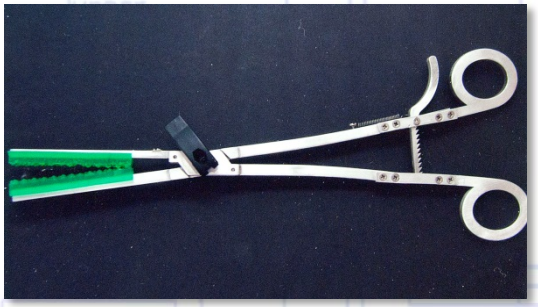
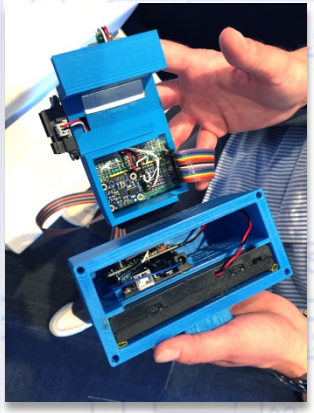
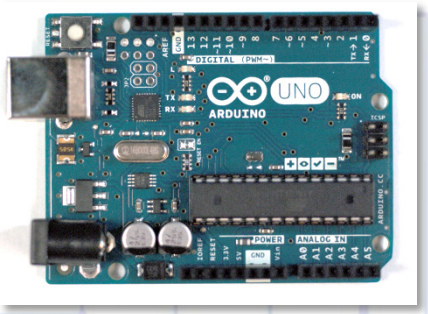
Introductory Medical Device Prototyping

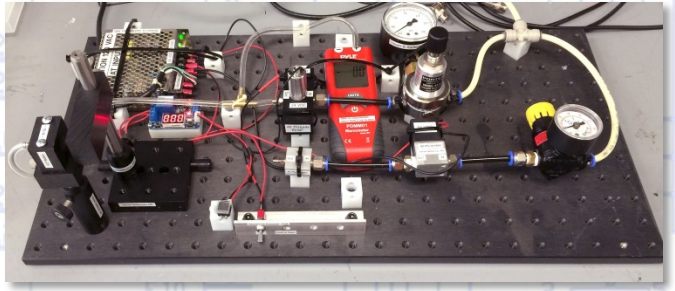
Arduino Part 2

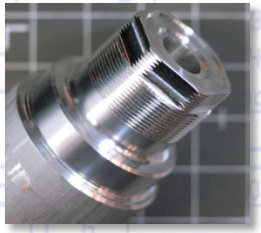
Prof. Steven S. Saliterman, http://saliterman.umn.edu/
Department of Biomedical Engineering, University of Minnesota







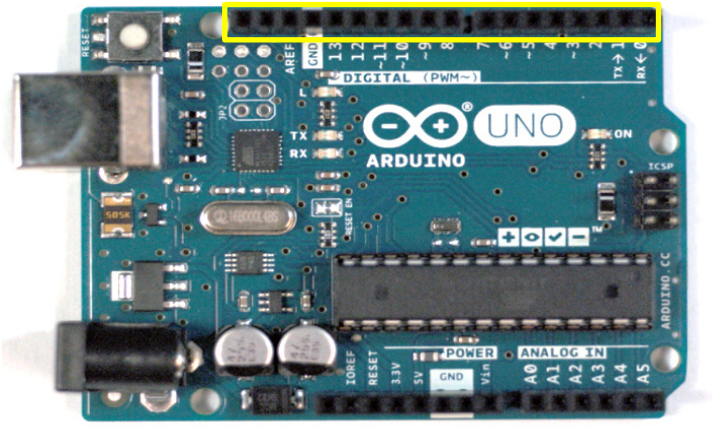




More Arduino Programming

- Digital I/O (Read/Write)
- Analog I/O
- Tones
- Delays

Digital I/O (Digital Pins)



- Digital pins on the Arduino can be defined as being Inputs or Outputs using the function pinMode().
- The state of a digital pin can be determined with the function digitalRead().
- The state of an output can be set as High or Low with the function digitalWrite().

digitalWrite()…

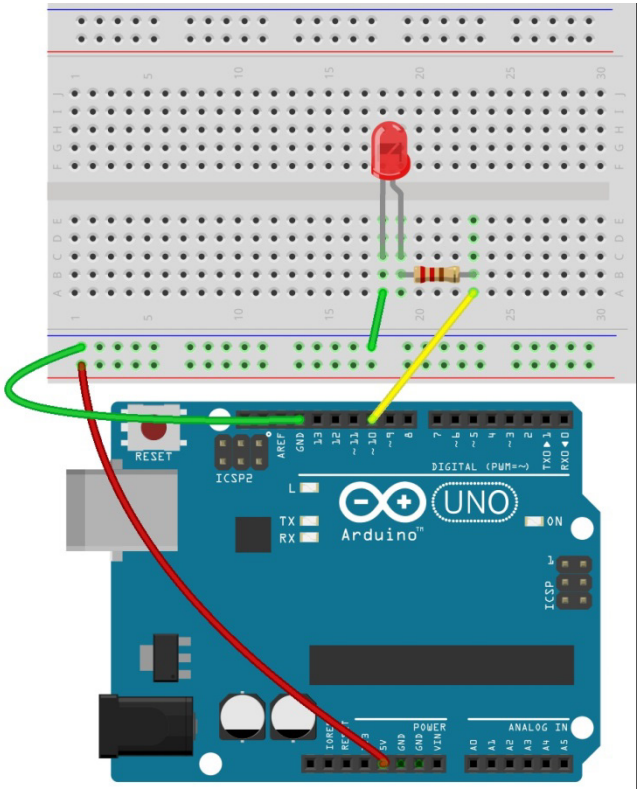
- digitalWrite() writes a HIGH or a LOW value to a digital pin.
- If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.
- If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) *the internal pullup resistor on the input pin*.
  - It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor. See the digital pins tutorial for more information.
- e.g. digitalWrite(ledPin, HIGH)

digitalRead()…

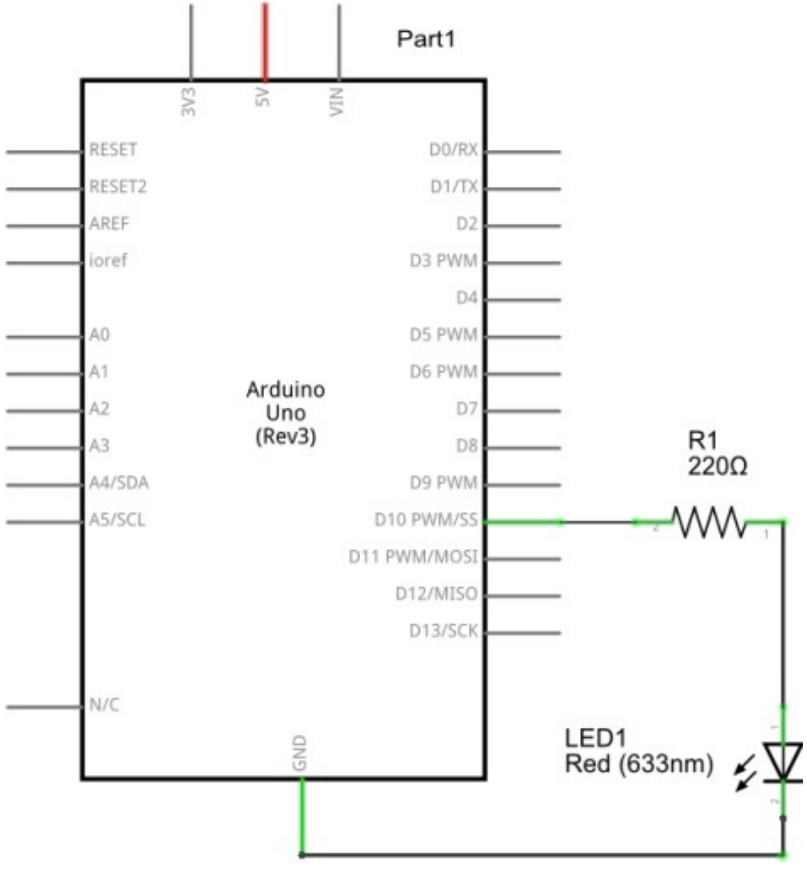
- digital read() reads the value from a specified digital pin, either HIGH or LOW.
- e.g. digitalRead(inPin);

Example: Blinking an LED

```
const int LED = 10;
int blinks = 5;                    // blink 5 times;
bool done = false;

void setup()
{
   pinMode(LED, OUTPUT);           //set pin 10  as an OUTPUT
   digitalWrite(LED, LOW);         // Initialize off
}

void loop()
{
  while (done != true)
   {
     for (int i = 1; i<= blinks; i++)      // i++ same as i = i+1
     {
        digitalWrite(LED, HIGH);           // Turn on LED
        delay(500);                        //Pause
        digitalWrite(LED, LOW);            // Turn off LED
        delay(500);                        //Pause
     }
     done = true;
   }
}
```



Schematic…

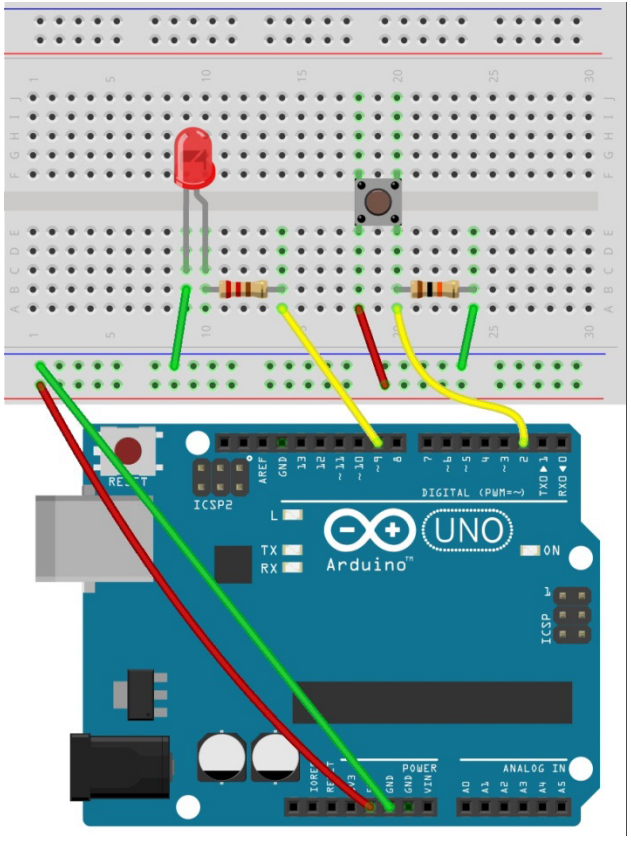

Part1
3V3
5V
VIN
RESET
RESET2
AREF
Ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
GND
Arduino
Uno
(Rev3)
R1
220Ω
LED1
Red (633nm)

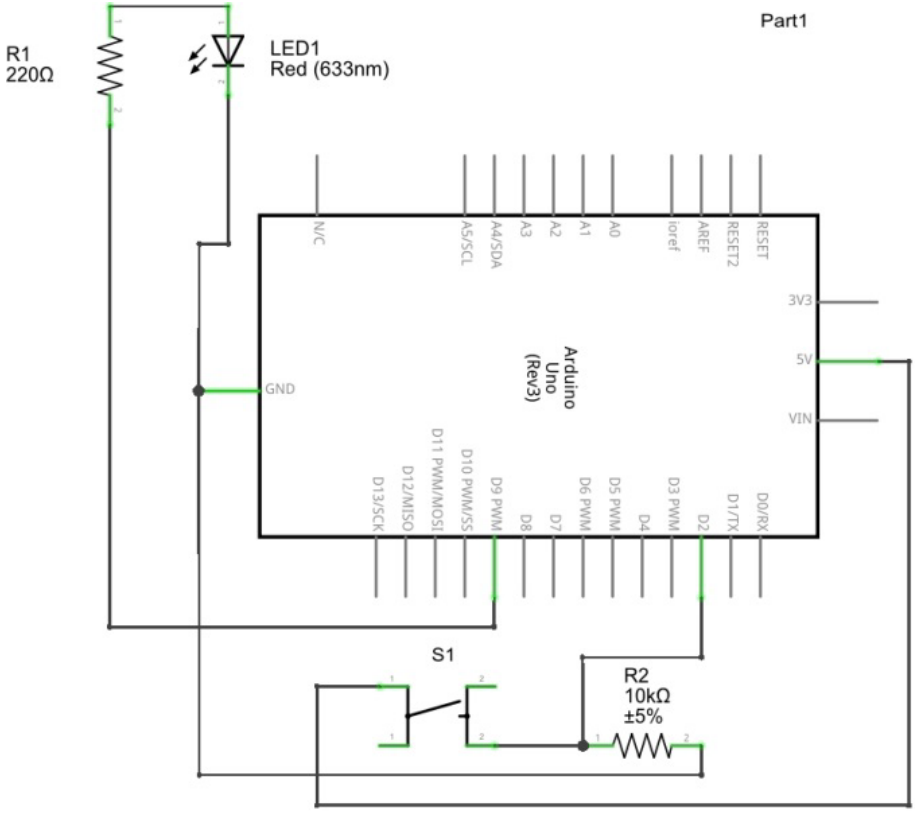
Example: Debouncing a Pushbutton

```
const int LED = 9, BUTTON = 2;
bool lastState = LOW, currentState = LOW, lit = false;

void setup()
{
  pinMode(LED, OUTPUT);
  pinMode(BUTTON, INPUT);
}

boolean debounce(boolean last) //function
{
  boolean state = digitalRead(BUTTON);
  if(last != state) // has button settled down
  {
    delay(5); //delay if not
    state= digitalRead(BUTTON); //and read again
  }
  return state;
}
```

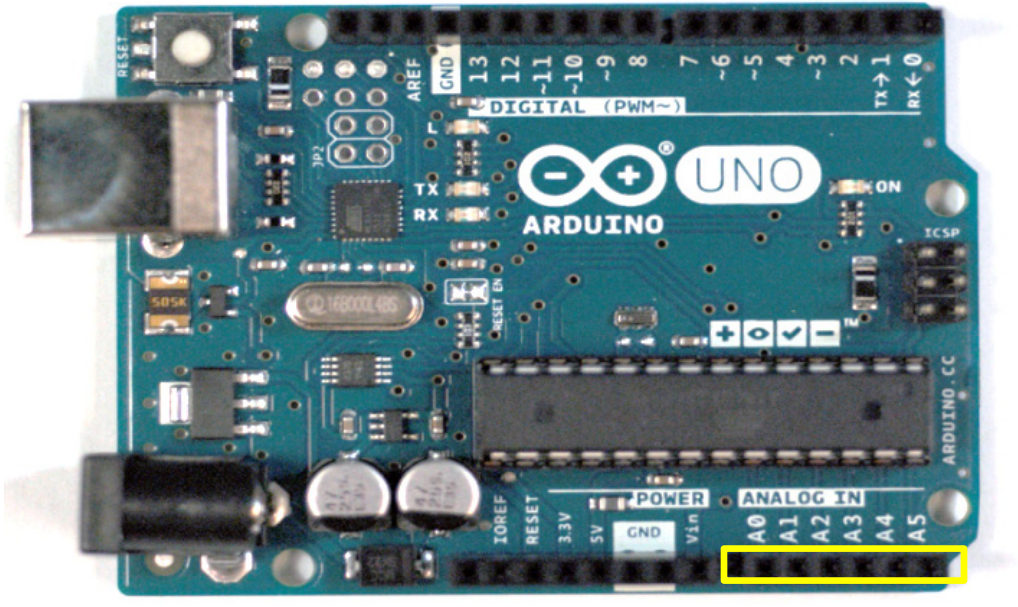
```
void loop()
{
   currentState = debounce(lastState); //call function
   if (lastState == LOW && currentState == HIGH)
   {
    lit = !lit;     //toggle LED
   }
   lastState = currentState;
   digitalWrite(LED, lit); //update LED
}
```

Debouncing a Pushbutton…





Analog I/O Pins



analogRead()…

- analogRead() reads the value from the specified analog pin.
- The Arduino board contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter.
  - This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023.
  - This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit.
- It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second

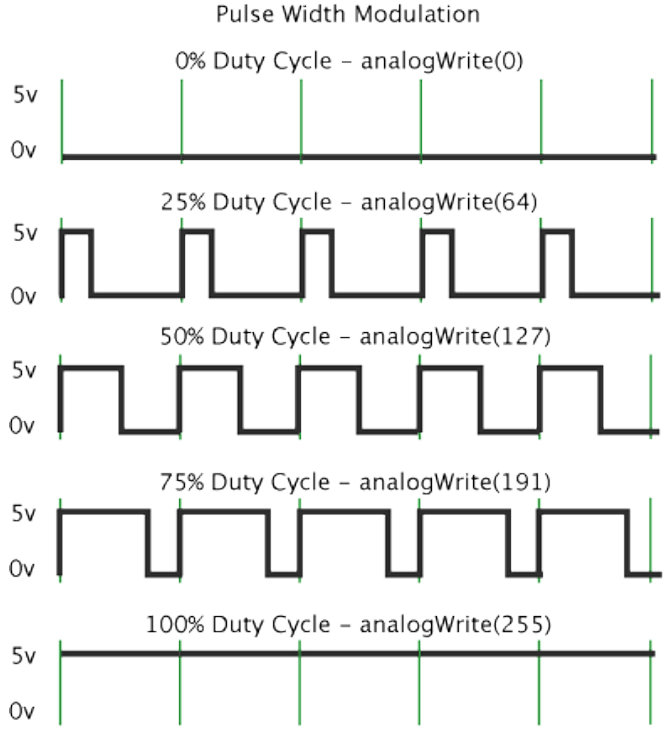
analogWrite()…

- analogWrite() writes an analog value as pulse width modulation (PWM) to a pin.
  - This is a technique for getting analog results with digital means.
  - Digital control is used to create a square wave (0 to 5 VDC).
  - You can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends “on” versus the time that the signal spends “off”.
  - The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5 VDC controlling the brightness of the LED.

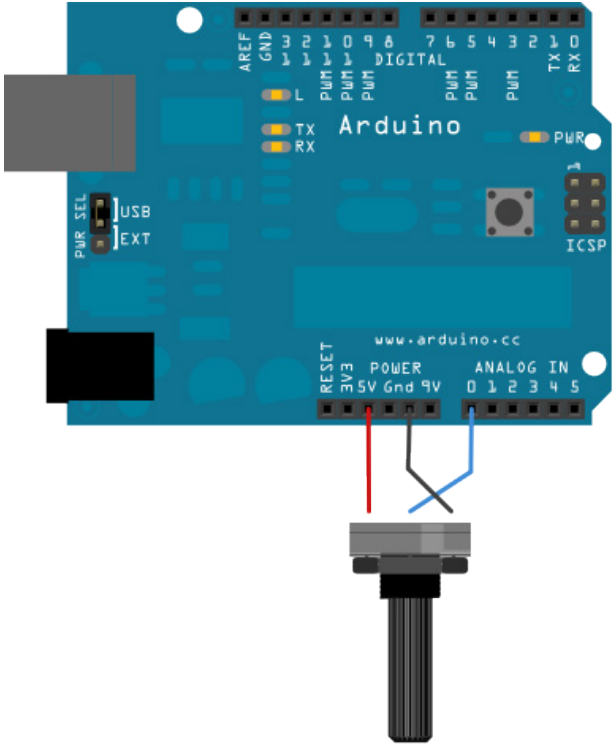
PWM…

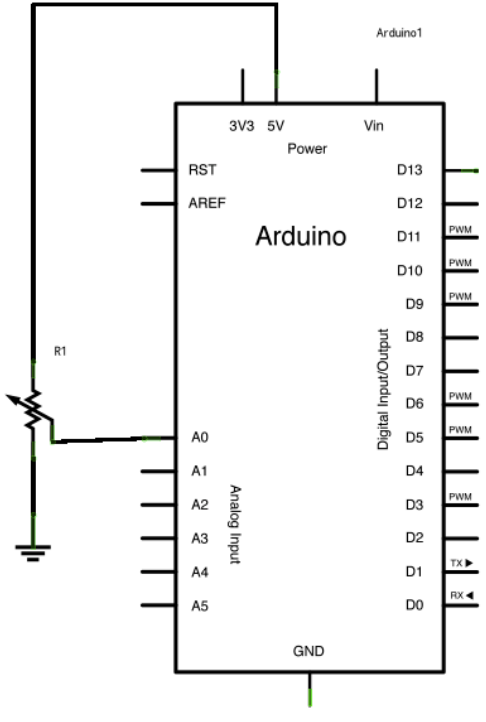
- Can be used to light a LED at varying brightnesses or drive a motor at various speeds.
  - After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite() on the same pin).
- The frequency of the PWM signal on most pins is approximately 490 Hz.
  - On the Uno and similar boards, pins 5 and 6 have a frequency of approximately 980 Hz.

PWM…


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

Reading a Potentiometer





Igoe, Tom. If statement tutorial. Arduino 2012

Reading a Potentiometer…

```
const int analogPin = A0;                    // pin that the potentiometer is attached to
const int ledPin = 13;                       // pin that the LED is attached to on UNO
const int threshold = 400;                   // an arbitrary threshold level that's in the range
                                                of the analog input

void setup()
{
   pinMode(ledPin, OUTPUT);                  // initialize the LED pin as an output
   Serial.begin(9600);                       // initialize serial communications
}

void loop()
{
  int analogValue = analogRead(analogPin);   // read the value of the potentiometer
  if (analogValue > threshold)               // if the analog value is high enough, turn on the LED
    {
      digitalWrite(ledPin, HIGH);
    }
else
    {
     digitalWrite(ledPin, LOW);
    }
 Serial.println(analogValue);                // print the analog value
 delay(1);                                   // delay in between reads for stability
}
```


Igoe, Tom. If statement tutorial. Arduino 2012

Analog Input Pins as Digital Pins…

- The analog pins can be used identically to the digital pins, using the aliases A0 (for analog input 0), A1, etc. For example, the code would look like this to set analog pin 0 to an output, and to set it HIGH:

```
pinMode(A0, OUTPUT);
digitalWrite(A0, HIGH);
```

- The analog pins also have pull-up resistors, which work identically to pull-up resistors on the digital pins. They are enabled by issuing a command such as:

```
pinMode(A0, INPUT_PULLUP);  // set pull-up on analog pin 0
```

Advanced I/O: tone()

- tone(*pin*, frequency) - Generates a square wave of the specified frequency (and 50% duty cycle) on a pin. A duration can be specified, otherwise the wave continues until a call to noTone(). The pin can be connected to a piezo buzzer or other speaker to play tones.
- Only one tone can be generated at a time. If a tone is already playing on a different pin, the call to tone() will have no effect. If the tone is playing on the same pin, the call will set its frequency.
- Use of the tone() function will interfere with PWM output on pins 3 and 11 (on boards other than the Mega).

Tone – Min/Max Frequencies…

Board	Min frequency (Hz)	Max frequency (Hz)
Uno, Mega, Leonardo and other AVR boards	31	65535
Gemma	Not implemented	Not implemented
Zero	41	275000
Due	Not implemented	Not implemented

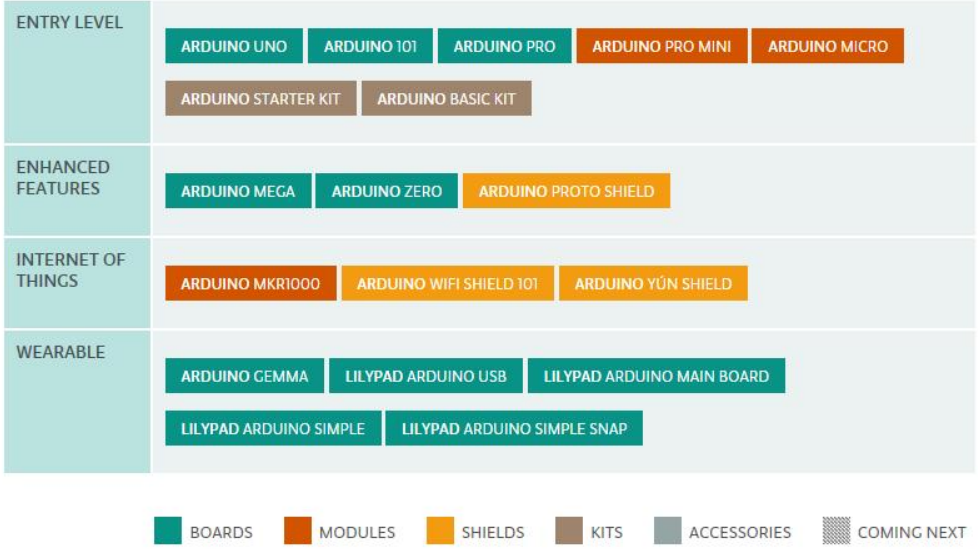
Time – millis() and micros()…

- millis() - returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.
- micros() - returns the number of microseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 70 minutes.
  - On 16 MHz Arduino boards, this function has a resolution of four microseconds (i.e. the value returned is always a multiple of four).
  - On 8 MHz Arduino boards (e.g. the LilyPad), this function has a resolution of eight microseconds.

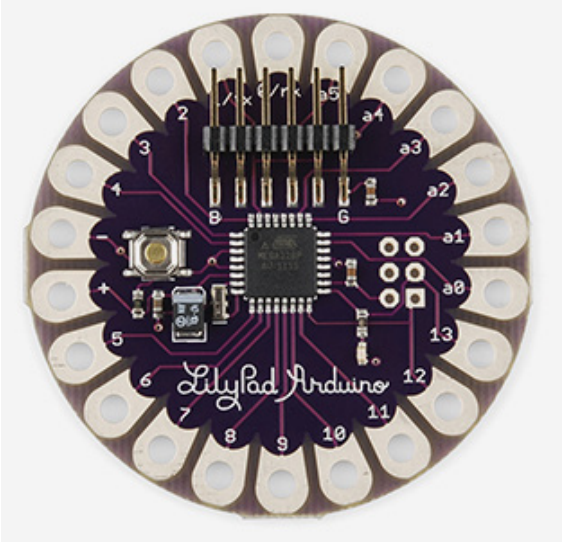
delay() and delayMicroseconds()…

- delay(*ms*) - Pauses the program for the amount of time (in milliseconds) specified as parameter. There are a thousand milliseconds in a second.
- delayMicroseconds() – Pauses in microseconds.
- Currently, the largest value that will produce an accurate delay is 16383. This could change in future Arduino releases. For delays longer than a few thousand microseconds, you should use delay() instead.

Other Arduino Boards



Other Arduino boards available.



e.g. LilyPad Arduino

Summary

- Functions:
  - Digital I/O (Read/Write)
    - Digital pins are defined as INPUT or OUTPUT and having levels of HIGH and LOW.
  - Analog I/O
  - Tones
  - Delays
- Examples reviewed:
  - Blinking an LED.
  - Debouncing a pushbutton.
  - Reading a potentiometer.