E11 Lecture 4: More C!!!
Outline

- Analog Inputs
- Randomness
- Operators
- Control Statements
# Mudduino Pinout

<table>
<thead>
<tr>
<th>Digital Pin #</th>
<th>Analog Pin #</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Serial TXD – don’t use</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>Serial RXI – don’t use</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Header D2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Team (0 = green / 1 = white) read only</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Header D4, Buzzer</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Header D5 / green LED / programming indicator</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Left Motor Enable</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Right Motor +</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Left Motor -</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Left Motor +</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Header D10 / Servo (use servo.write)</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Right Motor Enable</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Right Motor -</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>Header D13 / red LED</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>Distance Sensor</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>Header A1</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>Header A2</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>Header A3</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>Header A4, Reflectance Sensor</td>
</tr>
<tr>
<td>19</td>
<td>5</td>
<td>Header A5, Phototransistor</td>
</tr>
</tbody>
</table>
void setup()
{
    Serial.begin(9600);
    pinMode(14, INPUT);  // D14/A0 as input
}

void loop()
{
    int randNum;

    Serial.print("Pin 0: ");
    randNum = analogRead(0);
    Serial.println(randNum);
    delay(800);
}
void setup()
{
    Serial.begin(9600);
    pinMode(5, OUTPUT);  // D5 (green LED)
}

void loop()
{
    analogWrite(5, 0);  // 0 = off
    delay (500);
    analogWrite(5, 127); // 127 = half (2.5V)
    delay(500);
    analogWrite(5, 255); // 255 = full (5 V)
    delay(500);
}
#define

Makes the program easier to read and keep up to date

- no magic numbers!
#define

Makes the program easier to read and keep up to date
- no magic numbers!

So instead of ...

```c
void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT); // red LED pin
}

void loop()
{
  Serial.println("Testing LED");
  digitalWrite(13, HIGH); // turn red LED on
  delay(200);
  digitalWrite(13, LOW); // turn red LED off
  delay(200);
}
```

#define

Makes the program easier to read and keep up to date

- no magic numbers!

We have...

```c
#define REDLED 13
void setup()
{
    Serial.begin(9600);
    pinMode(REDLED, OUTPUT); // red LED pin
}

void loop()
{
    Serial.println("Testing LED");
    digitalWrite(REDLED, HIGH); // turn red LED on
    delay(200);
    digitalWrite(REDLED, LOW); // turn red LED off
    delay(200);
}
```
Pseudo-randomness

```cpp
void setup()
{
    Serial.begin(9600);
    Serial.println("Here are some random numbers between 0 and 43.");
}

void loop()
{
    int randNum;

    randNum = random(0, 43);
    Serial.println(randNum);
    delay(1000);
}
```
Pseudo-randomness

- What happens if you run the program again?
- Random number seed
Your turn!

Write a program that repeatedly plays a random tone (between 200 and 500 Hz) to the speaker for 800 ms. The speaker should then turn off for ½ a second.
#define SPEAKER 4

void setup()
{
    Serial.begin(9600);
    // set speaker pin as output
    pinMode(SPEAKER, OUTPUT);  // speaker pin
}

void loop()
{
    int randNum = random(200, 501);

    tone(SPEAKER, randNum);  // write tone to speaker
    delay(800);  // tone lasts 800 ms
    noTone(SPEAKER);  // turn the speaker (pin 4) off
    delay(500);  // speaker is off for 500 ms
Seeding the Random Numbers

```cpp
void setup()
{
  long randSeed;

  Serial.begin(9600);
  Serial.println("Press a key to begin.");
  while (Serial.available() == 0); // wait until a key is pressed
  randomSeed(micros()); // Seed the random number generator with time
}

void loop()
{
  int randNum = random(0, 43); // set the random number
  Serial.println(randNum); // print the random number
  delay(randNum);
}
```
# Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>addition</td>
<td>y = a + 2;</td>
</tr>
<tr>
<td>-</td>
<td>subtraction</td>
<td>y = a - 2;</td>
</tr>
<tr>
<td>*</td>
<td>multiplication</td>
<td>y = x * 12;</td>
</tr>
<tr>
<td>/</td>
<td>division</td>
<td>z = x / 3;</td>
</tr>
<tr>
<td>%</td>
<td>modulo</td>
<td>z = 5 % 2;</td>
</tr>
<tr>
<td>=</td>
<td>assignment</td>
<td>x = 22;</td>
</tr>
<tr>
<td>==</td>
<td>equals</td>
<td>(y == 2)</td>
</tr>
<tr>
<td>!=</td>
<td>not equals</td>
<td>(x != 7)</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>(y &lt; 12)</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>(val &gt; max)</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal</td>
<td>(z &lt;= 2)</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal</td>
<td>(y &gt;= 10)</td>
</tr>
<tr>
<td>&amp; &amp;</td>
<td>AND</td>
<td>(x &amp; &amp; y)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OR</td>
</tr>
<tr>
<td>!</td>
<td>NOT</td>
<td>!x</td>
</tr>
<tr>
<td>&amp;</td>
<td>bitwise AND</td>
<td>y = a &amp; 15;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bitwise OR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bitwise XOR</td>
</tr>
<tr>
<td>~</td>
<td>bitwise NOT</td>
<td>z = ~x;</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>bit shift left</td>
<td>z = 4 &lt;&lt; 2;</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>bit shift right</td>
<td>x = x &gt;&gt; 8;</td>
</tr>
<tr>
<td>++</td>
<td>increment</td>
<td>a++; // a = a + 1</td>
</tr>
<tr>
<td>--</td>
<td>decrement</td>
<td>x--; // x = x - 1</td>
</tr>
<tr>
<td>+=</td>
<td>addition and assignment</td>
<td>y += 3; // y = y + 3</td>
</tr>
<tr>
<td>-=</td>
<td>subtraction and assignment</td>
<td>z -= 10; // z = z - 10</td>
</tr>
<tr>
<td>*=</td>
<td>multiplication and assignment</td>
<td>x *= 4; // x = x * 4</td>
</tr>
<tr>
<td>/=</td>
<td>division and assignment</td>
<td>y /= 10; // y = y / 10</td>
</tr>
<tr>
<td>&amp;=</td>
<td>bitwise AND and assignment</td>
<td>y &amp; = 15; // y = y &amp; 15</td>
</tr>
<tr>
<td></td>
<td>=</td>
<td>bitwise OR and assignment</td>
</tr>
</tbody>
</table>
## Arithmetic and Comparison

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>+</code></td>
<td>addition</td>
<td>y = a + 2;</td>
</tr>
<tr>
<td><code>-</code></td>
<td>subtraction</td>
<td>y = a - 2;</td>
</tr>
<tr>
<td><code>*</code></td>
<td>multiplication</td>
<td>y = x * 12;</td>
</tr>
<tr>
<td><code>/</code></td>
<td>division</td>
<td>z = x / 3;</td>
</tr>
<tr>
<td><code>%</code></td>
<td>modulo</td>
<td>z = 5 % 2;</td>
</tr>
<tr>
<td><code>=</code></td>
<td>assignment</td>
<td>x = 22;</td>
</tr>
</tbody>
</table>

### Comparison

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==</code></td>
<td>equals</td>
<td>(y == 2)</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>not equals</td>
<td>(x != 7)</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>less than</td>
<td>(y &lt; 12)</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>greater than</td>
<td>(val &gt; max)</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>less than or equal</td>
<td>(z &lt;= 2)</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>greater than or equal</td>
<td>(y &gt;= 10)</td>
</tr>
</tbody>
</table>
# Boolean and Bitwise

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>AND</td>
<td>(x &amp;&amp; y)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>NOT</td>
<td>!x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>bitwise AND</td>
<td>y = a &amp; 15;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bitwise OR</td>
</tr>
<tr>
<td>^</td>
<td>bitwise XOR</td>
<td>y = a ^ b;</td>
</tr>
<tr>
<td>~</td>
<td>bitwise NOT</td>
<td>z = ~x;</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>bitshift left</td>
<td>z = 4 &lt;&lt; 2;</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>bitshift right</td>
<td>x = x &gt;&gt; 8;</td>
</tr>
</tbody>
</table>
## Compound Operations

<table>
<thead>
<tr>
<th>Compound</th>
<th>Symbol</th>
<th>Operation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>++</td>
<td>increment</td>
<td>a++; // a = a+1</td>
</tr>
<tr>
<td></td>
<td>--</td>
<td>decrement</td>
<td>x--; // x = x -1</td>
</tr>
<tr>
<td></td>
<td>+=</td>
<td>addition and assignment</td>
<td>y += 3; // y = y + 3</td>
</tr>
<tr>
<td></td>
<td>-=</td>
<td>subtraction and assignment</td>
<td>z -= 10; // z = z - 10</td>
</tr>
<tr>
<td></td>
<td>*=</td>
<td>multiplication and assignment</td>
<td>x *= 4; // x = x * 4</td>
</tr>
<tr>
<td></td>
<td>/=</td>
<td>division and assignment</td>
<td>y /= 10; // y = y / 10</td>
</tr>
<tr>
<td></td>
<td>&amp;=</td>
<td>bitwise AND and assignment</td>
<td>y &amp;= 15; // y = y &amp; 15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>=</td>
<td>bitwise OR and assignment</td>
</tr>
</tbody>
</table>
Operators Example

```c
int z, x = 14; int y = 43; // x = 1110, y = 101011

z = y / x;
z = y % x;
z = x && y;
z = x && 0;
z = x || y;
z = x || 0;
z = x & y;
z = x | y;
z = x ^ y;
z = x << 2;
z = y >> 3;
x += 2;
y &= 15;
```
Control Statements

- if
- if / else
- switch / case
- while
- do / while
- for
if (i == 25) {
    Serial.println("You guessed the magic number!");
}
y = 42;
if (i == 25) {
    Serial.println("You guessed the magic number!");
}
else {
    Serial.println("Try again!");
}
switch (var) {
    case 0:
        Serial.println(“Nice choice!”);
        break;
    case 1:
        Serial.println(“I wouldn’t have done that!”);
        break;
    default:
        Serial.println(“You pressed an invalid number”);
}
while Statement

```cpp
int x = 1;
while (x < 1000) {
    Serial.println(x);
    x = x*2;
}
```
do / while Statement

int x = 0;

do {
    delay(100);  // delay 100 ms between readings
    x = analogRead(0);
} while (x < 300);
for Loop

for (initialization; condition; loop operation)

  loop body

- **initialization**: executes before the loop begins
- **condition**: is tested at the beginning of each iteration
- **loop operation**: executes at the end of each iteration
- **loop body**: executes each time the condition is met
for Loop

```c
int i;
int x = 1;

for (i = 2; i < 10; i++)
    x = x * i;
```
Write a program that turns on an LED for a length of time depending on a user input of 1, 2, or 3. The choices correspond to LED on times of 300, 800, or 2000 ms. The LED should then turn off for at least ½ a second until the next user input.

Assume you already have the user input: `int choice;` choice is 0 if there is no user input.
switch(choice) {
    case 0: break;
    case 1:
        digitalWrite(REDLED, HIGH); // turn red LED on
        delay(300);
        break;
    case 2:
        digitalWrite(REDLED, HIGH); // turn red LED on
        delay(800);
        break;
    case 3:
        digitalWrite(REDLED, HIGH); // turn red LED on
        delay(2000);
        break;
    default:
}

if (choice) {
    digitalWrite(REDLED, LOW); // turn red LED off
    delay(500);
}