Microcontroller and DSP System

Various Applications Using Arduino Microcontroller….

Third Year, 1st Semester

Lecture No.2

Ass. Lecturer: Yousif Allbadi
M.Sc. of Communications Engineering
Yousif_allbadi_eng@uodiyala.edu.iq
yousifallbadi@uodiyala.edu.iq

University of Diyala
College of Engineering
Department of Communications Engineering
2020-2021

Embedded Systems

An embedded computer is frequently a computer that is implemented for a particular purpose. In contrast, an average PC computer usually serves a number of purposes: checking email, surfing the internet, listening to music, word processing, etc... However, embedded systems usually only have a single task, or a very small number of related tasks that they are programmed to perform.

Every home has several examples of embedded computers. Any appliance that has a digital clock, for instance, has a small embedded microcontroller that performs no other task than to display the clock. Modern cars have embedded computers onboard that control such things as ignition timing and anti-lock brakes using input from a number of different sensors.

In general, an Embedded System:

- Is a system built to perform its duty, completely or partially independent of human intervention?
- Is specially designed to perform a few tasks in the most efficient way.
- Interacts with physical elements in our environment, viz. controlling and driving a motor, sensing temperature, etc.

An embedded system can be defined as a control system or computer system designed to perform a specific task. Examples:

- Pen drives (for controlling the communication between P.C. and Flash Chip and also the small LED!).
- Hard disks (again for the same purpose).
- Mouse (Reads and Interprets the Sensors and send final result to P.C.), Keyboards.
- Printers: Ever opened a printer for installing ink cartridge? Then you must have seen the printed head. There are motors to control the print head and the paper movement. Your P.C. is not directly connected to them but there is built in MCU of printer to control all

these. Your P.C. just sends the data (pixels) through the communication line (USB or parallel). But the MCU used here is fairly fast and has lots of RAM.

- Automobiles.
- Calculators, Electronic vending machines, Electronic weighing scales, Phones (digital with LCD and phonebook).
- Cell phones.

Embedded Programming

Key Points in Embedded Programming

- Code Speed - Timing constraints, limited proc. power
- Code Size - Limited memory, power, physical space

Programming Methods

- Machine Code
- Low level language - Assembly
- High level language - C, C++, Java
- Application level language - Visual Basic, scripts, Access

Why use C in embedded programming?

- Fairly efficient
- Provides an additional level above assembly programming
- Supports access to I/O
- Ease of management of large embedded projects

Why use assembly?

- High speed, low code size
- However, difficult to do a large project in assembly

Software Development

You are probably already familiar with software development in general, either due to software engineering courses or because you have done some private /commercial projects. You therefore know about the different phases of software development, from the requirements analysis down to testing and maintenance. You may have used some tools to support these phases, and you have most likely used some nice integrated development environment (IDE) paired with elaborate debugging support to do your programming and debugging. Software development for embedded systems is in large parts comparable to development for a workstation. There are, however, some crucial differences which make development for embedded systems more difficult.

The development of low-level embedded software seems to have its own rules. Ultimately, one would assume that the specification of the underlying hardware constitutes a programming interface just like the application programming interface of an operating system. It could be argued that if your development process is sound, and if you follow the rules given in the datasheet just like you adhere to the API specification of your operating system, you should be fine. That would make anyone who is good at writing, say, database applications a good embedded software developer as well. Unfortunately, despite the fact that there are no conceptual differences to the development process, due to the pitfalls and peculiarities of direct hardware access, embedded software developers actually face quite different challenges than those working in higher level environments.

In fact, one of the most dangerous pitfalls of embedded software development seems to be the perceived similarity to high-level application programming. Not too long ago, embedded systems were almost exclusively programmed in Assembler, which made for an apparent difference, as Assembler was widely perceived as a rather obscure and demanding programming language. Nowadays, however, the language of choice is C, sometimes even higher-level languages like C++ oder Ada. And since most people seem to think that a C source code is a C source code, it is sometimes expected that anyone who is sufficiently prolific in C should do just fine in embedded systems development.

The key problem seems to be that embedded software development is more than just software development. Embedded developers do not work merely at the border between software and hardware, handing over bits and bytes for them to be miraculously transformed into light, sound, or motion. Instead, they routinely have to cross over to the hardware side to fully understand the interaction between their program and the associated hardware.

Finally, an embedded system has tight resource constraints, which may affect the way you program. Whereas in workstation development, memory usage is generally of no importance, and even speed may be of low importance, these factors become pivotal in embedded systems. Another issue is power consumption, which is of no concern in workstations but is of extreme importance in battery-powered embedded systems. These constraints may affect your choice of algorithm and your implementation style.

Programming

- We will study a programing of microcontroller (Arduino) in details later