Hardware Triggered Scanning: Exercise Solutions

Emma Arandjelovic

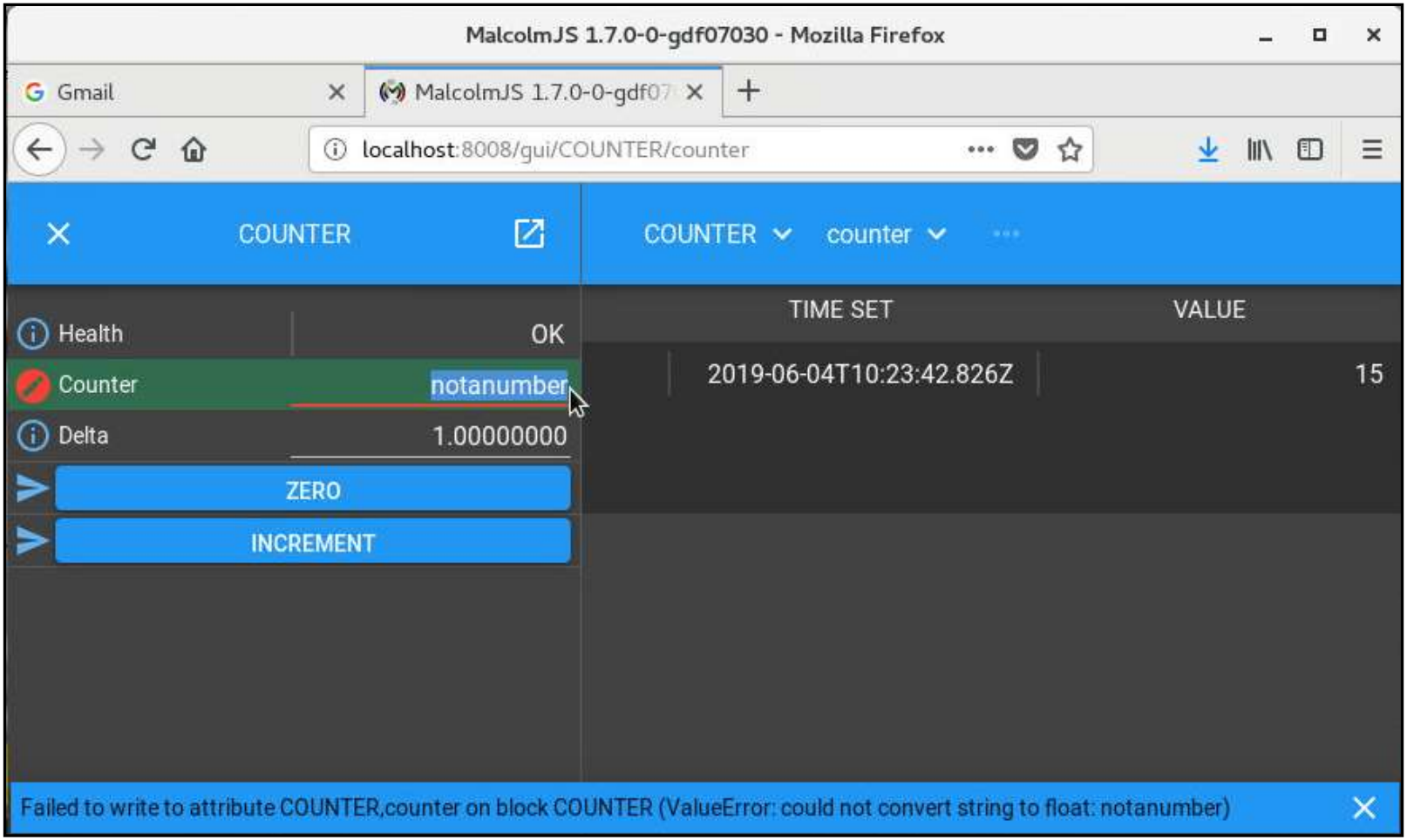
Observatory Sciences Limited

Exercise 1

Modify demo/parts/hellopart.py:

```
def greet(self, name=None, sleep=0):
    # type: (AName, ASleep) -> AGreeting
    """Optionally sleep <sleep> seconds, then return a greeting to <name>"""
        print("Manufacturing greeting...")
        sleep_for(sleep)
        if name == "":
            self.error()
        greeting = "Hello %s" % name
        return greeting
```

Exercise 2


MalcolmJS 1.7.0-0-gdf07030 - Mozilla Firefox
Gmail
MalcolmJS 1.7.0-0-gdf07
localhost:8008/gui/COUNTER/counter
COUNTER
COUNTER
counter
Health
OK
Counter
notanumber
Delta
1.00000000
ZERO
INCREMENT
TIME SET
VALUE
2019-06-04T10:23:42.826Z
15
Failed to write to attribute COUNTER,counter on block COUNTER (ValueError: could not convert string to float: notanumber)

Exercise 3

- Modify demo/parts/counterpart.py to add the new method:

```
def decrement(self):
    """Subtract delta from the counter attribute"""
    self.counter.set_value(self.counter.value - self.delta.value)
```

- Register this new method with the PartRegistrar in the setup method:

```
registrar.add_method_model(self.decrement)
```

Exercise 4

Modify *motion_block.yaml* to create the new parameters and pass them to MotorMovePart:

```
- builtin.parameters.float64:
    name: high_limit
    description: High s/w limit
    default: 0
- builtin.parameters.float64:
    name: low_limit
    description: Low s/w limit
    default: 0
- demo.parts.MotorMovePart:
    name: x
    mri: $(mri):COUNTERX
    hi: $(high_limit)
    lo: $(low_limit)
```

Provide values for the limits in *DEMO-MOTION.yaml*:

```
- demo.blocks.motion_block:
    mri: MOTION
    config_dir: /tmp
    high_limit: 20
    low_limit: 0
```

Exercise 4 (continued)

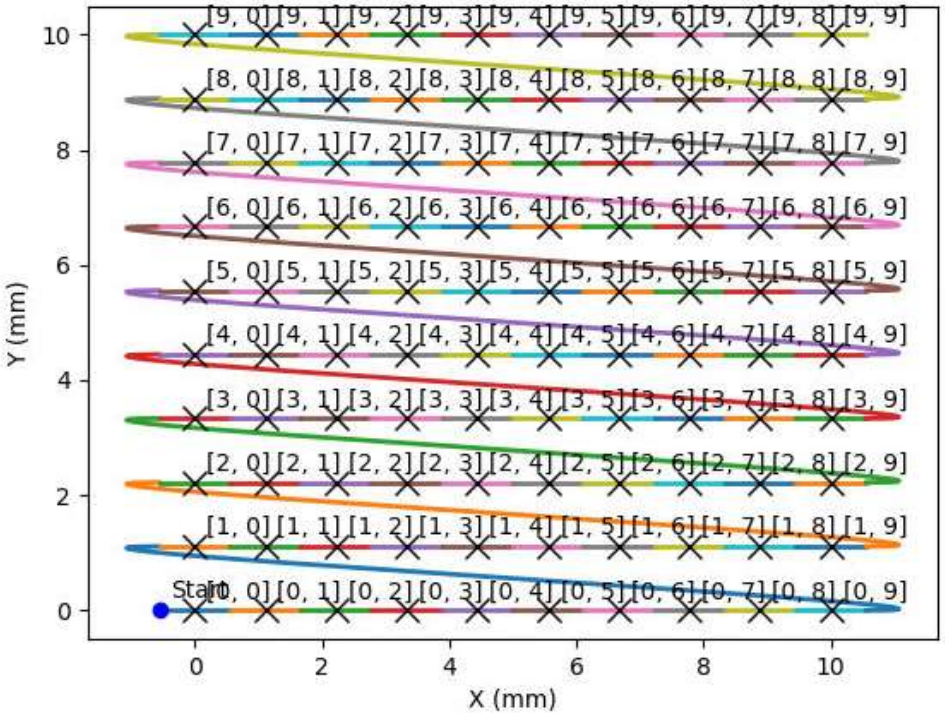
- Modify *motormovepart.py* to use the new parameter:

```
with Anno("High s/w limit"):
    AHi = float
with Anno("Low s/w limit"):
    ALo = float

def __init__(self, name, mri, hi=0, lo=0):
        ....
        self.hi = hi
        self.lo = lo
        if (hi==0 and lo==0):
            self.limits_enabled = False
        else:
            self.limits_enabled = True

def move(self, context, demand):
        ....
        if (self.limits_enabled and (demand>self.hi or demand<self.lo)):
            raise RuntimeError("Motor demand out of range!")
```

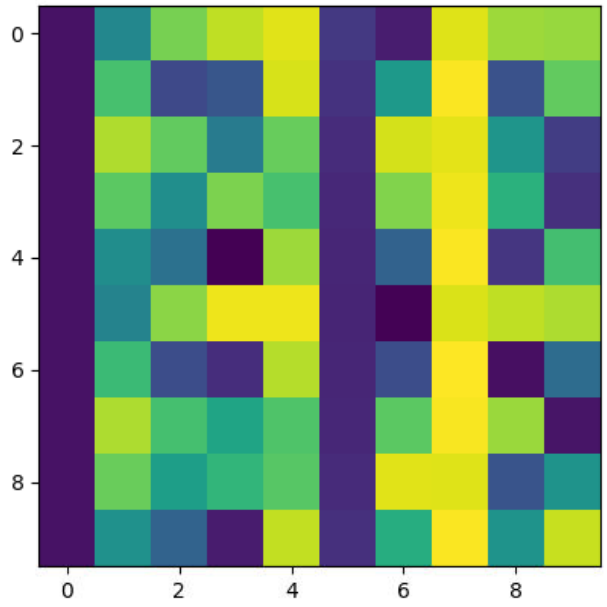
Exercise 5

Example for a 10x10 map:

```
>>> xline = LineGenerator("x","mm",0,10,10)
>>> yline = LineGenerator("y","mm",0,10,10)
>>> generator = CompoundGenerator([yline,xline],[],[],duration=0.01)
```




```
>>> im.shape
(10, 10, 1, 1)
```



Exercise 6

```
[p47user@localhost demo]$ h5dump -d /entry/NDAttributes/NDArrayUniqueId /tmp/RAMP.h5
HDF5 "/tmp/RAMP.h5" {
DATASET "/entry/NDAttributes/NDArrayUniqueId" {
  DATATYPE  H5T_STD_I32LE
  DATASPACE  SIMPLE { ( 6, 6, 1, 1 ) / ( H5S_UNLIMITED, H5S_UNLIMITED, 1, 1 ) }
  DATA {
  (0,0,0,0): 1,
  (0,1,0,0): 2,
  (0,2,0,0): 3,
  (0,3,0,0): 4,
  (0,4,0,0): 5,
  (0,5,0,0): 6,
  (1,0,0,0): 7,
  (1,1,0,0): 8,
  (1,2,0,0): 9,
  (1,3,0,0): 10,
  (1,4,0,0): 37,
  (1,5,0,0): 38,
  (2,0,0,0): 39,
  (2,1,0,0): 40, .....
```