#include <Time.h> // include time library

#include <TimeLib.h>

#define echoPinLeft 7 // Echo Pin for Left Module

#define trigPinLeft 8 // Trigger Pin for Left Module

#define echoPinRight 2 // Echo Pin for Right Module

#define trigPinRight 4 // Trigger Pin for Right Module

#define buzzer 12 // Onboard Buzzer

#define collision 10 //Collision Pin for Module

#define TIME\_HEADER "T" // Header tag for serial time sync message

#define TIME\_REQUEST 7 // ASCII bell character requests a time sync message

void setup() {

Serial.begin (9600);

pinMode(trigPinLeft, OUTPUT);

pinMode(trigPinRight, OUTPUT);

pinMode(echoPinLeft, INPUT);

pinMode(echoPinRight, INPUT);

pinMode(buzzer, OUTPUT);

pinMode(collision, INPUT);

setSyncProvider(requestSync); //set function to call when sync required

Serial.println("Waiting for sync message");

}

int val; // conditional for collision module

long durationRight, distanceRight, durationLeft, distanceLeft; // Duration used to calculate distance

int maximumRange = 100; // Maximum range needed

int minimumRange = 1; // Minimum range needed

int distanceLeftTemp, distanceRightTemp; // conditionals variables for checking if distance has changed

void loop() {

if (Serial.available()) {

processSyncMessage(); // process cellular sync if available, 1970 1 1 is set to default

}

// determine if collision occurs

val = digitalRead(collision);

if(val == LOW){

Serial.println("Collision");

digitalClockDisplay(); // print time in memory

digitalWrite(buzzer, HIGH);

}

else {

Serial.println("Good");

digitalWrite(buzzer, LOW);

}

/\* The following trigPin/echoPin cycle is used to determine the

distance of the nearest object by bouncing soundwaves off of it. \*/

digitalWrite(trigPinLeft, LOW);

delayMicroseconds(2);

digitalWrite(trigPinLeft, HIGH);

delayMicroseconds(2);

digitalWrite(trigPinLeft, LOW);

durationLeft = pulseIn(echoPinLeft, HIGH);

//Calculate the distance (in cm) based on the speed of sound.

distanceLeft = durationLeft/58.2;

digitalWrite(trigPinRight, LOW);

delayMicroseconds(2);

digitalWrite(trigPinRight, HIGH);

delayMicroseconds(2);

digitalWrite(trigPinRight, LOW);

durationRight = pulseIn(echoPinRight, HIGH);

//Calculate the distance (in cm) based on the speed of sound.

distanceRight = durationRight/58.2;

if ((distanceLeft <= maximumRange && distanceLeft > minimumRange ) || (distanceRight <= maximumRange && distanceRight > minimumRange)){

/\* Send the distance to the computer using Serial protocol, and

turn buzzer on to indicate successful reading. \*/

if((distanceLeft == distanceLeftTemp) || (distanceRight == distanceRightTemp)){

Serial.println("Distance Hasn't Changed");

digitalWrite(buzzer, LOW);

distanceLeftTemp = distanceLeft;

distanceRightTemp = distanceRight;

}

else {

Serial.println("Distance Left: ");

Serial.println(distanceLeft);

Serial.println("Distance Right: ");

Serial.println(distanceRight);

distanceLeftTemp = distanceLeft;

distanceRightTemp = distanceRight;

digitalWrite(buzzer, HIGH);

}

}

else {

/\* Send a negative number to computer and Turn buzzer off

to indicate "out of range" \*/

Serial.println("-1");

// digitalWrite(buzzer, LOW);

}

//Delay 100ms before next reading.

delay(1000);

}

void digitalClockDisplay(){

// digital clock display of the time

Serial.print(hour());

printDigits(minute());

printDigits(second());

Serial.print(" ");

Serial.print(day());

Serial.print(" ");

Serial.print(month());

Serial.print(" ");

Serial.print(year());

Serial.println();

}

void printDigits(int digits){

// utility function for digital clock display: prints preceding colon and leading 0

Serial.print(":");

if(digits < 10)

Serial.print('0');

Serial.print(digits);

}

void processSyncMessage() {

unsigned long pctime;

const unsigned long DEFAULT\_TIME = 1357041600; // Jan 1 2013

if(Serial.find(TIME\_HEADER)) {

pctime = Serial.parseInt();

if( pctime >= DEFAULT\_TIME) { // check the integer is a valid time (greater than Jan 1 2013)

setTime(pctime); // Sync Arduino clock to the time received on the serial port

}

}

}

time\_t requestSync()

{

Serial.write(TIME\_REQUEST);

return 0; // the time will be sent later in response to serial mesg

}