Embedded Software Projects
Version 0.1

There have been a few requests by students and newbies for project ideas over the past few months. Here I have tried to compile a list of possible projects around embedded systems software. Please note, that these projects focus mostly on embedded software and may not be suitable for folks focusing on hardware side of things.

At the end of the day, treat these as starting points and build on top of them or modify them based on hardware and software resources available on hand.

I also have overview slides on ARM programming available at:
http://ajaydudani.googlepages.com/Programming_the_ARM_Microprocessor_for_Embedded_Systems.pdf

If you like to see modifications of submit more ideas to this compilation, you can send them to me at ajay.dudani@gmail.com

Cheers,
Ajay Dudani
12th March 2007

Project Title	Basic sensor
Level	Beginner
Keywords	Sensors
Description	There are different type of sensors available and in use out in the market. All sensors serve different purpose like measuring pressure, temperature, and velocity among others. Normally, a simple pressure sensor may not require an ARM processor, but it can make a student project to get used to ARM development environment. There are atleast three aspects of project that you can select and customize. 1. The sensor itself. The application/end result could differ based on the sensor you select. Possible options include the ones mentioned above – temperature, pressure and velocity. Once could also go on to select a webcam as a sensor (think of a security monitor that sees if image has changed and alert activity in a room). Another possible sensor is joystick where you detect how the use is pushing buttons and controlling joystick. 2. Communication between sensor and board. This could be over a serial or parallel interface. This would expose you to bunch of protocols involved here – USB, serial bus, parallel lines, communication directly with processor, shared bus, DMA etc. 3. Application on board. On the board, you could go with a full fledged embedded OS – which will do all the driver work for you (eg in case of USB) or you could go with your own binary based on basic C routines and build your own driver. You also have the leverage of building an application that does something useful with the sensor data that you read.

Project Title	Fast address space switching
Level	Intermediate
Keywords	Embedded OS, MMU
Description	This project would deal with embedded operating systems and how it takes advantage of ARM processor capabilities. Please check on the features enabled on your board to see if it support MMU. Search the net for "fast address space switching". Generally caches are used to speedup application execution. But if memory protection is enabled on the processor then whenever the application context changes the the cache has to be flushed. Fast address space switching addresses this to some extent. Your task in the project would be to understand this mechanism of fast address space switching. Select an embedded operating system and implement the use of fast address space switching. Ofcourse you will need to have the source code to OS you want to modify. So it would be good to select embedded Linux or eCos for this project. Once you have a good handle of the implementation, you should compare the performance improvements achieved with your fast address space switching implementation in an environment where many application contexts are involved along with cache hit/miss data among other statistics.

Project Title	Power savings
Level	Intermediate
Keywords	Embedded OS, power
Description	Use of power continues to remain an important aspect in embedded systems. Modern embedded processors provide frequency and voltage scaling – meaning you can switch the processor frequency at different levels based on work loads. This is different for 90's technology where there were just two basic modes – sleep and 100% running. With today's architectures you can run the processor at say 30% or 60% of processor maximum frequency. Since power consumption is directly proportional at frequency at which processor is running – lower frequency contributes to lower power usage. Important criteria is to make sure that user experience or processes running are not affected by running the processor at lower frequency. Possible tracks to take with this project would involve: 1. Modify the scheduler to take into account the processes running on the system and send commands to processor to run at proportional frequency. Search for "Energy-conserving feedback EDF scheduling for embedded systems with real-time constraints" as a starting point 2. Another option is to take a specific application – say MPEG4 viewer and understand how frequency scaling affects the decoding of frames. 3. Consider possibilities of changes to compiler or mechanisms by which the application developer can hint the compiler about its CPU usage. For example, if application is not critical or can accept lower CPU speed – it can indicate that using a pragma directive to compiler. The compiler can then, when generating assembly code, auto insert code to switch the processor speed to desired level. As a part to deliverables for this project, it would be nice to have an analysis of change in power consumption before and after your changes. Also study how it impacts the processes or user experience as appropriate.

Project Title	Embedded device monitoring
Level	Intermediate
Keywords	Embedded OS, SNMP, networking
Description	There are a whole lot of embedded devices sitting around the world performing specific functions. In case of routers and other networking equipments the network administrations want to monitor and control these devices from a single location over the network rather than go around to each one of them and configuring and monitoring them. An industry standard way to do this is based on SNMP – Simple Network Management Protocol. SNMP runs on top of TCP/UDP and allows the administrator to monitor and control the devices over the networks. The goal of this project are to familiarize yourself with SNMP, understand how it interacts with embedded device and the tools available on the desktop side – the SNMP client. Also take a look at net-snmp code. On implementation side, your task would be to define a custom MIB, code up so you can get/set values on that custom MIB on the device and control these attributes of custom MIB from the desktop SNMP client.

Project Title	Embedded microkernel
Level	Advanced
Keywords	Embedded OS
Description	There are quite a few monolithic kernel embedded OS. Also, there have been development of microkernel for desktops. Your task in this project would be to understand differences between monolithic and microkernel and then take a microkernel like GNU Hurd or L4 microkernel and port it to ARM based board. Microkernels in the basic form are simpler than monolithic kernel since they basic mechanisms they provide is mosly inter process communication. Other services are provided by higher level components. The project would involve interacting with open developer communities of GNU Hurd or L4 for support. Also lookup the websites of these projects for more ideas around this project. Another challenging aspect here is to modification & design decisions involved to make these suitable for embedded systems.

Project Title	Flash file system
Level	Intermediate
Keywords	Embedded OS
Description	It is easy to find development boards that have MMC or Compact flash reader on the board. Your task in this project is to write FAT16 or FAT32 file system. You should be able to read and write data on the MMC/CF card and format the card based on FAT32. An extension of the project is to make the MMC/CF card contents available on a Windows machine through USB interface. Your board will needs USB interface for this. You will have to setup the USB port as a slave and as a UID device so Windows detects it as an external HD. The key here is to make sure that contents of MMC/CF card shows up as a D: drive or E: drive on the Windows machine. On similar lines, you could develop support for ext3 or reiserfs file system for MMC/CF card and be able to mount on a Linux machine. To reiterate there are two aspects to this project – the file system aspect and other is to work on USB side and understand how USB devices are setup, detected and communication between master and slave device.

Project Title	Lego Mindstorms
Level	Intermediate
Keywords	Embedded OS
Description	Lego Mindstorms provides a set of programmable components like camera, electric motors, sensors etc. There are programming tools available from Lego as well as opensource tools available (like BrickOS) using which you can create your own fun and useful embedded systems. Look at the Wikipedia entry for Lego Mindstorms. You are only limited by your imagination (and ofcourse hardware availability) on what you can do using these lego components.

Project Title	Biometrics
Level	Intermediate
Keywords	Embedded OS, security
Description	Biometrics is becoming more important for security in embedded systems. Think of systems like eye scanner, or fingerprint authentication. There are these embedded devices that need to perform such functions. You could select one of the following as the means for biometrics: ✓ Fingerprint reader ✓ Camera – for face capture and authentication ✓ Voice based The security aspect comes in when you connect the data captured from above system – say finger print, validate it and allow the user to perform a previliged operation – like logging on to a Windows Mobile device, or unlock a file on the local file system. The issues here are not entirely unique to embedded systems space, but some restrictions come into the picture due to limited CPU, memory availability on the embedded device