Arduino Ideas

[Scale speedometer 2](#_Toc184685944)

[Traffic lights – single button/road, vs multiple buttons/roads 9](#_Toc184685945)

[RGB LED/LDR colour mixer sensor/pen 10](#_Toc184685946)

[Diesel the Hamster’s tachometer and speedometer 14](#_Toc184685947)

[Variable speed disco light +/- pulsing with active RC filters 23](#_Toc184685948)

[Strobe light for stroboscopy, variable settings 24](#_Toc184685949)

[Simple 3-digit thermometer with LM35 +/- max/min display 25](#_Toc184685950)

# Scale speedometer

## Thoughts/Work in progress

* Time taken over a set length using reed switches or light sensor
  + Fixed distance between posts
  + Output to display in true fixed scale
* Time taken for a vehicle length to pass multiplied by true size
  + Need to set length of vehicle first
  + Then display speed in equivalent scale speed
* Learning:
  + 12-pin 8-digit 7-segment display  
    <https://www.instructables.com/12-Pin-7-Segment-Display-Wiring-Tutorial/>

## Code for Speedometer

/\*

TEENSY 2.0

4-bit 7-segment LED display with decimal points

https://www.instructables.com/12-Pin-7-Segment-Display-Wiring-Tutorial/

Looking down on chip, Pin 1 is bottom left.

Anodes = Pins 12, 9, 8, 6

Segments = A 11, B 7, C 4, D 2, E 1, F 10, G 5

Decimal point = Pin 19

Connect bit CATHODES via resistors to pins 1, 2, 3, 4, respectively.

Connect segment ANODES to pins 12 to 18 respectively

Connect decimal point cathode to pin 19.

Switches-

Pin 7: On/Off switch for setup mode vs. run mode

Pin 8: PTM switch DOWN

Pin 9: PTM switch UP

Selecting segment anode HIGH and bit cathode LOW will light a segment within that bit.

Number Display logic table:

Number | A | B | C | D | E | F | G |

------------------------------------

    0  | x | x | x | x | x | x |   |

    1  |   | x | x |   |   |   |   |

    2  | x | x |   | x | x |   | x |

    3  | x | x | x | x |   |   | x |

    4  |   | x | x |   |   | x | x |

    5  | x |   | x | x |   | x | x |

    6  | x |   | x | x | x | x | x |

    7  | x | x | x |   |   |   |   |

    8  | x | x | x | x | x | x | x |

    9  | x | x | x | x |   | x | x |

\*/

int digitPins[] = { 1, 2, 3, 4 };

int segmentPins[] = { 12, 13, 14, 15, 16, 17, 18, 19 };

int startLDRpin = 21;

int stopLDRpin = 20;

int setupSwitchPin = 6;

int buttonDownPin = 7;

int buttonUpPin = 8;

int switchPinVal = 0;

int buttonDownPinVal = 0;

int buttonUpPinVal = 0;

unsigned long buttonTime = 0;

float valStartLDR;

float valStopLDR;

int triggerVal = 500;

int timerRun = 0;

unsigned long timeStart;

unsigned long timeNow;

unsigned long timeStop = 0.000;

float speed;

float LDRdistance = 10;     // cm

float cms2mph = 0.0223694;  // cm per sec in mph

float scale = 76.0;

float speedcmsec;

//      #  = {a,b,c,d,e,f,g};

int number[10][7] = {

  { 1, 1, 1, 1, 1, 1, 0 },  // number 0

  { 0, 1, 1, 0, 0, 0, 0 },

  { 1, 1, 0, 1, 1, 0, 1 },

  { 1, 1, 1, 1, 0, 0, 1 },

  { 0, 1, 1, 0, 0, 1, 1 },

  { 1, 0, 1, 1, 0, 1, 1 },

  { 1, 0, 1, 1, 1, 1, 1 },

  { 1, 1, 1, 0, 0, 0, 0 },

  { 1, 1, 1, 1, 1, 1, 1 },

  { 1, 1, 1, 1, 0, 1, 1 }  // number 9

};  // end array

float timeValue = 0.000;

void setup() {

  pinMode(startLDRpin, INPUT);     //topLDR - start

  pinMode(stopLDRpin, INPUT);      //bottomLDR - stop

  pinMode(setupSwitchPin, INPUT);  //pin for setup switch

  pinMode(buttonDownPin, INPUT);   //pin for down button

  pinMode(buttonUpPin, INPUT);     //pin for down button

  // Cathodes (digits) pin initialization

  for (int i = 0; i < 4; i++) {

    pinMode(digitPins[i], OUTPUT);

  }

  // Anodes (Segments) pin initialization

  for (int i = 0; i < 8; i++) {  //8th pin is the decimal point

    pinMode(segmentPins[i], OUTPUT);

  }

  delay(500);

  for (int k = 0; k < 4; k++) {

    digitalWrite(digitPins[k], LOW);

  }

  for (int i = 0; i < 8; i++) {

    digitalWrite(segmentPins[i], HIGH);

    delay(50);

    digitalWrite(segmentPins[i], LOW);

    delay(50);

  }

  for (int k = 0; k < 4; k++) {

    digitalWrite(digitPins[k], HIGH);

  }

}  // end setup loop

void displayChar(int input, int position, int totalLength, int dotPosition) {

  /\*

  The displayChar function takes the inputs, as integers, input (digit to be displayed), position (where it lies in the number),

  totalLength (length of the whole number to be displayed, in characters), and dotPosition (where the decimal point is) from the

  function sendToDisplay.

  displayChar firstly ensures that if the character position to be displayed is beyond the decimal point - which itself

  is given its own position by the sendToDisplay function, then the position of that character is reduced by one.

  If the position is 3 or less (i.e., the 4th digit in the 7-segment display where the leftmost is the 0th digit), then

  it drops the cathode for that digit to zero (via digitPins[i]) and a loop starts, firstly a loop to run the ON code 10x for brightness,

  containing instructions to loop through each individual segment and light it according to whether the input number's array determines if it's

  meant to be on or off; then a section to turn the decimal point on or not.

  Finally the function resets the cathodes and anodes to their default state ready to be called again.

  \*/

  if (position > dotPosition) {

    position = position - 1;

  }

  if (position <= 3) {

    digitalWrite(digitPins[position], LOW);  // sets the digit's cathode to low to allow current to flow

    for (int j = 0; j < 10; j++) {

      for (int i = 0; i < 7; i++) {

        digitalWrite(segmentPins[i], number[input][i]);  //turns on the segmentPin[] as determined by the 'input' row of the number[] array

        if (position == dotPosition - 1) {

          digitalWrite(segmentPins[7], HIGH);

        }

      }

    }

    digitalWrite(digitPins[position], HIGH);

  }

  for (int j = 0; j < 8; j++) {

    digitalWrite(segmentPins[j], LOW);

    digitalWrite(segmentPins[7], LOW);

  }

}  //end displayChar function

void sendToDisplay(float input) {

  /\*

Takes the raw value as a float, and converts it to a string

so that each character can be sent across to the displayChar function.

The decimal point position is noted and sent separately.

\*/

  String strFromInput = String(input);

  int lenInputString = strFromInput.length();

  int dotPosition = String(input).indexOf(".");

  for (int i = 0; i < lenInputString; i++) {

    String stringDigit = strFromInput[i];

    int intStringDigit = stringDigit.toInt();

    // displayChar(int number, int position, int totalLength, int dotPosition)

    if (i != dotPosition) {

      displayChar(intStringDigit, i, lenInputString - 1, dotPosition);

    }

  }  // end length for loop

}  // end sendToDisplay function

void loop() {

  /\*

  pinMode(setupSwitchPin, INPUT);  //pin for setup switch

  pinMode(buttonDownPin, INPUT);   //pin for down button

  pinMode(buttonUpPin, INPUT);     //pin for down button

  int switchPinVal = 0;

  int buttonDownPinVal = 0;

  int buttonUpPinVal = 0;

  \*/

  // READING VALUES FROM THE LDRs SECTION //

  valStartLDR = analogRead(startLDRpin);

  valStopLDR = analogRead(stopLDRpin);

  if (valStartLDR < 500) {

    if (valStopLDR < 500) {

    } else if (timerRun == 0) {

      timerRun = 1;

      timeStart = millis();

      Serial.println("Timer triggered");  // only change if not that value

    }

  }

  if (valStopLDR < 500) {

    if (timerRun == 1) {

      timeStop = millis() - timeStart;

      timerRun = 0;

      Serial.println("Timer stopped");

      speedcmsec = (LDRdistance \* scale) / (timeStop / 1000.000);

      Serial.print("Time taken (sec): ");

      Serial.println(timeStop / 1000.000);

      Serial.print("Speed in cm/sec: ");

      Serial.println(speedcmsec);

      speed = speedcmsec \* cms2mph;

      Serial.print("Speed in mph: ");

      Serial.println(speed);

    }

  }

  if (digitalRead(setupSwitchPin)) {

    if (digitalRead(buttonDownPin)) {

      buttonTime = millis();

      scale = scale - 0.5;

      delay(50);

    }

    if (digitalRead(buttonUpPin)) {

      scale = scale + 0.5;

      delay(50);

    }

    sendToDisplay(scale);

  } else {

    if (timerRun == 1) {

      sendToDisplay((millis() - timeStart) / 1000.000);

    } else {

      sendToDisplay(speed);

    }  //sends the value as a raw number to sendToDisplay function

  }

}  // end main loop

# Traffic lights – single button/road, vs multiple buttons/roads

# RGB LED/LDR colour mixer sensor/pen

// Needs Arduino NANO to run

// Must have drivers for CH340 installed

// Must use Old Bootloader selection within IDE > Tools > Processor

/\* Connection requirements

Arduino NANO 3

Pins D6, D5, D3 through 220R to R/G/B of output RGB LED. Cathode of this to ground.

Pins A0, A1, A2 through 220R to R/G/B of input RGB LED. Cathode to this to ground.

Nano ground and Vin to ground and +v respectively

Pin A5 to switch, with capacitor in parallel, and other pin to ground.

LDR connects to +v and top of pot; bottom of pot to ground (bridged wiper). Pin A7 at the node of the LDR and the pot.

\*/

int InRGBLEDpins[3] = { 14, 15, 16 };       // R G B - ( B G Cat R [flat]) - Common CATHODE

int OutRGBLEDpins[3] = { 3, 5, 6 };         // R G B - ( B G Cat R [flat]) - Common CATHODE

String colour[4] = { "R", "G", "B", "D" };  // allows for debugging understanding of saved colours

int LDRpin = A7;

int buttonPin = 19;

bool buttonState = false;

int toggleState = 0;  // initially sets the board into setup mode to set white balance

int LDRvalue;

int delayTime = 20;  // time to keep the RGB LED in ON or OFF state to allow LDR to 'catch up'

int repetitions = 2;

int powerfactor = 2;

int lightVal[2][4] = {};     // rows | columns - raw values of light from LDR, saved in a 'repetitions'-row table (R G B Dark)

float lightLevels[4] = {};   // the raw, averaged values of the 'repetitions' rows from lightVal above (R G B Dark)

float outputLevels[3] = {};  // the output, in percentage, of brightness of values later calculated from between white and dark points

float white[3] = { 1150, 1150, 1150 };  // values for white paper at 15mm - dark is not included, as this will change with reflective material

void setup() {

  Serial.begin(115200);

  pinMode(buttonPin, INPUT\_PULLUP);  // sets the button with pullup resistor

  pinMode(LDRpin, INPUT);            // pin used for LDR

  for (int i = 0; i < 3; i++) {  // sets the RGB LED and LED Strip outputs - the LED Strip has a common CATHODE

    pinMode(InRGBLEDpins[i], OUTPUT);

    pinMode(OutRGBLEDpins[i], OUTPUT);

    digitalWrite(InRGBLEDpins[i], LOW);

    digitalWrite(OutRGBLEDpins[i], LOW);

  }

}  //end Setup

void loop() {

  // this part flashes the RGB LED through R/G/B/Off 5x, and fills the 5 rows of the lightVal array with

  // the corresponding valuves from the LDR

  if (toggleState == 0) {

    for (int c = 0; c < repetitions; c++) {  // runs 'repetitions' times, filling each row of lightVal array

      for (int i = 0; i < 3; i++) {          // flashes through R/G/B/Off and writes to corresponding lightVal array column

        digitalWrite(InRGBLEDpins[i], HIGH);

        delay(delayTime);

        lightVal[c][i] = analogRead(LDRpin);

        digitalWrite(InRGBLEDpins[i], LOW);

        delay(delayTime);

      }

      lightVal[c][3] = analogRead(LDRpin);

    }

    for (int j = 0; j < 4; j++) {                          // R/G/B/Dark columns

      for (int i = 0; i < repetitions; i++) {              // each row of lightVal array

        lightLevels[j] = lightLevels[j] + lightVal[i][j];  //adds each row's value to the last

      }

      lightLevels[j] = lightLevels[j] / repetitions;  //averages out the values

    }

    for (int j = 0; j < 3; j++) {

      if (lightLevels[j] < white[j]) {

        outputLevels[j] = (lightLevels[j] - lightLevels[3]) / (white[j] - lightLevels[3]);  // turns into percentage

        outputLevels[j] = pow(outputLevels[j], powerfactor);                                // increases sensitivity;

      } else {

        outputLevels[j] = 1.0;

      }

      analogWrite(OutRGBLEDpins[j], 255 \* (outputLevels[j]));

    }

  } else if (toggleState == 1) {

    for (int i = 0; i < 3; i++) {

      digitalWrite(InRGBLEDpins[i], LOW);

  }

}

else if (toggleState == 2) {

  // white set mode - white paper needed at ~15mm from sensor/RGB LED

  digitalWrite(OutRGBLEDpins[0], HIGH);  // turns on the red LEDs of the strip to indicate setup mode

  Serial.print("White set - ");

  for (int i = 0; i < 3; i++) {

    white[i] = lightLevels[i];

    Serial.print(colour[i]);

    Serial.print(": ");

    Serial.print(lightLevels[i]);

    Serial.print(" ");

  }

  Serial.print("\n");

}

if (digitalRead(buttonPin) != true) {  //toggles buttonState value to 0 or 1 after button push

  if (buttonState == false) {          // if button not noted as being down

    buttonState = true;

    toggleState++;

    toggleState = toggleState % 2;

    Serial.println("Button down");

    Serial.print("Toggle state: ");

    Serial.println(toggleState);

  } else if (buttonState == true) {  // if button already noted as being down

    Serial.print("Toggle state: ");

    Serial.println(toggleState);

    // do nothing

  }

} else {  //when released

  buttonState = false;

  Serial.print("Toggle state: ");

  Serial.println(toggleState);

}

}  //end buttonState toggle

# Diesel the Hamster’s tachometer and speedometer

/\*

How much exercise does your hamster get?

This is a project to display very important factors regarding your hamster's exertion

by using a magnet glued onto their wheel to trigger a reed switch placed outside the cage.

Data stored will be output via the 7-segment LED display, in different modes chosen by the modeButton;

data is also outputted to the Serial Monitor if plugged in via USB within the Arduino IDE.

Board required: TEENSY 2.0 (other Arduino boards are possible but the pin numbers will vary)

Inputs-

Pin 7: On/off switch for LED display (saves power if you're running on batteries)

Pin 8: Toggle switch between different modes of the display (ie, distance moved, time elapsed, current/average/maximum speed)

Pin 9: Reed switch input

NB to reduce switch bouncing/noise, place a 1nF capacitor in parallel with both switch legs. It is also possible to do this in code.

To prevent floating pins, 10k-ohm pulldown resistors have been employed for all switches.

Output (4-bit 7-segment LED display COM-11405 with decimal points part number HS420561K-32 / Mouser No:474-COM-11405)

see https://www.instructables.com/12-Pin-7-Segment-Display-Wiring-Tutorial/ for example circuit

Looking down on the 7-segment display chip, Pin 1 is bottom left.

Anodes = Pins 12, 9, 8, 6

Segments = A 11, B 7, C 4, D 2, E 1, F 10, G 5

Decimal point = Pin 19

Connect bit CATHODES via PNP transistors (2n3906's, via 1k-ohm resistors) to pins 1, 2, 3, 4, respectively.

You can connect the cathodes directly to the Arduino HOWEVER increasing numbers of lit segments will make the digits appear dimmer that way.

Connect segment ANODES to pins 12 to 18 respectively, via 100-ohm resistors.

Connect decimal point cathode to pin 19 via a 100-ohm resistor.

Selecting segment anode HIGH and bit cathode LOW will light a segment within that bit.

Number Display logic table:

Number | A | B | C | D | E | F | G |

------------------------------------

    0  | x | x | x | x | x | x |   |

    1  |   | x | x |   |   |   |   |

    2  | x | x |   | x | x |   | x |

    3  | x | x | x | x |   |   | x |

    4  |   | x | x |   |   | x | x |

    5  | x |   | x | x |   | x | x |

    6  | x |   | x | x | x | x | x |

    7  | x | x | x |   |   |   |   |

    8  | x | x | x | x | x | x | x |

    9  | x | x | x | x |   | x | x |

\*/

int digitPins[] = { 1, 2, 3, 4 };

int segmentPins[] = { 12, 13, 14, 15, 16, 17, 18, 19 };

int onOffSwitchPin = 6;

int modeButtonPin = 7;

int reedSwitchPin = 8;

//initialise values for inputs

int onOffSwitchPinVal = 0;

int modeButtonPinVal = 0;

int reedSwitchPinVal = 0;

int systemMode = 0;  // system modes, as per the Array systemModes below

String systemModes[5] = { "Elapsed distance (m)", "Elapsed time (sec)", "Current speed (m/sec)", "Average speed (m/sec)", "Maximum speed (m/sec)" };

float wheelDiameter = 13.5;                          // hamster wheel diameter (cm)

float wheelCircumf = 3.14159 \* wheelDiameter / 100;  // pi \* d (m) - converts the wheel diameter into a circumference in metres

unsigned long timePause = 10000;                     // time considered a break between runs in msec (so the average speed display and timeElapsed work properly)

int displayBlanked = 0;  //used later by the blankDisplay() function to determine if the function has been run already

unsigned long timeTriggered = millis();

unsigned long timeElapsed;

unsigned long modeSwitchTime;  // whenever the mode is switched, sets to current millis() value so screen can change for that time only.

float displayData;

float distance = 0.000;  // total distance travelled in m

float timeMoving = 0;    // time in seconds

float speed;

float aveSpeed = 0.0;  // average speed in m/sec

float maxSpeed = 0.0;  // max speed in m/sec

// array to define 7-segment display segments {a,b,c,d,e,f,g} required to display numbers 0 thru 9

int number[10][7] = {

  { 1, 1, 1, 1, 1, 1, 0 },  // number 0

  { 0, 1, 1, 0, 0, 0, 0 },

  { 1, 1, 0, 1, 1, 0, 1 },

  { 1, 1, 1, 1, 0, 0, 1 },

  { 0, 1, 1, 0, 0, 1, 1 },

  { 1, 0, 1, 1, 0, 1, 1 },

  { 1, 0, 1, 1, 1, 1, 1 },

  { 1, 1, 1, 0, 0, 0, 0 },

  { 1, 1, 1, 1, 1, 1, 1 },

  { 1, 1, 1, 1, 0, 1, 1 }  // number 9

};  // end array

void setup() {  // sets all the pins and then lights all the digits' segments in turn as a display test

  pinMode(onOffSwitchPin, INPUT);  //pin for LED display on/off switch

  pinMode(modeButtonPin, INPUT);   //pin for the MODE button

  pinMode(reedSwitchPin, INPUT);   //pin for REED switch

  // Cathodes (digits) pin initialization

  for (int i = 0; i < 4; i++) {

    pinMode(digitPins[i], OUTPUT);

  }

  // Anodes (Segments) pin initialization

  for (int i = 0; i < 8; i++) {  //8th pin is the decimal point

    pinMode(segmentPins[i], OUTPUT);

  }

  delay(500);

// brief set of loops to show light all available segments in order on the display and then turn it off

  for (int k = 0; k < 4; k++) { //writes cathodes LOW, i.e., on

    digitalWrite(digitPins[k], LOW);

  }

  for (int i = 0; i < 8; i++) { // in order writes each segment high then low, i.e., turns on then off

    digitalWrite(segmentPins[i], HIGH);

    delay(50);

    digitalWrite(segmentPins[i], LOW);

    delay(50);

  }

  for (int k = 0; k < 4; k++) { // writes all cathodes HIGH, i.e., off.

    digitalWrite(digitPins[k], HIGH);

  }

}  // end setup loop

void displayChar(int input, int position, int totalLength, int dotPosition) {

  /\*

  The displayChar function takes the inputs, as integers, input (digit to be displayed), position (where it lies in the number),

  totalLength (length of the whole number to be displayed, in characters), and dotPosition (where the decimal point is) from the

  function sendToDisplay.

  displayChar firstly ensures that if the character position to be displayed is beyond the decimal point - which itself

  is given its own position by the sendToDisplay function, then the position of that character is reduced by one.

  If the position is 3 or less (i.e., the 4th digit in the 7-segment display where the leftmost is the 0th digit), then

  it drops the cathode for that digit to zero (via digitPins[i]) and a loop starts, firstly a loop to run the ON code 10x for brightness,

  containing instructions to loop through each individual segment and light it according to whether the input number's array determines if it's

  meant to be on or off; then a section to turn the decimal point on or not.

  Finally the function resets the cathodes and anodes to their default state ready to be called again.

  \*/

  if (position > dotPosition) { //because the dot passed to this function counts as one of the characters in the string

    position = position - 1;

  }

  if (position <= 3) {

    digitalWrite(digitPins[position], LOW);  // sets the particular digit's cathode to low to allow current to flow

    for (int j = 0; j < 10; j++) { // repeats the whole thing 10 times for brightness purposes (longer time on than off)

      for (int i = 0; i < 7; i++) {

        digitalWrite(segmentPins[i], number[input][i]);  //turns on the segmentPin[] as determined by the 'input' row of the number[] array

        if (position == dotPosition - 1) {

          digitalWrite(segmentPins[7], HIGH);

        }

      }

    }

    digitalWrite(digitPins[position], HIGH);

  }

  for (int j = 0; j < 8; j++) { //turns the segments and decimal points off again

    digitalWrite(segmentPins[j], LOW);

    digitalWrite(segmentPins[7], LOW);

  }

}  //end displayChar function

void sendToDisplay(float input) {

/\*

Takes the raw number given to it as a float, and converts it to a string

so that each character can be sent across to the displayChar function.

The decimal point position is noted and sent separately.

\*/

  String strFromInput = String(input); //turns the float into an array full of strings

  int lenInputString = strFromInput.length(); // determines the length of the input string

  int dotPosition = String(input).indexOf("."); // finds where the decimal point is

  for (int i = 0; i < lenInputString; i++) {

    String stringDigit = strFromInput[i];

    int intStringDigit = stringDigit.toInt();

    // displayChar(int number, int position, int totalLength, int dotPosition)

    if (i != dotPosition) {

      displayChar(intStringDigit, i, lenInputString - 1, dotPosition);

    }

  }  // end length for loop

}  // end sendToDisplay function

void blankDisplay() {

  // function to turn off the LED display to save power. When the switch is set to OFF,the main loop calls for a blank display

  // checks value of displayBlanked, defaulted to 0 (i.e., "no"), and if that's the case, resets all the segments to a low state and sets displayBlanked to 1

  // if displayBlanked == 1, then does nothing, assuming all segments are already blanked.

  if (displayBlanked == 0) {

    for (int k = 0; k < 4; k++) {

      digitalWrite(digitPins[k], LOW);

    }

    for (int i = 0; i < 8; i++) {

      digitalWrite(segmentPins[i], LOW);

      delay(50);

    }

    for (int k = 0; k < 4; k++) {

      digitalWrite(digitPins[k], HIGH);

    }

    displayBlanked = 1;

  } else {

  };

}  // end blankDisplay function

void loop() {

/\* Variables/functions in use:

sendToDisplay(float); // sends the float to the LED display via sendToDisplay and displayChar functions

onOffSwitchPin = setting of SPDT switch to switch the display on or off

modeButtonPin = status of the pin for toggling between display modes

reedSwitchPin = pin which is closed whenever magnet passes reed switch

float distance: total distance travelled in m

unsigned long timeElapsed: time in seconds spent moving

float speed: current speed in m/sec

float aveSpeed: average speed in m/sec

float maxSpeed: max speed in m/sec

unsigned long modeSwitchTime - millis() time at which the mode was switched

unsigned long timeTriggered - millis() time the reed switch was triggered

int timePause; // time to consider a break between runs in msec

onOffSwitchPinVal = 0;

modeButtonPinVal = 0;

reedSwitchPinVal = 0;

int systemMode - systemModes as per array

\*/

  if (digitalRead(reedSwitchPin)) {  // event happens whenever reedSwitch is triggered, only if it's not already noted to be triggered

    if (reedSwitchPinVal == 0) {

      reedSwitchPinVal = 1;

      if (timeTriggered != 0) {

        if (millis() - timeTriggered < timePause) { // increments timeElapsed so long as the time is less than the designated break

          timeElapsed = timeElapsed + (millis() - timeTriggered);

        }

      }

      Serial.print(millis());

      Serial.print(" | Time moving: ");

      Serial.print(timeElapsed / 1000.0);

      if (millis() - timeTriggered < timePause) { // displays the speed again so long as a long break hasn't occurred

        speed = 1000 \* wheelCircumf / (millis() - timeTriggered);

      }

      timeTriggered = millis();  // saves the time when the reedSwitch is triggered for reference ABOVE

      distance = distance + wheelCircumf;

      Serial.print(" s | Distance: ");

      Serial.print(distance);

      aveSpeed = distance / (timeElapsed / 1000);

      Serial.print(" m | Speed: ");

      Serial.print(speed);

      Serial.print(" m/sec | Average Speed ");

      Serial.print(aveSpeed);

      Serial.print(" m/sec | Maximum Speed ");

      Serial.print(maxSpeed);

      Serial.println(" m/sec");

      if (maxSpeed < speed) { maxSpeed = speed; };

    }

  } else {

    if (reedSwitchPinVal == 1) { reedSwitchPinVal = 0; };

  }

  if (digitalRead(modeButtonPin)) {  // event happens whenever modeButton is triggered

    if (modeButtonPinVal == 0) {

      modeSwitchTime = millis() + 1000;  //the time value for when the mode display should changeSerial.print("Mode switch time = ");

      modeButtonPinVal = 1;

      if (systemMode < 4) {

        systemMode = systemMode + 1;

      } else {

        systemMode = 0;

      };

      Serial.print("\nSystem mode: ");

      Serial.print(systemModes[systemMode]);

      Serial.print(" at time: ");

      Serial.println(modeSwitchTime);

    }

  } else {

    if (modeButtonPinVal == 1) { modeButtonPinVal = 0; };

  }

  if (systemMode == 0) {  // distance travelled

    displayData = distance;

  } else if (systemMode == 1) {  // time taken

    displayData = timeElapsed / 1000.0;

  } else if (systemMode == 2) {  // current speed

    displayData = speed;

  } else if (systemMode == 3) {  // average speed

    displayData = aveSpeed;

  } else {  // max speed

    displayData = maxSpeed;

  }

  // the following needs to come LAST, if the display switch is turned on, it'll display whatever is within the displayData variable

  if (digitalRead(onOffSwitchPin)) {

    blankDisplay();

  } else {

    displayBlanked = 0;

    if (modeSwitchTime > millis()) {

      displayChar(systemMode, 1, 1, 1);

    } else {

      sendToDisplay(displayData);

    }

  }

}  // end main loop

# Variable speed disco light +/- pulsing with active RC filters

# Strobe light for stroboscopy, variable settings

# Simple 3-digit thermometer with LM35 +/- max/min display