Arduino Programming Part 2

EAS 199A
Lecture 6
Fall 2011

Overview

- Variable types
  - int
  - float
- Loops
  - for loops
  - while loops (another day)

Assigning and Using Variables

Arduino web site

- http://arduino.cc/en/Reference/HomePage
- http://www.arduino.cc/en/Tutorial/Variables
- http://arduino.cc/en/Tutorial/Foundations

The more common variable types are

- integers:
  - int, long, unsigned int, unsigned long
- floating point values: (numbers with fractional parts
  - float, double
- characters and character strings
  - char, string, String
- arrays

Integers are used for counting

`int`

- integers in the range –32,768 to 32,767

`unsigned int`

- integers in the range 0 to 65,535

`long`

- integers in the range –2,147,483,648 to 2,147,483,647

`unsigned long`

- integers in the range 0 to 4,294,967,295

Practical usage of `int` and `long`

Use an `int` for most common tasks requiring integers

- Use an int for most loop counters:

```
int  i, n=16;

for ( i=0; i<n; i++) {
   //  loop body
}
```

- An int is returned by a built-in functions, e.g. analogRead

```
int val, photo_pin=4;
val = analogRead(photo_pin);
```

Practical usage of `int` and `long`

Use a `long` when the range of values is very large, e.g. measuring the system time in milliseconds

```
long start_time, current_time;
long wait_time = 86400000;  // one day

void setup() {
  start_time = millis();
  Serial.begin(9600);
}

void loop() {
  current_time = millis();
  if ( (current_time - start_time) > wait_time ) {
     Serial.println(“24 hours has passed”);
     start_time = current_time;
  }
}
```

Floating point numbers are used for computing with fractional values

`float`

- numbers with fractional part
- values in the range –3.4028235 × 10^{38} to 3.4028235 × 10^{38}

Practical advice

- Use a `float` in formulas when fractional values are needed
- A `float` can be very large or small
- floating point math involves small rounding errors

Integer and floating point variables use different arithmetic rules

Integer math: Division rounds to nearest int

```
int  a, b, c;

a = 4;
b = 3;
c = a/b;     //  Value of 1 is stored in c
```

Floating point math

```
float  x, y, z;

x = 4.0;     //  Include “point zero” to reinforce
y = 3.0;     //     that x and y are floats
z = x/z;     //  Value of 1.3333333 is stored in z
```

Use conversion functions to change type

Convert to an integer:

```
a = int(x);
```

Convert to a floating point value:

```
x = float(i);
```

Practical Advice

Use explicit type conversion functions to convey your intent

Defining and Using Variables

- All variables must be declared before use
- Declaration consists of a type specification and the variable name
- A declaration may also include an assignment
- Use meaningful variable names
- Add comments to further clarify meaning

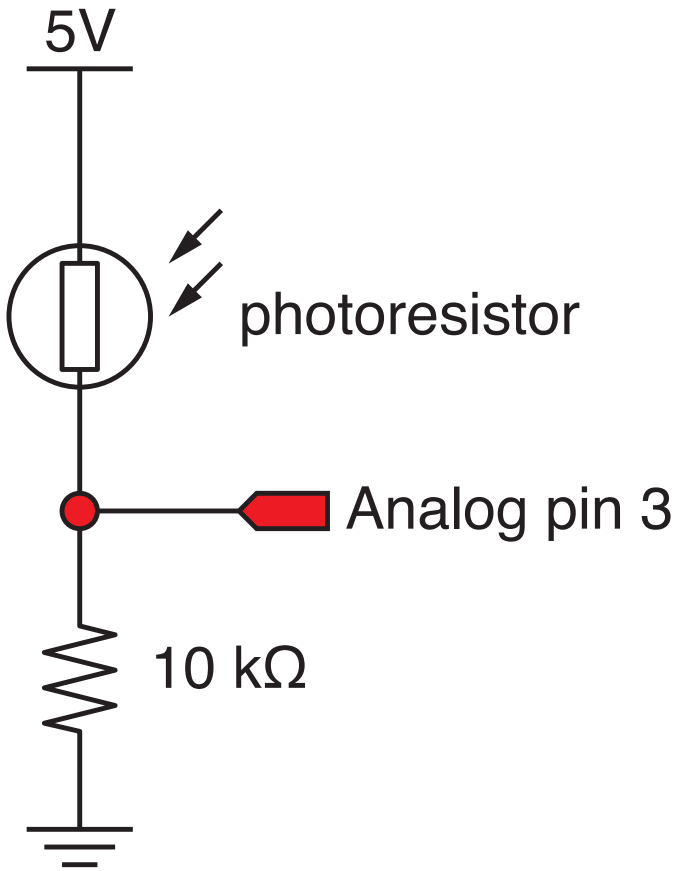
Examples

```
int    red_pin;           //  declaration only
int    blue_pin = 5;      //  declaration and assignment
int    greenPin = 0;

float voltage;           //  Voltage of the input signal
float maxVoltage = 5.0;  //  Maximum range of analog input

sensorVal = analogRead(sensorPin);     // get reading

// convert to floating point voltage
voltage = float(sensorVal)*maxVoltage/float(range);
```

Case study: Use floats to store sensor values

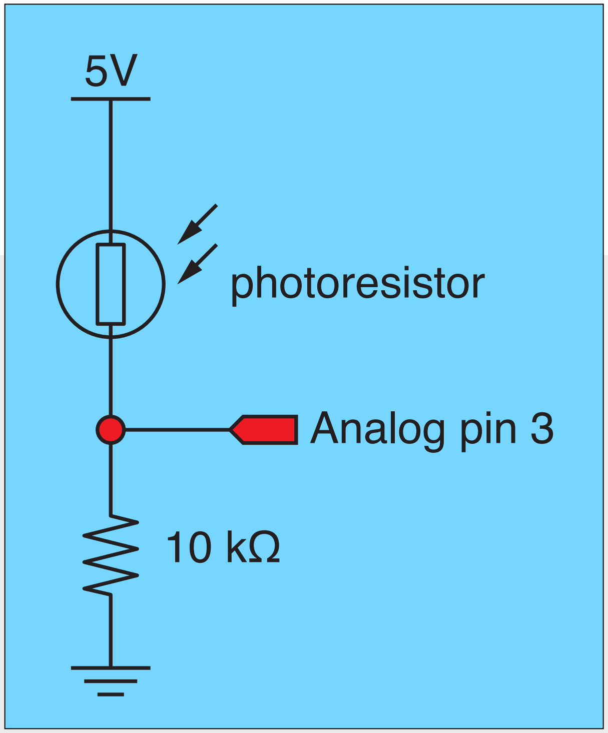
Use photo-resistor circuit to create sensor input

- Convert input reading to a voltage using floating point variables
- Use loops to compute the average of sensor readings



Try it! Measure photoresistor output

Build the photo-resistor circuit
and run this program



```
int   sensorVal;
int   sensorPin = 3;
float voltage;
float input2volts = 5.0/1023.0;

void setup () {
  Serial.begin(9600);
}

void loop () {
  sensorVal = analogRead(sensorPin);
  voltage = float(sensorVal)*input2volts;
  Serial.print(“sensorVal, voltage = “);
  Serial.print(sensorVal);  Serial.print(“  “);
  Serial.println(voltage);
}
```

Loops

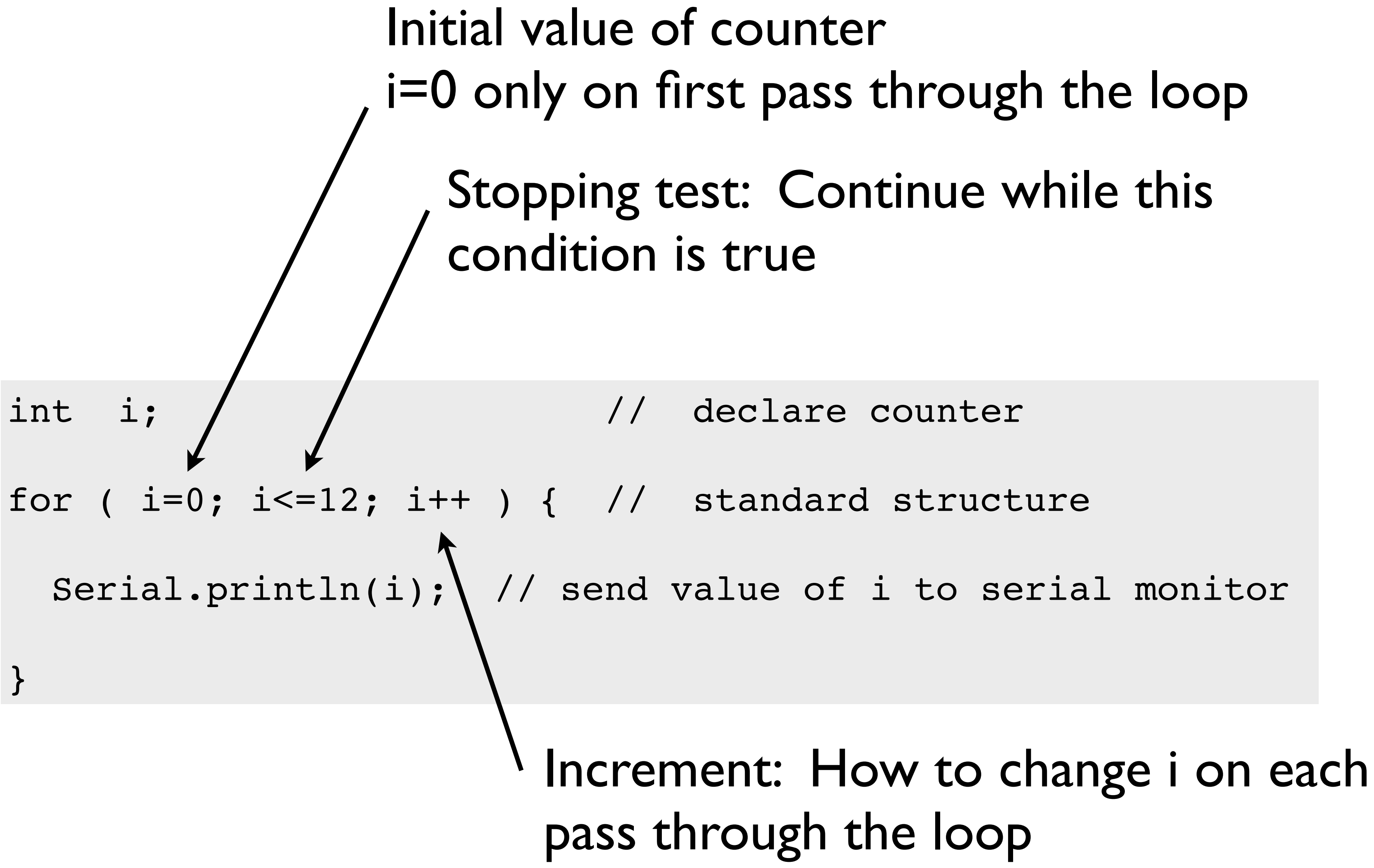
Loops

Loops allow code to be repeated

- Repeated code goes in a block, surrounded by { }
- for loops
    - need a counter
- while loops
    - need an escape

```
int  i;                    //  declare counter
for ( i=0; i<=12; i++ ) {  //  standard structure
  Serial.println(i);   // send value of i to serial monitor
}
```

Loops



Initial value of counter
i=0 only on first pass through the loop

Stopping test: Continue while this condition is true

```
int  i;                        //  declare counter

for ( i=0; i<=12; i++ ) {  //  standard structure

  Serial.println(i);  // send value of i to serial monitor

}
```

Increment: How to change i on each pass through the loop

Loops

Common loop: increment by one

```
for ( i=0; i<=12; i++ ) {  //  increment by one
  ... code block goes here
}
```

Common loop: increment by two

```
for ( i=0; i<=12; i+=2 ) {  //  increment by two
  ... code block goes here
}
```

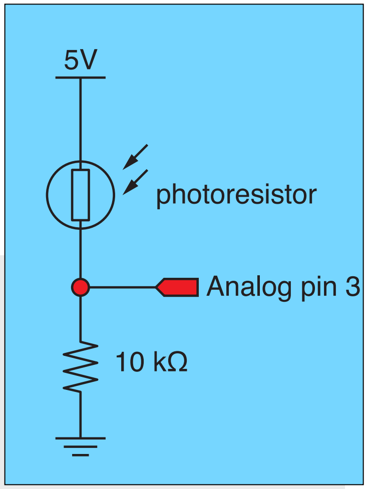
Decrement by one

```
for ( i=12; i>=0; i-- ) {  //  decrement by one
  ... code block goes here
}
```

Try it!   Modify the photoresistor program

Change the loop function

(modify your previous code)



```
void loop () {
  float sensorAve;
  int   sensorSum;
  int   nave=5;

  sensor_sum = 0.0;
  for ( i=1; i<=nave; i++ ) {
    sensorVal = analogRead(sensorPin);
    sensorSum = sensorSum + sensorVal;
  }
  sensorAve = float(sensorSum)/float(nave);
  voltage = sensorAve*input2volts;
  Serial.print(“Average voltage = “);
  Serial.println(voltage);
}
```

This code contains errors that you will need to fix before it runs!

Test it! Break your code to learn how it works

Change nave

- Increase `nave` from 5 to 10, 50, 100, 500
- Why is the reading negative for large nave?
- How can you fix this by changing the variable type for `sensorSum`?

Add print statements inside the averaging loop

```
Serial.print(“\t Reading = “);
Serial.println(sensorVal);
```