ECE155: Engineering Design with Embedded Systems Winter 2013

Lecture 2 — January 9, 2013

Patrick Lam *version 1*

Sensors and Actuators

Embedded systems differ from general-purpose computers in that their interaction with the outside world is much more important than in a general-purpose computer.

What are some examples of phenomena that embedded systems care about?

Sensors.

Sensors convert input from the external environment into a form suitable for use by a computer system. What are examples of sensors?

Most phenomena in the outside world are analog and continuous-time, while the computer is digital and discrete-time, so something has to convert between these realms.

For instance, a light sensor might represent intensity as an analog voltage, while the computer needs a stream of bits.

Analog-to-Digital Converters. To go between the realms, use an **ADC**, which converts a continuous-time analog signal into a discrete-time digital signal.

The conversion is approximate, and loses information if the samples are too infrequent[1] or if the samples don't capture enough information (i.e. their resolution is too poor).

Note that the digital signal is a finite set of pairs, each of which contains a sample and a time for that sample.

[1] See the Nyquist sampling theorem.

Actuators. Actuators convert output from the computer system into some effect on the environment. What are some examples of actuators?

Some actuators require analog voltage signals. In those cases, you have to feed the discrete-time data to a digital-to-analog converter (**DAC**).

Combined Sensors and Actuators. Of course, we can combine sensing and actuating in a single device. Some devices automatically do this, like piezoelectric sensors. Any relatively modern game controller also combines sensors and actuators, particularly the Wii Remote[2]. Phones also combine sensors and actuators. What sensors and actuators did we see in class?

We also saw quite a bit of special-purpose hardware on the phone. What did we see?

Programming the Embedded System

We'll discuss how to program the CPU next. Programming an embedded system is somewhat similar to programming a computer, but there are differences in the programming environment and in the structure of the code. We run an *embedded control program* on an embedded system, e.g. a modem. It monitors sensor inputs and controls actuators.

Let's talk first about the structure of the code. Embedded control programs:

- boot automatically on device power up;
- never terminate under normal use;
- process a stream of inputs and outputs; and
- care about timing.

Embedded control programs may run on top of embedded operating systems. An *embedded operating system* is a special type of operating system designed for use in embedded computer systems; it has these properties:

- is compact and efficient;
- cares about battery life;
- provides APIs between devices and application software; and
- favours portability.

What are some examples of embedded operating systems?

[2]Wii Remote, http://en.wikipedia.org/wiki/Wii_Remote, Accessed January 5, 2010.