CprE 288 – Introduction to Embedded Systems
(Syllabus & Course Overview)

Instructor:
Dr. Phillip Jones
Overview of Today’s Lecture

• Announcements
• Syllabus
  – Policies
  – Grading Scale
• Course Overview
  – Lab Hardware Introduction
  – Learning Objectives
  – Course Schedule
• Embedded Programming
  – Base Conversion
  – Processor Architectures

http://class.ece.iastate.edu/cpre288
Announcements

• HW 1: Will be posted on course website (Due Sunday 1/24)
• Labs start next week
• http://class.ece.iastate.edu/cpre288 (class website)
• There will be weekly assignments (Due on BlackBoard Learn)
  – Short assignments to keep everyone active
• **Individual** assignments; work alone
• **Typed** homework answers are required
• Use Blackboard for HW submission
• Solutions will be posted 2-3 days after the due date
  – 10% for each day late (time based on when your lecture is)
• Lab attendance is mandatory
  – Automatically fail a lab with an unexcused absence
• Labs are partner activities for the purpose of teamwork (no exception)
• If you have to miss a lab, inform the instructor prior to the start of lab

• There are 9 labs and a lab project (Mars Rover)
  – Prelab, if given, is due in the beginning of the lab
  – Lab demo is due in the beginning of lab the following week
  – Labs can be demo’ed 1 week late for a 20% deduction
• Lab is in Coover 2041
  – See course webpage for Section times

http://class.ece.iastate.edu/cpre288
• Lecture attendance strongly encouraged
  – Please review the lecture notes if not attending
  – Occasional absence is OK
  – Exams questions will reward those that participate in lecture activities
• Review sessions
  – The lecture before each exam
• Exam 1 – 7th week – based on materials covered in the first 5 weeks – mostly C), 75 minutes
• Exam 2 – 12th week – based on second 5 weeks of lectures – mostly peripherals), 75 minutes
• Exam 3 – During Final week – based on last 5 weeks of lectures – mostly AVR architecture and assembly, 75 minutes

This schedule is tentative

• Open notes (must be hard copies)
• Exams 45%
  – Exam 1: 15%
  – Exam 2: 15%
  – Exam 3: 15%
• Homework: 15%
• Laboratory Exercises: 25%
  – Nine laboratory exercises
• Laboratory Project: 15%
• Work independently on homework & exams
• Seek peer help to better your knowledge and skills rather than your grades
• Do not barrow code from others.

• Good questions to ask:
  – “Could you explain how pointers work?”
  – “I don’t understand this io_t struct. What is it?”
  – “Can you explain successive approximation?”
• Bad Question / Actions:
  – “Can you show me your answer for question 3?”
  – “Can you e-mail me your homework?”
  – “E-mail me your source code for taking a Sonar measurement”
  – “If I do homework question 1, will you do question 2 and then we can trade?”
The following acts are considered a violation of the University’s student conduct policy (not exclusively). Suspected offenders will be sent to the Dean of Students Office for investigation.

- Sending or receiving any fragment of source code from another group, or from someone who previously took the class, is an offense.
- Sending or receiving answers to homework assignments is an offense.
- Copying answers from another person’s exam is an offense.

Those convicted of academic dishonesty will receive at minimum a zero on the assignment, and may, at the discretion of the instructor, receive an F for the course.
• Take time to look at the course website:
  – http://class.ece.iastate.edu/cpre288
  – Shows list of supplementary books on Syllabus page
CprE 288 is:
A class where students learn about embedded systems through writing C code for the VORTEX platform.

VORTEX is the codename for:
- Cerebot II
- ATmega128 microcontroller
- iRobot Create
- Attachments (stepper motor, servo, sonar, IR distance, LCD, etc.)

http://class.ece.iastate.edu/cpre288
• Digilent’s Cerebot II
  – “Break-out” board for the microcontroller
  – Microcontroller is an ATmega128
Course Overview | ATmega128

- MCU (Microcontroller unit)
- Manufactured by Atmel
- Clock speed
  - 16 MHz processor
- Memory
  - 4 KB of EEPROM (for long term storage)
  - 4 KB of SRAM (data memory)
  - 128 KB of Flash (program memory)
- Lots of features
  - Timers, Input Capture, PWM, ADC, SPI, UART, etc

http://class.ece.iastate.edu/cpre288
• **List of Sensors**
  – Omnidirectional IR sensor
  – Left & Right bump sensors
  – Four cliff sensors along the front
  – Wall sensor
  – All three wheels have drop sensors

• **2 wheels for movement**

• **Students program the MCU on the Cerebot II**
  – Communication between the MCU and iRobot Create occurs over serial
  – We will use an API called *Open Interface* to communicate
• AT JTAG ICE MKII
  – JTAG = Joint Test Action Group
  – Interface between Cerebot II board and the computer
  – Enables debugging of many of Atmel’s AVR MCUs (microcontroller unit)
  – Lab TA’s will stress how fragile this can be. So use with care.
Course Overview

• Hardware is not cheap! (This isn’t just an Arduino)
• iRobot Create
  – $129
• Cerebot II
  – $39.95
• AT JTAG ICE MKII
  – $299
• Making a cool robot
  – Priceless
• Learn to read datasheets/manuals in order to develop practical applications
• Learn basic hardware and software debugging
• Be able to program and design applications for embedded systems
• Gain experience programming in C for the Atmega128
• Understand basic computing concepts such as:
  – Interrupts
  – Interrupt Service Routines (ISR)
  – I/O subsystems
  – How processors work, registers, program memory, etc
• Understand the Atmel processor architecture
• Understand how C is converted to assembly code
Course Overview | Schedule

Three general phases:

Exam 1: (C-programming, Micro-controller basics)
- **Weeks**
  - Overview of course and lab hardware
  - Review of C programming
  - Special function registers
  - iRobot Create overview
  - Interrupt handling (ISR)

Exam 2: (Peripherals)
- **Weeks**
  - Timers
  - Serial (USART)
  - Distance sensors (IR & Sonar)
  - Analog to Digital Conversion (ADC)
  - Input Capture
  - Output Compare and Pulse Width Modulation (PWM)

Exam 3: (Assembly)
- **Weeks**
  - Start of Lab Project
  - AVR Assembly programming

http://class.ece.iastate.edu/cpre288
EMBEDDED SYSTEMS
What are Embedded Systems?

• Examples:
  – Programmable thermostats
  – GPS Asset tracking
  – Remote controls
  – iRobot Create

http://class.ece.iastate.edu/cpre288
What are Embedded Systems?

• Example:
  – Mars Sojourner Rover (1997)
    • ~25 pounds
    • 25 x 19 x 12 inches
  – 8-bit Intel 80C85
    • 100 KHz Clock speed
Embedded Programming

• Key factors in embedded systems
  – Code speed – timing constraints, limited processor
  – Code size – Limited memory size
  – Energy – portability means less battery consumption

• Programming Methods
  – Machine Code 0001 1101 1100
  – Low Level Languages Assembly
  – High Level Languages C, C++, Java
  – Application Level Languages VBA, Access, scripting
Embedded Programming

• Why use C for embedded systems?
  – Designed to expose machine details for efficiency
  – Borrows features of contemporary high level programming languages
  – Easier to manage large embedded projects
  – There is a C standard

• Why use assembly?
  – Pros: High speed, low code size, low energy
  – Cons: Low programmer productivity, there is no standard (every processor can have its own special instructions)
Methods for Representing Data

- **Bit**
  - 1 (True)
  - 0 (False)
- **Nibble** (less commonly used)
  - 4 bits
- **Byte**
  - 8 bits
- **Word**
  - 16 bits
- **Double Word**
  - 32 bits
Methods for Representing Data

• Three of the most common forms of notation
  – Decimal (base 10) 0123456789
  – Hexadecimal (base 16) 0123456789ABCDEF
  – Binary (base 2) 01

• Another less common form is octal (base 8)

• Converting between forms
  – When converting binary to hexadecimal, every group of 4 bits (nibble) represents a hexadecimal digit
  – Examples:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>1010</td>
<td>A</td>
</tr>
</tbody>
</table>

http://class.ece.iastate.edu/cpre288
Base Conversion

• Methods to convert between bases
  – Use a calculator or the internet
    • TI 89
    • Microsoft’s Calculator in Programmer mode
    • Google
      – Example searches:
        – 128 in binary
        – 0b0010 in hex
        – 0x03ef in decimal
    • Wolfram Alpha
      – Example searches:
        – All Google queries
        – 0xef32
        – 0b0101
  – Compute by hand
    • Every EE/CprE/CS/SW engineer should know how to change base

http://class.ece.iastate.edu/cpre288
Base Conversion (by hand)

- Base n to base 10

Problem: Convert 0b01001011 to base 10

Solution:
Label each column and add.

\[
\begin{array}{cccccccc}
2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
128's & 64's & 32's & 16's & 8's & 4's & 2's & 1's \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
\end{array}
\]

\[64 + 8 + 2 + 1 = 75\]

- Examples will be demonstrated on the white board
Base Conversion (by hand)

• Base 10 to base n

Problem: Convert 175 to base 16

Solution:
Create a table of the columns in a base 16 number and subtract from the original number:

\[
\begin{array}{c|c}
16^1 & 16^0 \\
\hline
16's & 1's \\
A & F \\
\end{array}
\]

\[
175 - 160 = 15
\]

0xAF

• Examples will be demonstrated on the white board
• Syntax in C (for AVR Studio)
  – Computers understand binary
  – The following lines of code are all the same (the compiler does not care what base the programmer uses):

```c
char x = 2 + 1;
char x = 0b10 + 1;
char x = 0x2 + 1;
char x = 0x02 + 0x01;
```
Components of a Computer

• Central Processing Unit
  – Interprets and carries out all the instructions contained in software

• Memory
  – Used to store instructions and data
  – Random Access Memory (RAM)
  – Read Only Memory (ROM)

• Input/Output
  – Used to communicate with the outside world
How Processor Works

**Machine instruction**: Tell computer what to do in a single step
- A bundle of binary bits with certain formats
- Only asks for simple operations
- **Assembly**: textual notations of machine program

Example in C:

```c
x = a + b;
```

Execution steps at assembly/machine level:
- \( R1 \leftarrow a \)
- \( R2 \leftarrow b \)
- \( R3 \leftarrow R1 + R2 \)
- \( x \leftarrow R3 \)

A compiler does the translation between C code and machine code!
Microprocessor

• A single chip that contains a whole CPU
• Examples:
  – Intel P4 or AMD Athlon in desktops/notebooks
  – ARM processor in Apple iPod
  – Has the ability to fetch and execute instructions stored in memory
• Has the ability to access external memory, external I/O and other peripherals
Processor Architecture

• von Neumann Architecture
  – Single data area that stores both program memory and data memory

• Harvard Architecture
  – Separate memories, one for data and one for program instructions

• RISC Architecture (Reduced Instruction Set Computing)
  – Reasoning: reduced number of instructions will increase simplicity and lead to faster processors, fewer transistors, and less power.
ATmega128 Processor Architecture

- 8 bit processor
  - size of bus is 8 bits
  - size of registers is 8 bits
- Harvard architecture
- RISC architecture
- 133 instructions
History of Microprocessors

• 1950s - The beginning of the digital era and electronic computing
• 1969 – Intel is a small startup company in Santa Clara with 12 employees
  – Fairchild, Motorola are large semiconductor companies
  – HP and Busicom make calculators
• 1971 – Intel makes first microprocessor the 4-bit 4004 series for Busicom calculators (~100 KHz)
• 1972 – Intel makes the 8008 series, an 8-bit microprocessor,
  – ATARI is a startup company
  – Creates a gaming console and releases PONG
History of Microprocessors

• 1974 – the first real useful 8-bit microprocessor is released by Intel – the 8080
  – Motorola introduces the 6800 series
  – Zilog has the Z80

• 1975 – GM and Ford begin to put microcontrollers in cars
  – Many cars today have over 100 microcontrollers
  – TI gets into the microprocessor business with calculators and digital watches

• 1977 – Apple II is released using MOS 6502 (similar to motorola 6800). Apple II dominated from 1977 to 1983

• 1978 – Intel introduces the first 16-bit processor, the 8086
  – Motorola follows with the 68000 which is ultimately used in the first Apple Macintosh
History of Microprocessors

• 1981 – IBM enters the PC making market and uses the Intel 8088 – proliferation of the home computer
• 1982-1985 – Intel introduces the 32-bit 80286 (4 MHz )and 80386
• 1989 – 80486 is being used in PC’ s, able to run Microsoft Windows
• 1992 – Apple, IBM and Motorola begin to make PowerMac and PowerPC’ s using Motorola chips
• 1993 – Pentium chip is released (60 MHz)
• 2000 – Intel Pentium 4 chip is released (1.3 GHz)
• 2001 – IBM Power 4 chip, first commercial (non-embeded) multicore (2 cores, 1.3 GHz).
• 2011 – Intel E7-8870, 10 cores (2.8 GHz).
Microcontroller

- Essentially a microprocessor with on-chip memories and I/O devices
- Designed for specific functions
- All in one solution - Reduction in chip count
- Reduced power consumption
- Reduced cost
- Examples
  - MC68332, MC68HC11, PPC555, Atmel family (e.g. Atmega128)
- More details of components later
  - A/D converters, temperature sensors, communications, timing circuits, many others
Microcontroller (ATMega128)

- **On-chip memory**
  - Instruction
  - Data
- **Microprocessor**
  - 8-bit
- **I/O modules**
  - ADC (Analog to Digital Converters)
  - Timers/Counters
    - Many uses
  - USARTs
  - Multi-Purpose Ports
  - A-G
Microcontroller (ATMega128)

- On-chip memory
  - Instruction
  - Data
- Microprocessor
  - 8-bit
- I/O modules
  - ADC (Analog to Digital Converters)
  - Timers/Counters
    - Many uses
  - USARTs
  - Multi-Purpose Ports
  - A-G
Microcontroller (ATMega128)

- On-chip memory
  - Instruction
  - Data
- Microprocessor
  - 8-bit
- I/O modules
  - ADC (Analog to Digital Converters)
  - Timers/Counters
    - Many uses
  - USARTs
  - Multi-Purpose Ports
  - A-G
Microcontroller (ATMega128)

- On-chip memory
  - Instruction
  - Data
- Microprocessor
  - 8-bit
- I/O modules
  - ADC (Analog to Digital Converters)
  - Timers/Counters
    - Many uses
  - USARTs
  - Multi-Purpose Ports
  - A-G
Why Study Microcontrollers

This course may serve for several purposes:

– Build useful applications
– Practice programming and debugging skills
– Understand computer internals

It paves the way to learning computer design, operating systems, compilers, embedded systems, security and other topics.

– Microcontrollers have everything in a typical computer: CPU, memory and I/O.
LAB 1 QUICK OVERVIEW
Lab 1: Introduction to the Platform

Purpose: Introduction to the AVR Studio 6 and VORTEX Platform

– AVR Studio 6: The integrated development environment (IDE) for Atmel AVR platforms

– VORTEX: An integrated hardware platform of iRobot Create and Cerebot II microcontroller board
An IDE from Atmel for AVR platforms

- Source code editing
- Compiling building
- Download binary to boards
- Debug
- Simulation
Lab 1

Lab 1: Introduction to the AVR Studio 6

• Part 1 “Hello, world”
  – Build, download, and execute

• Part 2 Simulated Environment

• Part 3 Rotating Banner
  – The message has 34 characters and the LCD can only show 20 characters per line at a time
Programming Example

How to display a rotating banner?
A smaller example: 10-char. display, 19-char. message

The screen Message

Shift for one character every one second

Welcome to CPRE 288
What’s the desired program behavior?

First display “Welcome to ” and wait
Then display “elcome to C” and wait
Then display “lcome to CP” and wait
Then display “come to CPR” and wait
and so on
Give a general but precise description

First show characters 0-9 and wait
Then show characters 1-10 and wait
Then show characters 2-11 and wait
Then show characters 3-12 and wait
and so on
Describe program’s behavior

set starting position at 0
loop forever
  clear the screen
  display 10 chars from the starting pos.
  shift the starting pos. to the next position
  wait for one second
end loop
Some details to take care

“display 10 chars from the starting pos.”

“shift the starting pos. to the next position”
Part 3. Rotating Banner
Show “Microcontrollers are loads of fun!” in a rotating style
  – The message has 34 characters and the LCD line has 20
  – Shift in first 20 characters one by one, with 0.3 second delay
  – Start to rotate and continue till the last character is shown, with 0.5 second delay
  – Continue rotating until the screen becomes clear, with 0.5 second delay
  – Repeat this procedure
First, have a function to print the banner for one time

```c
void print_banner(char *msg, int start, int end);
```

This makes the rest of programming easier
Lab 1 Programming Exercise

Idea 1: A forever loop of **three phases**

Phase 1: Shift in the first 20 characters
Phase 2: Rotate until the last character is displayed
Phase 3: Rotate until the last character is shifted out
int main()
{
    while (1)
    {
        for (...) // Phase 1
            ...
        for (...) // Phase 2
            ...
        for (...) // Phase 3
            ...
    }
}