```
/*
   This "SPEEDOMETER" sketch was written by Dave Ackmann (ackmanns@charter.net) in October, 2018.
   Its purpose is to time a train as it transits a loop, and then report the average
   speed for the transit.  It is unique in that it uses only a single sensor (not two)
   and uses the entire length of a loop, (rather than a short length of track) to generate
   a more accurate speed reading.

   The sketch is written for an Arduino Nano (AMAZON ASIN: B07G99NNXL),
   a 4 line I2C display (AMAZON ASIN: B01GPUMP9C), a 10K fixed resistor, and a light
   dependent resistor (AMAZON ASIN: B01N7V536K)(aka: LDR, photo resistor, opto resistor,
   photo cell; it has many names); jumpers, shrink wrap, a barrier strip and battery or 12V
   power supply.  When using a Nano, I like to plug it in to an expansion board (AMAZON
   ASIN: B07MGVC18K) because no soldering is required, it uses tight fitting screws to
   hold the wires.

   Wiring can be complex, but using specific colored wired for specific purposes can make
   things easier.  I like to use black wires for things connected to Ground (GND).  I use red
   wires for things connected to positive voltage from a power supply, a voltage regulator
   (this project does not require a voltage regulator), or from the 5V pin of the Arduino;
   (some times I also use white wires for this purpose).  Other than these purposes, I will use
   any wire I have on hand.  I usually use 20 to 24 gauge wire.

   Four places in this circuit have multiple connections: Voltage INput, Ground, the Nano's A0
   socket, and 5V.  Since we can't place multiple wires in one socket, I like to use a
   barrier strip with four connection points.  Let's label the points VIN, GND, A0, and 5V,
   with a piece of masking tape.  Run wires from the Arduino's VIN, 5V, GND and A0 pins to their
   corresponding locations on the barrier strip, using spade connectors on one end of the
   wire and male pins on the other.  (I like to solder spade connectors to wire ends that will
   go to barrier strip screws, but you can get by with just stripping 3/4 of an inch from the
   end of a wire, tinning the end, and wrapping it around a screw on the barrier strip).
   Run a red wire from the positive wire of the power supply to connection point VIN.
   Run a black wire from the negative wire of the power supply (usually the black wire)
   to the GND barrier strip connection point.
```

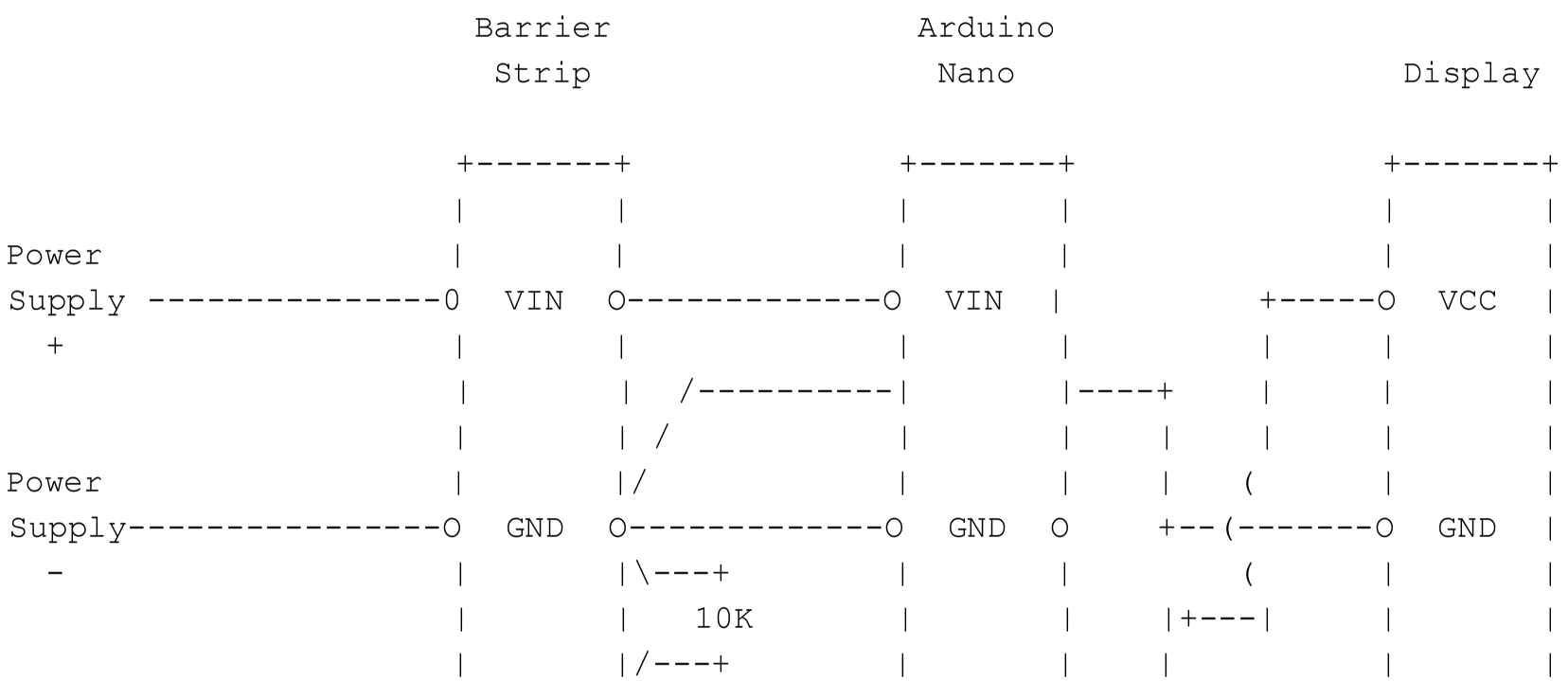
The light dependent resistor is to be mounted between rails of the track. Solder wires to its two leads, long enough to reach the barrier strip. One lead of the LDR is attached to barrier strip's A0 connection point , and the other end to 5V connection point.

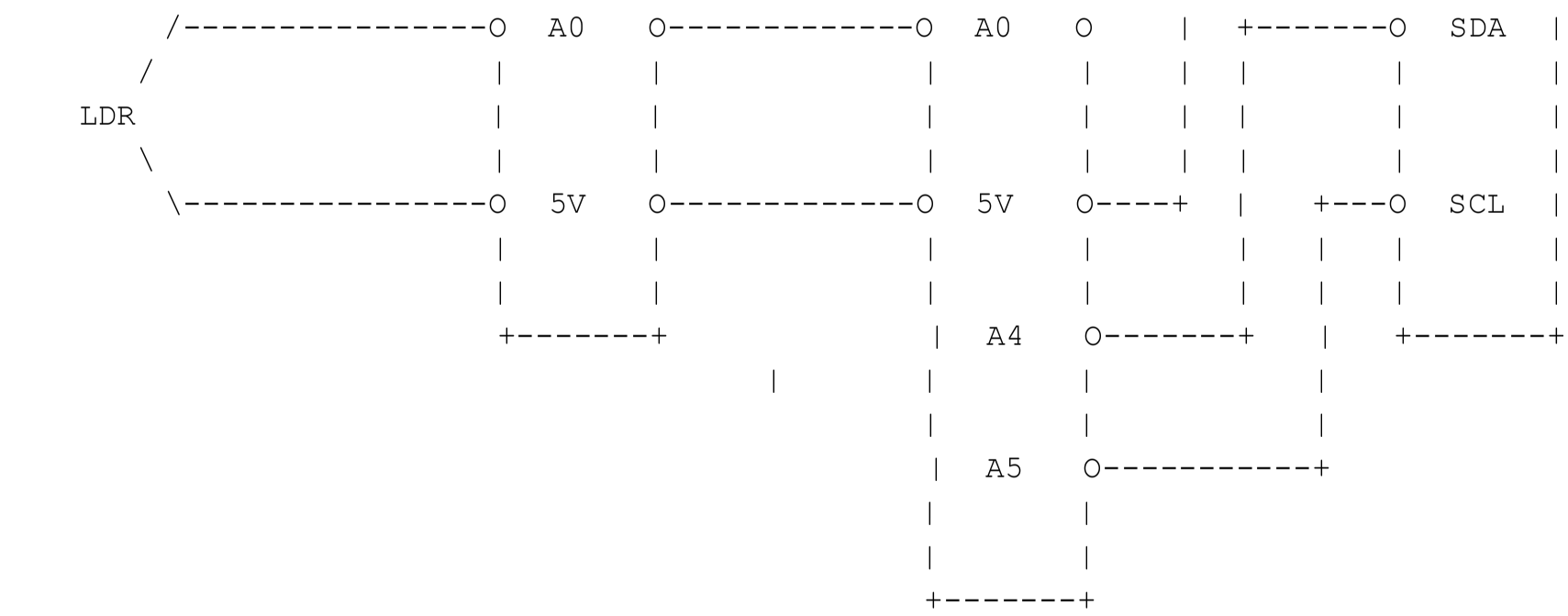
The fixed 10K resistor is also connected to barrier strip'a A0 connection point on one end, and the other end is connected to connection point GND. The GND connection point is also connected to the display's GND pin; all connections on the display require female connectors on the end of thir jumper wires.

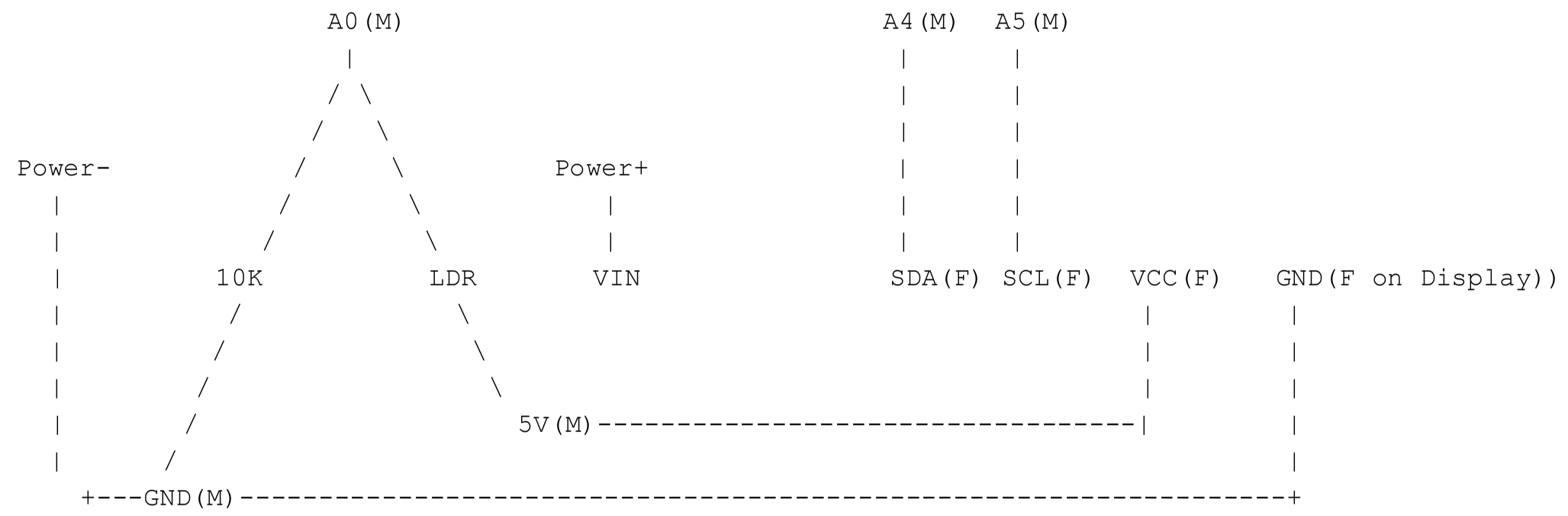
The 5V connection point is also connected to the display's VCC pin

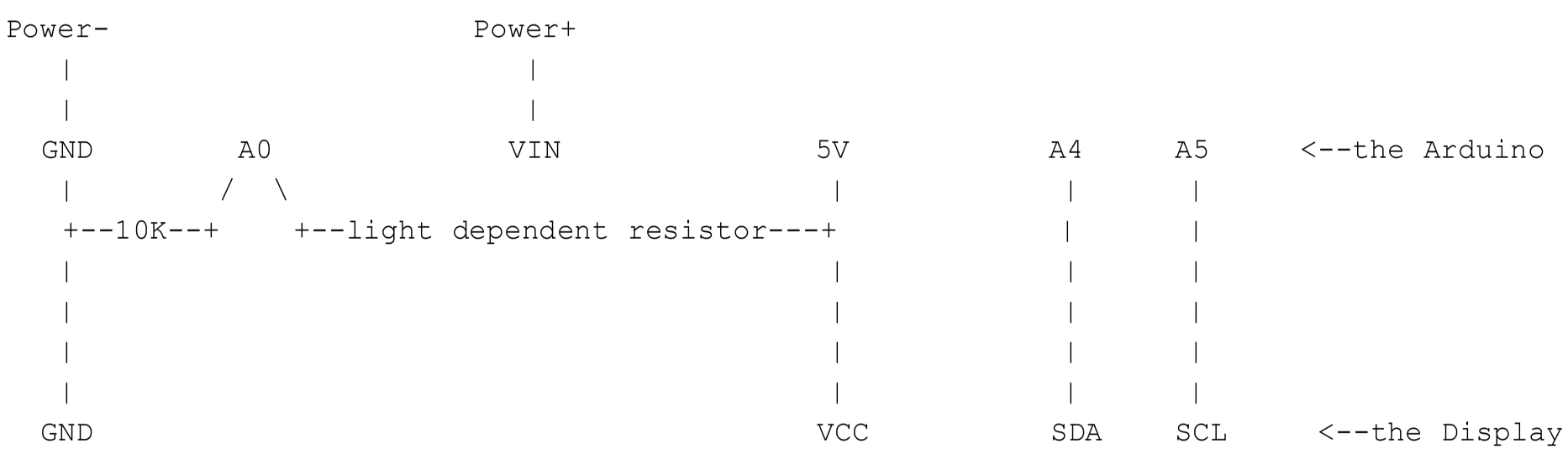
Connect A4 on the Arduino to SDA on the display. Connect A5 on the Arduino to SCL on the displa

Three different but equivalent drawings of the wiring are shown below; the latter two have omitt the barrier strips due to limited space. Wiring diagrams are difficult to create using only alphabetic characters, so I also have a better wiring diagram in my GitHub account at http://www.github.com/daackm; just open the "Speedometer" repository.




LDR
A0
5V
A0
5V
A4
A5
SDA
SCL


A0(M)
A4(M)
A5(M)
Power-
Power+
10K
LDR
VIN
SDA(F)
SCL(F)
VCC(F)
GND(F on Display))
5V(M)
GND(M)



The sketch needs to know the length of the loop ("trackLength", in inches), and the Scale ("scale"), so change these variables accordingly in the "setup". The variable "darkBoundary" represents a value of about half way between the resistance of the light dependent resistor when it is in ambient light and when it is covered. 450 worked for my layout and my resistor, but Your Mileage May Vary; if the system is in a lower ambient light situation, and it bypasses "AWAITING TRANSIT" and goes directly to "Now Timing Transit" you may want to decrease the "darkBoundary" to perhaps 250, or if you are in a brighter environment you might want to raise it to 650.

At about line 220 you see a "LiquidCrystal" line containing a "0x35" parameter. This represents the factory-set address of the display (the display I used came with address "0x35", but address "0x27" is more common). You can use the SCANNER sketch in the LiquidCrystal library to see the address of the one you purchased, snd change the address in that line, if necessary. OH, and you also must

have the LiquidCrystal library loaded into the IDE in order to successfully compile; there are several good YouTubes which show how ths is done.

Before compiling, don't forget to use the TOOLS to specify that you are using a ARDUINO Nano, and that you have set the proper COM port. Also, be sure to adjust the potentiometer on the back of the display so that it starts to display. If it isn't bright, then turn the potentiometer clockwise, but not too far since if it is too bright it will burn an image into the display.

The sketch begins by displaying "SPEEDOMETER" and a version number, plus the selected scale and loop length. 2.5 seconds later it displays "AWAITING TRANSIT", and it sits there until a train passes over the light dependent resistor. When this happens, The top line of the display says "Now timing transit..", and the line below says "Elapsed Time: " and the number of seconds since the transit started. When the train again passes over the light dependent resistor, the bottom line displays the total transit time and the average scale speed in miles per hour. At this point another measurement begins, but the results of the previous transit continu to be displayed until the next transit is completed.

If the train begins a transit but does not complete the transit in a reasonable time, the transit is aborted and the system returns to the "AWAITING TRANSIT" state. A "reasonable time" is defined as being the amount of time that a transit should take at 0.9 miles per hour, in the scale specified.

The Arduino need not be connected to the USB port in order to function. Power to the Arduino is up to you, but I used a 12V "wall wart" (center positive) from AMAZON (ASIN:B07VBR327W). A 9V battery with a cylindrical connector may also be used.

Version 0.50 was the first release in October, 2018
Version 0.51 was never released
Version 0.52 added the logic to handle a locomotive with cars; was completed on 10 December 2018

```
   Version 0.53 restructured the comments and added barrier strips on 08 April 2020.
*/


#include <Wire.h>
#include  <LiquidCrystal_I2C.h>




const unsigned long trackLength=440;    //track loop length is 441 inches
const  String scaleText="HO";
const  int  display=0x27;
unsigned couplerGapMinimum=2000;         //minimum time in milliseconds for an acceptable transit
const unsigned long darkBoundary=450;   //at what light level do we consider it "dark"?
                                         //if 450 doesn't trigger (high ambient light situations,
                                         //try 650,or try 250 in lower light environments




const int photocellPin=0;                 // photocell (aka:sensor, aka:detector) is in pin A0
                                          // (zero) of the Arduino
unsigned long timeSketch;                 // elapsed time in seconds since sketch started
unsigned long timeSec ;
unsigned  long  transitElapsedTime;
unsigned  long  shortElapsedTime;
```

```
unsigned long couplerGapStart;
unsigned long couplerGap;
char version="0.53";
float  transitElapsedTimeSec;
float smph;                                 //scale miles per hour

float scale;
unsigned long elapsedTime;                  //how long have we spent on this transit (millisecond)?
long ElapsedTimeSec ;                       //elapsedTime divided by 1000
unsigned long timeOut;
int trainState=0;                           //zero means that no train has yet passed the detector
                                            //one means that a train has passed the detector
                                            //two means locomotive has completed transit over detector
                                            //and is continuing around loop

unsigned long timeStart=0;                  //stores the time a train started passing a sensor
const int sampleRate=50;                    //how frequently (millisecond) do we sample the photo
resistor?
int resistance=0;                           //What resistance is the photocell reading?


void(* resetFunc)  (void)= 0;
unsigned long time;

LiquidCrystal_I2C lcd(display,20,4);  // set the LCD address to 0x27 for a 20 chars and 4 line
display

void setup()
{

  if(scaleText=="G")
    {
       scale=25.0;   //G is 25:1 of real size
    }
```

```
if(scaleText=="O")
  {
     scale=48.0;    //O is 48:1 of real size
  }
if(scaleText=="S")
  {
     scale=64;    //O is 64:1 of real size
  }
if(scaleText=="HO")
  {
     scale=87.1;    //HO is 87.1:1 of real size
  }
if(scaleText=="T")
  {
     scale=120;    //T is 120:1 of real size
  }
if(scaleText=="N")
  {
     scale=160.0;    //N is 160:1 of real size
  }
if(scaleText=="Z")
  {
     scale=220;    //HO is 220:1 of real size
  }



Serial.begin(9600);
lcd.init();                          // initialize the lcd display
lcd.backlight();
lcd.home();
lcd.setCursor(0,0);
         //12345678901234567890
```

```
lcd.print("Speedometer V        ");
 lcd.setCursor(13,0);
 lcd.print(version);
 lcd.setCursor(0,1);
 lcd.print(" Scale=");
 lcd.print(scaleText);
 lcd.print("  Display=");lcd.print(display);
 lcd.setCursor(0,2);
lcd.print(" Track Length=");
 lcd.print(trackLength);
 lcd.print('"');
delay(4000);


 lcd.setCursor(0,1);
lcd.print("                    ");
 lcd.setCursor(0,2);
lcd.print("                    ");


 lcd.setCursor(0,0);
lcd.print("                    ");
 lcd.setCursor(0,0);
lcd.print("Awaiting transit    ");
  timeOut=((((trackLength*scale)/12.0)/5280.)*.9)*3600.*1000.;
    /* Timeout Calculation:

           scale inches divided by 12 = scale feet
           divided by 5280 = scale miles
           times 0.9 miles per hour = expected hours
           to travel loop @ 0.9 mph
           times 3600 seconds per hour = expected
           seconds to travel loop
           times 1000 = expected milliseconds to
```

```
            travel loop @ 0.9 mph

     */


   Serial.print("timeOut=");Serial.println  (timeOut);

}

void loop()                                          //GRAND LOOP
{

  resistance=(analogRead(photocellPin));             //What is the current value of the photocell?
  timeSketch=millis();                               //How long has this sketch been running?
   timeSec=timeSketch/1000.0;

/*
    If you are having difficulty triggering the speedometer circuit,
    it is most likely that the ambient light on your light dependent
    resistor is significantly different than the environment on
    my railroad.  I suggest that you arm the following code and note
    the value of "ohms" being reported by your LDR.  For my railroad,
    the value under normal conditions (ie: not covered) is about 680
    ohms, and it drops to about 390 ohms when covered.  My "trigger"
    value is set in the code at 450 ohms, so when the value drops
    below 450, the LDR is being covered and the circuit should begin
    timing a new transit.  If ohms is below 450 to start with (indicating
    a fairly dark room), you might want to lower the value of the
    "darkBoundary" value around line 182; conversely, if the circuit
    never triggers, you may want to raise the value of darkBoundary.
*/
   lcd.setCursor(0,2);
   lcd.print("Ohm=");
   lcd.setCursor(4,2);
```

```
 lcd.print(resistance);
 lcd.setCursor(10,2);
 lcd.print("darkB=");
 lcd.setCursor(16,2);
 lcd.print(darkBoundary);



if(trainState==0)                  //if no train has yet triggered the sensor
   {
     if(resistance<darkBoundary)   //have we triggered the sensor?
       {
       trainState=1;
       timeStart=millis();
       lcd.setCursor(0,3);
       lcd.print("                    ");
       lcd.setCursor(0,0);
                 //12345678901234567890   How many characters are in the string?
       lcd.print("Now timing transit..   ");
       lcd.setCursor(0,1);
       lcd.print("                    ");
       lcd.setCursor(0,1);
       lcd.print("Elapsed Time: 0");
      }                                //end of resistance<darkBoundary
   }                                   //end of trainState=0


if(trainState==1)                  //train is passing over the sensor
  {
     elapsedTime=millis()-timeStart;
     lcd.setCursor(14,1);
     lcd.print("     ");
     lcd.setCursor(14,1);
     shortElapsedTime=elapsedTime/1000;
```

```
       //Serial.print("shortElapsedTime=");Serial.println(shortElapsedTime);
      //lcd.print(shortElapsedTime/1000);


      if(resistance>darkBoundary)
      {
        trainState=2;                //train has passed sensor, start looking for its return
        couplerGapStart=millis();
      }
    }

if(trainState==2)                  //train has completed transit over sensor; await its return
    {
      elapsedTime=millis()-timeStart;
      lcd.setCursor(14,1);
      lcd.print(elapsedTime/1000.0,0);

      if(resistance<darkBoundary)     //train has again passed sensor, report findings and restart
timer
      {
        couplerGap=millis()-couplerGapStart;
          Serial.println(" ");
          Serial.println(" ");
          Serial.println(" ");
           Serial.print("couplerGap=");Serial.print(couplerGap);
           Serial.print("    couplerGapMinimum=");Serial.println(couplerGapMinimum);
        if (couplerGap>couplerGapMinimum)   //make sure the minimum transit time has passed
                                            //otherwise, assume it is a gap between cars
        {
          Serial.println("minimum time has past");
        trainState=1;
         transitElapsedTime=elapsedTime;
          transitElapsedTimeSec=transitElapsedTime/1000.0;
        timeStart=millis();
```

```
        lcd.setCursor(0,3);
       lcd.print("                    ");
        lcd.setCursor(0,3);
        lcd.print(transitElapsedTimeSec,1);
       lcd.print(" sec  ");
         smph=(((trackLength*scale)/12.0)/5280.)/((transitElapsedTime/1000.)/3600.);
         //Serial.println("length=");Serial.print(((trackLength*scale)/12.0)/5280);
       //Serial.print("  ");
        //Serial.print(transitElapsedTime);
       lcd.print(smph,1);
       lcd.print(" mph");
       }       //end of couplerGap test TRUE
       else    //beginning of couplerGap test FALSE
      {
         //we are in this piece of logic because the sensor has gone dark (it is covered),
         //but the minimum time of lightness has not occurred (ie: we have a gap between a
         //locomotive and its cars, or between cars).  SO, we want to wait for lightness to
         //again occur, and then return to the state where we are checking
         //for a return to darkness after an acceptable time.
        trainState=2;
         Serial.println("couplerGap logic entered");
        do
        {
         resistance=(analogRead(photocellPin));
        delay(sampleRate);
        }
         while (resistance>darkBoundary);     //end of do while
          Serial.print("exiting couplerGap logic.  elapsedTime=");Serial.println(elapsedTime,
1);
         couplerGapStart=millis();
       }      //end of else
     }        //end of resistance>darkBoundary
  }           //end of trainState=2
```

```
    if (elapsedTime>timeOut)          //if train hasn't returned to sensor in a reasonable time
                                       //assume it is stopped and return to state 0 (awaiting transi
      {
         lcd.setCursor(0,3);
        lcd.print("                      ");
         lcd.setCursor(0,3);
         lcd.print("Transit Timeout");
        trainState=0;
         lcd.setCursor(0,0);
        lcd.print("                      ");
         lcd.setCursor(0,0);
        delay(2000);
         lcd.print("Awaiting transit    ");
         lcd.setCursor(0,1);
        lcd.print("                      ");
      }                               //endif of elapsedTime>timeOut




  if(timeSketch>50000000)
     {
      resetFunc();
     }

  delay(sampleRate);
}                                 //end of GRAND LOOP
```