Lecture 27: Complementary Notes

Seyed-Hosein Attarzadeh-Niaki

Outline

- Fixed-point numbers
  - Representation
  - Arithmetic
  - ARM CSMSIS-DSP
- Arduino platform
  - Hardware
  - Software

FIXED-POINT NUMBERS

Number Systems

- We have studied the following integer number systems in computer
  - Unsigned numbers
  - Sign/magnitude numbers
  - Two's complement numbers
- What about rational numbers?
  - For example, 2.5, -10.04, 0.75 etc.
- Two common notations to represent rational numbers in computer
  - Fixed-point numbers
  - Floating-point numbers

Fixed-Point Numbers

- Fixed point notation has an *implied binary point* between the integer and fraction bits
  - The binary point is not a part of the representation but is implied
  - Example:
    - Fixed-point representation of 6.75 using 4 integer bits and 4 fraction bits:

      01101100

      0110.1100

      $2^2 + 2^1 + 2^{-1} + 2^{-2} = 6.75$
- The number of integer and fraction bits must be agreed upon by those generating and those reading the number
  - There is no way of knowing the existence of the binary point except through agreement of those people interpreting the number

Signed Fixed-Point Numbers

- As with whole numbers, negative fractional numbers can be represented in two ways
  - Sign/magnitude notation
  - Two's complement notation
- Example
  - -2.375 using 8 bits (4 bits each to represent integer and fractional parts)
    - 2.375 = 0010 . 0110
    - Sign/magnitude notation: 1010 0110
    - Two's complement notation:

      1. flip all the bits: 1101 1001
      2. add 1: + 1

      1101 1010
- Addition and subtraction works easily in computer with 2's complement notation like integer addition and subtraction

Example

- Suppose that we have 8 bits to represent a number
  - 4 bits for integer and 4 bits for fraction
- Compute 0.75 + (-0.625)
  - 0.75 = 0000 1100
  - 0.625 = 0000 1010
  - -0.625 in 2's complement form: 1111 0110

0.75	0000 1100
+ - 0.625	1111 0110
0.125	0000 0010

Reflection

- Which representation is better for implementing a DSP algorithm?
  - Integer fixed-point
  - Fractional fixed-point

Integer Fixed-Point Representation

- N-bit fixed point, 2's complement integer representation
  $X=-b_{N-1}\,2^{N-1}+b_{N-2}\,2^{N-2}+\ldots+b_0 2^0$
- Difficult to use due to possible overflow
- In a 16-bit processor, the dynamic range is -32,768 to 32,767.
  - Example
    200 × 350 = 70000, which is an overflow!

Fractional Fixed-Point Representation

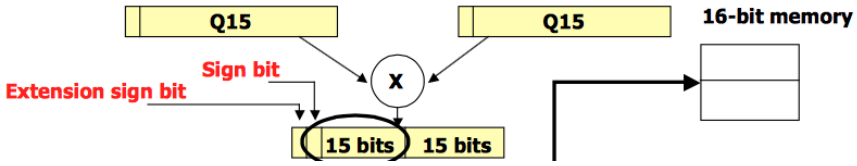
- Also called Q-format
- Fractional representation used extensively for DSP algorithms.
- Fractional number range is between 1 and -1
- Multiplying a fraction by a fraction always results in a fraction and will not produce an overflow (e.g., 0.99 x 0.9999 less than 1)
- Successive additions may cause overflow
- Represent numbers between
  - -1.0 and $1 - 2^{-(N-1)}$, when N is number of bits

Fractional Fixed-Point Representation

- Equivalent to scaling
- Q represents the "Quantity of fractional bits"
- Number following the Q indicates the number of bits that are used for the fraction.
- **Q15** used in 16-bit data type, resolution of the fraction will be 2^–15 or 30.518e–6
  - Q15 means scaling by $1/2^{15}$
  - Q15 means shifting to the right by 15
- Example: how to represent 0.2625 in memory:
  - Method 1 (Truncation): INT[$0.2625*2^{15}$]= INT[8601.6] = 8601 = 0010000110011001
  - Method 2 (Rounding): INT[$0.2625*2^{15}+0.5$]= INT[8602.1] = 8602 = 0010000110011010

Q Format Multiplication

- Product of two Q15 (Q1.15) numbers is Q30 (Q2.30).
- So we must remember that the 32-bit product has *two bits* in front of the binary point.
  - Since NxN multiplication yields 2N-1 result
  - Addition MSB sign extension bit
- Typically, only the most significant 15 bits (plus the sign bit) are stored back into memory, so the *write operation requires a left shift by one*.



Convert between Fixed- and Floating-Point in Cortex-M4

VCVT, VCVTR between floating-point and integer

Converts a value in a register from floating-point to a 32-bit integer.

Syntax

```
VCVT{R}{cond}.Tm.F32 Sd, Sm
VCVT{cond}.F32.Tm Sd, Sm
```

Where:

- *'R'*.
  If R is specified, the operation uses the rounding mode specified by the FPSCR.
  If R is omitted. the operation uses the Round towards Zero rounding mode.
- *'cond'* is an optional condition code, see *Conditional execution on page 65*.
- *'Tm'* is the data type for the operand. It must be one of:
  S32 signed 32-bit value.
  U32 unsigned 32-bit value.
- *'Sd, Sm'* are the destination register and the operand register.

ARM CMSIS-DSP Library

Data Types of Optimized Functions

- 8-bit integers
- 16-bit integers
- 32-bit integers
- 32-bit floating point

Using the library

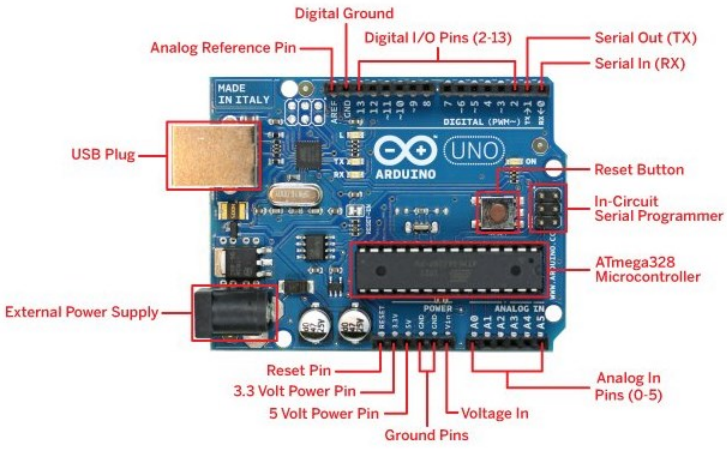
- Include: `arm_math.h`
- Link with:
  - arm_cortexM4lf_math.lib (Cortex-M4, Little endian, Floating Point Unit)
  - arm_cortexM4bf_math.lib (Cortex-M4, Big endian, Floating Point Unit)
  - arm_cortexM4l_math.lib (Cortex-M4, Little endian)
  - arm_cortexM4b_math.lib (Cortex-M4, Big endian)

Implemented Functions

- Basic math functions
- Fast math functions
- Complex math functions
- Filters
- Matrix functions
- Transform functions
- Motor control functions
- Statistical functions
- Support functions
- Interpolation functions

ARDUINO PLATFORM

Arduino UNO



Arduino UNO

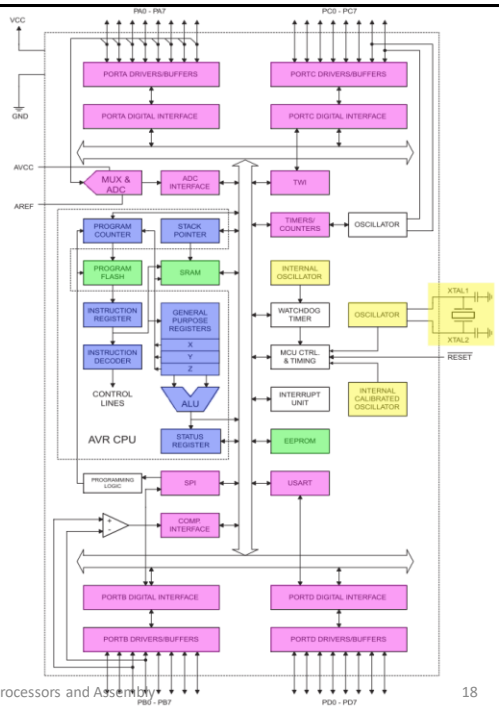
- Micro Controller-based
  - Atmel ATMega328 chip 16 MHz (8 bit microcontroller)
- 14 Digital I/O Pins (6 can be PWM)
- 6 Analog Pins
- Serial TX/RX for USB and serial communications
- I^2C, SPI and Single Wire interfaces
- 32KB Flash Memory
- 2KB SRAM and 1KB EEPROM
- 5 Volts (input 7-12 volts)

ATMega32

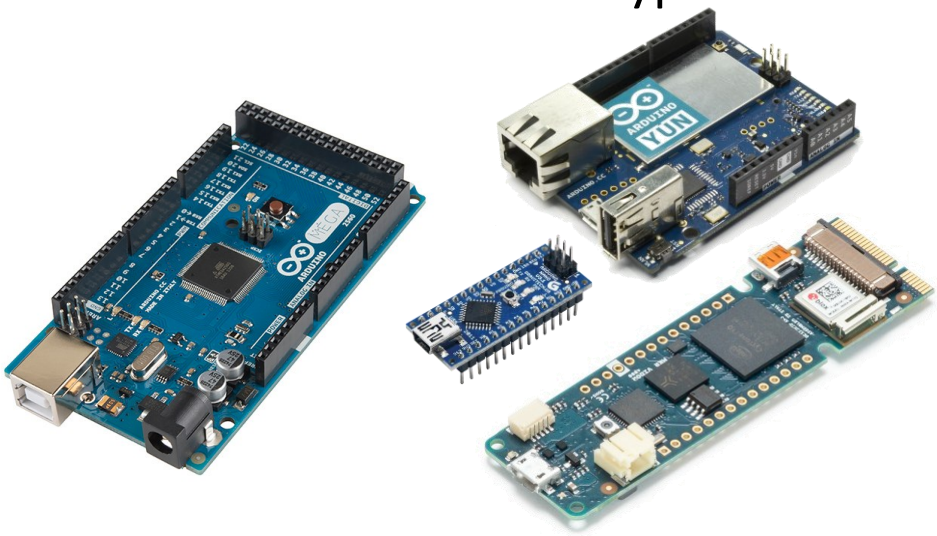
Advanced RISC Architecture
- 131 Instructions (most single-clock cycle)
- 32 x 8 General Purpose Working Registers
- Up to 16 MIPS Throughput at 16 MHz

Non-volatile Program and Data Memory
- 32Kbytes of In-System Self-programmable Flash program memory
- 1024Bytes EEPROM
- 2Kbyte Internal SRAM

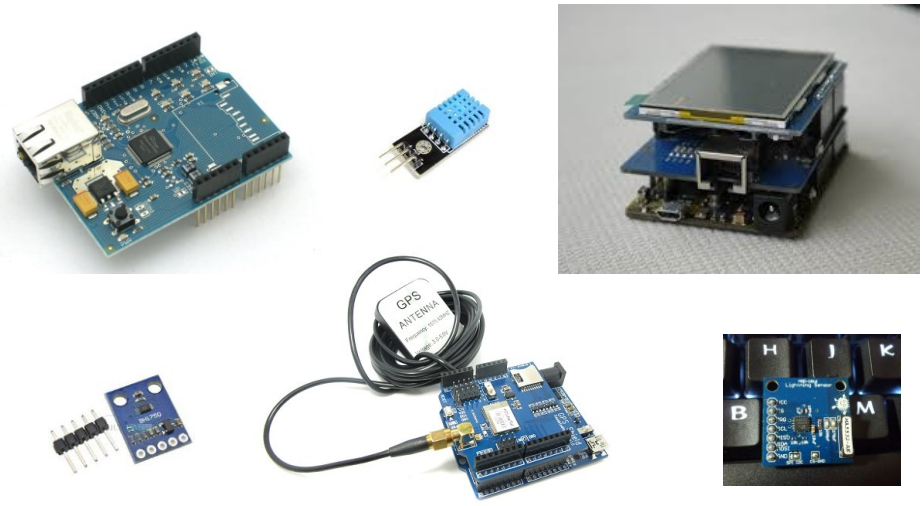
Peripheral Features
- Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
- One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
- Real Time Counter with Separate Oscillator
- Four PWM Channels
- 8-channel, 10-bit ADC
- Two-wire Serial Interface (TWI)
- Programmable Serial USART
- Master/Slave SPI Serial Interface
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator



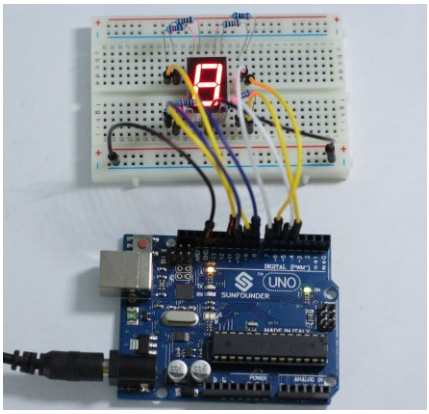
Arduino – Other Types

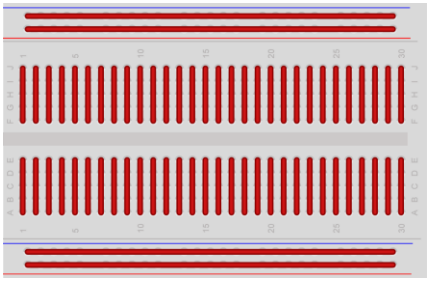


Arduino Shields and Sensors



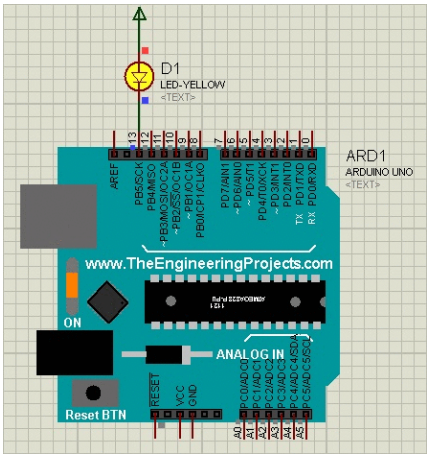
Connecting Devices to Arduino



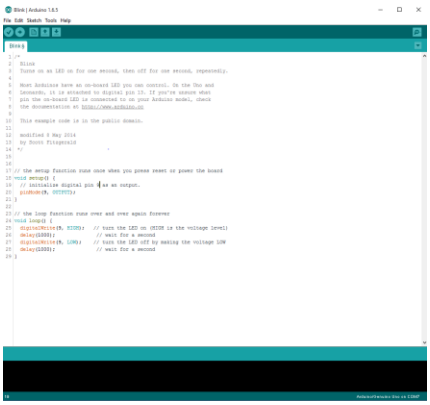


Arduino Simulation

- You can simply simulate the ATmega328 component
- Additional libraries for Proteus include Arduino boards
  - Freely-available online libraries



The Arduino IDE



- The board model (Arduino Uno) and connected port (depends on your PC) Should be set
- Verify
  - Checks code for errors
- Upload
  - Compiles and uploads code to the Arduino I/O board
- Serial Monitor
  - Display serial data being sent from the Arduino board
- Example programs
- Libraries

Structure of an Arduino "sketch"

```
void setup()
{
// put your setup code here, to run once:

}
void loop()
{
 // put your main code here, to run repeatedly:

}
```

NB: A copy of this can be found in File>Examples>1. Basics>BareMinimum

Basic Software Constructs

- Hardware related
  - pinMode(), setup the functions of hardware pins
  - digitalWrite(), set a pin to a digital level : ‘1’ or ‘0’
  - digitalRead(), read the digital level of a pin: ‘1’ or ‘0’
  - delay()
- Software related
  - If-then-else
  - For
  - Switch-case

STM32duino

- Arduino support for STM32 MCUs
- Run your sketches on STM32!