

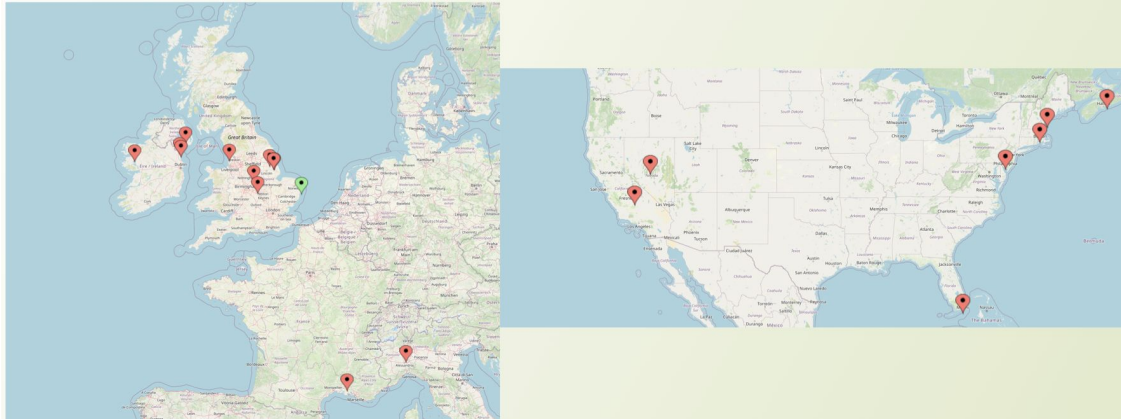
CSS Isotek Discussion

Charlie Elliott

Housekeeping

- Who Am I (quick re-introduction)
- The Test
 - Design
 - Cool parts
 - Bit's i'd do different
 - Challenges
 - Fun Bits...
 - Demo & Discussion
- Q&A and extra discussion

Rosindale Systems



Naim Audio



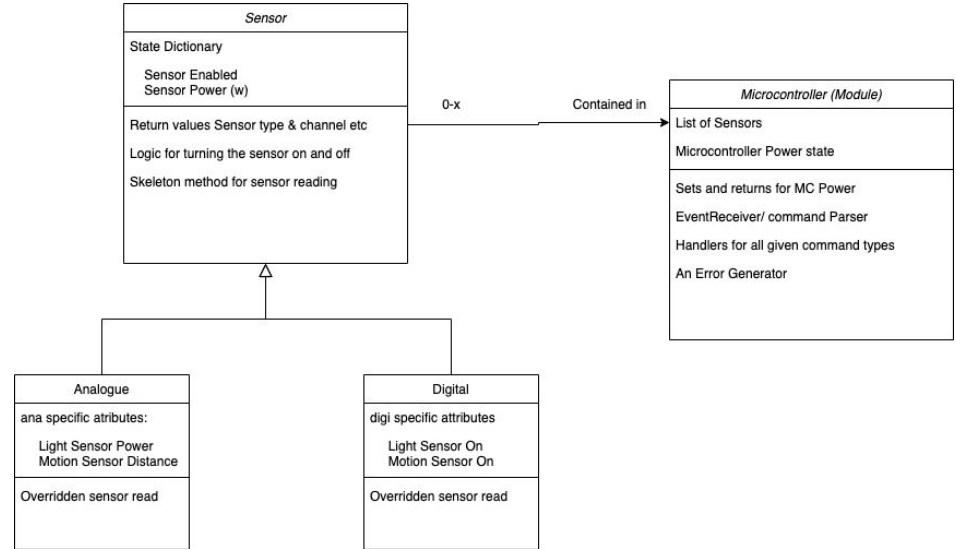
Access to the code:



<https://github.com/TheLastCD/HardwareModule>

My Approach

- Build supporting class that duplicates the sensors and the microcontroller
- Mock the returns and model the behaviour in the implementation



Notable Parts

- Using Enums to provide easy abstraction
- Making use of regex:
 - `re r'^\^.*\n$'`
 - Easy disqualification
- Using a factory to build the sensors
 - Easily scalable

```
class SensorType(Enum):
    DIGI = "D"
    ANA = "A"

class SensorDirection(Enum):
    INPUT = "I"
    OUTPUT = "O"

class ProtocolCommands(Enum):
    ECHO = "E" # Only requires Sequence Number
    INPUT = "I" # Read inputs or return current output value on a given output
    OUTPUT = "O" # sets the value of a given output
    PUS = "P" # Only requires sequence number

class ProtocolReturnCode(Enum):
    """
    return codes are a fixed size coming back in
    """
    OK = "OK "
    ERROR = "ERR"
    RANGE = "RNG"
```

What i'd do differently

- Not being able to use switches
 - Syntactically looks worse in my opinion

```
# dictionary goes: name/alias, relative channel (IE: analogue channel 1), sensortype
sensorDict = {
    "Digi0": [0, SensorType.DIGI],
    "Digi1": [1, SensorType.DIGI],
    "Ana0": [0, SensorType.ANA],
    "Ana1": [1, SensorType.ANA]
```

- Using a dict to instantiate
 - it's simple
 - But rather use a manifest in xml ...

```
<sensor>
  <GPIO>1</GPIO>
  <type>Analogue</type>
  <relativeChannel>0</relativeChannel>
  <notes>lorem ipsum</notes>
</sensor>
```


Challenges

- No Hardware
 - Lack of ability to tinker
 - Having to make estimations and assumptions i'm not happy with
 - Hardware orients the software makes understanding easier
- Scope creep
 - I would have made the server if i wasn't told not too
 - Wanted to do Parallel solutions but they're not necessary

Fun Bits...

- Implementing protocols somewhat from scratch
 - Supper fun to chase down how it works
 - If only i was reading it straight from hardware
 - Fun parts of working at naim
- Made me rethink how i've been writing code for microcontrollers
 - Was a great learning experience
- Having something to mull over
 - The bouldering session was more fun

2 brief questions on the specification

- Why `\n` and not `\0`?
 - It's more standard especially in C/C++
- The arbitrary length of SS
 - Seems like an area where we could cause malloc issues/ buffer overflows on the microcontroller
 - Makes catching errors on the python side harder as well

Quick Demo

Any Questions