سخت افزار کنترل فاصله برای ربات شیلد به کمک ماژول SRF04

سجاد باغستانی لینک پروژه

مقدمه

در این پروژه قصد داریم سخت افزاری برای اندازه گیری اجسام نزدیک به ربات شیلد به کمک ماژول التراسونیک SRF04 طراحی کنیم. این پروژه شامل دو بخش طراحی سخت افزار شامل انتخاب قطعات، نحوه ی اتصال، جانمایی و طراحی نرم افزار شامل برنامه نویسی های مربوطه، تست و اجرا می باشد. به دلیل تغییرات ساده تر در بخش نرم افزار مقل جابجایی پین ها به صورت نرم افزار و حساسیت بیشتر سخت افزار در ادامه ابتدا سخت افزار پروژه طراحی شده و سپس به سراغ نرم افزار می رویم.

طراحی سخت افزار

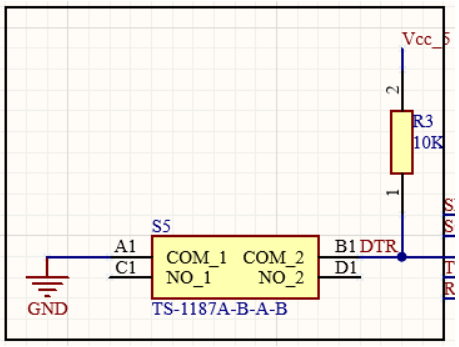
برای بخش سخت افزاری پروژه شامل مدارات و جانمایی از نرم افزار Altium Designer و همچنین برای افزایش سرعت در افزودن قطعات به پروژه از افزونه Altium Library Loader استفاده شده است.

انتخاب میکروکنترلر

به جهت هزینه ی کمتر، امکان مونتاژ به کمک ربات از نسخه ی SMD میکروکنترلر ATMEGA328P-AU اسفاده شده است.

مدار ریست

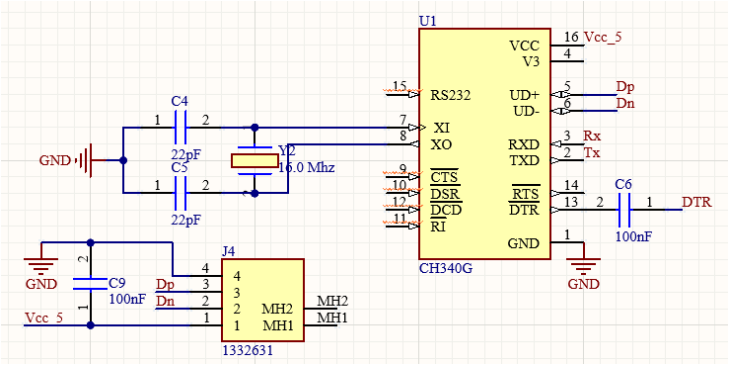
از آنجایی که ریست میکروکنترلر به صورت Active-Low صورت می گیرد. مدار زیر شامل یک push-button و یک مقامت pull-up می باشد، در نظر گرفته شده است.



شکل ۱- مدار reset

مدار پروگرمر

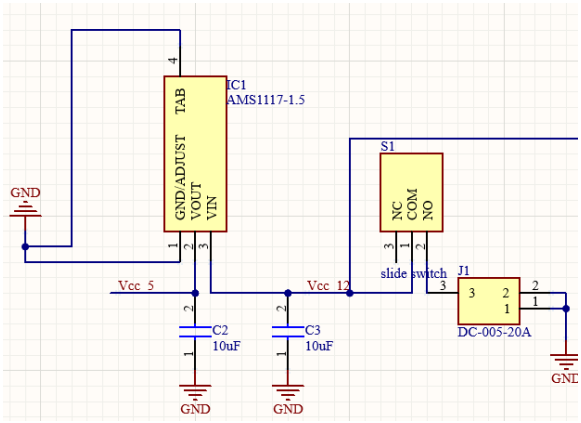
برای پروگرم کردن میکروکنترلر از یک مبدل USB به Serial(UART) آی سی CH340 و یک متصل کننده ی USB مشابه شکل زیر در نظر گرفته شده است که امکان پروگرم کردن میکروکنترلر را به کمک USB و Arduino IDE فراهم می کند.



شکل ۲- مدار پروگرمر

مدار تغذیه

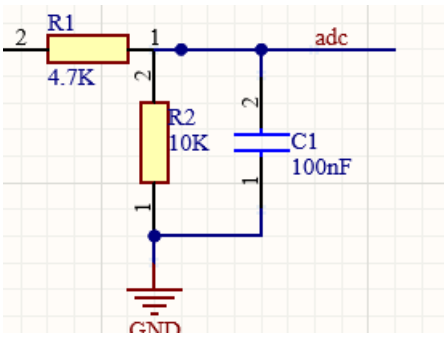
برای تغذیه ی سخت افزار یک باتری ۱۲ ولتی در نظر گرفته شده است و به دلیل ۵ ولت بودن تغذیه ی بقیه ی قطعات اکتیو شامل میکروکنترلر و ... نیاز به یک رگولاتور ۱۲ به ۵ ولت داریم که از AMS1117 به همراه خازن های جانبی که برای کاهش نویز منبع و جلوگیری از اثرات مخرب افت ناگهانی ولتاژ در ورودی و حفظ پایداری و کاهش ریپل ولتاژ در خروجی مطابق شکل ۳ در نظر گرفته شده است.



شکل ۳- مدار تغذیه

مدار سطح ولتاژ باتری

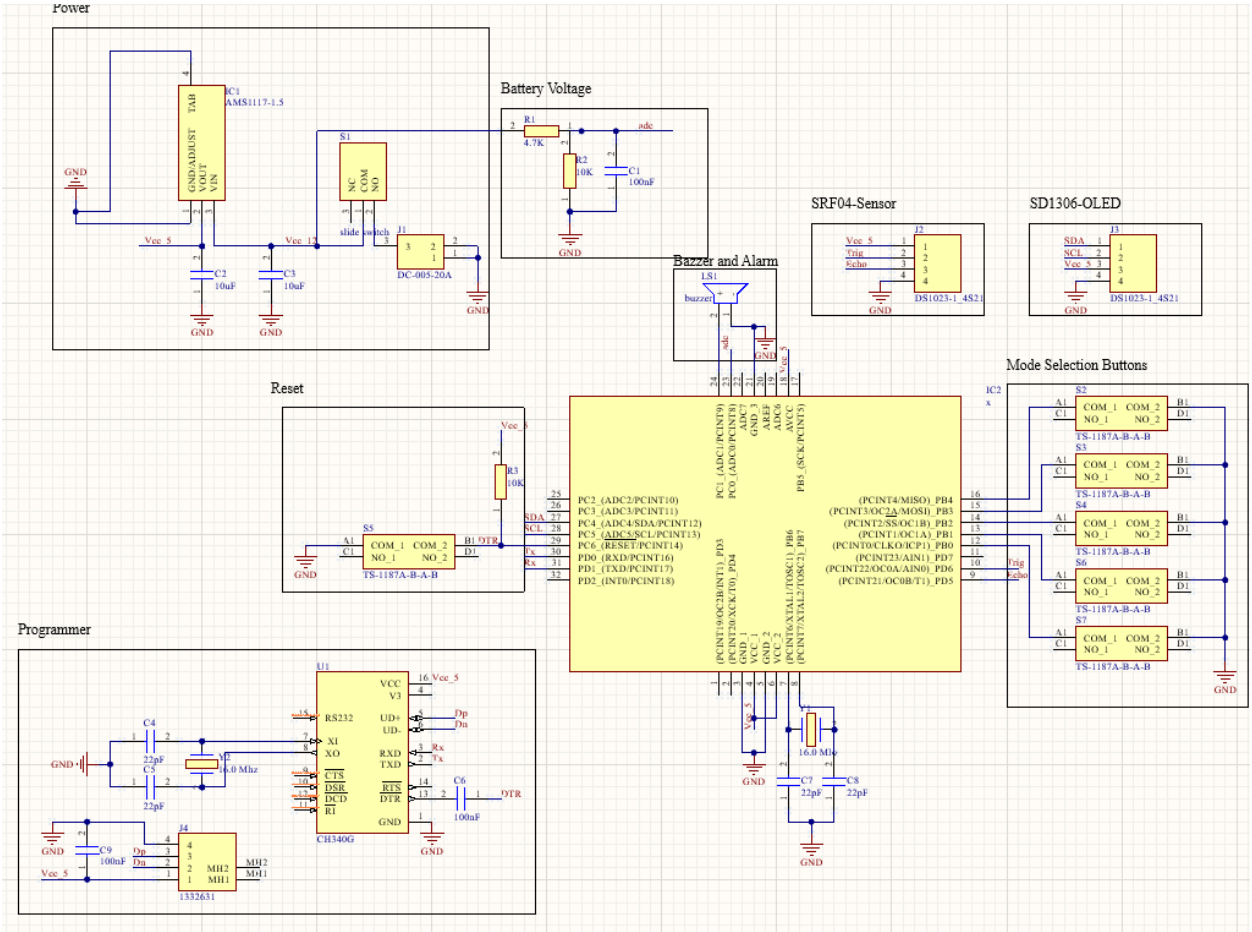
به جهت خوانش سطح ولتاژ باتری در نظر داریم که از مبدل آنالوگ به دیجیتال میکروکنترلر استفاده کنیم اما به دلیل وجود اختلاف ۱۲ ولت باتری و ۵ ولت بودن سطح ولتاژ مبدل ADC ابتدا به کمک یک تقسیم مقامت مطابق شکل ۴ سطح ولتاژ را تغییر داده و سپس به کمک نرم افزار آنرا جهت نمایش به کاربر اصلاح می کنیم.



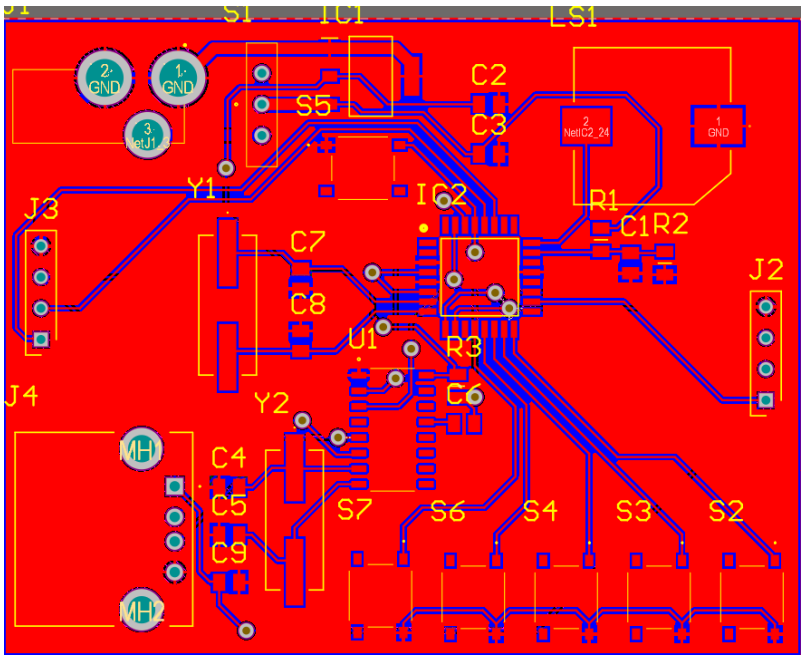
شکل ۴- خوانش ولتاژ باتری

سایر سخت افزار

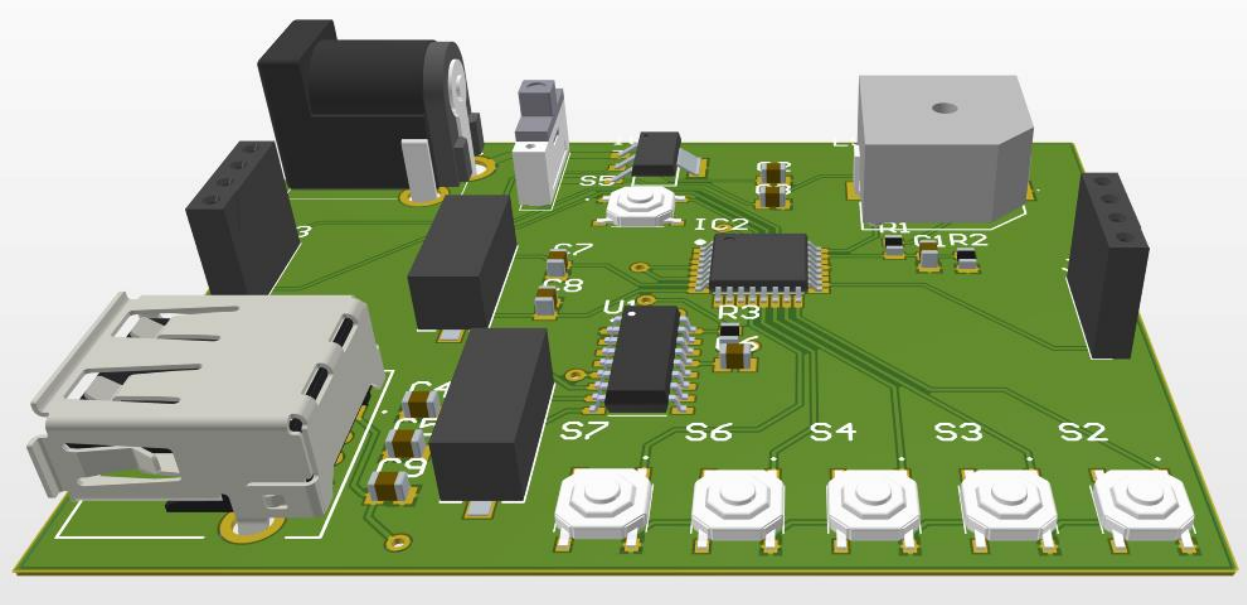
در ادامه مطابق خواسته ی مساله از ۵ عدد push-button با pull-up داخلی جهت تغییر مد سخت افزار،یک عدد Buzzer برای هشدار که از مدل Active با جریان کم(بدون نیاز به مدار جانبی)، یک عدد OLED LCD SSD1306 که با رابط I2C به میکروکنترلر ارتباط می گیرد و سنسور SRF04 التراسونیک جهت اندازه گیری فاصله استفاده شده است. شماتیک سخت افزار، جانمایی و شکل سه بعدی مدار به ترتیب در شکل های ۵-۷ قابل مشاهده است.



شکل ۵- شماتیک سخت افزار



شکل ۶- جانمایی ۲بعدی



شکل ۷-جانمایی ۳بعدی

طراحی نرم افزار

برای بخش نرم افزار از Arduino IDE 1.18.19 استفاده شده است. مطابق خواسته ی مساله ابتدا یک کتابخانه به منظور خواندن دیتای سنسور و متد های مختلف آماده شده و سپس به کمک توابع مختلف سخت افزار کنترل می شود.

کتابخانه ی سنسور SRF04

برای ساختن کتابخانه ابتدا دو فایل با پسوند های h. برای تعریف کلاس مربوطه و متغیر ها و cpp. برای عملکرد کلاس شامل نحوه ی ساختن شئ و عمل کردن توابع مربوطه ساخته می شود.

فایل h. مطابق زیر تعریف شده است که شامل include guards و افزودن آردوئنو و تعریف کلاس می باشد.

```
#ifndef ULTRASONICSRF04_H
#define ULTRASONICSRF04_H

#include <Arduino.h>

class UltrasonicSRF04 {
  private:
    uint8_t trigPin;
    uint8_t echoPin;

  public:
    UltrasonicSRF04(uint8_t trig, uint8_t echo);
    float getDistance(String unit = "cm");
    bool inBound(float Bound);
    float getFilteredDistance();
};

#endif

```

سپس به پیاده سازی نحوه ی عملکرد سنسور در فایل cpp. می پردازیم. در ابتدا نیاز به constructor داریم که هنگام صدا زدن شی سنسور را مطابق با تعریف انجام شده که تنظیم کردن پین های ارسال و دریافت است داریم. سپس، مطابق دیتاشیت سنسور التراسونیک برای اندازه گیری فاصله نیاز به ارسال یک پالس و اندازه گیری زمان پالس دریافتی می باشد که باتوجه به زمان پالس دریافتی و سرعت صوت فاصله به صورت زمان دریافتی تقسیم بر ۵۸ و یا زمان دریافتی تقسیم بر ۱۴۶به ترتیب فاصله برحسب سانتی متر یا اینچ مشخص می گردد که در تابع getDistance پیاده شده است. در متد دیگر خواسته شده است که در صورت وجود جسم در بازه ی مشخص شده یک متغیر trueو falseبرگردانده شده که در تابع inbound پیاده شده است و در نهایت برای افزایش دقت سنسور در مواردی که سنسور روی جسم متحرک می باشد از متد میانگین گیری در تابع getFilteredDistance پیاده شده است. در این متد به دلیل ثابت بودن فاصله و متغیر بودن نویز هنگام میانگین گیری مقدار نویز به طور متوسط بر تعداد دفعات اندازه گیری تقسیم شده اما سیگنال اصلی ثابت خواهد ماند.

```cpp
#include "UltrasonicSRF04.h"

UltrasonicSRF04::UltrasonicSRF04(uint8_t trig, uint8_t echo) {
  trigPin = trig;
  echoPin = echo;

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

float UltrasonicSRF04::getDistance(String unit) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH, 30000);

  if (duration == 0) return -1;

  if (unit == "inch") {
    return duration / 148.0;
  } else {
    return duration / 58.0;
  }
}

bool UltrasonicSRF04::inBound(float bound) {
    return getDistance() <= bound;
}

float UltrasonicSRF04::getFilteredDistance(){
      float distance = 0;
      int trueSample = 0;
      for(int i=0; i < 5; i++){
                float newDistance = getDistance();
                if(newDistance >= 0){
                          distance = distance + newDistance;
                          trueSample++;
      }
                delay(100);
      }
                if(trueSample == 0){
                          return -1;
                } else{
                          return distance / trueSample;
                }


}

```

پیکربندی فایل اصلی

پس از نهایی کردن کتابخانه ی مربوط به راه اندازی و استفاده ی سنسور التراسونیک نیاز به توسعه ی پیکره ی اصلی نرم افزار به کمک Arduino IDE در فایل ino. می باشد که در ادامه مرحله به مرحله به توضیح آن پرداخته خواهد شد.

افزودن کتابخانه ها

به منظور استفاده از نمایشگر OLED SSD1306، کنترل سنسور التراسونیک و ارتباط lcd و میکروکنترلر نیاز به کتابخانه ها مربوطه می باشد که مطابق با زیر تعریف شده اند. (از آنجایی کتابخانه ی سنسور التراسونیک به منظور تست در وبسایت wokwi به صورت محلی تعریف می شود کتابخانه ی مربوطه بین "" نیز تعریف شده است.)

```
1. #include <Wire.h>
2. #include <Adafruit_GFX.h>
3. #include <Adafruit_SSD1306.h>
4. #include <UltrasonicSRF04.h>
5. //#include "UltrasonicSRF04.h"
6.
```

تعریف ها و متغیرها

در این بخش به منظور حفظ پیوستگی کد نویسی و تغییرات ثانویه ی ساده تر متغیر ها پیش پردازش ها و متغیر های ثابت مطابق با زیر انجام می شود. که شامل طول و عرض lcd، پین ورودی آنالوگ جهت خوانش سطح ولتاژ باتری، مقادیر ثابت مقاومت برای تغییر سطح ولتاژ از ۱۲ به ۵ ولت و مد های مختلف حالت می باشد.

```
 1. #define SCREEN_WIDTH 128
 2. #define SCREEN_HEIGHT 64
 3. #define BUZZER_PIN 24
 4.
 5. const int BATTERY_PIN =  A0;
 6. const float R1 = 4700.0;
 7. const float R2 = 10000.0;
 8. const int MODE_BUTTONS[] = {12, 13, 14, 15, 16};
 9. const int MODE_NUMBERS = 5
10.
```

;

ساخت شئ

نیاز به ساخت شئ از کلاس مربوطه و سپس کنترل آنها می باشد که مطابق با OLED LCDبرای استفاده از سنسور التراسونیک و زیر انجام شده است.

```
1. Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
2. UltrasonicSRF04 sensor(10, 9);
3.
```

تنظیمات و راه اندازی OLED LCD

به منظور راه اندازی اولیه lcd ابتدا نیاز به تعریف lcd می باشد که مطابق زیر انجام شده است و سپس در مراحل جداگانه جهت نمایش مقادیر مختلف کنترل خواهد شد. ابتدا ارتباط میکروکنترلر با lcd چک شده و در صورت عدم ارتباط یه پیغام به ترمینال مبنی بر عدم شناسایی lcd و در صورت شناسای پیغامی را مبنی بر شناسایی lcd روی خود lcd نمایش می دهد.

```
1. void setOled(){
2.   if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)){
3.     Serial.println("OLED LCD Not Found");
4.     while(true);}
5.
6.     display.clearDisplay();
7.     display.setTextSize(1);
8.     display.setTextColor(SSD1306_WHITE);
9.     display.setCursor(0,0);
10.    display.println("HELLO, OLED LCD Found");
11.    display.display();
12. }
13.
```

تنظیمات خواندن باتری و نمایش سطح ولتاژ

مطابق با خواسته ی مساله مبنی بر نمایش سطح ولتاژ باتری ۱۲ ولت که پس از تقسیم مقاومت، به سطح ۵ ولت رسیده است، از طریق پین آنالوگ A0 خوانده شده و مقدار آن از سطح ۵ ولت مشخص شده و مجددا به سطح ۱۲ولت در تابع batteryStatus تبدیل می شود. این مقدار برگردانده شده به کمک تابع showBatteryStatus در صورت نیاز(مد پیشفرض صفر) روی نمایشگر نمایش داده می شود.

```
1. float batteryStatus(){
2.   int adc_val = analogRead(BATTERY_PIN);
3.   float vout = adc_val * 5.0 / 1023;
4.   float vout_bat = vout * (R1 + R2)/R2;
5.   return vout_bat;
6. }
7.
8. void showBatteryStatus(){
9.  float battery = batteryStatus();
10.    display.clearDisplay();
11.    display.setTextSize(1);
12.    display.setTextColor(SSD1306_WHITE);
13.    display.setCursor(0,0);
14.    display.print("Battery: ");
15.    display.print(battery);
16.    display.println(" V");
17.    display.display();
18.
19. }
20.
```

نمایش فاصله

جهت نمایش فاصله از تابع showDistance استفاده شده است که با دریافت مقدار فاصله از جسم و واحد اندازه گیری، آنرا روی نمایشگر، نمایش می دهد.

```
void showDistance(float distance, String unit) {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.print("Distance: ");
  display.print(distance);
  display.print(" ");
  display.println(unit);
  display.display();
}

```

تنظیمات بازر

تنظیمات بازر درون تابع setBuzzer و نحوه ی به صدا درآمدن آن در تابع alarm مطابق زیر توسعه داده شده است.

```
void setBuzzer(){
 pinMode(BUZZER_PIN, OUTPUT);
 digitalWrite(BUZZER_PIN, LOW);
}

void alarm(bool cond){
  if (cond){
    digitalWrite(BUZZER_PIN, HIGH);}else{
      digitalWrite(BUZZER_PIN, LOW);}
}

```

تغییر Mode

طبق خواسته ی مساله ۵ کلید push-button جهت تغییر مد کاربری ربات تعبیه شده است. تابع setButtons تنظیمات مربوط به کلید ها به صورت push-pull را انجام داده تابع buttonStatus وضعیت فشرده شدن کلید را بررسی کرده و برمیگرداند و درنهایت تابع selectMode وضعیت ربات را مشخص می کند. لازم به ذکر است متغیر mode برای حفظ مد در نظر گرفته شده است که به صورت پیشفرض صفر می باشد. مد های در نظر گرفته شده به صورت زیر می باشد:

```
0 -> Battery Status
1 -> Show Distance(cm)
2 - > Show Distance(inch)
3 -> Show Filtered Distance
4 -> Show Battery Status and Show Distance(cm)
```

```
void setButtons(){
  for(int i = 0; i< MODE_NUMBERS; i++){
    pinMode(MODE_BUTTONS[i], INPUT_PULLUP);
    }
}

bool buttonStatus(int ind){
  return digitalRead(MODE_BUTTONS[ind]);
}
int mode = 0;
void selectMode(){
  for (int i=0; i<5; i++){
    if (!buttonStatus(i)){
      mode = i;
      break;
      }
    }
}
void showCurrentMode(){
   switch(mode){
    case 0:
      showBatteryStatus();
      break;
    case 1:
      showDistance(sensor.getDistance(), "cm");
      break;
    case 2:
      showDistance(sensor.getDistance("inch"), "inch");
      break;
    case 3:
      showDistance(sensor.getFilteredDistance(), "cm");
      break;
    case 4:
      showBatteryStatus();
      delay(1000);
      showDistance(sensor.getDistance(),"cm");
    }
}

```

تنظیمات نهایی

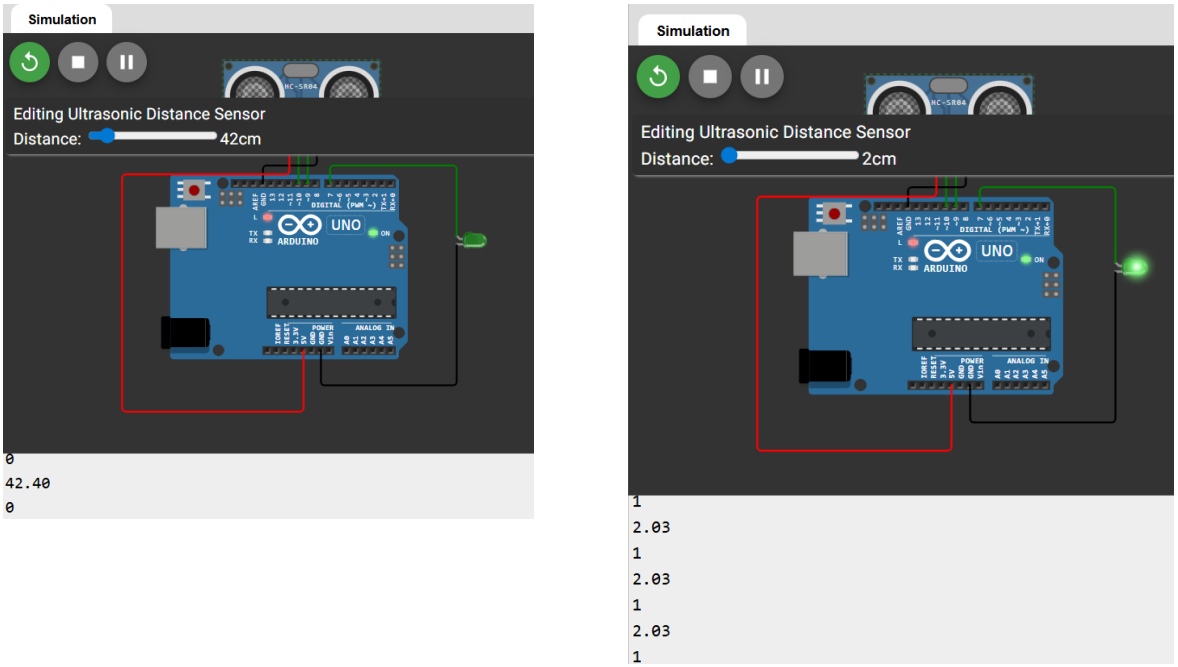
درنهایت توابعی که در بالا به صورت کاملا ماژولار طراحی شد را در دو تابع setup و loop صدا زده تا ابتدا تنظیمات اولیه ی آنها انجام شده و سپس به ترتیب گفته شده اجرا شوند.

```
void setup() {
  Serial.begin(9600);
  setOled();
  setBuzzer();
  setButtons();
}

void loop() {
  selectMode();
  showCurrentMode();
  bool isInBound = sensor.inBound(30.0);
  alarm(isInBound);
  delay(500);
}
```

تست نهایی

به منظور آزمایش نهایی قسمت بازر به کمک یک led در وبسایت wokwi تست شد که در تصویر ۸ نتایج آن قابل مشاهده است. به هنگام فاصله ی بیشتر از ۳۰ سانتی متر led در وضعیت خاموش(۸-الف) و فواصل کمتر از ۳۰ سانتی متر در حالت روشن(۸-ب) قرار دارد.



شکل ۷- تست بازر

تمامی فایل های مربوط به طراحی سخت افزار، نرم افزار و تست نهایی در لینک گیت هاب پروژه موجود است.