Embedded made easy: piCore

Mark Olson
Solon Technology Consulting Services
2019-01-22

Agenda

1. Embedded is hard
2. Conventional approaches
3. piCore on Raspberry Pi
4. Introduction to piCore
5. Getting started
6. What I have learned

Embedded systems are hard

- Filthy conditions
- Arbitrary input
- Crappy power
- Unreliable connectivity
- No exception handling
- Challenging users

Embedded systems are hard

- Make it small!
- Make the screen big!
- Make it light!
- Make the battery last forever!
- Make it do complex things!
- Make it easy to use!

Typical approach to embedded

- Microcontroller and custom PCB
- Some oddball proprietary environment
- Advantages
  - robust
  - low power
  - cheap in large quantities

Typical approach to embedded

- Challenges
  - limited language and platform options
  - specialized, often proprietary, coding tools
  - limited library availability
  - high start up cost

Make an app!

- Apps are sexy
- All the infrastructure is provided
  - hardware
  - architecture
  - distribution
  - connectivity
  - revenue collection

Make an app!

- Problems:
  - Robustness
  - No tactile input
  - Your data is trapped in their ecosystem
  - Constantly chasing someone elses architecture
  - Retrieve your $1700 iPhone from a fresh cow pat. Then come back and convince me this is a good idea.

Arduino

- Better option for
  - realtime
  - extensive physical interaction
- Problem
  - low level coding in C
- Rapidly getting more capable
- MicroPython offers an interesting option, but it is early days

piCore on Raspberry Pi

- Open hardware and software
- Range of readily available boards
  - single core, gumstick size, $10
  - quad core, credit card size, $45
- Robust and stable
- Development with existing host based tools

piCore on Raspberry Pi

- Access to the open source community and libraries
- Familiar platform and tool sets
- Piggyback on established certifications
- Low start up cost
- Low cost in small volumes

Why not piCore on Raspberry Pi?

- hard real time
- instant start up
- analog inputs/outputs

Introduction to piCore

From the README:

Tiny Core Linux is not a traditional distribution but a toolkit to create your own customized system. It offers not only flexibility and a small footprint but a very recent kernel and a set of applications, making it ideal for custom systems, appliances, as learning Linux, especially on the Raspberry Pi.

Introduction to piCore

- A port of Tiny Core Linux to the Raspberry Pi.
  - Tiny Core x86 based
- Same basic philosophy and architecture
- Much of the documentation and on line guidance is applicable
- **Caution:** Many packages available on Tiny Core have not been, and may never be, ported to piCore.

piCore architecture

- / is a RAM drive
- executables are added as read-only loopback mounts
  - resilient to power loss
- stripped down
  - documentation optional
  - busybox
  - simple script based init
- base environment, boot to console, is 36M
- whole operating environment for flokk is 41.7M storage, 128M RAM.

piCore 9.0

- Current version is 9.0.3
  - 4.9.22 kernel
  - glibc 2.25
  - busybox 1.26.2
  - 1300 extensions

piCore 9.0

- development tools
  - gcc 7.2.0
  - MicroPython 1.91-31
  - perl 5.24.0
  - Python 3.5.2, 3.6.0
- X Window available
  - FLWM, XFCE
- Warning: will not boot on a Pi 3 Model A+ or Pi 3 Model B+

PiCore 10

- In beta
  - 4.19 kernel
  - glibc 2.28
  - gcc 8.2.0
  - busybox 1.29.3
  - many extensions missing
- Tiny core (X86) has just gone to beta 1; piCore progress should follow shortly

Raspberry Pi Zero W

Exactly what I needed:

- 512M ram
- bluetooth
- WiFi
- USB is a USB OTG port
  - can be placed into device mode
- UART
- Python

Getting started

- Get a Raspberry Pi
  - 3 Model B, Zero W, Zero, 2 Model B
- Download piCore
  - http://tinycorelinux.net/9.x/armv6/releases/Rpi/
- Unzip
- Copy image to micro SD card
  - dd

Getting started

- Repartition
  - cfdisk
  - Easier to do with linux host if you got one
  - Include a swap partition
- Put into Pi
- Boot
- Use tce-ab to load applications
- filetool.sh -b

A simple embedded device

- CKUA audio player
  - Get 9.03 booted
  - Run tce-ab
    - Install alsa, cmus, and screen extensions
  - Add to /opt/bootlocal.sh:

```
screen -d -m cmus
sleep 5
cmus-remote -l "http://ckua.streamon.fm:8000/CKUA-64k-m.mp3"
sleep 5
cmus-remote -p
```

  - filetool.sh -b

flokk

- piCore 8.1.5
- 5300 lines of Python 3.5
- Functionality:
  - Asynchronous, threaded architecture
  - Excel XML based workbook manipulation
  - Online and at hand verified software updates
  - USB storage device emulation
  - .ini configuration file
  - WiFi credential and connection manager
- $1k in hardware, 300 hours effort

What I have learned

- Platform is stable and reliable
- Python libraries avoid a lot of the grunt work
- Plentiful online resources:
  - documentation
  - code samples
  - problem resolution

What I have learned

- Make “filetool.sh -b” your mantra
- Easy to add a partition for storage, but be careful not to undo failure resiliency
- Automated python module installation (pip) does not work
  - download and build your own tcz

What I have learned

- Building .tcz files is easy
  - mirror directory tree
  - copy files
  - mksquashfs
- Not all kernel modules are included
  - most are available

What I have learned

- Setting time zone is a pain
  - edit /mnt/mmcblk0p1/cmdline.txt
  - add “tz=MST+7MDT,M3.2.0/2,M11.1.0/2”
- If you can't install extensions, you have run out of loopback devices
  - edit /mnt/mmcblk0p1/cmdline.txt
  - add “max_loop=XX”

What I have learned

- Gadget mode is not for sissies
- Buying Pi Zero boards individually
  - Can buy low cost packages at canakit.ca
  - Can buy individual boards in bulk directly from Raspberry Pi foundation
    - suzie@raspberrypi.org

Solon Technology Consulting Services

- Development
- Appplication management
- Project management
- Audio visual
- Contact
  - cuug.ab.ca/~olsonm
  - olsonm@cciwireless.ca

SOLON
Technology Consulting Services