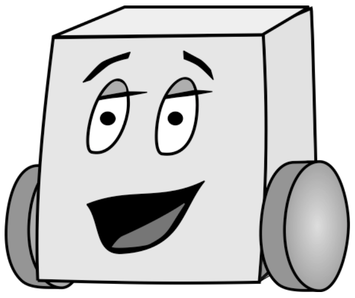


E11: Autonomous Vehicles

Fall 2014

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.

Part 0: Installing and Testing the Arduino Software

We recommend that you install the Arduino tools on your personal laptop so that you'll be able to easily reprogram your bot in the lab. The tools run on Windows, Mac, and Linux.

Download the Arduino software from http://arduino.cc/en/Main/Software. Choose your operating system and grab the corresponding files. Uncompress them into a folder on your computer. If you have trouble, bring your laptop to tutoring hours or make friends with a computer-savvy neighbor.

If you are on a Mac, you will likely need to download and install the FTDI Virtual COM Port (VCP) driver to use the FTDI cable to communicate with your board. You will probably want the latest version for x64 (64-bit) Mac OS X. You will also likely have to adjust your security settings under System Preferences -> Security & Privacy to install the drivers.

http://www.ftdichip.com/Drivers/VCP.htm

If you are a Windows user and have trouble with communication, try installing the Windows Setup Executable version of drivers from the web site above.

Begin by opening Arduino. Files in Arduino are organized in folders called sketches. Arduino should open a new sketch immediately, with the date as the temporary title. Rename this sketch by saving it as ps1_0_`LastName_FirstName` when prompted, where xx are your initials. This will create a `ps1_0_LastName_FirstName` folder with `ps1_0_LastName_FirstName.ino` 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 a 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.

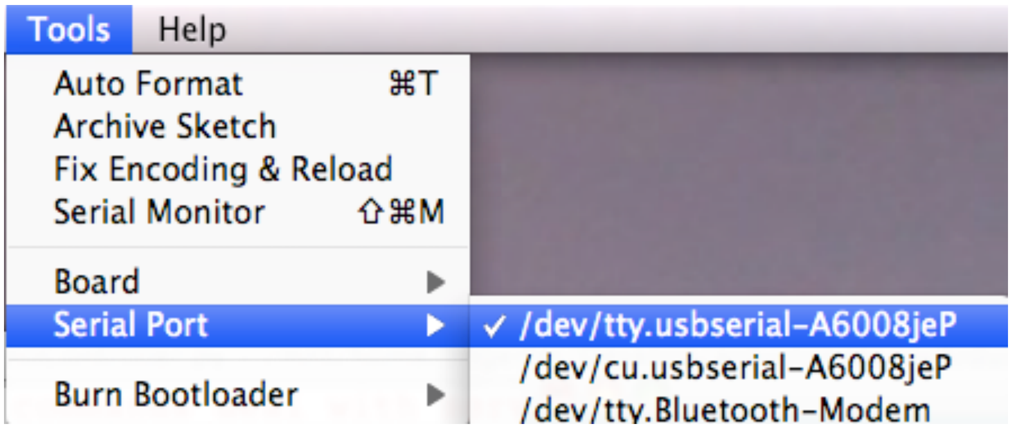
```
// ps1_0
// clark@hmc.edu 9 Sept 2012
// Test the serial port

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 function `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.

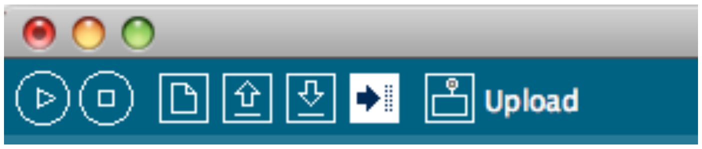
Plug your Arduino board into a USB port on the computer using the FTDI cable. Make sure the black wire end of the adapter plugs into the GND on the Arduino board. If you are using your own computer, you may be prompted to install the USB drivers. Accept the default options. Be sure your "Main Power" switch is set to get power from the USB port. Also, set your "Team Select" switch to the left position. Insert a spare LED in the LED header pins, with the long pin going in the + header. If the board is getting power, the LED should light up.

Use Tools -> Board -> Arduino Uno to tell the tools the type of processor on your board. Now, make sure Arduino is using the correct port using Tools -> Serial Port. Try 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. If there are several COM ports listed on Windows, you may have to try them all until you find the one that works.

[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.



Click the Upload button in the Arduino window.

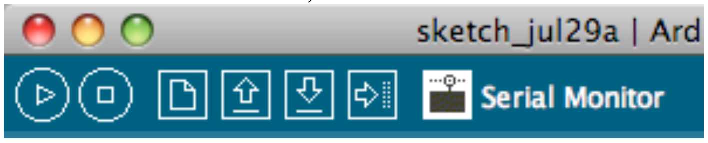


You can also do this by pressing Command+U (Mac) or Ctrl-U (PC). You should see a message reading

```
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. Note that with Mac's you may need to download the driver for the FTDI USB cable.

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_`LastName_FirstName`. **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_LastName_FirstName`.

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, use the following statements in the `loop()` function:

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

[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.

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 `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.

NOTE: Another option is to use `digitalWrite(pin, value)`, where value is `HIGH` or `LOW`. digitalWrite turns the pin completely on or off.

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_`LastName_FirstName`.

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

You are responsible for turning in 3 Arduino files to the “Resources/Problem Set 1/ ” Folder in the E11 page on Sakai:

- `PS1_1_LastName_FirstName.ino`
- `PS1_2_LastName_FirstName.ino`
- `PS1_3_LastName_FirstName.ino`

The files are due before class.

Your code will be graded as follows

- 0.5 points for each program that compiles
- 0.5 additional points for each program that works according to the requirements described above.
- 1.0 additional points for your 3 programs being adequately commented
- This results in 4.0 points maximum