One More Control Structure

case statement

```
switch (var) {
   case 1:
      //do something when var equals 1
      break;
    case 2:
      //do something when var equals 2
      break;
    case 86:
      //do something when var equals 86
      // you can jump around!
      break;
    case someConstantName:
      //do something when var equals
      // a constant defined at the top
      // of your code
      // you can jump around!
      break;
    default:
      // if nothing else matches, do the default
      // default is optional
  }
```

Analog I/O Examples

analogRead()

```
void loop() {
     // read the input pin
     val = analogRead(analogPin);
     // debug value
     Serial.println(val);
}
```

analogWrite()

```
//must be on one of the PWM Pins
//9,10,11 NEW-> 3,5,6
//Must be a value 0-255
analogWrite(ledPin, 255);
```

map()

```
//linear mapping (i.e. normalization function)
blinkOnPeriod = map(sensorValue, sensorMin,
sensorMax , blinkShortest , blinkLongest);

//non variable pimped out
byte myPWM = map(sensorValue, 0, 1023, 0, 255);
```

old way was something like:

(newMax - newMin) / (oldMax-oldMin) * valueToBeMaped

constrain()

```
//truncates values to fit
int prntblChar = constrain(inByte,32,126);
```

Serial Sending

Serial.begin()

common rates & size variable it would take to hold them:

int	300
int	1200
int	4800
int	9600
int	14400
int	19200
int	28800
word	38400
word	57600
long	115200

```
int baudrate = 9600;

void setup() {
     // read the input pin
     val = analogRead(analogPin);
     // debug value
     Serial.println(val);
}
```

there is a Serial.end but it is uncommon, especially wen the begin is only in the setup!

Serial.print()

```
//how each of these would handle
someValue = 65;

//depending on what you send it to
//might give you a "A"
Serial.print(someValue, BYTE);

//ASCII encoded binary "1000001"
Serial.print(someValue, BIN);

//ASCII encoded decimal "65"
Serial.print(someValue, DEC);

//ASCII encoded hexadecimal "41"
Serial.print(someValue, HEX);

//ASCII encoded octal notation "101"
Serial.print(someValue, OCT);

// print a tab, ASCII 9
Serial.print('\t');

//print a line feed, ASCII 10
Serial.print('\n');

//print a carriage return, ASCII 13
Serial.print('\r');
//more common to just use…
```

Serial.println()

```
someValue = 65;

//prints a 65 followed by a
Serial.println(someValue, DEC);
```

Serial.write()

```
someValue = 65;

//depending on what you send it to
//might render as "A" but it is just the
//idea of 65, less than a byte of
//information, vs "65" which is two
//bytes
Serial.write(someValue);
```

Serial Receiving:

Serial.available()

```
//if there is nothing waiting for me to read…
void establishContact() {
  while (Serial.available() <= 0) {
       Serial.println("hello");
       delay(300);
   }
 }

//or if you want to know how much
//is in the buffer
//(buffer holds up to 128 bytes)
byte bytesWaiting = Serial.available;
```

Serial.read()

```
//print what you receive

byte incomingByte;

//using while instead of if for this will
//stick the program here until its done clearing
//the buffer. Can be a better idea to use if's
//and for loops depending what you're up to..

while (Serial.available() > 0) {
   // read the incoming byte:
   incomingByte = Serial.read();
   // say what you got:
   Serial.print("I received: ");
   Serial.println(incomingByte, DEC);
}
```

Serial.flush()

```
//will take the first byte from the buffer...
if (Serial.available() > 0) {
   // read the incoming byte:
   incomingByte = Serial.read();
   // say what you got:
   Serial.print("I received: ");
   Serial.println(incomingByte, DEC);
}
//... and then discards the rest of the
//buffer so it'll be a fresh batch
//the next time you hit this code
Serial.flush;
```