



# Embedded Systems

---

Fatima Anwar  
Networked & Embedded Systems Lab  
Aug 2-3, 2017

*Note: modified from original LACC 2016 slides by Paul Martin*



# We typically think of computers as...

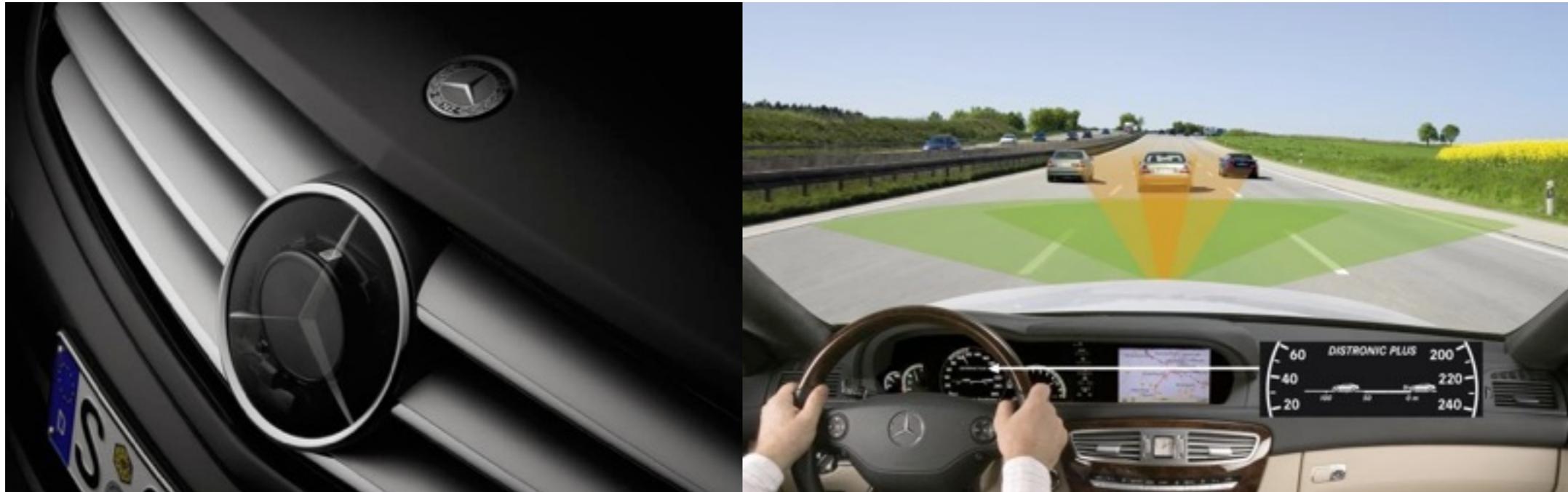
---



[www.amazon.com](http://www.amazon.com)



# Embedded systems that assist



- RADAR & ultrasonic sensors
- Compute distances & localize
- Assist users who are tired

1. <http://www.daimler.com/dccom/0-5-1210218-1-1210321-1-0-0-1210228-0-0-135-0-0-0-0-0-0.html>  
2. <http://www.planetbenz.com/category/news/page/200/>



# Mercedes: Learns how you drive

---

[http://www.youtube.com/watch?  
v=A66zgJ4Oj8o&list=UUfRUa1Z5gTknsMMaKLthZlg&in  
dex=21&feature=plpp](http://www.youtube.com/watch?v=A66zgJ4Oj8o&list=UUfRUa1Z5gTknsMMaKLthZlg&index=21&feature=plpp) video



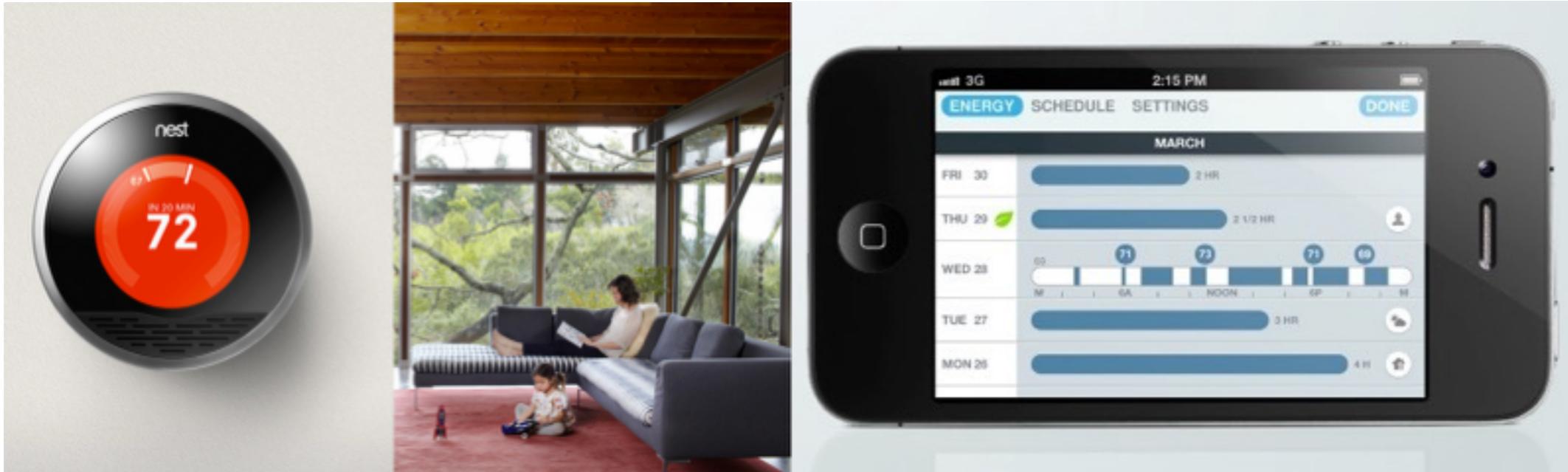
# Questions

---

- What are some of the key components that make this possible?
- What are some of the critical operations of this system?



# Embedded systems that monitor



- Measure environmental factors
- Quietly adapt & control
- Provide real time data to the users



#### Activity sensors

Nest's activity sensors have a 150° wide-angle view so Nest knows when to set itself to Auto-Away.



#### Humidity Sensor

Nest's humidity sensor activates Airwave. When indoor humidity is low, Airwave can cut your cooling costs up to 30%.



#### Weather aware

Nest uses its Wi-Fi connection to keep an eye on current weather conditions and forecasts so it can understand how the outside temperature affects your energy use.



#### Temperature Sensors

Three temperature sensors track your home's heating and cooling. A one-degree difference can reduce energy use up to 5%, so precision is important.

# NEST: The Learning Thermostat

---

<https://youtu.be/L8TkhHgkBsg>



# Questions

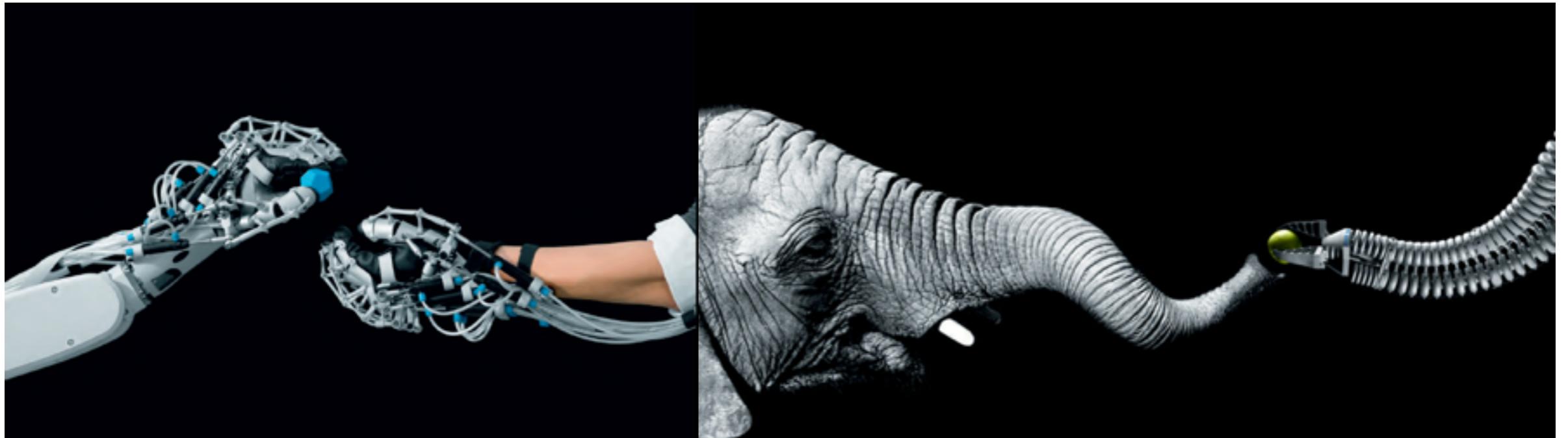
---

- How is this user interaction different from the Mercedes?
- What kinds of components might you need to make this work?

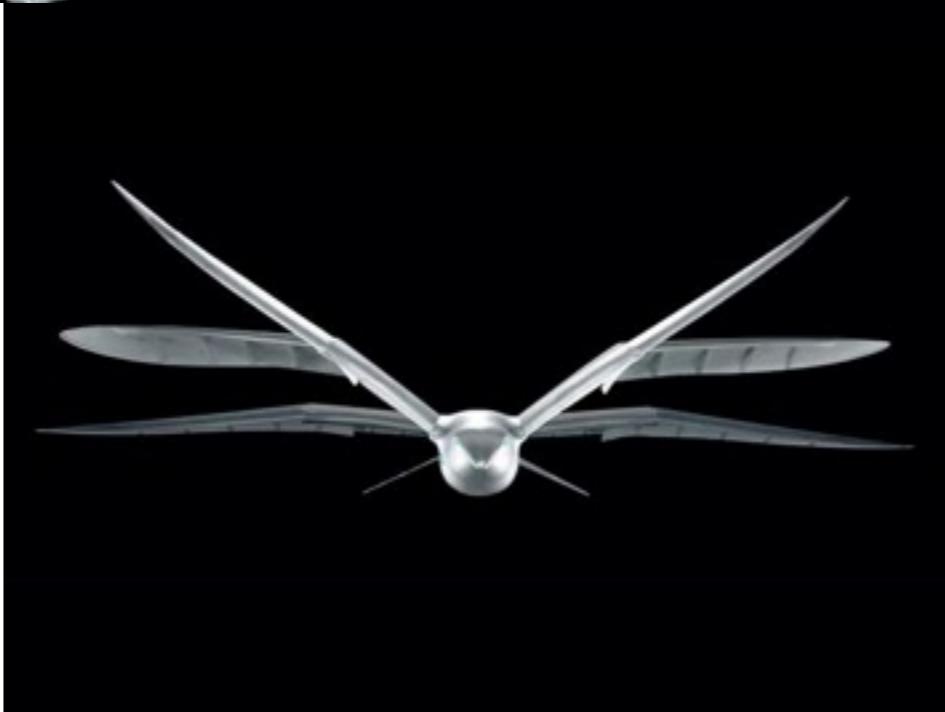


# Embedded systems that control

---



- Designed for complex motions
- Calculate angles & trajectories
- Accomplish a variety of tasks



TECH  
LAC

# FESTO: Robots that mimic

---

Vote on a video:

Bird

Kangaroo

Aqua Ray

Air Ray

Jellyfish

*Are you sure? The other  
ones are cooler...*



# Questions

---

- Why do we mimic nature?
- When does it make sense to mimic nature? When does it not?
- What are some common devices or inventions that mimic nature?



# Geckos help clean space junk

---

[https://www.facebook.com/pg/  
ajplusenglish/videos/?ref=page internal](https://www.facebook.com/pg/ajplusenglish/videos/?ref=page_internal)



# Embedded systems that care

---



- Food insecurity, supply shortage, difficulty in communication
- Use embedded systems to better distribute resources

# Embedded System or Cyber-Physical Systems

---

## A Definition:

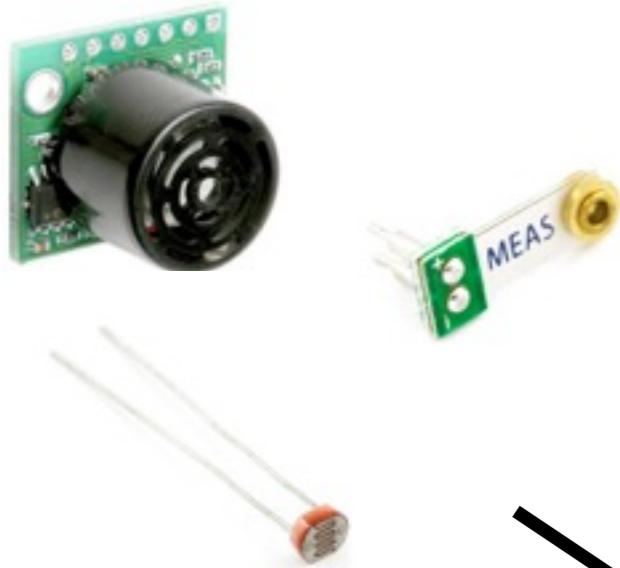
“A system of collaborating computational elements that sense or control physical entities”



# Embedded Systems Can do Anything!

---

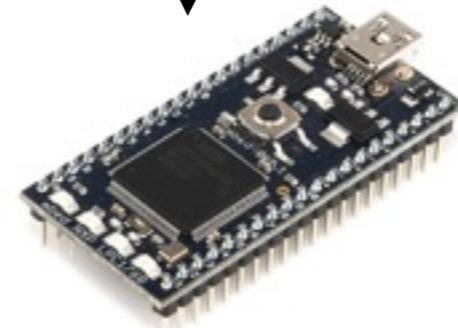
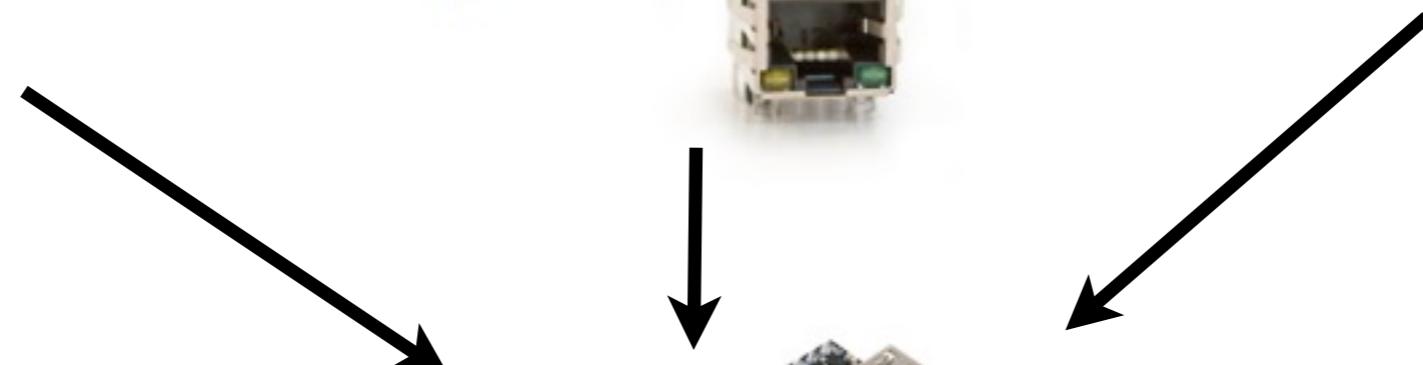
Sensors



Communication



Actuators



1. All images from [www.Sparkfun.com](http://www.Sparkfun.com)

Computer / Processor



# Stanford's Stanley: Autonomous Vehicle

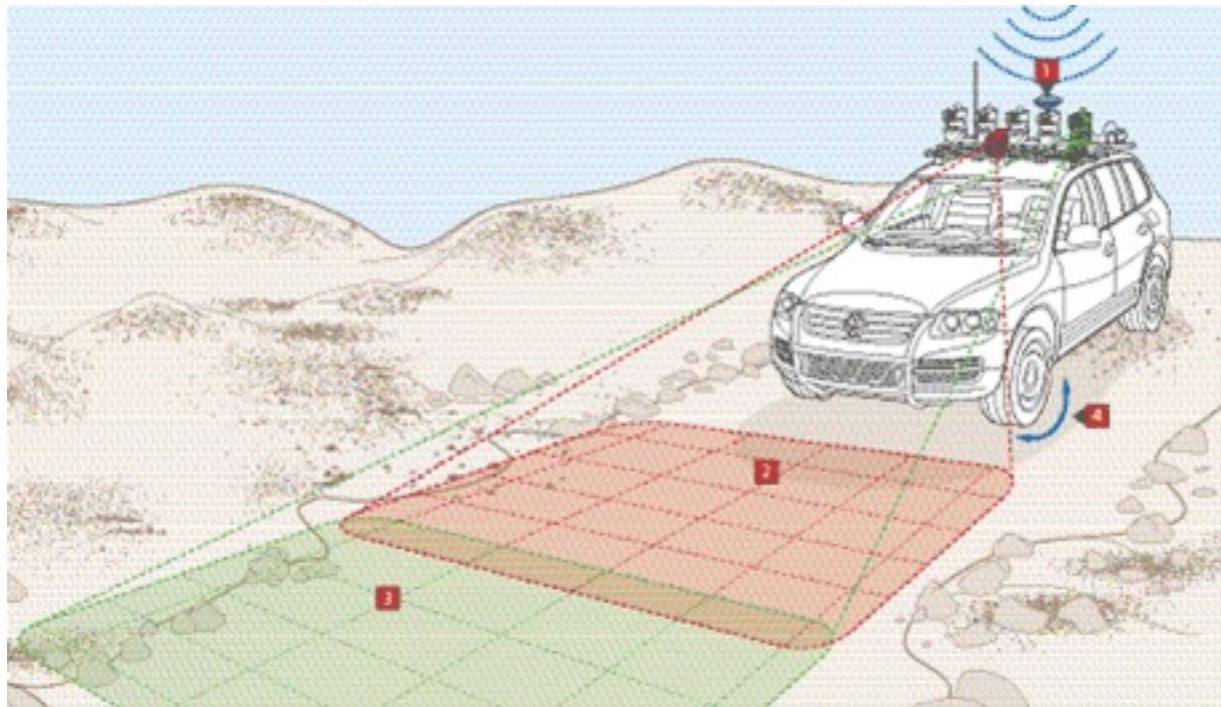
---



- GPS
- SICK Laser Scanner
- Stereo Camera
- Monocular Camera
- 6DOF Inertial Sensors
- RADAR
- Speed sensor
- Pentium M
- Drive-by-Wire
- Battery System

1. <http://cs.stanford.edu/group/roadrunner//old/technology.html>

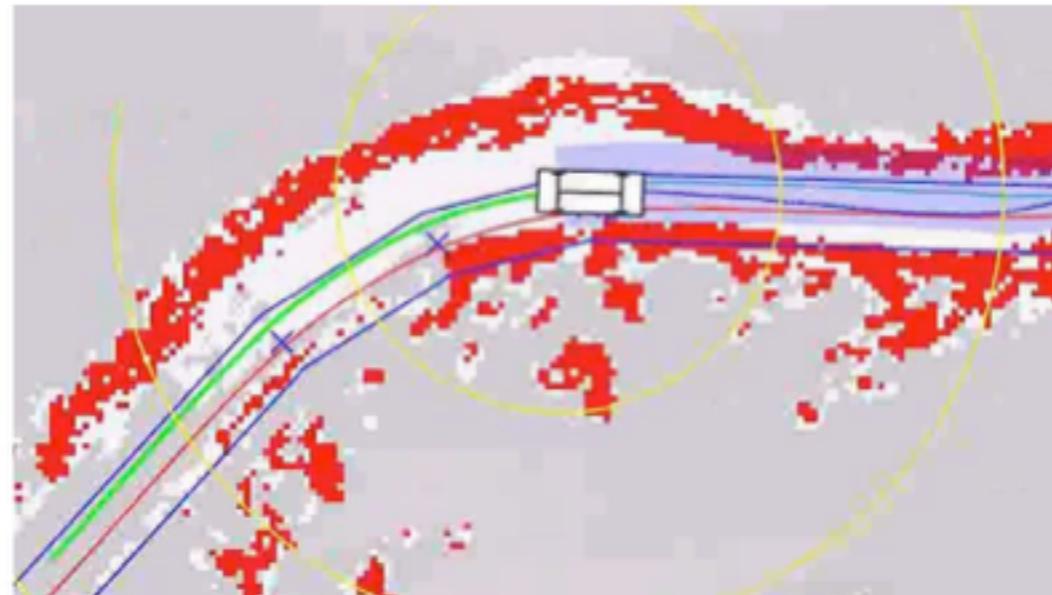
# Stanford's Stanley: Probabilistic Terrain Analysis



**(a) Beer Bottle Pass**

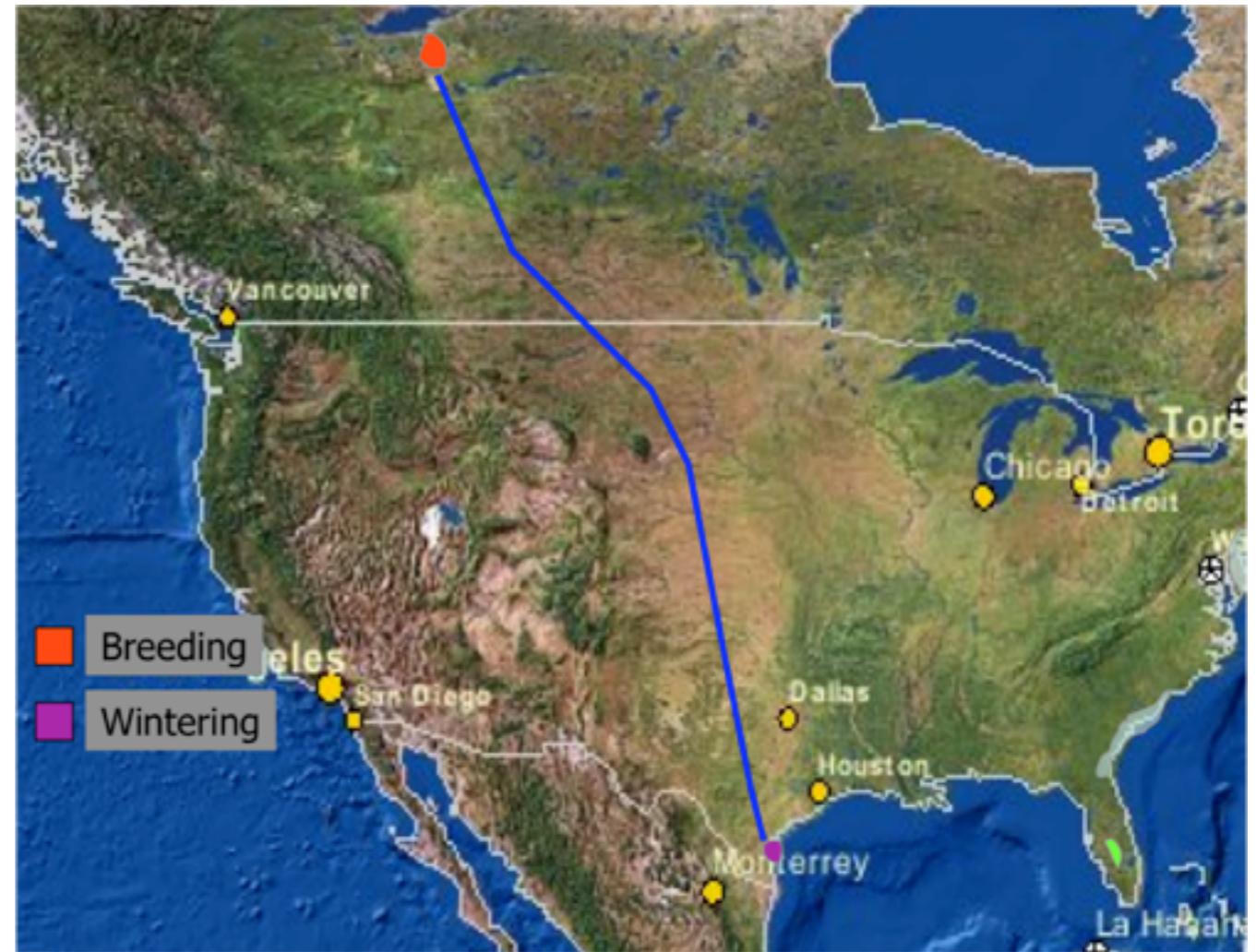
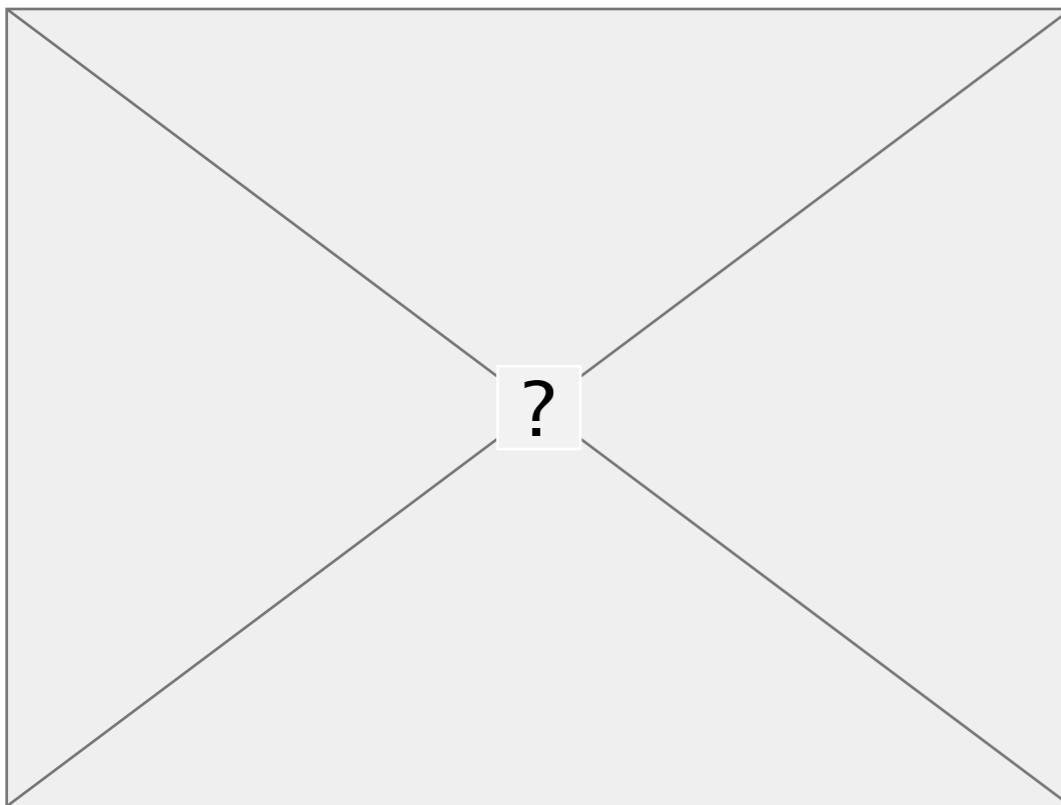
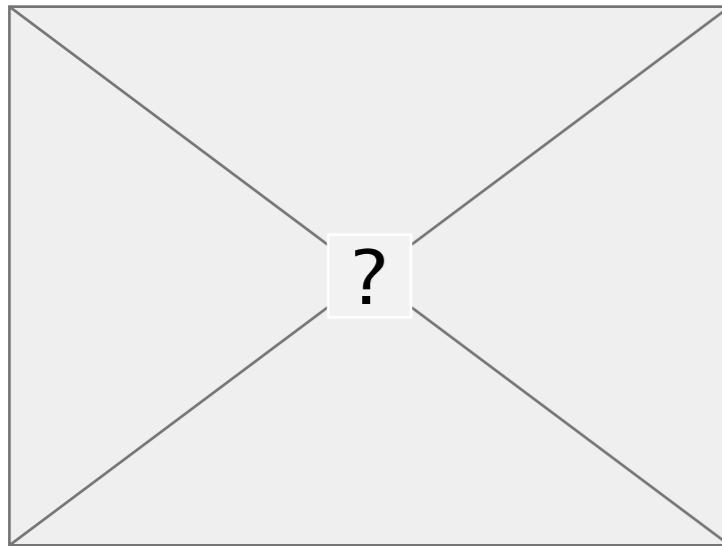
**(b) Map and GPS corridor**

Scan terrain and collect data  
↓  
Run through probabilistic models  
↓  
Calculate paths & obstacles



red: obstacles  
clear: paths  
gray: unknown

# Tracking Cranes

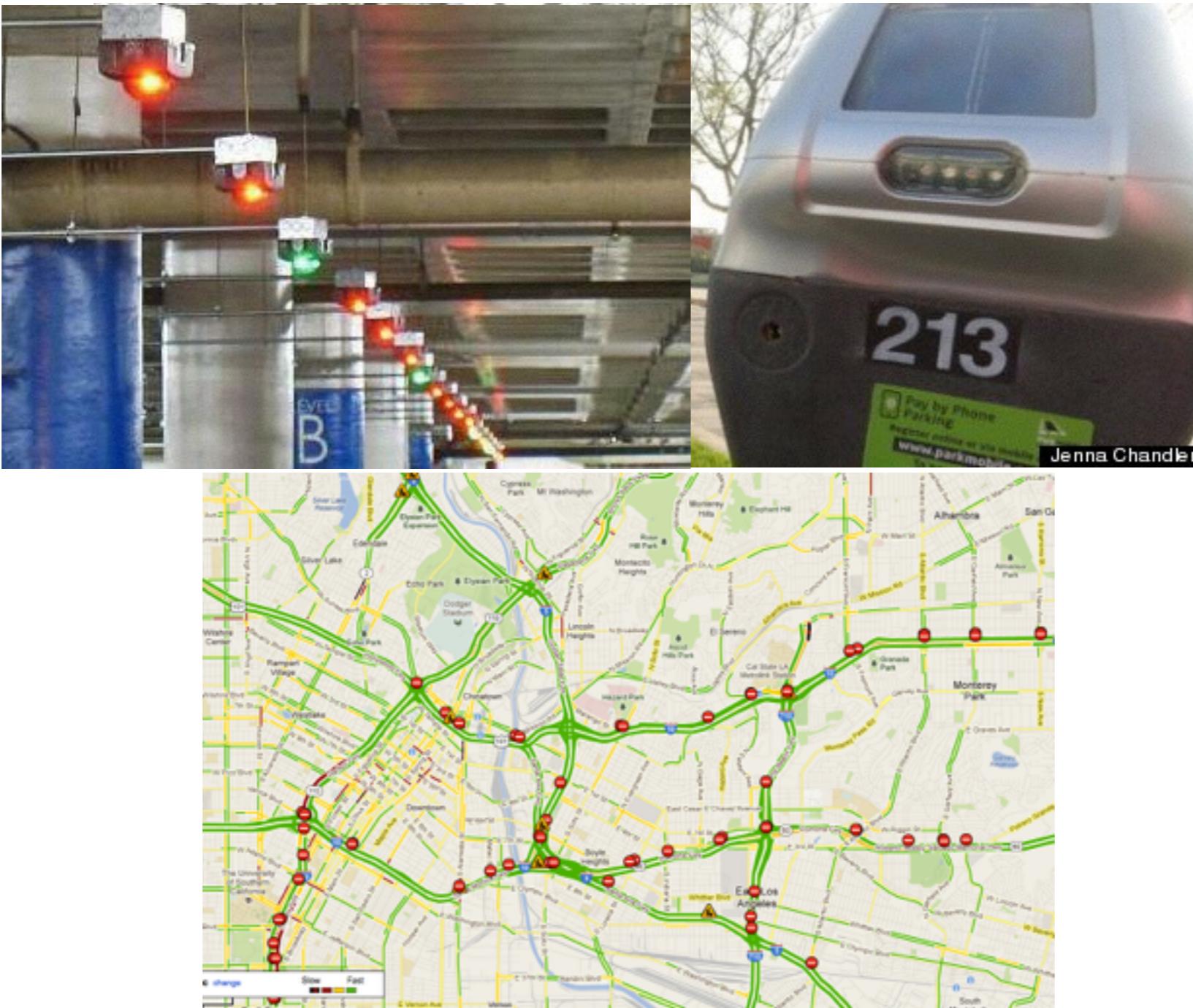


## cellular + shortwave radio

University of Nebraska: Cyber Physical Networking Lab



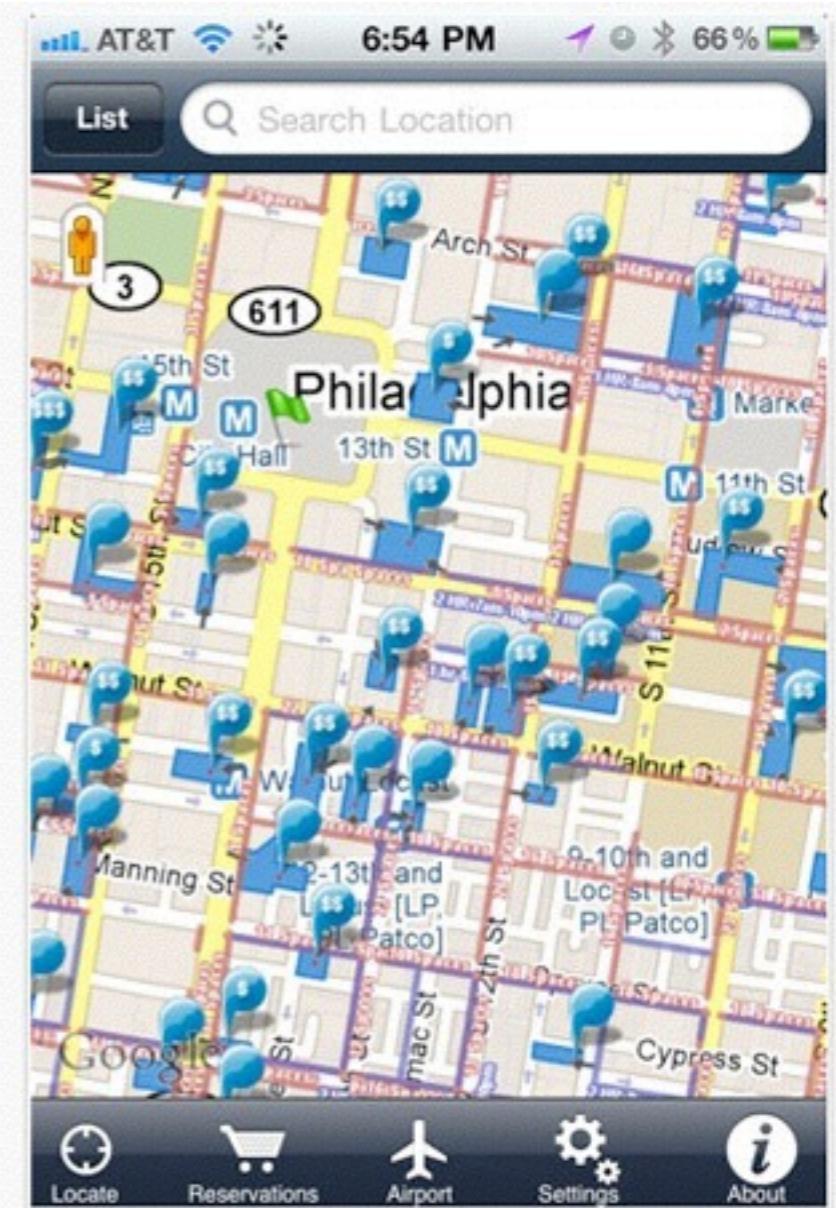
# Santa Monica Parking Sensors



# Information at your fingertips...



**parking in motion**  
a smarter way to park

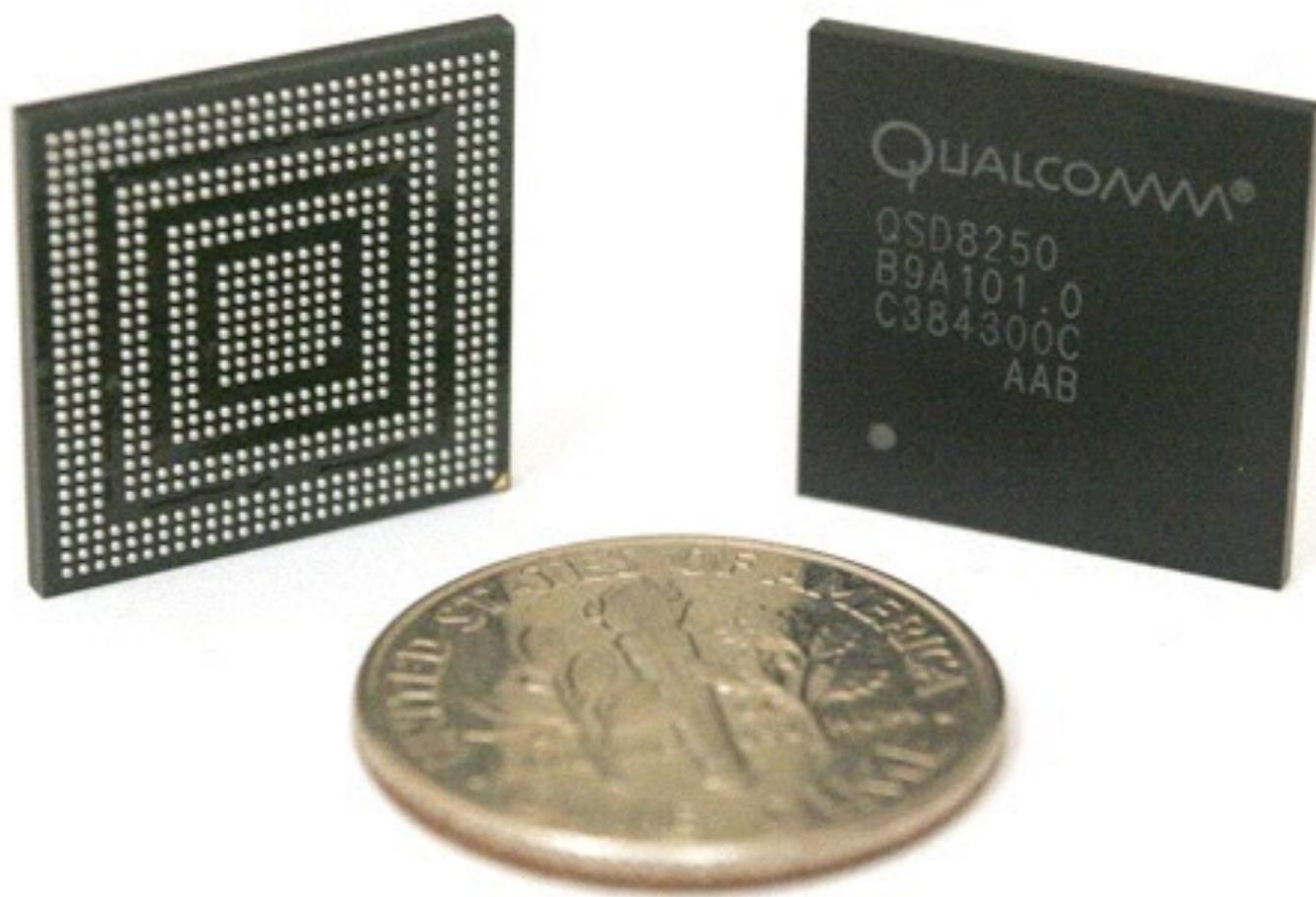


# Embedded Computing



# Embedded Processor

---



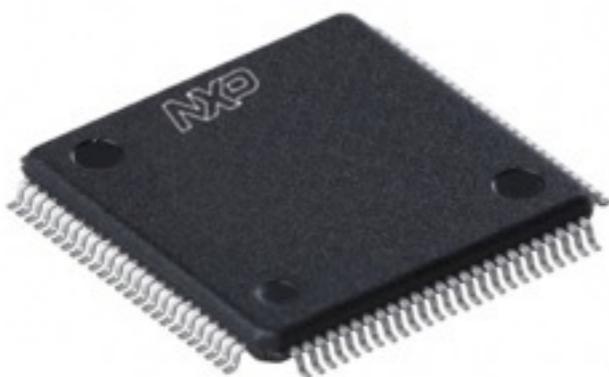
# Embedded Processor: Comparison

---



## Intel Core i7

- Speeds: 3-4 GHz
- Cores: 4+
- Memory DDR3: 32 GB
- Power: 50 ~300 Watts

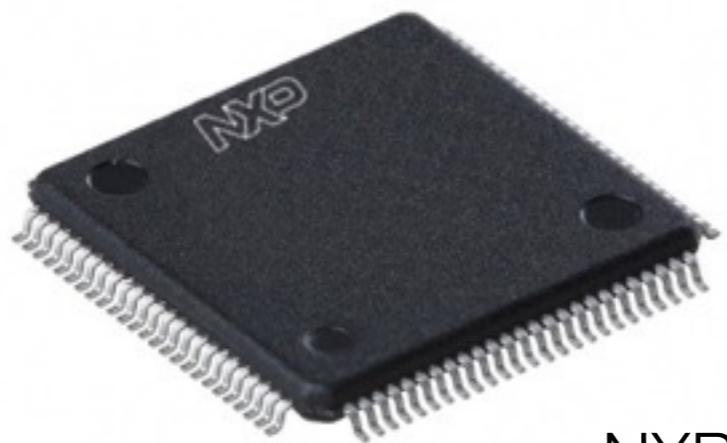


## ARM Cortex-M3

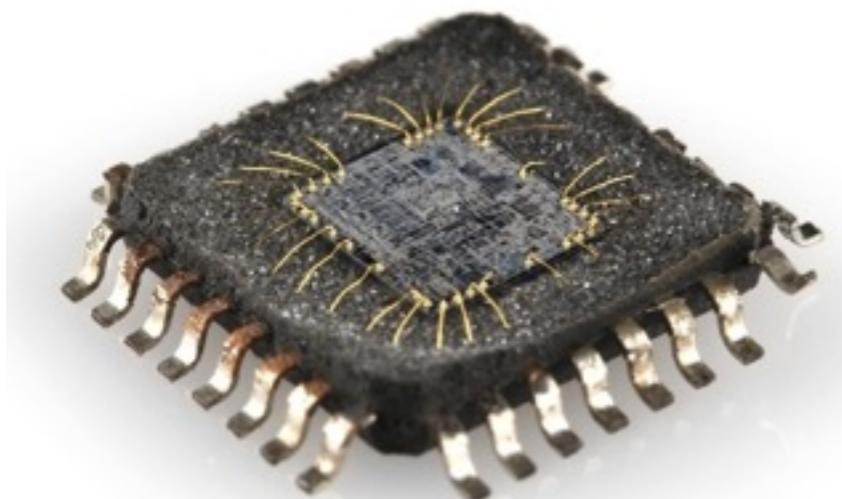
- Speeds: 100 MHz
- Cores: 1
- Memory SRAM: 64 KB
- Power: ~0.5 W

# Embedded Processor: Guts

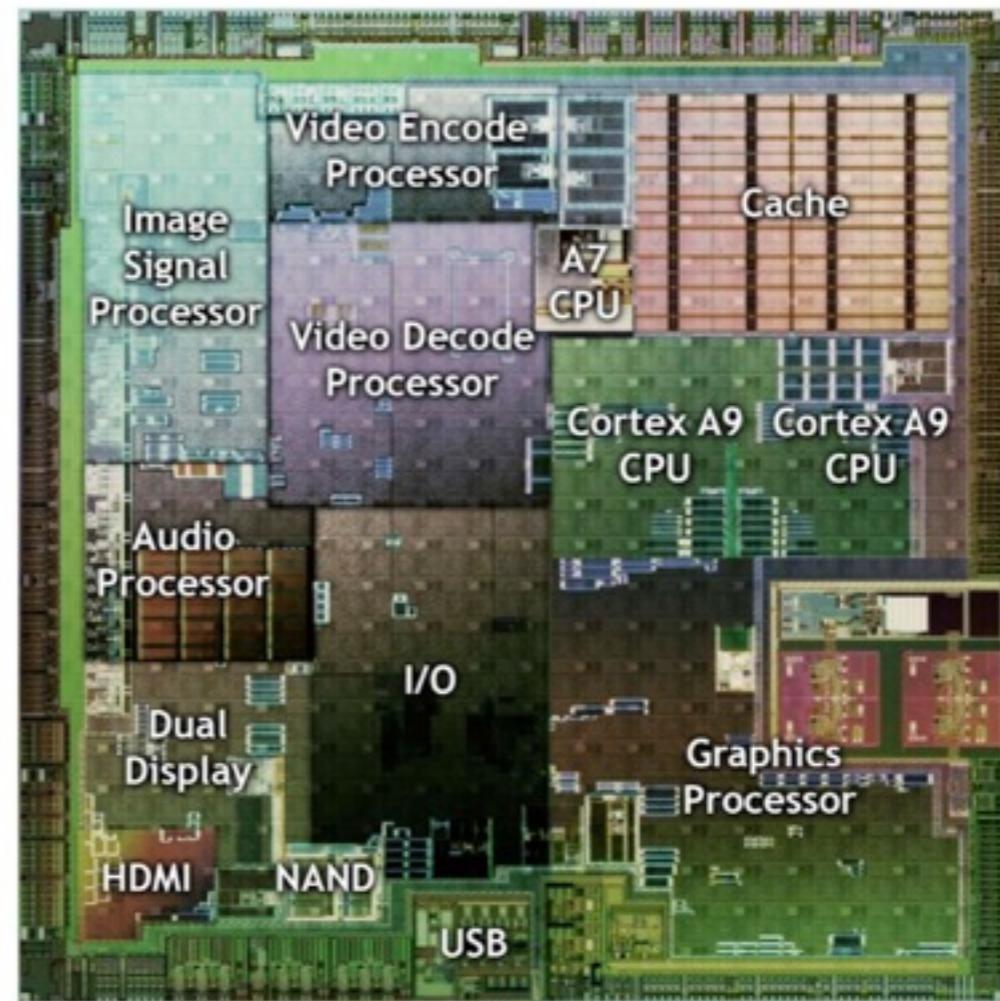
---



NXP



Atmega

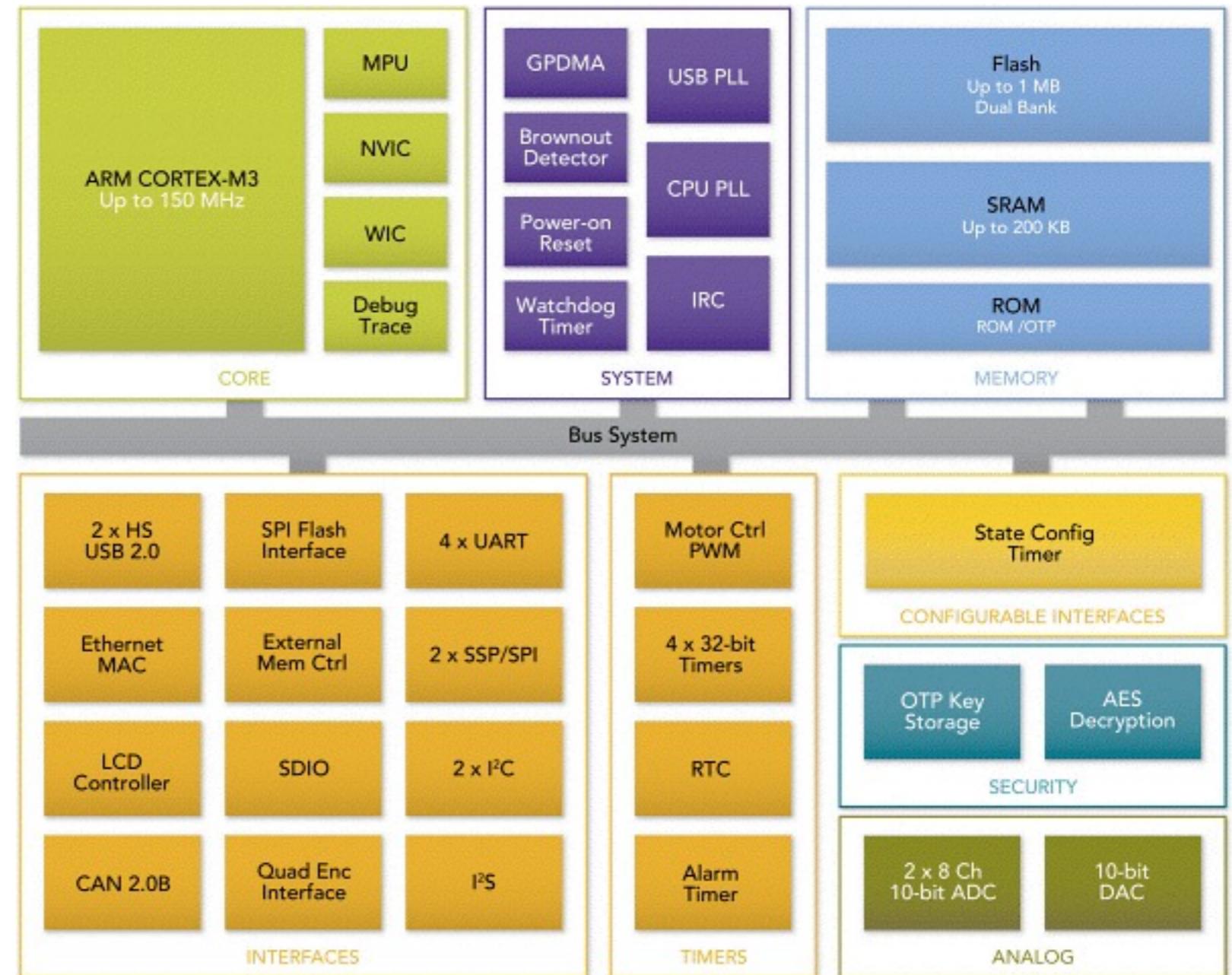


Tegra 2



# Embedded Processor: LPC1768 Block Diagram

- Core computing unit
- Memory stack
- Clocks & timers
- Digital I/O
- Analog I/O
- Communication
  - USB
  - Serial
  - Ethernet
- System drivers
  - Touch screens
  - audio
  - :
  - :



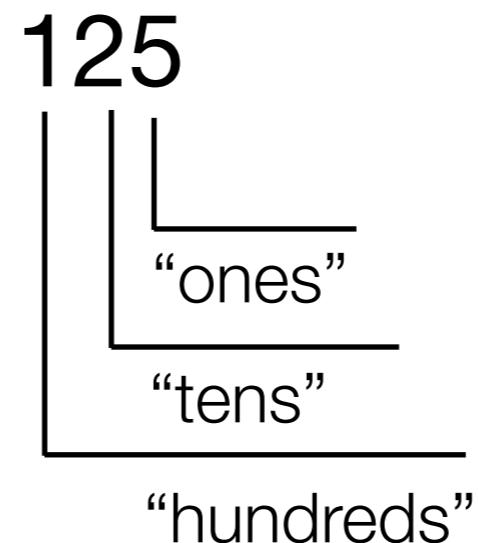
# Number Systems



# Binary Number Systems: *a primer*

---

- What do we mean when we say we use a “Decimal” number system?

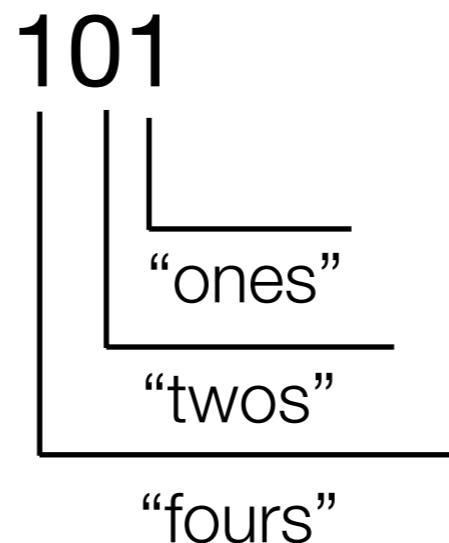


$$1(10^2) + 2(10^1) + 5(10^0) = 100 + 20 + 5$$

# Binary Number Systems: *a primer*

---

- What do we mean when we refer to a “Binary” number system?



$$1(2^2) + 0(2^1) + 1(2^0) = 4 + 0 + 1$$

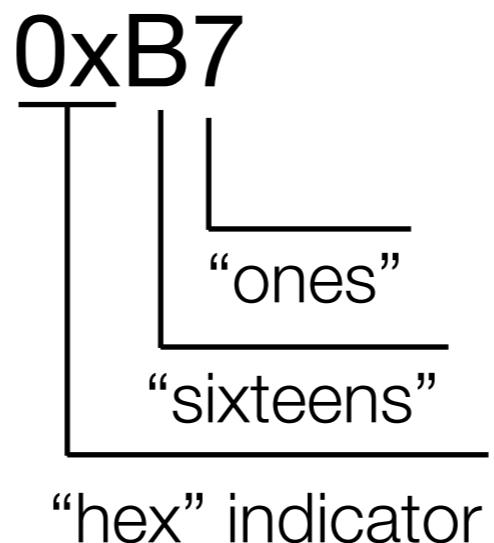
Any **base** can work:

- Base 8: “octal”
- Base 16: “hexadecimal”

# Binary Number Systems: *a primer*

---

- What do we mean when we refer to a “Hexadecimal” number system?



$$11(16^1) + 7(16^0) = 176 + 7 = 183$$

Digits after 0...9 are lettered A...F

# Bits, Bytes, & Binary

---

- A “bit” is just a “binary digit” – either a 0 or a 1
- A “Byte” is a string of 8 “bits,” – 01011010 etc., with 256 combinations
- How many Bytes is a kilobyte (1 kB)?  
Answer: 1024, or  $2^{10}$ . A megabyte is  $2^{20}$  and a gigabyte is  $2^{30}$ .

*“There are 10 kinds of people in the world: those who understand binary, and those who don’t.”*



# Storing Numbers in a Computer

---

- Storing an integer
  - Signed integers? How many bits?
- Storing decimal numbers
  - Fixed point decimals
  - Floating point and dynamic range
- How does a computer store a String (of characters)?



# Communicating Digital Information



# Communication

---

- Embedded Systems are composed of many parts, often physically separated
- How can we communicate information from one “subsystem” to another?
  - With wires? Without wires (another lecture)?
- What are some of the design considerations when communicating between two computers?

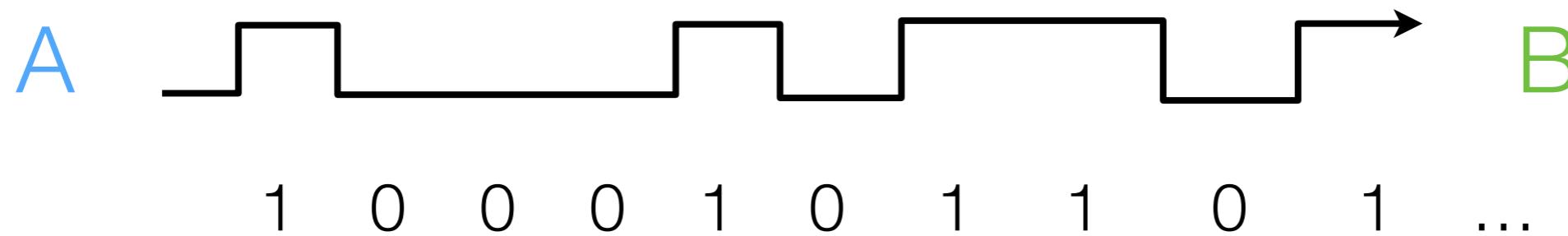


# Wired Digital Communication

---

- (Embedded) computer A wants to tell computer B what temperature it is. How might it do this using a few wires?

One way:

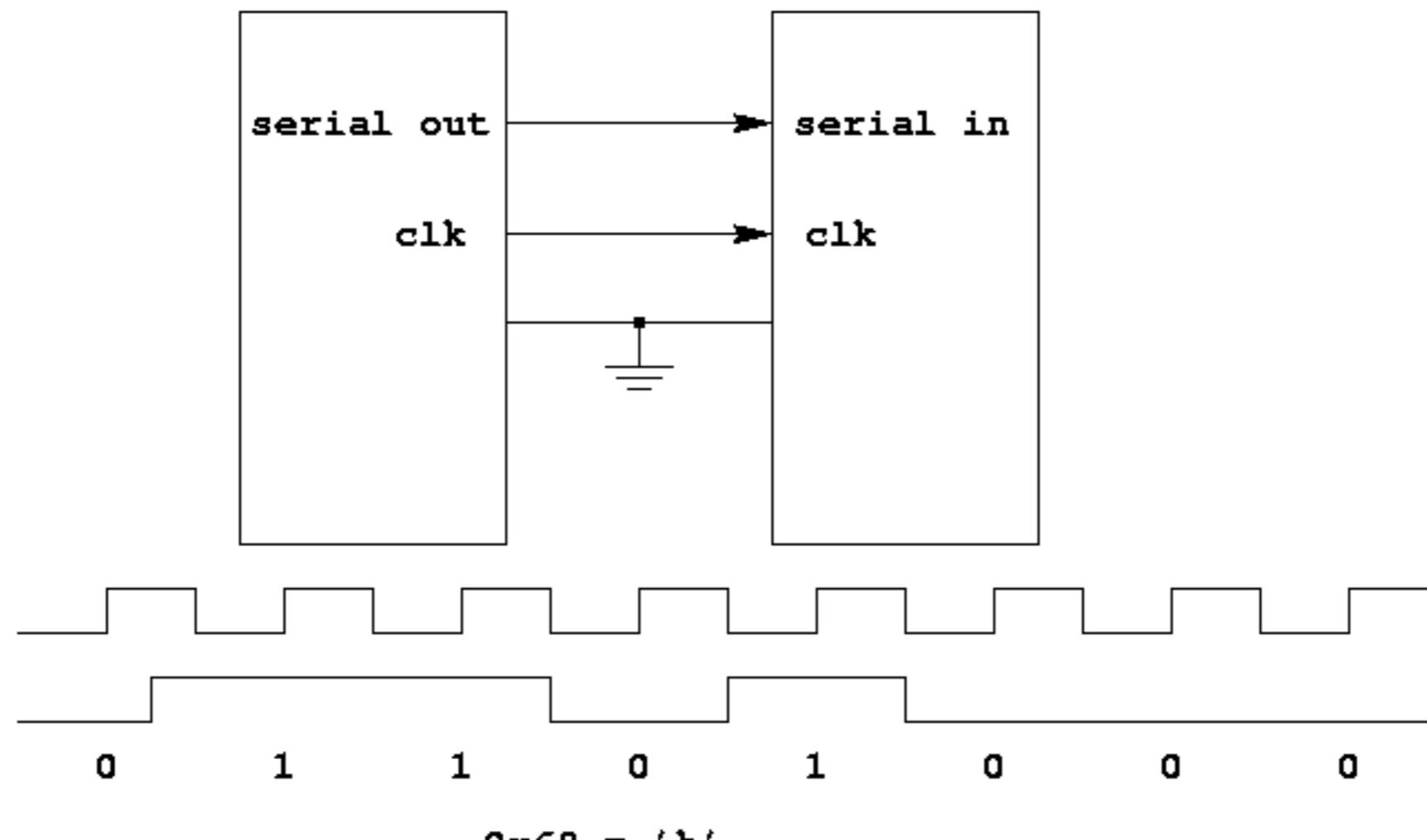


B must sample this wire at even intervals, at a rate that A and B agree upon. This is called “serial” communication — data is communicated serially.

For more information, look up UART, SPI, I<sup>2</sup>C, & CAN protocols.

# Synchronous Communication

## SERIAL COMMUNICATIONS

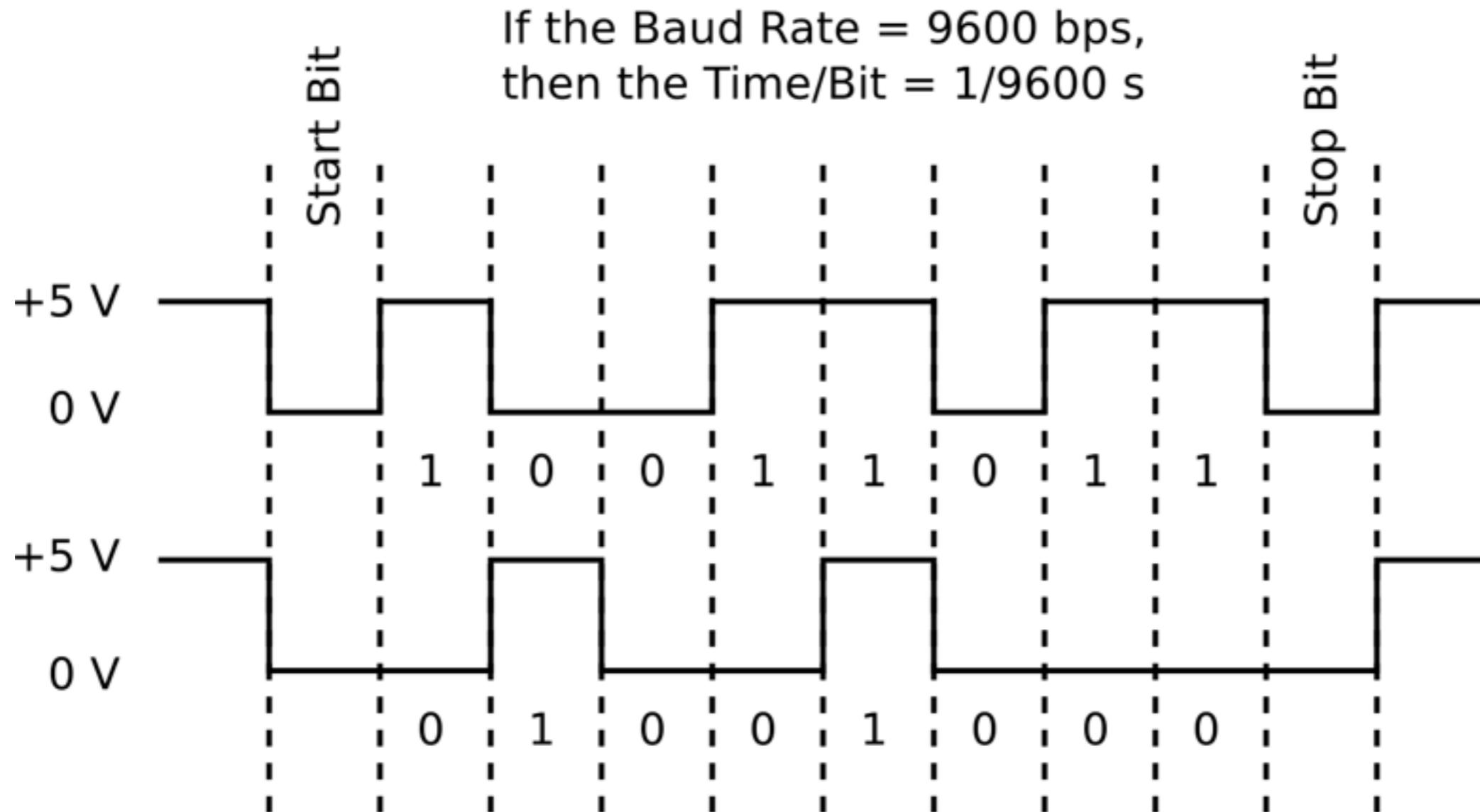


Need 3 wires to transmit 1 bit at a time

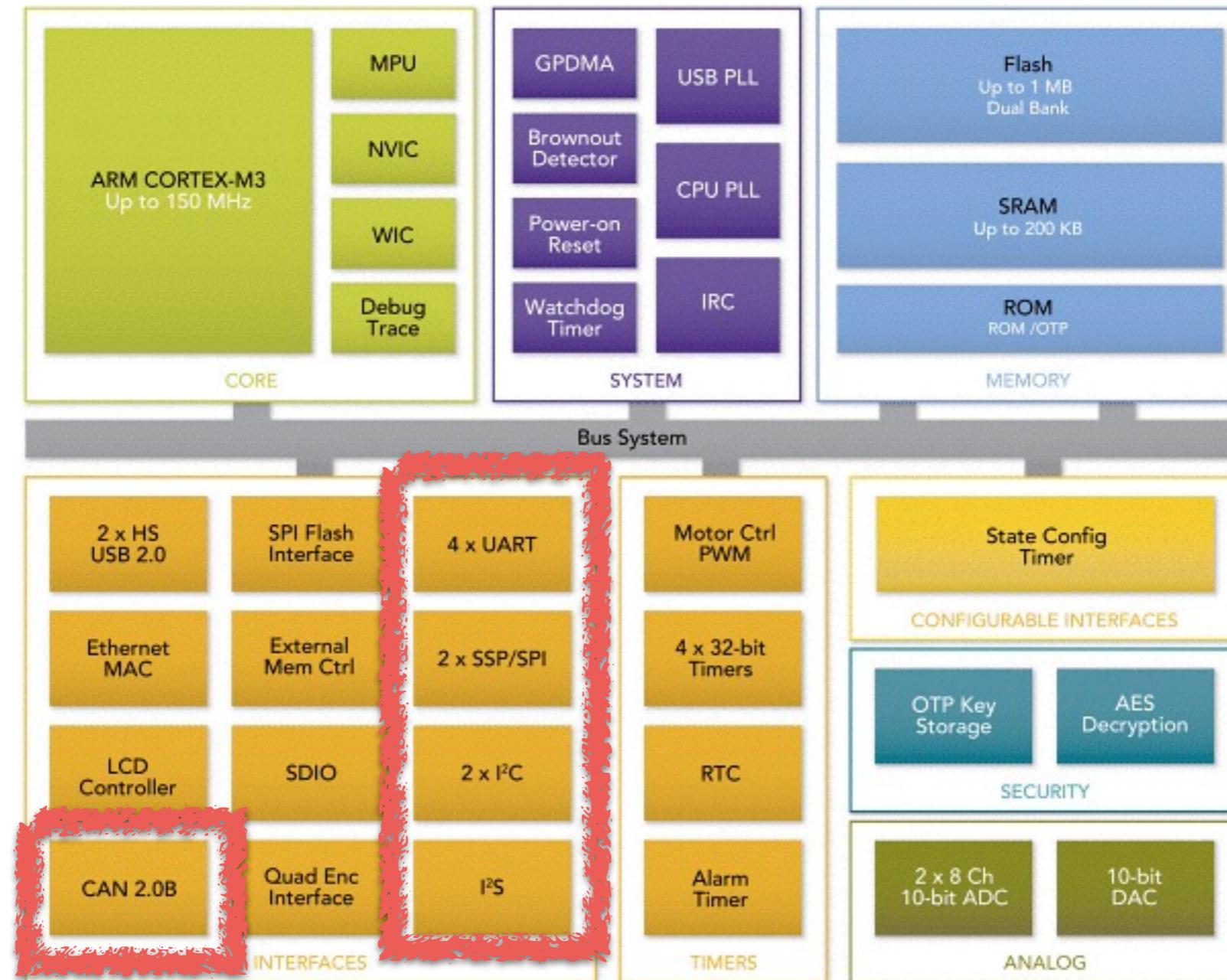
[http://www.ee.nmt.edu/~rison/ee308\\_spr99/lectures.html](http://www.ee.nmt.edu/~rison/ee308_spr99/lectures.html)



# Serial “Word”



# Embedded Processor: LPC1768 Block Diagram

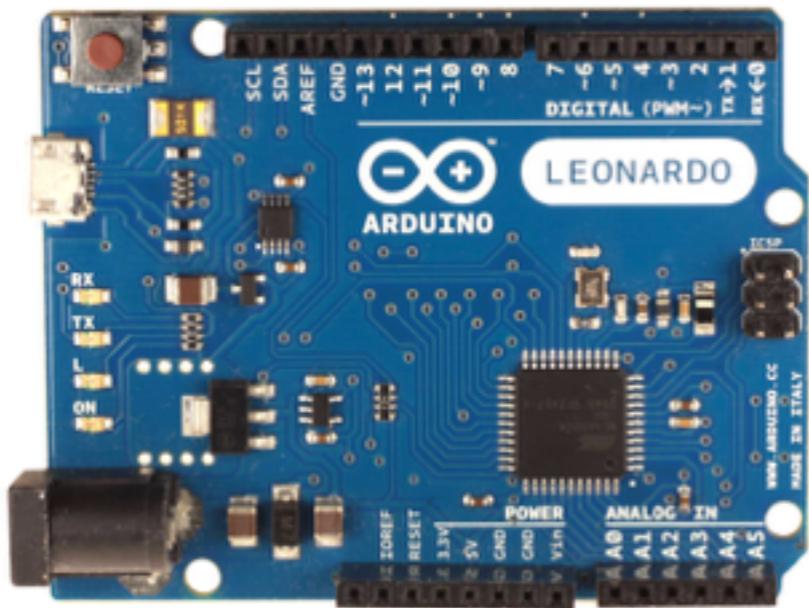
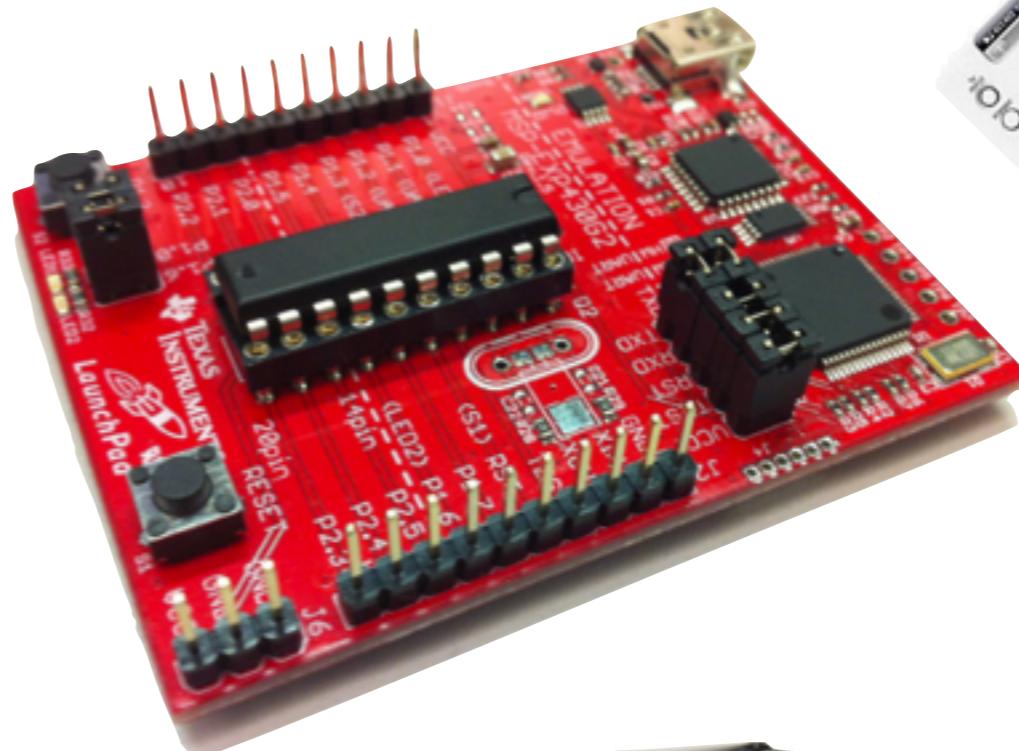
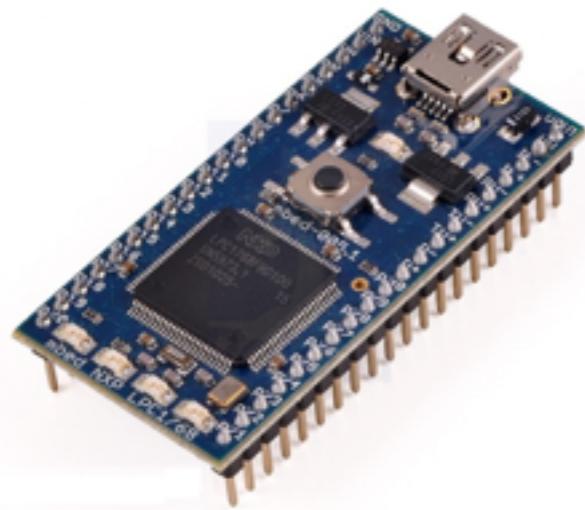


# How do I develop an Embedded System?



# Typical *Embedded Platforms*

---



# Typical Development Environments

Mikro-Elec AVR Studio

File Edit View Wizards Project Build Debug Tools Window Help

PC-Anl Analysis Status

Show: Mikro-Elec (19 issues; 41% analysed)

Analysis Status

File	Analysis Status	Total Issues
avrmain.c	Complete	19
glfont.c	Complete	0
oled.c	Complete	0
shake.c	Complete	0
wifi205.c	Complete	0
bmp05.c	Not analysed	n/a
status.c	Not analysed	n/a
Mem.c	Not analysed	n/a
sampling.c	Not analysed	n/a
playaudio.c	Not analysed	n/a

PC-Anl Analysis Results

Project: Mikro-Elec File: avrmain.c Status: Complete; Total Issues: 19

Category	ID	Source File	Line	Text
Information	701	C:\Documents\avrmain.c	15	Shift left of sign
Information	701	C:\Documents\avrmain.c	16	Shift left of sign
Information	701	C:\Documents\avrmain.c	42	Shift left of sign
Information	701	C:\Documents\avrmain.c	43	Shift left of sign
Warning	500	C:\Documents\avrmain.c	45	Symbol result
Information	830	C:\Documents\avrmain.c	18	Location cited in
Directive Note	953	C:\Documents\avrmain.c	46	Variable result
Information	830	C:\Documents\avrmain.c	19	Location cited in
Information	490	C:\Documents\avrmain.c	20	location of result

Terminal

```
def open
  @imap = Net::IMAP.new(@imap_server, :imap_port, true, nil, false)
  log @imap.login(@username, @password)
  list @mailboxes = prefetch @mailboxes.list
end

# expects a block, closes on finish
def with_open
  @imap = Net::IMAP.new(@imap_server, :imap_port, true, nil, false)
  log @imap.login(@username, @password)
  yield self
  close
end

def close
  log "Closing connection"
  Timeout.timeout(5) do
    begin
      @imap.close(true)
    rescue Net::IMAP::BadResponseError
      @imap.disconnect rescue IOError
    end
  rescue Timeout::Error
  end
end

def select_mailbox(mailbox, force=false)
  if mailbox.allboxes[mailbox]
    mailbox = mailbox.allboxes[mailbox]
  end
  log "Selecting mailbox #{@mailbox.inspect}"
  reconnect_if_necessary do
    log @imap.select(mailbox)
  end
  log 'Done'

  @mailbox = mailbox
  @label = Label.new(@mailbox) || Label.create(name: @mailbox)

  log 'Setting mailbox status'
  get_mailbox_status
  log 'Setting highest message id'
  get_highest_message_id
  return 'ok'
end

def reload_mailbox
  return unless @imap.thru?
  select_mailbox(@mailbox, true)
end
```

C:\Documents and Settings\Anna\Desktop\backcountry\_looper\_20110814\avrmain.c: 66 issues; last

mbed Compiler - /TextLCD\_HelloWorld/main.cpp

New Import Save Save All Compile Commit Revisions

Program Workspace

- My Programs
  - AD7490\_example
  - Default\_Program
  - HTTPServerHelloWorld
  - NetServices\_HelloWorld
  - NTPClient\_HelloWorld
  - PS3\_BlueUSB
  - TextLCD\_HelloWorld
    - TextLCD
      - Classes
        - TextLCD.cpp
        - TextLCD.h

TextLCD.h x main.cpp x

```
1 // Hello World! for the TextLCD
2
3 #include "mbed.h"
4 #include "TextLCD.h"
5
6 TextLCD lcd(p15, p16, p17, p18, p19, p20); // rs,
7
8 int main() {
9     lcd.printf("Hello World!\n");
10 }
```

ActionController::helpers

```
+ includes:
ActionController::Helpers (from gem actionpack-3.0.5)
ActionController::Helpers (from gem actionpack-3.0.5)

The Rails framework provides a large number of helpers for working with
assets, dates, forms, numbers and model
objects, to name a few. These helpers are available to all templates by
default.

In addition to using the standard template helpers provided in the Rails
framework, creating custom helpers to extract complicated logic or reusable
functionality is strongly encouraged. By default, the controller will include
a helper whose name matches that of the controller. E.g.,
MyController will automatically include MyHelper.

Additional helpers can be specified using the helper class method in
ActionController::Base or any controller which inherits from it.

==== Examples
The to_s method from the Time class can be wrapped in a helper method
to display a custom message if the time object is blank:

  module FormattedTimeHelper
    def format_time(time, format='long', blank_message='<br>')
      time.blank? ? blank_message : time.to_s(format)
    end
  end

FormattedTimeHelper can now be included in a controller, using the
helper class method:

  class EventsController < ActionController::Base
    helper FormattedTimeHelper
    def index
      @events = Event.find(:all)
    end
  end

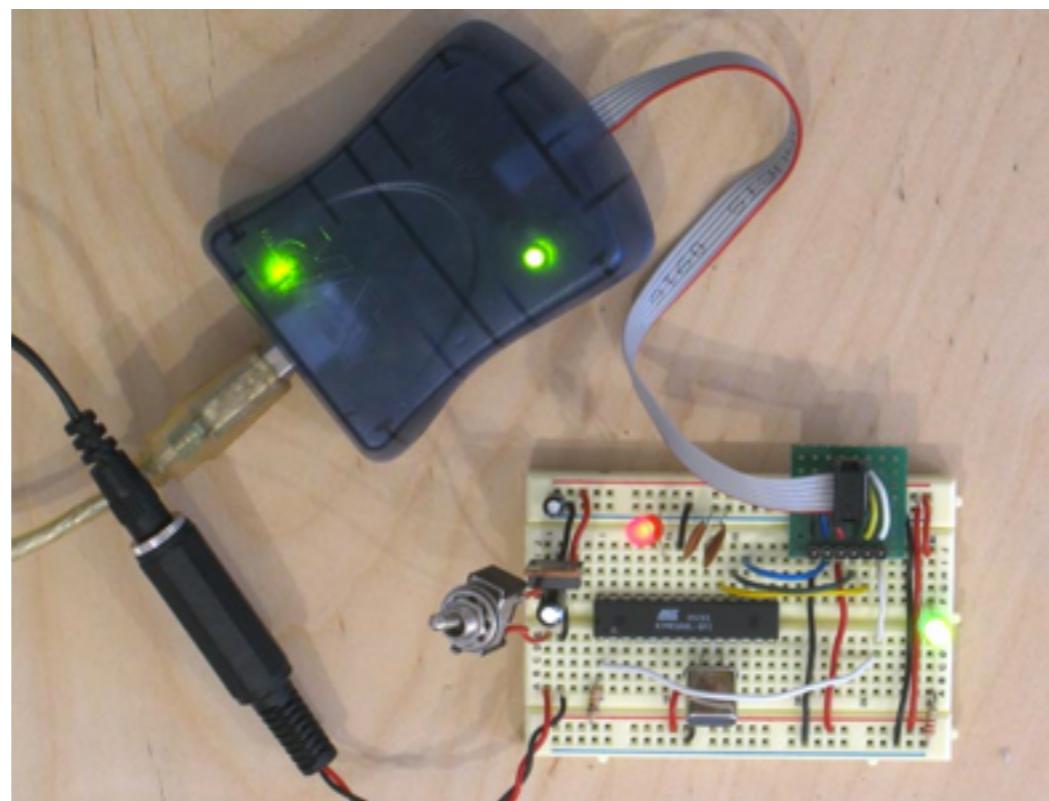
Then, in any view rendered by EventsController, the
format_time method can be called:

  <% events.each do |event| %>
    <p>
      <%= format_time(event.time, :short, '%A') %> | <%= event.name %>
    </p>
```

LAC

# Typical Development Environments

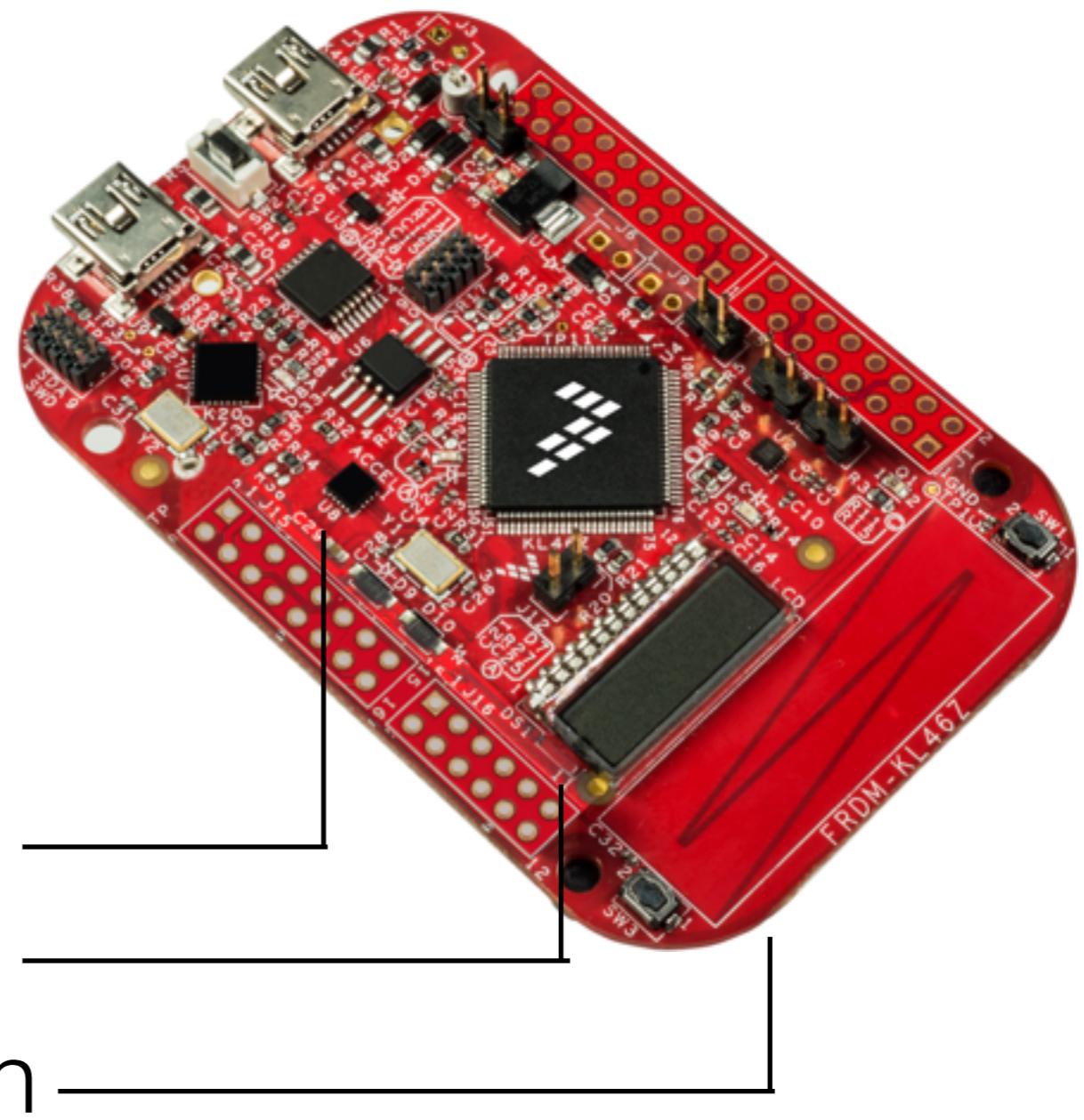
---



# Your Embedded Platform – “mbed”

---

- Online IDE
- Compiles Online
- C++
- Tons of libraries
- Fast prototyping platform
- Drag and drop programming



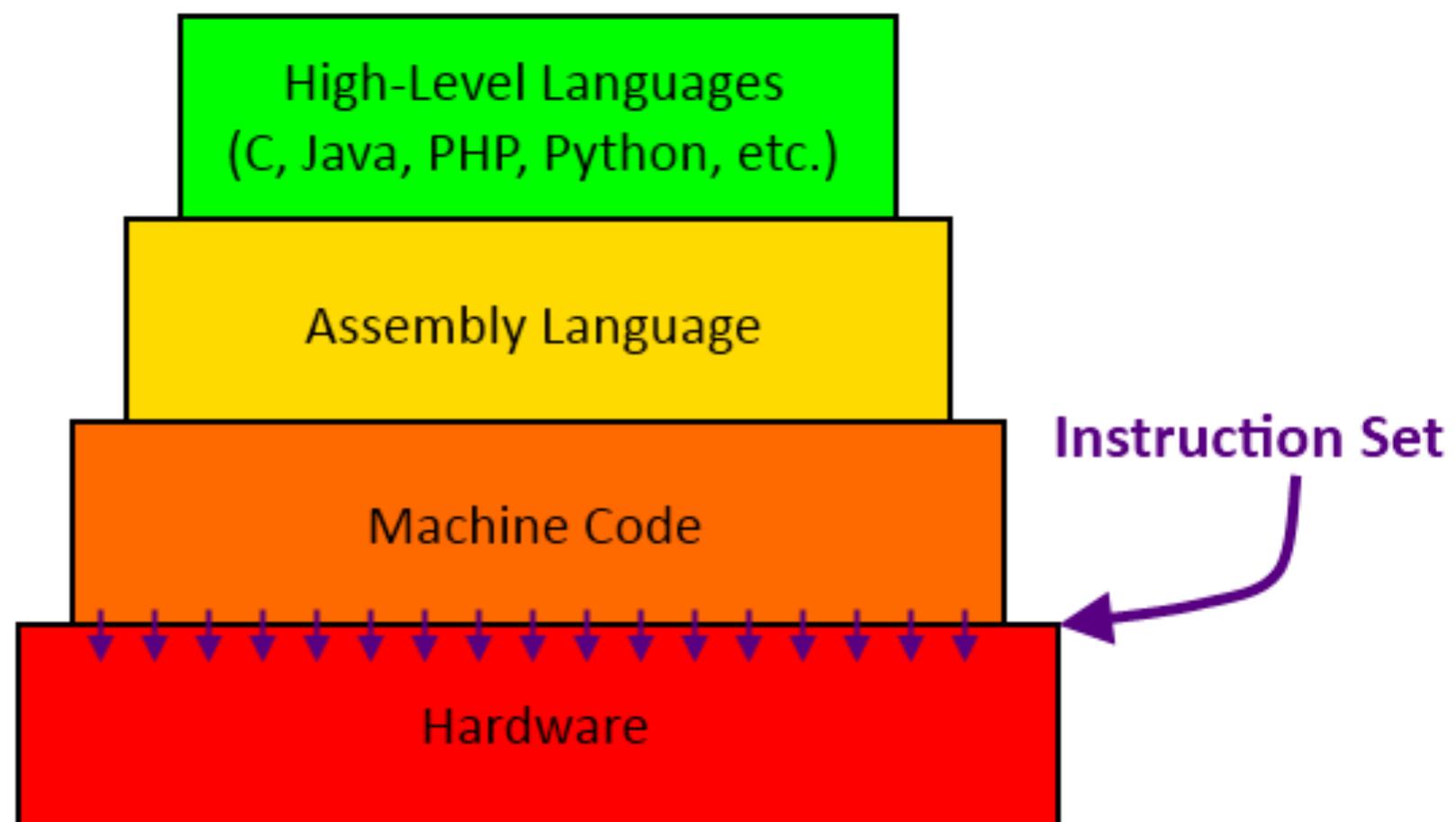
# Programming an Embedded Device

- Program code is closely tied to physical hardware, so we usually program in

C language

and occasionally

C++, Java, Lua



# Introduction to C and C++



# Primitives and Operators

---

## Data types

- integers - int
  - `int x = 1;`
- floating point & double
  - `float x = 0.23;`
  - `double x = 1.55;`
- characters - char
  - `char x = 'c';`
- multiple characters - string
  - `string x = "hello world";`

## Operators

- assignment =
- arithmetic +,-,\*,/,%
- relational <, >, <=, >=, !=, ==
- logic !, &&, ||

## Comments

- `//, /* ... */`

## Arrays

- `int myArray[3] = { 5, 5, 5 };`
- `int a[2][2] = { {0, 1} ,{2,3} };`
- Accessed like python



# C: Control Structure

---

```
if ( a > b ) {  
    ...  
} else {  
    ...  
}
```



# C: Control Structure

---

```
while ( a != b ) {  
    ...  
}
```



# C: Control Structure

---

```
for (int i = 0; i < 100; i++) {
```

...

```
}
```

```
for (initialize; end condition; update) {
```

...

```
}
```



# C: Functions

---

```
int addition (int a, int b) {  
    int results;  
    results = a + b;  
    return results;  
}
```

## Return Types

- int, double, char, void, ...



# C: Array Operations

---

```
float sensorvals[10];
for(int i=0; i<10; i++){
    sensorvals[i] = i*i;
}

float mean = 0.0;
for(int i=0; i<10; i++){
    mean += sensorvals[i]/10.0;
}
```



# C++: Objects (Classes)

```
// classes example
```

Object name

```
class Rectangle {
```

Object variables

```
    int width, height;
```

```
public:
```

```
    void set_values (int,int);
```

Object functions

```
    int area() {return width*height;}
```

```
};
```

```
void Rectangle::set_values (int x, int y) {
```

Defining object functions  
outside of a class

```
    width = x;
```

```
    height = y;
```

```
}
```

```
int main () {
```

Initializing an object

```
    Rectangle rect;
```

Calling an object function

```
    rect.set_values (3,4);
```

```
    printf("area: %d", rect.area() );
```

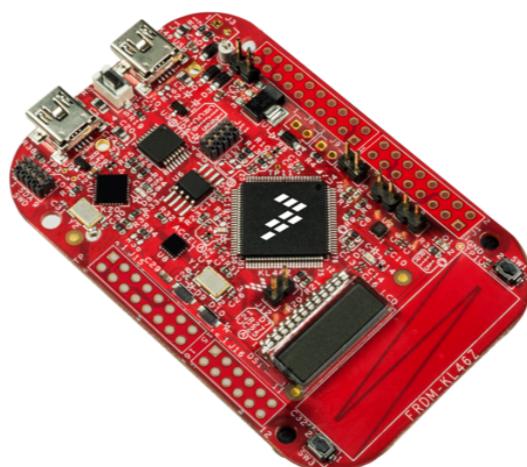
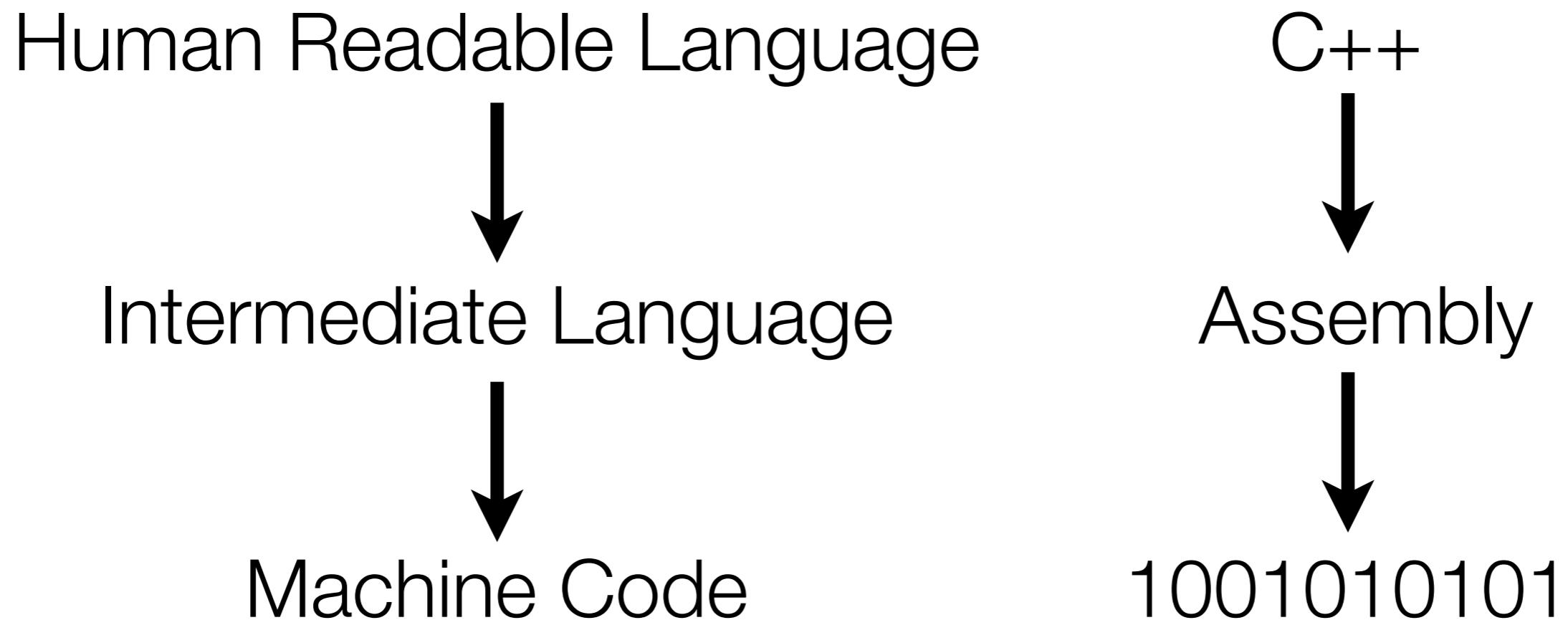
```
    return 0;
```

```
}
```



# C++ Compilation

---



# C++ vs Python

---

- C++ programs run faster
- Compiler optimizes code for processor
- C++ requires about 4 times more lines of code
- Python works better as a “glue” language
- Python is a good rapid prototyping language



# Welcome to Embedded Programming!



# Your Embedded Programming Task

---

- Use an mbed to design a [remote controller](#) for a [2 player video game!](#) The better your controller, the better you'll be able to play the game. The [winning team will be rewarded!](#) (*mbed?*)
- **Steps:**
  1. Become familiar with mbed programming, C/C++, and reading the various sensors on the mbed
  2. Be able to perform gesture recognition using the on-board 3D accel.
  3. Be able to communicate with your wireless radio
  4. Create your wireless controller combining steps 2 and 3
  5. Compete!



# mBed Account

---

- Sign up for mbed account @ [www.mbed.org](http://www.mbed.org)
- Your device information @ <http://mbed.org/platforms/FRDM-KL46Z/>



# C++ Program

mbed Compiler - /HelloWorld3/main.cpp

New Import Workspace Save Save All Compile Commit Revisions

**Program Workspace**

- My Programs
  - + blinker
  - + ee202a\_hw1
  - + ee202a\_hw3\_Part\_1
  - + ee202a\_hw3\_Part\_2\_1
  - + ee202a\_hw3\_Part\_2\_2
  - + ee202a\_hw3\_part\_2\_3
  - + ee202a\_hw4\_jason
  - + ee202a\_hw4\_newton\_v1
  - + ee202a\_hw4\_newton\_v2
  - + ee202a\_hw6\_masterCOM
  - + ee202a\_hw6\_slaveCOM
  - + FountainTest
  - + HelloCosm
  - HelloWorld

**main.cpp x**

```
1 #include "mbed.h" ← library
2
3 DigitalOut myled(LED1);
4
5 int main() {
6
7
8
9
10 }
11
```

< | Compiler Output for Program: HelloWorld3 | Description | ErrNo | Resource



# C++ Program

mbed Compiler - /HelloWorld3/main.cpp

New Import Workspace Save Save All Compile Commit Revisions

**Program Workspace**

- My Programs
  - + blinker
  - + ee202a\_hw1
  - + ee202a\_hw3\_Part\_1
  - + ee202a\_hw3\_Part\_2\_1
  - + ee202a\_hw3\_Part\_2\_2
  - + ee202a\_hw3\_part\_2\_3
  - + ee202a\_hw4\_jason
  - + ee202a\_hw4\_newton\_v1
  - + ee202a\_hw4\_newton\_v2
  - + ee202a\_hw6\_masterCOM
  - + ee202a\_hw6\_slaveCOM
  - + FountainTest
  - + HelloCosm
  - HelloWorld

**main.cpp x**

```
1 #include "mbed.h" ← name of "library"
2 ← directive to include said "library"
3 DigitalOut myled(LED1);
4
5 int main() {
6
7
8
9
10 }
11
```

< | Compiler Output for Program: HelloWorld3 | Description | ErrNo | Resource



# C++ Program

mbed Compiler - /HelloWorld3/main.cpp

New Import Workspace Save Save All Compile Commit Revisions

**Program Workspace**

- My Programs
  - + blinker
  - + ee202a\_hw1
  - + ee202a\_hw3\_Part\_1
  - + ee202a\_hw3\_Part\_2\_1
  - + ee202a\_hw3\_Part\_2\_2
  - + ee202a\_hw3\_part\_2\_3
  - + ee202a\_hw4\_jason
  - + ee202a\_hw4\_newton\_v1
  - + ee202a\_hw4\_newton\_v2
  - + ee202a\_hw6\_masterCOM
  - + ee202a\_hw6\_slaveCOM
  - + FountainTest
  - + HelloCosm
  - HelloWorld

**main.cpp x**

```
1 #include "mbed.h"
2
3 DigitalOut myled(LED1); ← Create a digital pin object
4
5 int main() {
6
7
8
9
10 }
11
```

< | Compiler Output for Program: HelloWorld3 | Description | ErrNo | Resource



# C++ Program

mbed Compiler - /HelloWorld3/main.cpp

New Import Workspace Save Save All Compile Commit Revisions

**Program Workspace**

- My Programs
  - + blinker
  - + ee202a\_hw1
  - + ee202a\_hw3\_Part\_1
  - + ee202a\_hw3\_Part\_2\_1
  - + ee202a\_hw3\_Part\_2\_2
  - + ee202a\_hw3\_part\_2\_3
  - + ee202a\_hw4\_jason
  - + ee202a\_hw4\_newton\_v1
  - + ee202a\_hw4\_newton\_v2
  - + ee202a\_hw6\_masterCOM
  - + ee202a\_hw6\_slaveCOM
  - + FountainTest
  - + HelloCosm
  - HelloWorld

**main.cpp x**

```
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() {
6
7
8
9
10 }
11
```

set specific pin

variable name

Object Name

**Compiler Output for Program: HelloWorld3**

Description	FrrNo	Resource
-------------	-------	----------



# C++ Program

mbed Compiler - /HelloWorld3/main.cpp

New Import Workspace Save Save All Compile Commit Revisions

**Program Workspace**

- My Programs
  - + blinker
  - + ee202a\_hw1
  - + ee202a\_hw3\_Part\_1
  - + ee202a\_hw3\_Part\_2\_1
  - + ee202a\_hw3\_Part\_2\_2
  - + ee202a\_hw3\_part\_2\_3
  - + ee202a\_hw4\_jason
  - + ee202a\_hw4\_newton\_v1
  - + ee202a\_hw4\_newton\_v2
  - + ee202a\_hw6\_masterCOM
  - + ee202a\_hw6\_slaveCOM
  - + FountainTest
  - + HelloCosm
  - HelloWorld

**main.cpp x**

```
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() { ← Main function
6
7
8
9
10 }
11
```

< | Compiler Output for Program: HelloWorld3 | Description | ErrNo | Resource



# C++ Program

mbed Compiler - /HelloWorld3/main.cpp

New Import Workspace Save Save All Compile Commit Revisions

**Program Workspace**

- My Programs
  - + blinker
  - + ee202a\_hw1
  - + ee202a\_hw3\_Part\_1
  - + ee202a\_hw3\_Part\_2\_1
  - + ee202a\_hw3\_Part\_2\_2
  - + ee202a\_hw3\_part\_2\_3
  - + ee202a\_hw4\_jason
  - + ee202a\_hw4\_newton\_v1
  - + ee202a\_hw4\_newton\_v2
  - + ee202a\_hw6\_masterCOM
  - + ee202a\_hw6\_slaveCOM
  - + FountainTest
  - + HelloCosm
  - HelloWorld

**main.cpp x**

```
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() { ← parameters (none)
6   ← function name
7
8
9
10 }
11
```

**Compiler Output for Program: HelloWorld3**

Description	FrrNo	Resource
-------------	-------	----------

parameters (none)  
function name  
return type (why is it an int?)



# C++ Program

mbed Compiler - /HelloWorld3/main.cpp

New Import Workspace Save Save All Compile Commit Revisions

**Program Workspace**

- My Programs
  - + blinker
  - + ee202a\_hw1
  - + ee202a\_hw3\_Part\_1
  - + ee202a\_hw3\_Part\_2\_1
  - + ee202a\_hw3\_Part\_2\_2
  - + ee202a\_hw3\_part\_2\_3
  - + ee202a\_hw4\_jason
  - + ee202a\_hw4\_newton\_v1
  - + ee202a\_hw4\_newton\_v2
  - + ee202a\_hw6\_masterCOM
  - + ee202a\_hw6\_slaveCOM
  - + FountainTest
  - + HelloCosm
  - HelloWorld

**main.cpp x**

```
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() {
6
7
8
9
10 }
11
```

curly braces

**Compiler Output for Program: HelloWorld3**

Description	FrrNo	Resource
-------------	-------	----------



# C++ Program

mbed Compiler - /HelloWorld3/main.cpp

New Import Workspace Save Save All Compile Commit Revisions

Program Workspace

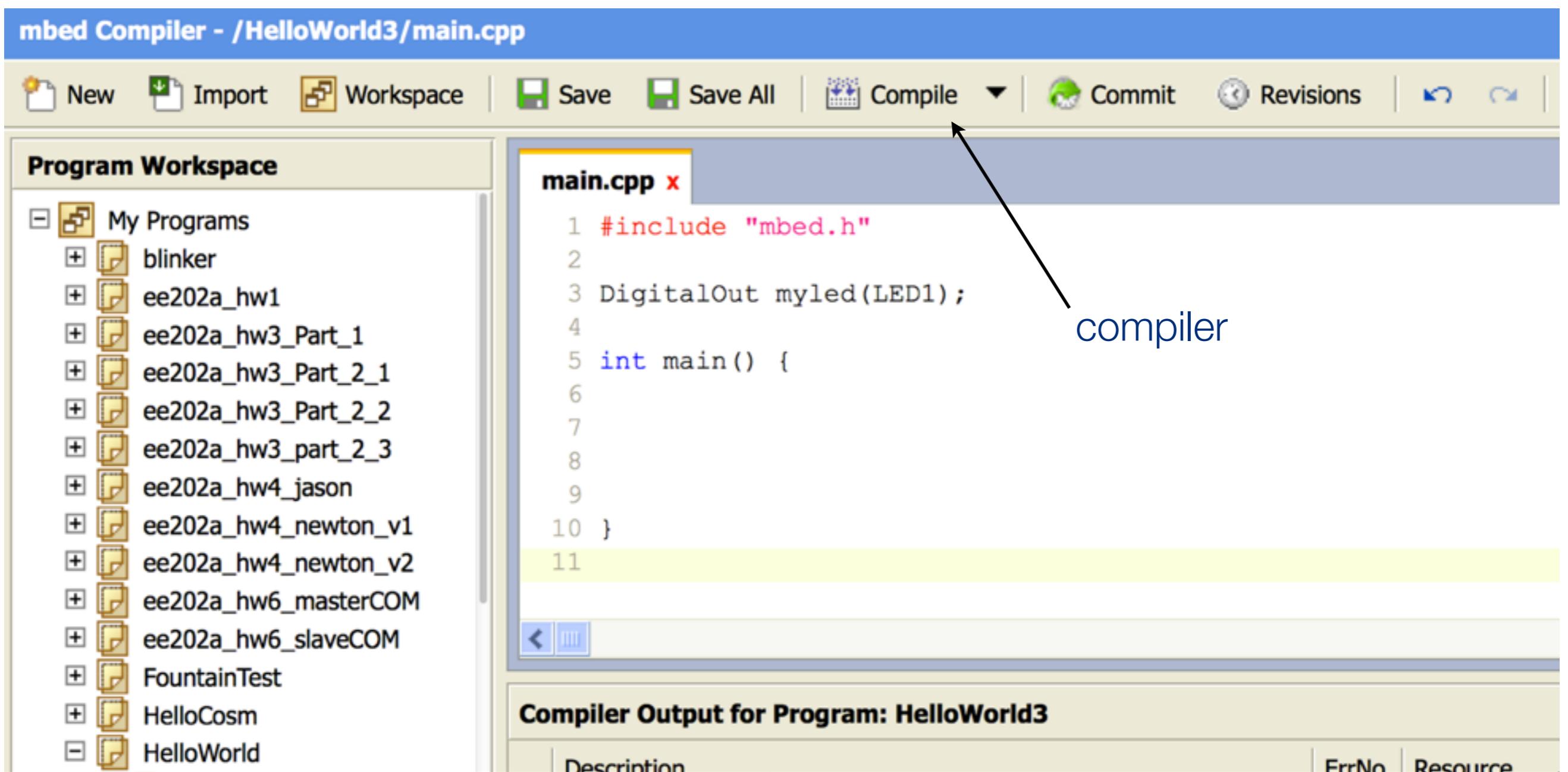
- My Programs
  - + blinker
  - + ee202a\_hw1
  - + ee202a\_hw3\_Part\_1
  - + ee202a\_hw3\_Part\_2\_1
  - + ee202a\_hw3\_Part\_2\_2
  - + ee202a\_hw3\_part\_2\_3
  - + ee202a\_hw4\_jason
  - + ee202a\_hw4\_newton\_v1
  - + ee202a\_hw4\_newton\_v2
  - + ee202a\_hw6\_masterCOM
  - + ee202a\_hw6\_slaveCOM
  - + FountainTest
  - + HelloCosm
  - HelloWorld

main.cpp x

```
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() {
6
7
8
9
10 }
11
```

Compiler Output for Program: HelloWorld3

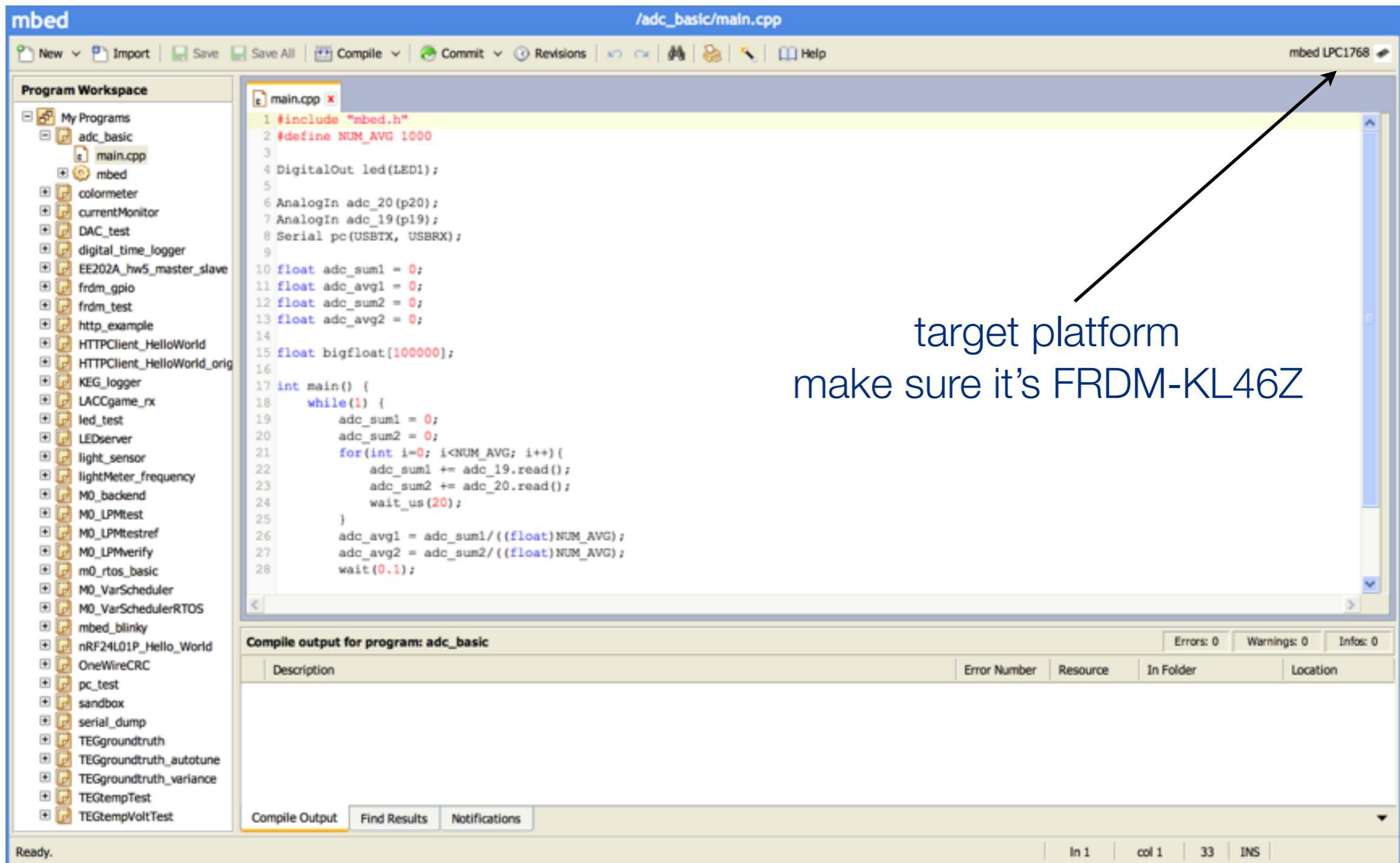
Description ErrNo Resource



compiler



# C++ Program



mbed /adc\_basic/main.cpp

Program Workspace

- My Programs
  - adc\_basic
    - main.cpp
  - mbed
  - colorimeter
  - currentMonitor
  - DAC\_test
  - digital\_time\_logger
  - EE202A\_hw5\_master\_slave
  - frdm\_gpio
  - frdm\_test
  - http\_example
  - HTTPClient\_HelloWorld
  - HTTPClient\_HelloWorld\_orig
  - KEG\_logger
  - LACCgame\_rx
  - led\_test
  - LEDserver
  - light\_sensor
  - lightMeter\_frequency
  - M0\_backend
  - M0\_LPMtest
  - M0\_LPMtestref
  - M0\_LPMverify
  - m0\_rtos\_basic
  - M0\_Varscheduler
  - M0\_VarschedulerRTOS
  - mbed\_blinky
  - nRF24L01P\_Hello\_World
  - OneWireCRC
  - pc\_test
  - sandbox
  - serial\_dump
  - TEGgroundtruth
  - TEGgroundtruth\_autotune
  - TEGgroundtruth\_variance
  - TEGtempTest
  - TEGtempVoltTest

main.cpp

```
1 #include "mbed.h"
2 #define NUM_AVG 1000
3
4 DigitalOut led(LED1);
5
6 AnalogIn adc_20(p20);
7 AnalogIn adc_19(p19);
8 Serial pc(USBTX, USBRX);
9
10 float adc_sum1 = 0;
11 float adc_avg1 = 0;
12 float adc_sum2 = 0;
13 float adc_avg2 = 0;
14
15 float bigfloat[100000];
16
17 int main() {
18     while(1) {
19         adc_sum1 = 0;
20         adc_sum2 = 0;
21         for(int i=0; i<NUM_AVG; i++) {
22             adc_sum1 += adc_19.read();
23             adc_sum2 += adc_20.read();
24             wait_us(20);
25         }
26         adc_avg1 = adc_sum1/((float)NUM_AVG);
27         adc_avg2 = adc_sum2/((float)NUM_AVG);
28         wait(0.1);
29     }
30 }
```

Compile output for program: adc\_basic

Description	Error Number	Resource	In Folder	Location

Compile Output Find Results Notifications

Ready. In 1 col 1 33 INS

target platform  
make sure it's FRDM-KL46Z



# C++ Program

mbed Compiler - /HelloWorld3/main.cpp

The screenshot shows the mbed Compiler interface with the file `main.cpp` open. The code defines a `DigitalOut` object `myled` and a `main()` function that toggles the pin every 0.2 seconds. Handwritten annotations explain the logic:

- `while(1)` → while loop
- `myled = 1;` → set pin to output 1 or 3.3 V
- `wait(0.2);` → pause 0.2 seconds
- `myled = 0;` → set pin to output 0 or 0 V

```
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() {
6     while(1) {
7         myled = 1;
8         wait(0.2);
9         myled = 0;
10        wait(0.2);
11    }
12 }
```

Compiler Output for Program: HelloWorld3



# Blinking LEDs

---

1. Sign up for mbed account
2. Modify the blinking LED program
3. Write a program to blink LEDs in series

Create new pin operations for:  
LED1, LED2, LED3, LED4



# “Hello World” Terminal

---

- Windows
  - 1. Download driver from [www.mbed.org](http://www.mbed.org)
  - 2. Download TeraTerm
  - 3. Write “Hello World” program
  - 4. Compile and program mbed
- MAC & Linux
  - 6. Write “Hello World” Program
  - 7. Open terminal -> screen /dev/tty.usbmodem (tab complete)
  - 8. Compile and program mbed



# “Hello World” Terminal (Windows)

The screenshot shows the mbed website homepage. At the top, there is a navigation bar with links for File, Edit, View, History, Bookmarks, Tools, Window, Help, and a user account section. Below the navigation bar, there are several tabs: "Rapid Prototyping for Microcontrollers | mbed", "mbed Compiler", and "Inbox (1) - newton.truong@gmail.com". The main content area features the mbed logo and the tagline "Rapid Prototyping for Microcontrollers". A large heading says "mbed is a tool for Rapid Prototyping with Microcontrollers". Below this, there are four categories: "Prototyping hardware", "C++ SDK", "Powerful online tools", and "Active community". To the right, there is a photograph of a yellow mbed NXP LPC11U24 development board. Further down, there are sections for "Hardware" (with images of two boards) and "Online IDE" (with a screenshot of the mbed Compiler interface). On the right side, there is a search bar labeled "Search mbed" and a "Recent Activity" feed. At the bottom, there are links for "Standard Libraries" and "Community", along with icons for various software applications.



# “Hello World” Terminal (Windows)

The screenshot shows a Firefox browser window with the following details:

- Address Bar:** mbed.org/handbook/Homepage
- Toolbar:** Firefox, File, Edit, View, History, Bookmarks, Tools, Window, Help
- Right Sidebar:** A sidebar titled "mbed Handbook" with several links:
  - Order
  - mbed Developer Website
  - CMSIS RTOS
  - RTOS
  - Exporting to offline toolchains
  - Collaboration/Getting started
  - Collaboration/Forking
  - Collaboration/Publishing
- Content Area:**
  - CorTEX-M3 designs:**
    - mbed Compiler - All about the mbed Compiler
    - mbed SDK - All about the mbed C/C++ SDK and peripheral libraries
    - mbed Developer Website - The tools and collaborative features of the mbed website
    - Sponsorship - The mbed Sponsorship Program
    - Education - The mbed Educational Program
    - About - The background of mbed, press resources and [press articles](#) about mbed
    - mbed official - All the code officially supported by the mbed team
  - Getting started with mbed:**
    - Setup guide - Getting signed up with an mbed account
    - Downloading a program - Running a program binary on your mbed microcontroller for the first time
    - Creating a program - Creating your own program with the mbed compiler
    - Communicating over USB Serial - Communicate between an mbed Microcontroller and a PC (This link is highlighted with a red box and has a black arrow pointing to it.)
    - Debugging - A guide to help find and solve errors and bugs in your programs
    - FAQs - Frequently asked technical questions about working with mbed
    - Nontechnical FAQs - Frequently asked questions about accounts, university/college use, etc. See also [Technical FAQ](#).
    - Help - How to ask for help
  - More advanced topics:**
    - Code sharing
      - The Cookbook - Personal projects
      - List of all published programs and libraries
      - Writing a library
      - Auto generated documentation
    - Technical details



# “Hello World” Terminal (Windows)

Host interface and terminal applications

Your mbed Microcontroller can appear on your computer as a serial port. On Mac and Linux, this will happen by default. For Windows, you need to install a driver:

**Windows**

See [Windows-serial-configuration](#) for full details about setting up Windows for **serial communication** with your mbed Microcontroller

It is common to use a **terminal application** on the host PC to communicate with the mbed Microcontroller. This allows the mbed Microcontroller to print to your PC screen, and for you to send characters back.

- **Terminals** - Using Terminal applications to communicate between the Host PC and the mbed Microcontroller

Some terminal programs (e.g. TeraTerm) list the available serial ports by name. However, if you do need to know the identity of the serial port so that you can attach a terminal or an application to it:

- **Windows** - Look under the "Ports" section in "Device Manager" ("Start -> Control Panel -> System -> Hardware -> Device Manager"). The name will be "mbed Serial Port (COMx)", where "x" is the number of the COM port allocated.
- **Mac OS X** - Use the command `ls /dev/tty.usbmodem*`
- **Linux** - Use the command `ls /dev/ttyACM*`



# “Hello World” Terminal (Windows)

The mbed serial port works by default on Mac and Linux, but Windows needs a driver. These instructions explain how to setup the mbed Microcontroller to use the USB serial port on Windows.

## 1. Download the mbed Windows serial port driver

Download the installer to your PC, e.g. your desktop.

[Download latest driver](#)

## 2. Run the installer

With your *mbed plugged in*, and *no explorer drive windows open*, run the installer:

It will take some time (especially on Vista), and pop up a few 'unsigned driver' warnings, but after a while you should have a Serial port.

### Where Next

- SerialPC - Communication with a PC
- [Terminals](#) - Guide to using terminal applications

#### Troubleshooting

If you have multiple mbed microcontrollers, but the serial port only appears for one of them:

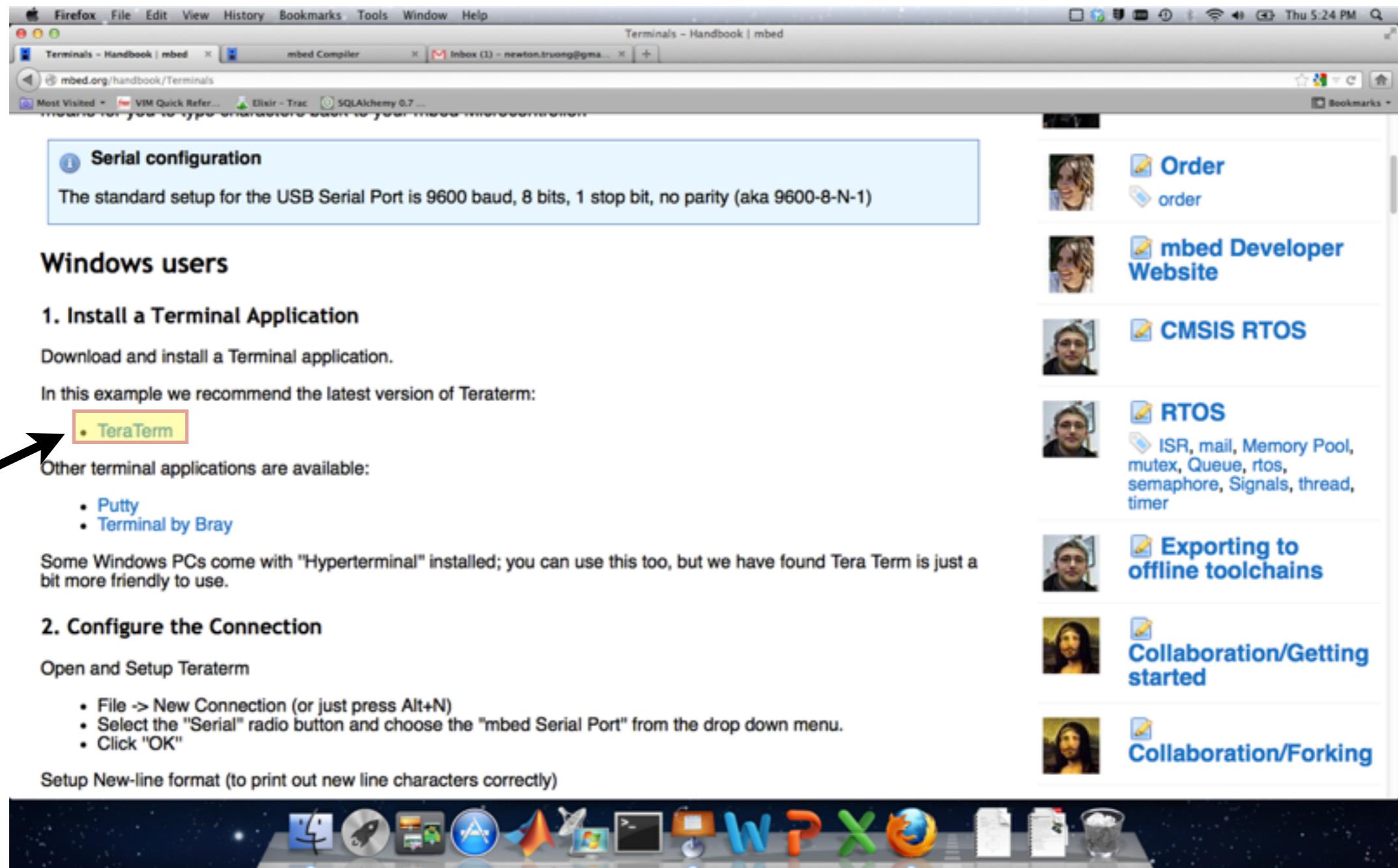
- Make sure you run the installer for every mbed; windows loads the driver based on the serial number, so it

Recent changes

- VodafoneK3770 Interface
- Ethernet Interface
- Order
- mbed Developer Website
- CMSIS RTOS
- RTOS
- Exporting to offline toolchains



# “Hello World” Terminal (Windows)



The standard setup for the USB Serial Port is 9600 baud, 8 bits, 1 stop bit, no parity (aka 9600-8-N-1)

## Windows users

### 1. Install a Terminal Application

Download and install a Terminal application.

In this example we recommend the latest version of Teraterm:

- [TeraTerm](#)

Other terminal applications are available:

- Putty
- Terminal by Bray

Some Windows PCs come with "Hyperterminal" installed; you can use this too, but we have found Tera Term is just a bit more friendly to use.

### 2. Configure the Connection

Open and Setup Teraterm

- File -> New Connection (or just press Alt+N)
- Select the "Serial" radio button and choose the "mbed Serial Port" from the drop down menu.
- Click "OK"

Setup New-line format (to print out new line characters correctly)

Firefox File Edit View History Bookmarks Tools Window Help  
Terminals - Handbook | mbed mbed Compiler Inbox (1) - newton.truong@gmail.com  
mbed.org/handbook/Terminals  
Most Visited VIM Quick Refer... Elixir - Trac SQLAlchemy 0.7 ...  
Bookmarks

Order  
order

mbed Developer Website

CMSIS RTOS

RTOS  
ISR, mail, Memory Pool, mutex, Queue, rtos, semaphore, Signals, thread, timer

Exporting to offline toolchains

Collaboration/Getting started

Collaboration/Forking



# “Hello World” Program

mbed Compiler - /HelloWorld3/main.cpp

The screenshot shows the mbed Compiler IDE interface. The title bar reads "mbed Compiler - /HelloWorld3/main.cpp". The menu bar includes "File", "Import", "Workspace", "Save", "Save All", "Compile", "Commit", "Revisions", "Format", and "mbed NXP LPC1768". The left sidebar, titled "Program Workspace", lists several projects: FountainTest, HelloCosm, HelloWorld, HelloWorld2, and HelloWorld3. The "HelloWorld3" project is selected, and its "main.cpp" file is open in the main editor window. The code in "main.cpp" is:

```
1 #include "mbed.h"
2
3 Serial pc(USBTX,USBRX); // my comments
4
5 int main() {
6     while(1) {
7         pc.printf("Hello World\r\n");
8         wait(2);
9         pc.printf("Goodbye World\r\n");
10        wait(2);
11    }
12 }
```

Below the editor is a "Compiler Output for Program: HelloWorld3" panel with columns for "Description", "ErrNo", "Resource", "In Folder", and "Location". The status bar at the bottom shows "Ready." and "INS".



# “Hello World” Terminal (Mac)

---

- Windows
  - 1. Open TeraTerm
  - 2. Connect to mbed
- MAC
  - 3. Open terminal -> screen /dev/tty.usbmodem (tab complete)



# Reading Sensors

---

- The FRDM-KL46Z (mbed) has several built-in sensors:
  - 3D accelerometer (measures accelerations)
  - Capacitive touch sensor
- The mbed API very nicely wraps the reading of these sensors into easy-to-use functions:

```
TSIAalogSlider touchsensor(ELEC0, ELEC1, 40);
```

```
float touchval = touchsensor.readPercentage();
```



# About printf – formatted printing

---

```
int i = 3;  
pc.printf("The value of i is %d\n", i);
```

console >> The value of i is 3

- %d - integers
- %f - floats
- %c - character
- %s - string of characters

```
int num_dogs = 3;  
float num_cats = 2.5;
```

```
pc.printf("There are %d dogs and %f cats\n", num_dogs, num_cats);
```



# Putting it All Together

---

- After looking through mbed docs, you should be able to:
  - Read the touch sensor
  - Read all 3 axes of the accelerometer
  - Print sensor readings over USB and display on a computer
- Now we need to:
  - Detect “gesture” using your accelerometer
  - Communicate commands wirelessly to play a game



# Wireless Communication

---

- You each will be given an [nRF24L01+](#) wireless module—a 2.4 ghz transceiver (both RX & TX) that communicates using wired serial communication (called [SPI](#))
- There is a preexisting mbed library for the nRF24L01+ which you should find and use in your library. This will include functions for initializing your wireless module, such as this:

```
// initialize wireless comm.  
my_nrf24l01p.powerUp();  
my_nrf24l01p.setTransferSize( TRANSFER_SIZE );  
my_nrf24l01p.setReceiveMode();  
my_nrf24l01p.enable();  
my_nrf24l01p.setAirDataRate(2000);  
my_nrf24l01p.setRxAddress(0xE7E7E7E7E7);  
my_nrf24l01p.setTxAddress(0xCDABCDABCD);
```



# Wireless Communication

---

- Follow instructions online for connecting your wireless module to your mbed using the provided wires, and ask for assistance if necessary.
- Follow example code to ensure that you can properly read from your radio
- Use example code to test both RX and TX with another team's radio!



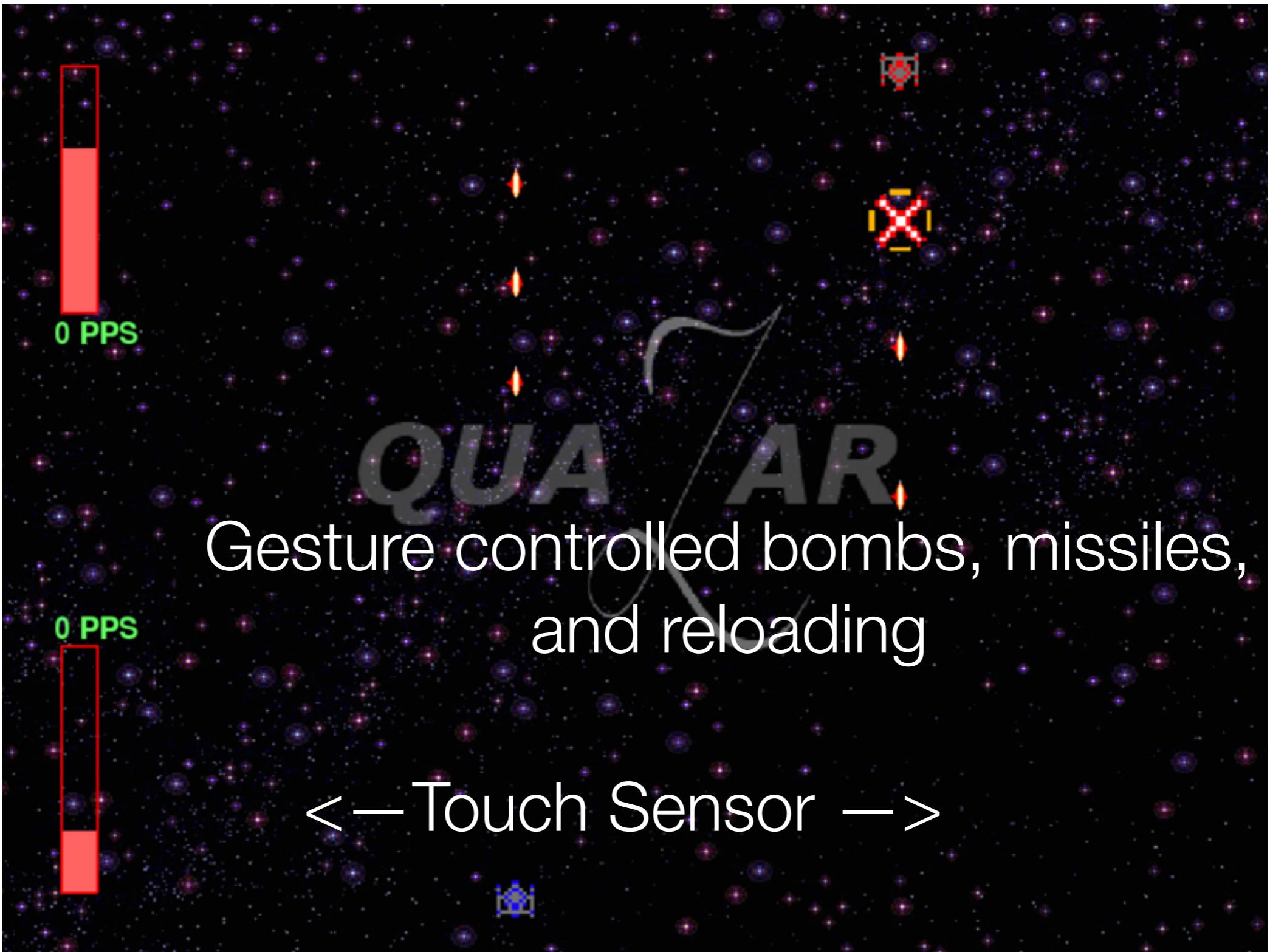
# Quazar!



# Quazar, the LACC-original Python Game



# Quazar, the LACC-original Python Game



# Quazar, the LACC-original Python Game

---

DEMO!



# The Rules of Engagement

---

- Each team is given a unique 1-byte (0-255) identifier
- To control your player, you must wirelessly transmit a packet to the address “CDABCDABCD” at 2.402 ghz and 2000 kbps formatted like so:

Team ID	Command Type	Command Value

0-255: provided

0: move  
1: fire missile  
2: fire bomb  
3: reload missiles  
4: reload bombs

Only used for moving,  
0 is far left, 255 is far right



# The Rules of Engagement

---

- You start with 15 **missiles**, 3 **bombs**, and 12 **HP**
  - **missile**: 1 damage
  - **bomb**: 3 damage
- You must reload for more ammo (roughly 1-2 seconds)
- Your ship is out in the far reaches of space, and you can only talk to it infrequently! If you transmit more than **100 packets per second**, your ship will freeze for an entire second as a penalty!



# Final Words of Wisdom

---

- Your gestures should be recognized quickly by mbed, but with low false positives—look into **filtering** your data
- You'll need to limit your wireless data rate—the easiest way to do this is by **adding delays** in your infinite loops
- You can **add smarts!** Instead of one gesture shooting one missile, why not make one gesture shoot a barrage of 5 missiles?
- **Don't script** your entire controller—if you can't demonstrate full control over your vehicle (i.e. your controller is automated), you'll be disqualified!



# Further Research Topics

---

- If you're curious and you'd like to learn more about embedded computing, check out these cool topics:
  - Real Time Operating Systems (RTOS) and real time scheduling
  - Time Synchronization
  - Wired communication protocols (I2C, SPI, CAN, UART/USART)
  - Low power embedded computing practices (duty cycling, interrupts, low power modes, processor frequencies). TI MSP430 Processors
  - RISC architectures and ARM processors — Cortex M0-M4
  - Specialized hardware accelerators—DSP and FPGA processors.



# See You on the Battlefield

---

- Program and test today
- You'll do battle tomorrow, winners will walk away with something special!
- Happy coding :)

