

Digital Hot Wheels 64 Scale Finish Line - Remix



Alex Oliva

VIEW IN BROWSER

updated 7. 8. 2024 | published 7. 8. 2024

Summary

Hot Wheels Lap Counter with Optical

[Toys & Games](#) > [Other Toys & Games](#)

Tags: [thingiverse](#)

NOTE - sensitivity is adjusted with TriggerThresholdLane values
Make sure you keep all serial commented out for use. This effects
response time.

Works with USB (to customize Arduino Nano and power) or 9v battery
Start button starts countdown timer for manual starting gate
Final shows place of finish and elapsed time in seconds.

Video: <https://youtu.be/Gd426bAjsRk>

Arduino Nano v3

4 x GA1A12S202 Log-scale Analog Light Sensor

4 x 0.36"

4-Digit Tube LED Segment Display Module TM1637 Drive Chip

4 x LEDs

Arduino code is below

NOTE - this uses AREF for 3v so be sure to follow instructions in comment so you don't fry your Nano :)

I used supports only on the piece that holds LEDs

You will need to make a battery holder for the 9v

You will need to cut hole for USB plug

Print Settings

Printer:

Qidi Tech 1 and 10S Pro V2

Rafts:

No

Supports:

Doesn't Matter

Resolution:

,2

Infill:

20%

Filament: Hatchbox PLA Orange

Notes:

ARDUINO CODE

```
// include the SevenSegmentTM1637 library
// https://github.com/bremme/arduino-tm1637
#include "SevenSegmentTM1637.h"
// #include "SevenSegmentExtended.h"
// #include "SevenSegmentFun.h"

/*
introDuceNextDemo("SCROLLING TEXT DEMO");
byte repeats = 2;
display.scrollingText("ARDUINO TM1637 FUN", repeats);
*/
```

```
/*
```

Sensor can be powered by 2.3-6V, and has an analog output of up to 3v

max.

To use with an Arduino, wire it as follows:

VCC -> 5v

OUT -> A0

GND -> GND

Connect 3.3v to the AREF pin

Since output range of the sensor is 0-3v, for maximum resolution, it is best to

use the 3.3v pin for your voltage reference. To do this, run a jumper from 3.3v to AREF on your Arduino.

In your code, be sure to specify "analogReference(EXTERNAL);" before performing any analogRead() operations.

Otherwise, your external reference will be shorted to the default internal 5v reference.

This can damage your Arduino.

*/

```
int sensorPinLane1=A0;
```

```
int sensorPinLane2=A1;
```

```
int sensorPinLane3=A2;
```

```
int sensorPinLane4=A3;
```

```
int sensorPinReadingLane1;
```

```
int sensorPinReadingLane2;
```

```
int sensorPinReadingLane3;
```

```
int sensorPinReadingLane4;
```

```
float TriggerValueLane1;
```

```
float TriggerValueLane2;
```

```
float TriggerValueLane3;
```

```
float TriggerValueLane4;
```

```
float rawRange = 1024; // 3.3v
```

```
float logRange = 5.0; // 3.3v = 10^5 lux
```

```
const float TriggerThresholdLane1 = .72; // % of light
```

```
const float TriggerThresholdLane2 = .72;
```

```
const float TriggerThresholdLane3 = .72;
```

```
const float TriggerThresholdLane4 = .72;
```

```
// Button pins
```

```
const int startButtonPin = 11;
```

```
const int resetButtonPin = 12;
```

```
/* initialize global TM1637 Display object
```

```
* The constructor takes two arguments, the number of the clock pin and
```

the digital output pin:

```
* SevenSegmentTM1637(byte pinCLK, byte pinDIO);  
*/
```

```
// Module connection pins (Digital Pins)
```

```
#define CLK 2
```

```
#define DIO1 3
```

```
// Note that even though we used another connection for CLK, you may be  
able to use the same CLK for all
```

```
// You would not define any other CLK - later you would just use the sole  
CLK parameter
```

```
// TM1637Display display2(CLK, DIO2);
```

```
#define DIO2 4
```

```
#define DIO3 5
```

```
#define DIO4 6
```

```
SevenSegmentTM1637 display1(CLK, DIO1);
```

```
SevenSegmentTM1637 display2(CLK, DIO2);
```

```
SevenSegmentTM1637 display3(CLK, DIO3);
```

```
SevenSegmentTM1637 display4(CLK, DIO4);
```

```
//SevenSegmentFun display1(CLK, DIO1);
```

```
//SevenSegmentFun display2(CLK, DIO2);
```

```
//SevenSegmentFun display3(CLK, DIO3);
```

```
//SevenSegmentFun display4(CLK, DIO4);
```

```
int startButtonState = 0;
```

```
int resetButtonState = 0;
```

```
int raceStatus = 0;
```

```
int lane1Status = 0;
```

```
int lane2Status = 0;
```

```
int lane3Status = 0;
```

```
int lane4Status = 0;
```

```
int currentPlace = 0;
```

```
float currentMillis;
```

```
float startMillis;
```

```
float elapsedMillis;
```

```
float elapsedDisplayMillis;
```

```
float displayMillis;
```

```
byte resetDisplayFlag;
```

```
float elapsedSecs1;
```

```
float elapsedSecs2;
```

```

float elapsedSecs3;
float elapsedSecs4;

String dispPlace1;
String dispPlace2;
String dispPlace3;
String dispPlace4;

#define ARRAYSIZE 4
String placeArray[ARRAYSIZE] = { "1st ", "2nd ", "3rd ", "4th " };
char floatString[4];
char scrollBuffer[50];

String strElapsedSecs1;
String strElapsedSecs2;
String strElapsedSecs3;
String strElapsedSecs4;

void setup()
{
  pinMode(startButtonPin, INPUT_PULLUP);
  pinMode(resetButtonPin, INPUT_PULLUP);

  //Serial.begin(9600);
  // This is IMPORTANT - READ INITIAL COMMENTS!
  analogReference(EXTERNAL);

  display1.begin(); // initializes the display - might be display.init()
  display1.setBacklight(100); // set the brightness to 100 %
  display2.begin();
  display2.setBacklight(100);
  display3.begin();
  display3.setBacklight(100);
  display4.begin();
  display4.setBacklight(100);

  delay(1000); // wait 1000 ms

  display1.print("Ln01");
  display2.print("Ln02");
  display3.print("Ln03");
  display4.print("Ln04");
}

void loop()
{
  sensorPinReadingLane1 = analogRead(sensorPinLane1);
  sensorPinReadingLane2 = analogRead(sensorPinLane2);

```

```

sensorPinReadingLane3 = analogRead(sensorPinLane3);
sensorPinReadingLane4 = analogRead(sensorPinLane4);

resetButtonState = digitalRead(resetButtonPin);
startButtonState = digitalRead(startButtonPin);

// For testing - use serial monitor to see what's happening
//Serial.print("Lane1: ");
//Serial.print(sensorPinReadingLane1);
//Serial.print(" - Lux = ");
//Serial.print(RawToLux(sensorPinReadingLane1));
//Serial.print(" - Trig = ");
//Serial.println(TriiggerValueLane1);

//Serial.print("Lane2: ");
//Serial.print(sensorPinReadingLane2);
//Serial.print(" - Lux = ");
//Serial.print(RawToLux(sensorPinReadingLane2));
//Serial.print(" - Trig = ");
//Serial.println(TriiggerValueLane2);

//Serial.print("Lane3: ");
//Serial.print(sensorPinReadingLane3);
//Serial.print(" - Lux = ");
//Serial.print(RawToLux(sensorPinReadingLane3));
//Serial.print(" - Trig = ");
//Serial.println(TriiggerValueLane3);

//Serial.print("Lane4: ");
//Serial.print(sensorPinReadingLane4);
//Serial.print(" - Lux = ");
//Serial.print(RawToLux(sensorPinReadingLane4));
//Serial.print(" - Trig = ");
//Serial.println(TriiggerValueLane4);

//Serial.print("startButtonState: ");
//Serial.print(startButtonState);
//Serial.print(" resetButtonState: ");
//Serial.println(resetButtonState);
if(resetButtonState == LOW) {

delay(1000); // wait 1000 ms
display1.print("Ln01");
display2.print("Ln02");
display3.print("Ln03");
display4.print("Ln04");
lane1Status = 0;

```

```

lane2Status = 0;
lane3Status = 0;
lane4Status = 0;
currentPlace = 0;
raceStatus = 0;
}

if (startButtonState == LOW) {
// start the race raceStatus == 0 &&

display1.print("----");
display2.print("----");
display3.print("----");
display4.print("----");
delay(2000);
display1.clear();display1.print(" 4");
delay(1000);
display1.clear();display2.print(" 3");
delay(1000);
display2.clear();display3.print(" 2");
delay(1000);
display3.clear();display4.print(" 1");
delay(1000);
display1.print("----");
display2.print("----");
display3.print("----");
display4.print("----");

TriggerValueLane1 = sensorPinReadingLane1 * TriggerThresholdLane1;
TriggerValueLane2 = sensorPinReadingLane2 * TriggerThresholdLane2;
TriggerValueLane3 = sensorPinReadingLane3 * TriggerThresholdLane3;
TriggerValueLane4 = sensorPinReadingLane4 * TriggerThresholdLane4;
startMillis = millis();

lane1Status = 0;
lane2Status = 0;
lane3Status = 0;
lane4Status = 0;
currentPlace = 0;

//gateServo.write(openGateServoPosition);
// wait for servo to drop
//delay(150);
raceStatus = 1;
}
if (raceStatus == 1) {
// race is running

```

```

currentMillis = millis();
elapsedMillis = (currentMillis - startMillis);
elapsedDisplayMillis = currentMillis - displayMillis;
if (elapsedDisplayMillis > 10000) {
if (lane1Status > 0) {display1.clear();display1.print(strElapsedSecs1);}
if (lane2Status > 0) {display2.clear();display2.print(strElapsedSecs2);}
if (lane3Status > 0) {display3.clear();display3.print(strElapsedSecs3);}
if (lane4Status > 0) {display4.clear();display4.print(strElapsedSecs4);}
displayMillis = currentMillis;
resetDisplayFlag = 0;
}

else if (elapsedDisplayMillis > 5000 && resetDisplayFlag == 0) {
if (lane1Status > 0) {display1.clear();display1.print(disPlace1);}
if (lane2Status > 0) {display2.clear();display2.print(disPlace2);}
if (lane3Status > 0) {display3.clear();display3.print(disPlace3);}
if (lane4Status > 0) {display4.clear();display4.print(disPlace4);}
resetDisplayFlag = 1;
}

if (sensorPinReadingLane1 < TriggerValueLane1 && lane1Status == 0) {
currentPlace++;
elapsedSecs1 = elapsedMillis / 1000;
if (elapsedSecs1 < 10) {
strElapsedSecs1 = dtostrf(elapsedSecs1,2,2,floatString);
} else {
strElapsedSecs1 = dtostrf(elapsedSecs1,1,1,floatString);
}
disPlace1 = placeArray[currentPlace - 1]; // arrays start with 0 position

// For testing - use serial monitor to see what's happening
//Serial.print("Lane 1: ");
//Serial.print(disPlace1);
//Serial.print(" [");
//Serial.print(elapsedSecs1);
//Serial.println("]");
display1.print(disPlace1);
lane1Status = currentPlace;
}

if (sensorPinReadingLane2 < TriggerValueLane2 && lane2Status == 0) {
currentPlace++;
elapsedSecs2 = elapsedMillis / 1000;
if (elapsedSecs2 < 10) {
strElapsedSecs2 = dtostrf(elapsedSecs2,2,2,floatString);
} else {
strElapsedSecs2 = dtostrf(elapsedSecs2,1,1,floatString);
}
}

```



```

}
dispPlace2 = placeArray[currentPlace - 1]; // arrays start with 0 position

// For testing - use serial monitor to see what's happening
//Serial.print("Lane 2: ");
//Serial.print(dispPlace2);
//Serial.print(" ");
//Serial.print(elapsedSecs2);
//Serial.println("]");
display2.print(dispPlace2);
lane2Status = currentPlace;
}

if (sensorPinReadingLane3 < TriggerValueLane3 && lane3Status == 0) {
currentPlace++;
elapsedSecs3 = elapsedMillis / 1000;
if (elapsedSecs3 < 10) {
strElapsedSecs3 = dtostrf(elapsedSecs3,2,2,floatString);
} else {
strElapsedSecs3 = dtostrf(elapsedSecs3,1,1,floatString);
}
dispPlace3 = placeArray[currentPlace - 1]; // arrays start with 0 position

// For testing - use serial monitor to see what's happening
//Serial.print("Lane 3: ");
//Serial.print(dispPlace3);
//Serial.print(" ");
//Serial.print(elapsedSecs3);
//Serial.println("]");
display3.print(dispPlace3);
lane3Status = currentPlace;
}

if (sensorPinReadingLane4 < TriggerValueLane4 && lane4Status == 0) {
currentPlace++;
elapsedSecs4 = elapsedMillis / 1000;
if (elapsedSecs4 < 10) {
strElapsedSecs4 = dtostrf(elapsedSecs4,2,2,floatString);
} else {
strElapsedSecs4 = dtostrf(elapsedSecs4,1,1,floatString);
}
dispPlace4 = placeArray[currentPlace - 1]; // arrays start with 0 position

// For testing - use serial monitor to see what's happening
//Serial.print("Lane 4: ");
//Serial.print(dispPlace4);
//Serial.print(" ");
//Serial.print(elapsedSecs4);

```

```
//Serial.print(":");
//Serial.print(strElapsedSecs4);
//Serial.println("]");
display4.print(disPlace4);
lane4Status = currentPlace;
}

delay(10); // This will make the display update at 100Hz.*/
}
}

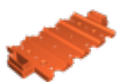
float RawToLux(int raw)
{
float logLux = raw * logRange / rawRange;
return pow(10, logLux);
}
```

Model files



hotwheelsfinishlinebottomassembly.stl

hot_wheel_finish_line_components_complete_v8.f3d



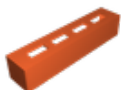
track.stl



hot_wheels_track_joiner.stl



column.stl



hotwheelsfinishlinemain.stl

License ©

This work is licensed under a
Creative Commons (4.0 International License)



Attribution-NonCommercial

- ✗ | Sharing without ATTRIBUTION
- ✓ | Remix Culture allowed
- ✗ | Commercial Use
- ✗ | Free Cultural Works
- ✗ | Meets Open Definition