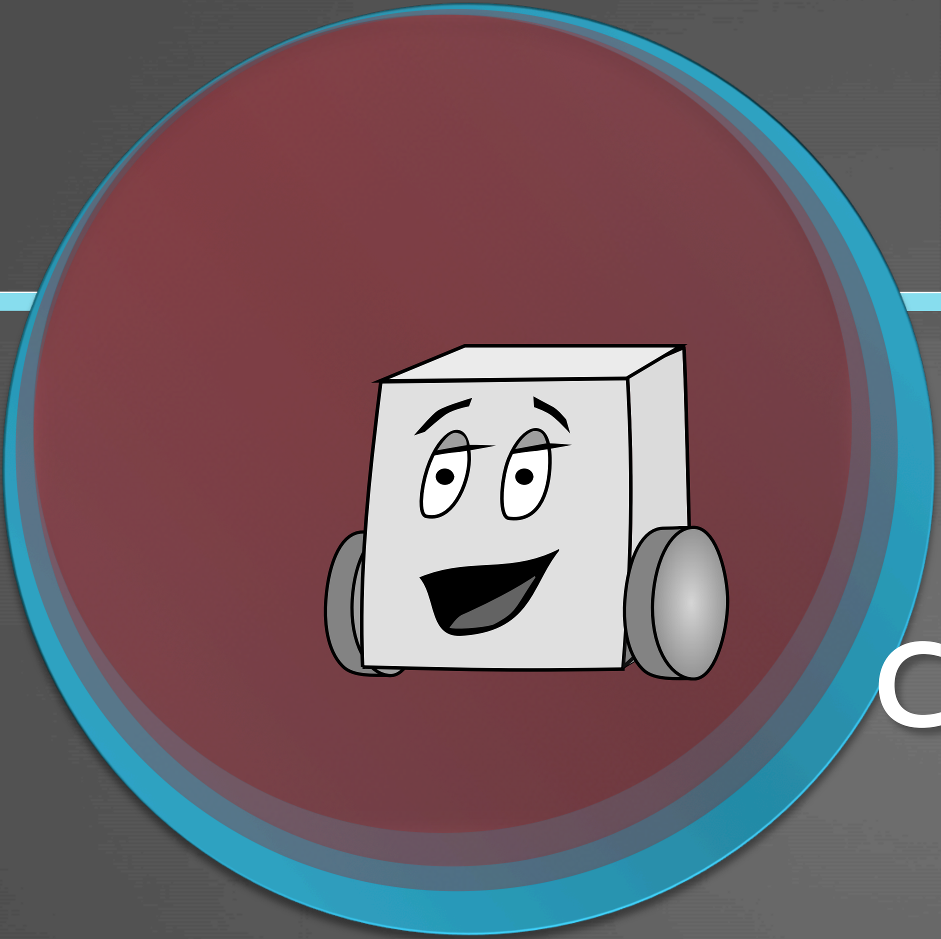


E11 Lecture 8: C – never enough!

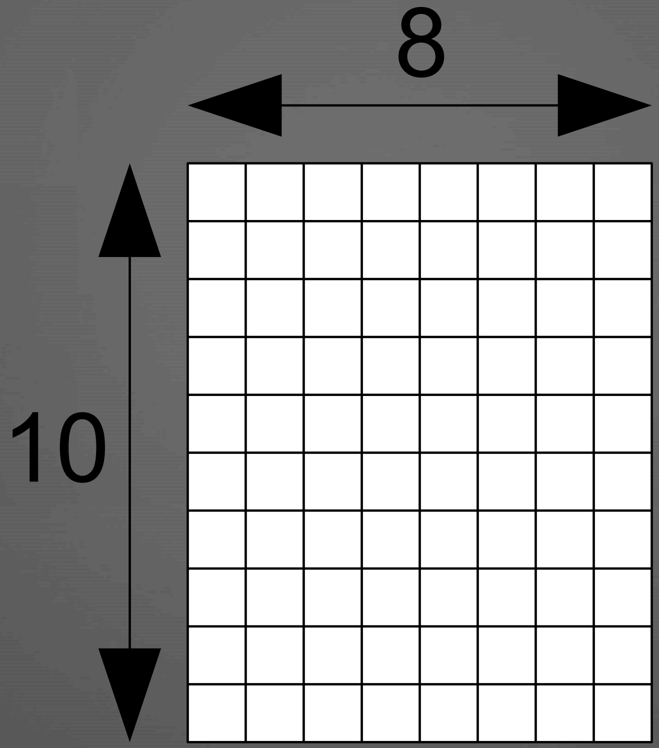
Prof. David Money Harris
Fall 2014

Outline

- Multi-dimensional arrays
- Testing the limits
- Programming Practice
- Nuts and bolts
  - Multiple files
    - other C files
    - #include
  - Other useful functions

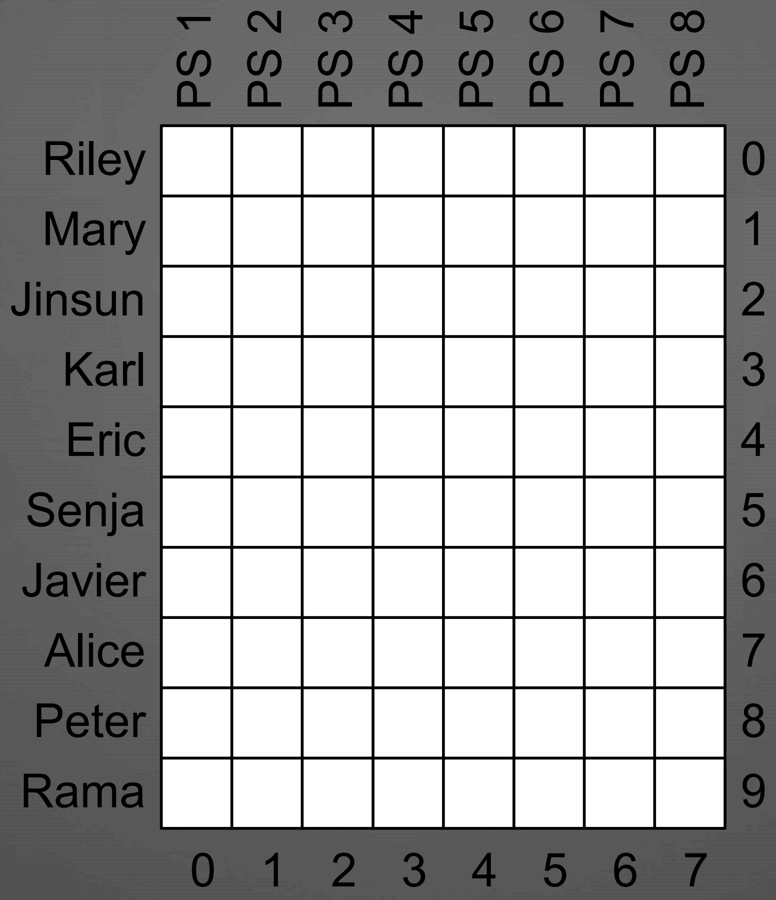
Multi-dimensional Arrays

```
int grades[10][8];
```



Multi-dimensional Arrays

```
int grades[10][8];
```



Multi-dimensional Arrays

```
int grades[10][8];


// initialize all entries in array to 0
int i, j;

for (i=0; i<10; i++)
  for (j=0; j<8; j++)
    grades[i][j] = 0;
```

Multi-dimensional Arrays

```
// initialize array at declaration
int grades[10][8] =
   { {100, 107, 99,  101, 100, 104, 109, 117},
     {103, 101, 94,  101, 102, 106, 105, 110},
     {101, 102, 92,  101, 100, 107, 109, 110},
     {114, 106, 95,  101, 100, 102, 102, 100},
     {98,  105, 97,  101, 103, 104, 109, 109},
     {105, 103, 99,  101, 105, 104, 101, 105},
     {103, 101, 100, 101, 108, 105, 109, 100},
     {100, 102, 102, 101, 102, 101, 105, 102},
     {102, 106, 110, 101, 100, 102, 120, 103},
     {99,  107, 98,  101, 109, 104, 110, 108} };
```

Multi-dimensional Arrays

```
// get the mean for a problem set and overall
 for (i=0; i<8; i++) {      // for each of the 8 problem sets
   total_tmp = 0;
   for (j=0; j<10; j++) {
     total_tmp += grades[j][i];  // calculate sum of scores
   }
   mean_ps[i] = total_tmp/10;    // calculate p.s. mean
   Serial.print("Problem Set "); Serial.print(i+1);
   Serial.print(": "); Serial.println(mean_ps[i]);

   mean_overall += total_tmp;    // sum all the scores
 }
 mean_overall = mean_overall/(10*8); // calculate overall mean
 Serial.print("Overall mean:"); Serial.println(mean_overall);
```

Testing the Limits

- Atmega328
  - Program memory: 32 KB of Flash Memory (retains value when powered off)
  - Data memory: 2 KB of static random access memory (SRAM) (loses value when powered off)

Data memory: 2 KB

- How big of an int array can I declare?
  - 2048 Bytes/(2 Bytes/element) = 1024-element array
  - But also other data (bootloader, Serial library data, etc.) – so can't use entire 2 KB.

Data memory: 2 KB

- How big of an int array can I declare?
  - 2048 Bytes/(2 Bytes/element) = 1024-element array

```
// datalimit.pde  - 19 September 2011
// Sarah Harris - sarah_harris@hmc.edu
// testing limits on data

#define SIZE 800

int array[SIZE];  // vary array size to see what happens

void setup() {
  int i;

  Serial.begin(9600);  Serial.println("Starting program...");

  for (i = 0; i < SIZE; i++) {
    array[i] = random(0,101);
    Serial.println(array[i]);
  }
```

Data memory: 2 KB

- How big of an int array can I declare?
  - 2048 Bytes/(2 Bytes/element) = 1024-element array

```
// datalimit.pde  - 19 September 2011
// Sarah Harris - sarah_harris@hmc.edu
// testing limits on data

#define SIZE 900

int array[SIZE];  // with size of 900, program starts behaving randomly

void setup() {
  int i;

  Serial.begin(9600);  Serial.println("Starting program...");

  for (i = 0; i < SIZE; i++) {
    array[i] = random(0,101);
    Serial.println(array[i]);
  }
```

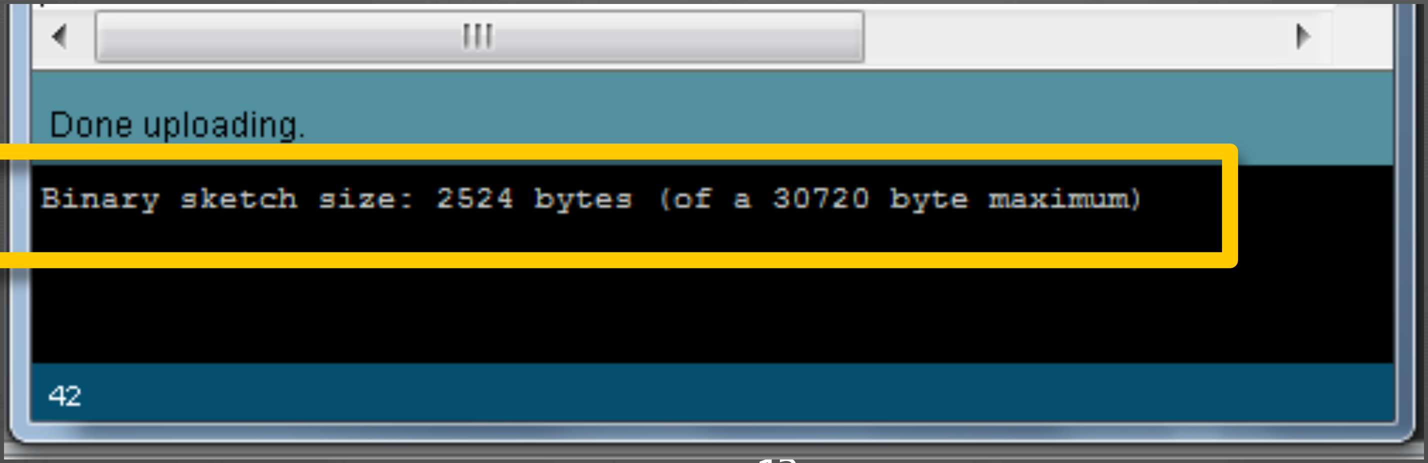
Data memory: 2 KB

- How big of an int array can I declare?
  - 2048 Bytes/(2 Bytes/element) = 1024-element array

```
// datalimit.pde  - 19 September 2011
// Sarah Harris - sarah_harris@hmc.edu
// testing limits on data

#define SIZE 1000

int array[SIZE];  // at 1000, program acts as if uploads but doesn't

void setup() {
  int i;

  Serial.begin(9600);  Serial.println("Starting program...");

  for (i = 0; i < SIZE; i++) {
    array[i] = random(0,101);
    Serial.println(array[i]);
  }
```

Program memory: 32 KB

- **How big** can program be?
  - Many instructions – can look at size when compiling or uploading
  - Some of it used by bootloader (1/2 KB)
  - Some used by libraries (like Serial library)



Done uploading.

```
Binary sketch size: 2524 bytes (of a 30720 byte maximum)
```

42

Outline

- Timing
- Multi-dimensional arrays
- Testing the limits
- Programming Practice
- Nuts and bolts
  - Multiple files
    - other C files
    - #include
  - Other useful functions

Programming Practice

- How do you approach writing a program?

Programming Practice

- How do you approach writing a program?
- Before you sit in front of a computer:
  - Write down the steps of the program (in English)
    - Start with major steps, then break them down into smaller steps
- Work on one step at a time
  - Write code (using functions – modularity!)
  - Test that small piece of code thoroughly
  - Then move on to the next step

Nuts and Bolts: Multiple Files

- Enables:
  - organization
  - code reuse

Multiple Files in a Single Sketch

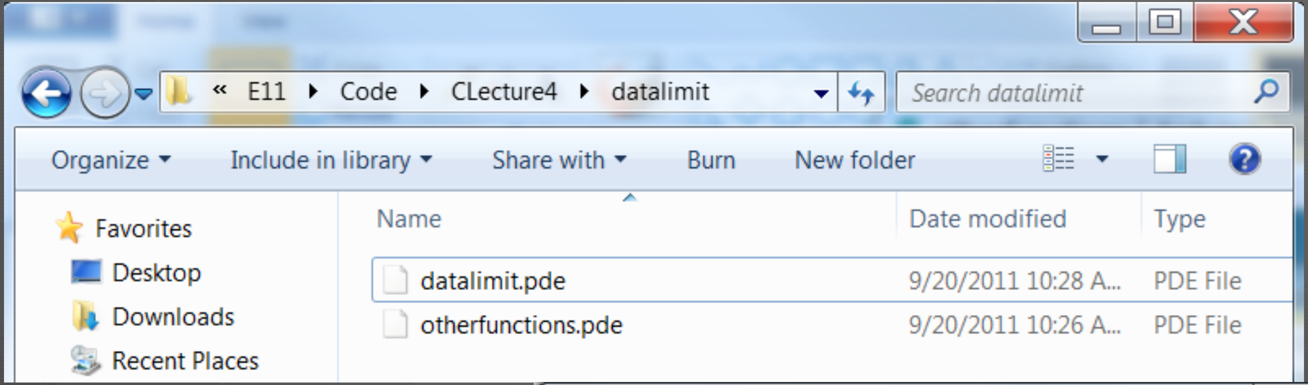
- For example, you may have a group of functions that you consistently use.
- By adding the .pde file to the sketch, you can use any of the functions.
- Be sure you only have extra functions in your added .pde – not `setup()` or `loop()`.

Multiple Files in a Single Sketch

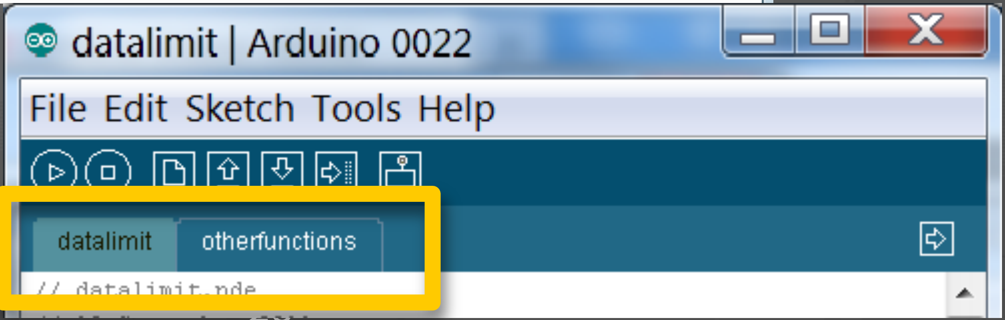
```
// otherfunctions.pde
void printArray(int array[], int length)
{
…
}

int getKeyPress()
{
…
}
```

Multiple Files in a Single Sketch

- How to do this – two ways:
  1. Place extra .pde file in the sketch folder. (Now it will show up as a tab in the sketch, and you can use the functions.)



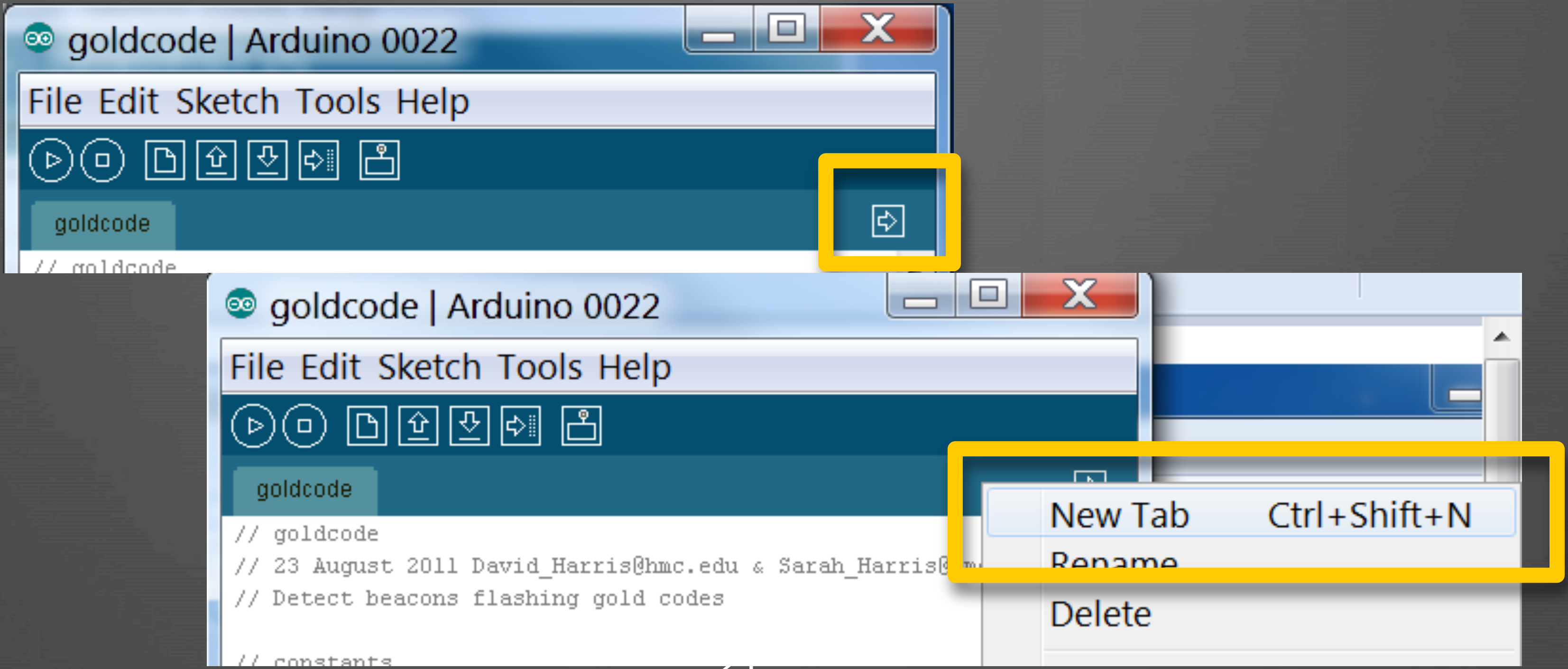
Name	Date modified	Type
datalimit.pde	9/20/2011 10:28 A...	PDE File
otherfunctions.pde	9/20/2011 10:26 A...	PDE File



```
// datalimit.pde
```

Multiple Files in a Single Sketch

- How to do this – two ways:
  2. Add a tab yourself manually and type in the functions in that tab.



```
// goldcode
// 23 August 2011 David_Harris@hmc.edu & Sarah_Harris@
// Detect beacons flashing gold codes

// constants
```

Multiple Files in a Single Sketch

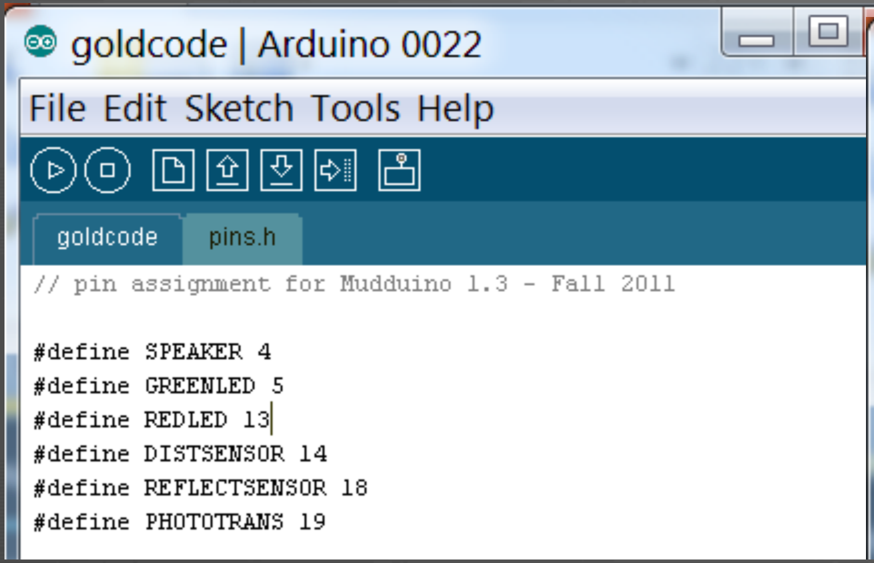
- Remove the file from the sketch by simply removing it from the sketch folder.

Multiple Files in a Single Sketch

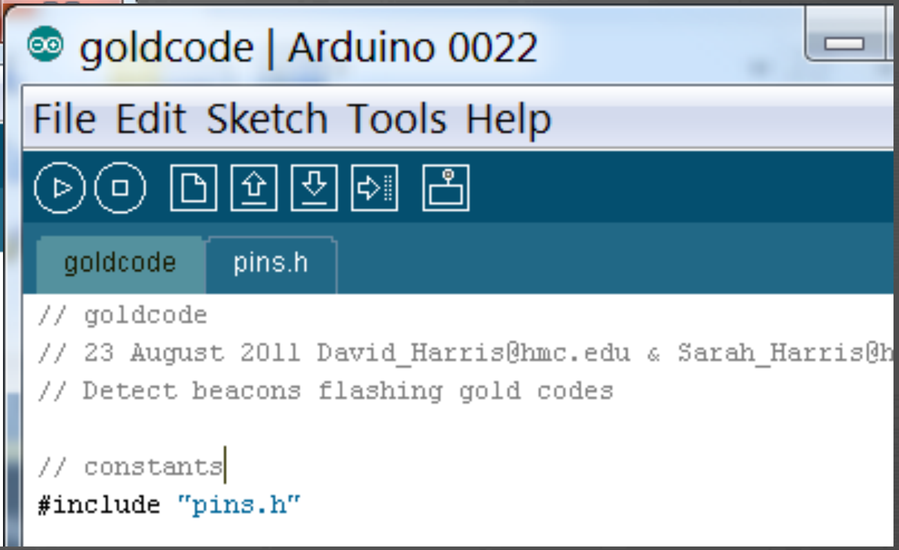
- Or you may have some #defines that you consistently use.

Multiple Files in a Single Sketch

- Or you may have some #defines that you consistently use.
  1. Add new tab
  2. Name it with a ".h" extenstion. For example, pins.h
  3. Place this line in .pde file: #include "pins.h"



```
// pin assignment for Mudduino 1.3 - Fall 2011

#define SPEAKER 4
#define GREENLED 5
#define REDLED 13
#define DISTSENSOR 14
#define REFLECTSENSOR 18
#define PHOTOTRANS 19
```



```
// goldcode
// 23 August 2011 David_Harris@hmc.edu & Sarah_Harris@h
// Detect beacons flashing gold codes

// constants
#include "pins.h"
```

Some other useful functions

http://arduino.cc/en/Reference/HomePage

- abs(var) – returns the absolute value of var
  - Example:

```
int y = -20;
int x = abs(y);  // x = 20
```

- min(x, y) – returns the minimum of x or y
  - Example:

```
int x = 4;
int y = 2;
int minimum = min(x, y);  // minimum = 2
```

- casting characters: char(x), int(x), long(x), float(x)
  - Casts x to the corresponding type
  - Example:

```
char x = 2;        // x is a 1-byte data type:  00000010
int y = int(x);    // y is a 2-byte data type:  00000000 00000010
```