Modeling Physical Systems

with Arduino and C

Kevin D'Angelo

The Pendulum Wave Model

YouTube Pendulum Wave:

https://www.youtube.com/watch?v=yVkdfJ9PkRQ

Here's one with a black light:

https://www.youtube.com/watch?v=7_AiV12XBbI

Arduino LightBox Modeling Task:

Model the pendulum wave using an LED for each pendulum

LED is ON for the first half cycle

LED is OFF for the second half cycle

Pendulum Wave

Uses a simple relationship between periods of pendulums such that they synchronize after an integer number of cycles.

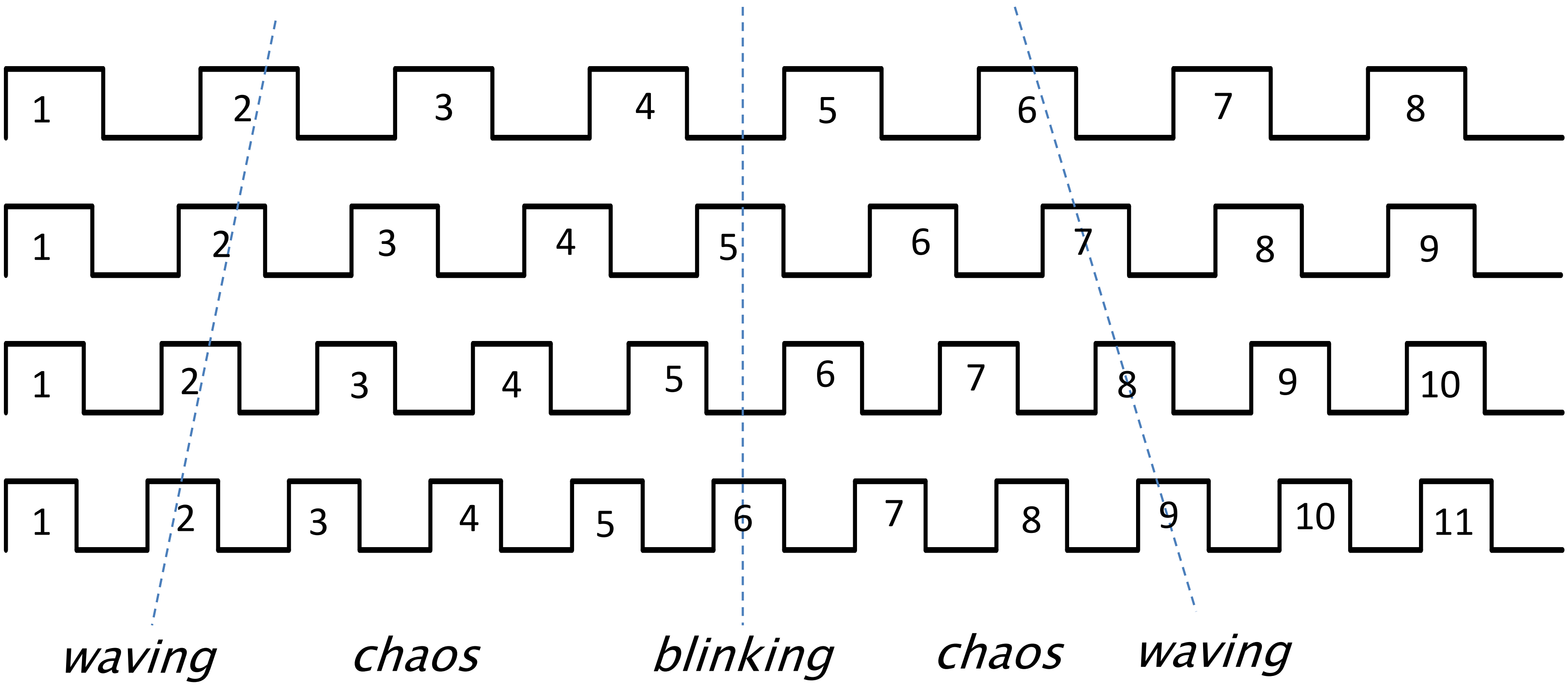
"For S cycles of a given pendulum, the adjacent pendulums have (S+1) cycles and (S−1) cycles"

or for adjacent periods T_1 and T_2

$$ST_1 = (S+1)T_2$$

Pendulum Wave

Paper model example: S=8



Calculate the Periods

Base Period $= T$

Number of Cycles between syncs $= S$

Generalized for the nth pendulum of period Tn,

$$(S+n)Tn = ST$$

or

$$Tn = \frac{S}{S+n}T$$

Get the Arduino to Calculate Periods

$$\textit{Remember}\ Tn = \frac{S}{S+n} T$$

Coding for 8 elements to fill array T[n]:

```
for (n=0; n<8; n++) {
        T[n] = S/(S+n) * T[0];
}
```

Keeping Track of Time

the `micros()` instruction returns the time in μs.

e.g. for a single blinking LED

```
t_stamp = micros();  //store time in t_stamp var
                     // t_stamp is a long var
void loop() {
  time = micros();
  if (time - t_stamp >= T){  //time is up
    t_stamp = micros();
    digitalWrite(LED[0], !digitalRead(LED[0]));
  }
}
```

Experimentation

What parameters can we change?

number of cycles for a sync

longest period

…

Improve It (advanced)

Model the pendulum elements using an LED for each pendulum.

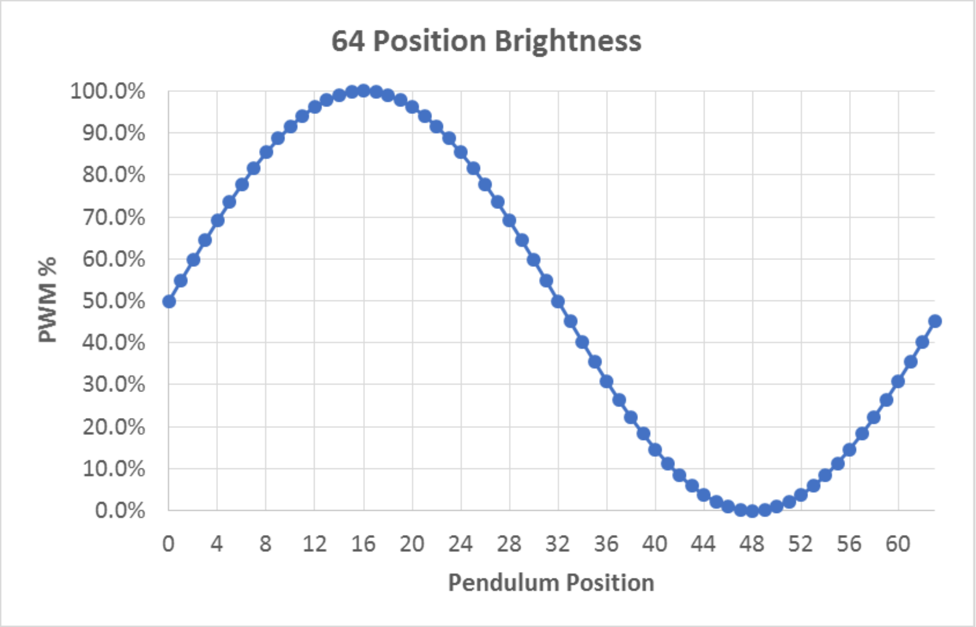
Let LED brightness be a function of its position in the period.

Use discrete positions.

1. Calculate pendulum periods (as before)
2. Divide the period into e.g. 64 positions
3. Determine a PWM duty cycle for each of those 64 positions
4. Update the PWM duty cycle every T/64 seconds

PWM Duty Cycles

Since periodic motion is sinusoidal, use a sine wave look up table for PWM duty cycle:




```
// populate PWM duty cycle look up table //
// amplitude set to half duty cycle      //
// offset set to half duty cycle         //
for (i=0; i<64; i++) {
  sineTable[i] = Tpwm/2*sin(2.0*pi*i/64)+Tpwm/2;
}
```

Manual PWM'ing

A good rule is to use a Period that is a power of 2 to make the math more efficient,

e.g. `Tpwm` $= 2^{13} = 8192\mu s$, or 122 Hz

<u>To PWM the LEDs</u>

Set the LED high at the beginning of the period and get the time stamp.

Set the LED low, when the duty cycle is reached.

<u>Make an array to hold the PWM values of each individual LED</u>

```
Unsigned long t_PWM[8];       // initialize to 0
```

Some Useful Variable Declarations

```
long Tbase = 1500000;             // longest pendulum period
long sync=16.0;                   // number of cycles between syncs
unsigned long time;               // variable for present time
unsigned long t_PWM[8];           // time stamp for PWM
unsigned long t_T[8];             // time stamp for pendulum periods
unsigned long T[8];               // pendulum periods
int pwm[8];                       // array for each LEDs pwm duty cycle
int sineTable[64];                // look up table of sine values
int sinVal[8];                    // index for sine table look up <0:63>
long Tpwm=8192;                   // pwm period: 8192 us
float pi=3.14159;                 // pi
int LED[8]={4,5,6,7,8,9,10,11}; // LED ports array
int n;                            // general purpose index variable
```

Adjust Periods

Consider each position in discrete time

Break up each period into 64 discrete positions from sine table

Calculate 1/64 of a period for each pendulum:

```
for (n=0; n<8; n++) {
  T[n] = S/(S+n) * T[0] / 64.0;
}
```

Then use this 1/64 period as the time base to index though the brightness lookup table