EuroForth 2015

A Forth Programmer Jumps Into The Python Pit

N.J. Nelson B.Sc. C. Eng. M.I.E.T.
Micross Automation Systems
4-5 Great Western Court
Ross-on-Wye
Herefordshire
HR9 7XP UK
Tel. +44 1989 768080
Email njn@micross.co.uk

Abstract
A unique opportunity arose to compare two similar applications on closely related platforms, one written in Python and one written in Forth.

1. Introduction

In the past two years, the economics of display devices in industrial automation has been transformed by the introduction of very low cost micro-PCs. These can be regarded as circuit boards with an RJ45 Ethernet connector at one end, and an HDMI digital video output at the other end. In our industry, several applications were immediately suggested. Initially, no Forth compiler was then available, so the first application was written using Python. Shortly afterwards, a version of Forth became usable, and therefore a second, similar application was written using Forth, giving an excellent means of comparing the efficiency of constructing new applications in each language.

2. Economics

An industrial display application (excludes screen etc. common to both solutions)
BUYING IN COST (Not sales price)

a) Before Micro-PCs

Industrial fanless PC, including disc & PSU	£475
PC mounting brackets	£23
Operating system (Windows OEM license)	£49
Replication time (approx. 1 engineer-hour)	£45
TOTAL	**£592**

b) Using Micro-PC

Micro-PC	£26
Steel protective enclosure	£11
PSU	£7
SD card	£10
Replication time (approx. 5 engineer-minutes)	£4
TOTAL	**£58**

3. Hardware overview

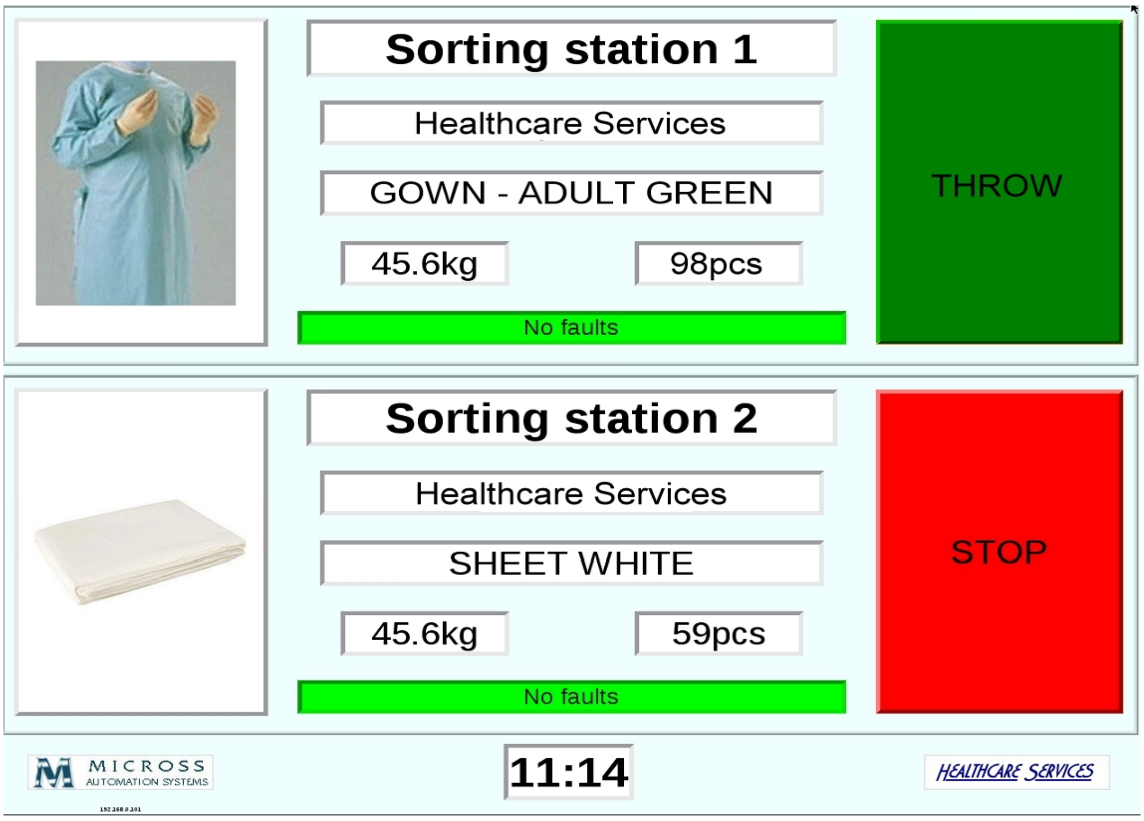
There is a wide variety of micro-PCs now available, all with similar function. By far the best-selling of these devices is known as the "Raspberry Pi". It consists of a circuit card 85mm x 56mm with a highly integrated ARM-based CPU, memory, and a variety of ports. The "disc" consists of a plug-in micro-SD card.

4. Operating System

The recommended operating system for the Raspberry Pi is a version of Linux which is very similar to Debian. However, when used in industrial applications, this has a very serious drawback, in common with many other versions of Linux. In the event of an unplanned power interruption, there is a high chance of corrupting the "disc". If this happens in a conventional implementation, with a keyboard and mouse, there are repair programs that can be run. However, in a standalone display only application, it is not practical to have to repair the disc on a regular basis. To overcome this limitation, we have constructed our own implementation of Linux that makes the SD card "read-only". The systems are now highly reliable and maintenance free. Development work takes place on a standard Linux, and on completion of development, is copied onto the reliable Linux system, which can be switched between read-only and read-write modes.

5. Application overview

Typical applications in our industry are for the purpose of displaying dynamic data to a shop floor operator. The display device is in constant communication with a central control PC, which sends out information to be displayed, typically every 0.5s, using UDP messages over the Ethernet network.

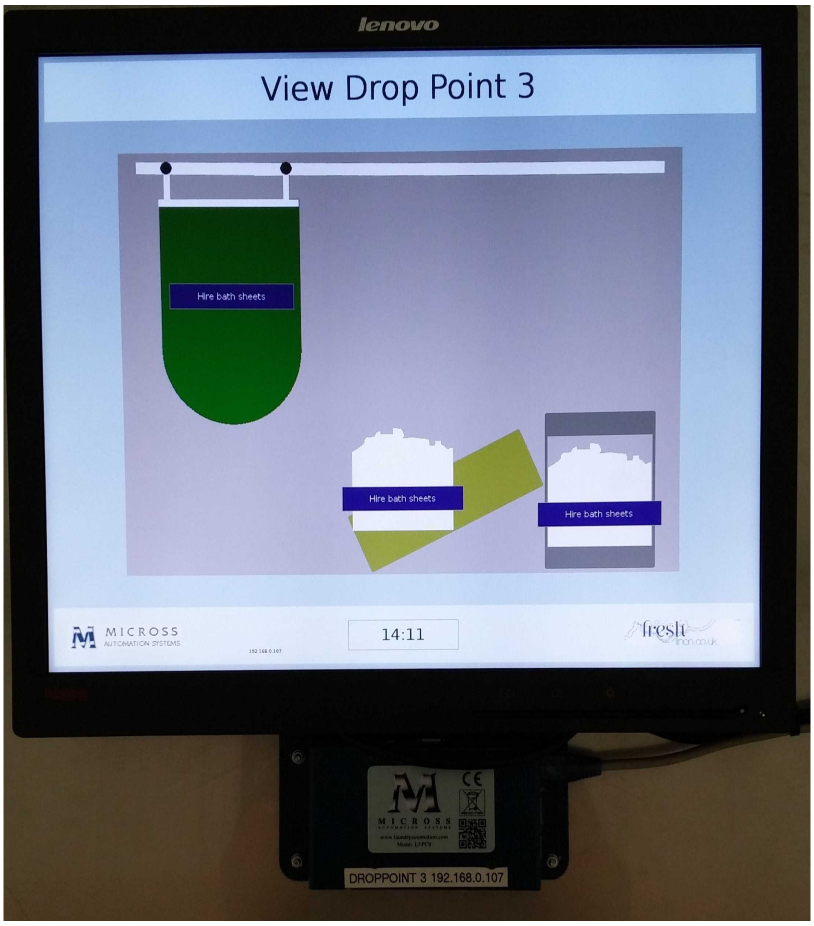


Sorting station 1
Healthcare Services
GOWN - ADULT GREEN
45.6kg
98pcs
No faults
THROW

Sorting station 2
Healthcare Services
SHEET WHITE
45.6kg
59pcs
No faults
STOP

In the example shown on the previous page, the display unit shows the status of two healthcare laundry sorting stations. The customer, category, weight and piece count are shown. The category is illustrated for quick identification. Indicator beacons show the operator clearly when to sort items. This application was implemented using Python.

In the example shown below, the display unit shows loads of towels approaching a towel folder feed station. The operator can clearly see the classification of the towels in the overhead rail bag approaching, on an intermediate belt conveyor, and in the feed bin. The photograph also shows the Micro-PC protective enclosure, just below the screen. This application was implemented using Forth.



6. Program structure

The structure of both programs is exactly the same. On startup, the graphical user interface and the UDP port are initialised. A thread is started to process the communications, and a GUI timer is started to keep the graphics up to date.
The communications thread accepts UDP messages from the central control PC, indicates to the GUI that new data is available, and replies to the central control PC.
The timer compares each item of display data, and updates each part of the display as required. It also displays error information if no new data has been received recently.

7. The Python Experience

Although I have very extensive experience of programming in Forth, and quite wide experience of several other languages, this was my first foray into Python. In addition, I had not worked very much with Linux before. So there was quite a big learning process.

a) Ease of learning
Python is often suggested as a beginner's language, and indeed it is possible to write elementary programs after only a few hours study. However, some of the concepts of the language are extremely subtle, and in order to produce a serious and reliable commercial program, a long period of study is required. It is very unfortunate that there is currently no good book available which covers the latest version of the language. The online documentation is complete and well organised but can be somewhat terse.

8. The Forth Experience

Being already very familiar with Forth, I had only to master a new version, and the interface with Linux.

a) Ease of learning
Like Python it is possible to write elementary programs after only a few hours study. However, equally, in order to produce a serious and reliable commercial program, a long period of study is required. In this case, a good and up to date book is readily available.

b) Dialects and variations
A Forth programmer will be shocked to learn that the Python language is highly regulated. There is essentially only one version of the language that is current at any time. Language development follows a formal process, and the originator of the language is regarded as a "Benevolent Dictator For Life" and is assumed to have a power of veto.

c) Programming paradigm
Although Python bills itself as having multiple programming paradigms, in practice you are compelled to use an object-oriented model, because all of the really useful library functions assume this.

d) Standard libraries
The best recommendation for the Python language is the very comprehensive standard library support, covering almost every eventuality. Python bills itself as "batteries included". In practice, there are some important libraries that have not been updated to conform to the last major release, which came out in 2008.

e) Graphical user interface
Python strongly encourages the use of the TCL/Tkinter GUI. This is extremely unfortunate because it lacks the flexibility of GUIs that are more regularly maintained.

b) Dialects and variations
A Python programmer will be shocked to learn that the Forth language is unregulated to the point of anarchy. There are as many versions of the language as there are programmers. Any "Benevolent Dictator" would be instantly overthrown. From here downwards, this paper will describe the VFX Forth for Linux by MPE.

c) Programming paradigm
Forth is its own paradigm.

d) Standard libraries
VFX Forth comes with a fair range of extensions covering many frequently needed functions. It is in the nature of Forth that these are regarded as suggestions only, and are frequently modified to suit a particular application.

e) Graphical user interface
VFX Forth provides elementary wrappers for the GTK+ GUI. Unfortunately the wrappers support only the older principal version 2 of GTK+, rather than the current principal version 3. In Forth, it is not difficult to create upgraded wrappers using the older code as a model. GTK+ is highly complete, very flexible, regularly updated, and far superior in every respect to the Tkinter GUI preferred by Python.

f) Style and compactness
Python is the only language I have come across that approaches the compactness of Forth. In my view this is a very important feature of any language because it enables complete functions to be read in one glance, which greatly assists in bug detection. But, this compactness is achieved by the use of indentation to delimit blocks. This prevents free formatting which we use regularly to improve code readability. In addition, multiple statements per line are discouraged.

g) Readability
All languages can be used to produce more, or less, readable code. But Python programs when well written are definitely easier to read than most other languages.

h) Community support
Python has a very large user base, and as one might expect there is a variety of community forums. In practice these tend to be clogged with elementary queries from beginner programmers, and it can be hard to get good advice on the very subtle difficulties of the language.

f) Style and compactness
Forth is still the most compact language to code, primarily due to its low structuring overhead and concatenative programming model.

g) Readability
In a language that contains such "bad" key words as -ROT and PICK, it is of course easy to write obfuscated code in Forth. However, with careful naming, use of Forth's completely free formatting, and careful structuring, Forth code can still be the best language for readability and maintainability.

h) Community support
Forth has a rather small active base of practising commercial programmers, and therefore the chance of finding anyone else working in the same dialect and in a similar application area is slight. On the other hand, there is the opportunity (possibly after liquid bribery) to consult the actual author of the compiler.

i) Difficulties
In addition to the problems with the obsolete default GUI, and the out of date libraries, we encountered the following problems with Python.

i. Scope of variables
There is an assumption in Python that all variables are as local as possible. Python is extremely averse to global variables, and for anyone used to the opposite assumption, this leads to frequent misunderstandings. In fact the whole subject of scoping of variables is so tricky in Python that it fills the community forums with obscure problems.

ii. Structures
The whole approach to structures in Python is completely different from that in C or Forth. The Python approach is really quite clever, but it is also quite complex and hard for a beginner to grasp. This makes it very difficult when writing a communications protocol, in which the data is normally defined as a C-like structure.

iii. Garbage collection
This is the biggest weakness of Python, and one which occupies reams of forum discussion about the difficulty of debugging. Extreme coding care is needed to avoid either memory leaks (caused by uncollected garbage) or miraculously dereferenced variables (due to over-zealous binmen).

i) Difficulties
In addition to the problems with the obsolete GUI version, we encountered the following problems with VFX Forth for Linux.

i. No initial support for floating point
Unfortunately this meant that for the first application, it was not possible to use Cairo for drawing. The floating point is now working and will be used on the next application.

ii. Difficulties with cache flushing in Linux
This means that perfectly correct code will sometimes fail to compile. It is a temporary irritation only, as a second (sometimes third) compilation attempt will succeed.

9. Equivalent codes examples

The communications thread, with almost identical function in both applications:

a) Python

```
def coms(app):
    pcsock=socket_init()
    while apprunning:
        if socket_ready(pcsock):
            rbytes, raddress = pcsock.recvfrom(2000)
            app.mq.put_nowait(rbytes)
            sbytes = struct.pack('B', 0)
            pcsock.sendto(sbytes, raddress)
        time.sleep(0.05)
```

b) Forth

```
: COMACTION ( --- ) \ Communications task action
  SOCKET-INIT                         \ Initialise socket
  NEWFLAG OFF                         \ Clear new data available
  BEGIN
    SOCKET-READY IF                   \ Message ready
      SOCKET-GETPACKET LFPC4TX1 = IF  \ Correct message length
        SOCKET-PROCESS                \ Process received packet
        SOCKET-TRANSMIT               \ Reply
      THEN
    THEN
    50 ms PAUSE
  AGAIN
;
```

9. Conclusion

Each language has both advantages and disadvantages. Full mastery of either language could only be achieved by constant practice. Since all our principal applications are written in Forth, we will continue to use it for future applications in Linux on Micro-PCs.

NJN
September 2015