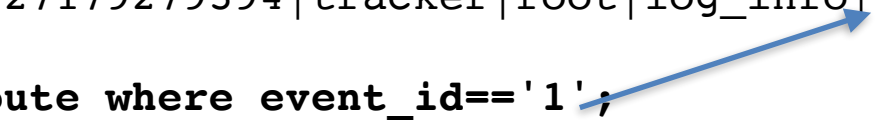
- Events can be captured and written to either standard output or database using the event monitor command line utility
- Simple C++ APIs allow for event capture either “as received" or via a cache and overwrite mechanism
- Python APIs provide an “as received” interface, OCD implements objects to support a cache and average interface
- Javascript bindings, bit rotting, are available to cache events in a web socket server
- Automated monitoring dumps database daily

Database structure

McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

The database is a simple SQLite format with a schema-less-like approach to mapping attributes to events.

```
sqlite> select * from event acs limit 1;
ts|origts|system|source|key|event_id
1527612027241595162|1527612027179279394|tracker|root|log_info|1

sqlite> select * from attribute where event_id=='1';
event_id|keyname|value|data_type
1|__data|false|boolean
1|__data_time|1527612027.179279394|string
1|__key|log_info|string
1|__pid|10785|number
1|__source|root|string
1|__system|tracker|string
1|__wire_time|1527612027.179338110|string
1|file|tracker_driver.cpp|string
1|function|TrackerDriver|string
1|line|34|number
1|message|Accepting commands...|string
```



Tools

McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

- sqlite3, other direct APIs to sqlite queries
- Event query shell script provides extraction of single event to table

```
$ event_query.sh test.db tracker.root.log_info | awk tr '\t' ,


,message,line,function,file,__wire_time,__system,__source,__pid,__
key,__data_time,__data
,Accepting commands…,34,TrackerDriver,tracker_driver.cpp,
1527612027.179338110,tracker,root,10785,log_info,
1527612027.179279394,false
,Client connecting to TMCS.,128,tracker_client,tracker_driver.cpp,
1527612027.197808785,tracker,root,10785,log_info,
1527612027.196852566,false
,TMCS server connected.,78,tracker_server,tracker_driver.cpp,
1527612027.207170471,tracker,root,10785,log_info,
1527612027.207143654,false
,TMCS client connected.,148,tracker_client,tracker_driver.cpp,
1527612028.207350792,tracker,root,10785,log_info,
1527612028.207297490,false
```

Tools

McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

tcsdb

- a structured python abstraction of the dynamic database contents
- clean syntax
- supports constraints on start and stop times
- provides configurable interpolation

```
> from tcsdb import *
> d = tcsdb('test.db')
> d.tracker.root.log_info.message()

([1527612027.1792793,
  1527612027.1968527,
  1527612027.2071435,
  1527612028.2072976],
 ['Accepting commands...',
  'Client connecting to TMCS.',
  'TMCS server connected.',
  'TMCS client connected.'])
```

Extracting the “message” field from the event “tracker.root.log_info”.

Examples

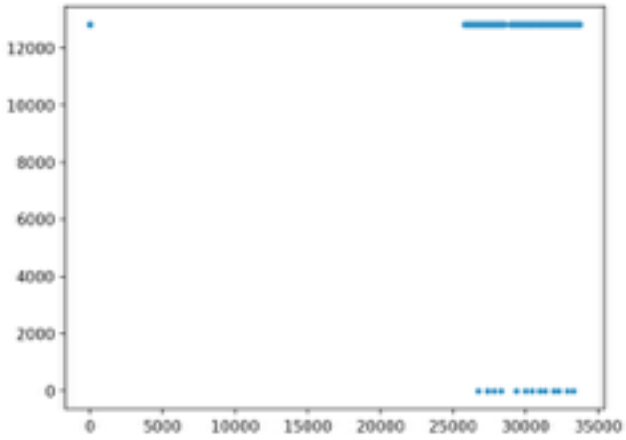
McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

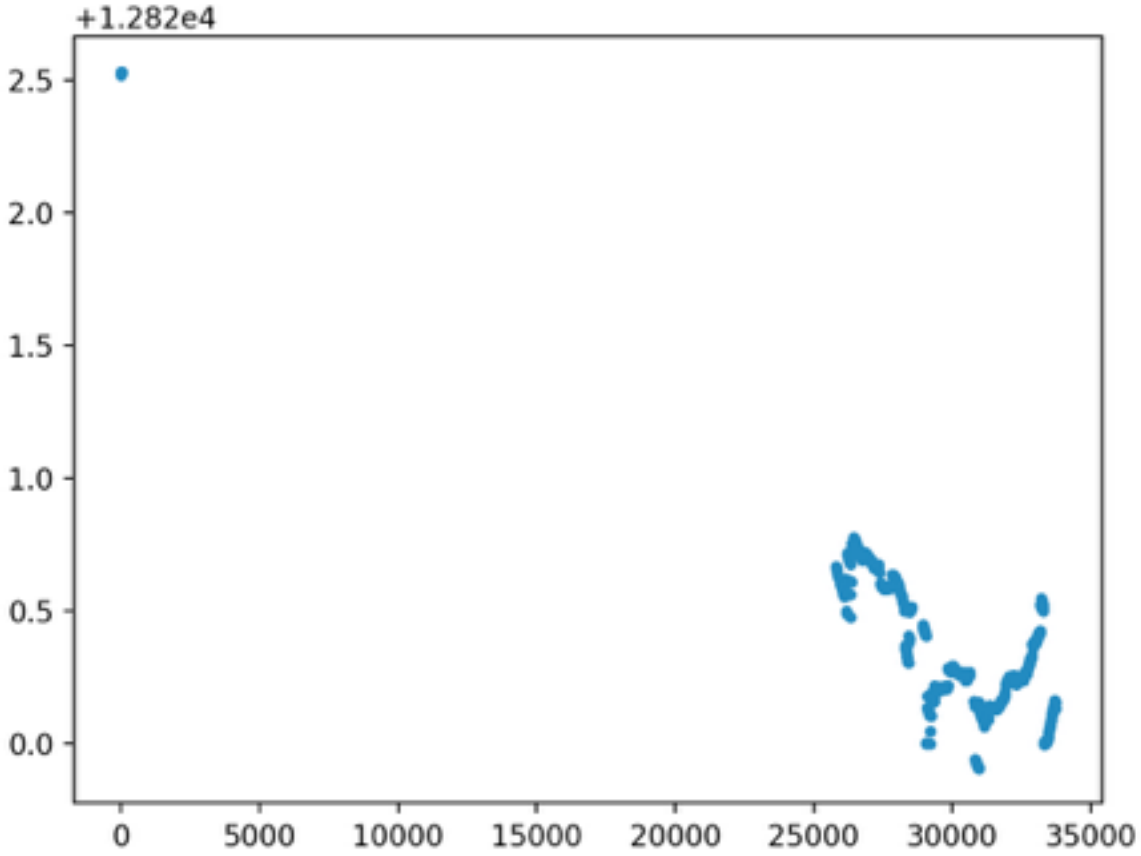
```
> d.tracker.tap.recv()[1][0]

{'____ack': 'false',
 '____async': 'false',
 '____data': 'false',
 '____done': 'false',
 '____effective_id': 1,
 '____error': 'false',
 '____handler': 'initTMCS',
 '____origin': 'tracker.localhost.localdomain.10787.06b8b4567-
>tcp://127.0.0.1:30400',
 '____original_id': 1,
 '____pid': 10787,
 '__data': 'false',
 '__data_time': 1527612069.8925824,
 '__key': 'recv',
 '__source': 'tap',
 '__system': 'tracker',
 '__wire_time': 1527612069.8925838,
 'tap_time': 1527612069.8951235}
```

Examples

McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

```
> import matplotlib.pyplot as plt
> [t,dmi] = d.tcs.DMI.measurement.w()
> import numpy as np
> t_ = np.array(t)-t[0]
> plt.plot( t_, dmi, '.' )
> plt.show()
```






```
> plt.plot( [T for i,T in enumerate(t_) if dmi[i] > 1], [D for i,D in
enumerate(dmi) if dmi[i] > 1], '.' )
> plt.show()
```

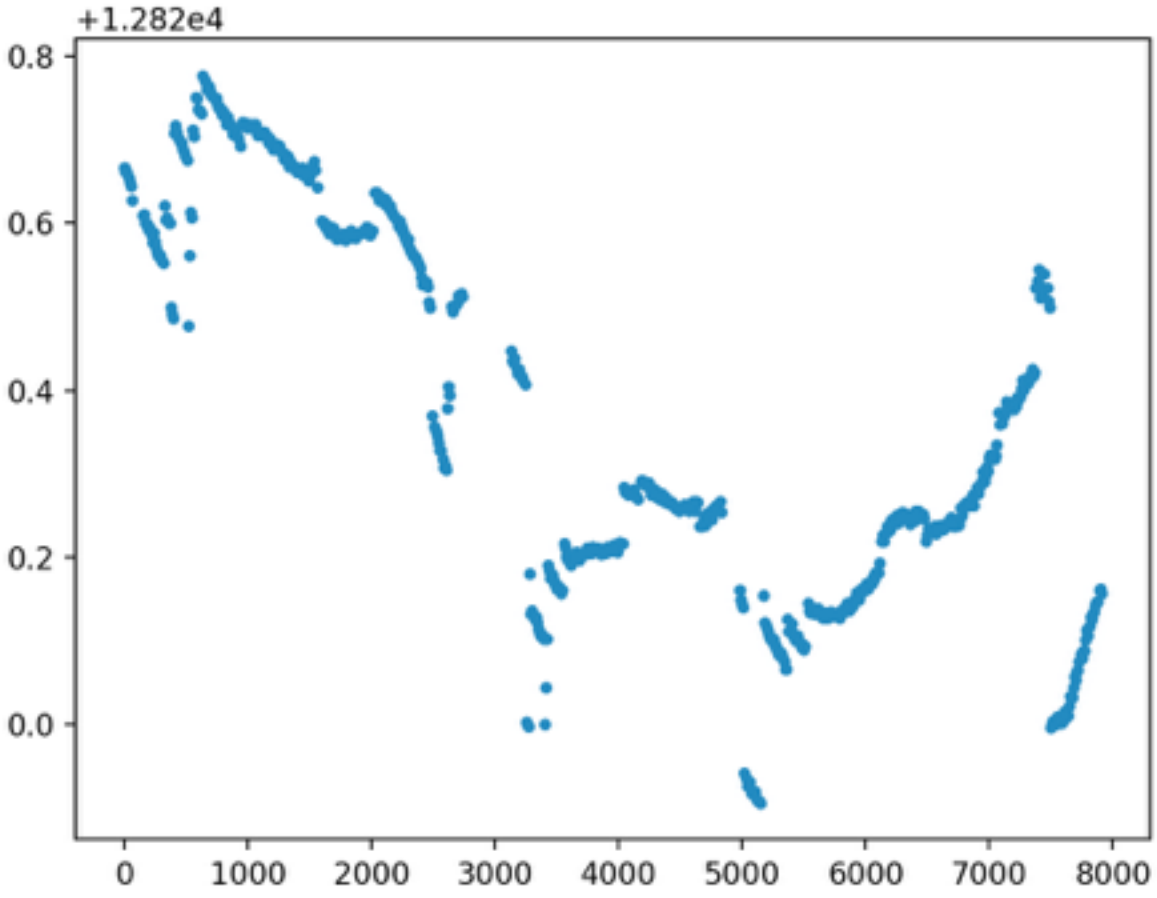
Examples

McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

```
> start_index = np.where(t_ > 25000)[0][0]
> t = t[start_index:-1]
> dmi = dmi[start_index:-1]
> t_ = np.array(t)-t[0]
> plt.plot( [T for i,T in enumerate(t_) if dmi[i] > 1], [D for i,D in
enumerate(dmi) if dmi[i] > 1], '.' )
```

Renormalize t_

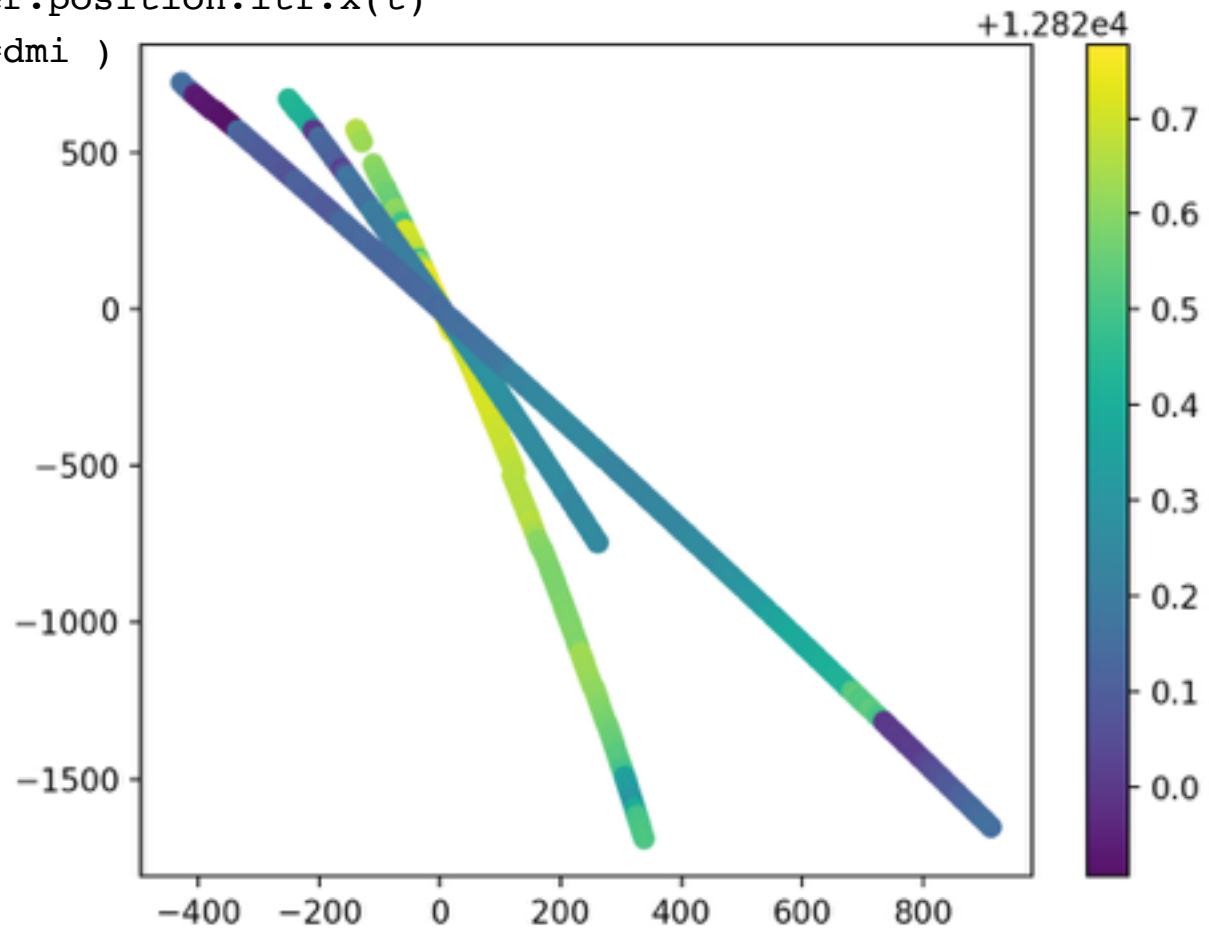
Reusing prior plot with t_ renormalized



Examples

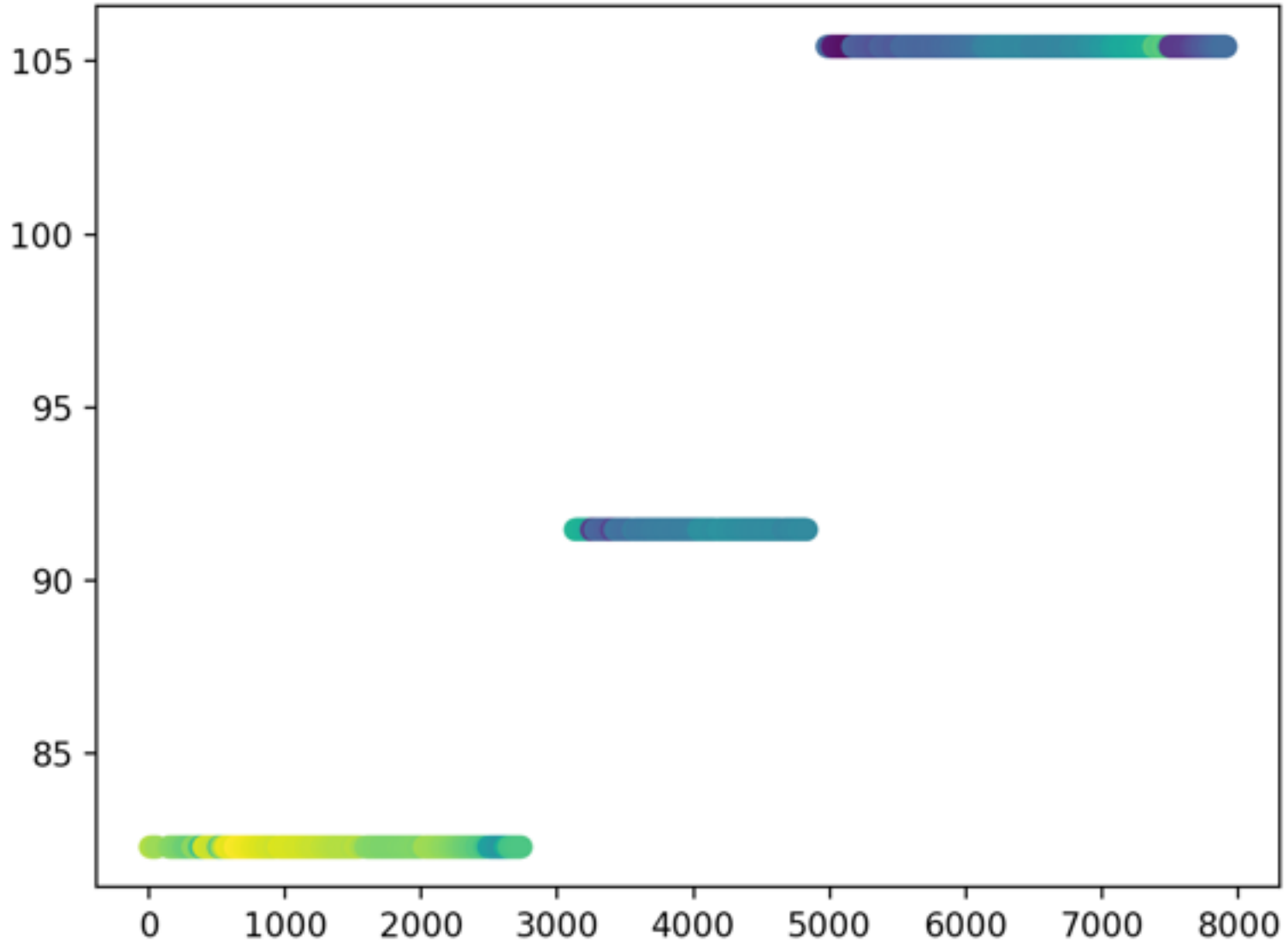
McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

```
> t = [T for i,T in enumerate(t) if dmi[i] < 12821.0 and dmi[i] > 12810.0]
> dmi = [D for i,D in enumerate(dmi) if dmi[i] < 12821.0 and dmi[i] >
12810.0]
> [_, x] = d.tcs.tracker.position.itf.x(t)
> [_, y] = d.tcs.tracker.position.itf.x(t)
> plt.scatter( x, y, c=dmi )
> plt.colorbar()
```



Examples

```
> [_,az] = d.tcs.root.ra_dec.az()
> plt.scatter( t, az, c=dmi )
```



Examples

McDonald Observatory
THE UNIVERSITY OF TEXAS AT AUSTIN

```
> [_,temp] = d.legacy.weather.status.MIRROR_TEMP_AVG(t)
> plt.scatter( t, temp, c=dmi )
```