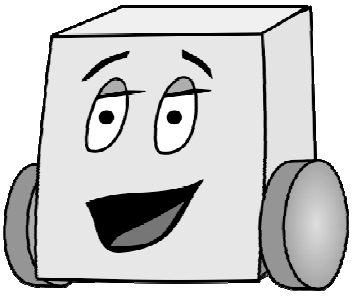


E11: Autonomous Vehicles

Fall 2011

Harris & Harris

PS 1: Welcome to Arduino

This is the first of five programming problem sets. In this assignment you will learn to program the Arduino board that you recently built. If you need help a good reference containing all of the commands you will need to use is the Arduino website at http://arduino.cc/en/Reference/HomePage.

You are welcome to do the problem sets on your own or with a partner; working with a partner is highly encouraged and choosing a different partner each week is even better. You'll need to choose a teammate for the robot competition and it is good to learn who you work well with. If you work with a partner, turn in a single copy of your work with both of your names on it.

Part 0: Installing and Testing the Arduino Software

You may do the problem sets on your own computer or in the PC or Mac labs (e.g. in Linde or Sprague). If you have a laptop running Windows, MacOS, or Linux, you may wish to do the assignments on your own laptop so that you can bring it to class for later labs.

The HMC labs are already setup with the Arduino software. If you prefer to work on your own computer, you will first need to install the Arduino software. You can do so in two ways (choose whichever is easiest for you):

1. The software is located at \\Charlie\Courses\Engineering\E11\Fall2011\code. Grab the PC or Mac version (arduino-mac or arduino-PC). Copy the files and you're ready to go.

2. You can also go to: http://arduino.cc/en/Main/Software and download the files you need.

If you have trouble getting the software working, bring your laptop to tutoring hours or make friends with a computer-savvy neighbor.Begin by opening the arduino application (arduino.exe on Windows). Files in Arduino are organized in folders called *sketches*. Arduino should open a new sketch immediately, with the date as the temporary title. Re-name this sketch by saving it as ps1_0 when prompted. This will create a `ps1_0` folder with `ps1_0.pde` as the only file inside it. Arduino will default to saving all of your

sketch folders in a directory named `Arduino` in your `Documents` folder. This is fine if you are working on your own laptop. It's also convenient because Arduino will easily find your sketches when you select File -> Sketchbook. However, if you are working on an HMC lab computer, be sure to save your sketch in your personal Charlie folder instead so that you'll still be able to access it if you log into a different computer.

Type the following program into your sketch, substituting your own name and date. Save it.

```
// ps1_0
// David_Harris@hmc.edu 5 August 2011
// Test the serial port

void setup()
{
  Serial.begin(9600);

  Serial.println("Hello world!");
}

void loop()
{
}
```

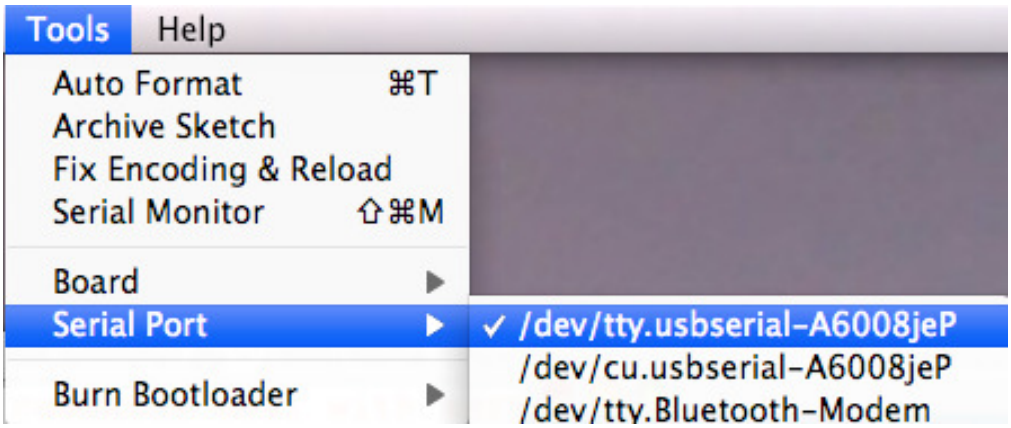
All Arduino programs absolutely must contain the two functions `setup()` and `loop()`. Remember that `setup()` will run just once at the beginning of your program. After that, as the name suggests, `loop()` will continue looping ad infinitum. **Be careful** when you put code in your `loop()` function: if you have not thought it through and make an error it will run forever and most likely be very troublesome to deal with. Be sure to put the line `Serial.begin(9600);` into your `setup()` function. This line tells the Mudduino to initiate a Serial connection at 9600 baud which will allow you to send information back to your computer over the USB cable[1]. Once you have this, you can use the functions `Serial.print()` which will print whatever is passed to it or `Serial.println()` which will do the same, but with a newline at the end.

Plug your Arduino board into a USB port on the computer. Make sure that your board is set to receive power from the USB (i.e., the top switch labeled "Main Power" is switched to the right, labeled "USB"). If the board is getting power, the white or green LED should light up, depending on the position of your Team Select switch. If you are using your own computer, you may be prompted to install the USB drivers. Accept the default options.

Now, make sure Arduino is using the correct board. You may have either the Uno or Duemilanove board. (If you have an Uno, the words "Uno" will probably be written in small letters on the white tape on top of the microcontroller in your board.)_ Use Tools -> Board to choose the target board. Choose either (1) Arduino Uno or (2) Arduino Duemilanove or Nano w/ ATmega328. If you're not sure which one you have, either ask a TA or you can try both options. Now check the port using Tools -> Serial Port. Try

[1] Arduino supports faster baud rates such as 115200. However, the serial monitor defaults to 9600 baud, so it's easiest to leave your program at 9600 rather than having to remember to change the serial monitor.

choosing the first or second port listed. On a Mac, it will have a funny name such as `/dev/tty.usbserial-A6008jeP` (where `A6008jeP` is a seemingly random string of numbers and letters), as shown below. On Windows, it will have a name like COM7.



Click the Upload button in the Arduino window.

You can also do this by pressing Command+U (Mac) or Ctrl-U (PC). You should see a message reading

```
Binary sketch size: 2190 bytes (of a 30720 byte maximum)
```

Then you should see the status bar saying "Uploading to I/O Board…" and eventually "Done uploading." If you get an error, check that your board is powered up and try a different serial port. If the only options for Serial Port that are shown have Bluetooth in the name, ask one of the lab assistants for help.

To check the results in the Serial monitor, click on its icon:

The Serial Monitor window will open up for you. You will unfortunately need to re-open the Serial Monitor every time you upload.

Part 1: The Infamous Harris Numbers

In this section, you will compute and print the infamous Harris numbers. Many of you may be familiar with the Fibonocci numbers, 1, 1, 2, 3, 5, 8, 13, …, described by Leonardo of Pisa in 1202, but only the select few know about the numbers devised by his evil cousin, Doctor Harris. The first two Harris numbers are 1 and 3; each subsequent number is the sum of the last two.

Create a new sketch and save it as ps1_1. **Always include your name, email address, date, and purpose of the file in comments at the beginning of a program.**

We don't want the program to run forever, so only print the Harris numbers that are less than 1000. Before you begin, write out your expected results on paper.

Write your program and test it. All programs large enough to be interesting have bugs in them when first written; it's a common form of beginner's hubris to expect otherwise. Debug your program until it matches your expectations.[2]

Part 2: LEDs!

Your next task is to control the LEDs built into your Mudduino board using commands typed into the Serial Monitor. Specifically, your program should respond to the following commands. For example, if you type 3 into the Serial Monitor, the green LED should turn on.

1. Turns the red LED on
2. Turns the red LED off
3. Turns the green LED on
4. Turns the green LED off

For this problem make a new sketch named ps1_2.

If you study the pin description or schematic of your Mudduino board in Lab 1, you'll see that the board has 20 digital input/output pins, D0 – D19, for interacting with the external world. Several of the pins are tapped out to the header pins on the right and left side of the board. Also, D5 and D13 are connected to the green and red LEDs, respectively.[3]

To drive a pin from your program, you need to configure it as an output using the command `pinMode(pin, OUTPUT);` Include two lines in your setup function to configure pins 5 and 13 as outputs. Don't forget to put `Serial.begin(9600);` in your `setup()` function as well.

To repeatedly check for user input, you can include the following `if` statement in the `loop()` function:

[2] Note that it is wise to have written down your expectations in advance so that you don't fool yourself into believing the program is good when it is not!

[3] Note that pins numbers in the Arduino refer to the logical pin, not the physical pin. For example, pin 0 in an Arduino program refers to logical pin D0, which is actually pin 2 on the 28-pin package.

```
int pressed;
if (Serial.available())
{
  pressed = Serial.read() - 48;
  // now do something with it
}
```

Serial reads its input as characters, and so the integer that it will return is the ASCII value (http://en.wikipedia.org/wiki/ASCII#ASCII_printable_characters) of that character.

The ASCII value 48 corresponds to the character `0`, 49 to `1`, 50 to `2`, and so on. Thus, subtracting 48 from the ASCII value gives the number that was pressed.

You can manipulate the LEDs with `digitalWrite(pin, value)`, where value is `HIGH` or `LOW`. digitalWrite turns the pin completely on or off.

NOTE: Another option is to use `analogWrite(pin, value)`, where `value` should be an `int` from `0` - `255`. Setting `value` as `0` will turn the LED off, and setting `value` to `255` will turn the LED completely on. Intermediate values give intermediate brightnesses using a technique called ***Pulse Width Modulation*** (PWM). PWM rapidly and repeatedly turns the output on and off, such that the output is on for a specified fraction of the total time.

Write, upload, and test your program. Enter your command at the top of the Serial Monitor window and press Send.

Part 3: Randomness

Modify your previous program to make the LEDs blink on and off randomly. Save it as ps1_3.

Instead of taking input from the keyboard, choose a random command using the `random(min, max)` function. Note that the lower bound is inclusive but the upper bound is exclusive! Thus `random(1,5)` would randomly choose a number from 1 through *4*, not 1 through 5.

If the LEDs change too rapidly, your eye will just see a blur. Put a `delay(timeInMilliseconds)` between commands so that the you can see what is happening.

Again, test your program before submitting it. Once you are finished, congratulations! You just finished your first E11 programming assignment!

Deliverables

Please submit the following files to the Program Submission Website (see the E11 course website for the link).

You are responsible for turning in 3 Arduino files:

- `ps1_1.pde`
- `ps1_2.pde`
- `ps1_3.pde`

TIME: Please also keep track of how long it took you to complete this problem set. This won't affect your grade (or even be associated with you) but will help us manage the workload. We will ask you to report this number in class.