Embedded Systems

Embedded Programming and Communication

31 January 2017

Last time…

- Talking to an ESP8266 running micropython
- Interfacing with I^{2}C

This time…

- Communication for IoT
- Introduction to real-time systems

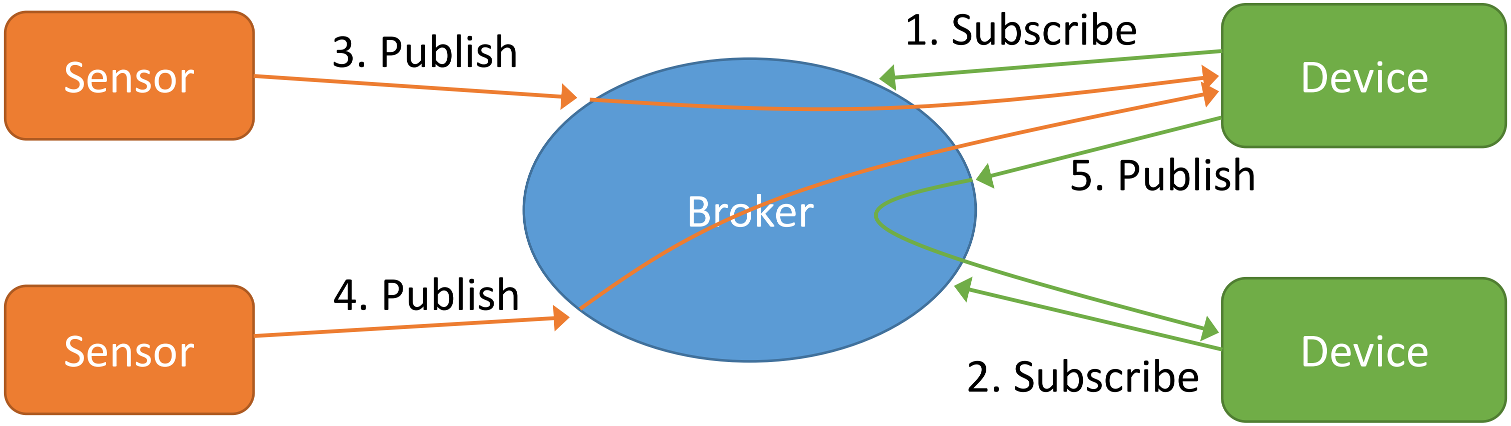
Communication for IoT

What do we want from IoT communication?

- Lightweight
- Secure
- Tolerant of poor connections
- Scalable for large numbers of devices

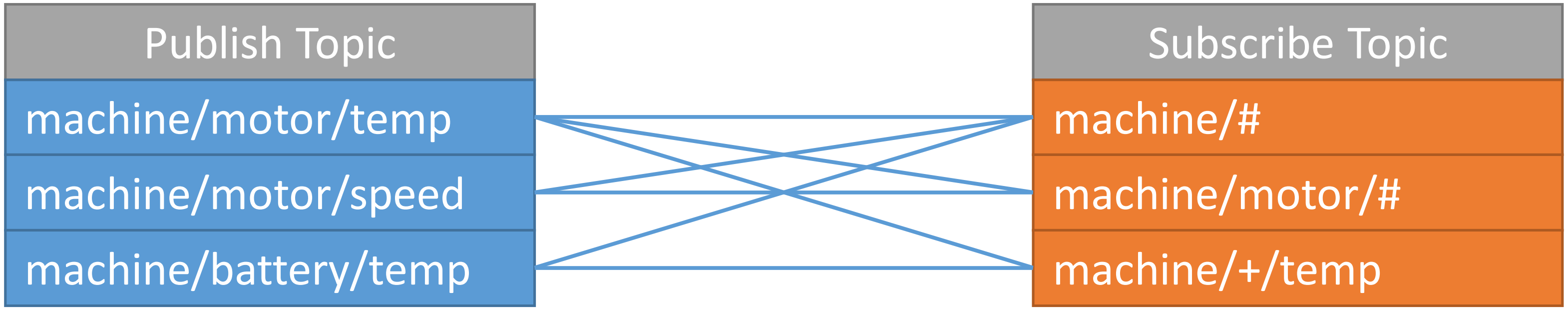
MQTT

- MQ Telemetry Transport
  - MQ (message queuing) is an IBM product family for comms in distributed systems
  - ISO standard
- Publish/Subscribe model



MQTT Topics

- Every message has a topic
- Subscribers receive every message that matches a topic
- Topics are hierarchical



Publish Topic
machine/motor/temp
machine/motor/speed
machine/battery/temp

Subscribe Topic
machine/#
machine/motor/#
machine/+/temp

Full spec here:
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html

MQTT in micropython

```
from umqtt.simple import MQTTClient
client = MQTTClient(CLIENT_ID,BROKER_ADDRESS)
client.connect()
```

Name of your device (string) `machine.unique_id()` returns a unique identifier

Broker that you want to connect to (string)

```
client.publish(TOPIC,bytes(data,'utf-8'))
```

Message topic (string)

data payload

MQTT sends bytes (chars)
Need to encode a string that contains non-ASCII characters

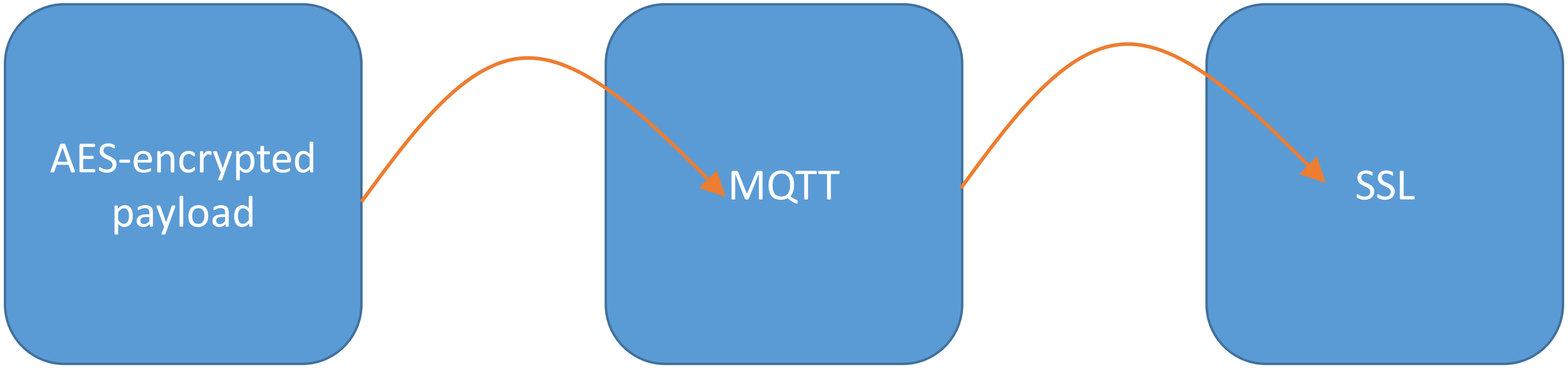
Details (including subscribing to topics):
https://github.com/micropython/micropython-lib/tree/master/umqtt.simple

MQTT on other platforms

- Mosquitto – an open source broker and client
  - set up your own broker
  - view and publish test messages
  - https://mosquitto.org/
- Paho
  - library for desktop Python
  - `pip install paho-mqtt`
  - https://pypi.python.org/pypi/paho-mqtt/1.1
- Mobile
  - MyMQTT

Encryption

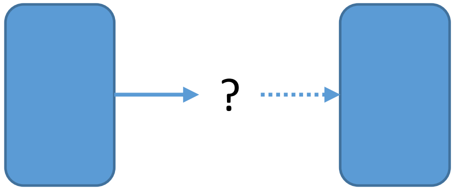
- MQTT standard does not include encryption
- But can encrypt payload or wrap MQTT in SSL
  - Optional for coursework



MQTT Quality of Service (QOS)

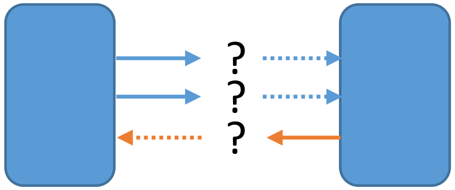
- For managing with unreliable connections

QoS 0



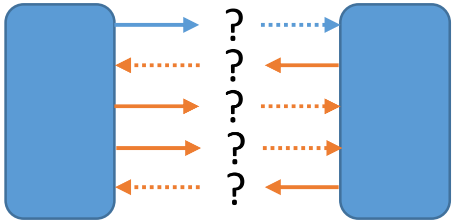
At most once

QoS 1



At least once

QoS 2



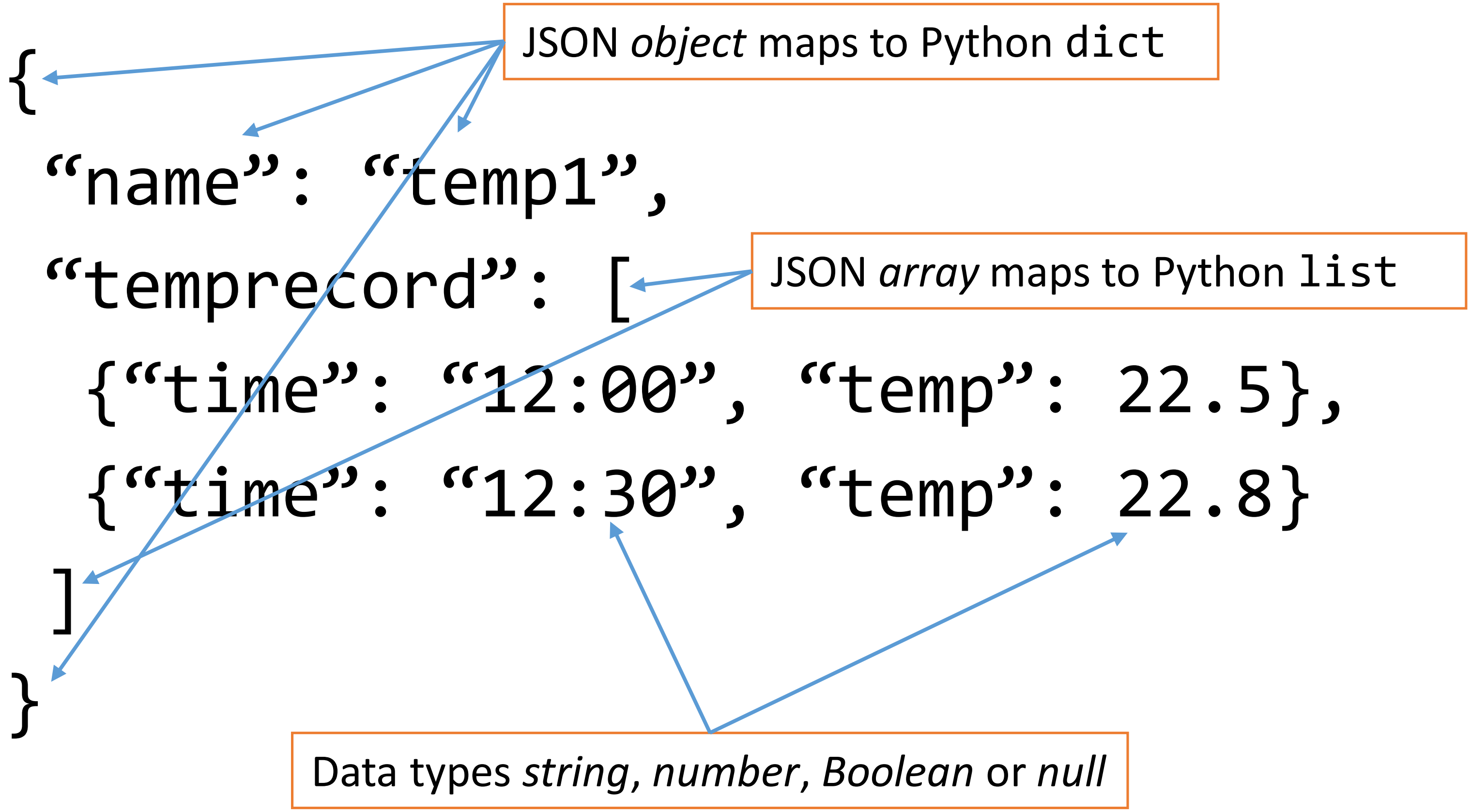
Exactly once

QoS 0 and 1 are currently supported in micropython

JSON Payload

- JSON = Javascript Object Notation
    - Widely used to pass data to/from web browsers
    - Increasingly used in databases
- Standard for data *serialisation*
- Suitable for passing data over MQTT
    - Flexible
    - Less verbose than XML or others
    - Translates readily to/from python objects

JSON example



```
{
 “name”: “temp1”,
 “temprecord”: [
  {“time”: “12:00”, “temp”: 22.5},
  {“time”: “12:30”, “temp”: 22.8}
 ]
}
```

JSON *object* maps to Python `dict`

JSON *array* maps to Python `list`

Data types *string*, *number*, *Boolean* or *null*

```
payload = json.dumps({‘name’:’temp1’, ’temprecord’:tempdata})
```

Coursework 1 Comms

What communication?

- What data are you sending?
- How often do you send it?
- Who/what will receive the data?
- What will they do with it?
- Does the IoT device subscribe to anything?

Connecting ESP8266 to a network

```
import network
ap_if = network.WLAN(network.AP_IF)
ap_if.active(False)

sta_if = network.WLAN(network.STA_IF)
sta_if.connect('<essid>', '<password>')
```

Disable automatic access point to reduce overheads

Connect to a specified WiFi network

https://docs.micropython.org/en/latest/esp8266/esp8266/tutorial/network_basics.html

MQTT broker

- ESP8266 cannot connect to WPA2 enterprise
  - Including college WiFi
- Closed WiFi network for labs and demo: `EEERover`
  - Password: `exhibition`
- MQTT broker: `192.168.0.10`
- Use MQTT topic: `esys/<group name>/…`

- Where should I put this at other times?

Other useful libraries

- `machine.RTC` – real time clock for finding the date and time
- `uheapq` – put items into a heap and retrieve them in priority order
- `socket` – send raw data over the network
  - e.g. request a webpage: `s.send(bytes('GET /%s HTTP/1.0\r\nHost: %s\r\n\r\n' % (path, host), 'utf8'))`
- `math` – mathematical functions
- `machine.sleep()` – save power

Website marketing concept

- Design a website* that shows
  - What your product does
  - How it does it
  - Why someone should buy it

*Not a functioning website – just a static document

Rough/sketch graphics are fine

Borrowed graphics are fine for non-technical details

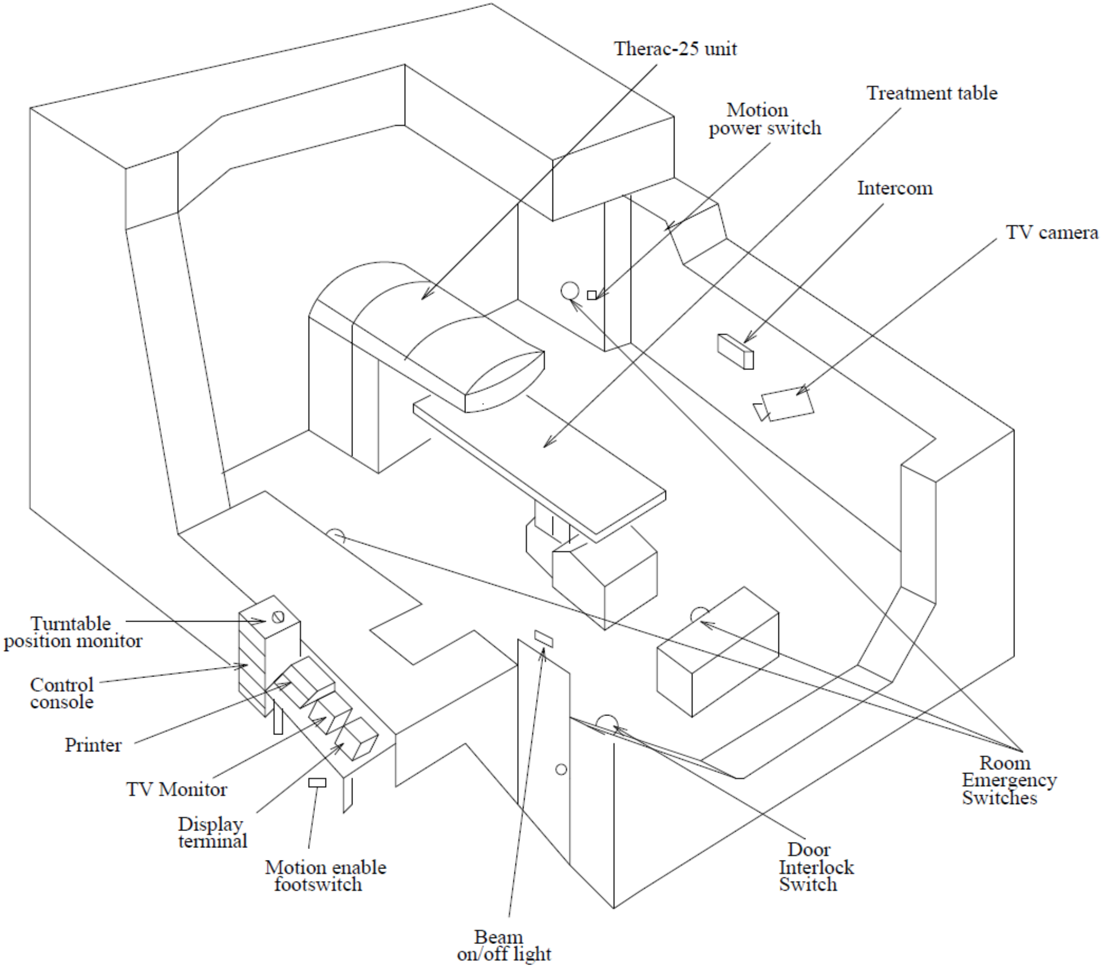
http://www.cattle-watch.com/

https://www.kickstarter.com/projects/582920317/hidrateme-smart-water-bottle

Embedded Systems Part II

Real-time programming

Case study: Therac-25

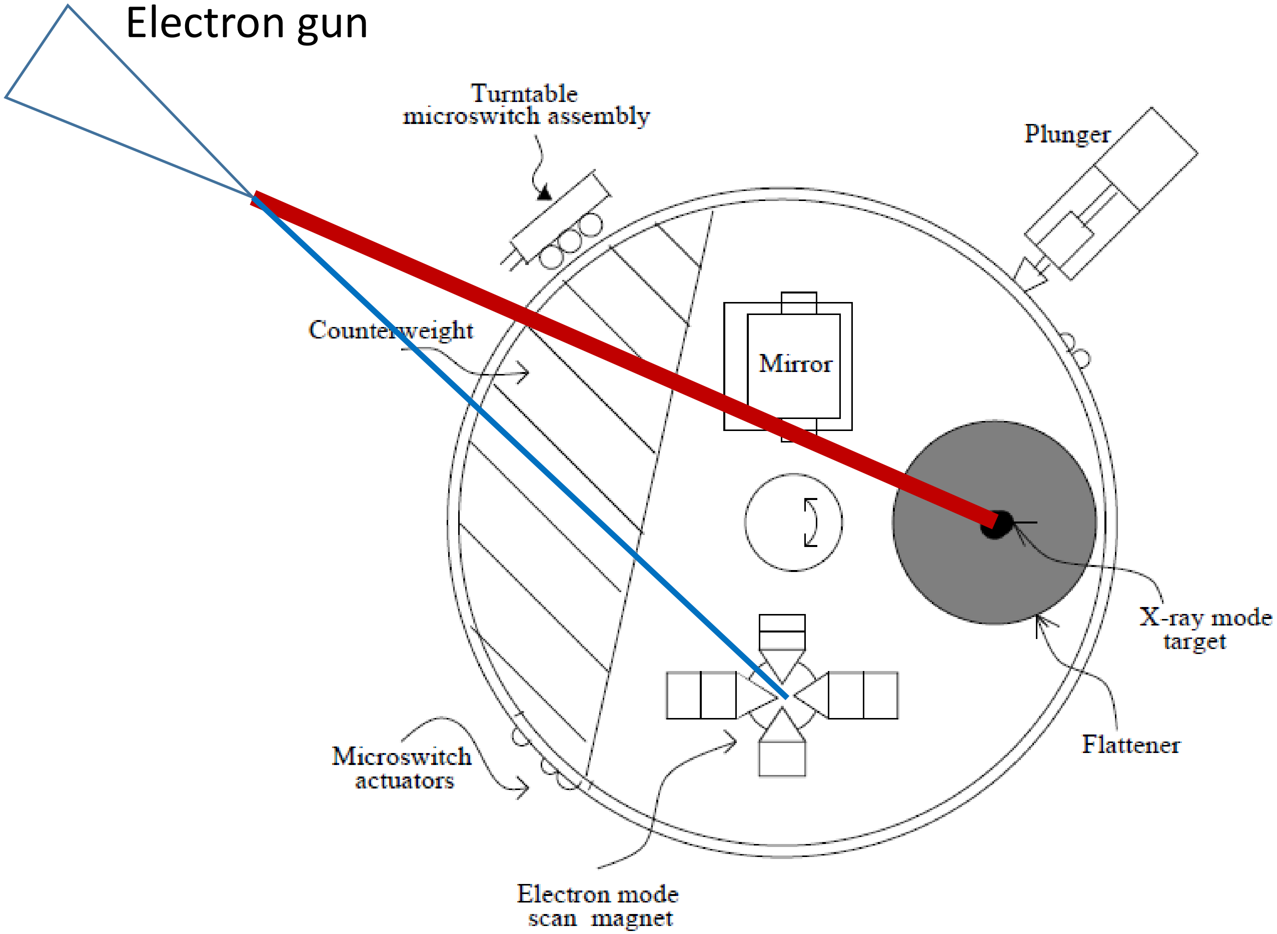


Leveson, *Safeware: System Safety and Computers* (1995)

What happened?

- 3 killed, 3 seriously injured between 1985 and 1987
- Introduction of computer control instead of manual set-up
  - Reduced set-up time
  - Reduced chance of operator error
  - More versatile machine
  - Removed need for complex mechanical interlocks!
- Two operating modes
  - Electron beam
  - X-Ray

What happened?



Leveson, *Safeware: System Safety and Computers* (1995)

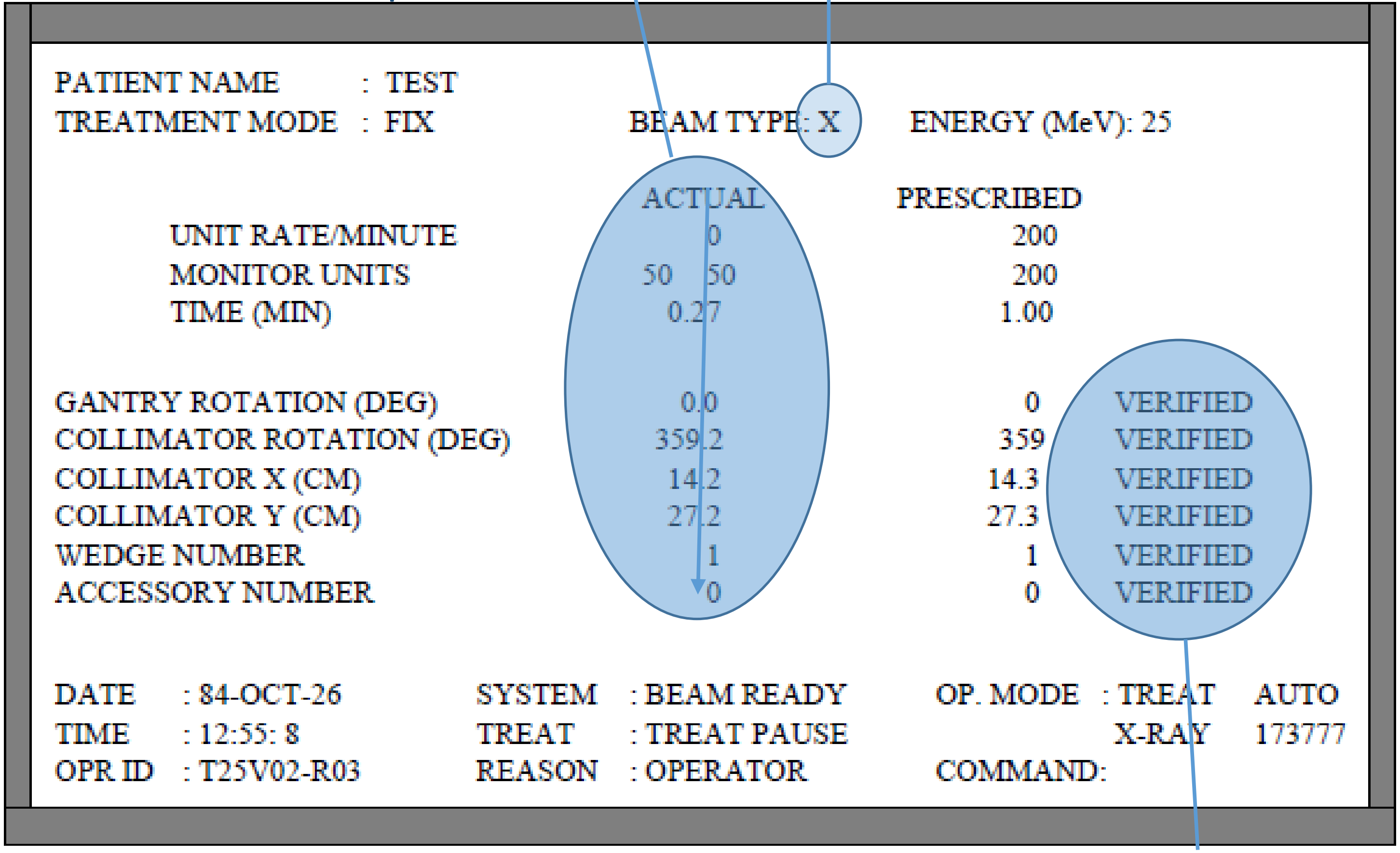
Why did it happen?

- Poor software engineering
- Written in assembly language
- Code reused with assumption that it worked
- No independent code review
- Error codes produced with no explanation
  - Operators learned to ignore them
- Code not suitable for automated test

Race condition

1. Operator accidentally selects X-Ray instead of electron

2. Operator corrects error and rapidly hits ‘enter’ to confirm other parameters

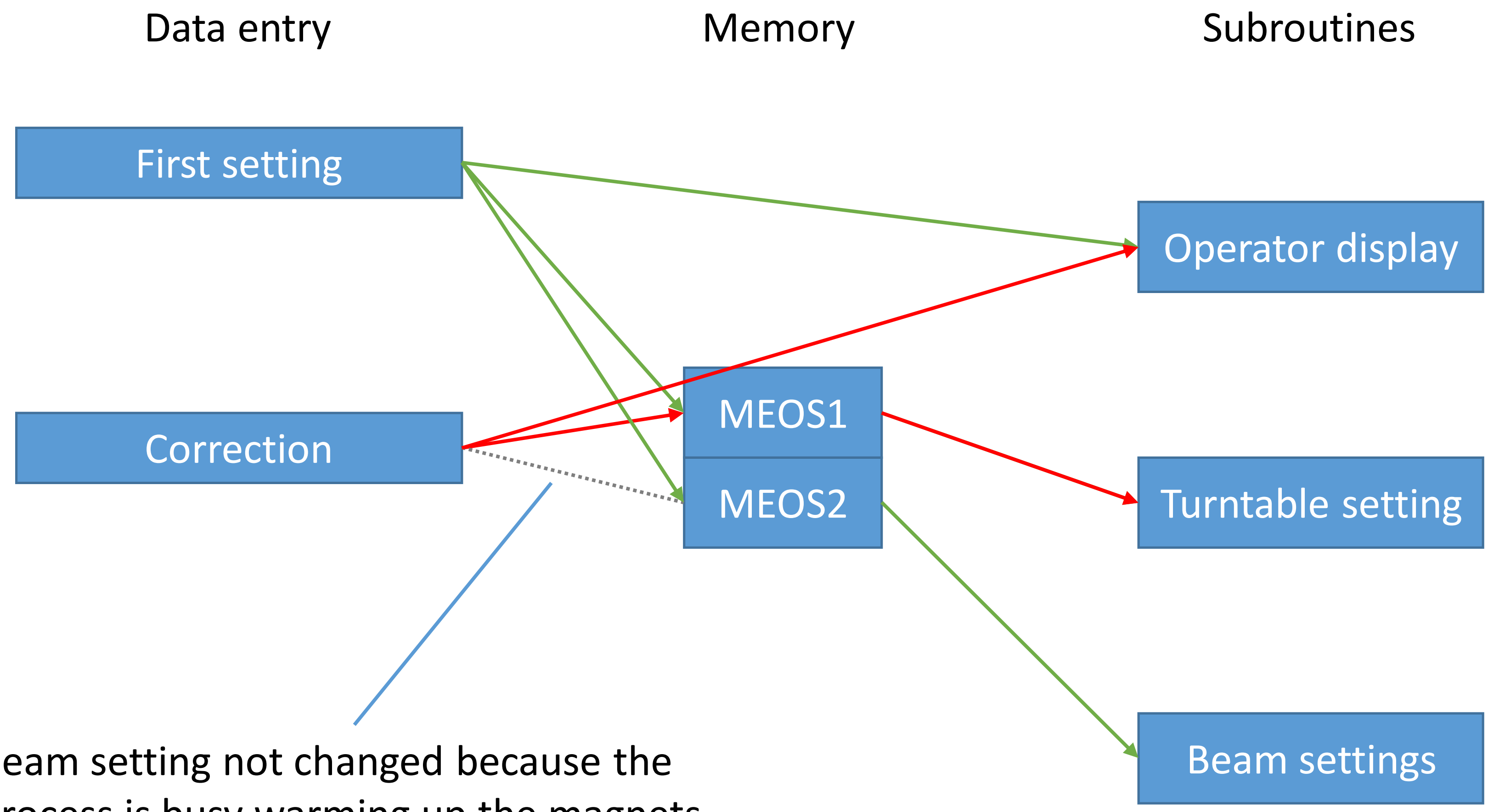


```
PATIENT NAME        : TEST
TREATMENT MODE      : FIX              BEAM TYPE: X      ENERGY (MeV): 25

                                        ACTUAL         PRESCRIBED
     UNIT RATE/MINUTE                         0             200
     MONITOR UNITS                       50   50            200
     TIME (MIN)                             0.27            1.00

GANTRY ROTATION (DEG)                       0.0             0      VERIFIED
COLLIMATOR ROTATION (DEG)                 359.2           359      VERIFIED
COLLIMATOR X (CM)                          14.2            14.3    VERIFIED
COLLIMATOR Y (CM)                          27.2            27.3    VERIFIED
WEDGE NUMBER                                  1             1      VERIFIED
ACCESSORY NUMBER                              0             0      VERIFIED

DATE     : 84-OCT-26      SYSTEM  : BEAM READY      OP. MODE  : TREAT    AUTO
TIME     : 12:55: 8       TREAT   : TREAT PAUSE                X-RAY     173777
OPR ID   : T25V02-R03     REASON  : OPERATOR        COMMAND:
```

3. Machine indicates verified already so operator begins treatment

4. Machine reports ‘MALFUNCTION 54’. Low dose reported so treatment repeated

Race condition



Beam setting not changed because the
process is busy warming up the magnets

What can we learn?

- Mostly a lesson in software engineering
- But also highlights pitfalls in embedded software
- How do we synchronise between software and the real world?
    - Real-world events are asynchronous – they could happen at any time
- How do we synchronise between concurrent tasks within the software?
    - State must be consistent between processes
    - Shared memory must be managed properly

On Thursday…

- Second lab session on coursework 1

Next week…

- Real-time programming