*Author: Slavu Vlad Constantin*

Date: 10.01.2022

Distance sensor

DMP Project (using Arduino µ-controller)

Table of Contents

[Introduction 2](#_Toc92647570)

[Components and Design 3](#_Toc92647571)

[Source code and Explanations 5](#_Toc92647572)

# Introduction

* *Goal description:* As previously mentioned in the title of this documentation, it can be deduced that the project focuses on the design, implementation, and physical assembling of a device capable of accurately determining distances. At first, this idea might seem simplistic, but it brings different subtasks and challenges which need to be implemented in order to make the device *user friendly* (easy to understand, read and manipulate). Unfortunately, the actual digital length measurement by its own is unsatisfactory for this prototype to be perceived as an easy-to-use tool. This is the reason why additional functionalities and components need to be brought up, as follows:
* LED display on which both the actual value and the measurement unit will be shown.
* Multiple RGB LEDs, arranged in a straight line, will be used to approximate more easily the measured distance even without looking at the already available display. (This feature better serves for presentational purposes than actual practicality.)
* Bluetooth module to ensure an additional communication between the device and user. Current acquired data will be displayed on a Bluetooth connected device which the user owns.
* *Utility:* A distance sensor can be used in numerous occasions from different technical domains. For instance, it needs to be present in building constructions, a domain which frequently requires distance measurements. The sensor also needs to be involved in the production of self-opening doors, because the opening and closing actions need to depend on whether an object gets close to the door. Using a distance sensor, this measurement can be easily delivered. Another domain in which a distance sensor is indispensable is the automotive industry: from a simplistic action, such as parking, to the most complex ones, meaning self-driving vehicles.
* *Summary:* I have tried to design and implement a device which has both, practical and entertaining attributes, by also attaching multiple LEDs to make using experience more interesting. The third attribute that I wanted to find consists in its utilization in the real world and market, previously detailed.

# Components and Design

The sensor will be implemented using Arduino microcontroller, which will act as the brain of the whole device, together with multiple Arduino compatible components. This build strategy implies different steps that need to be followed: after both the whole assembling process and the code writing have finished, the file containing the code needs to be uploaded into the internal instruction memory of the Arduino board. After the completion of these steps, the project magically comes to life.

In the following paragraph, all the necessary components will be presented. As microcontroller I have used the Arduino Mega 2560, due to its extended features compared to Uno or Micro boards. For the actual distance measurement, an ultrasonic transducer (sensor) was the chosen one. For result display and device communication were used two components: LCD controller Hitachi HD44780 and HC-05 Bluetooth module, respectively.

The last subtask that needs to be ticked is the one which presumes multiple aligned RGB LEDs which will approximately indicate the measured value. For this to happen, I have used a LED block containing 6 different LEDs.

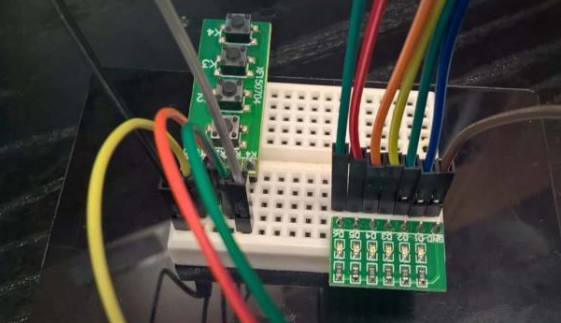
Just for making the whole idea more interesting and user interactive, I have also connected to the board a 4-button block, but only two of them serve real purpose. The first one will be responsible for the behavior of the 6 LEDs, resulting two different modes: one presuming that the LEDs will sequentially turn on as the object gets closer to the ultrasonic sensor and the other one can be describes as the opposite of the first.

The second button is switching between metric units, centimeter and meter in our case, the correct result being shown on the attached display.

The reasons that I have used these specific external components only resume to two major factors: pricing (cheap components) and integration level (they were easily introduces in the final project from both hardware and software perspectives).

|  |  |
| --- | --- |
| A close-up of a computer chip  Description automatically generated with medium confidence | A picture containing electronics  Description automatically generated |
| *Arduino Mega 2560* | *Ultrasonic Sensor* |

|  |  |
| --- | --- |
|  | Arduino with HD44780 based Character LCDs | Martyn Currey |
| *Bluetooth Module* | *LCD controller (display)* |



*Buttons and LEDs block (only K1 and K2 buttons are used)*

A screenshot of a video game

Description automatically generated

*Wokwi Design (unfortunately Bluetooth Module was not integrated within Wokwi Platform)*

A picture containing text

Description automatically generated

*Real Design of the Distance Sensor (Bluetooth Module can be seen in bottom-left corner next to the ultrasonic sensor)*

# Source code and Explanations

For better explanation and understanding, the main code will be divided into subparts:

1. Within the ***“setup loop”*** the whole initialization process is performed as follows: input and output definition for “trigger” and “echo” signals of the ultrasonic distance sensor. Then, the two buttons need to be initialized and marked as being “unpressed” (set LOW value to unpressed state). We then attach one interrupt for each of these buttons, specifying the function needed to be executed. Then, we initialize the lcd with its numbers of lines and columns, the LCD block, which is connected to PORT A, and lastly, the two serials, one for the serial monitor attached to the Arduino, and the other one for the Bluetooth Module.

Text

Description automatically generated

*Setup Loop*

1. ***“Interrupt functions”***: the first one, called “modeOne” is responsible for switching between the two LED modes, while the second one, “modeTwo” chooses the current measurement unit, toggling between “cm” and “m”.

Graphical user interface, text, application, email

Description automatically generated

1. ***“Main Loop (Infinite Loop)”***. First step is to read the data received from the ultrasonic sensor. The result, together with other variables and current states, are printed on the serial port of the Bluetooth Module.

The following lines of code were written to omit some unnecessary lcd clear function calls. Whenever the last measured value coincides with its previous one, the lcd is not cleared because the same display value needs to be shown. Otherwise, we clear it and rewrite the measured value. Then, refreshing LEDs status process comes next. This part compares the computed distance (in cm) with multiples of a constant distance. Based on these sequential comparisons, the status of the LEDs gets computed and then transferred to PORT A.

Source Code

#include <LiquidCrystal.h>

#include<math.h>

#include <stdlib.h>

#define DISTANCE\_UNIT 7

//init the LCD

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

//for the ultrasonic sensor

const int trigger = A5;

const int echo = A4;

int currentLedMode = 0;

volatile int currentUnitMode = 0;

//chosen values of leds based on which mode we are currently in

int chosen\_high = HIGH;

int chosen\_low = LOW;

int chosenUnitMode = 0;

//other variables for computing the distance in cm

int duration = 0;

int previous\_distance = 0;

int distance = 0;

float distanceInMeters = 0.0f;

float previousDistanceInMeters = 0.0f;

char buff[33];

//led status of the LED module

unsigned char LED\_status = 0;

void setup()

{

//configure the data pins for the ultrasonic sensor

pinMode(trigger , OUTPUT);

pinMode(echo , INPUT);

//configure digital pins as inputs (the buttons responsable for switching between lighting modes)

pinMode(20 , INPUT);

pinMode(21 , INPUT);

//initialize the buttons as being "unpressed"

digitalWrite(20, LOW);

digitalWrite(21, LOW);

//attach the interrupts to the buttons

attachInterrupt(digitalPinToInterrupt(20), modeOne, RISING);

attachInterrupt(digitalPinToInterrupt(21), modeTwo, RISING);

// Sets the no. of rows and columns of the LCD

lcd.begin(16, 2);

// activate PORTA, as output (for LED module)

DDRA = 0b11111111;

Serial.begin(9600); //for PC

Serial1.begin(9600); // Serial 1 interface for Bluetooth module

}

void loop()

{

//read data and compute distance

digitalWrite(trigger , HIGH);

delay(200);

digitalWrite(trigger , LOW);

previous\_distance = distance;

duration = pulseIn(echo , HIGH);

distance = (duration / 2) / 28.5 ;

previousDistanceInMeters = distanceInMeters;

distanceInMeters = (float) distance / 100.0f;

distanceInMeters = floor(10000 \* distanceInMeters) / 10000;

//write all the data into Serial port of the Bluetooth Module

if (Serial1.available()) {//send to Bluetooth and write the info there

Serial1.write("Distance: ");

Serial1.write(itoa(distance, buff, 10));

Serial1.write(" cm. ");

Serial1.write("Duration: ");

Serial1.write(itoa(duration, buff, 10));

Serial1.write(". ");

Serial1.write(itoa(currentUnitMode, buff, 10));

Serial1.write(" ");

Serial1.write(itoa(currentLedMode, buff, 10));

Serial1.write("\n");

}

//omit some unnecessary lcd clear function calls

if (previous\_distance != distance || previousDistanceInMeters != distanceInMeters)

lcd.clear();

lcd.setCursor(5, 0);

if (chosenUnitMode == 0) {

//print in CM

lcd.print(distance);

lcd.print(" cm");

}

else {

//print in M

lcd.print(distanceInMeters);

lcd.print(" m");

}

//refresh the led status

LED\_status = 0;

//biuld up current led status and transfer it to PORTA

if ( distance <= 1 \* DISTANCE\_UNIT )

{

LED\_status |= (chosen\_high << 5);

}

else

{

LED\_status |= (chosen\_low << 5);

}

if ( distance <= 2 \* DISTANCE\_UNIT )

{

LED\_status |= (chosen\_high << 4);

}

else

{

LED\_status |= (chosen\_low << 4);

}

if ( distance <= 3 \* DISTANCE\_UNIT )

{

LED\_status |= (chosen\_high << 3);

}

else

{

LED\_status |= (chosen\_low << 3);

}

if ( distance <= 4 \* DISTANCE\_UNIT )

{

LED\_status |= (chosen\_high << 2);

}

else

{

LED\_status |= (chosen\_low << 2);

}

if ( distance <= 5 \* DISTANCE\_UNIT )

{

LED\_status |= (chosen\_high << 1);

}

else

{

LED\_status |= (chosen\_low << 1);

}

if ( distance <= 6 \* DISTANCE\_UNIT )

{

LED\_status |= chosen\_high;

}

else

{

LED\_status |= chosen\_low;

}

PORTA = LED\_status;

}

void modeOne () {

if (currentLedMode == 0) {

//the lights turn on sequentially as the object gets closer relative to the ultrasonic sensor

chosen\_high = HIGH;

chosen\_low = LOW;

currentLedMode = 1;

}

else {

//the lights shut down sequentially as the object gets closer relative to the ultrasonic sensor

chosen\_high = LOW;

chosen\_low = HIGH;

currentLedMode = 0;

}

}

void modeTwo () {

if (currentUnitMode == 0) {

chosenUnitMode = 1;

currentUnitMode = 1;

}

else if (currentUnitMode == 1) {

chosenUnitMode = 0;

currentUnitMode = 0;

}

}