andyjpb@ashurst.eu.org
@databasescaling

Wednesday, 18th April 2013

Writing C For Constrained Systems

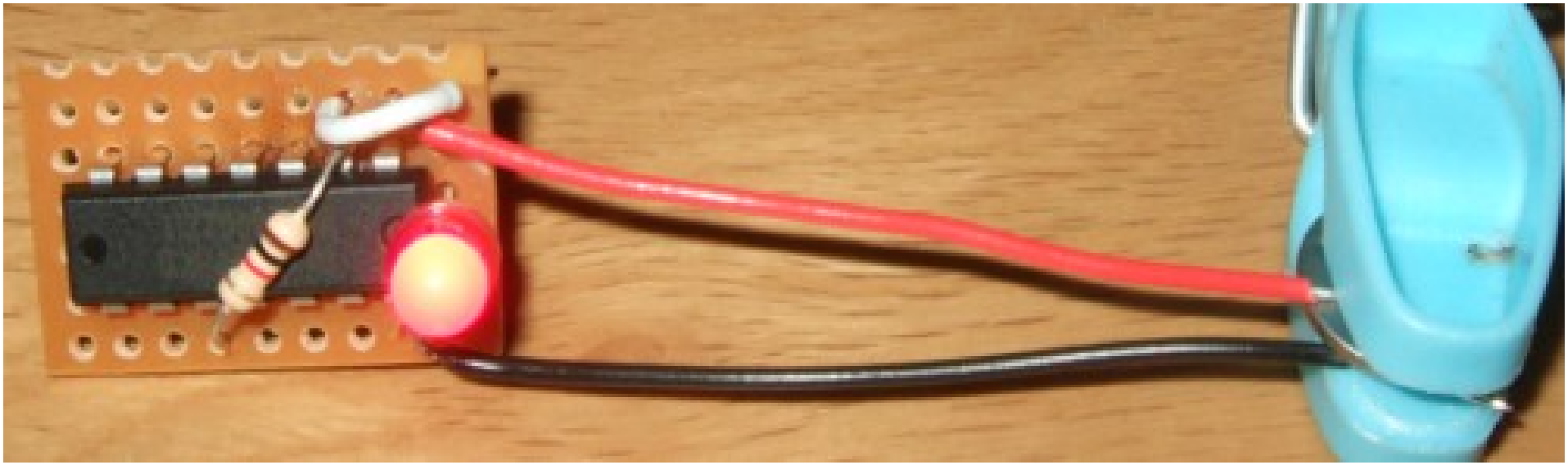
a@jpb.li
@databasescaling

Wednesday, 18th April 2013

Writing C For Constrained Systems
A Morse Code Beacon

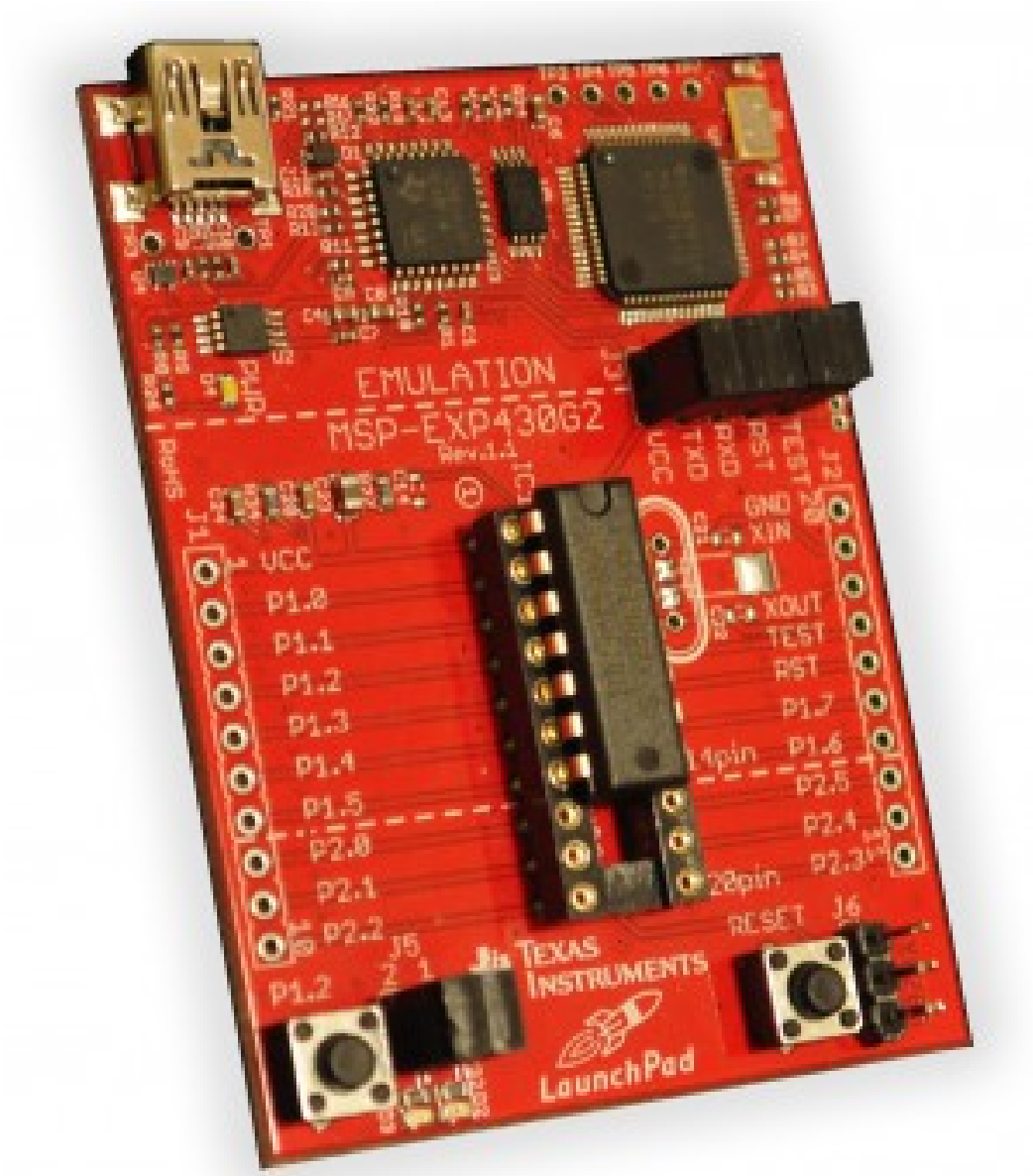
a@jpb.li
@databasescaling

Wednesday, 18th April 2013



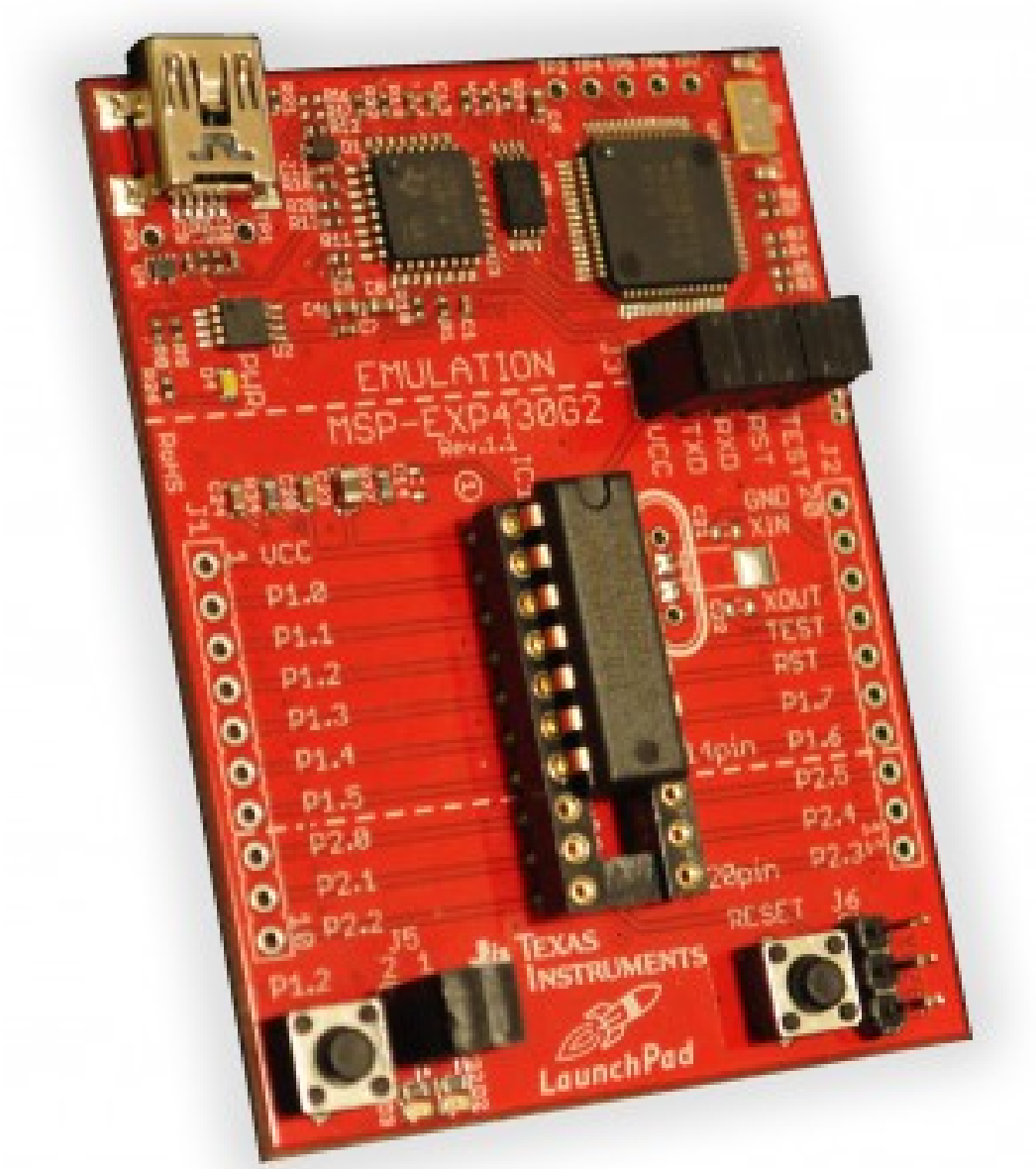
http://thestateofme.files.wordpress.com/2011/09/morse.png

TI Launchpad



http://processors.wiki.ti.com/images/thumb/a/ad/LaunchPad_wireframe.PNG/300px-LaunchPad_wireframe.PNG

TI Launchpad



$4.30

http://processors.wiki.ti.com/images/thumb/a/ad/LaunchPad_wireframe.PNG/300px-LaunchPad_wireframe.PNG

MSP430

Timers

$4.30

http://processors.wiki.ti.com/images/thumb/a/ad/LaunchPad_wireframe.PNG/300px-LaunchPad_wireframe.PNG

MSP430

$4.30

Timers
8 Channel 10-bit ADC

http://processors.wiki.ti.com/images/thumb/a/ad/LaunchPad_wireframe.PNG/300px-LaunchPad_wireframe.PNG

MSP430

$4.30

Timers
8 Channel 10-bit ADC
Comparator

http://processors.wiki.ti.com/images/thumb/a/ad/LaunchPad_wireframe.PNG/300px-LaunchPad_wireframe.PNG

MSP430

$4.30

Timers
8 Channel 10-bit ADC
Comparator
I2C, SPI, UART

http://processors.wiki.ti.com/images/thumb/a/ad/LaunchPad_wireframe.PNG/300px-LaunchPad_wireframe.PNG

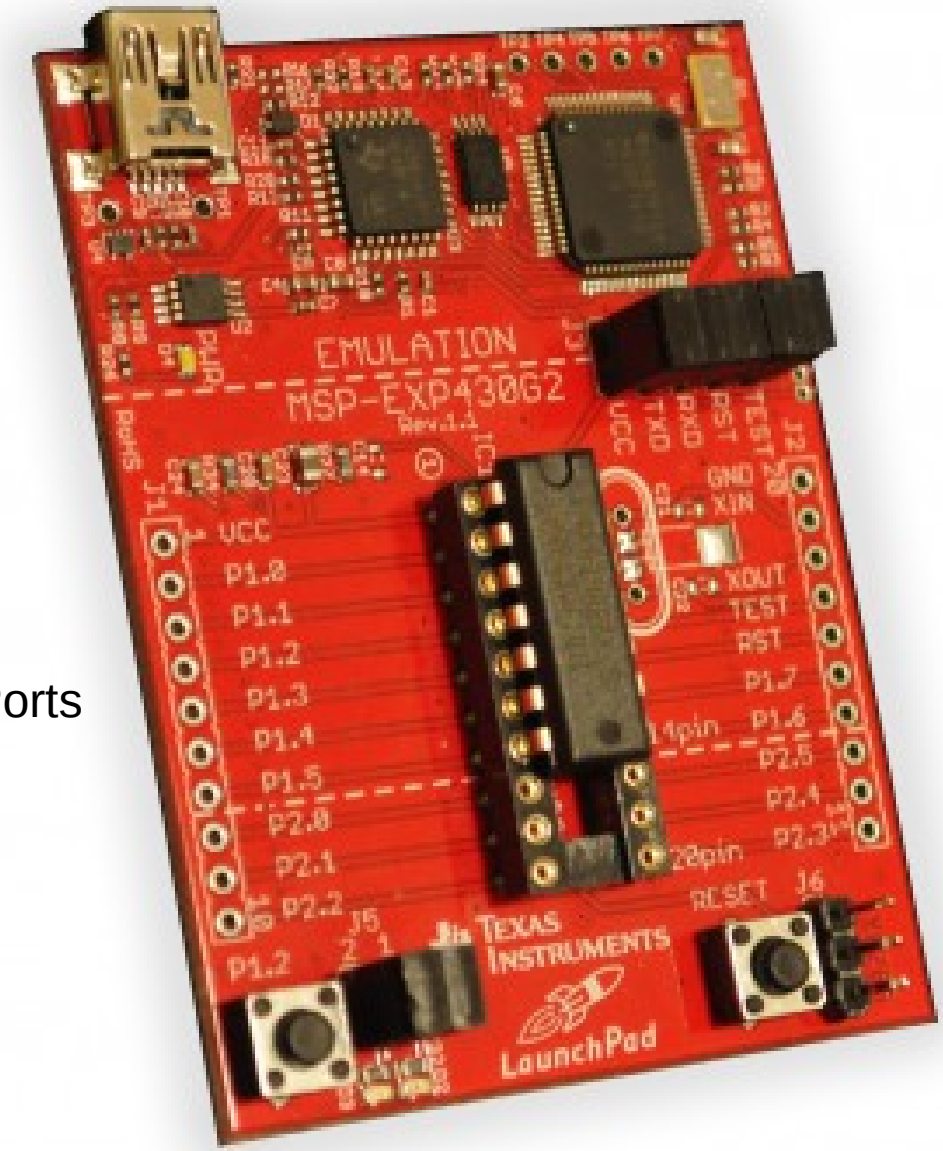
MSP430

$4.30

Timers
8 Channel 10-bit ADC
Comparator
I2C, SPI, UART
General Purpose I/O

http://processors.wiki.ti.com/images/thumb/a/ad/LaunchPad_wireframe.PNG/300px-LaunchPad_wireframe.PNG

MSP430



$4.30

Timers
8 Channel 10-bit ADC
Comparator
I2C, SPI, UART
General Purpose I/O
Capacitive Sense I/O Ports

http://processors.wiki.ti.com/images/thumb/a/ad/LaunchPad_wireframe.PNG/300px-LaunchPad_wireframe.PNG

MSP430

http://processors.wiki.ti.com/images/thumb/a/ad/LaunchPad_wireframe.PNG/300px-LaunchPad_wireframe.PNG

MSP430

16 MHz

16 KB Flash

256 Bytes RAM

http://processors.wiki.ti.com/images/thumb/a/ad/LaunchPad_wireframe.PNG/300px-LaunchPad_wireframe.PNG

MSP430

16 MHz

16 KB Flash

512 Bytes RAM

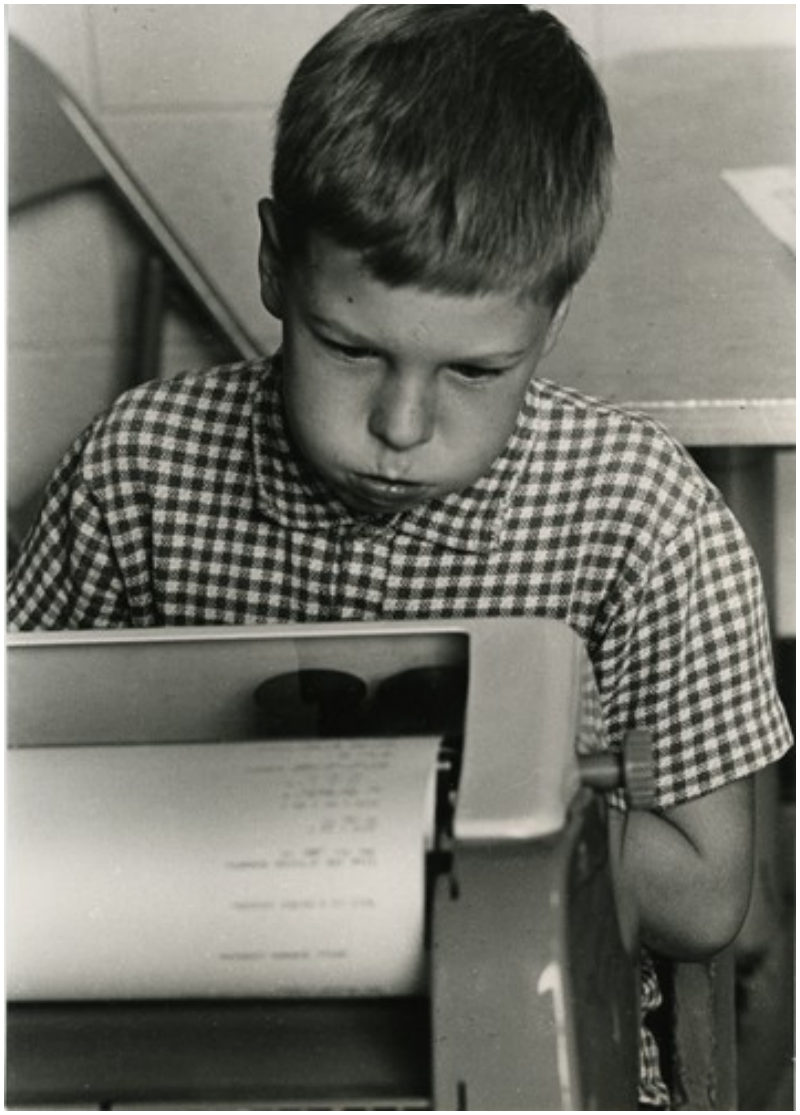
16 MHz

16 KB Flash

256 Bytes RAM

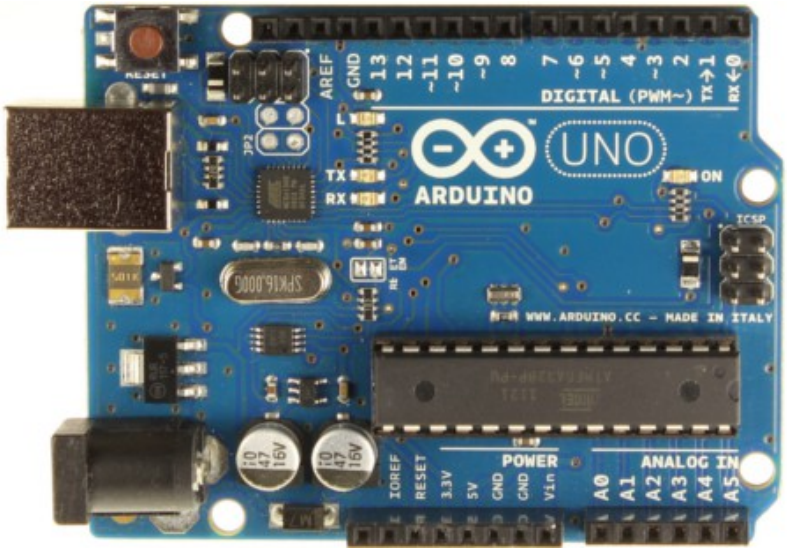
http://processors.wiki.ti.com/images/thumb/a/ad/LaunchPad_wireframe.PNG/300px-LaunchPad_wireframe.PNG

C



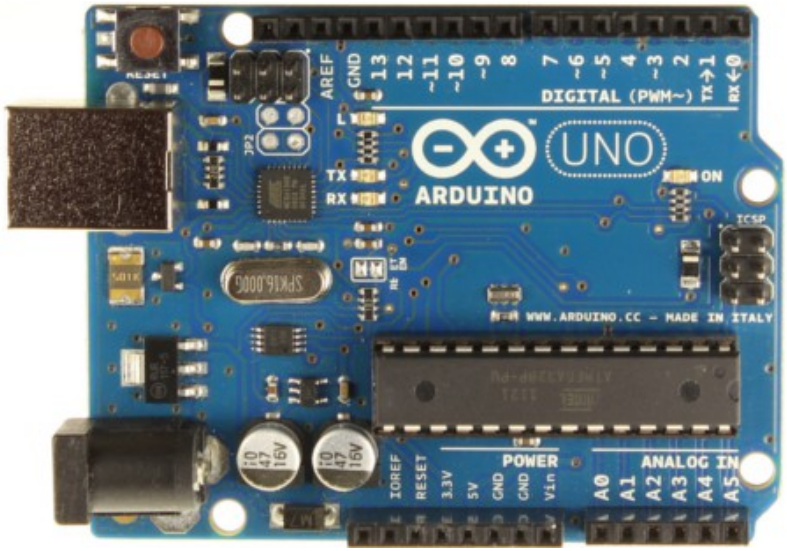
http://archive.computerhistory.org/resources/still-image/DEC/PDP-8/dec.boy_at_teletype.c1965.102627494.lg.jpg

Arduino Uno



http://arduino.cc/en/uploads/Main/ArduinoUno_R3_Front_450px.jpg

ATMega 328



16 MHz

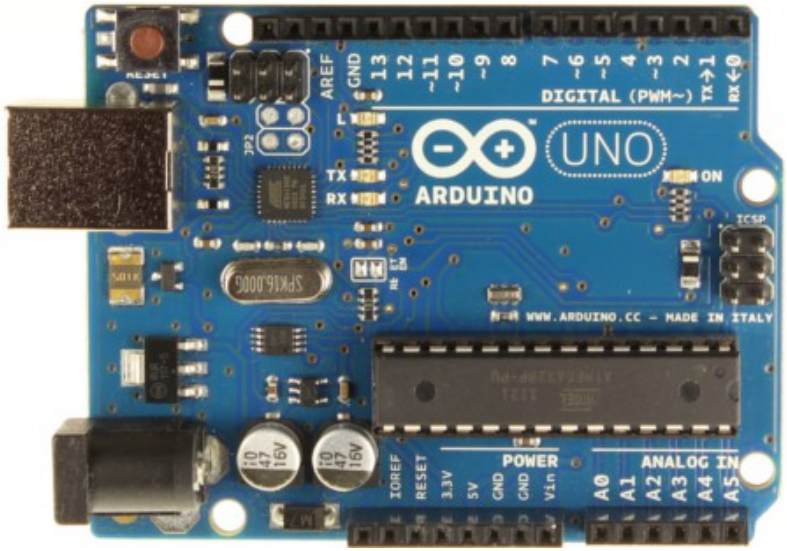
32 KB Flash

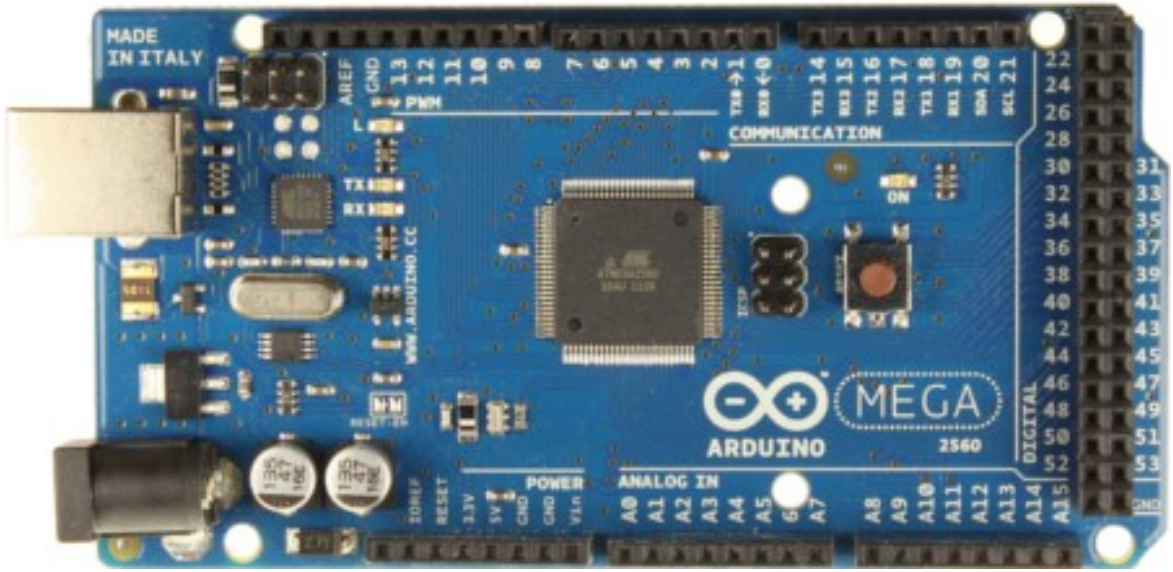
2 KB RAM

1 KB EEPROM

http://arduino.cc/en/uploads/Main/ArduinoUno_R3_Front_450px.jpg

Arduino Mega 2560





16 MHz

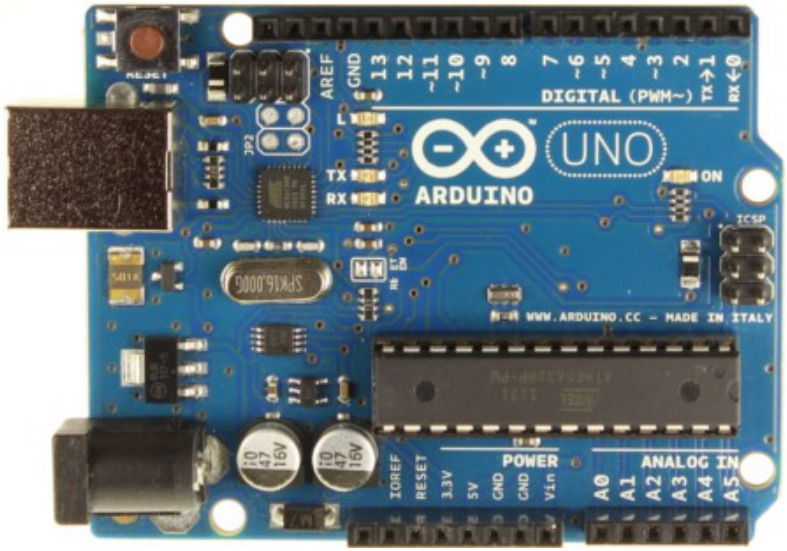
32 KB Flash

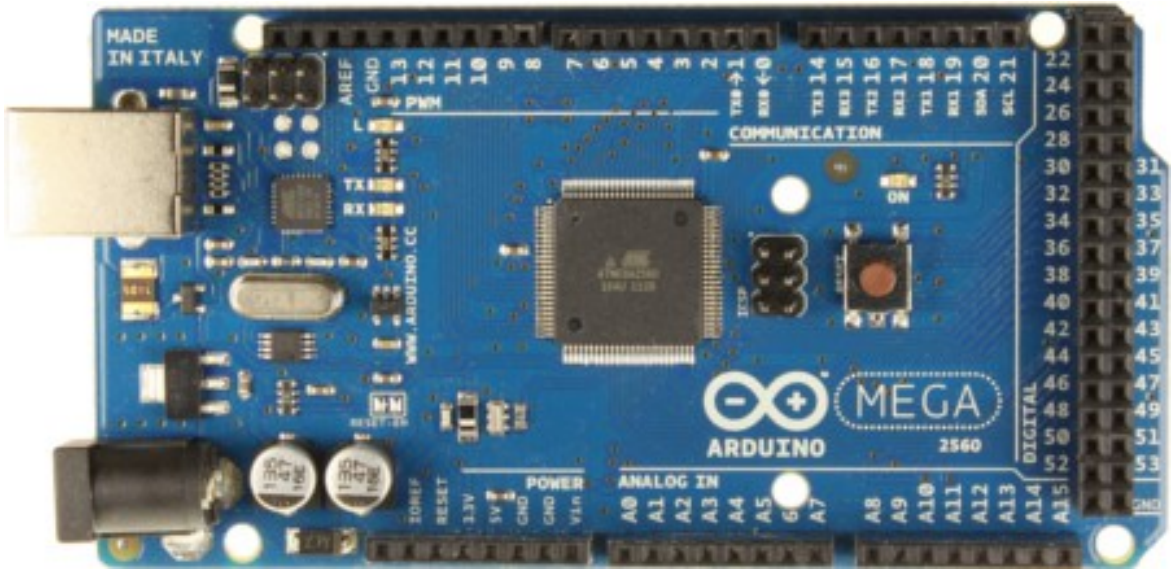
2 KB RAM

1 KB EEPROM

http://arduino.cc/en/uploads/Main/ArduinoMega2560_R3_Front_450px.jpg
http://arduino.cc/en/uploads/Main/ArduinoUno_R3_Front_450px.jpg

Arduino Mega 2560

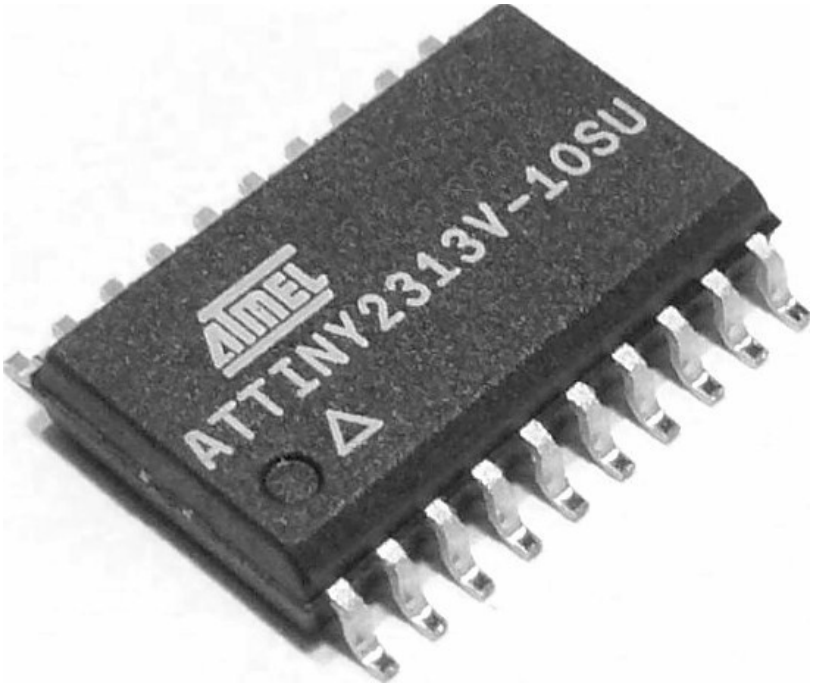




16	MHz		16	MHz	
32	KB	Flash	256	KB	Flash
2	KB	RAM	8	KB	RAM
1	KB	EEPROM	4	KB	EEPROM

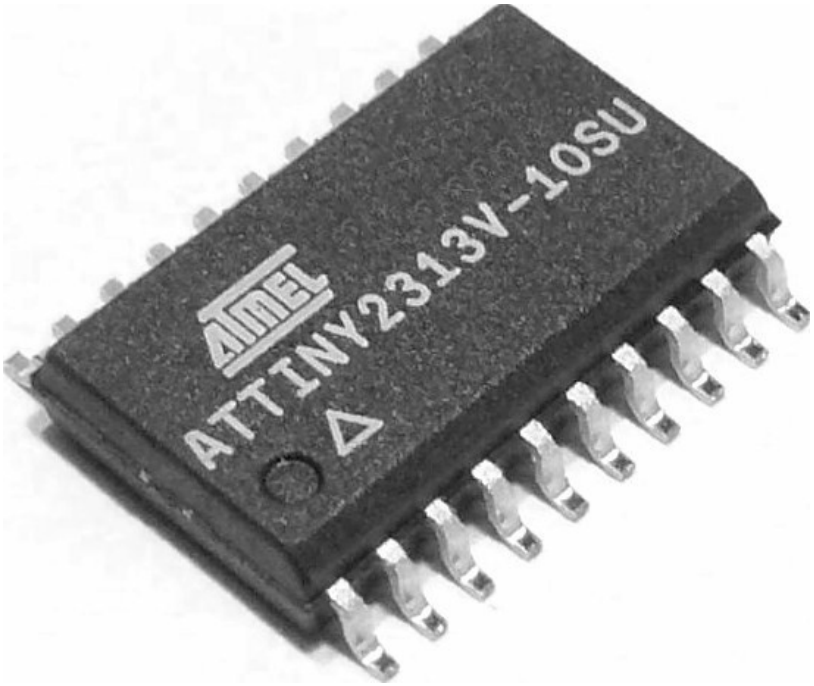
http://arduino.cc/en/uploads/Main/ArduinoMega2560_R3_Front_450px.jpg
http://arduino.cc/en/uploads/Main/ArduinoUno_R3_Front_450px.jpg

AT Tiny 2313



http://www.sklep.semics.pl/bilder/Attiny.2313v-10su.jpg

AT Tiny 2313

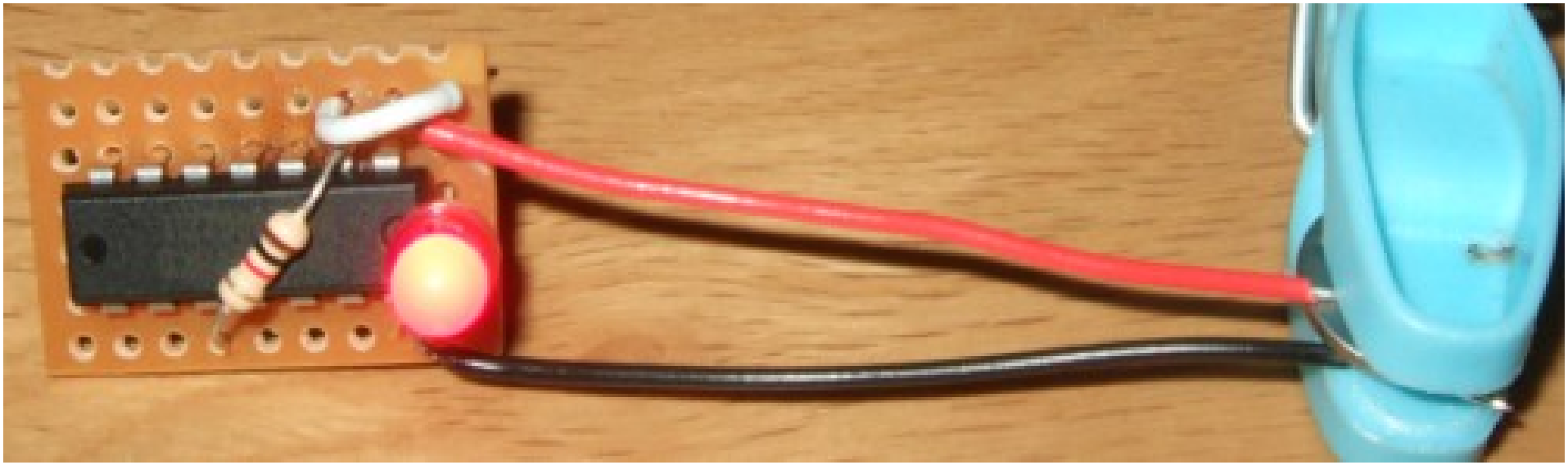


20	MHz	
2	KB	Flash
128	Bytes	RAM
128	Bytes	EEPROM

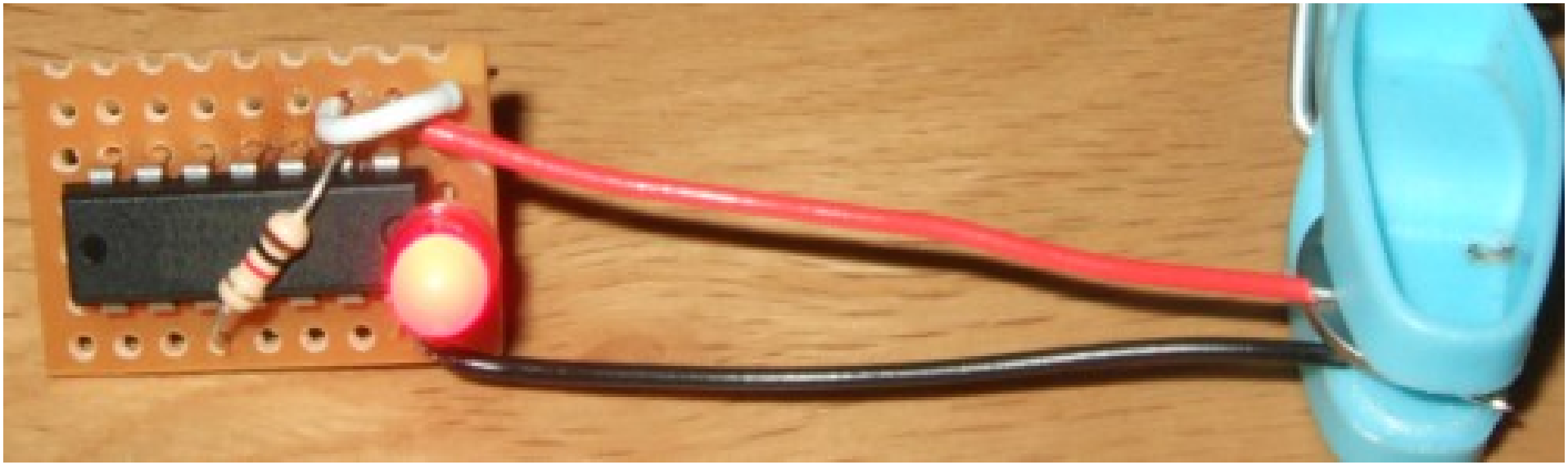
http://www.sklep.semics.pl/bilder/Attiny.2313v-10su.jpg

C

http://cm.bell-labs.com/cm/cs/who/dmr/kd14.jpg



http://thestateofme.files.wordpress.com/2011/09/morse.png



16 MHz 16 KB Flash 256 Bytes RAM

http://thestateofme.files.wordpress.com/2011/09/morse.png

main.c

```
//******************************************************************
//  Based on MSP430F20xx Demo - Software Toggle P1.0
//
//  Description; Sends characters by morse
//  ACLK = n/a, MCLK = SMCLK = default DCO
//
//                MSP430F20xx
//             -----------------
//         /|\|                 XIN|-
//          | |                    |
//          --|RST             XOUT|-
//            |                    |
//            |             P1.0|-->LED
//
//  Chris Swan
//  September 2011
//******************************************************************
```

https://github.com/cpswan/TI-LaunchPad/blob/master/morse/main.c

Examples

main.c

```
void dot(void)
{
    P1OUT = 0x01; // LED on

    shortpause();

    P1OUT = 0x00; // LED off

    shortpause();
}
```

https://github.com/cpswan/TI-LaunchPad/blob/master/morse/main.c

main.c

```
void dash(void)
{

    P1OUT = 0x01; // LED on

    shortpause(); // dash is three times longer than dot
    shortpause();
    shortpause();

    P1OUT = 0x00; // LED off

    shortpause();
}
```

https://github.com/cpswan/TI-LaunchPad/blob/master/morse/main.c

main.c

```
void shortpause(void)
{
    volatile unsigned int i;

    i = 25000; // Delay
    do (i--);
    while (i != 0);
}
```

https://github.com/cpswan/TI-LaunchPad/blob/master/morse/main.c

main.c

```
void tap_A(void)
{
    dot();
    dash();
    shortbreak();
}
```

https://github.com/cpswan/TI-LaunchPad/blob/master/morse/main.c

main.c

```
void tap_A(void)
{
    dot();
    dash();
    shortbreak();
}


void tap_B(void)
{
    dash();
    dot();
    dot();
    dot();
    shortbreak();
}
```

https://github.com/cpswan/TI-LaunchPad/blob/master/morse/main.c

main.c

```
void tap_A(void)
{
    dot();
    dash();
    shortbreak();
}
```

```
void tap_B(void)
{
    dash();
    dot();
    dot();
    dot();
    shortbreak();
}
```

```
void tap_C(void)
{
    dash();
    dot();
    dash();
    dot();
    shortbreak();
}
```

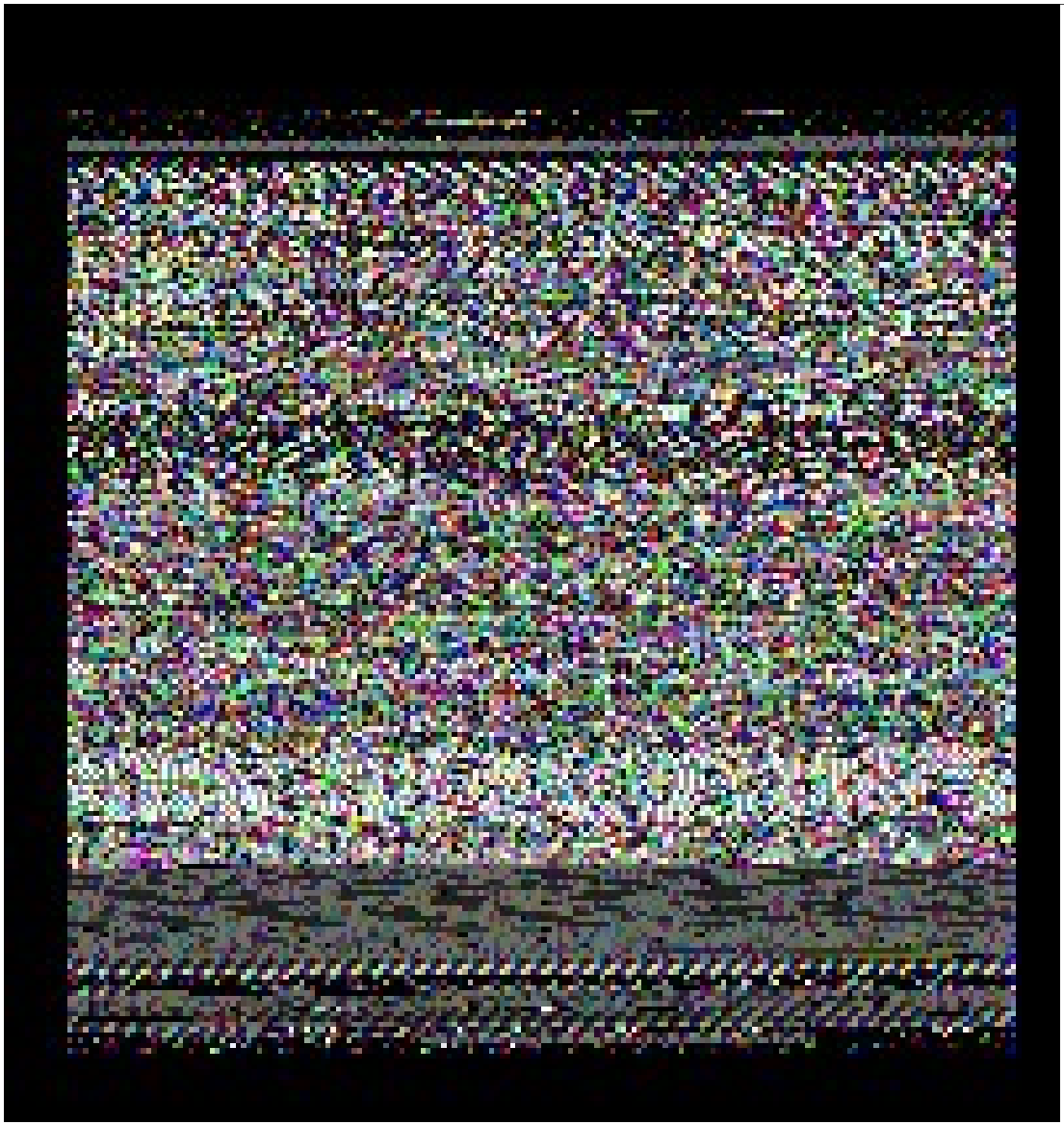
https://github.com/cpswan/TI-LaunchPad/blob/master/morse/main.c

```
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer
    P1DIR |= 0x01; // Set P1.0 to output direction

    for (;;)
    {
        tap_T();
        tap_H();
        tap_E();
        longbreak();
        tap_Q();
        tap_U();
        tap_I();
        tap_C();
        tap_K();
        longbreak();
        tap_B();
        tap_R();
        tap_O();
        tap_W();
        tap_N();
        longbreak();
```

https://github.com/cpswan/TI-LaunchPad/blob/master/morse/main.c

```
$ gcc -c main.c
```

TEXT

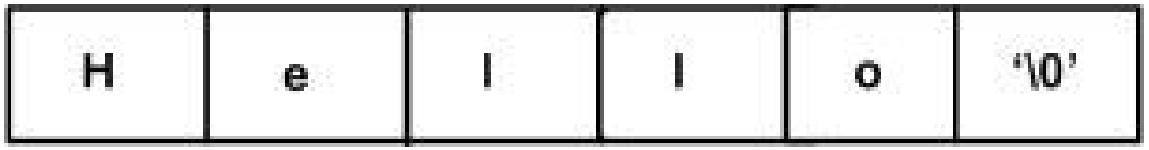


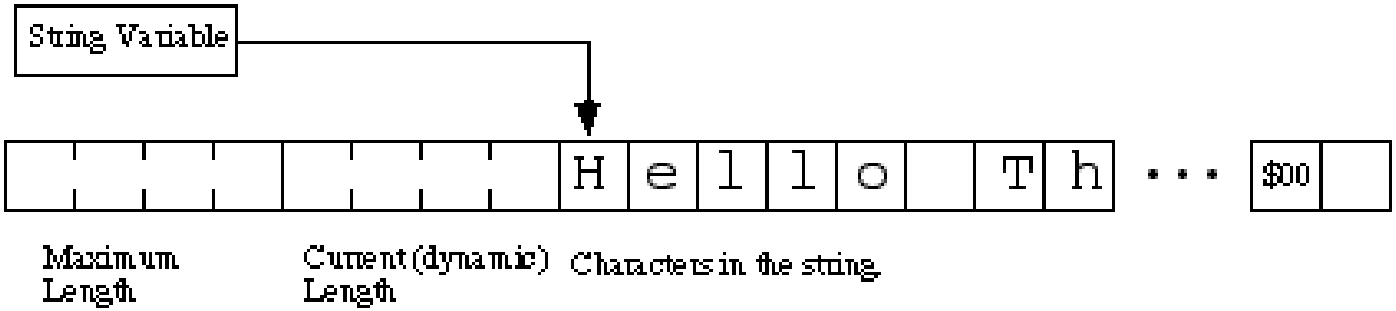
http://nullprogram.com/img/pngarch/bin-ls.png

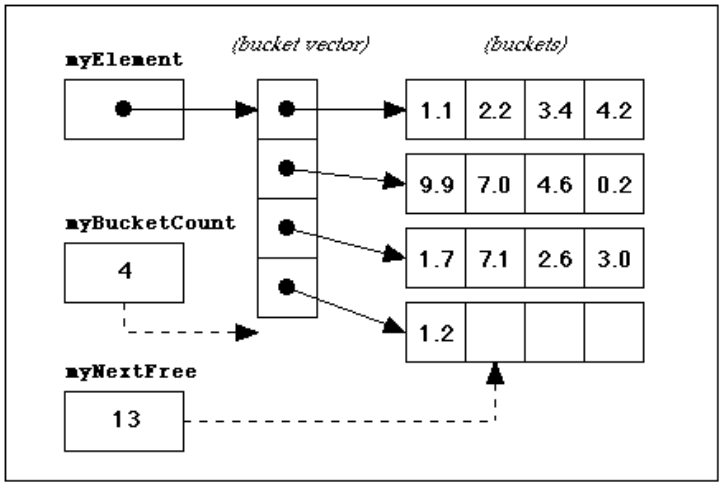
TEXT

```
void tap_A(void)
{
    dot();
    dash();
    shortbreak();
}
```

DATA







http://www.tutorialspoint.com/images/string_representation.jpg

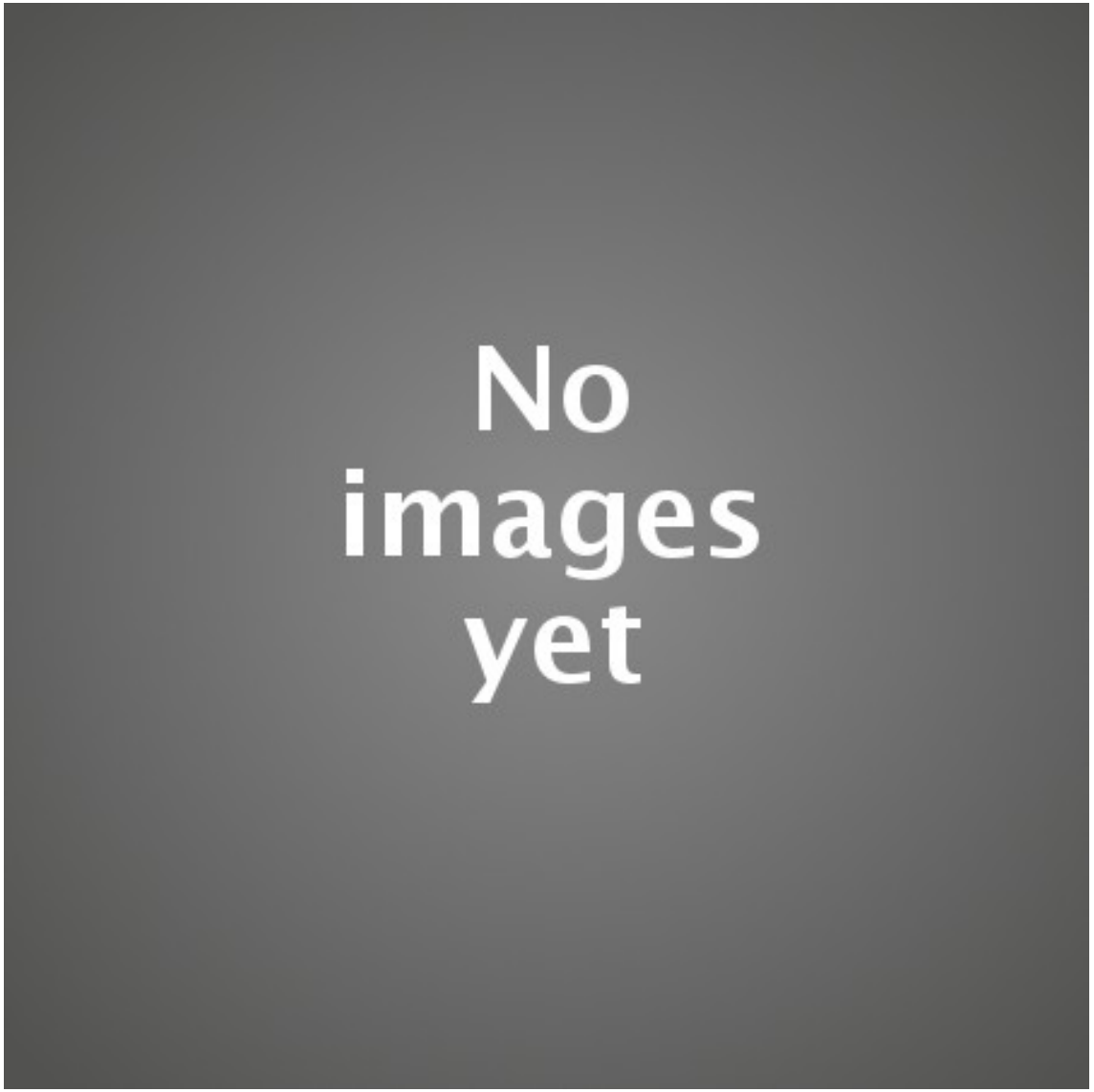
http://www.csse.monash.edu.au/~damian/Idioms/Topics/01.1.Array/html/ArrayInternals.gif

http://www.plantation-productions.com/Webster/www.artofasm.com/AoAExtra/HLAStrs-2.gif

DATA

```
char prompt[] = “WELCOME> “;


char greeting[] = “Hello World!“;


byte command[] = { start_byte, cmd_type, cmd_id, serial_number,
                   data_length_1, data_length_2, data_byte_1,
                   data_byte_2, ., end_byte };
```

BSS



http://commons.esipfed.org/sites/all/modules/contrib/media_gallery/images/empty_gallery.png

BSS

```
char input_buffer[20];

char flag;
```

```
$ gcc -c main.c
$ readelf -t main.o
```

```
$ gcc -c main.c
$ readelf -t main.o
There are 12 section headers, starting at offset 0x938:

Section Headers:
  [Nr] Name
       Type            Address          Offset            Link
       Size            EntSize          Info              Align
       Flags
```

```
$ gcc -c main.c
$ readelf -t main.o
There are 12 section headers, starting at offset 0x938:

Section Headers:
  [Nr] Name
       Type            Address          Offset            Link
       Size            EntSize          Info              Align
       Flags
```

```
$ gcc -c main.c
$ readelf -t main.o

[ 1] .text
     PROGBITS         0000000000000000  0000000000000040  0
     000000000000045d 0000000000000000  0                 4
     [0000000000000006]: ALLOC, EXEC
```

```
$ gcc -c main.c
$ readelf -t main.o

[ 1] .text
     PROGBITS         0000000000000000  0000000000000040  0
     000000000000045d 0000000000000000  0                 4
     [0000000000000006]: ALLOC, EXEC
```

```
$ gcc -c main.c
$ readelf -t main.o

[ 1] .text
     PROGBITS         0000000000000000  0000000000000040  0
     000000000000045d 0000000000000000  0                 4
     [0000000000000006]: ALLOC, EXEC
```

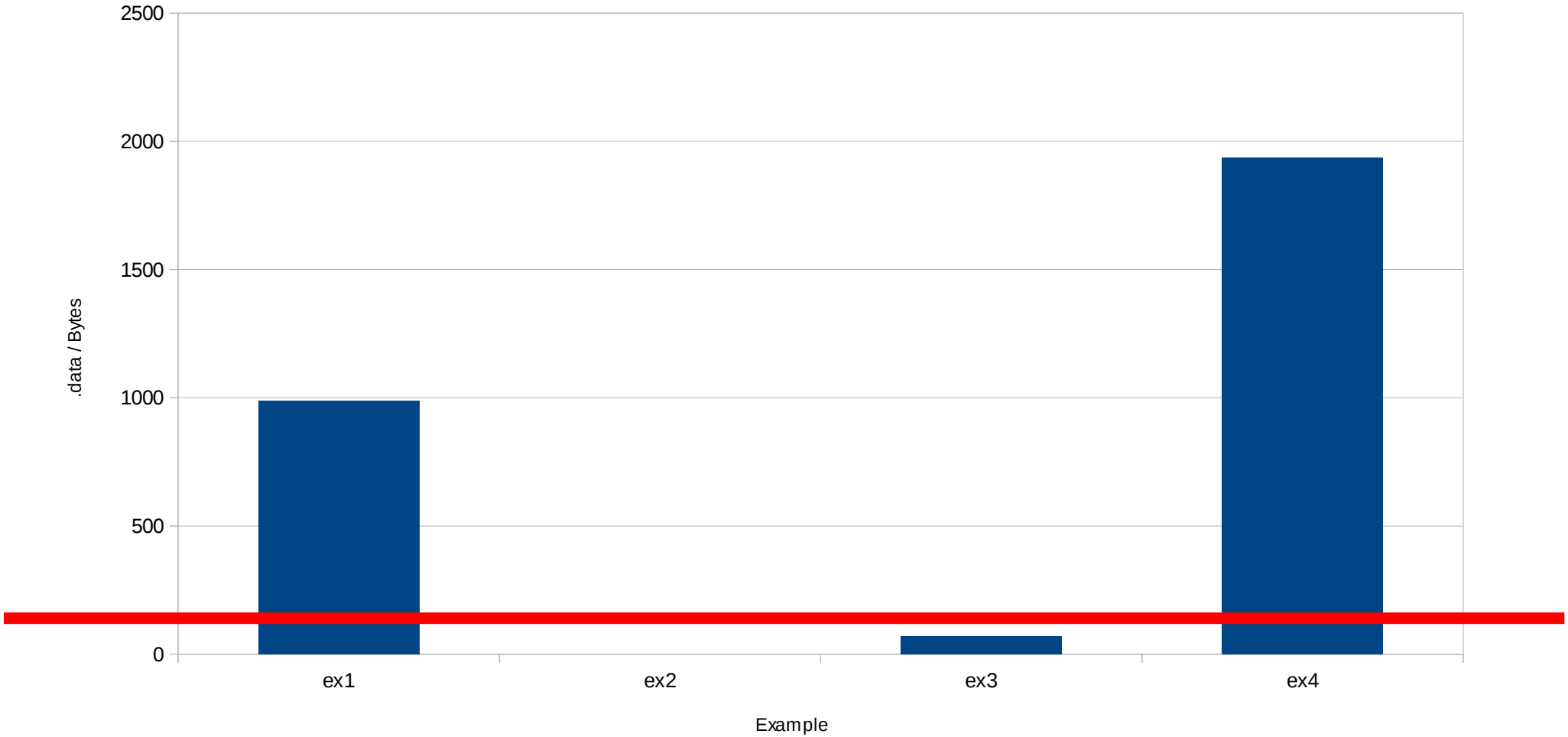
1,117 Bytes

```
$ gcc -c main.c
$ readelf -t main.o

[ 3] .data
     PROGBITS         0000000000000000  00000000000004a0  0
     0000000000000000 0000000000000000  0                 4
     [0000000000000003]: WRITE, ALLOC
```

```
$ gcc -c main.c
$ readelf -t main.o

[ 3] .data
     PROGBITS         0000000000000000  00000000000004a0  0
     0000000000000000 0000000000000000  0                 4
     [0000000000000003]: WRITE, ALLOC
```

```
$ gcc -c main.c
$ readelf -t main.o

[ 4] .bss
     NOBITS           0000000000000000  00000000000004a0  0
     0000000000000000 0000000000000000  0                 4
     [0000000000000003]: WRITE, ALLOC
```

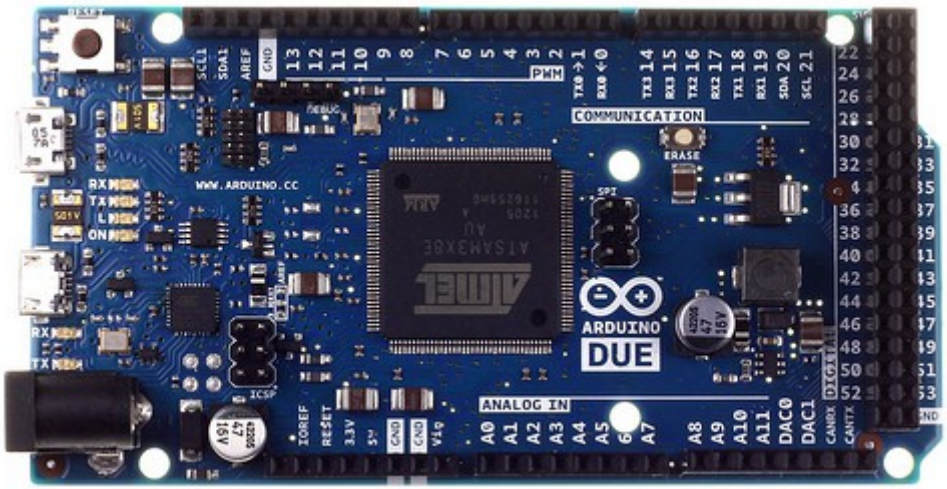
```
$ gcc -c main.c
$ readelf -t main.o

[ 4] .bss
     NOBITS          0000000000000000  00000000000004a0  0
     0000000000000000 0000000000000000  0                 4
     [0000000000000003]: WRITE, ALLOC
```


2500
2000
1500
1000
500
0
.data / Bytes
ex1
ex2
ex3
ex4
Example

- How do we fit this into an AT Tiny?
- Benchmark the tap_* procedures only
- Put the message in RAM
- Put the message in EEPROM

?

Arduino Due



The Due has a 32-bit ARM core that can outperform typical 8-bit microcontroller boards. The most significant differences are:

A 32-bit core, that allows operations on 4 bytes wide data within a single CPU clock. (for more information look int type page).

CPU Clock at 84Mhz.

96 KBytes of SRAM.

512 KBytes of Flash memory for code.

a DMA controller, that can relieve the CPU from doing memory intensive tasks.

http://arduino.cc/en/uploads/Main/ArduinoDue_Front_450px.jpg

main.c

```
void longbreak(void)
{
    shortbreak();
    shortbreak();
}
```

https://github.com/cpswan/TI-LaunchPad/blob/master/morse/main.c

- Ex1: text: 0x33c, data: 0xf0, bss 0x8
- Ex2: text: 60c, data: 0x10, bss: 0x8
- Ex3: text: 35c data: 2c, bss: 0x4
- Ex4: text: 0x4fc data: 0x10 bss: 0x8

- Ex1.o: text: 0x187, data: 0xd0 + 0x270, rodata: 9c, bs: 0
- Ex2.o: text: 0x462, data: 0 bss: 0
- Ex3.o: text: 0x1b2, data: 1a,rodata: 2c, bss: 0
- Ex4.o: text: 0x34c, data: 0, rodata 208 + 588 bss:0