Python for Serial Communication

PyCon APAC 2011, Singapore

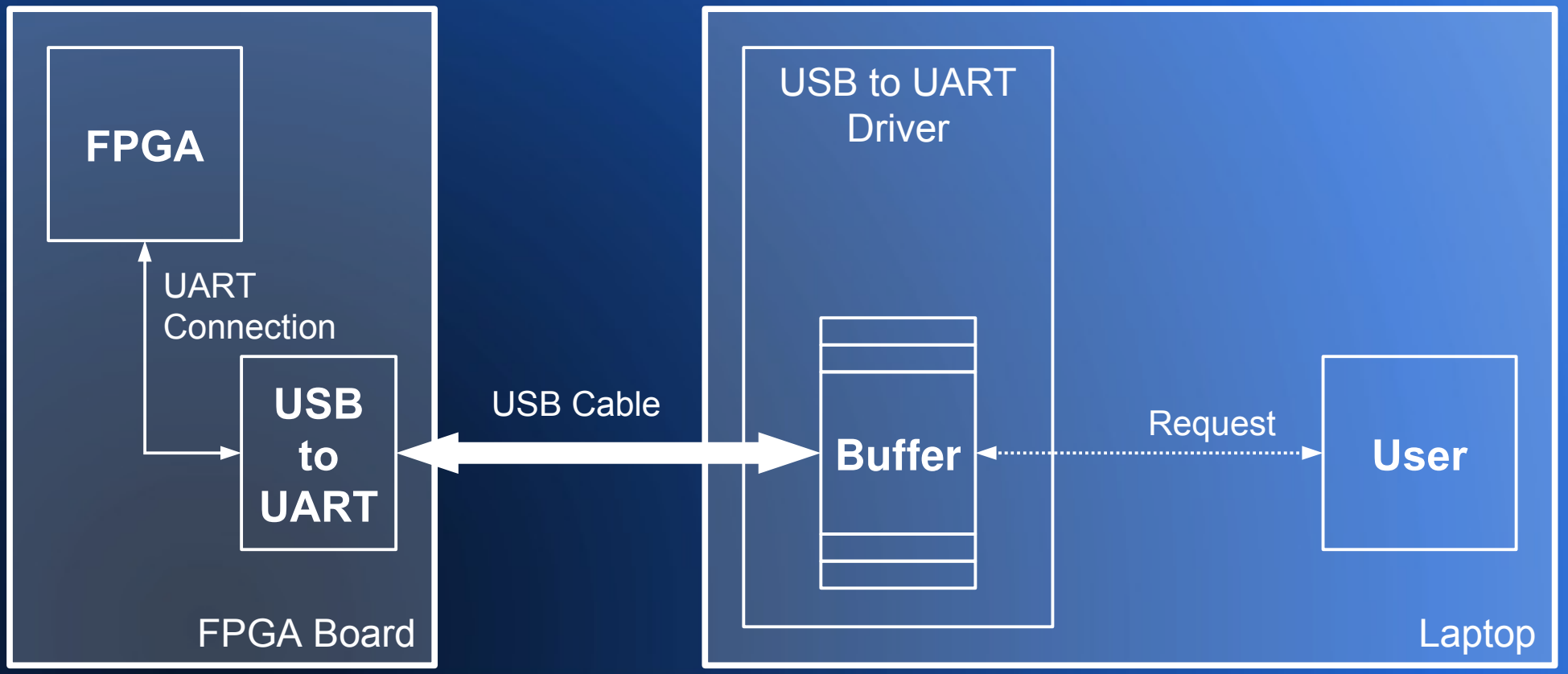
Eka A. Kurniawan
@ekaakurniawan

Outline

- Serial Communication Architecture
- Driver Installation
- pySerial Module
- Demo on Console
- GUI Tool Development
- Demo on GUI

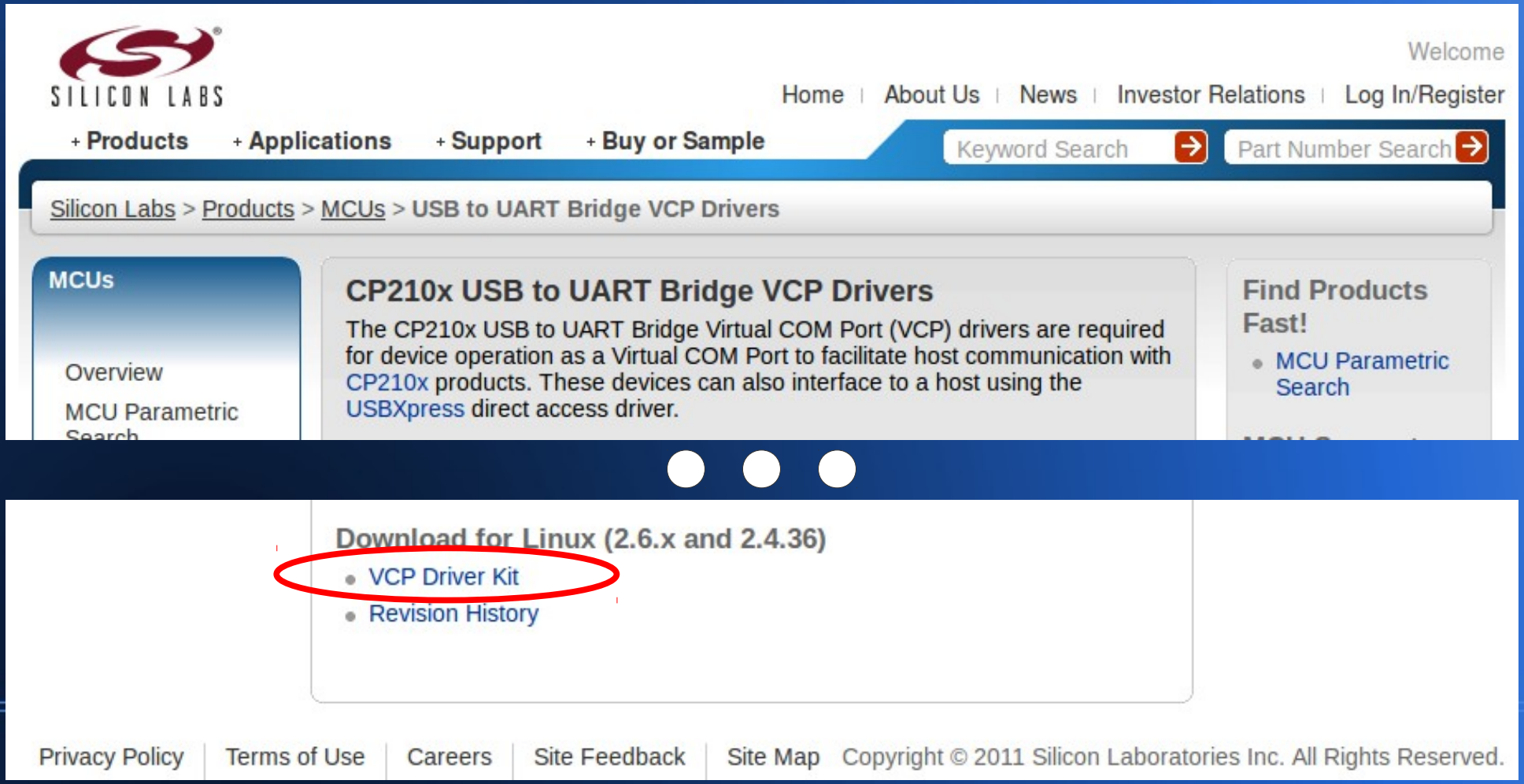
Serial Communication Architecture

- Data Flow Point of View



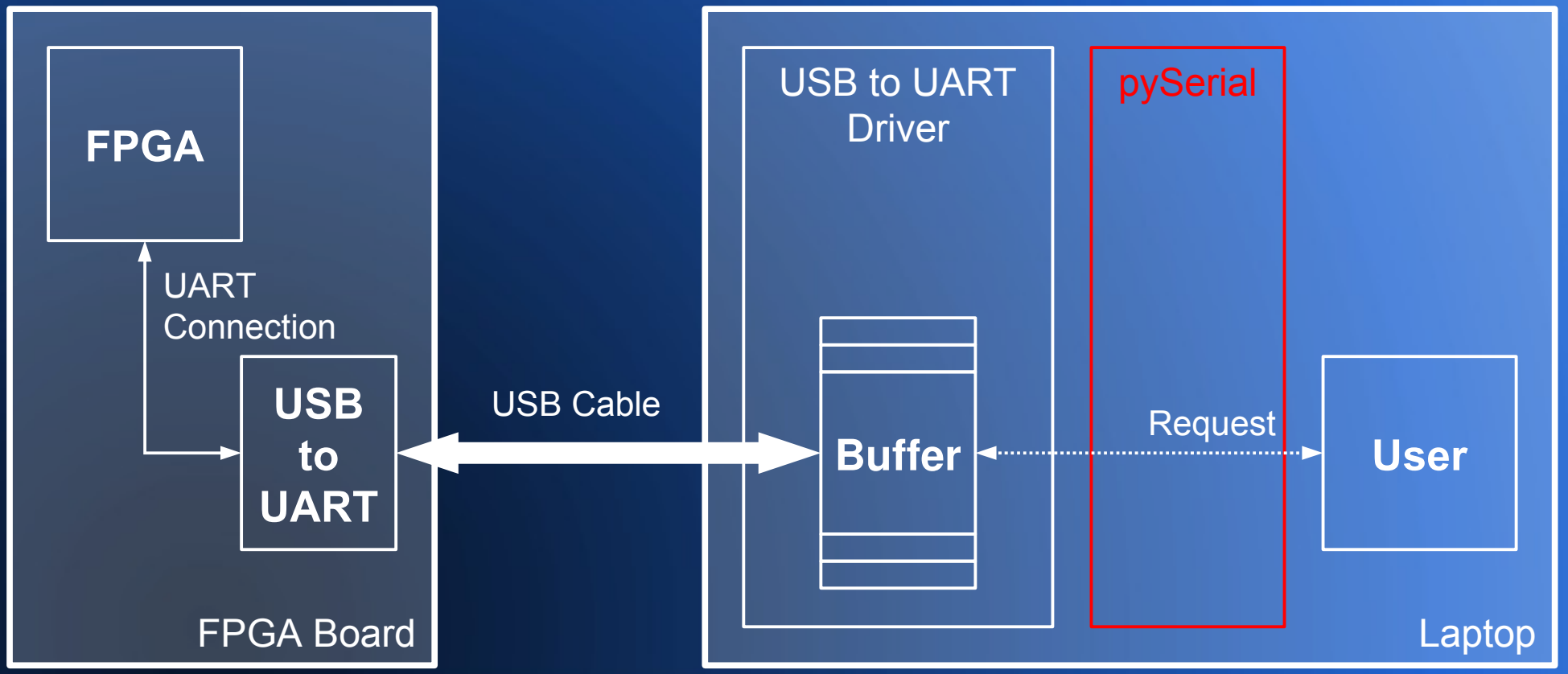
Driver Installation

- Silicon Labs CP2103 USB to UART Bridge VCP Drivers



Serial Communication Architecture with pySerial

- Data Flow Point of View



pySerial Module

pySerial module encapsulates the access for the serial port. It provides backends for Python running on Windows, Linux, BSD (possibly any POSIX compliant system), Jython and IronPython (.NET and Mono). The module named “serial” automatically selects the appropriate backend. - Chris Liechti

Benefits:

- Run on multi-platform
- 100% Python
- Easy to install
- Easy to use

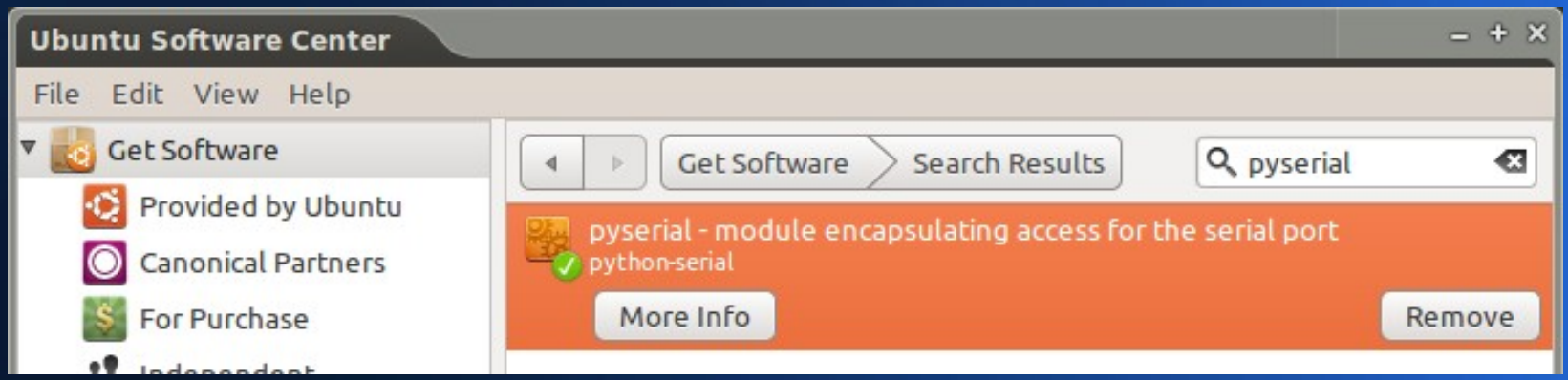
pySerial Module
Installation

- From Source

  Get the archive (pyserial-x.y.tar.gz) from http://pypi.python.org/pypi/pyserial. Unpack it, go to pyserial-x.y directory and run:

  ```
  python setup.py install
  ```

- Ubuntu 10.10



pySerial Module
Basic Functions

- Importing pySerial Module

```
import serial
```

- Serial Class

```
ser = serial.Serial('/dev/ttyUSB0', 9600)
```

- open and isOpen Functions

```
ser.open()
```

```
ser.isOpen()
```

- write Function

```
ser.write('1')
```

pySerial Module
Basic Functions

- inWaiting Function

```
ser.inWaiting()
```

- read Function

```
ser.read(455)
```

```
ser.read(ser.inWaiting())
```

- close Function

```
ser.close()
```

- pySerial API

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

Demo on Console

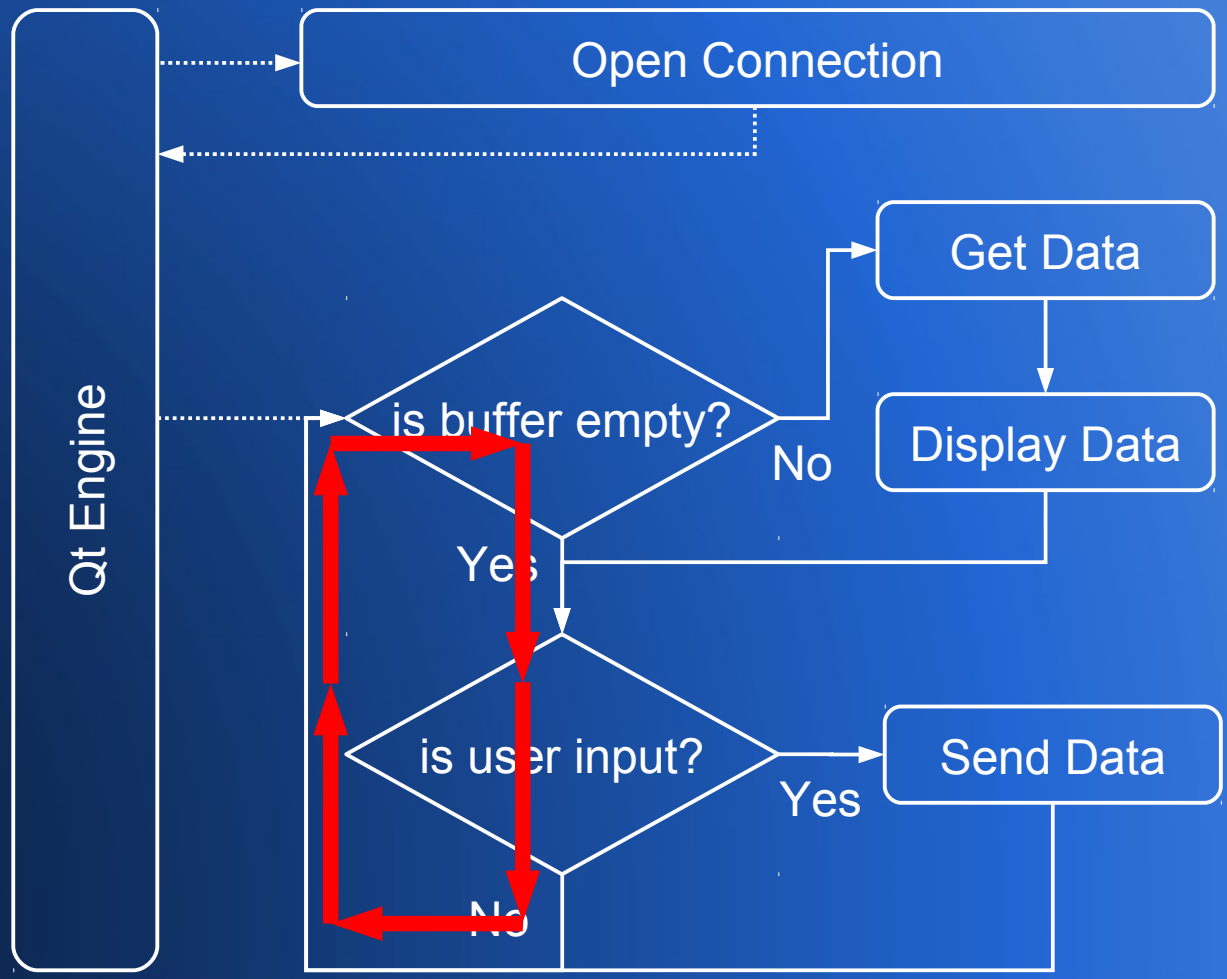
```
eka@eka-MacBookPro:/dev$ python
Python 2.6.6 (r266:84292, Sep 15 2010, 16:22:56)
[GCC 4.4.5] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import serial
>>> ser = serial.Serial('/dev/ttyUSB0', 9600)
>>> ser.isOpen()
True
>>> ser.inWaiting()
455
>>> print ser.read(ser.inWaiting())

************************************************************
************************************************************
**     Xilinx Spartan-6 FPGA SP601 Evaluation Kit     **
************************************************************
************************************************************
Choose Feature to Test:
1: UART Test
...
>>> ser.write('1')
>>> print ser.read(ser.inWaiting())
00000001
SRECORDS at: 0x87120000
...
>>> ser.close()
>>> ser.isOpen()
False
```

GUI Tool Development

- Using PyQt
- SPPyQt Tool
- Tip 1: Checking for New Data
- Tip 2: Detecting New Device

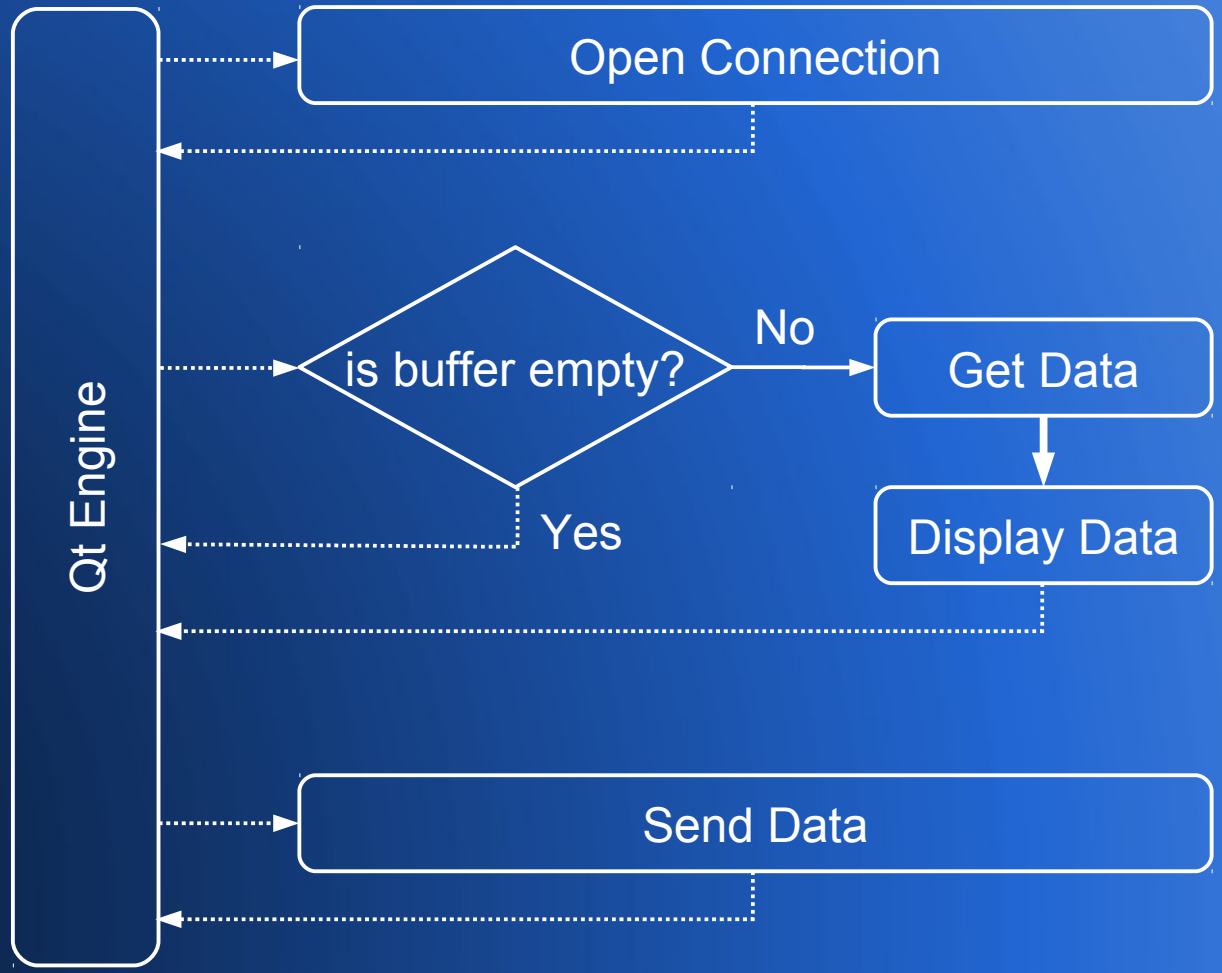
Tip 1: Checking for New Data

- Polling Method
- Advantage: Easy to Develop
- Disadvantages: Resource Inefficiency and Signal Blocking



Tip 1: Checking for New Data

- Timer Method
- Advantage: Resource Efficiency
- Disadvantage: Chance of Triggering Buffer Overflow



Tip 1: Checking for New Data

- Code Implementation for Timer Method Using PyQt
- During __init__

```
self.logTimer = None
```

- Inside connect Function

```
self.logTimer = Qtimer()

QObject.connect(self.logTimer, SIGNAL("timeout()"),
self.checkBuffer)

self.logTimer.start(100)
```

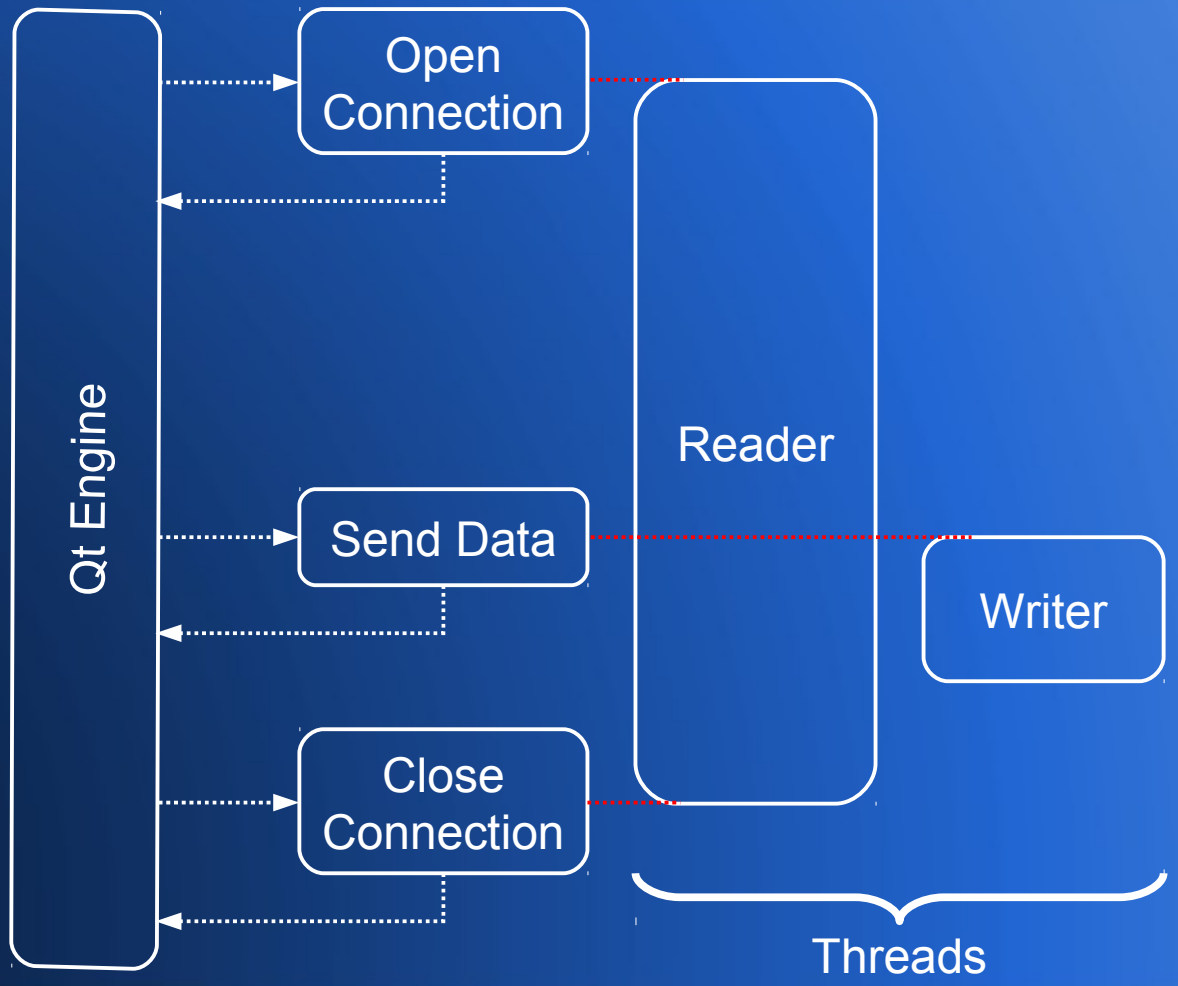
- Inside disconnect Function

```
self.logTimer.stop()
```

Tip 1: Checking for New Data

- Thread Method
- Advantage: No Blocking Signal
- Thread Combined with Read Blocking Provides Resource Efficiency

```
ser.Read(1)
```



Tip 1: Checking for New Data

- Code Implementation for Thread with Read Blocking Method Using PyQt
- Reader Thread Keeps Looping on Following Code

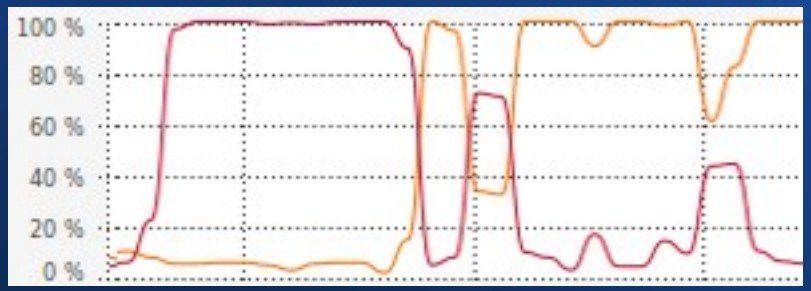
```
data = self.ser.read(1)
n = self.ser.inWaiting()
if n:
    data = data + self.ser.read(n)
self.emit(SIGNAL("newData(QString)"), data)
```

- Writer Thread Executes Following Code

```
self.ser.write(str(self.cmd))
```

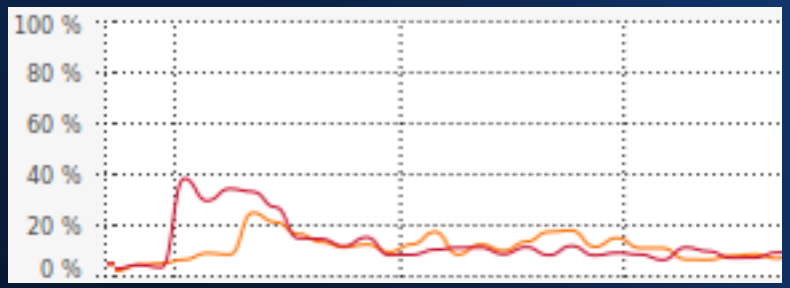
Checking for New Data
CPU Usage

- Timer Method (1ms)

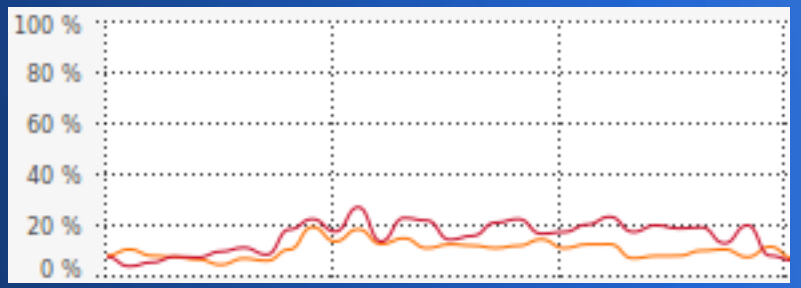


- Thread Polling Method

- Timer Method (10ms)



- Thread with Read Blocking



CPU1 CPU2

Tip 2: Detecting New Device

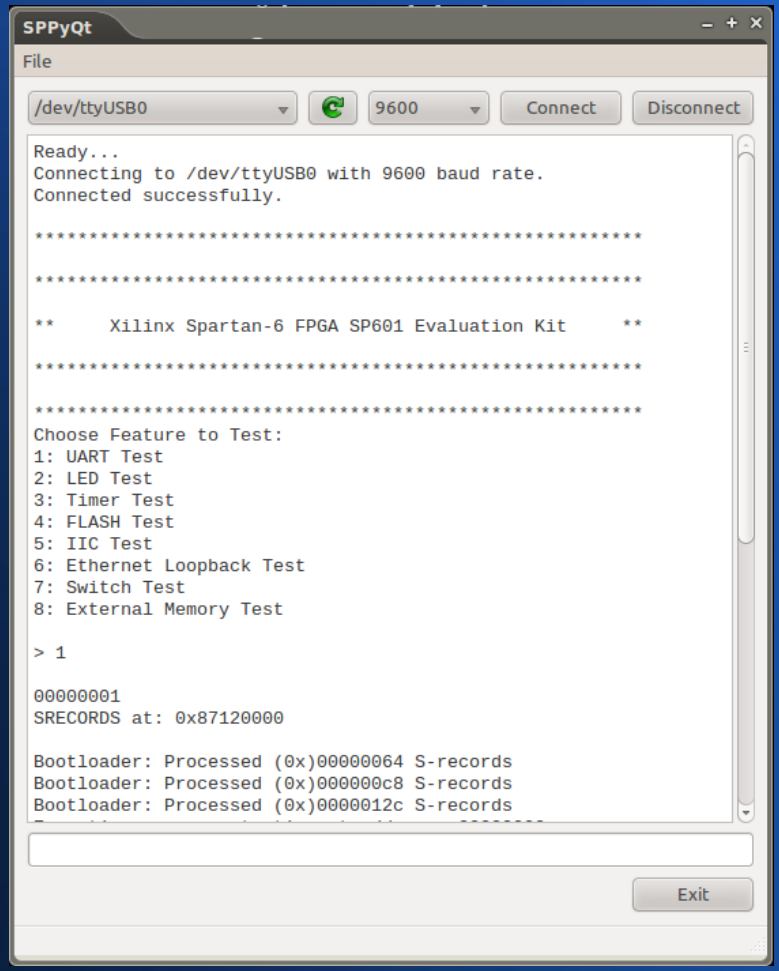
- Detecting New Device in Linux
- Serial Port Communication

  `/dev/ttySx`
- USB-to-Serial Communication

  `/dev/ttyUSBx`
- Code Implementation to Detect New Device Using Python

```
import glob
glob.glob("/dev/ttyS*")
glob.glob("/dev/ttyUSB*")
```

Demo on GUI


SPPyQt
File
/dev/ttyUSB0
9600
Connect
Disconnect
Ready...
Connecting to /dev/ttyUSB0 with 9600 baud rate.
Connected successfully.
****
****
** Xilinx Spartan-6 FPGA SP601 Evaluation Kit **
****
****
Choose Feature to Test:
1: UART Test
2: LED Test
3: Timer Test
4: FLASH Test
5: IIC Test
6: Ethernet Loopback Test
7: Switch Test
8: External Memory Test
> 1
00000001
SRECORDS at: 0x87120000
Bootloader: Processed (0x)00000064 S-records
Bootloader: Processed (0x)000000c8 S-records
Bootloader: Processed (0x)0000012c S-records
Exit

Links

- Silicon Labs CP2103 USB to UART Bridge VCP Drivers
- Downloading pySerial
- pySerial Documentation
- pySerial API
- PyQt Reference Guide
- SPPyQt Tool Project Home

Special Thanks

- Chris Liechti (pySerial Developer)
- Loke Kwan Ng