Robotic Merit Badge Session #3

- Merit Badge Counselor: Maurice Ling
- August 3, 2015
- http://bsatroop675.org

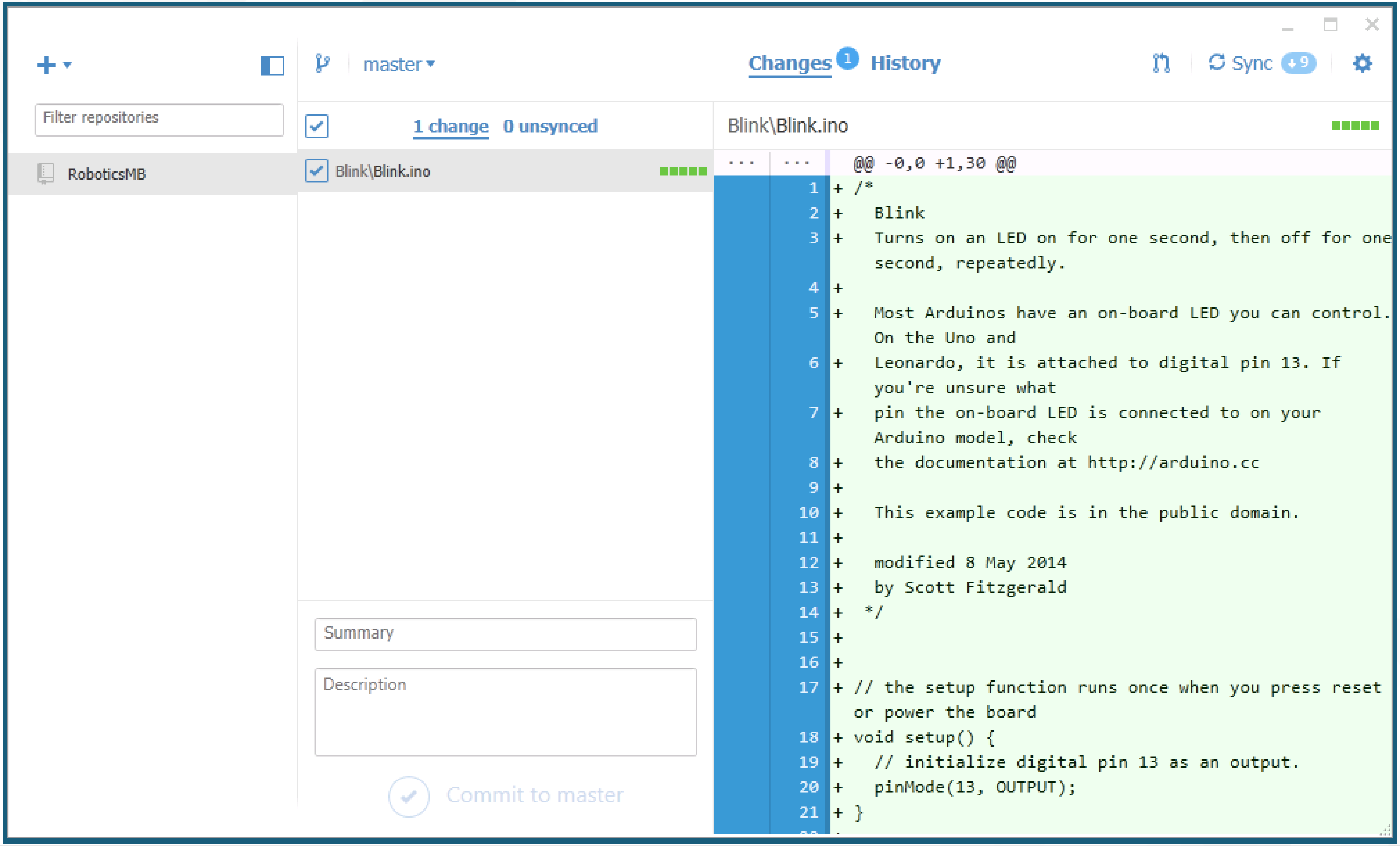
Agenda

- Homework Review
- More Competition Details
- Electronics
  - Basic Circuit Theory
  - Breadboards
- Programming
  - Exercises
- Session #3 Homework

Homework Review

- Using GitHub to check in files and Sync
- Common programming issues
- Using the remote control
- Processing Signals from the Obstacle Sensor
  - What difficulties did you encounter?
  - Techniques to make the signals more manageable.
- Designs for Competition
- Sensor/Motor interaction
- Switching from Auto to Manual control

GitHub Client



```
@@ -0,0 +1,30 @@
+ /*
+   Blink
+   Turns on an LED on for one second, then off for one second, repeatedly.
+
+   Most Arduinos have an on-board LED you can control. On the Uno and
+   Leonardo, it is attached to digital pin 13. If you're unsure what
+   pin the on-board LED is connected to on your Arduino model, check
+   the documentation at http://arduino.cc
+
+   This example code is in the public domain.
+
+   modified 8 May 2014
+   by Scott Fitzgerald
+  */
+
+
+ // the setup function runs once when you press reset or power the board
+ void setup() {
+   // initialize digital pin 13 as an output.
+   pinMode(13, OUTPUT);
+ }
```

Common Programming Issues – “;” and “{}”

- Semicolons (;)
  - Used to end a command
- Curly Braces ({})
  - Used to mark a portion of code that is executed together
- What's wrong with these code segments?

  -
    ```
    if (i == 10);
    {
      Serial.println(i)
    }
    ```

  -
    ```
    for (int j=0; j < 3; j++);
    {
      Serial.println(j);
    }
    ```

Common Programming Issues – “;” and “{}”

- Semicolons (;)
  - Used to end a command
- Curly Braces ({})
  - Used to mark a portion of code that is executed together
- What's wrong with these code segments?

  -
    ```
    if (i == 10); ←Extra Semicolon
    {
      Serial.println(i) ←Missing Semicolon
    }
    ```

  -
    ```
    for (int j=0; j < 3; j++); ←Extra Semicolon
    {
      Serial.println(j);
    }
    ```

Common Programming Issues - “=” vs “==”

- Assignment operator (=)
  - Used to assign a value to a variable

    ```
    int X = 5;
    ```
- Comparison operator (==)
  - Used to compare one value with another

    ```
    if (microM.ircommand == 50)
    {
       // do something
    }
    ```
- What's wrong with this?

  ```
  if (i = 10)
  {
    Serial.println(i);
  }
  ```

Common Programming Issues - “=” vs “==”

- Assignment operator (=)
  - Used to assign a value to a variable

```
int X = 5;
```

- Comparison operator (==)
  - Used to compare one value with another

```
if (microM.ircommand == 50)
{
    // do something
}
```

- What's wrong with this?

```
if (i = 10)  ←Used assignment instead of comparison
{
  Serial.println(i);
}
```

Common Programming Issues – Serial.print/println

- Serial.print/println is useful for seeing what is going on in your program.
- Serial.print()
  - Used to print multiple values on the same line.
- Serial.println()
  - The next print will begin on a new line
- What does the following print?

```
for (int I = 0; I < 10; i++)
{
   Serial.print(i);
   Serial.print(“ “);
}
Serial.println();
Serial.println(“Done”);
```

Using the Remote to Control Motors - Integrating What You Know

From IR_Command example:

```
If (microM.ircommand>0)
  {
    Serial.print("\tIR command:");
    Serial.println(microM.ircommand,DEC);
    microM.ircommand=0;
  }
}
```

Remote Key	IR Command Code
1-9	1-9
0	10
Left	124
Right	125
Up	122
Down	123
Enter	12
Play	51
Pause	58
Stop	57

From DC_Motors example:

```
microM.Motors(leftSpeed,rightSpeed,leftBrake,rightBrake);
```

Using the Remote to Control Motors - Using if/then

```
loop()
{
  // Define variables corresponding to commands
  const int leftCmd=124;
  const int rightCmd=125;
  Int speed=500;

  // Process commands
  If (microM.ircommand == leftCmd)
  {
     microM.Motors(speed,0,0,0);
  } else if (microM.ircommand == rightCmd)
  {
     microM.Motors(0,speed,0,0);
  } else
  {
    Serial.print(“Unprocessed: “);
    Serial.println();
  }
}
```

Using the Remote to Control Motors - Using switch/case

```
loop()
{
  // Define variables corresponding to commands
  const int leftCmd=124;
  const int rightCmd=125;
  Int speed=500;

  // Process commands
  switch(microM.ircommand)
  {
   case leftCmd:
      microM.Motors(speed,0,0,0);
      break;
   case rightCmd:
      microM.Motors(0,speed,0,0);
      break;
   default:
      Serial.print(“Unprocessed: “);
      Serial.println();
  }
}
```

Taming the Sensor Inputs

- Sensor readings include noise from sensors, environment
- Techniques to make the readings more manageable
  - The modulo operator (%)
  - Averaging – smooth out values
  - Threshold – simplify values

Taming the Sensor Inputs – modulo (%)

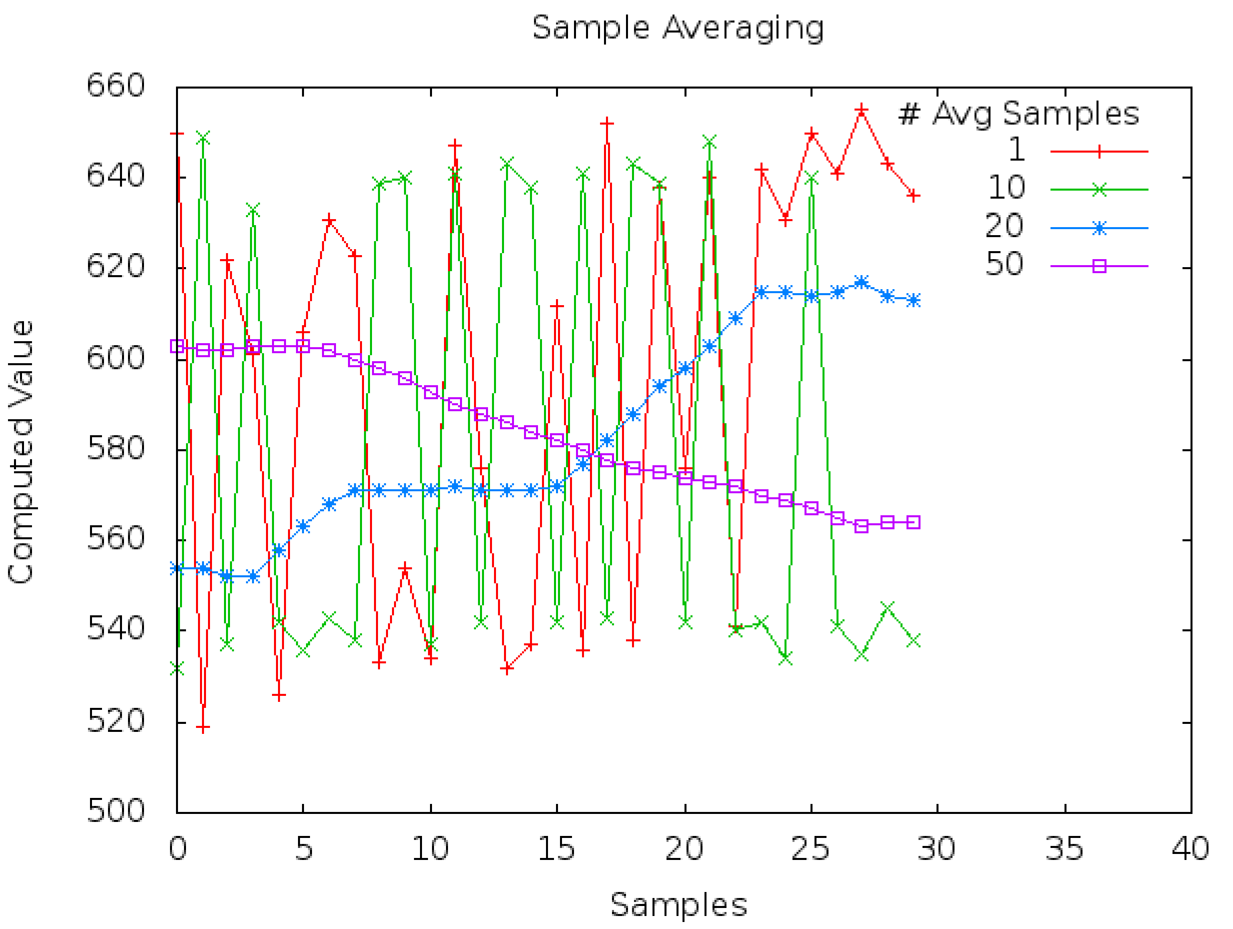
- Calculates the remainder when one integer is divided by another. It is useful for keeping a variable within a particular range (e.g. the size of an array).
- Syntax:
  - result = dividend % divisor
  - What is left over when you divide X by Y?
- What is X?
  - X = 7 % 5;
  - X = 9 % 5;
  - X = 5 % 5;
  - X = 4 % 5;
  - X = 100 % 5;

Taming the Sensor Inputs – modulo (%)

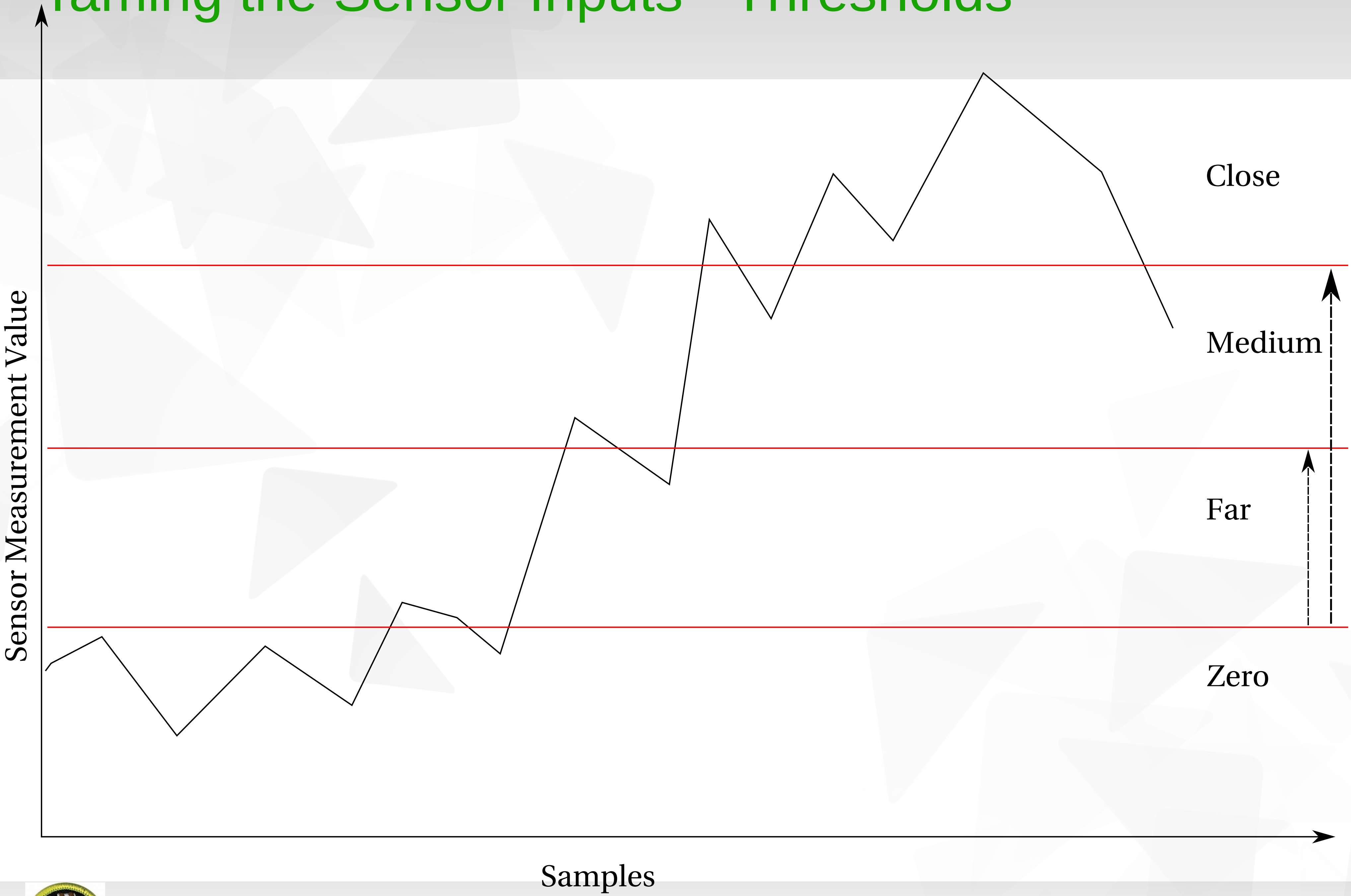
- Calculates the remainder when one integer is divided by another. It is useful for keeping a variable within a particular range (e.g. the size of an array).
- Syntax:
  - result = dividend % divisor
  - What is left over when you divide X by Y?
- What is X?
  - X = 7 % 5;
  - X = 9 % 5;
  - X = 5 % 5;
  - X = 4 % 5;
  - X = 100 % 5;

Taming the Sensor Inputs – Averaging the Input

- Sensor input is noisy. Averaging helps to reduce the noise
- Average = (S[0] + S[1] … S[N])/N



Taming the Sensor Inputs - Thresholds



Competition Review

Robotic Plutonium Carry

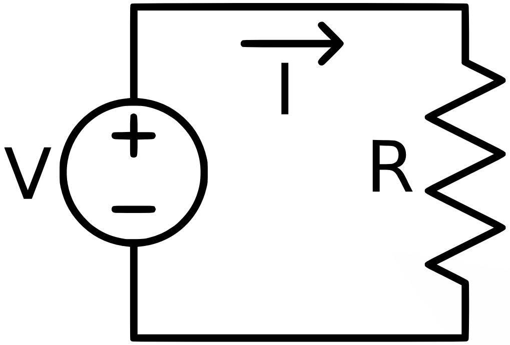
- Objective: Transport radioactive plutonium payload from one reactor to another.
- Payload will take the form of a ping pong ball
- Ball will start on a platform 3 inches high.
- Begin and end with a human-controlled segment to load and unload the capsule.
- Carefully navigate a course autonomously
- If you want to team up, you need to decide now.
- Complete rules in github.

Competition Scoring

- +20 Points scored for each segment completed
- Bonus points for time completed within 3 minutes
- Bonus Design points
- Penalty points for handling payload and manual intervention
- Robot must not be moved, turned, or otherwise transported physically by a human.
- Judge panel consisting of industry professionals.

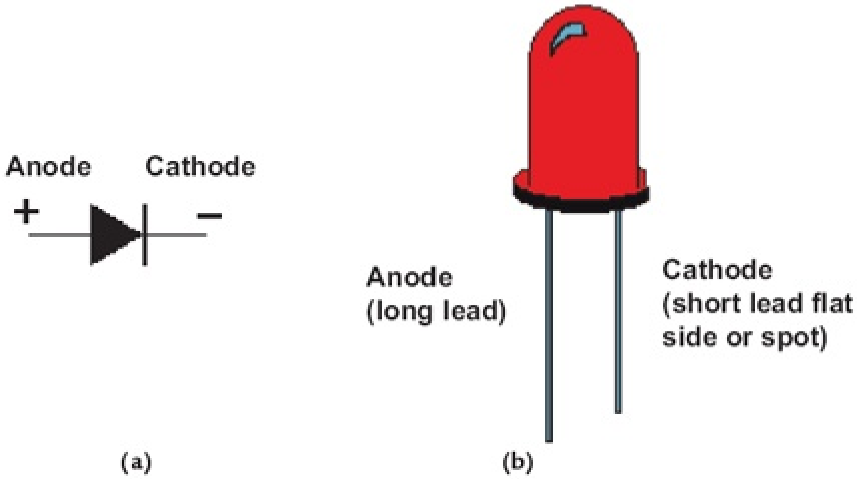
Basic Circuit Theory

- V = Voltage (Volts) – Electrical Energy that powers devices
- I = Current (Amps) – Flow of Electrons through the circuit
- R = Resistance (Ohms Ω) – Slows or opposes the current running through the circuit. Resistors allow you to control the power and current running through your components.
- Ohm's Law: V=IR

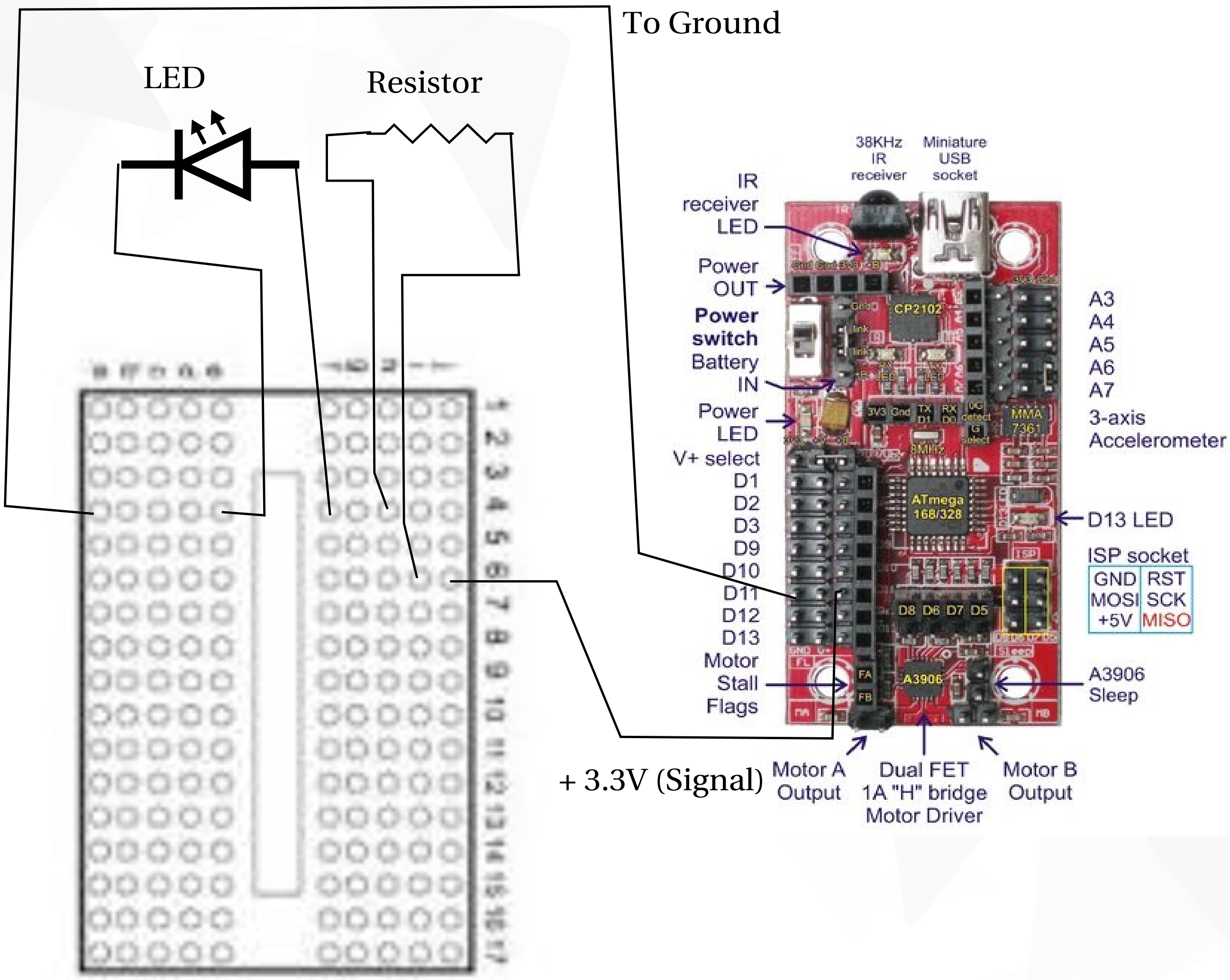


LED's

- LED's are Diodes, which allow current to go in only one direction
- Long lead is the Anode (+)
- Short lead is the Cathode (-)



LED Circuit


To Ground
LED
Resistor
+ 3.3V (Signal)
38KHz IR receiver
Miniature USB socket
IR receiver LED
Power OUT
Power switch
Battery IN
Power LED
V+ select
D1
D2
D3
D9
D10
D11
D12
D13
Motor Stall Flags
A3
A4
A5
A6
A7
3-axis Accelerometer
D13 LED
ISP socket
GND
RST
MOSI
SCK
+5V
MISO
A3906 Sleep
Motor A Output
Dual FET 1A "H" bridge Motor Driver
Motor B Output
CP2102
ATmega 168/328
A3906
MMA 7361
D8 D6 D7 D5
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

Exercises

- Hook up the LED circuit we talked about in class
- Modify your SOS Blink program to blink the LED
- Using the averaging and threshholding algorithms, program your robot based on the left, right, and center normalized values. Example Behavior Table:

L	R	C	Behavior
2	2	2	Full Speed Straight
0	0	0	Back up and turn left
L < C < R			Gradually turn right going slowly
R > C > L			Gradually turn left going slowly

- Test your autonomous navigation with various obstacles and refine your algorithm.

Session #3 Homework

- Complete class exercises
- In your notebook, continue your design for the competition
  - How to load/unload the payload
  - How to carry the payload
  - Sketch the logic required to navigate through an obstacle course
  - Behavior table based on normalized inputs.
- Program your robot to implement your obstacle navigation logic
- Robotics MB Workbook:
  - Work on any uncompleted sections