ChimeraTK.

A tool kit for control application development.

Martin Hierholzer, Martin Killenberg

4th December 2018

7th MicroTCA Workshop for Industry and Research
DESY, Hamburg

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

What is a control application?

Software which

- montors and changes process parameters / process variables
- is integrated into the control system environment via middleware (EPICS, DOOCS,OPC-UA)

Process variables live

- on a hardware device (usually in registers in some address space)
- in the application
- in other control application servers

Register = Process Variable

What do you want to do?

Concept

- First: What do you want to do?
- Second: How do you do it?

What do you want to do?

Concept

- First: What do you want to do?
- Second: How do you do it?

Find the right level of abstraction

What do you want to do?

- Stream 16 bit data through a 44 kHz analogue-digital-converter

What do you want to do?

Concept

- First: What do you want to do?
- Second: How do you do it?

Find the right level of abstraction

What do you want to do?

- ~~Stream 16 bit data through a 44 kHz analogue-digital-converter~~

What do you want to do?

- Listen to music

How do you do it?

- Play a CD
- The rest is implementation detail

What are we going to do today?

Our example today: Operate an oven

Step 1

- Talk to the hardware
- ChimeraTK DeviceAccess

Step 2

- Write the control algorithm
- ChimeraTK ApplicationCore

Step 3

- Integrate into the control system
- ChimeraTK ControlSystemAdapter

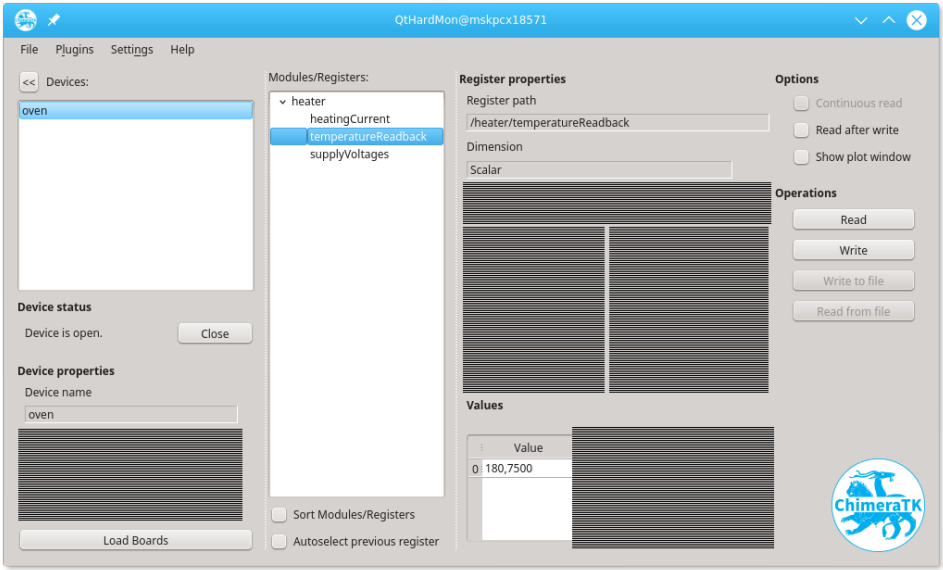
DeviceAccess.

What is a Register in DeviceAccess?

A register

- contains data (numerical or a string)
- is identified by a name
- lives on a device
- has a length ($1 \mathrel{\hat{=}}$ scalar, $> 1 \mathrel{\hat{=}}$ array)

QtHardMon


QtHardMon@mskpcx18571
File
Plugins
Settings
Help
<<
Devices:
oven
Device status
Device is open.
Close
Device properties
Device name
oven
Load Boards
Modules/Registers:
heater
heatingCurrent
temperatureReadback
supplyVoltages
Sort Modules/Registers
Autoselect previous register
Register properties
Register path
/heater/temperatureReadback
Dimension
Scalar
Values
Value
0
180,7500
Options
Continuous read
Read after write
Show plot window
Operations
Read
Write
Write to file
Read from file
ChimeraTK

DeviceAccess: Register Name Mapping

- DeviceAccess identifies registers by name
- Many devices use numerical addresses:
  - PCI Express identifies registers by address in a "Base Address Range" (BAR)
  - Dummies simulate devices in RAM

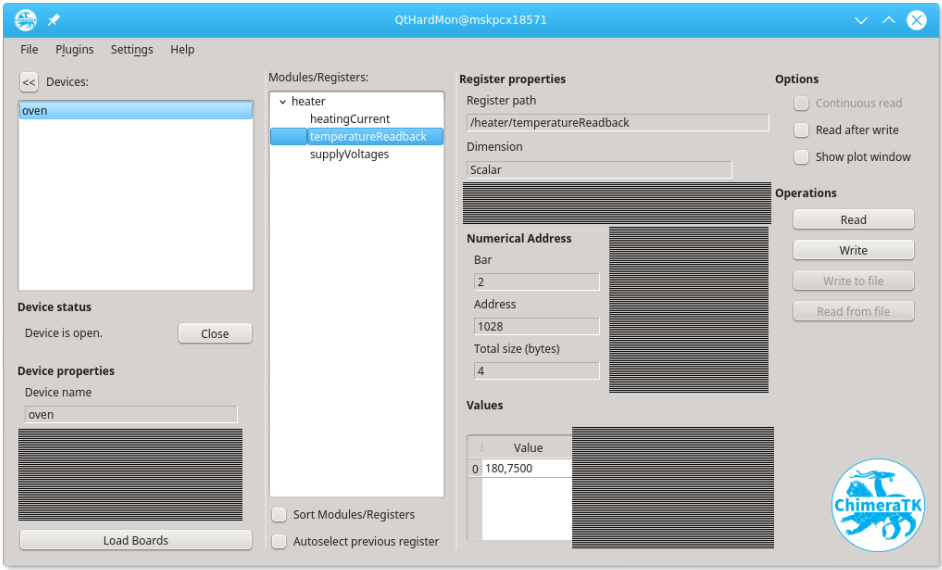
⇒ We need a mapping

Example map file

```
#name                       n_words address n_bytes BAR
heater.heatingCurrent             1    1024       4   2
heater.temperatureReadback        1    1028       4   2
heater.supplyVoltages             4    1032      16   2
```

- Map files are automatically created by the DESY (MSK) firmware framework
- Can easily be written manually

QtHardMon


QtHardMon@mskpcx18571
File
Plugins
Settings
Help
<<
Devices:
oven
Device status
Device is open.
Close
Device properties
Device name
oven
Load Boards
Modules/Registers:
heater
heatingCurrent
temperatureReadback
supplyVoltages
Sort Modules/Registers
Autoselect previous register
Register properties
Register path
/heater/temperatureReadback
Dimension
Scalar
Numerical Address
Bar
2
Address
1028
Total size (bytes)
4
Values
Value
0
180,7500
Options
Continuous read
Read after write
Show plot window
Operations
Read
Write
Write to file
Read from file
ChimeraTK

Important abstraction: Identify devices by an alias name

Example device map file

```
#alias_name    device_desciptor
oven           (pci:pcieunis6?map=oven.map)
#oven          (dummy?map=oven.map)
```

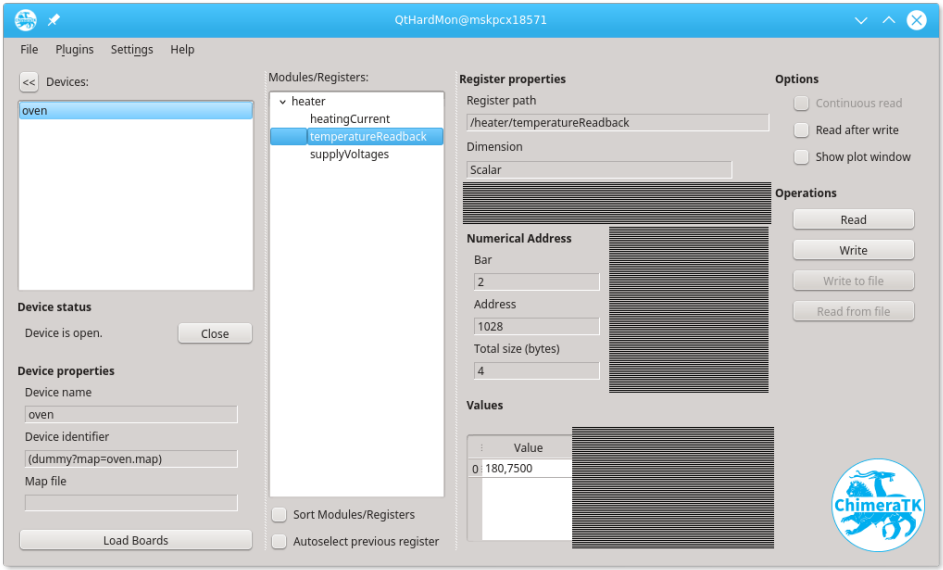
ChimeraTK Device Descriptor (CDD)

(backend_type:address?key1=value1&key2=value2)

Syntax

- Surrounded by parentheses – CDDs can be nested
- backend_type – Name of the backend, e.g. "pci", "dummy"
- address – Address of the device. The interpretation depends on the backend.
- keyX=valueX – List of key-value pairs. The interpretation depends on the backend.

QtHardMon


QtHardMon@mskpcx18571
File
Plugins
Settings
Help
<<
Devices:
oven
Device status
Device is open.
Close
Device properties
Device name
oven
Device identifier
(dummy?map=oven.map)
Map file
Load Boards
Modules/Registers:
heater
heatingCurrent
temperatureReadback
supplyVoltages
Sort Modules/Registers
Autoselect previous register
Register properties
Register path
/heater/temperatureReadback
Dimension
Scalar
Numerical Address
Bar
2
Address
1028
Total size (bytes)
4
Values
Value
0
180,7500
Options
Continuous read
Read after write
Show plot window
Operations
Read
Write
Write to file
Read from file
ChimeraTK

Accessing a Register

```
#include <ChimeraTK/Device.h>
#include <iostream>

int main(){

  ChimeraTK::setDMapFilePath("devices.dmap");
  ChimeraTK::Device d;
  d.open("oven");









}
```

Accessing a Register

```
#include <ChimeraTK/Device.h>
#include <iostream>

int main(){

  ChimeraTK::setDMapFilePath("devices.dmap");
  ChimeraTK::Device d;
  d.open("oven");

  auto heatingCurrent
    = d.getScalarRegisterAccessor<int>("heater/heatingCurrent");









}
```

- RegisterPath
  - Hierarchical register name
  - “/” as hierarchy separator
- RegisterAccessors

Accessing a Register

```
#include <ChimeraTK/Device.h>
#include <iostream>

int main(){

  ChimeraTK::setDMapFilePath("devices.dmap");
  ChimeraTK::Device d;
  d.open("oven");

  auto heatingCurrent
    = d.getScalarRegisterAccessor<int>("heater/heatingCurrent");

  heatingCurrent.read();
  std::cout << "Heating current is "
            << heatingCurrent
            << std::endl;



}
```

- RegisterPath
  - Hierarchical register name
  - “/” as hierarchy separator
- RegisterAccessors
  - read() and write() functions to synchronise with Device
  - Behaves like a primitive data type (or vector of it) in most use cases

Accessing a Register

```
#include <ChimeraTK/Device.h>
#include <iostream>

int main(){

  ChimeraTK::setDMapFilePath("devices.dmap");
  ChimeraTK::Device d;
  d.open("oven");

  auto heatingCurrent
    = d.getScalarRegisterAccessor<int>("heater/heatingCurrent");

  heatingCurrent.read();
  std::cout << "Heating current is "
            << heatingCurrent
            << std::endl;

  heatingCurrent += 3;
  heatingCurrent.write();

}
```

- RegisterPath
  - Hierarchical register name
  - "/" as hierarchy separator
- RegisterAccessors
  - read() and write() functions to synchronise with Device
  - Behaves like a primitive data type (or vector of it) in most use cases

Fixed Point Conversion

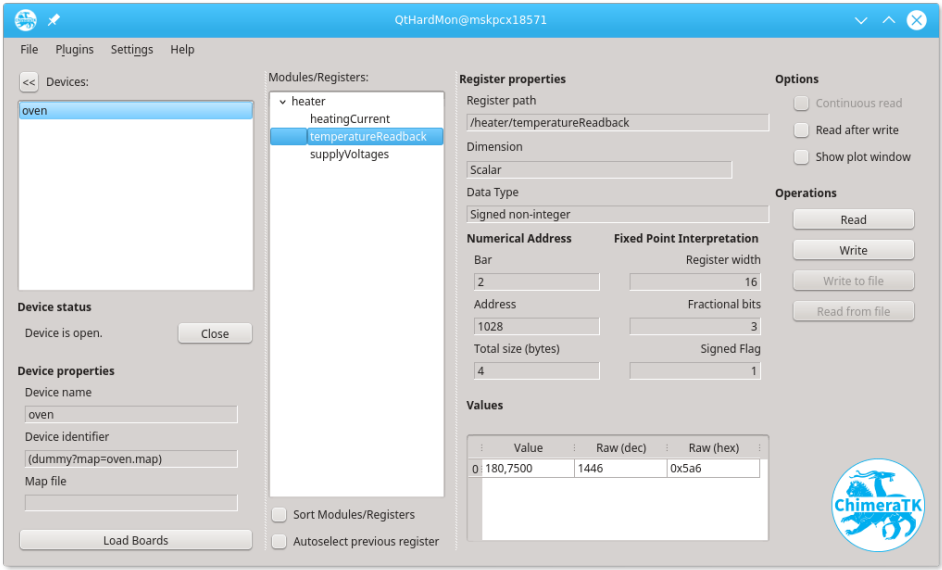
- Firmware often uses fixed-point arithmetic
- CPU uses floating point
- Transport layer (PCI Express) uses 32 bit words

⇒ Extend the mapping with conversion information*

Example map file

```
#name                       n_words address n_bytes BAR n_bits n_fractionalBits signed
heater.heatingCurrent             1     102       4   2     32                0      0
heater.temperatureReadback        1     102       4   2     16                3      1
heater.supplyVoltages             4     103      16   2     32                0      0
```

* Optional, default conversion is 32 bit signed integer, no fractional bits


QtHardMon@mskpcx18571
File
Plugins
Settings
Help
<<
Devices:
oven
Device status
Device is open.
Close
Device properties
Device name
oven
Device identifier
(dummy?map=oven.map)
Map file
Load Boards
Modules/Registers:
heater
heatingCurrent
temperatureReadback
supplyVoltages
Sort Modules/Registers
Autoselect previous register
Register properties
Register path
/heater/temperatureReadback
Dimension
Scalar
Data Type
Signed non-integer
Numerical Address
Bar
2
Address
1028
Total size (bytes)
4
Fixed Point Interpretation
Register width
16
Fractional bits
3
Signed Flag
1
Values
Value
Raw (dec)
Raw (hex)
0
180,7500
1446
0x5a6
Options
Continuous read
Read after write
Show plot window
Operations
Read
Write
Write to file
Read from file
ChimeraTK

DeviceAccess: Float Accessor

```
#include <ChimeraTK/Device.h>
#include <iostream>

int main(){

  ChimeraTK::setDMapFilePath("devices.dmap");
  ChimeraTK::Device d;
  d.open("oven");

  auto temperature
    = d.getScalarRegisterAccessor<float>("heater/temperatureReadback");

  temperature.read();
  std::cout << "Readback temperature is " << temperature << std::endl;




}
```

DeviceAccess: 1D Accessors

```cpp
#include <ChimeraTK/Device.h>
#include <iostream>

int main(){

  ChimeraTK::setDMapFilePath("devices.dmap");
  ChimeraTK::Device d;
  d.open("oven");

  auto supplyVoltages
    = d.getOneDRegisterAccessor<int>("heater/supplyVoltages");

  supplyVoltages.read();

  std::cout << "Supply voltages are ";
  for (auto voltage : supplyVoltages){
    std::cout << voltage << " ";
  }
  std::cout << std::endl;
}
```

Question

Each register can be accessed as scalar, 1D or 2D. And I have to chose the data type.
How do I know what to pick?

RegisterInfo

- `getRegisterName()`
- `getNumberOfDimensions()`
  - 0 (=scalar), 1, 2
- `getNumberOfElements()`
  per channel if 2D
- `getNumberOfChannels()`
- `getDataDescriptor()`

DataDescriptor

- `getFundamentalType()`
  - numeric, string, boolean, nodata, undefined
- `isIntegral()`
- `isSigned()`
- `nDigits()`
  Number of decimal digits for display purposes.
  E.g. int8_t is will return 4
  (-127..128 $\hat{=}$ 3 digits plus sign)
- `nFractionalDigits()`
  Number of digits after the decimal dot.

Available Backends

Build-in

pci	PCI-Express
rebot	***Re**gister **b**ased **o**ver **T**CP*, lightweigt, TPC/IP based inhouse protocol
subdevice	Show part of address space as own logical device
logicalNameMap	Rename and re-organise registers
dummy	Simulate register space in RAM
sharedMemoryDummy	Dummy with address space in shared memory

Loadable plugins

doocs	DOOCS client interface[1]
modbus	Modbus client interface (under development)

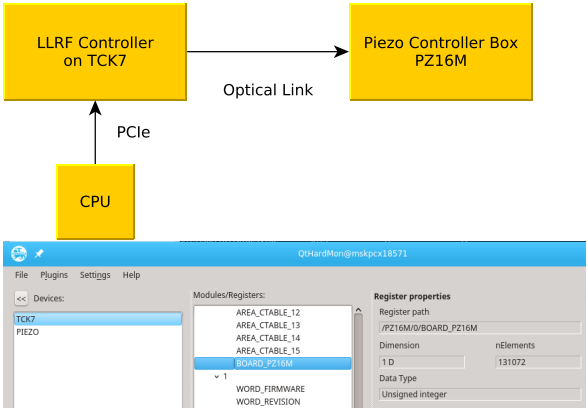
Various dedicated dummies for tests

Planned

epics	Native EPICS client interface
opu-ua	OPC UA client interface

[1] Can also read EPICS channels

Subdevice Backend



- Firmware maps the register space of a subdevice (PIEZO) into a 1D register of it's own address space (TCK7).
  (Usually offset address in a numerically addressed space).
- Both devices have firmwares which evolve separately.
- The Subdevice backend allows to access the subdevice through the parent device as a separete logical entity.

⇒ Separate map file to describe the subdevice.

```
#alias device_descriptor
TCK7   (pci:llrfutcs4?map=llrf_ctrl_tck7b_acc1_r2097.mapp)
PIEZO  (subdevice:area,TCK7,PZ16M.0.BOARD_PZ16M?map=piezo_pz16m_acc1_r2323.mapp)
```

What can I do with Logical Name Mapping?

Arrange the register content to logically match your application

- Rename registers
- Add constant registers or dummy registers
- Extract channels from 2D registers and give it a name
- Extract scalars from 1D registers and give it a name
- Extract bits from a scalar register

Logical Name Mapping Example

Abstract away cabling details

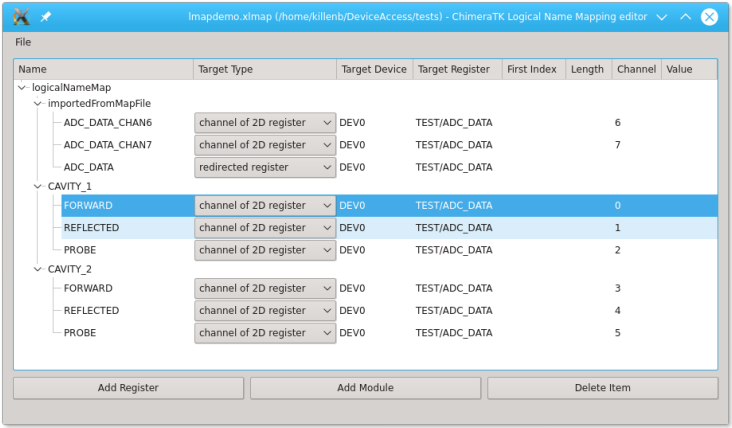
- Cavity with 3 signals: Forward, reflected, probe.
- Recorded on 8 channel ADC (2D array with data): adc_data[8][1024]
- Cabling:
  - Cavity 1 on channels 0..2
  - Cavity 2 on channels 3..5

Logical name mapping

adc_data[0]	→	cavity1/forward
adc_data[1]	→	cavity1/reflected
adc_data[2]	→	cavity1/probe
adc_data[3]	→	cavity2/forward
adc_data[4]	→	cavity2/reflected
adc_data[5]	→	cavity2/probe

You don't have to fiddle with channel numbers in you cavity module.
Use the logical names *forward, reflected, probe*.

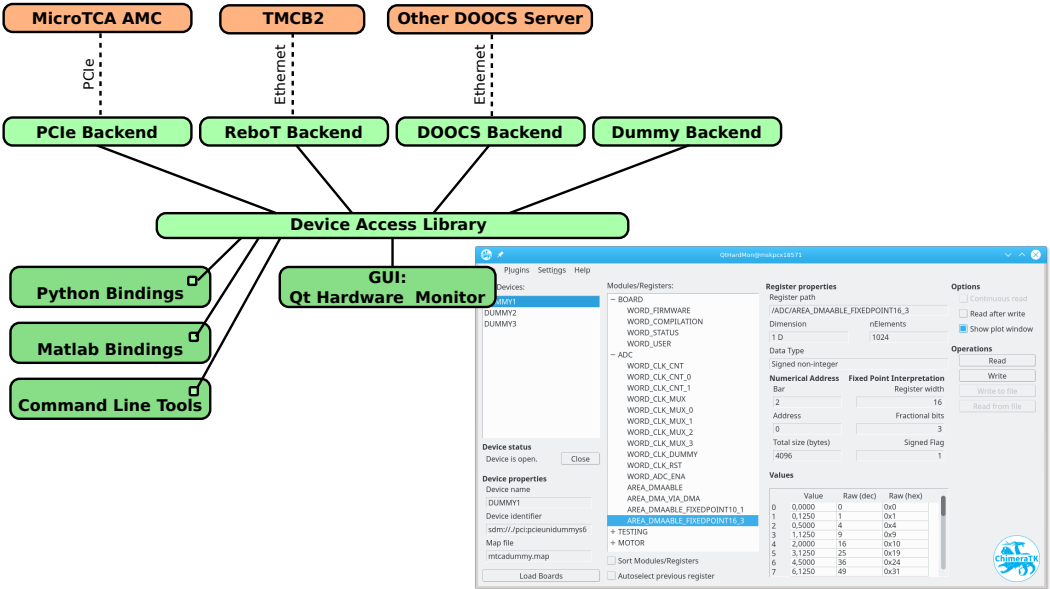
LMAP Editor



- Import map file as starting point
- Modify the mapping
- Save and load logical name mapping

Tool under development.
Please give feedback or implement missing features.

The DeviceAccess Library Tools



Python bindings

C++

```cpp
#include <ChimeraTK/Device.h>

int main(){
  ChimeraTK::setDMapFilePath("devices.dmap");
  ChimeraTK::Device d;
  d.open("oven");

  //"inefficient" shortcut to read a variable
  int temperature = d.read<float>("heater/temperatureReadback");
}
```

Python

```python
import mtca4u

mtca4u.set_dmap_location('devices.dmap')
d = mtca4u.Device('oven')

temperature = d.read('heater','temperatureReadback')
```

Matlab

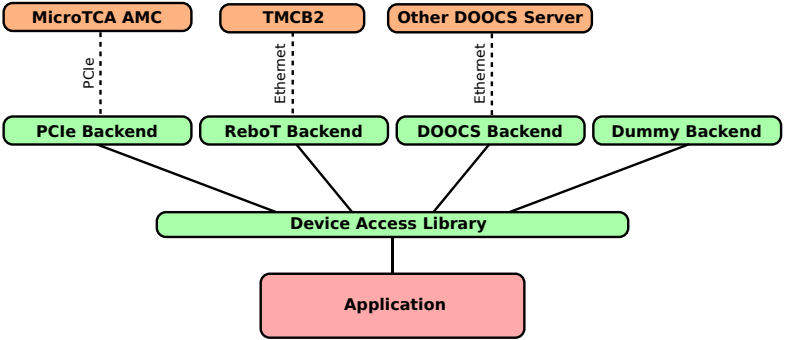
```
mtca4u.setDMapFilePath('devices.dmap')
d = mtca4u('oven')

temperature = d.read('heater','temperatureReadback')
```

Command line

```
$ mtca4u read oven heater temperatureReadback
```

- All needed arguments in one call
- Takes the first dmap-file it finds :-O

Next Step: Write the Application



Resources

Software Repositories

All software is published under the GNU GPL or the GNU LGPL.

- ChimeraTK source code: https://github.com/ChimeraTK
- Ubuntu 16.04 packages are available in the DESY DOOCS repository.

Documentation and Tutorials

- API documentation https://chimeratk.github.io/
- DeviceAccess live demo https://github.com/killenb/DeviceAccess_live_demo