**University of California, Santa Cruz**

**COSMOS Cluster 6:**

**Networking and Robotics**

**Lab 1: Hello World**

## Introduction

The first step in any programming project is writing and running a “Hello World” program. The “Hello World” program is the simplest program possible: It simply prints something to the user, and does nothing else. This might seem simple, but it can be very hard! It requires at least three things:

1. Successfully installing and connecting the toolchain used to build and/or compile and/or load and/or run a program
2. Being familiar enough with the programming language to write a valid program
3. Running the program

And these are rarely simple! Of course, we’ll do a lot of the work for you.

Note that this lab is not about learning C. We’ll have plenty of time for that soon.

## Instructions

Follow the steps below. There are questions throughout the manual. Discuss them with your partner and write down the answers.

## Concepts

* The RoachBot Toolchain
  + The C compiler
  + The Pic32
  + The MPLAB X Integrated development environment
  + The DS30 Bootloader
  + Serial Terminals
* Main functions
* The RoachBot libraries
  + Motors
  + Light sensor
  + Bumpers

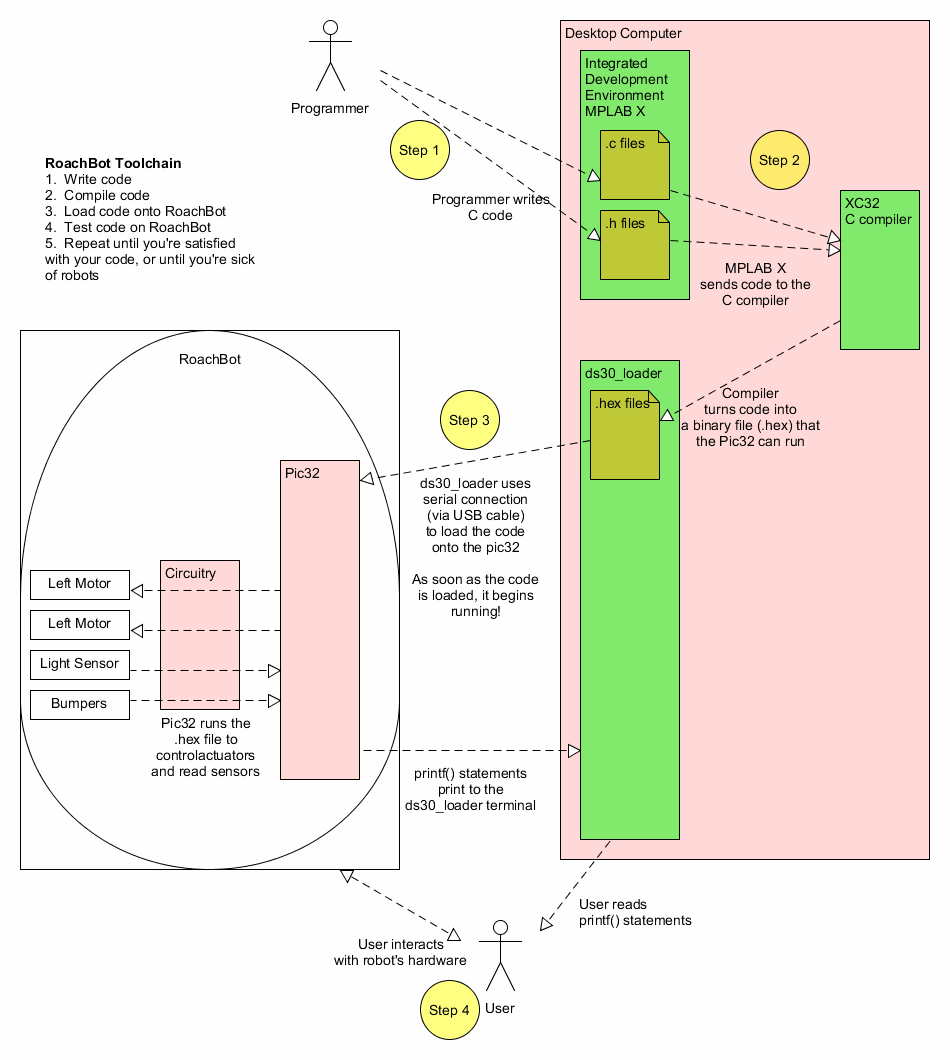
## Toolchain:

Technology is pretty complicated, and sometimes you need to use a lot tools to accomplish a particular goal. The sequence of tools you use is sometimes called a “toolchain.” Each tool in the process outputs something that the next tool uses. As a conceptual example, your toolchain for making an omlette might be something like:

Refrigerator > frying pan > stove > plate > table > eating

In this class, we’ll be using a similar toolchain to program a robot:

Code editor > Compiler > bootloader > interaction with robot

In this lab, you’ll use the whole toolchain, so it is familiar to you for the rest of the course. Here’s a diagram of the whole process:

Look over this diagram a bit!

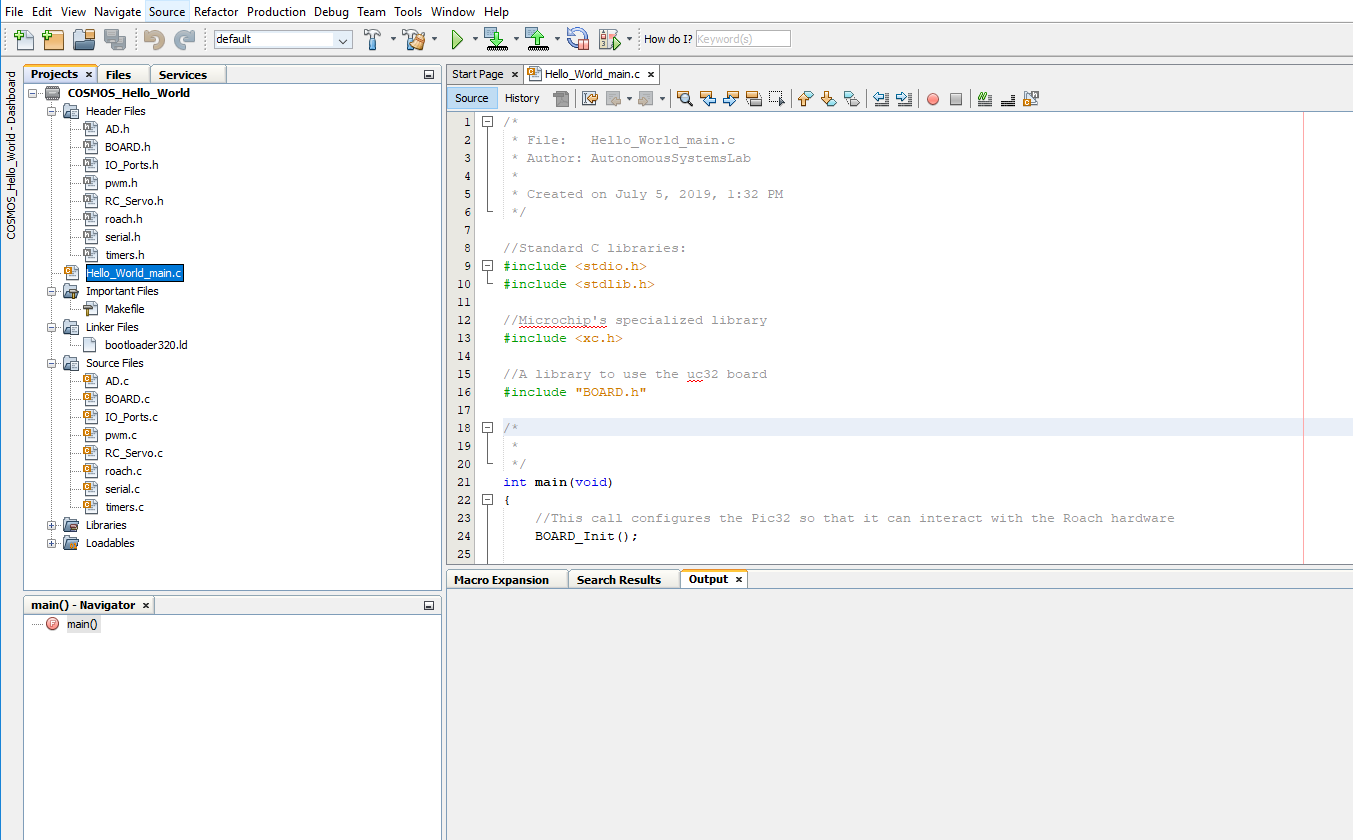
QUESTION: What kind of file needs to be loaded onto the roach?

Now, let’s go through each part of the toolchain

## Writing code in MPLAB X:

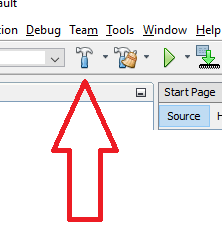
1. Download the Lab1 .zip file from the COSMOS Cluster 6 website (<https://cosmos-cluster-6.courses.soe.ucsc.edu/>) and unzip it by right-clicking it. Move the contents somewhere convenient, like the Desktop.
2. Open up MPLAB X (there’s a shortcut icon on the desktop)
3. Select “File -> Open Project.” Select the COSMOS\_Hello\_World.X folder (all MPLAB X projects are stored in folders that end in “.X”)

At this point, you should see something like this:



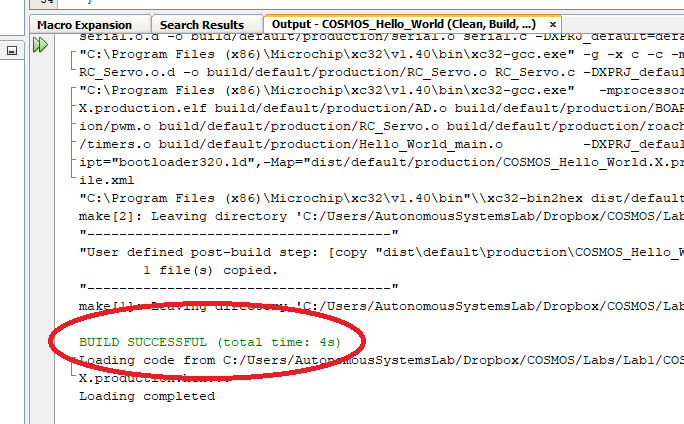
On the left, you’ll see the “Project” pane. It helps you navigate your files.

1. Every C program needs to have exactly one main() function. Usually, this is written in a file called “SomethingSomething\_main.c”. Double-click on the main file for this project.
2. Let’s try to compile the project. Press the “Build” button:



* 1. (The button to the right of that is “Clean Build”, and it’s pretty useful too because it often gives more detailed error messages)

1. When you press the button, you should see the “Output” pane show you a bunch of messages, followed by a success message:



* 1. You might instead see a red “Build Failed” message. If that’s the case, read the compiler errors (they’re written in red and blue, and if you can decipher them, they’ll tell you how to fix the problem)

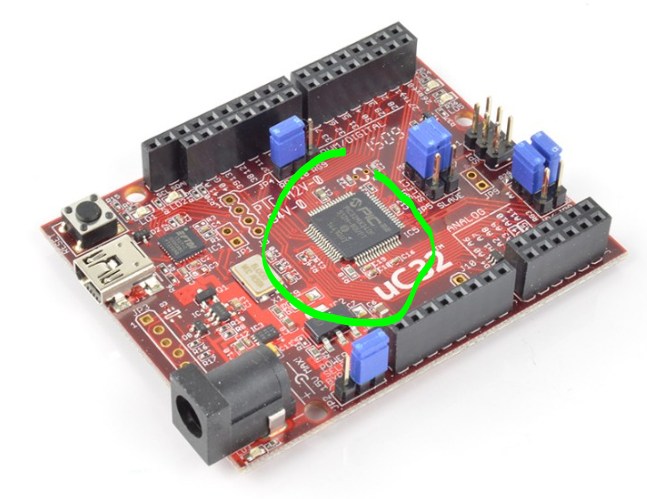
1. Great! But this code doesn’t do anything, so it’s not a good Hello World program yet. Let’s add an output. Write in the following line in the section for Initialization code:

printf("Hello World, I’m a RoachBot!\n")

1. Compile again. If successful[[1]](#footnote-1), you’re ready for the next step!

## Using the Bootloader to program the RoachBot and read output:

When you’re using the RoachBot, you’re really using two computers: There’s the desktop computer, which you’re using to type, and there’s a second computer on the roach itself. It’s a very tiny type of microcomputer called a “Microcontroller”. If you flip over the roach, you’ll see a red circuit board. That’s a uc32 board (pronounced “Micro32”), and it’s home to the Pic32 microcontroller:



Microcontrollers can only run a very specific type of code. If you look inside the .X folder, you’ll notice a file that wasn’t there before you compiled. It will be called something like COSMOS\_Hello\_World.production.hex. It’s a .hex file, which means it’s a “machine code” file, containing code in a format that the pic32 can run. It’s probably completely unreadable to you, though (try opening it with notepad++ or something to see!)

We need to load that .hex file off of the desktop computer, and onto the Pic32 on the roach. We’ll use a program called ds30Loader to do that.

1. Connect the USB cord on the roach to a USB port on your computer.
2. Open ds30LoaderGUI.exe.
   1. It should be on the desktop, but if not there’s a copy on the COSMOS website.
3. There are a lot of settings that we’ll need to set up:
   1. First, make sure that you’ve selected “View -> Advanced Mode”
   2. Under “basic” tab:
      1. Use the “…” next to the hexfile field to select your .hex file.
      2. “Device” should be set to PIC32MX / 320F128H
      3. The Baud Rate should be set to 115200
      4. The Port involves a bit of guesswork. It’s usually COM3 or COM4, but Windows uses a kind of mysterious system for numbering ports, so you might need to do some experimenting.
      5. Check “write flash”
   3. Under the “reset” tab, select DTR with a Reset time of 10 ms.
   4. At this point, click the “Check for bl” button (“bl” stands for “**B**oot**L**oader”). You should get a success message, something like “Found PIC32MX320F128H fw ver. 5.0.2.”
   5. Under the “Terminal” tab:
      1. Set the Baud Rate to 115200
      2. Check “Switch to after write”
4. At this point, you should be ready to load your code onto the roach! Click the “Write” button up top.
   1. The view should automatically switch to the terminal, and you should see “the phrase “Hello World, I’m a RoachBot” appear.
5. The bootloader uses the serial connection to do its loading magic, so it can’t run at the same time as the terminal. Notice that the “write” and “check for bl” buttons are greyed out now! You’ll also notice a little star by the “Terminal” tab, to show that the serial connection is open. You’ll need to close the Terminal connection, so click the “close” button to close the terminal before you can load new code.

At this point, your code is loaded, and you’ve successfully used the RoachBot toolchain!

QUESTION: What happens if you try to print inside of the while(1){ } loop? What do you think while(1) means?

## Extra challenges: Working with the Roach

Since you’ve finished using the toolchain, let’s get acquainted with some of the key Roach functions. You probably don’t know C yet, but maybe we can still make some of the magic happen.

Open up the roach.h file. This file contains all the functions you’ll need to use the roach. Let’s try out some of the basic functions to use the hardware. Feel free to try these in any order!

Note that the roach needs to be switched ON to use most of the hardware. Be sure to turn the roach off at the end of class!

### Motors:

If you’re going to use the motors, be **absolutely** sure that the roach isn’t going anywhere during programming. The roach **MUST MUST MUST** be on a block while you test it!

Try adding these lines to your initialization code in HelloWorld\_main.c:

Roach\_RightMtrSpeed(100);

Roach\_LeftMtrSpeed(-100);

Try some other speeds, too!

QUESTIONS:

How fast can the roach move? (The tiles on the floor are each 1 foot. Be sure to test it in an open space).

How fast can it spin around?

Is the roach faster on carpet than on tile?

What is the lowest number you can pass to Roach\_RightMtrSpeed() and Roach\_LeftMtrSpeed() and still have the roach run? Why do you think this is?

### LEDs:

Try adding this line to your initialization code:

Roach\_LEDSSet(0b1111111111011);

The LEDs should all light.

QUESTIONS:

What do you think the “0b” is for in “0b1111111111111”? If you aren’t sure, try Googling.

See if you can find a way to turn on only the yellow LEDs. Try to turn on every other LED.

Try adding these lines inside the while(1) loop: “Roach\_LEDSSet(0xFFF); Roach\_LEDSSet(0x333); Roach\_LEDSSet(0x333); Roach\_LEDSSet(0x333); “ What happens? Why do you think this happens?

### Bumpers

Try putting the following lines in your while(1) loop:

Roach\_LEDSSet(Roach\_ReadBumpers());

printf("%d \n", Roach\_ReadBumpers());

Press the bumpers and see what happens.

QUESTIONS:

What happens when you press more than 1 button?

### Light Sensor

Try putting the following line in your while(1) loop:

printf("%d \n", Roach\_LightLevel());

QUESTIONS:

What is the light sensor’s reading under your phone’s flashlight? What is its   
reading in complete darkness?

Can you make the roach drive faster in light, and slower in darkness?

1. Was it? We intentionally left a syntax mistake in the printf() line we gave you. See if you can figure it out – it’s one of the most common syntax mistakes in C. [↑](#footnote-ref-1)