#include <HCSR04.h> // include Library - use the extension manager to add this library to your environment

// Define the hardware connections:

const int pin\_buzzer = 9; // pin where the buzzer is connceted to

const int pin\_Echo = 8; // digital pin for HC-SR04 echo connector

const int pin\_Trigger = 7; // digital pin for HC-SR04 Trigger connector

// Define the signalling

const int tone\_repeat = 1; // How often a tone / smile will be repeated

const int tone\_low = 800; // Frequency for tone if emergency level is low

const int time\_low = 500; // Tone duration if emergency level is low

const int dist\_low = 150; // distance level for low emergency level

const int tone\_medium = 1600; // Frequency for tone if emergency level is medium

const int time\_medium = 350; // Tone duration if emergency level is medium

const int dist\_medium = 100; // distance level for medium emergency level

const int tone\_high = 2400; // Frequency for tone if emergency level is high

const int time\_high = 200; // Tone duration if emergency level is high

const int dist\_high = 50; // distance level for high emergency level

// to store the current distance:

float distance; // distance read from ultrasonic sensor

//create an instance to control the sensor:

UltraSonicDistanceSensor distanceSensor(7, pin\_Echo); // Initialize sensor that uses digital pins 13 and 12.

/\*

\* Setup steps

\*/

void setup () {

Serial.begin(9600); // Enabled for debzg

// set all LED pins as output

pinMode(3, OUTPUT);

pinMode(4, OUTPUT);

pinMode(6, OUTPUT);

pinMode(12, OUTPUT);

pinMode(A4, OUTPUT);

pinMode(2, OUTPUT);

pinMode(5, OUTPUT);

pinMode(10, OUTPUT);

pinMode(11, OUTPUT);

pinMode(A5, OUTPUT);

pinMode(A6, OUTPUT);

Start(200); // Call start eequence

}

/\*

\* Start Sequence

\* ==============

\*

\* Params:

\* -------

\* startpeed: time in milliseconds between each LED step

\*

\*/

void Start(int startspeed) {

digitalWrite(3,HIGH);

delay(startspeed);

digitalWrite(4,HIGH);

delay(startspeed);

digitalWrite(6,HIGH);

delay(startspeed);

digitalWrite(A4,HIGH);

delay(startspeed);

digitalWrite(12,HIGH);

delay(startspeed);

digitalWrite(2,HIGH);

delay(startspeed);

digitalWrite(5,HIGH);

delay(startspeed);

digitalWrite(A5,HIGH);

delay(startspeed);

digitalWrite(10,HIGH);

delay(startspeed);

digitalWrite(11,HIGH);

delay(startspeed);

delay(500);

digitalWrite(12,LOW);

delay(startspeed);

digitalWrite(A4,LOW);

delay(startspeed);

digitalWrite(6,LOW);

delay(startspeed);

digitalWrite(4,LOW);

delay(startspeed);

digitalWrite(3,LOW);

delay(startspeed);

digitalWrite(11,LOW);

delay(startspeed);

digitalWrite(10,LOW);

delay(startspeed);

digitalWrite(A5,LOW);

delay(startspeed);

digitalWrite(5,LOW);

delay(startspeed);

digitalWrite(2,LOW);

delay(startspeed);

}

/\*

\* Set all green LEDs on

\*/

void greenSmile() {

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(6,HIGH);

digitalWrite(12,HIGH);

digitalWrite(A4,HIGH);

digitalWrite(2,LOW);

digitalWrite(5,LOW);

digitalWrite(10,LOW);

digitalWrite(11,LOW);

digitalWrite(A5,LOW);

}

/\*

\* Set all red LEDs on

\*/

void redSmile() {

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(6,LOW);

digitalWrite(12,LOW);

digitalWrite(A4,LOW);

digitalWrite(2,HIGH);

digitalWrite(5,HIGH);

digitalWrite(10,HIGH);

digitalWrite(11,HIGH);

digitalWrite(A5,HIGH);

}

/\*

\* Set all LEDs off

\*/

void offSmile() {

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(6,LOW);

digitalWrite(12,LOW);

digitalWrite(A4,LOW);

digitalWrite(2,LOW);

digitalWrite(5,LOW);

digitalWrite(10,LOW);

digitalWrite(11,LOW);

digitalWrite(A5,LOW);

}

/\*

\* Signal to short distance

\* ========================

\*

\* Params:

\* -------

\* level: "loudness" of signaling -> 0 (off), 1 (low),2 (medium) or 3 (high)

\*

\*/

void emergency(int level) {

switch (level) {

case 0: // off

greenSmile();

break;

case 1: // low

for (int i = 1; i <= tone\_repeat; i++) {

tone (pin\_buzzer, tone\_low, time\_low);

redSmile();

delay (time\_low);

offSmile();

delay (time\_low);

}

break;

case 2: // medium

for (int i = 1; i <= tone\_repeat; i++) {

tone (pin\_buzzer, tone\_medium, time\_medium);

redSmile();

delay (time\_medium);

offSmile();

delay (time\_medium);

}

break;

case 3: // high

for (int i = 1; i <= tone\_repeat; i++) {

tone (pin\_buzzer, tone\_high, time\_high);

redSmile();

delay (time\_high);

offSmile();

delay (time\_high);

}

break;

default: // default = off

greenSmile();

delay (time\_low);

break;

}

}

/\*

\* The main programm loop:

\*

\* - read current distance

\* - compare with distance intervals defined above

\* - call signalling (emergency) routine to set smiling and beeping

\*/

void loop () {

distance = distanceSensor.measureDistanceCm(); // read current distance

Serial.println(distance); // output for debug

// check interval:

if (distance <= dist\_high)

{

emergency(3);

}

else if (distance <= dist\_medium)

{

emergency(2);

}

else if (distance <= dist\_low)

{

emergency(1);

}

else if (distance > dist\_low)

{

emergency(0);

}

delay(100); // wait a bit :-)

}