Microcontroller Basics

Gabe Cohn

CSE 599U – February 6, 2012

www.gabeacohn.com/teaching/micro

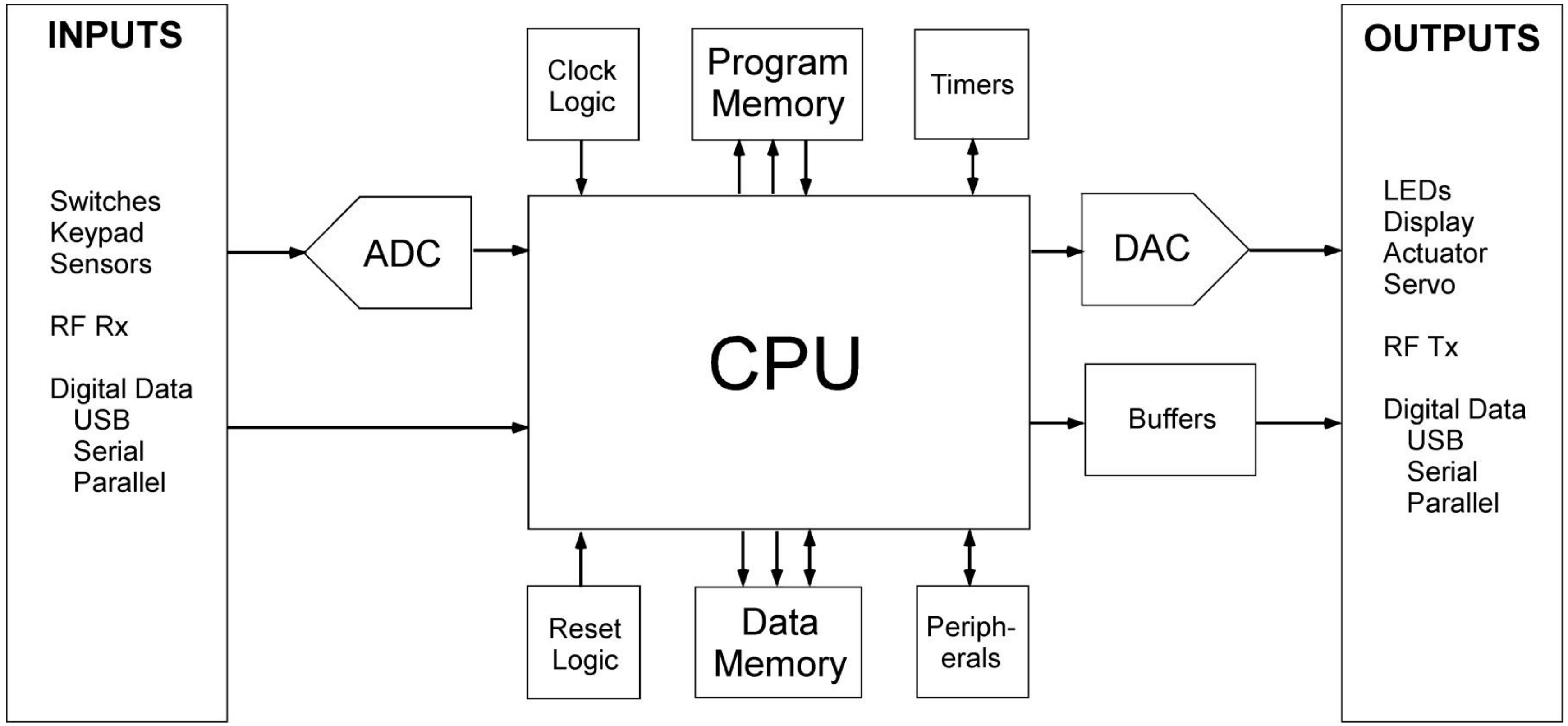
Outline

- Overview of Embedded Systems
- What is a Microcontroller?
- Microcontroller Features
- Common Microcontrollers
- Choosing a Microcontroller
- Types of Embedded Systems
- Tutorials (Phidgets, Arduino, MSP430)

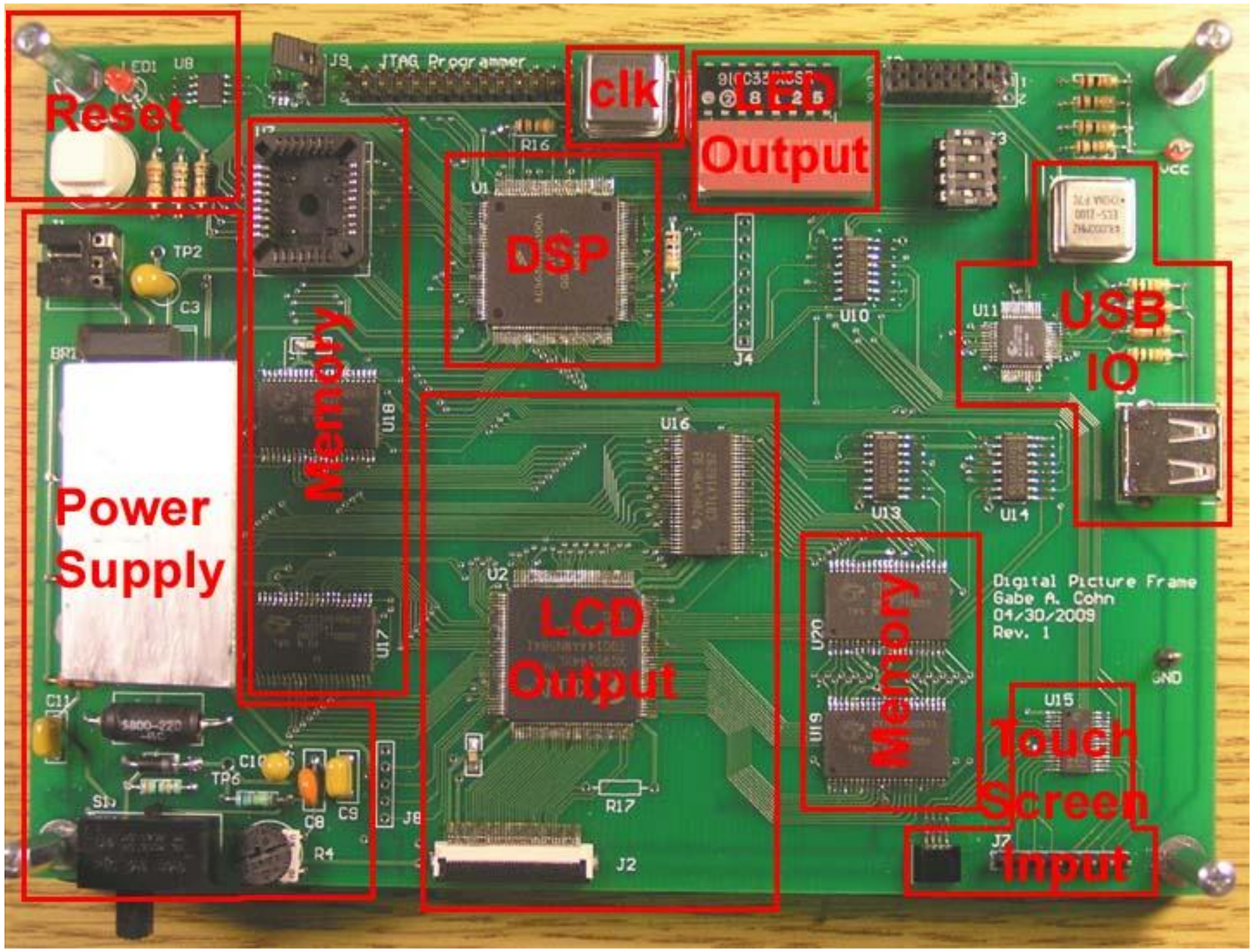
Overview of Embedded Systems

- Minimal computation, simple software (no/simple OS)
- Low power (typically battery powered)
- Event Driven Design
- Mostly IO (Inputs and Outputs)
  - Sensors, Switches, Keypad
  - Displays, LEDs
  - Actuators, Servos
  - Data communication (wired or wireless)
- Data Conversion
  - Analog-to-Digital, Digital-to-Analog

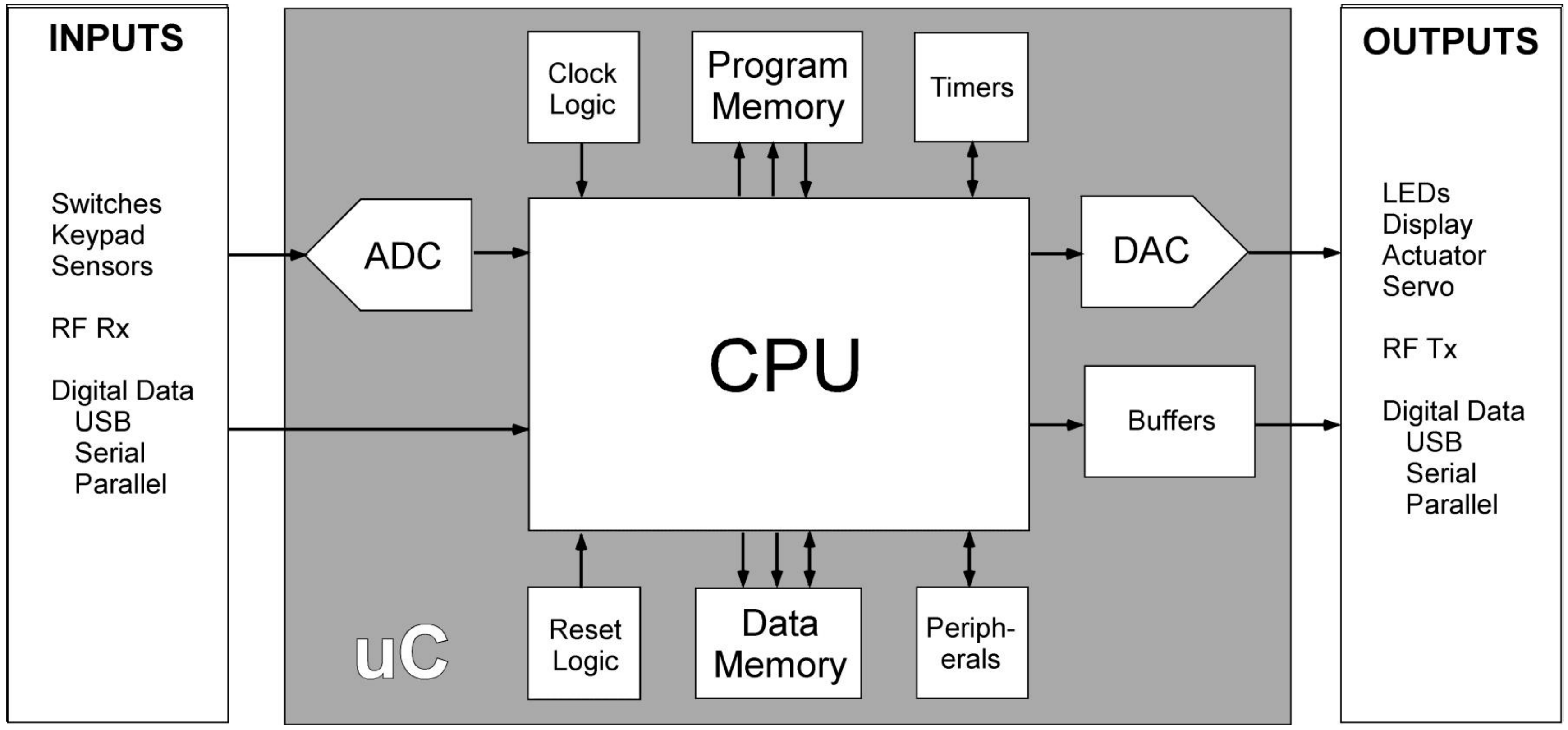
Generic Embedded Systems



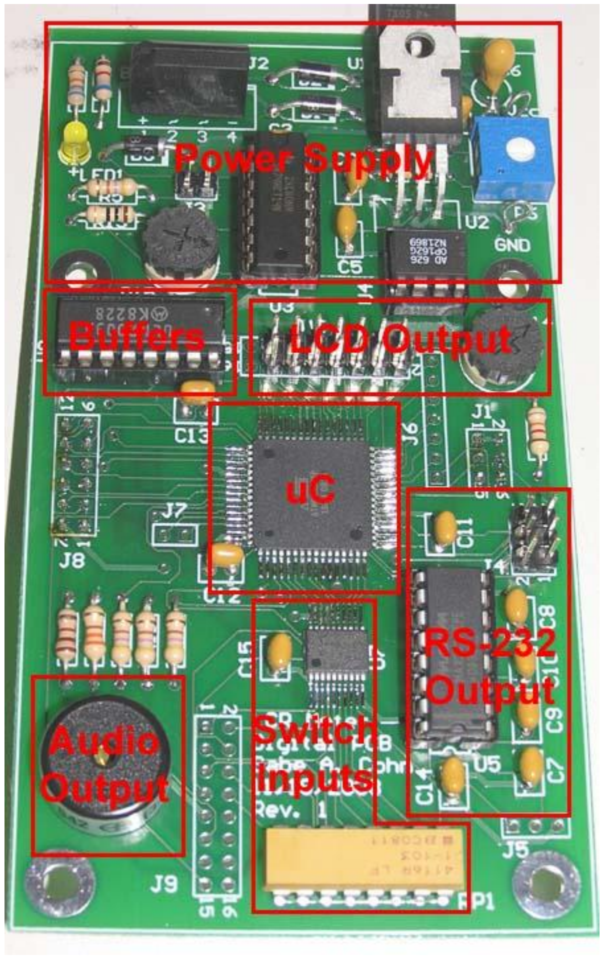
Example: Using Discrete Components



What is a Microcontroller?



Example: Using A Microcontroller

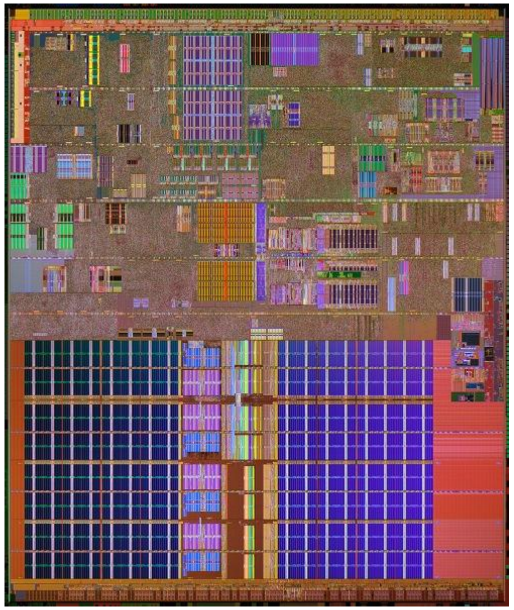


Microcontroller Features

- CPU
- Program and Data Memory (ROM and RAM)
- Reset and Oscillator Circuitry
- Timers
- Data Converters (ADC, DAC)
- Buffered GPIOs
- Simple Peripheral Interface
- Reduced System Size, Complexity and Cost

CPU

- Small ALU (8-bit typical)
- RISC
- Harvard Architecture (separate program and data memory)
- Pipelined Load-Store Architecture
- Lower clock speeds (8-32 MHz)
- Optimized for low-level compilers like C
- Typically no OS is used (sometimes RTOS)



Memory

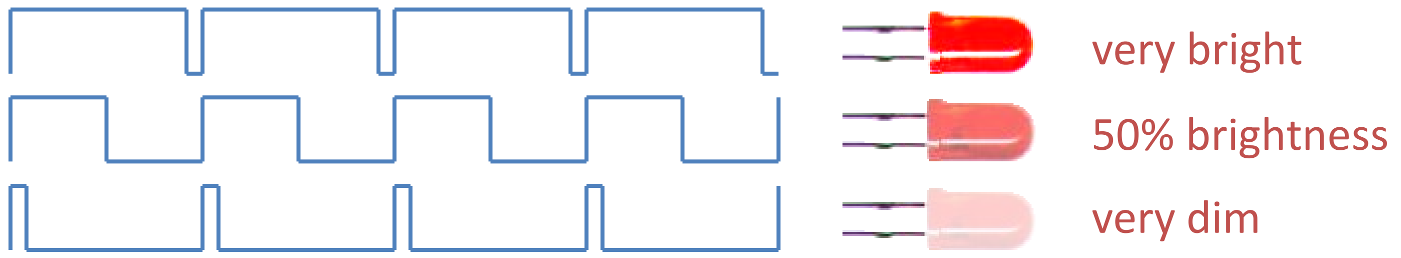
- On-chip RAM and ROM
- No external access to address and data buses
- Need a "programmer" to program the code into the ROM (typically Flash these days)
- Size range 10s of bytes to 100s of KB
  - main difference in price between similar products

Reset and Oscillator Circuitry

- Reset
  - Internal or External
  - Watchdog Timer (WDT)
  - Brownout Reset (BOR)
- Oscillator
  - Several sources to choose from
  - Internal or External
  - PLL and clock frequency adjustment

Timers

- Typically several different timers
  - Real-Time Clock (RTC)
  - Watchdog Timer (WDT)
  - Pulse Width Modulation (PWM) output



- Event based notification (Interrupts)
  - Allows CPU to focus on foreground tasks
  - Useful for input-based events
  - Useful for wake-up from sleep

Data Converters

- Analog-to-Digital Converter (ADC) [very common]
  - For digitizing analog inputs
    - important for ratiometric sensors
  - Several channels
  - Several different types
  - Comparators and other analog circuitry
- Digital-to-Analog Converter (DAC) [uncommon]
  - For producing analog outputs
  - Several different types

General Purpose Input/Outputs (GPIOs)

- Many General Purpose Analog/Digital IOs
  - Buffered to drive typical embedded loads (~20 mA)
  - Multiplexed for several functions
  - Switchable internal pull-up resistors
  - Edge detection
  - Schmitt trigger inputs on some
  - main difference in price between similar products

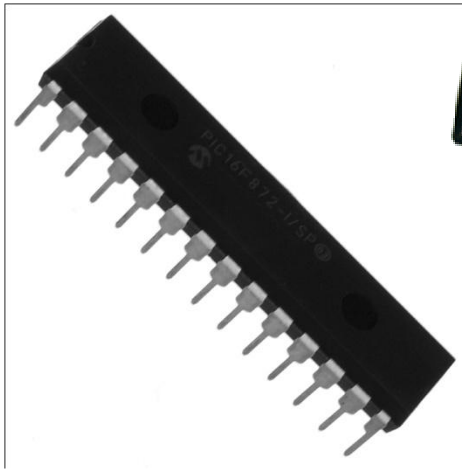
Simple Peripheral Interface

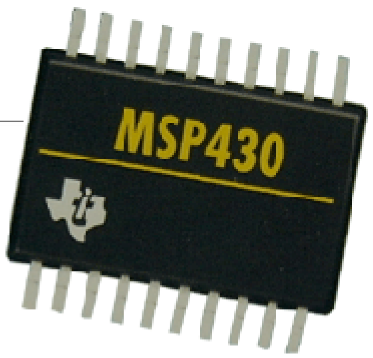
- Serial (Sync/Async, SPI, I^2C)
- CAN bus (automotive)
- LED and LCD controllers
- Ethernet, USB, and Video controllers
- DMA, DRAM, SDRAM controllers
- Host Processor Interface, External Memory Bus

Common Microcontrollers

Family	Manufacturer	Word Size*	Common Uses
ARM	Various	32-bit	Consumer Electronics
AVR	Atmel	8-bit	
PIC	Microchip	8-bit	Hobbyist
MSP430	TI	16-bit	Low Power
8051, 8048	Intel	8-bit	Legacy
6805, 6808, 6811	Motorola/Freescale	8-bit	Legacy

* Many of these µCs now come in wider bus architectures as well



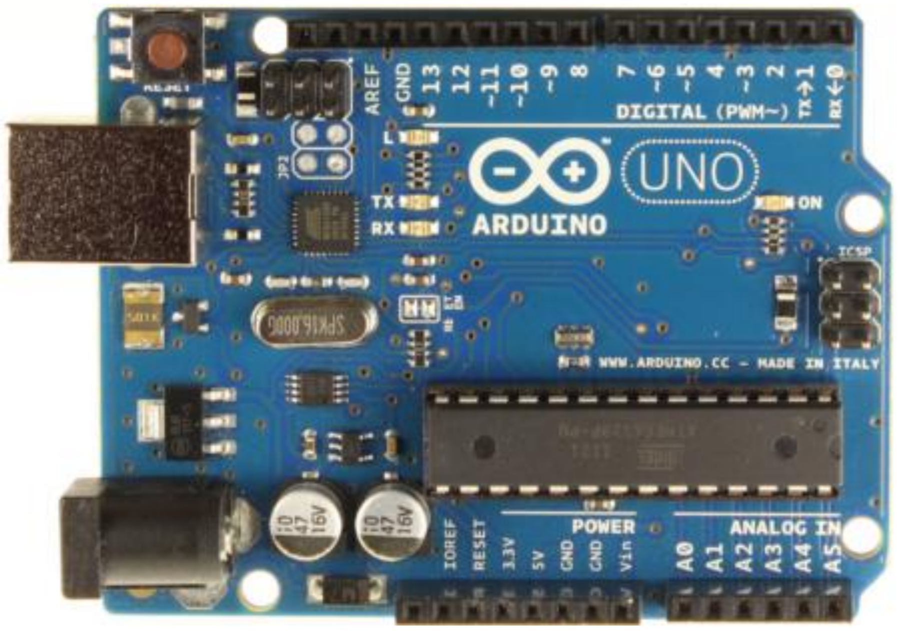


Choosing a Microcontroller

- All very similar, so stick with a family you know
- Required features
- Required number of GPIOs
- Memory requirements
- Availability of programmer (USB?)
- Availability of a good C compiler
- Packaging

Development Kits

- Fast and Easy!
- Everything you need is included
  - uC, power supply, USB connection, simple IO
- Low level code is written for you!
  - DigitalWrite(13, HIGH)
- Example code and projects
- Often large online forums for support



Example: Arduino (AVR based kit)

Types of Embedded Systems

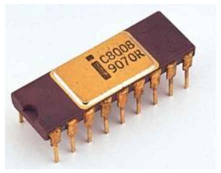
Development Kit | **Evaluation Board** | **Custom µC** | **Custom µP**

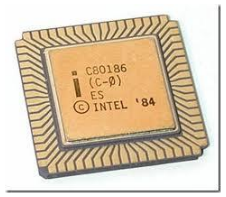
Arduino

MSP430 LaunchPad

eZ430

eZ430-Chronos





Easiest ← **Ease of Prototyping** → *Hardest*

Shortest ← **Prototyping Time** → *Longest*

Least ← **Design Flexibility** → *Most*

Focus of this Class

Rapid Prototyping Kit

Phidgets

Development Kit

Arduino

Evaluation Board

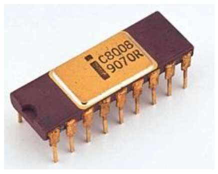
MSP430 LaunchPad

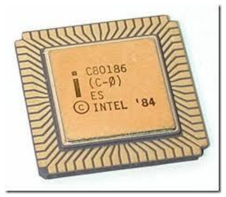
eZ430

eZ430-Chronos

Custom μC

Custom μP





Easiest — **Ease of Prototyping** — *Hardest*

Shortest — **Prototyping Time** — *Longest*

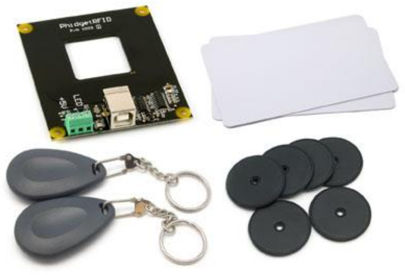
Least — **Design Flexibility** — *Most*

Phidgets Tutorial

Gabe Cohn

What are Phidgets?

- plug and play building blocks for low cost USB sensing and control from your PC
- Published in UIST 2001: Greenberg and Fitchett
- www.phidgets.com



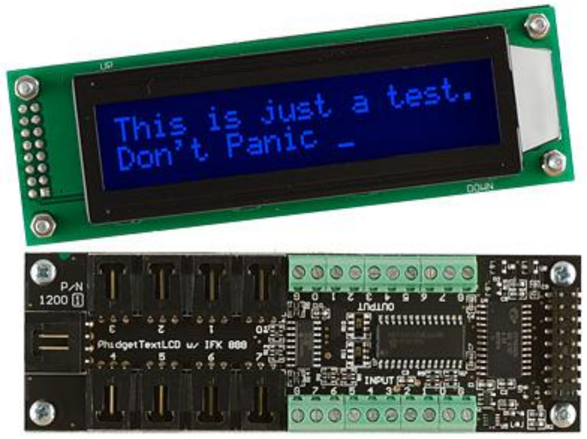
Inputs (Sensors)

- Linear Touch
- Circular Touch
- Temperature
- Knob
- Ph
- Accelerometer
- IR reflective
- Vibration
- Force
- Gas Pressure
- Light
- Magnetic
- Rotation
- Touch
- Motion
- Slider
- Joy Stick
- Pressure
- Current
- Voltage
- Sonar
- IR Distance

Best Selection of Sensors!

Outputs

- Motor
  - Servo
  - Stepper
  - DC
- Display
  - Text LCD
- Host Computer



Platforms

- Windows
- Linux
- Mac OS X
- Windows Mobile/CE
- SBC Firmware
- iPhone

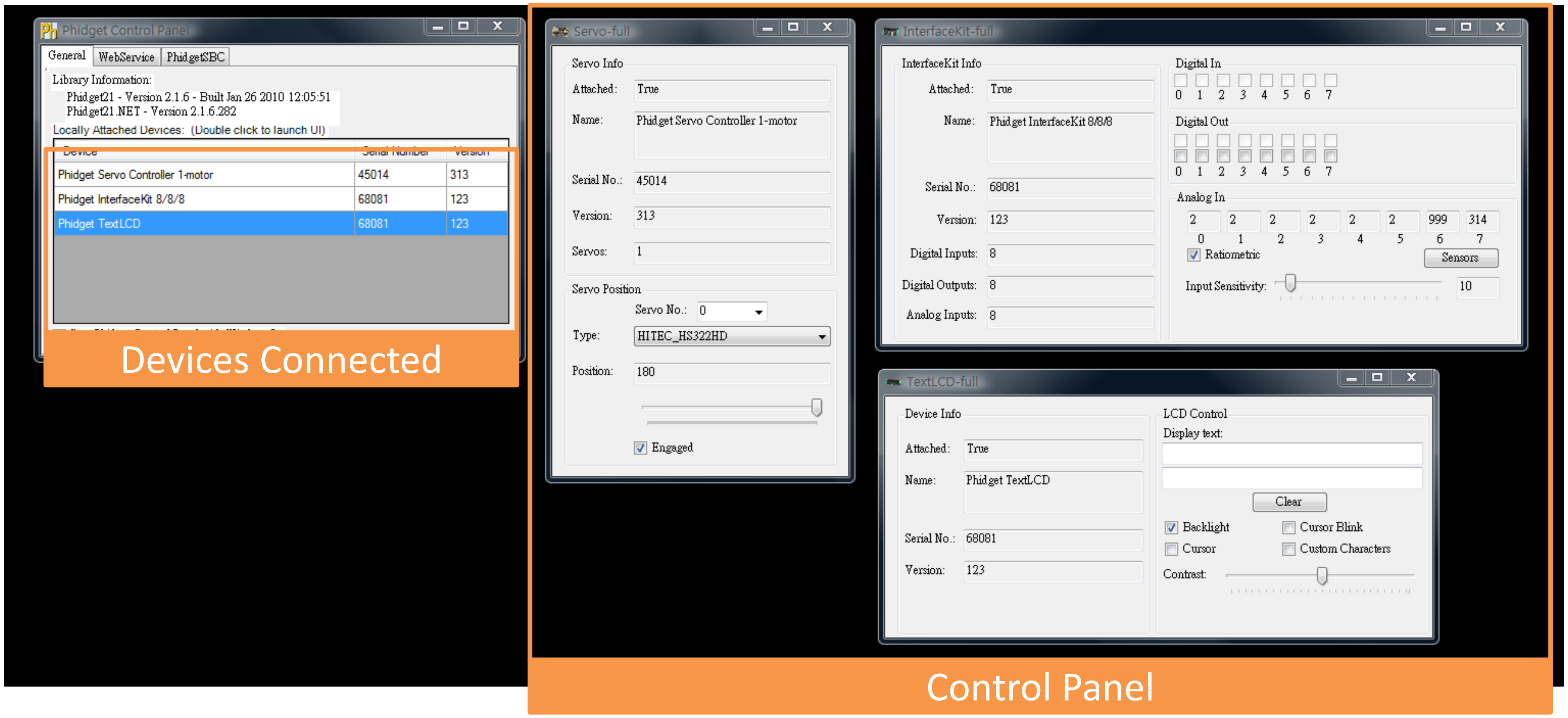
Software API

- Adobe Director
- C#
- Cocoa
- Flash AS3
- Java
- Matlab
- Micosoft Robotics Studio 1.5
- REALBasic
- Visual Basic 6.0
- Visual Basic Script
- AutoIt
- C/C++
- Delphi
- Flex AS3
- LabVIEW
- Max/MSP
- Python
- Visual Basic .NET
- Visual Basic for Application
- Visual C/C++/Borland

http://www.phidgets.com/programming_resources.php

Phidget Control Panel

- http://www.phidgets.com/drivers.php



Phidget Control Panel

General | WebService | PhidgetSBC

Library Information:
Phidget21 - Version 2.1.6 - Built Jan 26 2010 12:05:51
Phidget21.NET - Version 2.1.6.282

Locally Attached Devices: (Double click to launch UI)

Device	Serial Number	Version
Phidget Servo Controller 1-motor	45014	313
Phidget InterfaceKit 8/8/8	68081	123
Phidget TextLCD	68081	123

Servo-full

Servo Info

Attached:	True
Name:	Phidget Servo Controller 1-motor
Serial No.:	45014
Version:	313
Servos:	1

Servo Position

Servo No.:	0
Type:	HITEC_HS322HD
Position:	180

☑ Engaged

InterfaceKit-full

InterfaceKit Info

Attached:	True
Name:	Phidget InterfaceKit 8/8/8
Serial No.:	68081
Version:	123
Digital Inputs:	8
Digital Outputs:	8
Analog Inputs:	8

Digital In: 0 1 2 3 4 5 6 7

Digital Out: 0 1 2 3 4 5 6 7

Analog In

0	1	2	3	4	5	6	7
2	2	2	2	2	2	999	314

☑ Ratiometric

Input Sensitivity: 10

TextLCD-full

Device Info

Attached:	True
Name:	Phidget TextLCD
Serial No.:	68081
Version:	123

LCD Control

Display text:

☑ Backlight
☐ Cursor Blink
☐ Cursor
☐ Custom Characters

Contrast:

Demo

- **Inputs:** RFID reader, Slider
- **Output:** LCD, Servo motor, and Command Line

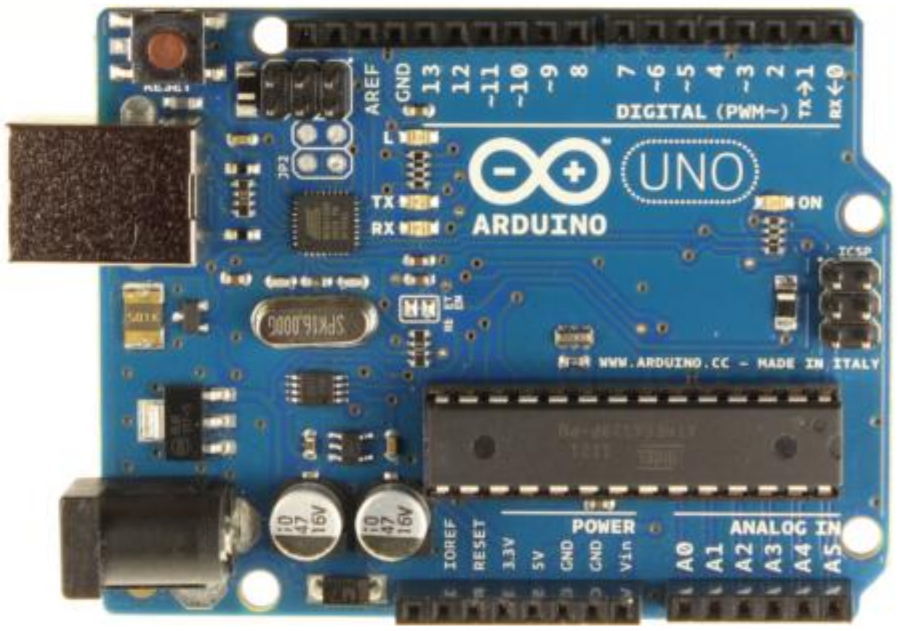
- All written in python using resources at http://www.phidgets.com/programming_resources.php
- Demo code available at: www.gabeacohn.com/teaching/micro

Arduino Tutorial

Gabe Cohn

Arduino

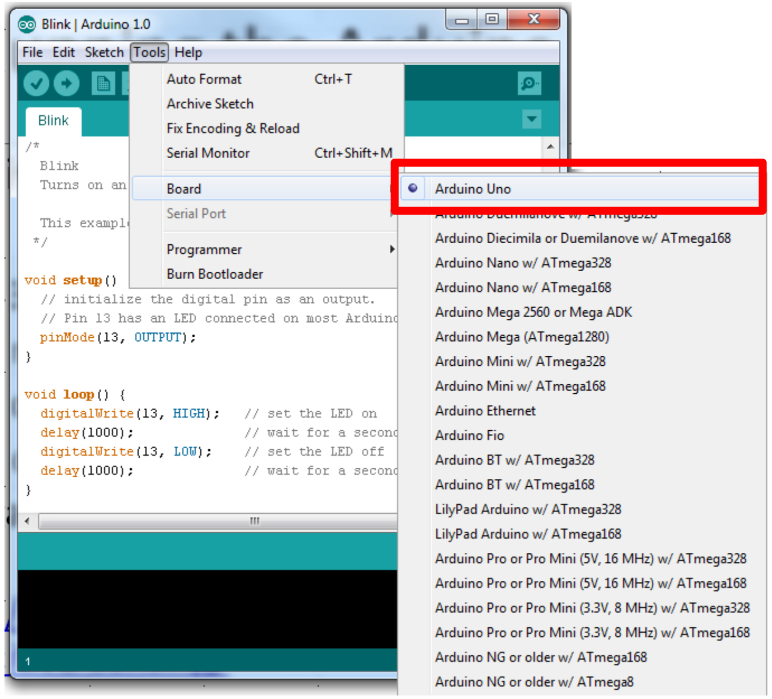
- Uses Atmel AVR
- Hardware contains everything you need
- Simple high-level C/C++ based programming language
- Very easy to use
- Example code and projects
- Large online forums for support
- Can also write to AVR registers for low-level functionality



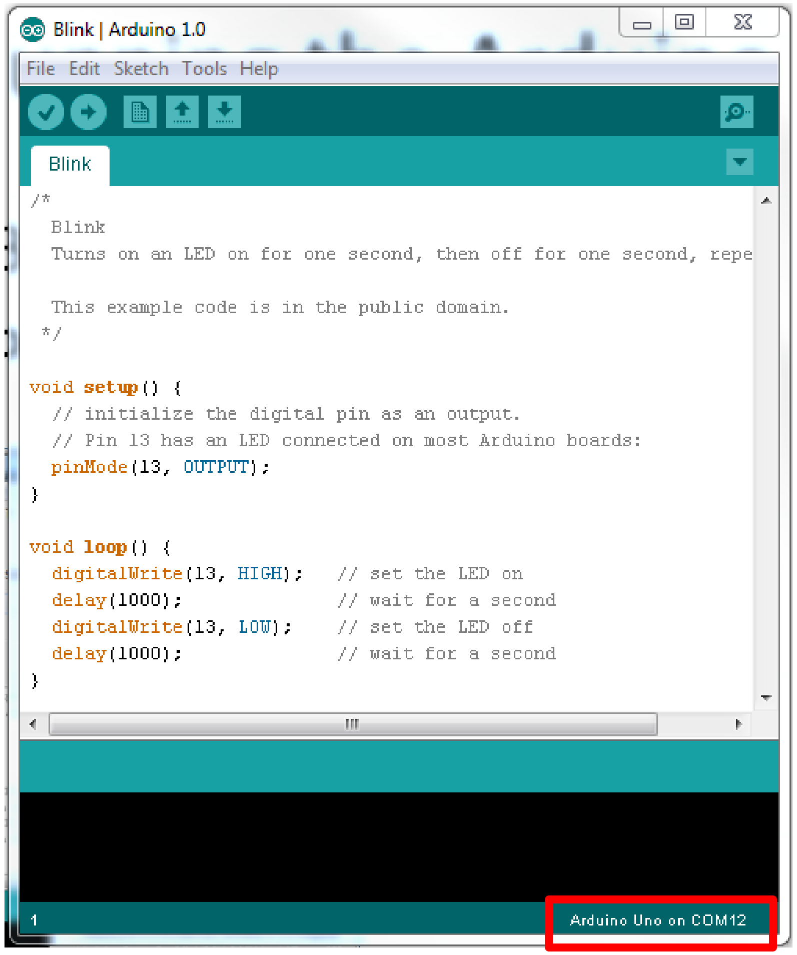
Arduino UNO

Running the Arduino IDE

- Select Board
- Select Port


Blink | Arduino 1.0
File Edit Sketch Tools Help
Blink
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
Arduino Uno
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8

```
/*
  Blink
  Turns on an
  
  This exampl
 */

void setup()
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```



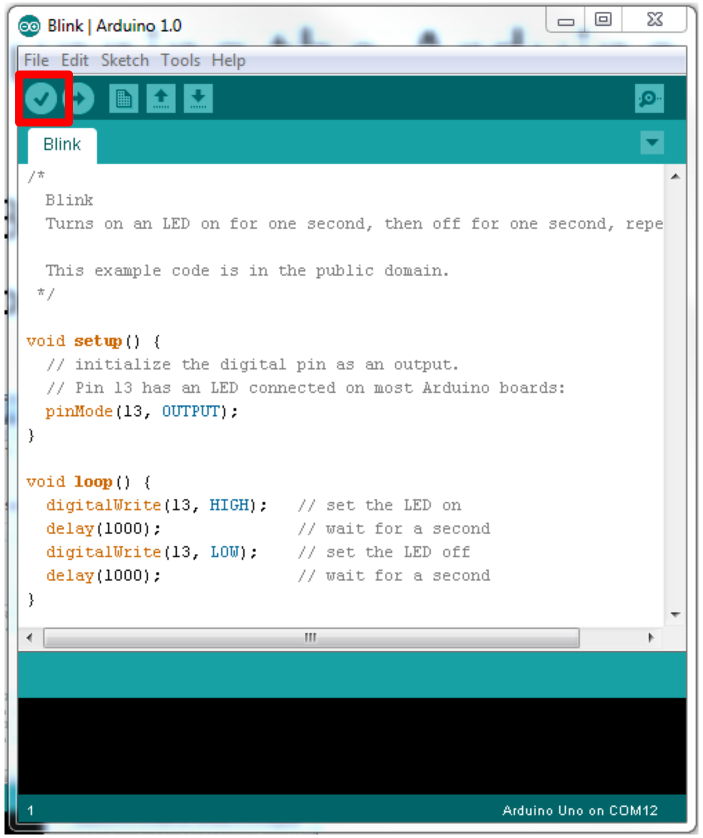
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

Arduino Uno on COM12

Verify correct board and port

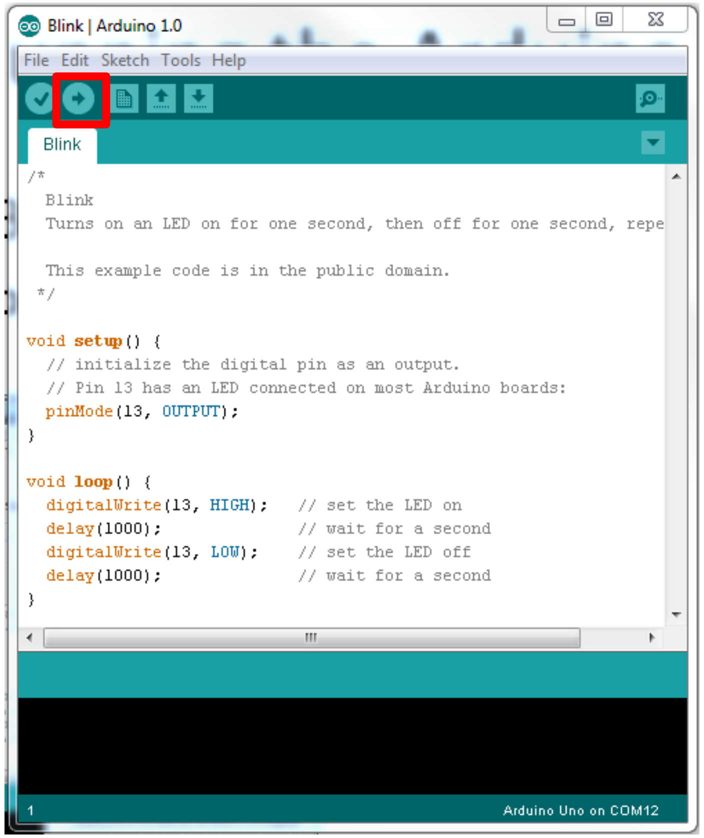
Running the Arduino IDE

- Compile Code



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

- Download Code to Board



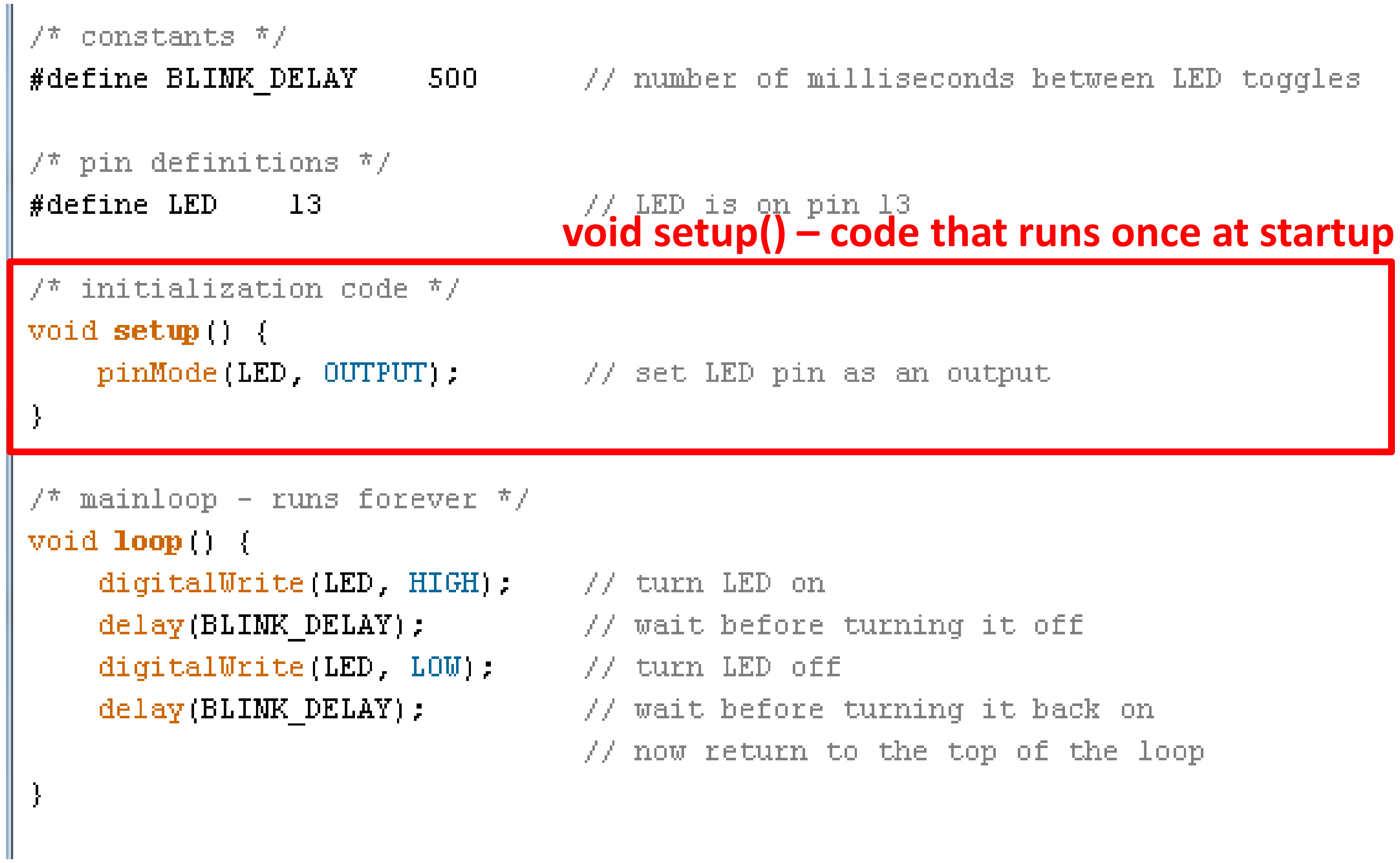
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

Arduino Code (Hello World)

Can define constants just like in C/C++

```
/* constants */
#define BLINK_DELAY    500       // number of milliseconds between LED toggles

/* pin definitions */
#define LED    13                // LED is on pin 13

/* initialization code */
void setup() {
    pinMode(LED, OUTPUT);        // set LED pin as an output
}

/* mainloop - runs forever */
void loop() {
    digitalWrite(LED, HIGH);     // turn LED on
    delay(BLINK_DELAY);          // wait before turning it off
    digitalWrite(LED, LOW);      // turn LED off
    delay(BLINK_DELAY);          // wait before turning it back on
                                 // now return to the top of the loop
}
```

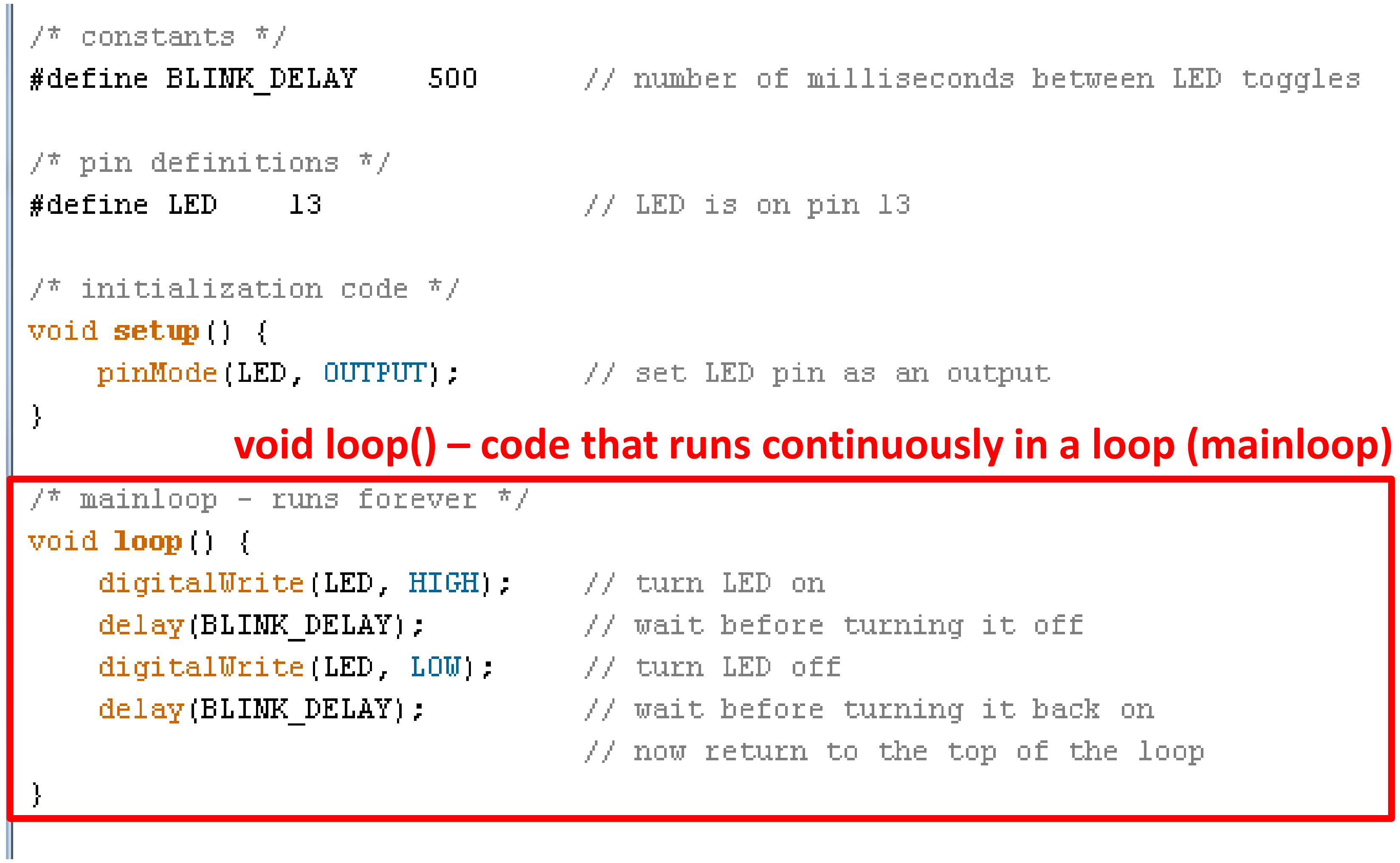
Arduino Code (Hello World)



```
/* constants */
#define BLINK_DELAY    500       // number of milliseconds between LED toggles

/* pin definitions */
#define LED     13               // LED is on pin 13

/* initialization code */
void setup() {
    pinMode(LED, OUTPUT);        // set LED pin as an output
}

/* mainloop - runs forever */
void loop() {
    digitalWrite(LED, HIGH);     // turn LED on
    delay(BLINK_DELAY);          // wait before turning it off
    digitalWrite(LED, LOW);      // turn LED off
    delay(BLINK_DELAY);          // wait before turning it back on
                                 // now return to the top of the loop
}
```

void setup() – code that runs once at startup

Arduino Code (Hello World)



```
/* constants */
#define BLINK_DELAY    500        // number of milliseconds between LED toggles

/* pin definitions */
#define LED    13                 // LED is on pin 13

/* initialization code */
void setup() {
    pinMode(LED, OUTPUT);         // set LED pin as an output
}
```

void loop() – code that runs continuously in a loop (mainloop)

```
/* mainloop - runs forever */
void loop() {
    digitalWrite(LED, HIGH);      // turn LED on
    delay(BLINK_DELAY);           // wait before turning it off
    digitalWrite(LED, LOW);       // turn LED off
    delay(BLINK_DELAY);           // wait before turning it back on
                                  // now return to the top of the loop
}
```

Arduino Demos

- **Hello World**

  Blinks an LED
- **Interrupts**

  Switch toggles blinking LED (switch press triggers ISR)
- **PWM**

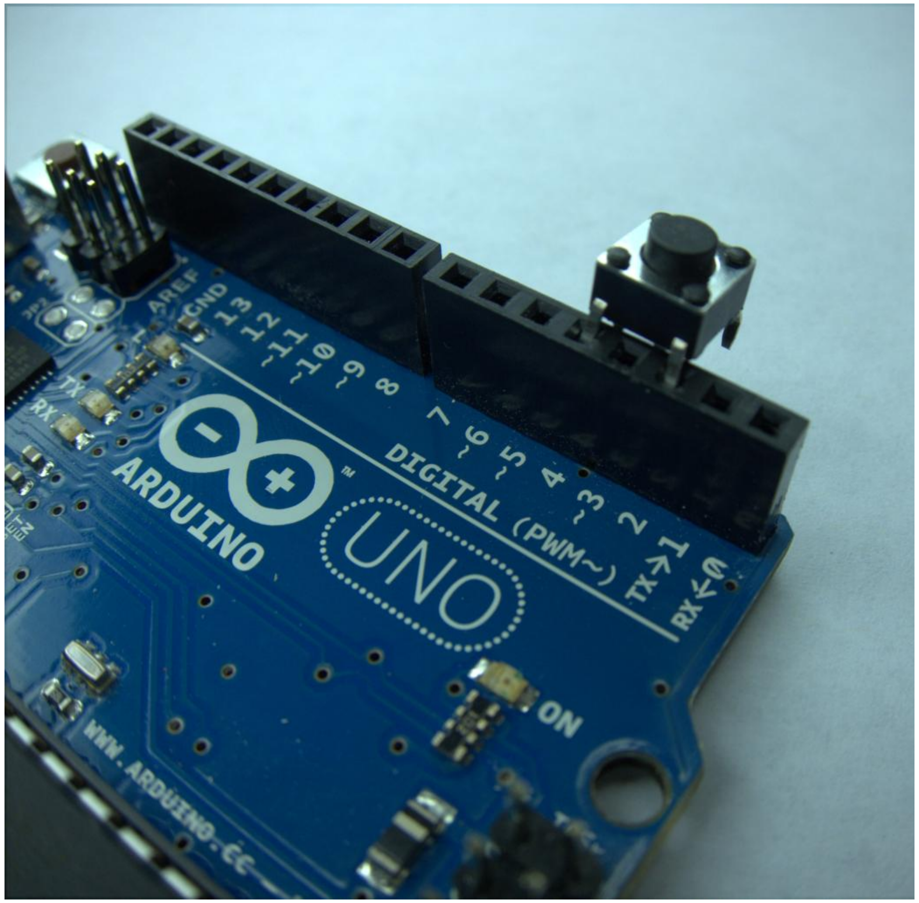
  LED brightness changes continuously using PWM
- **ADC**

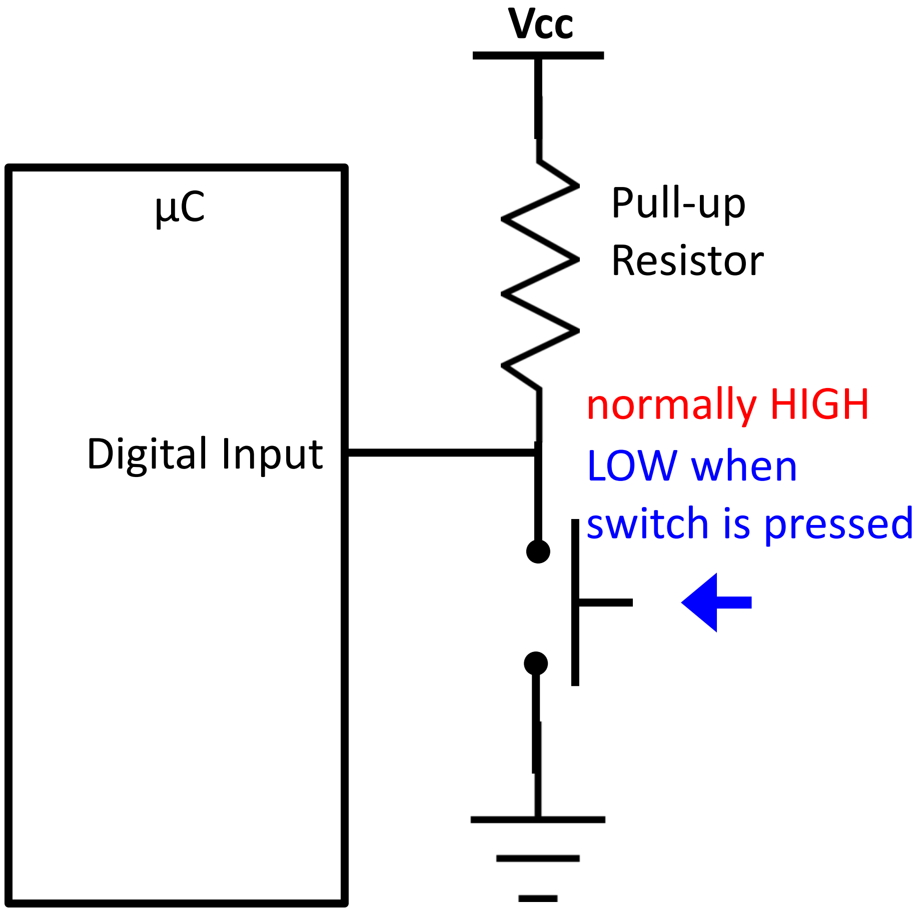
  Periodically samples voltage across light sensor and outputs brightness level using the LED

- Code Available at: www.gabeacohn.com/teaching/micro

Arduino Interrupts Demo

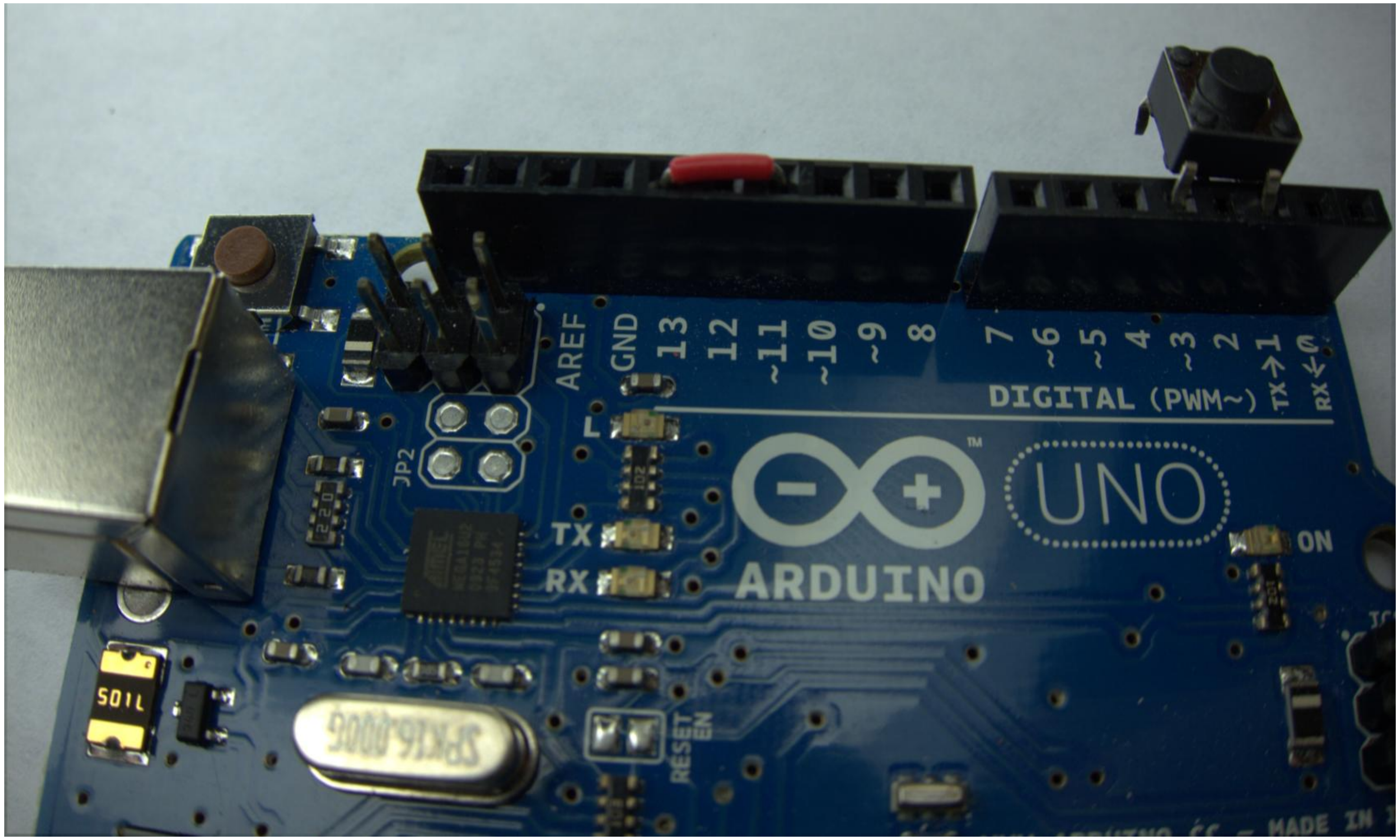
- Need to connect switch between pins 2 and 4





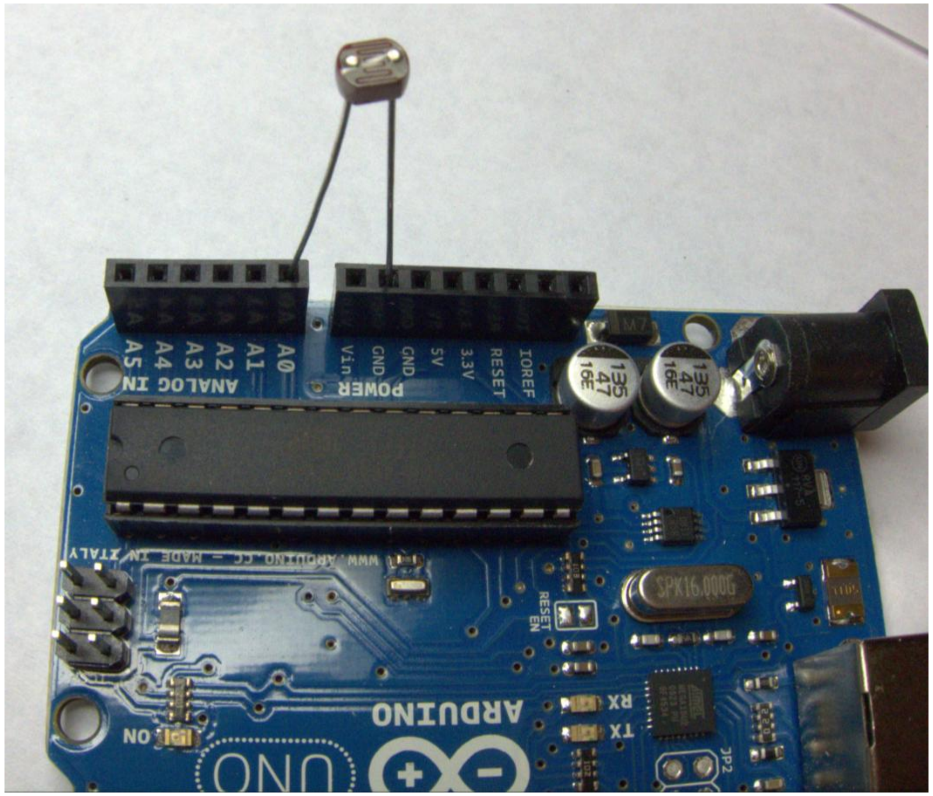
Arduino PWM Demo

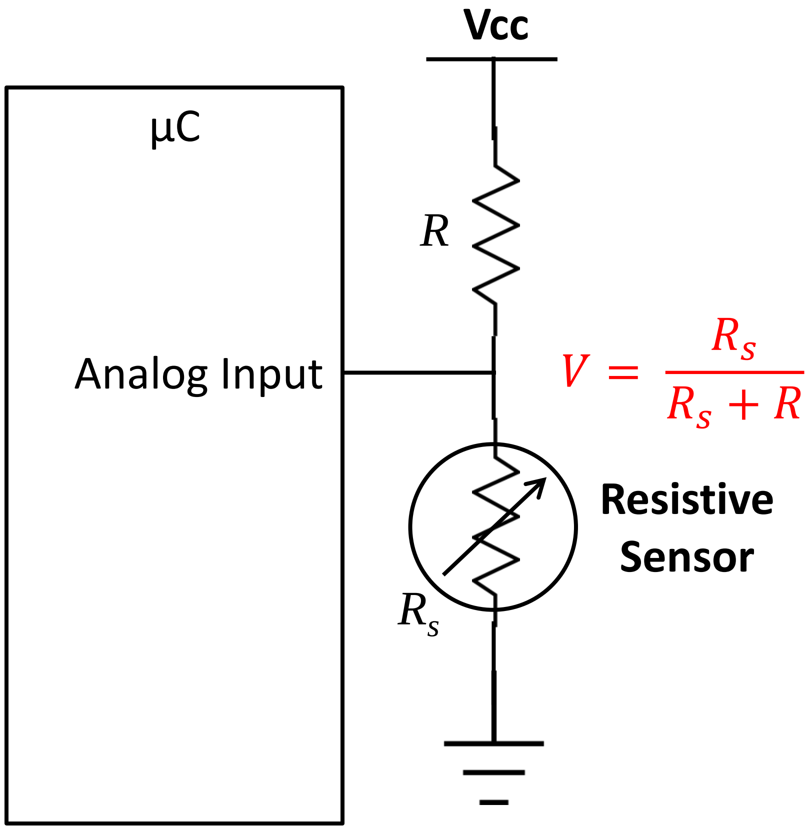
- Need to connect a wire between pins 11 and 13



Arduino ADC Demo

- Need to connect photo-resistor between pin A0 and GND





MSP 430 Tutorial

Gabe Cohn

TI MSP 430

- Ultra-low-power!
- Widely used in low-power research
  - Power harvesting
  - Ultra-low-power sensor networks
- More complicated than AVR (Atmega)
- Not used much in industry (yet…)
- Very low cost evaluation/dev kits

MSP430 Eval/Dev Kits

MSP430 LaunchPad

$4.30

eZ430

eZ430-Chronos

MSP430 Launch Pad Dev. Kit

- Very low cost!
- Simple MSP430
- USB programmer / debugger
- 1 PB-switch
- 2 LEDs (red and green)
- All I/O pins exposed
- **Only $4.30!**

eZ430 Dev. Kit

- USB thumb-drive form-factor
- Simple MSP430
- USB programmer / debugger
- Removable target board
- All I/O pins exposed
- RF versions available
  (e.g. eZ430-RF2500)

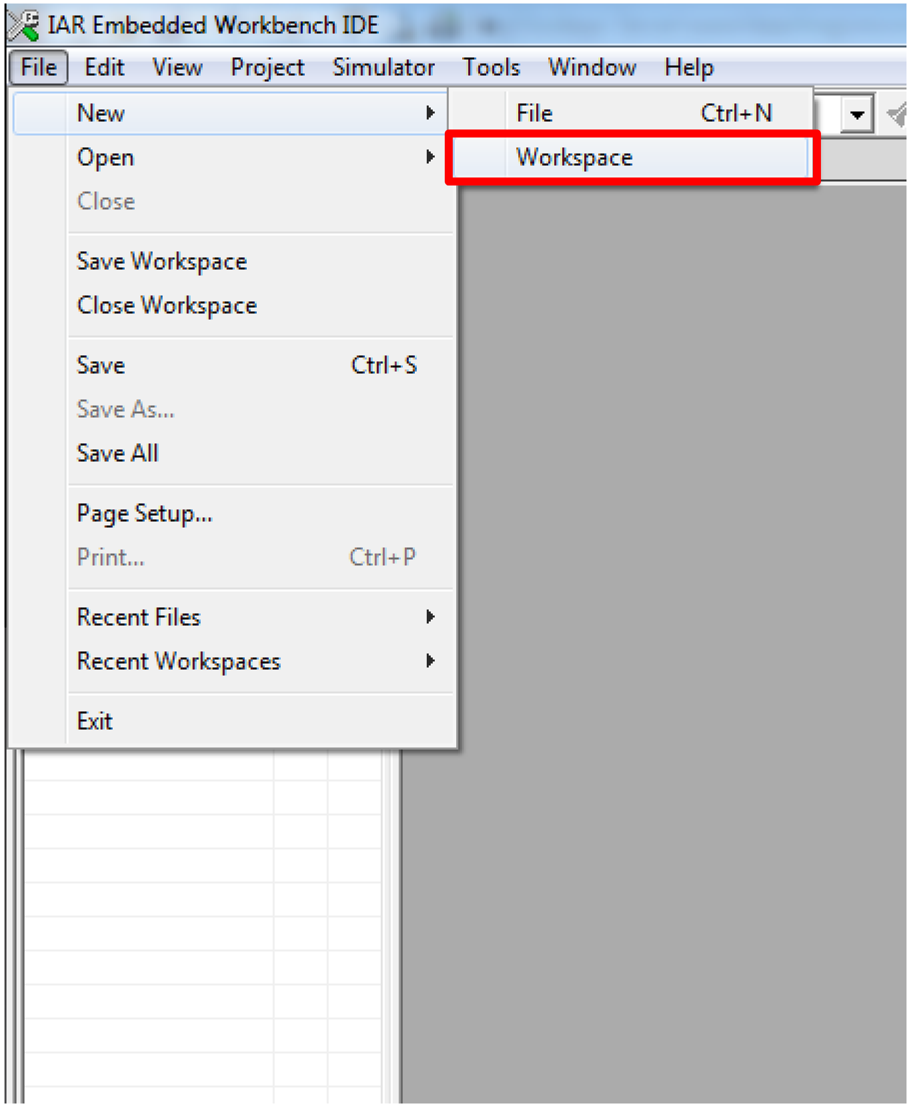
eZ430-Chronos Dev. Kit

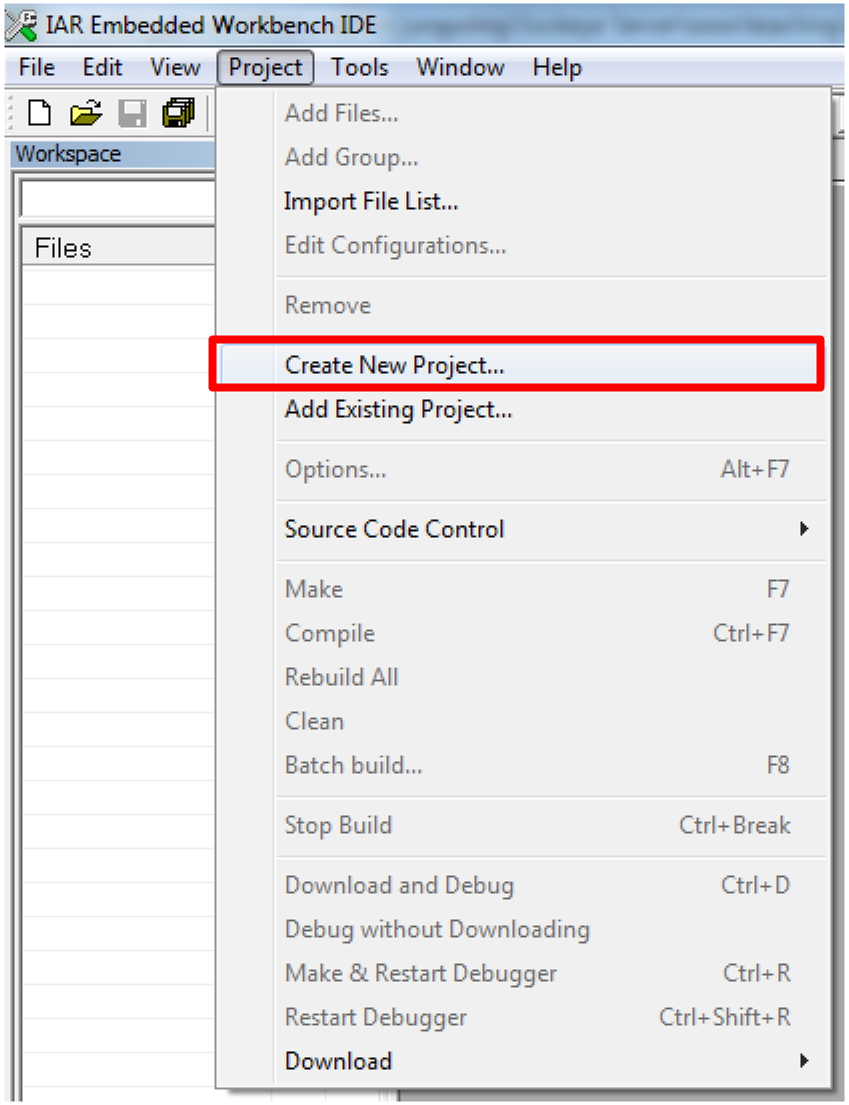
- Watch form-factor!
- Wireless programmer!
- USB programmer / debugger
- 3-axis accelerometer
- Barometric pressure sensor
- Temperature sensor
- Battery/Voltage sensor
- BlueRobin protocol (heart-rate)

Software Environment (IDE)

- IAR Embedded Workbench (IAR)
  - C/C++ compiler
  - simulator and debugger
  - Free version with 4 KB code size limit
  - easy to use and understand
- Code Composer Studio (CCS)
  - Eclipse
  - Free version with 16 KB code size limit
  - recommended for larger (RF) projects
  - complicated and buggy!

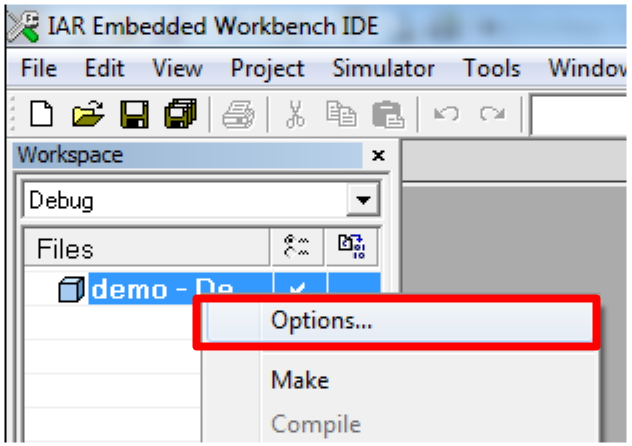
Create IAR Workspace and Project

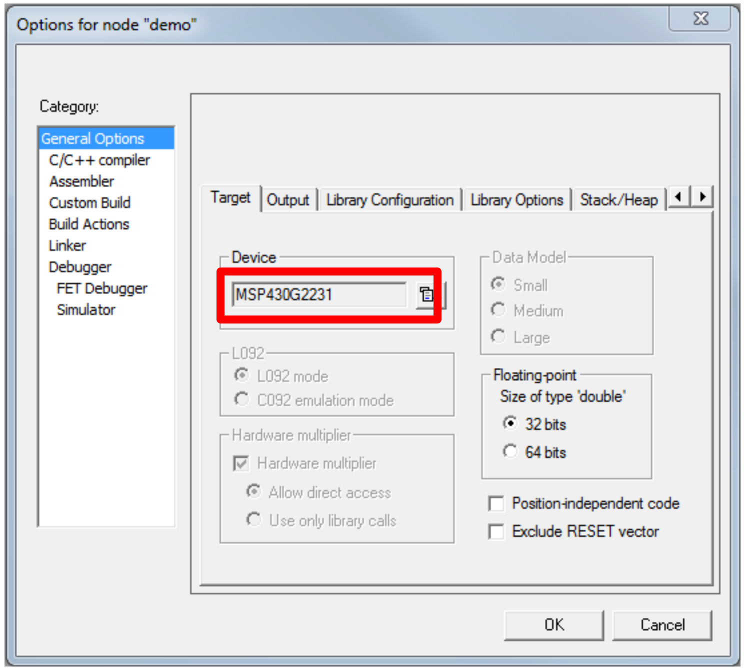




Set Project Options

- Device: MSP430G2231
- Debugger Driver: FET Debugger


IAR Embedded Workbench IDE
File Edit View Project Simulator Tools Window
Workspace
Debug
Files
demo - De
Options...
Make
Compile



Device: MSP430G2231

Data Model: [● Small] Medium Large

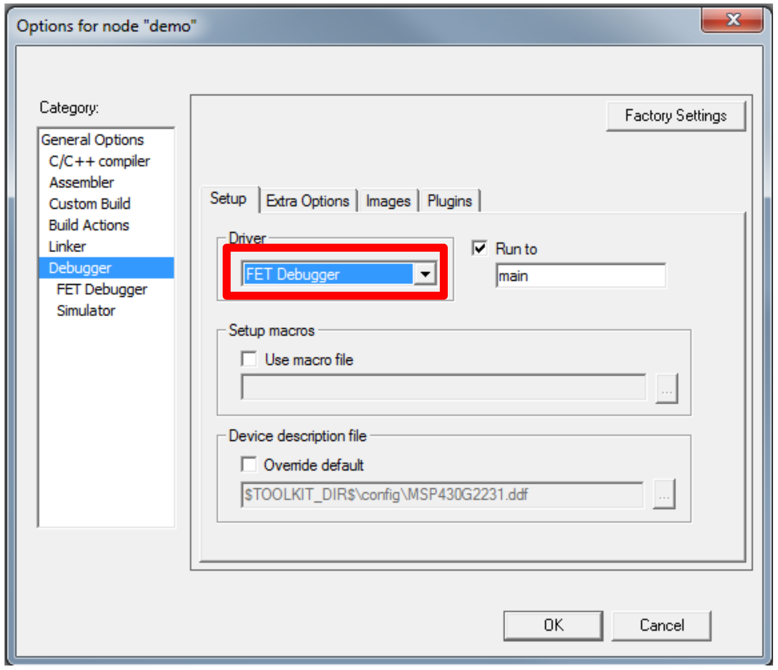
L092: [● L092 mode] C092 emulation mode

Hardware multiplier: [✓ Hardware multiplier] [● Allow direct access] Use only library calls

Floating-point: Size of type 'double': [● 32 bits] 64 bits

☐ Position-independent code

☐ Exclude RESET vector



Driver: FET Debugger

[✓ Run to] main

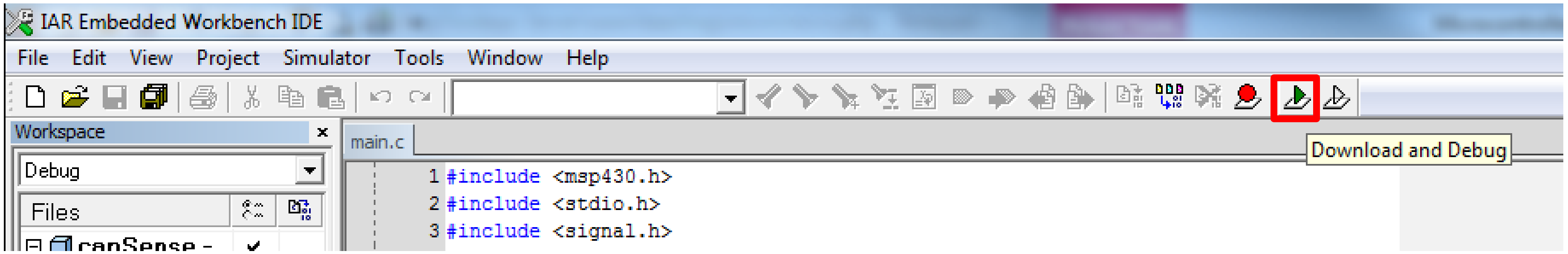
Setup macros: ☐ Use macro file

Device description file: ☐ Override default

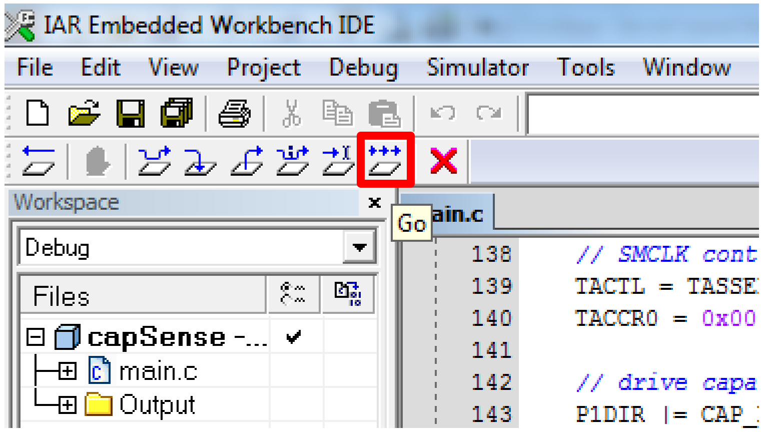
$TOOLKIT_DIR$\config\MSP430G2231.ddf

Program and Run the Code

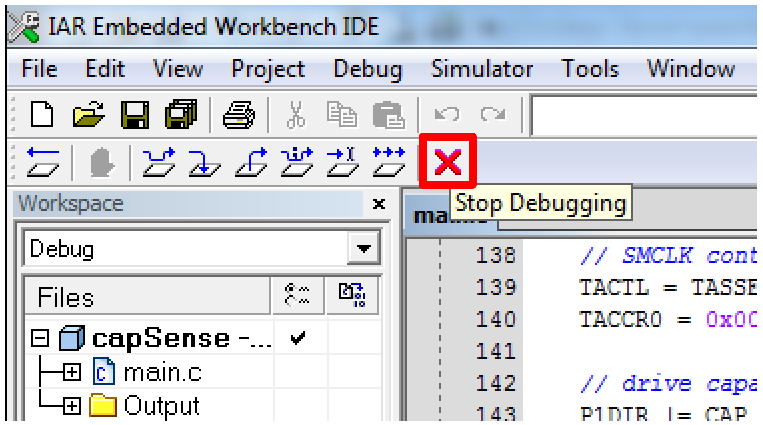
- Download and Run code on MSP 430


IAR Embedded Workbench IDE
File Edit View Project Simulator Tools Window Help
Workspace
Debug
Files
main.c
Download and Debug

```
#include <msp430.h>
#include <stdio.h>
#include <signal.h>
```


IAR Embedded Workbench IDE
File Edit View Project Debug Simulator Tools Window
Workspace
Go
Debug
Files
capSense -...
main.c
Output

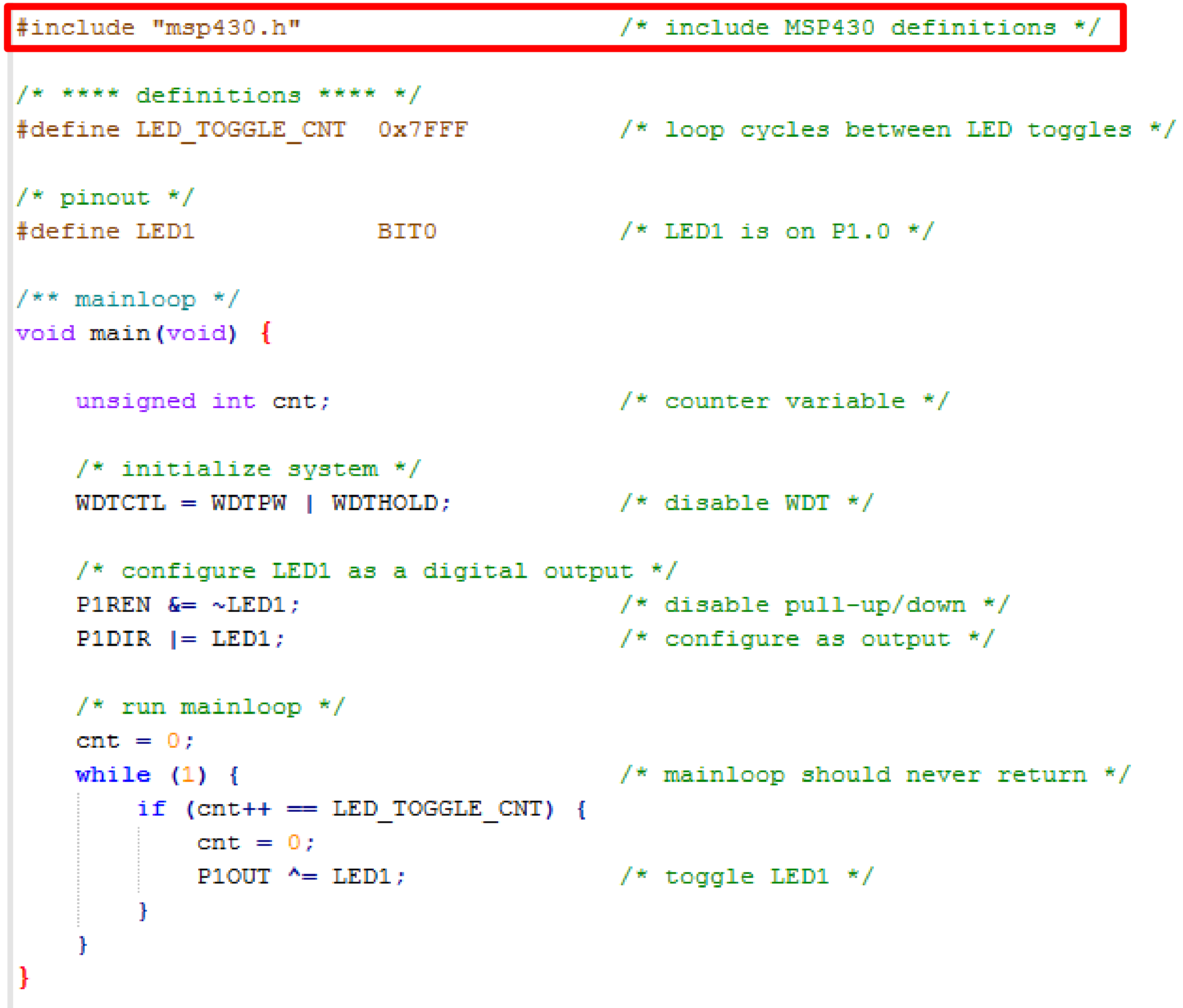
```
    // SMCLK cont
    TACTL = TASSE
    TACCR0 = 0x00

    // drive capa
    P1DIR |= CAP_
```


IAR Embedded Workbench IDE
File Edit View Project Debug Simulator Tools Window
Workspace
Stop Debugging
Debug
Files
capSense -...
main.c
Output

```
    // SMCLK cont
    TACTL = TASSE
    TACCR0 = 0x0C

    // drive capa
    P1DIR |= CAP
```

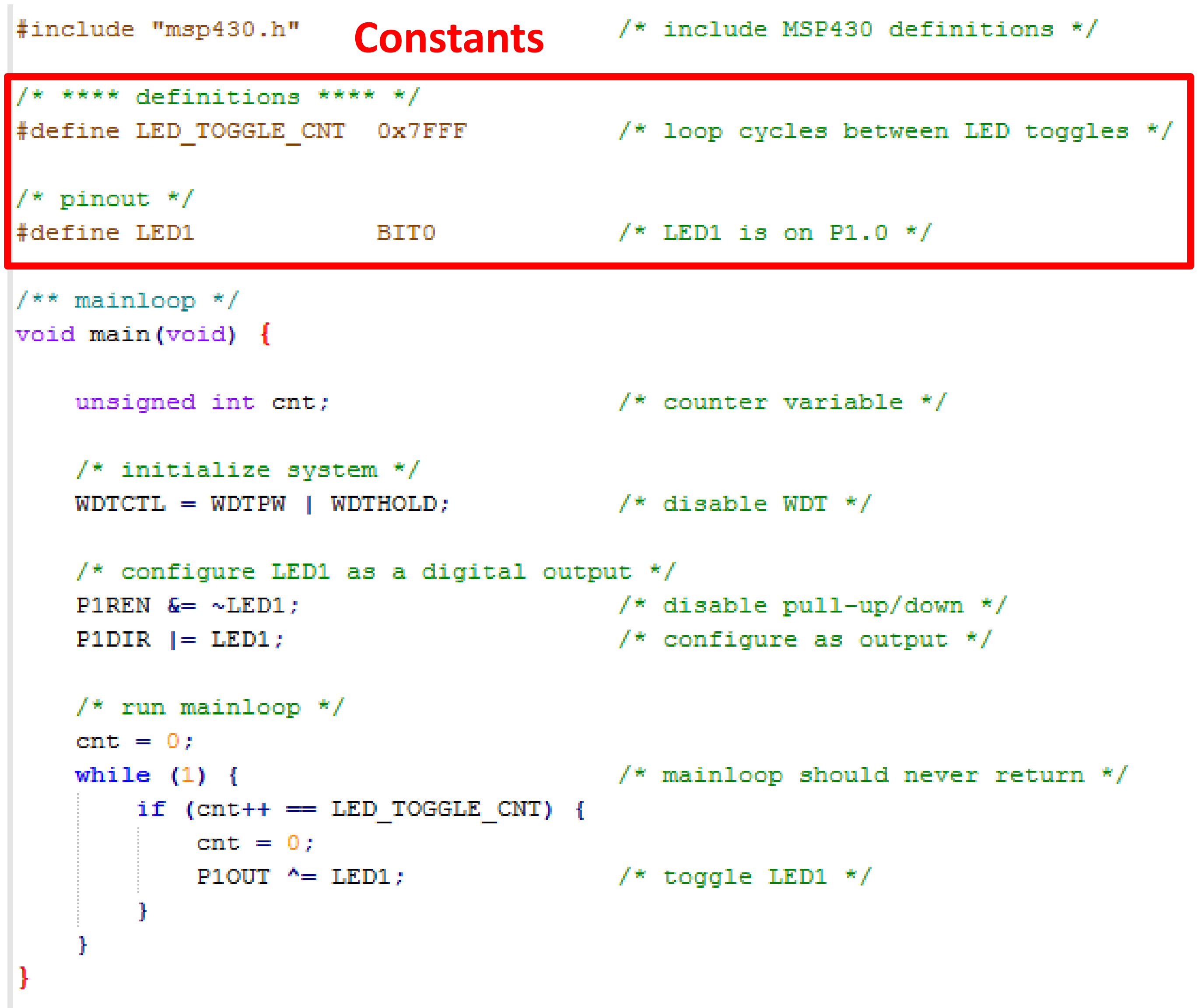
MSP 430 Code (Hello World)

Contains all definitions for specific device



```c
#include "msp430.h"                     /* include MSP430 definitions */

/* **** definitions **** */
#define LED_TOGGLE_CNT  0x7FFF          /* loop cycles between LED toggles */

/* pinout */
#define LED1            BIT0            /* LED1 is on P1.0 */

/** mainloop */
void main(void) {

    unsigned int cnt;                   /* counter variable */

    /* initialize system */
    WDTCTL = WDTPW | WDTHOLD;           /* disable WDT */

    /* configure LED1 as a digital output */
    P1REN &= ~LED1;                     /* disable pull-up/down */
    P1DIR |= LED1;                      /* configure as output */

    /* run mainloop */
    cnt = 0;
    while (1) {                         /* mainloop should never return */
        if (cnt++ == LED_TOGGLE_CNT) {
            cnt = 0;
            P1OUT ^= LED1;              /* toggle LED1 */
        }
    }
}
```

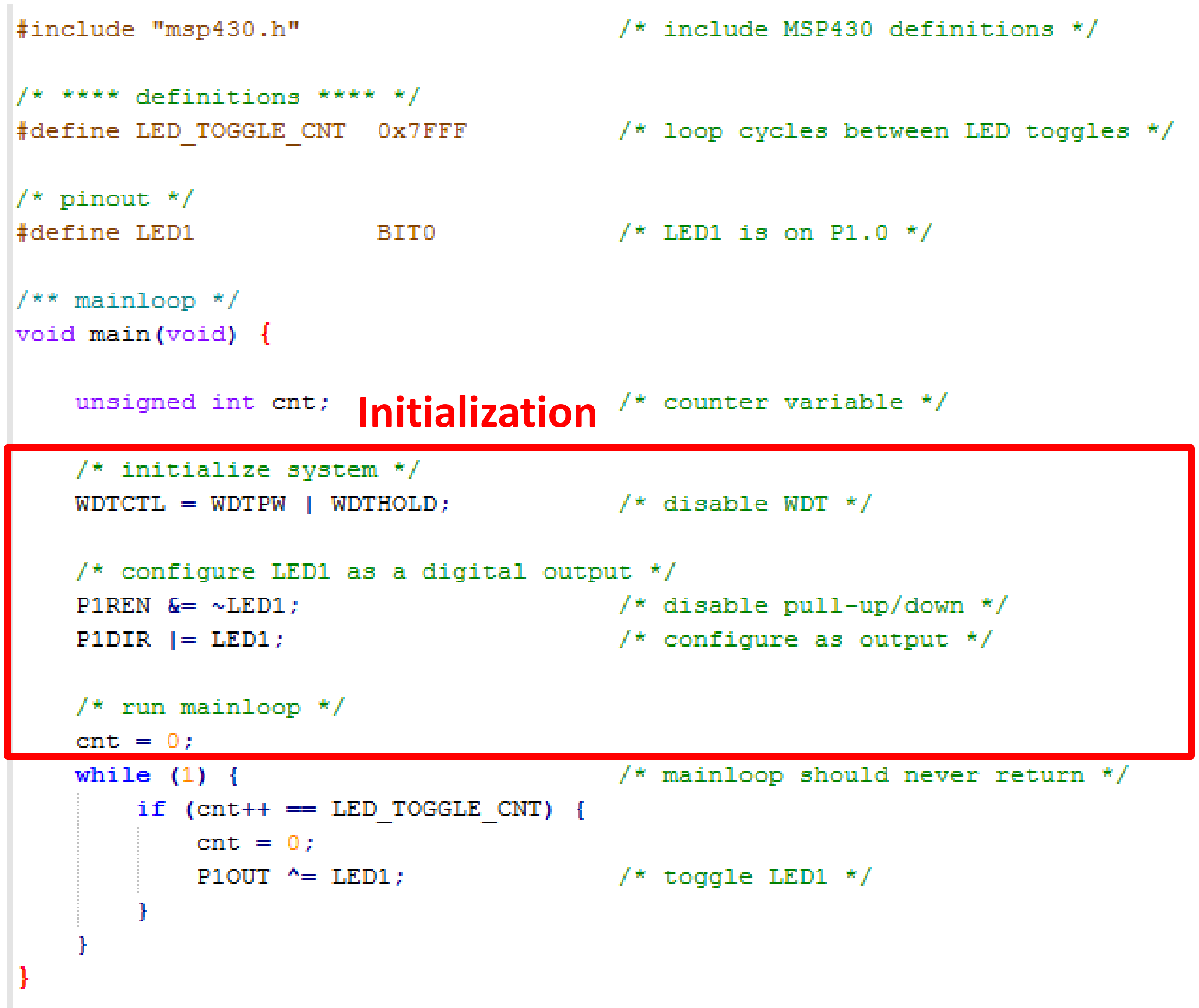
MSP 430 Code (Hello World)



Constants

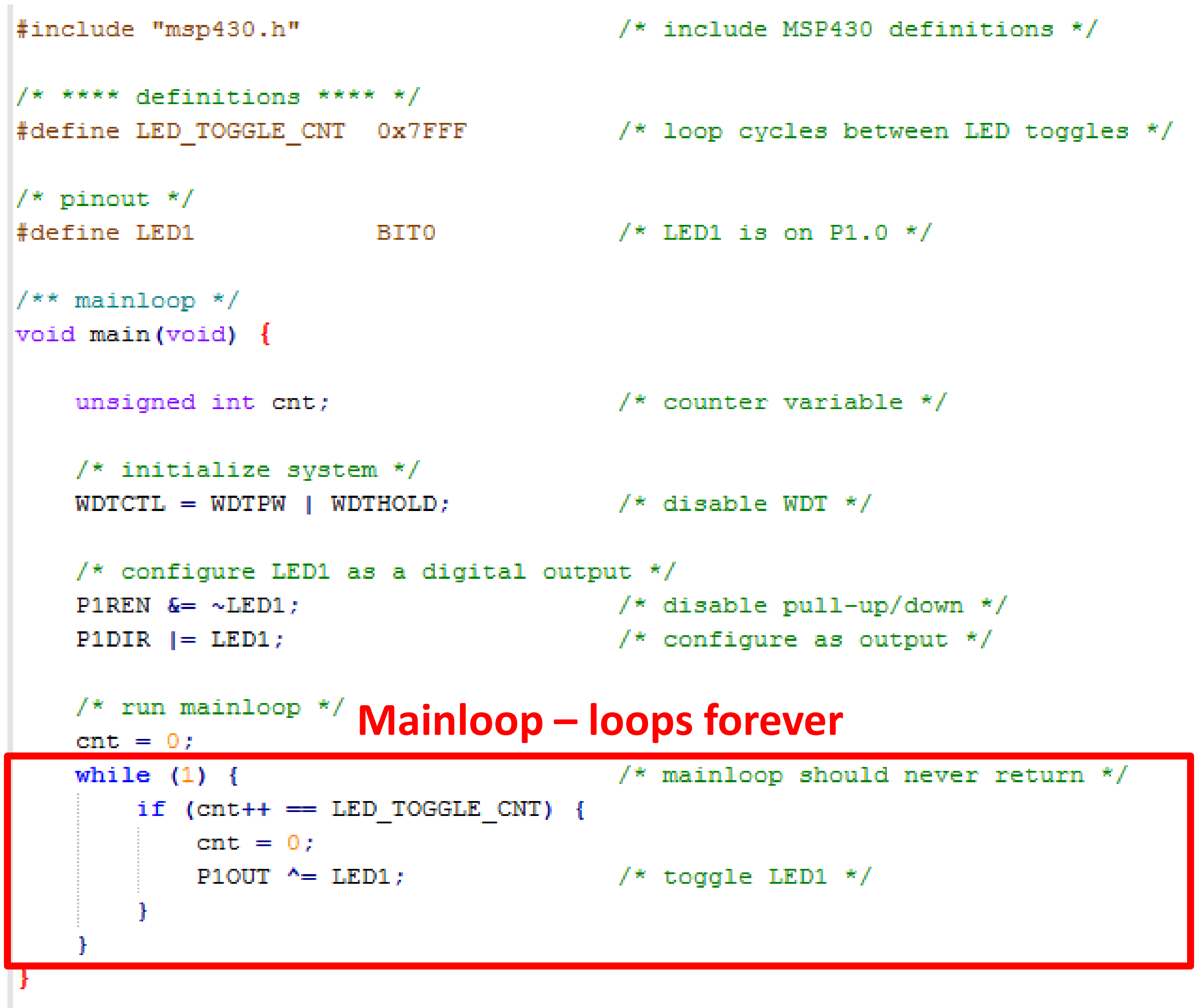
```c
#include "msp430.h"                         /* include MSP430 definitions */

/* **** definitions **** */
#define LED_TOGGLE_CNT   0x7FFF             /* loop cycles between LED toggles */

/* pinout */
#define LED1             BIT0               /* LED1 is on P1.0 */

/** mainloop */
void main(void) {

    unsigned int cnt;                       /* counter variable */

    /* initialize system */
    WDTCTL = WDTPW | WDTHOLD;               /* disable WDT */

    /* configure LED1 as a digital output */
    P1REN &= ~LED1;                         /* disable pull-up/down */
    P1DIR |= LED1;                          /* configure as output */

    /* run mainloop */
    cnt = 0;
    while (1) {                             /* mainloop should never return */
        if (cnt++ == LED_TOGGLE_CNT) {
            cnt = 0;
            P1OUT ^= LED1;                  /* toggle LED1 */
        }
    }
}
```

MSP 430 Code (Hello World)



```c
#include "msp430.h"                     /* include MSP430 definitions */

/* **** definitions **** */
#define LED_TOGGLE_CNT   0x7FFF          /* loop cycles between LED toggles */

/* pinout */
#define LED1             BIT0            /* LED1 is on P1.0 */

/** mainloop */
void main(void) {

    unsigned int cnt;                    /* counter variable */

    /* initialize system */
    WDTCTL = WDTPW | WDTHOLD;            /* disable WDT */

    /* configure LED1 as a digital output */
    P1REN &= ~LED1;                      /* disable pull-up/down */
    P1DIR |= LED1;                       /* configure as output */

    /* run mainloop */
    cnt = 0;
    while (1) {                          /* mainloop should never return */
        if (cnt++ == LED_TOGGLE_CNT) {
            cnt = 0;
            P1OUT ^= LED1;               /* toggle LED1 */
        }
    }
}
```

MSP 430 Code (Hello World)

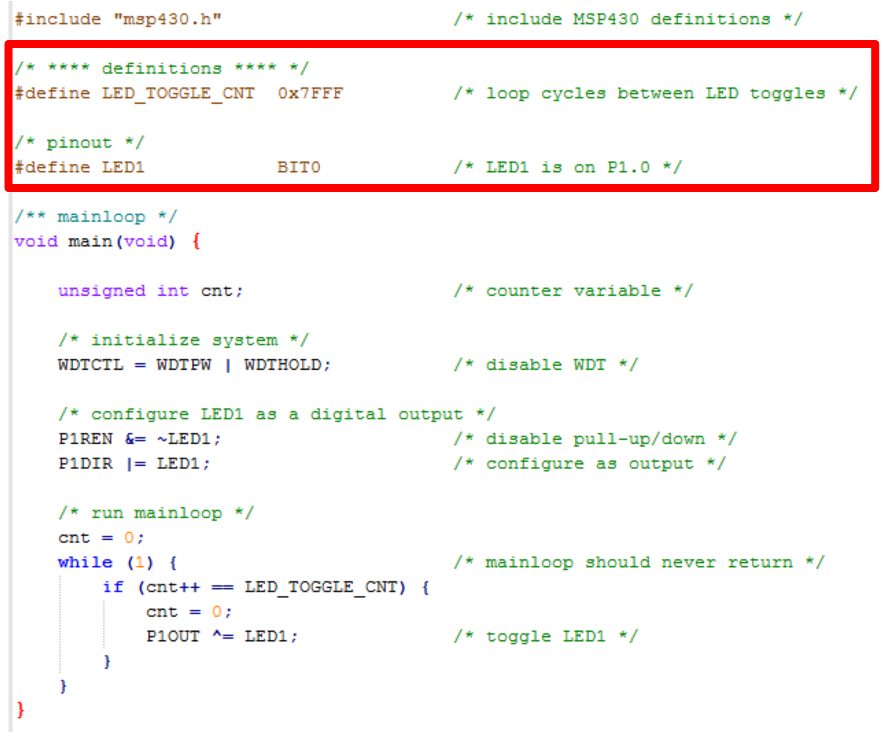


```c
#include "msp430.h"                         /* include MSP430 definitions */

/* **** definitions **** */
#define LED_TOGGLE_CNT   0x7FFF             /* loop cycles between LED toggles */

/* pinout */
#define LED1             BIT0               /* LED1 is on P1.0 */

/** mainloop */
void main(void) {

    unsigned int cnt;                       /* counter variable */

    /* initialize system */
    WDTCTL = WDTPW | WDTHOLD;               /* disable WDT */

    /* configure LED1 as a digital output */
    P1REN &= ~LED1;                         /* disable pull-up/down */
    P1DIR |= LED1;                          /* configure as output */

    /* run mainloop */
    cnt = 0;
    while (1) {                             /* mainloop should never return */
        if (cnt++ == LED_TOGGLE_CNT) {
            cnt = 0;
            P1OUT ^= LED1;                  /* toggle LED1 */
        }
    }
}
```

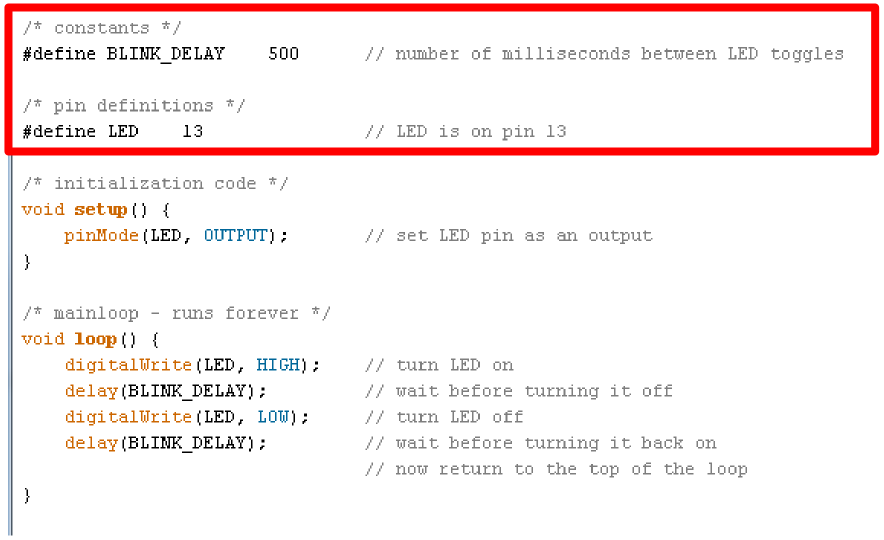
Mainloop – loops forever

MSP 430 vs. Arduino Code

Constant Definitions



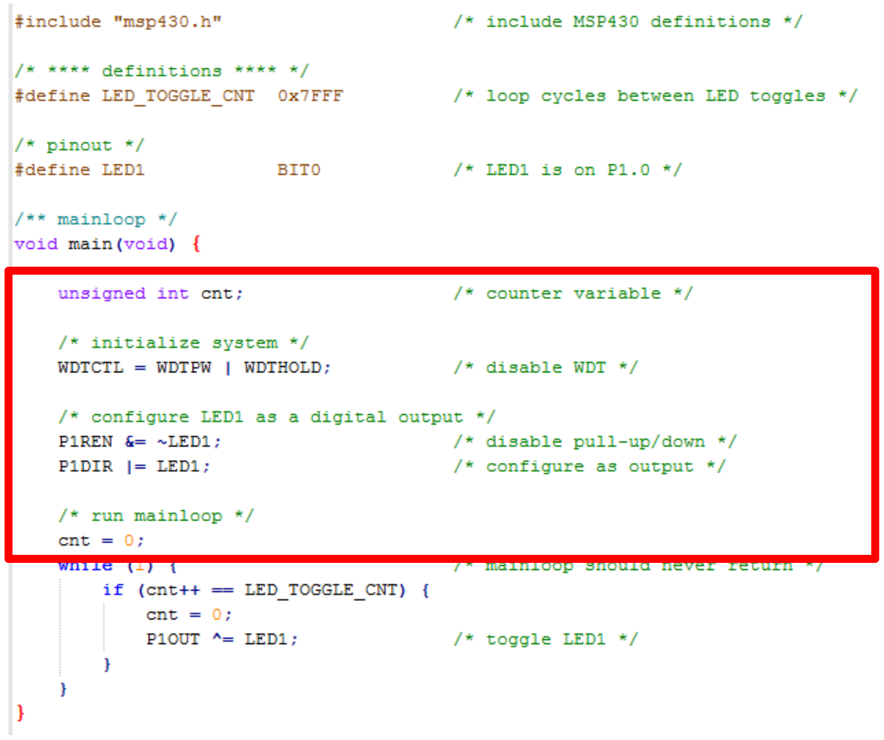
```
#include "msp430.h"                 /* include MSP430 definitions */

/* **** definitions **** */
#define LED_TOGGLE_CNT   0x7FFF      /* loop cycles between LED toggles */

/* pinout */
#define LED1             BIT0        /* LED1 is on P1.0 */

/** mainloop */
void main(void) {

    unsigned int cnt;                /* counter variable */

    /* initialize system */
    WDTCTL = WDTPW | WDTHOLD;        /* disable WDT */

    /* configure LED1 as a digital output */
    P1REN &= ~LED1;                  /* disable pull-up/down */
    P1DIR |= LED1;                   /* configure as output */

    /* run mainloop */
    cnt = 0;
    while (1) {                      /* mainloop should never return */
        if (cnt++ == LED_TOGGLE_CNT) {
            cnt = 0;
            P1OUT ^= LED1;           /* toggle LED1 */
        }
    }
}
```



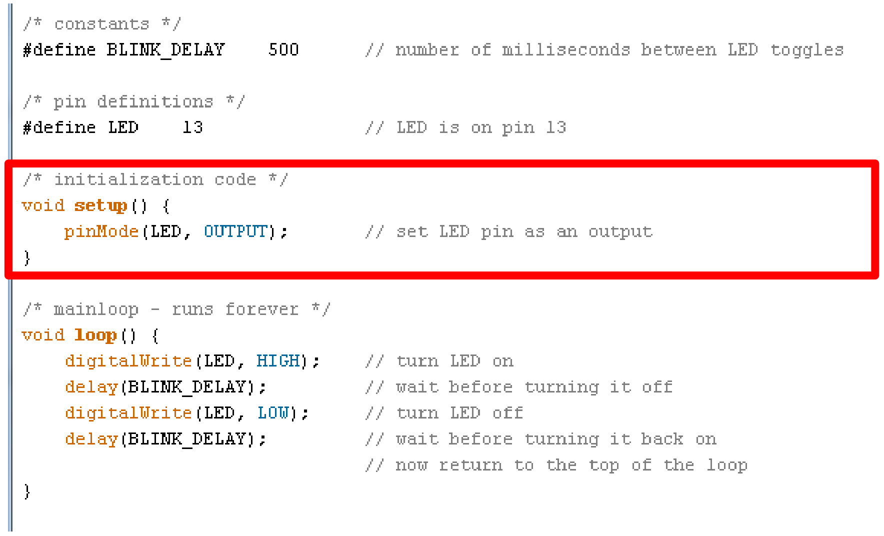
```
/* constants */
#define BLINK_DELAY    500       // number of milliseconds between LED toggles

/* pin definitions */
#define LED    13                // LED is on pin 13

/* initialization code */
void setup() {
    pinMode(LED, OUTPUT);        // set LED pin as an output
}

/* mainloop - runs forever */
void loop() {
    digitalWrite(LED, HIGH);     // turn LED on
    delay(BLINK_DELAY);          // wait before turning it off
    digitalWrite(LED, LOW);      // turn LED off
    delay(BLINK_DELAY);          // wait before turning it back on
                                 // now return to the top of the loop
}
```

MSP 430 vs. Arduino Code

Initialization Code
(run once at startup)



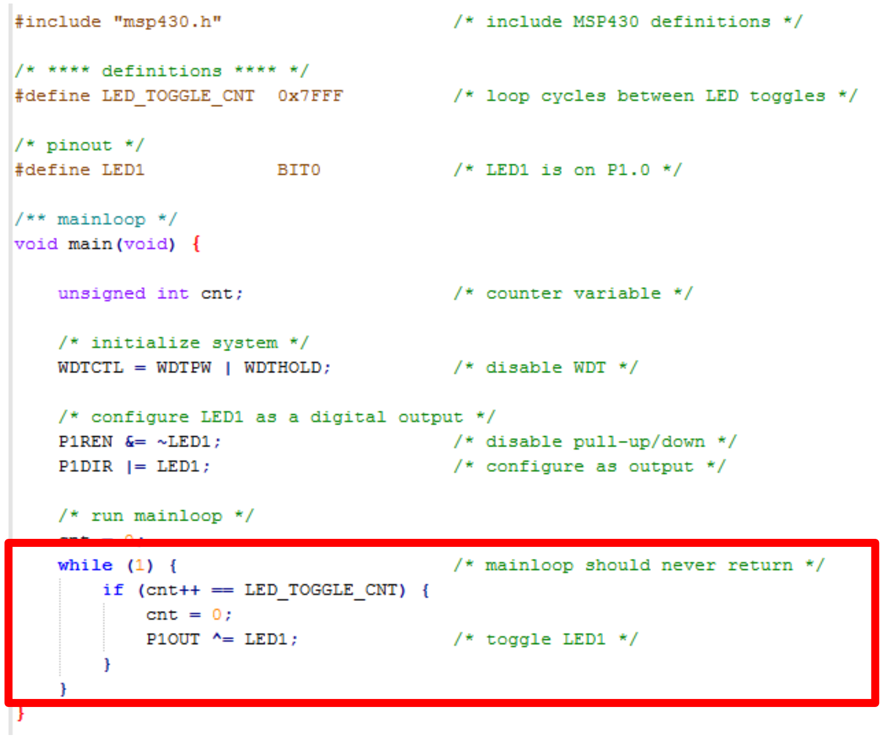
```
#include "msp430.h"                    /* include MSP430 definitions */

/* **** definitions **** */
#define LED_TOGGLE_CNT   0x7FFF        /* loop cycles between LED toggles */

/* pinout */
#define LED1             BIT0          /* LED1 is on P1.0 */

/** mainloop */
void main(void) {
    unsigned int cnt;                  /* counter variable */

    /* initialize system */
    WDTCTL = WDTPW | WDTHOLD;          /* disable WDT */

    /* configure LED1 as a digital output */
    P1REN &= ~LED1;                    /* disable pull-up/down */
    P1DIR |= LED1;                     /* configure as output */

    /* run mainloop */
    cnt = 0;
    while (1) {                        /* mainloop should never return */
        if (cnt++ == LED_TOGGLE_CNT) {
            cnt = 0;
            P1OUT ^= LED1;             /* toggle LED1 */
        }
    }
}
```



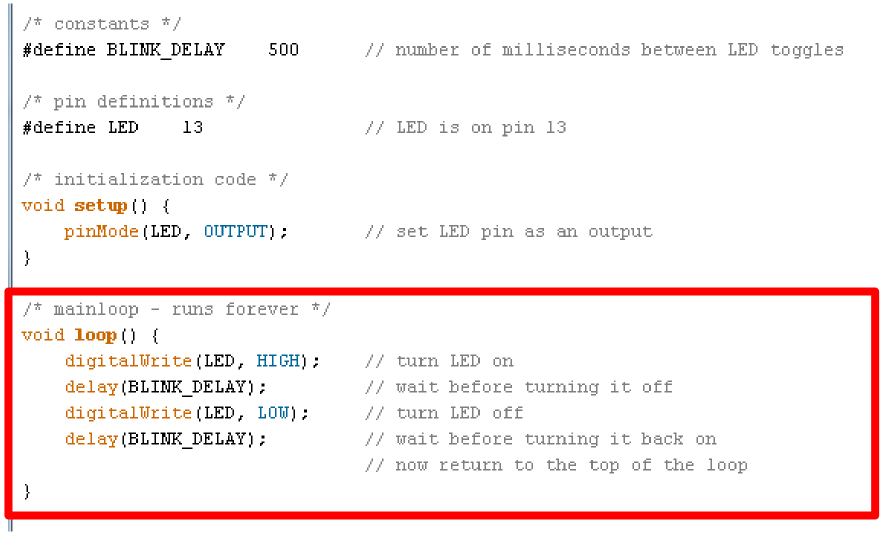
```
/* constants */
#define BLINK_DELAY    500      // number of milliseconds between LED toggles

/* pin definitions */
#define LED    13               // LED is on pin 13

/* initialization code */
void setup() {
    pinMode(LED, OUTPUT);       // set LED pin as an output
}

/* mainloop - runs forever */
void loop() {
    digitalWrite(LED, HIGH);    // turn LED on
    delay(BLINK_DELAY);         // wait before turning it off
    digitalWrite(LED, LOW);     // turn LED off
    delay(BLINK_DELAY);         // wait before turning it back on
                                // now return to the top of the loop
}
```

MSP 430 vs. Arduino Code

Mainloop
(runs in a loop forever)



```
#include "msp430.h"                  /* include MSP430 definitions */

/* **** definitions **** */
#define LED_TOGGLE_CNT   0x7FFF      /* loop cycles between LED toggles */

/* pinout */
#define LED1             BIT0        /* LED1 is on P1.0 */

/** mainloop */
void main(void) {

    unsigned int cnt;                /* counter variable */

    /* initialize system */
    WDTCTL = WDTPW | WDTHOLD;        /* disable WDT */

    /* configure LED1 as a digital output */
    P1REN &= ~LED1;                  /* disable pull-up/down */
    P1DIR |= LED1;                   /* configure as output */

    /* run mainloop */
    cnt = 0;
    while (1) {                      /* mainloop should never return */
        if (cnt++ == LED_TOGGLE_CNT) {
            cnt = 0;
            P1OUT ^= LED1;           /* toggle LED1 */
        }
    }
}
```



```
/* constants */
#define BLINK_DELAY    500      // number of milliseconds between LED toggles

/* pin definitions */
#define LED    13               // LED is on pin 13

/* initialization code */
void setup() {
    pinMode(LED, OUTPUT);       // set LED pin as an output
}

/* mainloop - runs forever */
void loop() {
    digitalWrite(LED, HIGH);    // turn LED on
    delay(BLINK_DELAY);         // wait before turning it off
    digitalWrite(LED, LOW);     // turn LED off
    delay(BLINK_DELAY);         // wait before turning it back on
                                // now return to the top of the loop
}
```

IAR Compiler Syntax

- Must include msp430.h

```
#include <msp430.h>
```

- To specify an interrupt routine:

```
#pragma vector=WDT_VECTOR
__interrupt void WDT_ISR(void)
```

- To enable global interrupts:

```
__enable_interrupt();
```

MSP 430 LaunchPad Demos

- **Hello World**

  Blinks an LED
- **Interrupts**

  Toggles one LED using timer interrupts and toggles other LED using user interrupts (when user presses a switch)
- **PWM**

  LED brightness changes continuously using PWM
- **ADC**

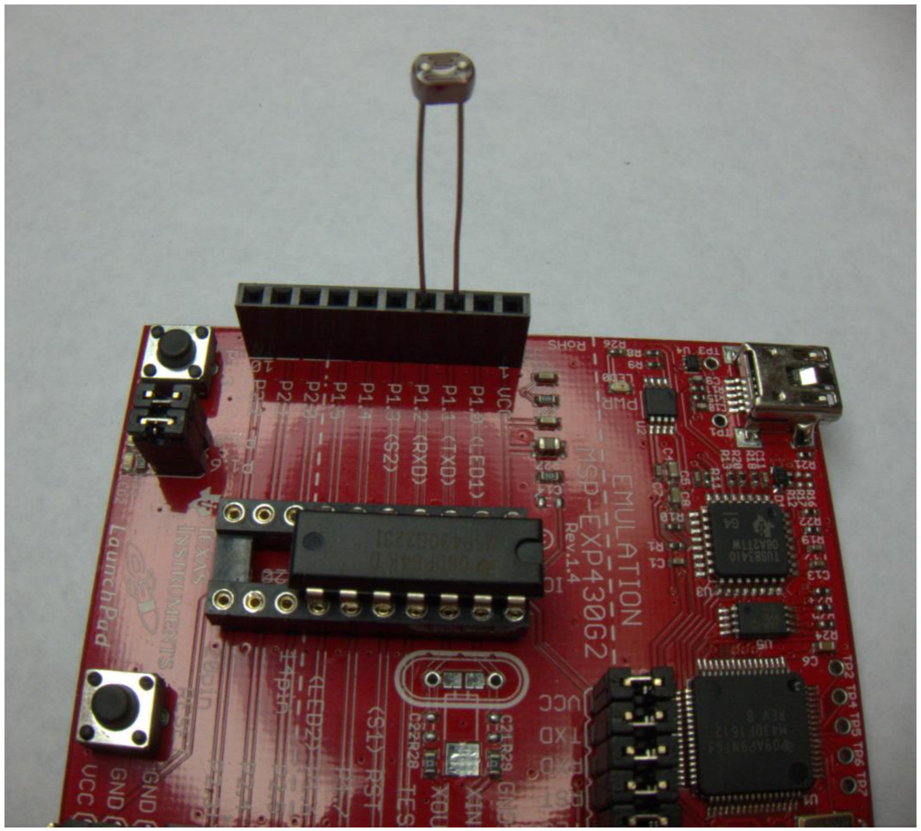
  Periodically samples voltage across light sensor and outputs brightness level using LEDs
- **Capacitive Sensing**

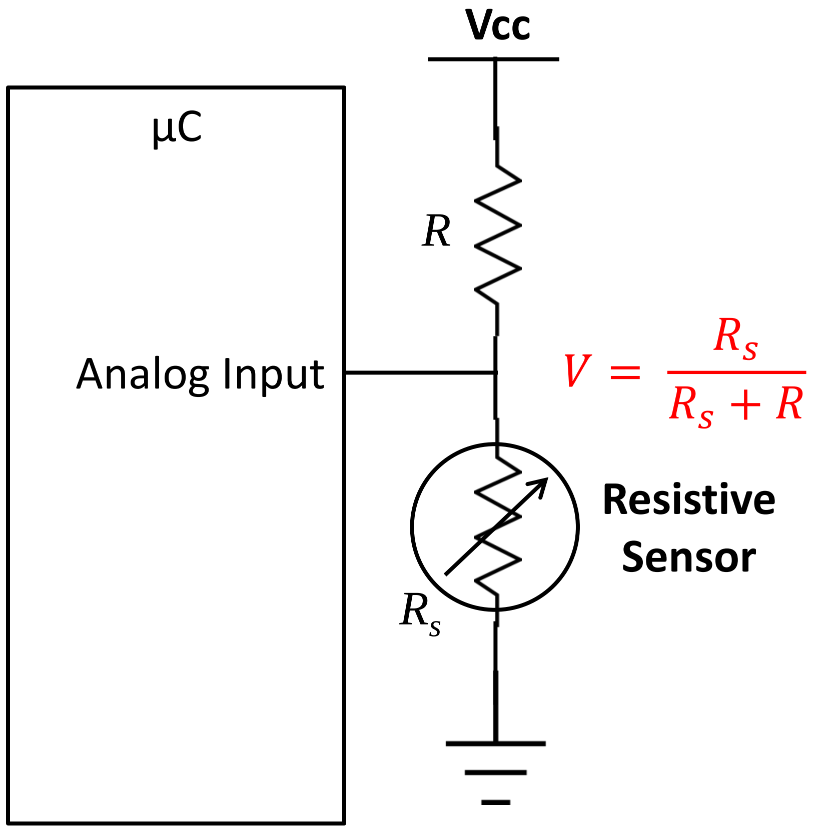
  Senses capacitance using Al foil and outputs user proximity on LED

- Code Available at: www.gabeacohn.com/teaching/micro

MSP430 LaunchPad ADC Demo

- Need to connect photo-resistor between P1.1 and P1.2





$$V = \frac{R_s}{R_s + R}$$

Capacitive Sensing Demo

- Capacitive Sensing in under $5!
- Parts:
  - MSP430 LaunchPad
  - 1 MΩ resistor
  - 47 pF ceramic capacitor
  - sheet of aluminum foil
  - 1 alligator clip
  - code: http://blog.hodgepig.org/2010/09/16/launchpad-capacitive-sensing/

Microcontroller Basics

www.gabeacohn.com/teaching/micro

Gabe Cohn