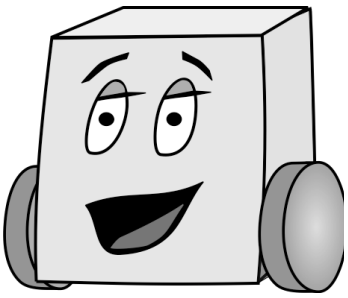

E11: Autonomous Vehicles

Due Wednesday, 9/20 at 11:59pm

PS 1: Welcome to Arduino

This is the first of six 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.

Important: keep track of how long it takes for each program by placing the following comment on the first line of each program file (where xx is the number of hours that it took you):

```
// Time to complete program = xx hours
```

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.[1]

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

[1] 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!

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.[2]

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
}
```

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 LED brightness 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.

[2] 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.

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 on Wednesday 9/20 at 11:59pm.

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