Lab 1
Embedded Programming

CS397/497 – Wireless Protocols for IoT

Branden Ghena – Winter 2021

Today’s Goals

- Provide background information for students not familiar with embedded systems software

- Describe details about the Nordic software system
  - And the nrf52x-base software system

Outline

- **Microcontrollers**
- Programming Embedded Systems
- Embedded Software

What is a microcontroller anyways?

- Entire computer in a single chip
  - Processor
  - Working memory: SRAM (like RAM)
  - Nonvolatile memory: Flash (like SSD)
  - Peripherals
    - I/O pins
    - Analog Inputs and Outputs
    - Timers
    - Wireless radios
    - Cryptography accelerators
  - Power management
  - Buses
    - UART
    - I2C
    - SPI
    - USB

How is a microcontroller different?

- A very constrained computer
  - Simple processor
    - 16 or 32 bits
    - Processor speed in MHz
    - Single core, pipelined processor
    - No cache, or maybe a very small instruction cache
  - Memory in KB
    - Code executes right from Flash (which is part of the address space)
  - Often no OS support
    - We'll be programming essentially "bare-metal" in this class

Nordic semiconductor microcontrollers

- 32-bit ARM microcontrollers with integrated wireless radios
  - 2012 – nRF51 series with Cortex-M0
  - 2015 – nRF52 series with Cortex-M4
    - nRF52840
      - 64 MHz processor
      - 512 KB Flash
      - 128 KB RAM
      - BLE and Thread support
  - Very capable and low energy compared to other microcontrollers
  - Also very good software support
    - Which is incredibly rare

Overview of nRF52840 capabilities

- Go to Figure 1: Block diagram
- https://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.1.pdf

- Online documentation:
  - https://infocenter.nordicsemi.com/topic/ps_nrf52840/keyfeatures_html5.html

nRF52840DK

- nRF52840 microcontroller
  - 4 buttons, 4 leds
  - Antenna for wireless
  - Various headers for connection to pins

- JTAG programmer
  - Connect through top USB port
  - Enables loading of code and runtime debugging

Outline

- Microcontrollers
- **Programming Embedded Systems**
- Embedded Software

Embedded systems are programmed in C

- Occasionally assembly
  - Rarely other things (Rust, C++, Lua, Python)

- But even the few things C gives you aren’t necessarily available
  - Heap space is likely nonexistent
    - You have to choose some space in RAM to save as a heap
    - And then include the algorithm for allocating that memory
  - Printf is often nonexistent too
    - There’s no STDIN/STDOUT/STDERR because there is no shell
    - We hooked up printf for you on the nRF52840DK though!

How do I build code for microcontrollers?

- You need a couple of things
  - Memory layout ".ld" file explains where memory is for linker
  - A compiler toolchain for the correct architecture

- Cross compilers
  - Run on one architecture but compile code for another
    - Example: runs on x86-64 but compiles armv7e-m
  - GCC is named: ARCH-VENDOR-(OS-)-ABI-gcc
    - arm-none-eabi-gcc
      - ARM architecture
      - No vendor
      - No OS
      - Embedded Application Binary Interface
    - Others: arm-none-linux-gnueabi, i686-unknown-linux-gnu

How do I load code onto microcontrollers?

- JTAG (Joint Test Action Group)
  - Hardware built into the microcontroller for testing purposes
  - Can arbitrarily read/write memory
  - Can single step process too, at runtime!
    - GDB can connect to it! (sort of)
  - RTT (Real Time Transfer) makes printf work over JTAG

- Serial bootloaders
  - Software runs on the microcontroller at boot that waits a short time for someone to contact it and upload code
  - Convenient, but sometimes flaky

Outline

- Microcontrollers
- Programming Embedded Systems
- **Embedded Software**

Embedded software

- There are a multitude of embedded software systems
  - Every microcontroller vendor has their own
  - Popular platforms like Arduino

- Embedded OSes
  - Contiki, Riot, Zephyr, Mynewt, FreeRTOS, Tock

- We’re talking about the Nordic software plus some extensions made by my research group

Softdevice

- Sort of like an OS that only manages the radio
  - Always running underneath your code
  - You don’t have control over it, but can request things
    - Actually through system calls
  - Block of code that you cannot edit

- Handles time-sensitive behaviors and can be certified
  - This is the reason GDB doesn’t really work

- We’ll load it automatically, and some of the library calls you make will interact with it, but you likely won’t have to think about it

Software Development Kit (SDK)

- Libraries provided by Nordic for using their microcontrollers
  - Actually incredibly well documented! (relatively)
  - Various peripherals and library tools

- SDK documentation
  - https://infocenter.nordicsemi.com/topic/sdk_nrf5_v16.0.0/index.html
  - Warning: search doesn't really work

- Most useful link is probably to the list of data structures
  - https://infocenter.nordicsemi.com/topic/sdk_nrf5_v16.0.0/annotated.html

nRF52x-base

- Wrapper built around the SDK by Lab11
  - Branden Ghena, Brad Campbell (UVA), Neal Jackson, a few others
  - Allows everything to be used with Makefiles and command line
  - https://github.com/lab11/nrf52x-base

- We include it as a submodule
  - It has a copy of the SDK code and softdevice binaries
  - It has a whole Makefile system to include to proper C and H files
  - We include a Board file that specifies our specific board's needs and capabilities

- Go to repo to explain

Outline

- Microcontrollers
- Programming Embedded Systems
- Embedded Software