Galileo Tutorial
Using Python on Galileo
Senzations 2014

Biograd na Moru
1. September 2013

intel Galileo

What will you make?

Alex Gluhak
Intel Labs Europe

Learning goals

Getting started with Python on Galileo

- Running Python interpreter and scripts
- Importing and using modules

I/O basics in Python

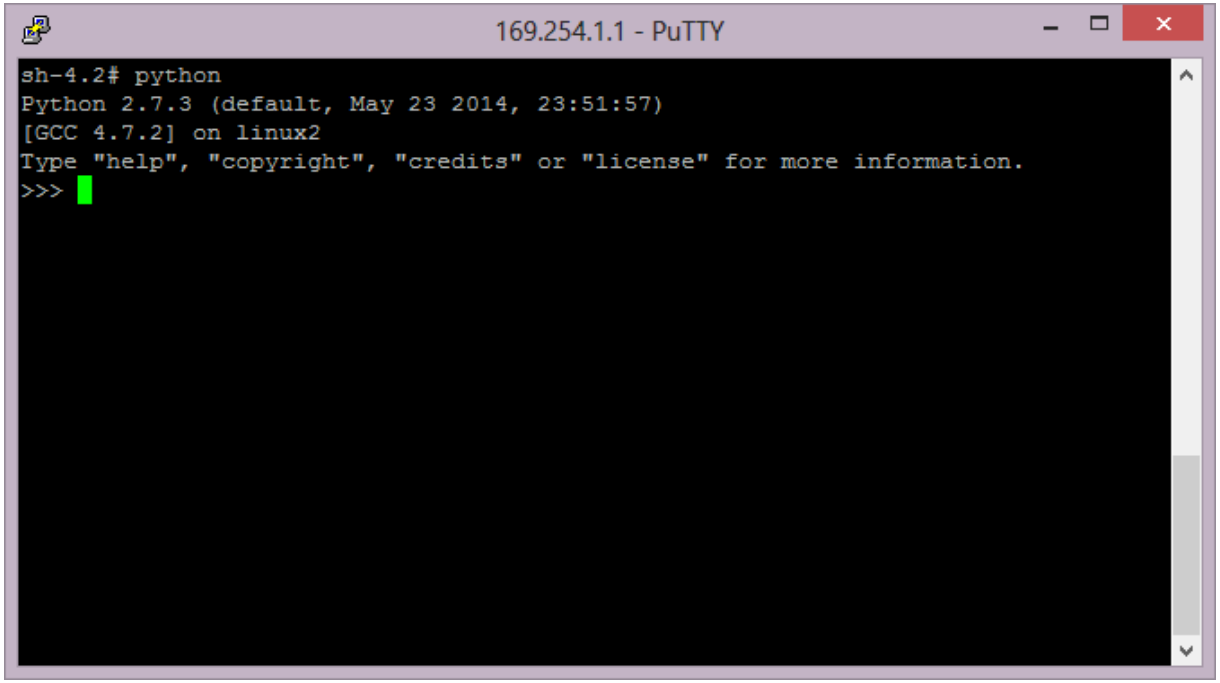
- Serial I/O
- Analog and digital I/O

Network communication using MQTT

Getting started with Python on Galileo

Python interpreter

- Connect to your Galileo via Ethernet
  - Restore sketch for port configuration if it does not work any more
  - Launch: *python or python2*
  - Leave: *Ctrl+D or exit()*



Launching the Python interpreter

- Creating a simple “hello world” script

```
root@clanton:~# vi hello.py     // test internet connectivity

Press “i”                  // insert
Type print “hello world”
Press “ESC”                // exit insert mode
Press “:w”                 // write to file
Press “:q”                 // quit
```

- Launching “Hello world” script

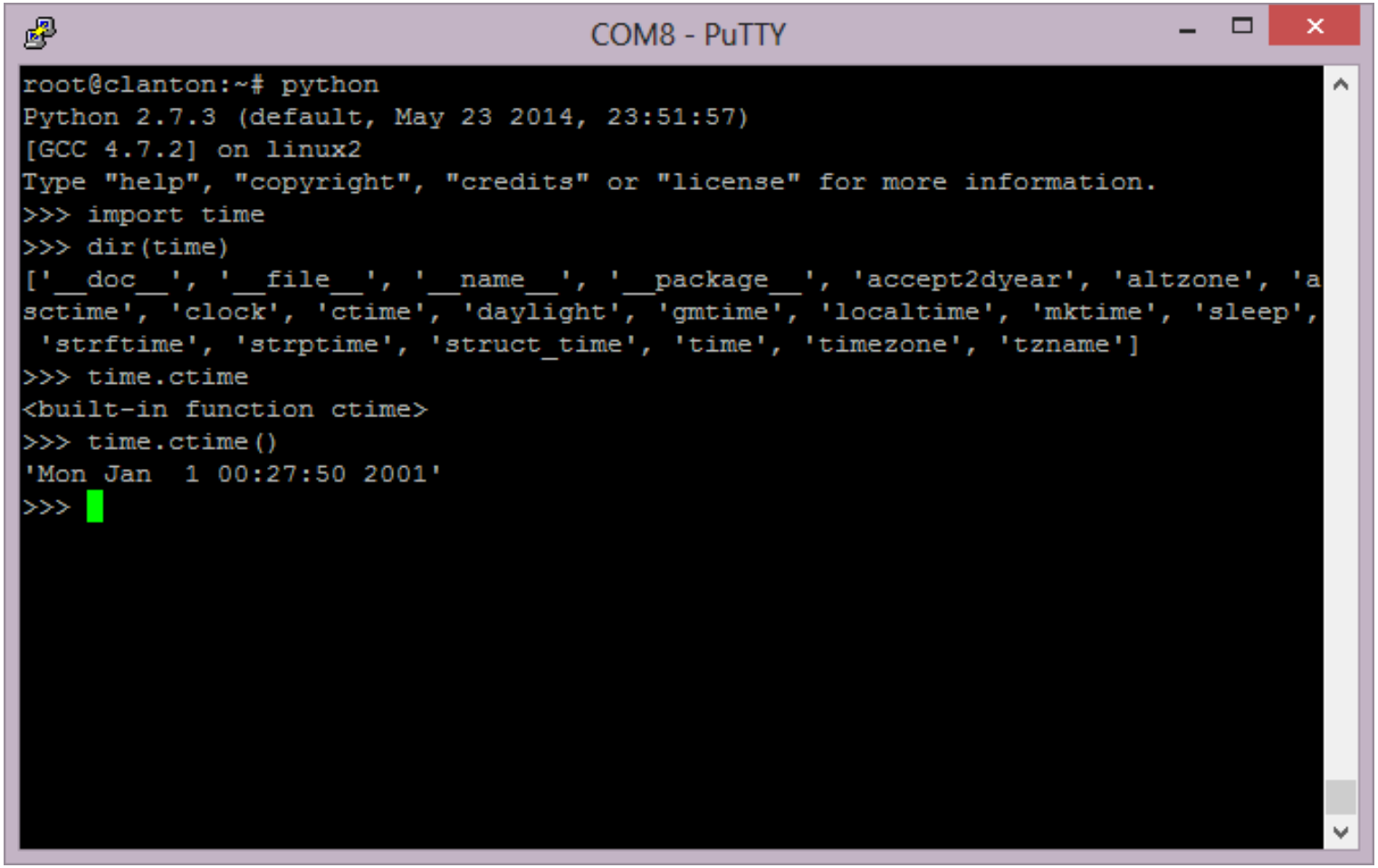
```
root@clanton:~# python hello.py     // “hello world”
```

Python modules

- A module is a file containing Python definitions and statements.
- Modules is a way of sharing code efficiently across different scripts
- Python comes with a library of useful standard modules
  - See library reference: https://docs.python.org/2/library/index.html

- Modules can be imported in other modules and accessed inside
  - Syntax: **import** *module_name*
- Modules can contain functions, variables, etc –
  - Syntax: **dir(***module_name) // list the*
- New modules can be installed from repositories using easy_install
  - Syntax: easy_install *module_name*

Importing and using modules



Exercise: using modules

- Extend your hello world script by printing the current system time after a delay of 5 seconds

Hint: look into the time module for a function that can be used for delay

Solution

Analog and Digital I/O in Python

Digital and analog I/O using python

- Module with Arduino style syntax pre-installed
  - Name: pyGalileo
  - Source: https://github.com/galileo-chofrock/pyGalileo

```
pinMode(pin, direction) // pin number, INPUT/OUTPUT"

digitalWrite (pin, value)  // pin number, HIGH/LOW

digitalRead (pin)  // pin number

analogWrite (pin, value)  // PWM pin number, value from 0..255

analogRead (pin)  // analog PIN number (A0 ...A5)

wait (time) // time in miliseconds
```

Exercise: Digital write

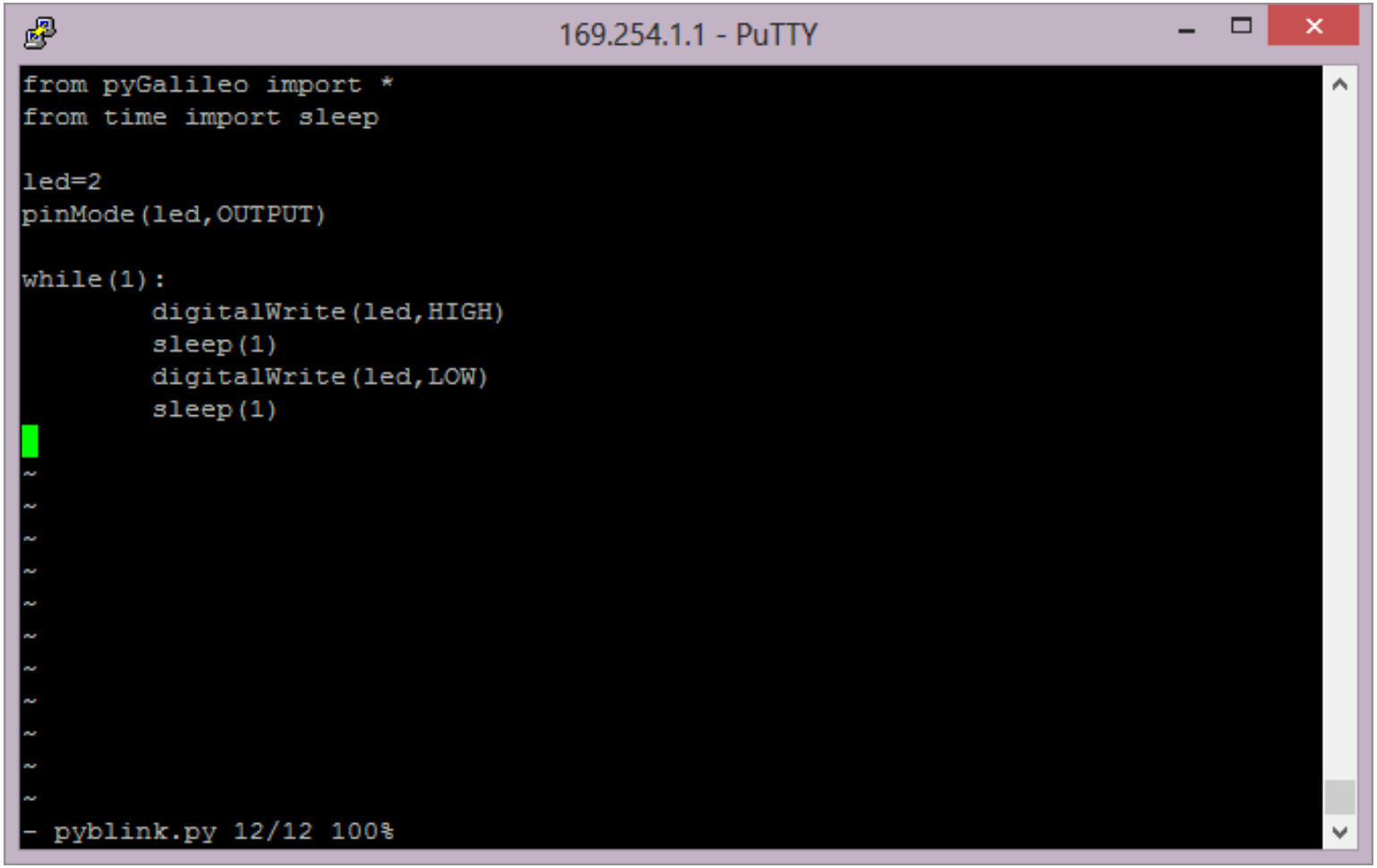
1. Connect Grove LED to D2

2. Write a python script to make it blink every second

Hints:

- If you are not familiar with the vi editor you can use your favourite on the PC and copy the file over via SFTP
- You can use the python interpreter first to experiment with the module

Solution – pyblink.py



Exercise: Analog read

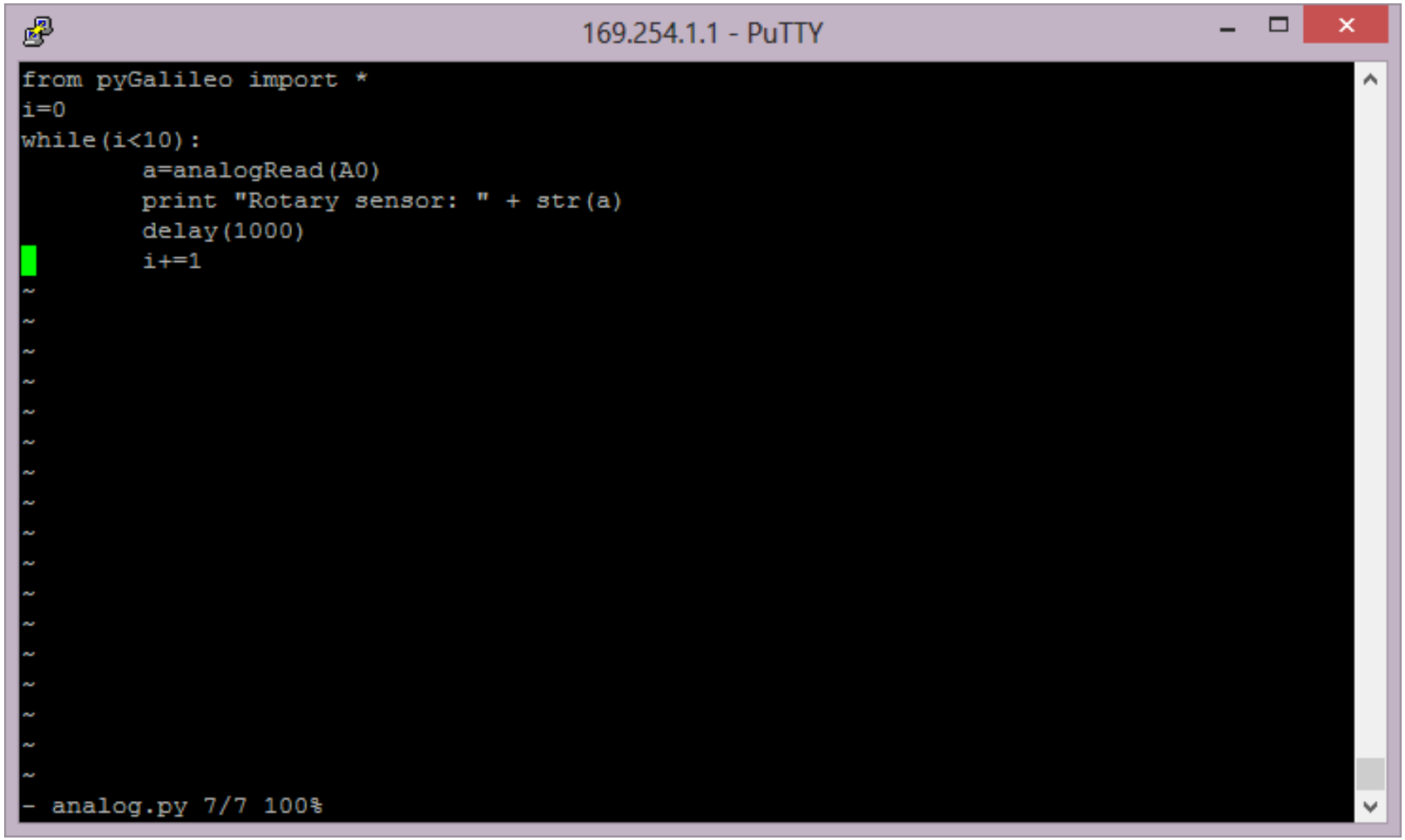
1. Connect Grove Light / Rotary angle sensor to A0

2. Write a python script to print out the reading every second

Hints:

- If you are not familiar with the vi editor you can use your favourite on the PC and copy the file over via SFTP
- You can use the python interpreter first to experiment with the module

Solution – analog.py



Serial communications on Galileo using Python

Serial communications in Python

- Python serial module pre-installed
  - Name: pySerial
  - Use: import serial
  - Source: http://pyserial.sourceforge.net/

- Main principles
  - Open a serial port on the Galileo
  - Perform read/write operations
  - Close port when you are done

- Serial Linux devices on Galileo
  - /dev/ttyS0 – UART0 (Pins: 0/1)
  - /dev/ttyS1- UART1 (3.5 mm jack)
  - /dev/ttyGS0 – serial port to Arduino IDE console

pySerial API overview

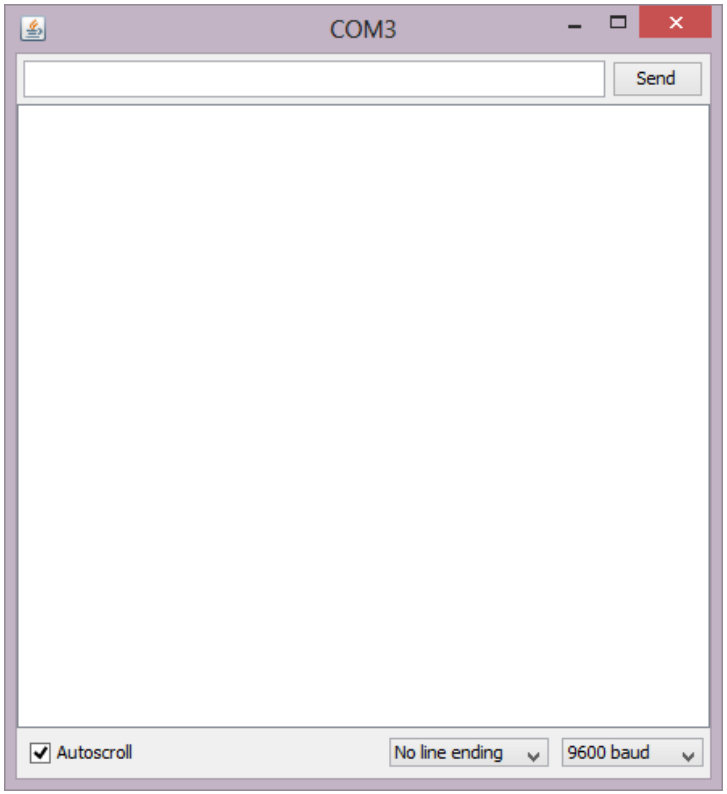
```
Serial (device, baudrate)     // create serial port object and return it

inWaiting()                   // returns number of bytes received

read (number of bytes)  // read number of bytes and return them

write (msg)                   // write msg buffer, e.g a string

close ()                      // close created serial port
```

Detailed API: http://pyserial.sourceforge.net/pyserial_api.html

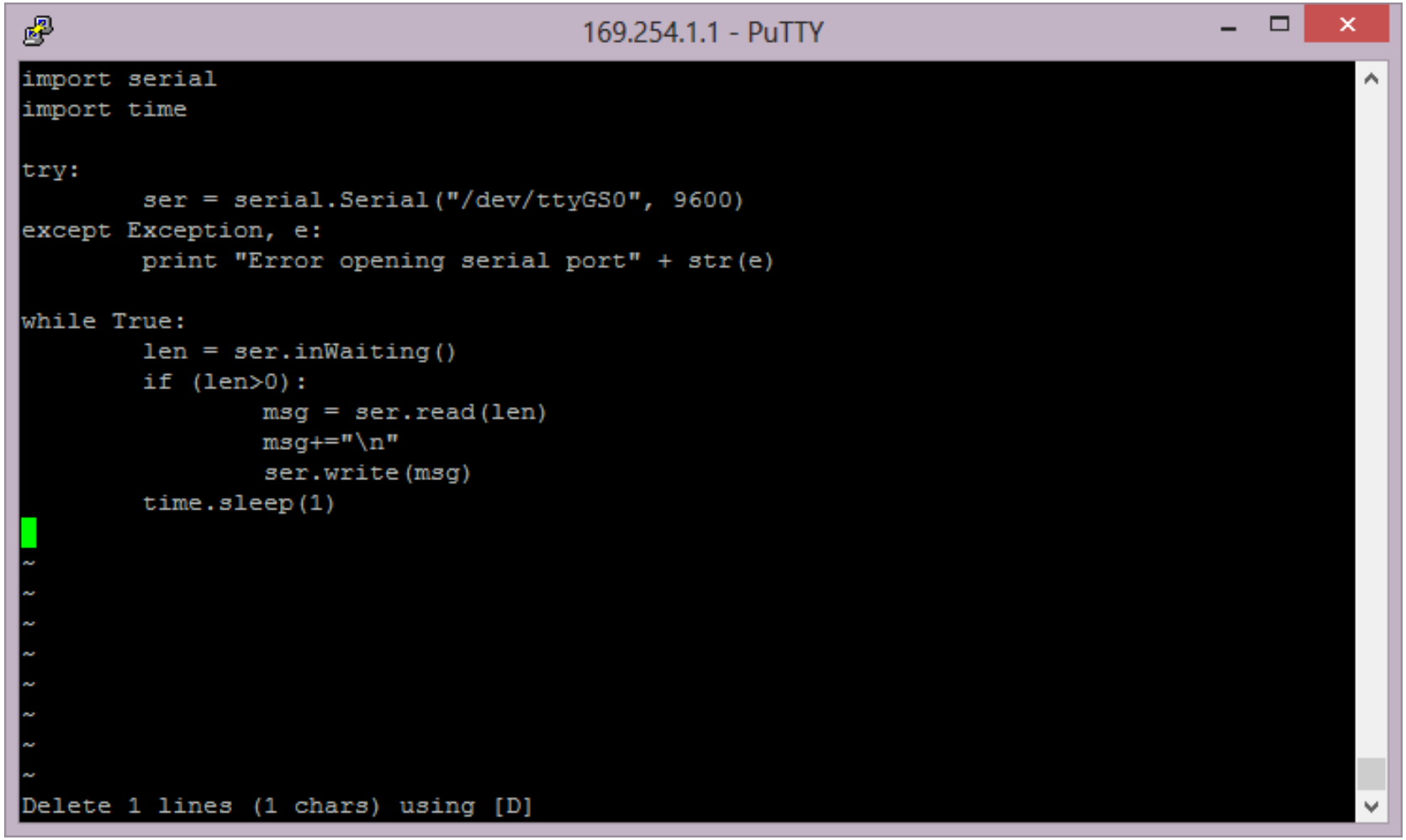
Exercise: Serial echo server

1. Write a basic echo server that connects to the serial monitor of the ArduinoDev UI

2. Send messages from the ArduinoDev UI to the echo server



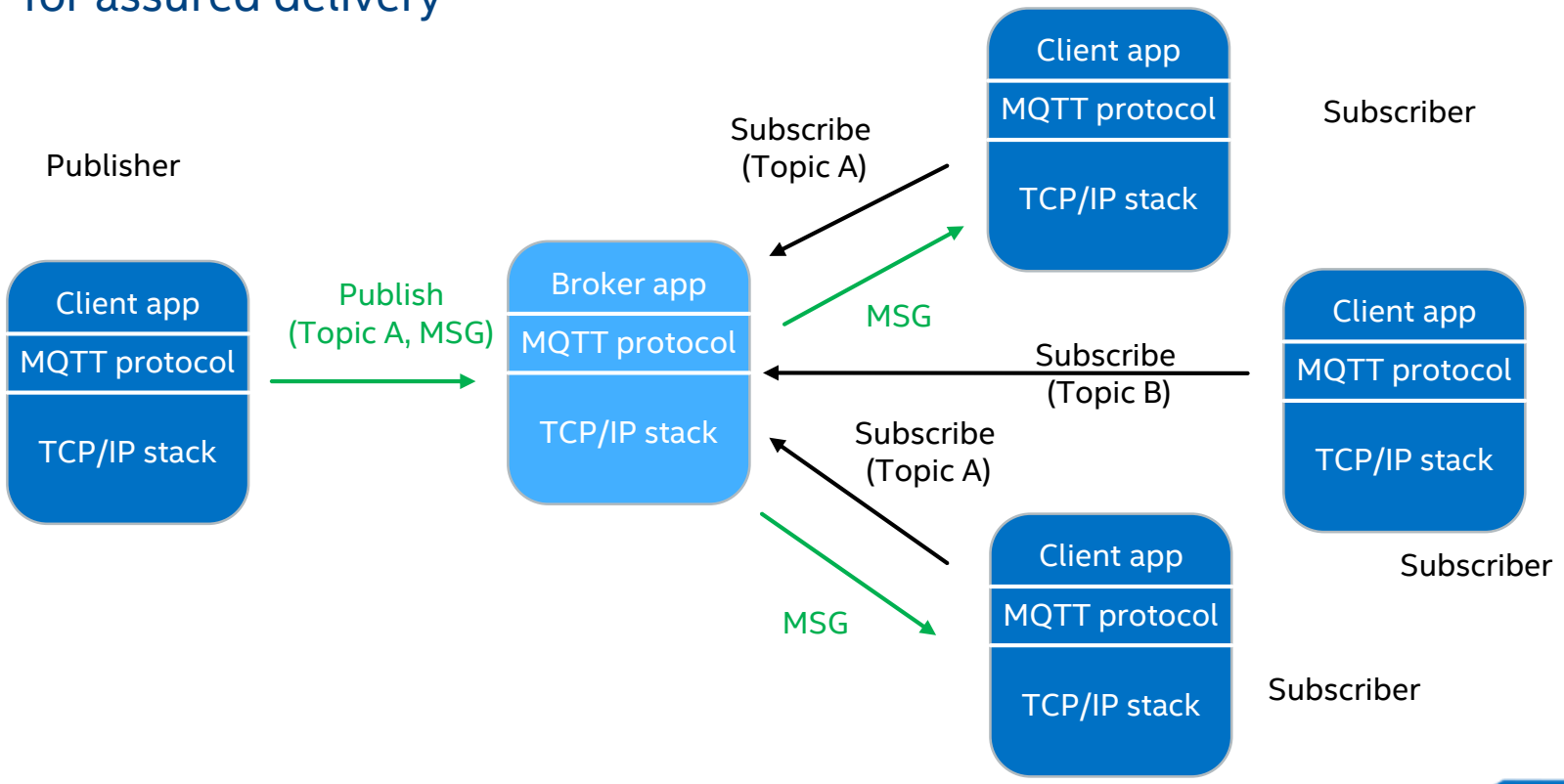
Solution - echoserver.py



Network communications using MQTT

MQTT Overview

- IoT/M2M connectivity protocol
  - Very lightweight publish/subscribe messaging transport
  - Minimizes network bandwidth and device resource requirements whilst and provides some degree of reliability for assured delivery



MQTT protocol basics

- 2 Byte fixed header
  - 14 Message types
  - 3 QoS levels (at most once, at least once, exactly once)

Bit	7	6	5	4	3	2	1	0
Byte 1	MESSAGE TYPE				DUP flag	QOS LEVEL		RETAIN
Byte 2	REMAINING LENGTH							

- Topic
  - Topic name is the key that identifies the information channel to which payload data is published
  - Topic names are UTF encoded and an upper length limit of 32,767 char

- Payload can be of varying length

Topic names

- A topic name is a string of varying length
- To allow subscription to multiple topics at once, a tree hierarchy can be introduced with a topic level separator "/"

"/BuildingA/Floor1/Room1"	(1)
"/BuildingA/Floor1/Room1/Light"	(2)
"/BuildingA/Floor1/Room1/Temperature	(3)

- Wild cards can be used to subscribe to a subset of the hierarchy tree
  - Multilevel wild card "#"
    - "/BuildingA/Floor1/Room1/#" subscribes to topics (2) and (3)
    - "/BuildingA/Floor1/#" subscribes to topics (1)-(3)
  - Single level wild card "+"
    - "/BuildingA/Floor1/+/Light" subscribe to light from all rooms on Floor1

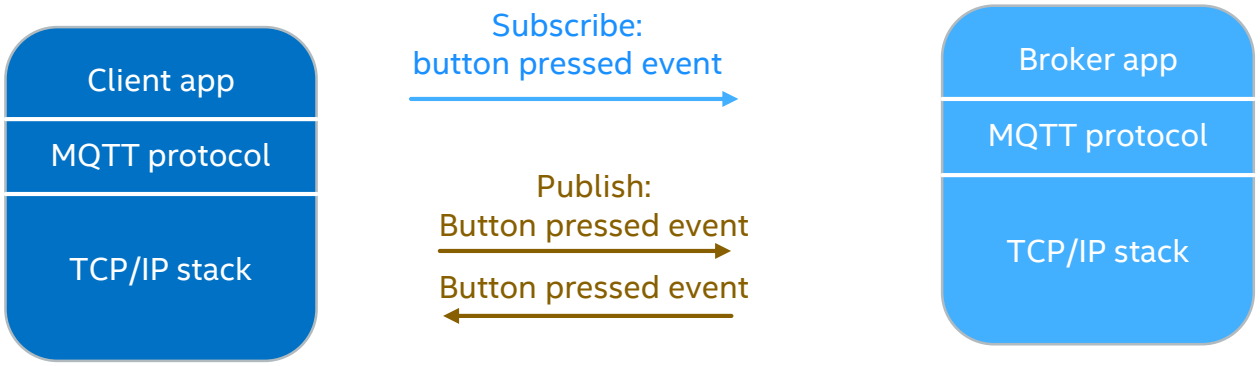
MQTT in Python

- mosquitto library (http://mosquitto.org/)

```
# create a mosquito client instance
mqttc = mosquitto.Mosquitto()

# Connect call backs to functions implementing them
mqttc.on_message = on_message
mqttc.on_connect = on_connect
mqttc.on_publish = on_publish
mqttc.on_subscribe = on_subscribe

# connect to a mosquitto broker
mqttc.connect(broker_address, port)

# listen for messages in a loop
mqttc.loop(1) or mqttc.loop_forever()

# publish a message
mqttc.publish(topic, msgbody)
```

Exercise

1. Look at the mqttGalileo example and run it
   - Code demonstrates both subscriber and publisher part in one client



2. Adapt the code to exchange information or trigger some action with one or more neighbouring Galileos

Resources and further reading

Python: https://www.python.org/

Python2 reference: https://docs.python.org/2/

PySerial libray: http://pyserial.sourceforge.net/

PyGalileo library: https://github.com/galileo-chofrock/pyGalileo

MQTT website: http://mqtt.org/

MQTT Protocol specification v3.1:

http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html

Mosquitto library: http://mosquitto.org/

Paho project: http://www.eclipse.org/paho/


intel