Breathing LED Code: Implementation on Arduino

ME 120

Mechanical and Materials Engineering

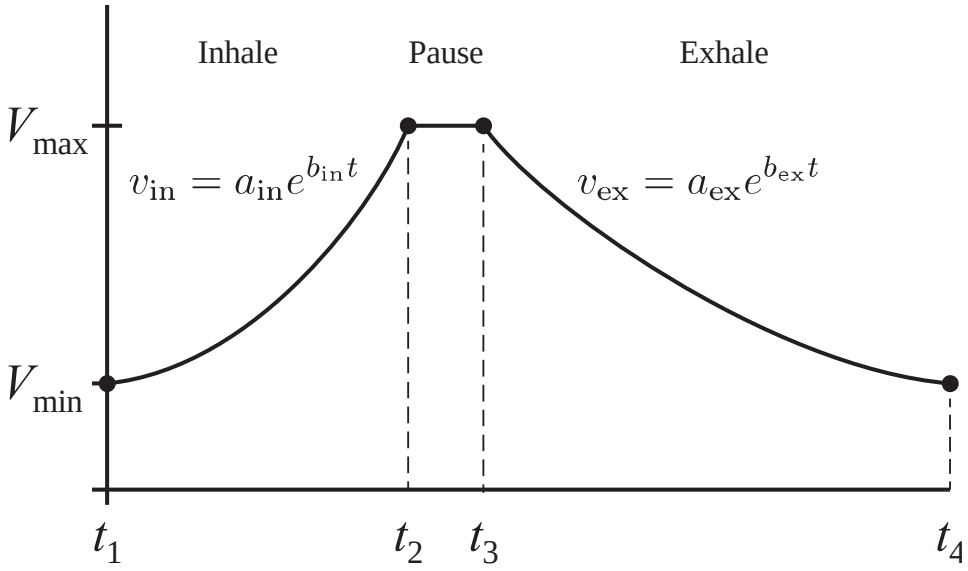
Portland State University

http://web.cecs.pdx.edu/~me120

Overview

- We have a mathematical model of brightness
- We can control LED brightness with PWM
- The next step is to develop the Arduino code

Review: Mathematical model

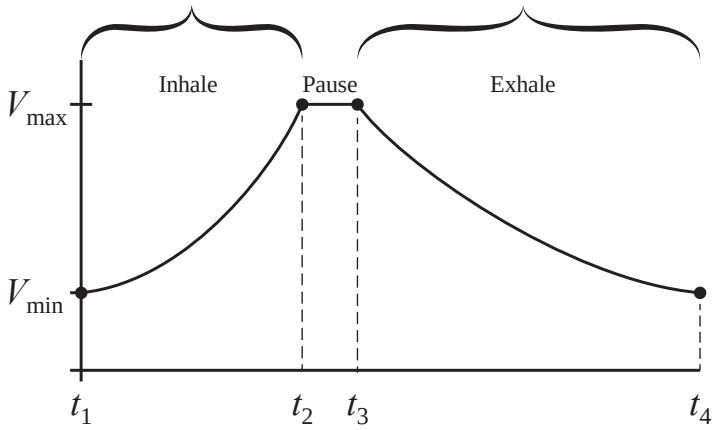


1. Choose $V_{\min}$ and $V_{\max}$
2. Choose t_2, t_3 and t_4 (t_1 is set to 0)
3. Use algebra to compute a_{in}, b_{in}, a_{ex} and b_{ex}
4. Formulas for $v_{\text{in}}(t)$ and $v_{\text{ex}}(t)$ are used to specify voltage to the LED

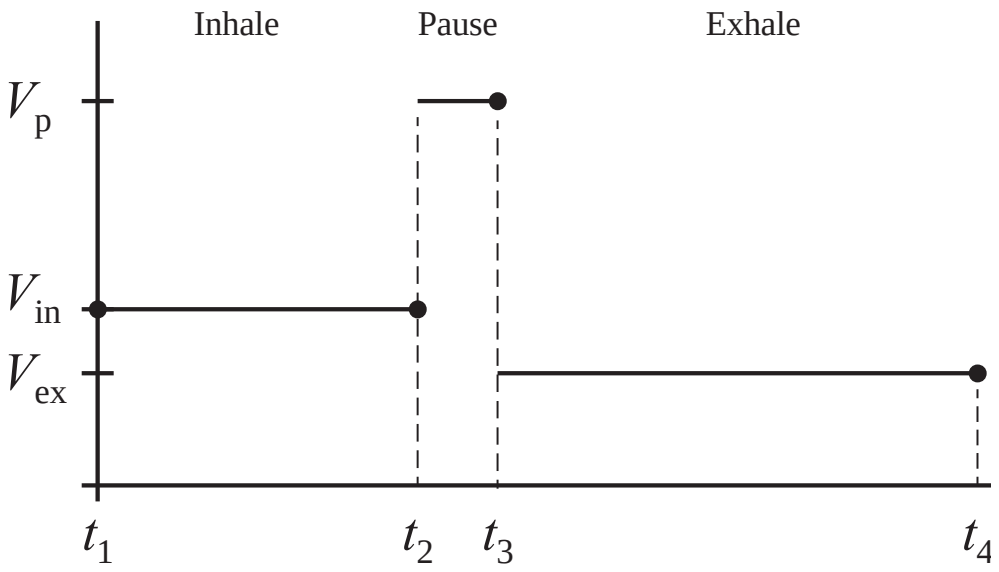
Implementation Preview

Code in the loop function creates the repeated pattern.

```
void loop() {

  // -- Inhale code

  // -- Pause code

  // -- Exhale code

}
```



A simplistic model with fixed delays



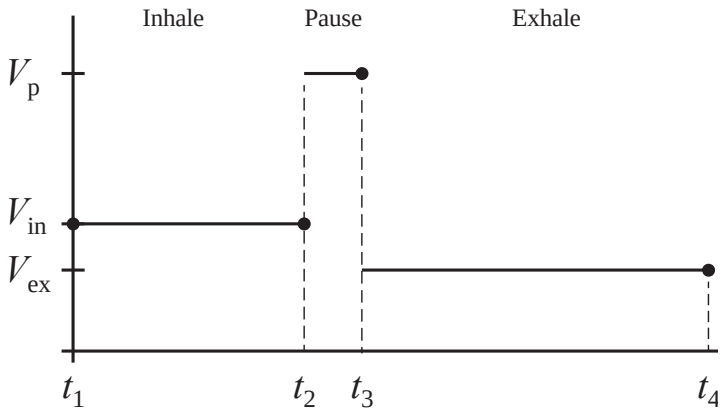

```
int LED_pin = 11;

void setup() {
  pinMode(LED_pin, OUTPUT);
}

void loop() {

  int vin=40, vpause=250, vex=20;

  analogWrite(LED_pin, vin);
  delay(2000);

  analogWrite(LED_pin, vpause);
  delay(500);

  analogWrite(LED_pin, vex);
  delay(2500);
}
```

- Constant levels do not simulate breathing
- Fixed delays do not allow PWM signal to vary with time

Working with the on-board timer

- One Arduino timer can be read with two commands
  - **`millis()`** returns the number of milliseconds
  - **`micros()`** returns the number of microseconds
- Values returned by **`millis()`** and **`micros()`** are relative to the last time the Arduino was restarted
- Values returned by **`millis()`** and **`micros()`** can be very large. Store the return values in **`unsigned long`** variables.

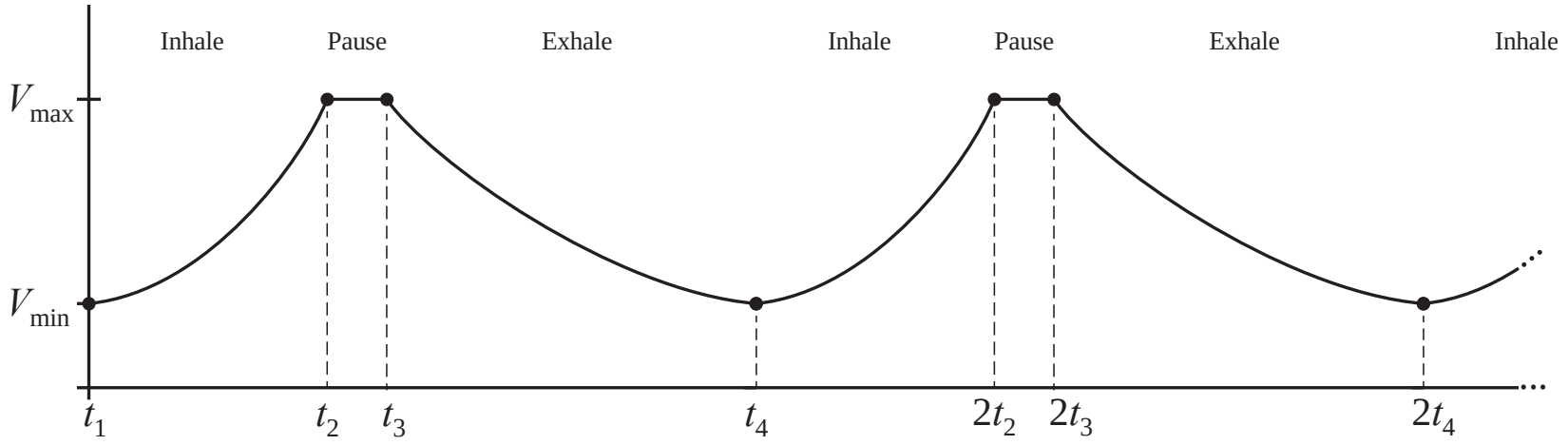
A simplistic model using the timer



- What happens when **`tm`** > **`t4`**?
- How can we account for the cyclic nature of breathing with a timer that increases continuously?

```
int LED_pin = 11;

void setup() {
  pinMode(LED_pin, OUTPUT);
}

void loop() {

  int v, vin=40, vpause=250, vex=20;
  unsigned long tm, t2=2000, t3=2500, t4=5000;

  tm = millis();

  if ( tm<=t2 ) {
    v = vin;
  } else if ( tm<=t3 ) {
    v = vpause;
  } else if ( tm<=t4 ) {
    v = vex;
  } else {
    v = 0;
  }
  analogWrite(LED_pin,v);

  Serial.print(tm);   Serial.print("  ");
  Serial.println(v);
}
```

The breathing pattern repeats



- The timer count returned by **`millis()`** increases indefinitely
- The breathing pattern repeats every t_4 milliseconds

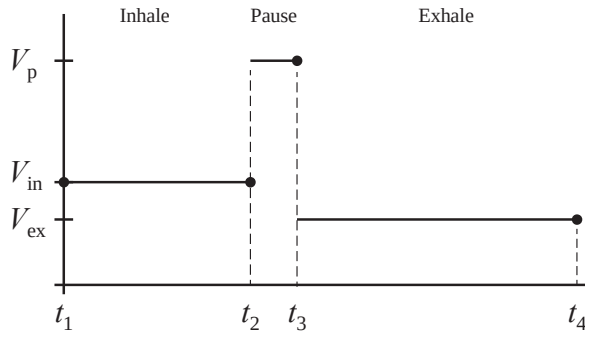
The modulo operator returns the remainder

- Arduino (C language) operator % is called modulo
- **`i%j`** returns the remainder of dividing **`i`** by **`j`**
- Examples:

`i`	**`j`**	**`i % j`**
2	2	0
5	2	1
10	6	4

- If **`t4`** is the time to complete one breathing cycle, then **`millis()%t4`** is the time (in milliseconds) in the current cycle

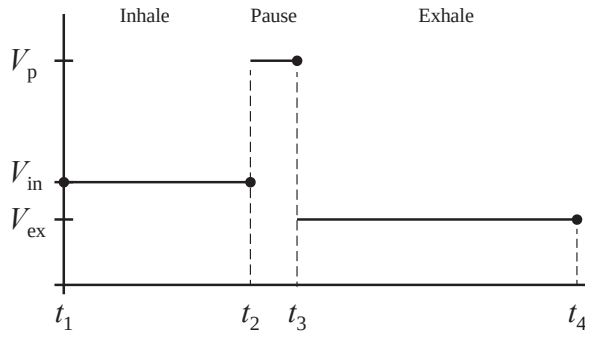
Use % to extract cycle time from **`millis()`**



- Pattern repeats because **`tm%t4`** is time from start of current cycle
- A subtle problem remains because **`millis()`** may not start at zero

```
int LED_pin = 11;

void setup() {
  pinMode(LED_pin, OUTPUT);
}

void loop() {

  int v, vin=40, vpause=250, vex=20;
  unsigned long t, tm;
  unsigned long t2=2000, t3=2500, t4=5000;

  tm = millis();   // Current timer
  t = tm % t4;     // Time from start of current cycle

  if ( t<=t2 ) {
    v = vin;
  } else if ( t<=t3 ) {
    v = vpause;
  } else if ( t<=t4 ) {
    v = vex;
  } else {
    v = 0;
  }
  analogWrite(LED_pin,v);

  Serial.print(tm);   Serial.print("  ");
  Serial.print(t);    Serial.print("  ");
  Serial.println(v);
}
```

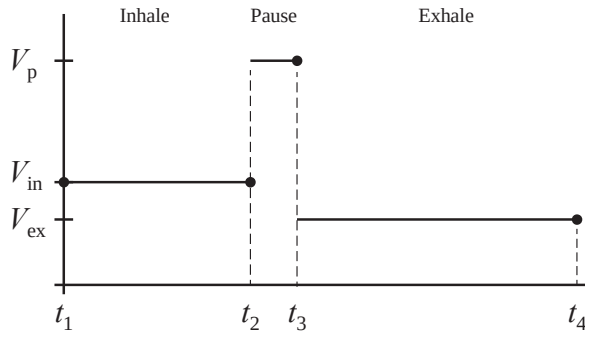
Correct for non-zero `millis()` at start-up



- Subtracting tstart from current **`millis()`** reading gives the timer value measured from the last reset or power-up

```
int LED_pin = 11;
unsigned long tstart;          // Global variable

void setup() {
  pinMode(LED_pin, OUTPUT);
  tstart = millis();           // Time at start of program
}

void loop() {

  int v, vin=40, vpause=250, vex=20;
  unsigned long t, tm;
  unsigned long t2=2000, t3=2500, t4=5000;

  tm = millis() - tstart;      // Count from program start
  t = tm % t4;                 // Time in current cycle

  if ( t<=t2 ) {
    v = vin;
  } else if ( t<=t3 ) {
    v = vpause;
  } else if ( t<=t4 ) {
    v = vex;
  } else {
    v = 0;
  }
  analogWrite(LED_pin,v);

  Serial.print(tm);    Serial.print("  ");
  Serial.print(t);     Serial.print("  ");
  Serial.println(v);
}
```

Convert to time in seconds



- It will be easier to evaluate the breathing formulas in floating point arithmetic than using **`long int`**s.
- Convert **`t`**, **`t2`**, **`t3`** and **`t4`** to time in seconds

```
int LED_pin = 11;
unsigned long tstart;          // Global variable

void setup() {
  pinMode(LED_pin, OUTPUT);
  tstart = millis();           // Time at start of program
}

void loop() {

  int v, vin=40, vpause=250, vex=20;
  unsigned long tm;
  float t, t2=2.0, t3=2.5, t4=5.0;

  tm = millis() - tstart;      // Count from program start
  t = tm % long(t4 * 1000.0);// Time in current cycle (ms)
  t = t/1000.0;                // Time in current cycle (s)

  if ( t<=t2 ) {
    v = vin;
  } else if ( t<=t3 ) {
    v = vpause;
  } else if ( t<=t4 ) {
    v = vex;
  } else {
    v = 0;
  }
  analogWrite(LED_pin,v);

  Serial.print(tm);    Serial.print("  ");
  Serial.print(t);     Serial.print("  ");
  Serial.println(v);
}
```

Finishing touches

- Modify the code given in these slides
  - Add variables and code to evaluate the two exponential functions: one for inhale, one for exhale
  - Compute coefficients a_{in}, b_{in}, a_{ex} and b_{ex} and update code
  - Make sure **`t2`**, **`t3`**, and **`t4`** in the code are consistent with the model equations for the exponentials (Consistency check)
- Experiment with the system and adjust voltage levels to get the desired brightness
- Experiment with the system and adjust time markers **`t2`**, **`t3`**, **`t4`** to get desired breathing speed
- Any change in voltage level or time markers will require recalculation of a_{in}, b_{in}, a_{ex} and b_{ex}