Harnessing Embedded Linux and Python for Stand-Alone MRI Applications

Pascal P Stang[1,2] and Greig Scott[2]

[1]Procyon Engineering, San Jose, CA, United States, [2]Electrical Engineering, Stanford University, Stanford, CA, United States

Introduction

Advances in RF electronics, high-speed data converters, and multi-core processors have long fueled high-end MRI techniques such as parallel imaging and real-time scanning. Yet these same technology advances can also be leveraged to benefit small-scale MR. Indeed there is substantial interest in applications including bench-top scanners for education, chemical spectroscopy, and relaxometery, portable MR "mouse" systems, RF ablation control, and interventional device safety monitoring [1-7]. We present a compact stand-alone MRI console powered by embedded Linux and programmed in Python to investigate the potential of such a platform to deliver modern performance and versatility for NMR/MRI applications constrained in size, power, cost, or user interface.

Methods

Hardware: Our compact stand-alone MRI console (Fig 1,2) is built around a BeagleBoneBlack commercially-available embedded computer platform [8]. The BeagleBone is comprised of a TI AM3359 1GHz ARM Cortex-A8 System-on-Chip (SoC), 512MB DDR3 RAM, and 2GB eMMC Flash for persistent storage. It also includes substantial I/O resources for hardware interfacing including Ethernet, microSD slot, HDMI video output, and numerous UART, SPI, I2C, and general-purpose I/O interfaces. Medusa RF Tx/Rx & Gradient modules [9] link directly with the Beaglebone General-Purpose Memory Controller (GPMC) and provide the interfaces necessary for performing MR imaging. A CircuitCo 7-inch 800x480 touchscreen LCD serves as a display and user interface. Additional components needed for specific applications can be controlled by, and integrated into, the package.

Software: We configured the BeagleBone to run a reduced version of the Ubuntu Linux OS. For flexibility, extensibility, and ease of development, all high-level programming including the graphical user interface, pulse sequence descriptions, and image reconstructions are written entirely in the Python language. The Python *PyQt* and *pqqtgraph* packages are used to build a versatile and fast user interface, while the *numpy* and *scipy* libraries provide computation tools needed for sequence development and reconstruction akin to those found in Mathworks Matlab. The RF and gradient hardware features are exposed to Python via simple low-level drivers written in C++.

Results

We have implemented gradient echo and spin echo pulse sequences (Fig 3), as well as supporting prescan tools. The RF subsystem covers 0-3T proton at excite/receive bandwidths up to 500ksps, and gradient outputs runs up to 250ksps. The BeagleBone GPMC transports RF/gradient data at 28MB/sec making real-time sequence execution possible. Image reconstruction of a 256x256 2DFT dataset is performed in 180ms without hardware acceleration, and the Python UI attains plotting and image update rates of 10-25Hz. The system measures 7.5x5x3 inches and power consumption is 9.5W (5V 1.9A) making battery-powered portability easily possible. Component cost is ~$500.

Discussion

While using Medusa components enabled rapid development, redesigning specifically for the BeagleBone would yield improved data rates and a ~50% size reduction. The AM3359 SoC also contains real-time co-processors (PRUs) and a NEON floating-point unit that could accelerate sequence execution and reconstruction respectively. The choice of Python was key to simple user-accessible development while maintaining low cost (no Matlab). Python's broad multi-platform support allows pulse sequences and user interfaces to be developed and tested on any Mac, Windows, or Linux machine before being deployed on the stand-alone console.

Conclusions

We have demonstrated a stand-alone user-programmable MRI console based on Linux and Python and adaptable to a wide variety of small-scale, portable, or embedded MR applications. Continuing work is focused on enhancing performance and expanding the set of Python tools and pulse sequences.

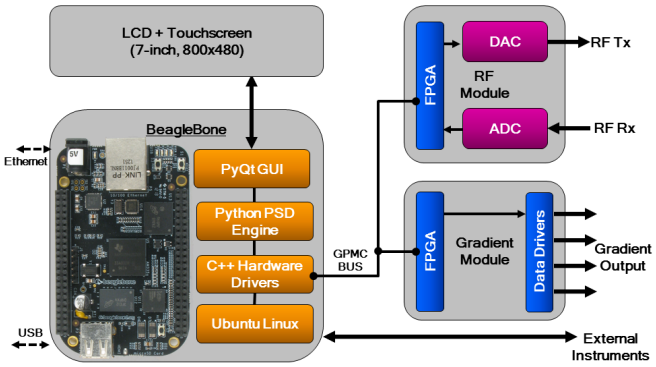

Figure 1: A block diagram of the compact MRI console. A linux-powered BeagleBone embedded computer ($45) handles user interface, pulse sequencing, and image reconstruction duties without need for a separate PC. Medusa RF and Gradient modules are connected via the BeagleBone GPMC to provide the hardware interfaces necessary for MR imaging.

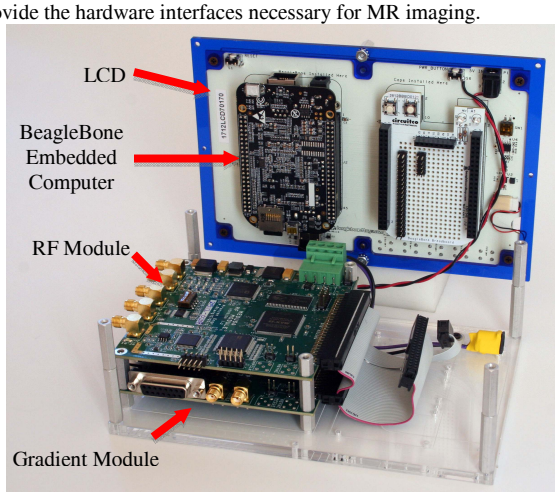

Figure 2: The major hardware components of the stand-alone console.

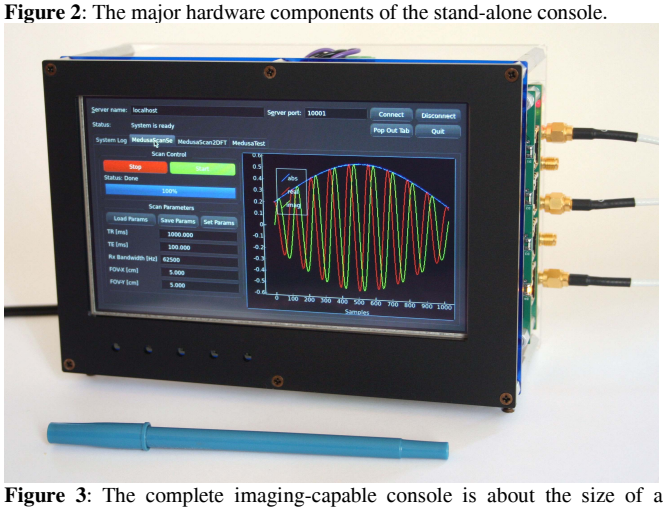

Figure 3: The complete imaging-capable console is about the size of a hardback book. A 7-inch touchscreen LCD provides a versatile user interface, with visual GUI elements and pulse sequences programmed in Python. The data plot shown is a spin-echo captured on a GE 1.5T magnet.

References [1] https://gate.nmr.mgh.harvard.edu/wiki/Tabletop_MRI. [2] Twieg, ISMRM 2013 #0139. [3] Etezadi-Amoli, ISMRM 2013 #723. [4] Shultz, IEEE-TMI 2013 Vol31:4 p938. [5] Wright SM, MAGMA 2002. [6] www.magritek.com [7] pure-devices.com [8] www.beagleboard.org [9] Stang, ISMRM 2007 #925.

Funding: Procyon Engineering, NIH R01EB008108, P01CA159992.