```
' {$STAMP BS2}
' {$PBASIC 2.5}

''''''''''''''''''''''''''
''''''''''''''''''''''''''
''' User Configurations

' Garage constants - distance from device, which is at x=0 (the wall is behind it, at some negative X.
' Units are in inches
' Required that 0<A<B<C<D
pointA      CON 1 ' Closest point where it is proper to stop. In real life, probably 5"
pointB      CON 2 ' Furthest point where it is proper to stop.  In real life, probably 10"
pointC      CON 5 ' Point at which the driver should be slowing down. 50"
pointD      CON 8 ' Car is completely out of the garage. Real life 180"

' Response Durations
glowDuration CON 2000

' Distance sensitivity threshold
distanceThreshold CON 2

''''''''''''''''''''''''''
''''''''''''''''''''''''''
''''''''''''''''''''''''''
''' Board configuration

' PIN Constants
SpeakerPin   CON 0 ' Speaker connected to Pin 0
GreenLEDPin  CON 1 ' Green LED connected to Pin 1
YellowLEDPin CON 3 ' Yellow LED connected to Pin 3
RedLEDPin    CON 5 ' Red LED connected to Pin 5

''''''''''''''''''''''''''
''''''''''''''''''''''''''
''''''''''''''''''''''''''
''' Nature configuration

' Sound Travel constant, divided by 2
HalfInchesPerSecond CON 890

' real time variables
pulseTime          VAR Word ' gets populated from PULSIN
currentDistance    VAR Word ' gets calculated immediately afterwards
previousDistance   VAR Word ' = currentDistance from previous iteration. Helps determine the direction below. starts
at 0
previousDistance = 0
direction          VAR Nib  ' 2=entering the garage, 1=Stationary, 0=Exiting
segmentName        VAR Nib  ' 0=between 0 and A, 1=AB, 2=BC, 3=CD, 4=D+
remainingProportion VAR Word ' to controls blinking rate, this is the proportion of the travel distance remaining:
0=stop now (slow), 100=at garage entry (fast)

' Desired Light Guidance. We will calculate that as a single number from 0 to 8. This will tell us what lights are to be
on.
```

```
' Note that some of these may not be called for now, but the definition will cover use for future versions
desiredLightGuidance     VAR Nib
'0=green glow steady
'1=green blink proportionately
'2=green blink fast
'3=yellow glow steady
'4=yellow blink proportionately
'5=yellow blink fast
'6=red glow steady
'7=red blink proportionately
'8=red blink fast

' Pertaining to the situation:
desiredLEDPinSeq      VAR Nib  ' we will divide the desiredBehavior by 3, keeping the integer portion: 0=Green,
1=Yellow, 2=Red
desiredLEDPin         VAR Nib  ' this will become one of GreenLEDPin, YellowLEDPin, RedLEDPin
desiredLEDBehaviorSeq VAR Nib  ' we will divide the desiredBehavior by 3, keeping the modulus: 0=Steady, 1=Blink
Proportionately, 2=Blink Fast
desiredLEDBehavior    VAR Nib  ' we will divide the desiredBehavior by 3, keeping the modulus: 0=Steady, 1=Blink
Proportionately, 2=Blink Fast
blinkOnDuration       VAR Word ' gets determined based on the remaining proportion
blinkOffDuration      VAR Word ' gets determined based on the remaining proportion

' Initialize - check that the points have been correctly defined
IF pointA <  0      THEN Segments_Are_Overlapping
IF pointB <= pointA THEN Segments_Are_Overlapping
IF pointC <= pointB THEN Segments_Are_Overlapping
IF pointD <= pointC THEN Segments_Are_Overlapping

' Enter the loop
DO
  PULSOUT 7, 5
  PULSIN 7, 1, pulseTime
  currentDistance = HalfInchesPerSecond ** pulseTime
  '
  '
  IF previousDistance = 0 THEN
    ' First time: skip everything else
  ELSE
    ' Assume stationary, direction only if movement over threshold
    direction = 1
    IF currentDistance < previousDistance THEN
      IF ( previousDistance - currentDistance ) * 10 > distanceThreshold THEN
        direction = 2 ' Entering
      ENDIF
    ENDIF
    IF currentDistance > previousDistance THEN
      IF ( currentDistance - previousDistance ) * 10 > distanceThreshold THEN
        direction = 0 ' Exiting
      ENDIF
    ENDIF
    ' what segment are we in?
    IF     currentDistance > pointD THEN
      segmentName       = 4
```

```
  ELSEIF currentDistance > pointC THEN
    segmentName      = 3
  ELSEIF currentDistance > pointB THEN
    segmentName      = 2
  ELSEIF currentDistance > pointA THEN
    segmentName      = 1
  ELSE
    segmentName      = 0
  ENDIF
  ' what proportion of the in-garage travel distance is remaining? This could have been combined with the previous structure
  IF     currentDistance > pointD THEN
    remainingProportion = 100 ' blink fast
  ELSEIF currentDistance < pointB THEN
    remainingProportion = 100 ' counter-intuitive: you would think it is 0. Setting to 100 so as to blink fast
  ELSE
    remainingProportion =  100 * ( currentDistance - pointB ) / ( pointD - pointB )
  ENDIF
  '
  ' Determine the Situation Number. While this is a verbose way of doing it, it enables intricate configuration
  IF     segmentName = 0 THEN ' too close to the sensor; entering or leaving
    desiredLightGuidance  = 8
  ELSEIF direction    = 0 THEN ' leaving. Cases below here pertain to dir=1 (stationary) and =2, entering. Same treatment, but more ELSEIFs can be written
    desiredLightGuidance  = 0
  ELSEIF segmentName = 1 THEN ' entering - stop here!
    desiredLightGuidance  = 6
  ELSEIF segmentName = 2 THEN ' entering - closing in
    desiredLightGuidance  = 4
  ELSEIF segmentName = 3 THEN ' entering - come on in!
    desiredLightGuidance  = 1
  ELSE ' i.e. segmentName = 4 THEN
    desiredLightGuidance  = 0
  ENDIF
  '
  ' Determine the selected LED Pin - first as 0, 1, 2...
  desiredLEDPinSeq = desiredLightGuidance / 3
  '
  ' Then as a pin# Green, Yellow, Red - extremely rare case when a variable name can be re-used
  IF     desiredLEDPinSeq      = 0 THEN
      desiredLEDPin        = GreenLEDPin
  ELSEIF desiredLEDPinSeq      = 1 THEN
      desiredLEDPin        = YellowLEDPin
  ELSE ' i.e. desiredLEDPinSeq = 2
      desiredLEDPin        = RedLEDPin
  ENDIF
  '
  ' Determine what the LED will do 0=glow, 1=blink proportionately, 2=blink fast
  desiredLEDBehaviorSeq = desiredLightGuidance // 3
  '
  ' Before doing ANYTHING, Report EVERYTHING to the debugger window
  DEBUG CLS
  DEBUG HOME, "Previous Distance: ", DEC3 previousDistance, " in"
  DEBUG CR, "Current Distance: ", DEC3 currentDistance, " in"
```

```
    DEBUG CR
    DEBUG CR, "Direction as a number: ", DEC3 direction
    IF direction = 0 THEN
      DEBUG CR, "(exiting)"
    ELSEIF direction = 1 THEN
      DEBUG CR, "(stationary)"
    ELSE
      DEBUG CR, "(entering)"
    ENDIF
    DEBUG CR
    DEBUG CR, "Segment #: " , DEC3 segmentName
    DEBUG CR, "% Proportion remaining:" , DEC3 remainingProportion
    DEBUG CR
    DEBUG CR, "Desired Light Guidance as a single number #: " , DEC3 desiredLightGuidance
    DEBUG CR
    DEBUG CR, "Guidance, decoded as a sequential PIN number: " , DEC3 desiredLEDPinSeq
    DEBUG CR, "Desired LED Pin #: " , DEC3 desiredLEDPin
    IF desiredLEDPinSeq = 0 THEN
      DEBUG CR, "(green)"
    ELSEIF desiredLEDPinSeq = 1 THEN
      DEBUG CR, "(yellow)"
    ELSE
      DEBUG CR, "(red)"
    ENDIF
    DEBUG CR
    DEBUG CR, "Guidance, decoded as a sequential Behavior number: " , DEC3 desiredLEDBehaviorSeq
    IF desiredLEDBehaviorSeq = 0 THEN
      DEBUG CR, "(glow steady)"
    ELSEIF desiredLEDBehaviorSeq = 1 THEN
      DEBUG CR, "(blink proportionately)"
    ELSE
      DEBUG CR, "(blink fast)"
    ENDIF
    '
    ' Do it
    '
    GOSUB Lights_Out ' this is not needed now because each sub turns off its light. It will be needed if we get an
additional controller to flash lights while this program is executing
    '
    ' fast is the same as proportional where proportion=100, so we handle both here
    ON desiredLEDBehaviorSeq GOSUB Glow, Blink, Blink
    '
  ENDIF
  previousDistance = currentDistance
LOOP
END

Blink:
  ' This math makes the blink on between 500 (slow) and 2000 (fast). Parentheses NOT needed; shown for readability
  blinkOnDuration = ( 100 - remainingProportion ) * 15 + 500
  ' we can play games with this one as well
  blinkOffDuration = 500
  HIGH desiredLEDPin
  PAUSE blinkOnDuration
```

```
  LOW  desiredLEDPin
  PAUSE blinkOffDuration
  RETURN

Glow:
  HIGH desiredLEDPin
  PAUSE glowDuration
  RETURN

Lights_out:
  LOW GreenLEDPin
  LOW YellowLEDPin
  LOW RedLEDPin
  RETURN

Segments_Are_Overlapping:
  DEBUG HOME, "Segment Definitions are Overlapping"
  DO
    HIGH RedLEDPin
    PAUSE 500
    LOW  RedLEDPin
    PAUSE 500
  LOOP
  END
```