Environment Monitoring with intrusion detection

Projektname “SecureNsense”

Sensor Networks Lab, WS 2014/15

Group 8:
Martin Kessel
Eridy Lukau
Alexander Platz

Agenda

- ❑ Application
- ❑ System Architecture
- ❑ Implementation
- ❑ Challenges
- ❑ Live Demo

Idea

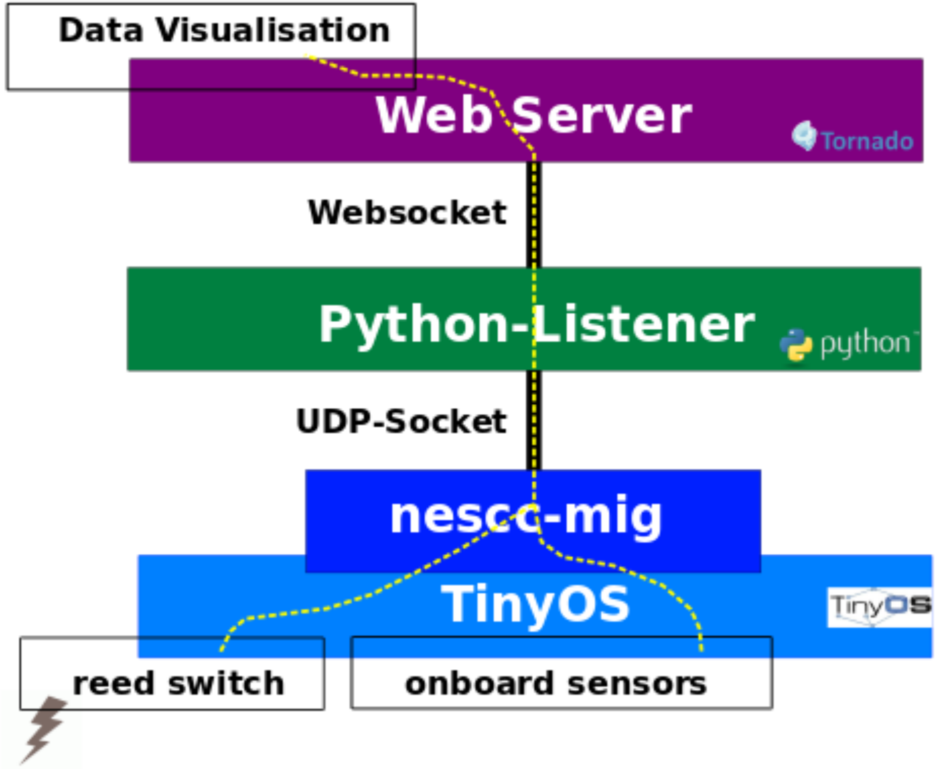
- Monitoring of environment with existing on-board sensors
- Realising an intrusion detection with a reed switch
- sending data to a web-interface
- Visualising data and intrusion on web-interface

Application scenarios

- Warehouse → night guard
- Greenhouse / plantation → gardener / farmer
- Office / Hotel → Staff
- Apartment → Owner / tenant

System Architecture

System Modules



Data Visualisation

Web Server

Websocket

Python-Listener

UDP-Socket

nescc-mig

TinyOS

reed switch

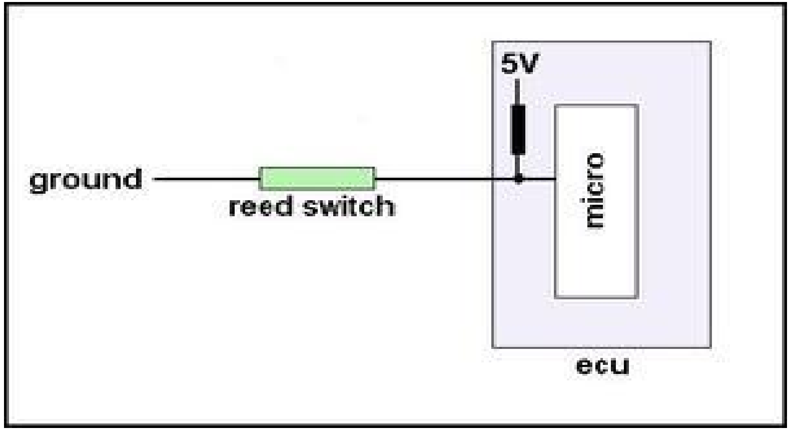
onboard sensors

Module Tasks

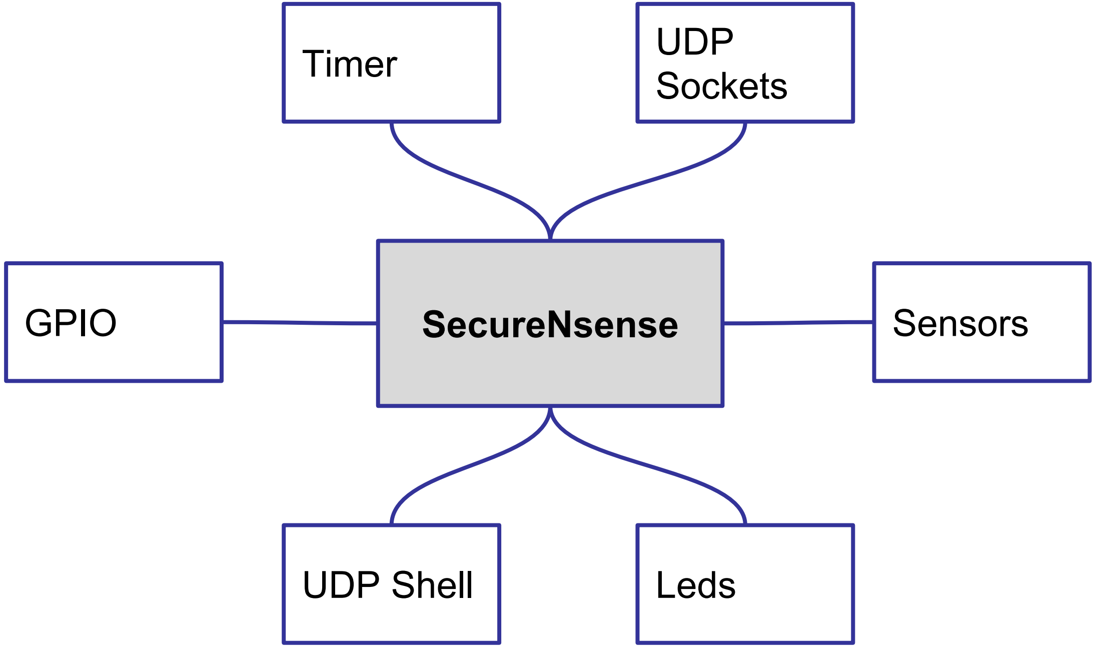
- **Webserver**
  - Tornado Websockets
  - fully asynchronous, event-based
  - less traffic and latency
- **Python Listener**
  - Request Handler
  - Distributes data into different channels via URIs
  - data processing and conversion
- **TinyOS + nescc mig**
  - collects data and sends it via UDP Socket
  - on-board sensors periodically
  - reed switch on interrupt

Node functionality

- Monitors environment via Temperature, Humidity and Light sensor
- Sensing period and alive message interval configurable via UDP shell
- Nodes exchange configuration on change
- Connects to a reed switch to detect intruders on e.g. doors or windows
- Intruder generates interrupt that leads to immediate alarm message



Node component overview



Hardware

- 1 telosB as a router
- 2 telosB as environment-sensing nodes

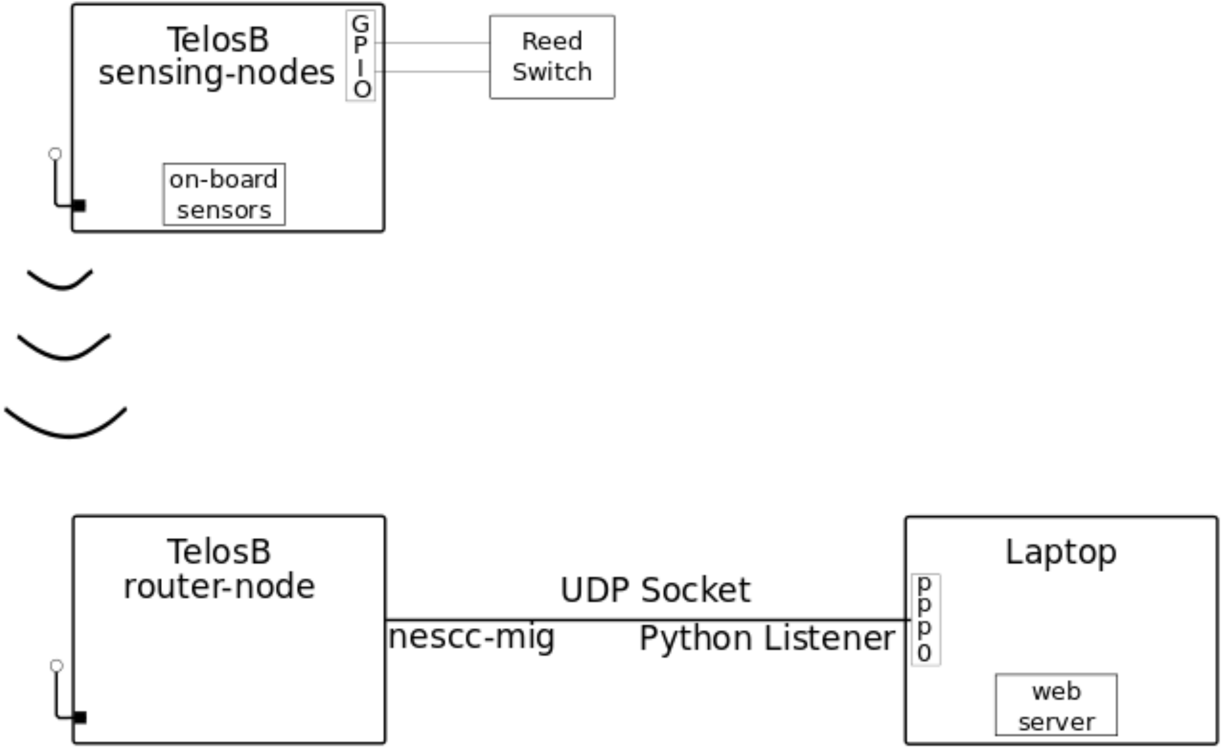
  being placed somewhere in the area

  Sensors: temperature, light, humidity
- 1-2 telosB as intrusion detection nodes:

  being placed wherever somebody could get in (windows, doors)

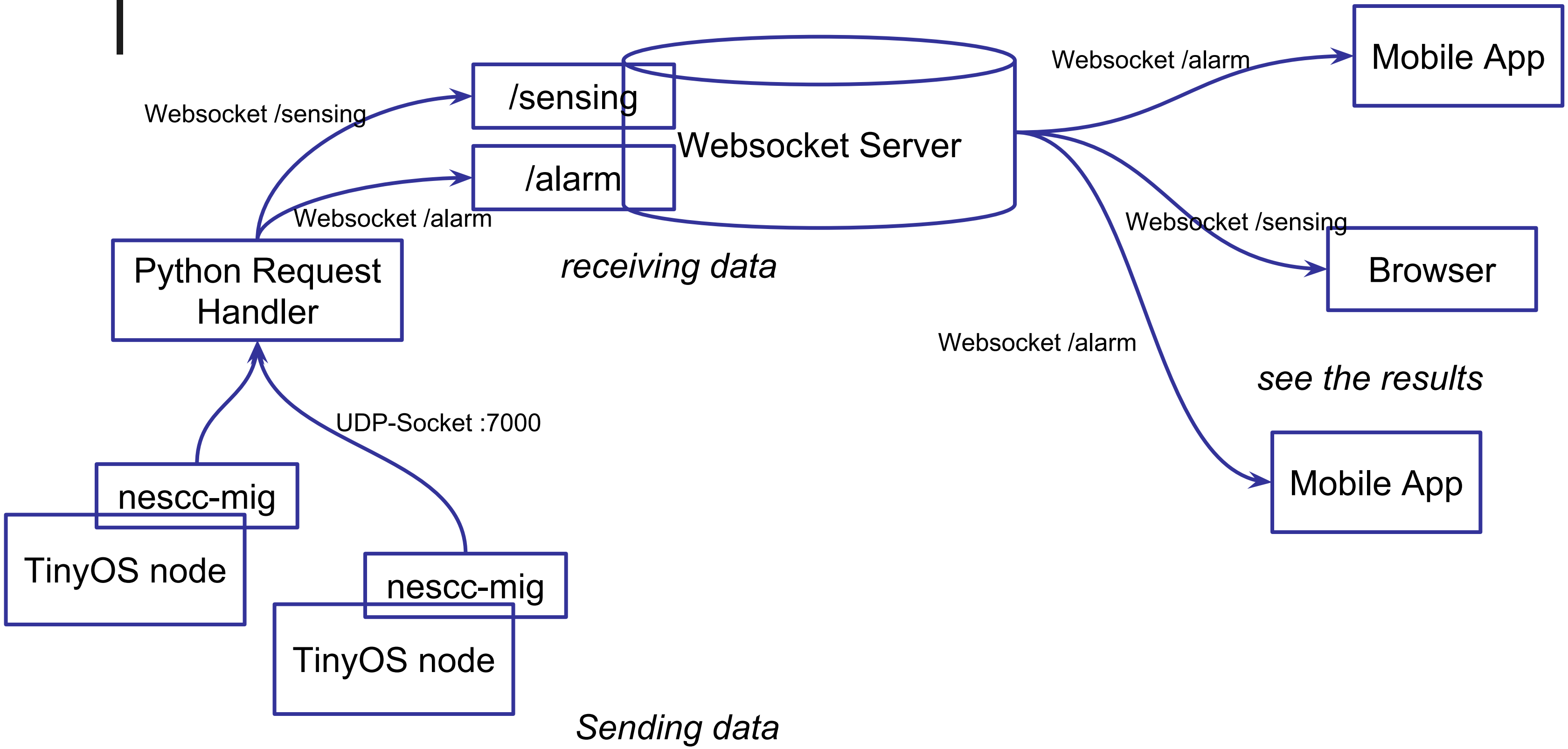
  Sensor: reed switch
- (Python) Webserver

Hardware Setup



Implementation

Module Interaction



Implementation - Websocket Server 1/2

```python
class WSHandlerSensing(tornado.websocket.WebSocketHandler):

    def check_origin(self, origin):
        return True
    def open(self):
        channelSensing.append(self) # this appends a connection instance
        for channel in channelSensing:
            print channel
        self.write_message('Connection established URI: /sensing')
    def on_message(self, message):
        #print 'message received via websocket, uri: "/sensing": %s' % message
        for self in channelSensing:
            self.write_message('Sensing: ' +message)
    def on_close(self):
        channelSensing.remove(self) # this removes an appended connection instance
        print 'connection closed URI: /sensing'
```

Implementation - Websocket Server 2/2

```
http_server = tornado.httpserver.HTTPServer(application)
http_server.listen(8888)
http_server.start()
tornado.ioloop.IOLoop.instance().start()


application = tornado.web.Application([
  (r'/sensing', WSHandlerSensing),
  (r'/alarm', WSHandlerAlarm),
])
```

Implementation - Python Request Handler

```python
try:
    wsSensing = websocket.create_connection('ws://localhost:8888/sensing')
    wsAlarm = websocket.create_connection('ws://localhost:8888/alarm')
except:
    print ('unable to connect')
while True:
    data, addr = rcvSocket.recvfrom(1024) # receive data from node
    if (len(data) > 0):
        # --- forwarding data via websocket to server
        rpt = Sensing.Sensing(data=data, data_length=len(data))
        rpt = str(rpt)
        wsSensing.send(msgSensing(rpt))
        wsAlarm.send(msgAlarm(rpt))
        resultSensing = wsSensing.recv()
        resultAlarm = wsAlarm.recv()
```

Challenges

Challenges

- **reed switch on one node only worked temporary, probably hardware problem**
- **no websocket support in TinyOS**
- **concurrent websockets:**
  - websockets in both directions for change settings in node
  - websocket blocks if no input from user

Live Demo

Thank you for your attention.

Appendix

Git repository:

https://github.com/platz/sensor-lab-project