**BLIND DISTANCE NOTIFIER**

**Short Description**

The device helps the blind people to interact with objects around them, notifying them when an object is in the close range. The device is able to measure precise distances from about 2m away.

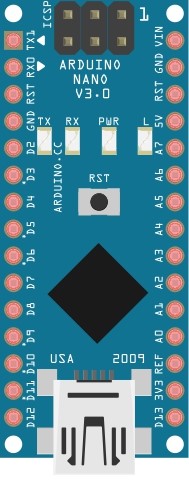
Once an object is found on that range, the device activates the notification state, turning on one or two vibration motors depending on where the obstacle is. It features two UltraSonic sensors, one for the right and one for the left.

These sensors send an ultrasonic sound wave and wait for it to come back, measuring the time it took to travel. The brain of the device is the ATmega328P Microcontroller placed in an Arduino Nano 328P Configuration.

The device is battery powered, it features three 18650 Li-Ion Cells connected in series to accomodate the minimum Vin of the Arduino Nano328P board.

**Full description**

**HARDWARE**

**Microcontroller:   
  
**

\*Image generated with the help of Fritzing – open source project prototyping software

The brain of the device is the ATmega 328P microcontroller in an Arduino Nano 328P board. This configuration features an AVR architecture with 32KB of flash memory, where 2KB are used by the bootloader, a 2KB SRAM and a clock speed of 16MHz. It also features an EEPROM of 1KB, but in our project this is not used.   
 **Analog Pins:** The ATmega328P offers 8 Analog I/O Pins,6 of them ideal for PWM output for our Vibration Motors. In our case, we will use two of them, one for the right vibration motor and one for the left one.   
 The values that represent the output of these pins are from 0 to 255 where 255 is 100% duty cycle and 0 is 0% duty cycle. The frequency of the signal on most PWM pins is 490Hz , but on the pins 3,5,6,9,10 and 11 the signal frequency is 980 Hz, so we will use pins 3 and 5 for a better and smooth transition between 0% duty cycle and 100% duty cycle.   
 To set the output duty cycle on these pins we will use the analogWrite() function. It takes two parameters :   
 analogWrite(pin,value)   
 - pin: The pin to write to  
 - value: The duty cycle , between 0 (always off) and 255 (always on) .  
  
 Due some interractions with the millis() and delay() functions, the pins 5 and 6 will have a “higher-than-expected” duty cycle, but this will cause us trouble only in very low duty cycle settings ( for example: 0-10 of 255) where our motors will not move at all. In conclusion we can use the pins 3 and 5 to control the vibration motors.

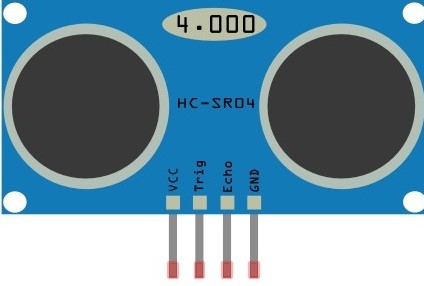
**Digital Pins:** The ATmega328P offers 22 digital I/O pins which will be ideal for our sensor setup. For these pins we have only two states, LOW ( for 0 V ) and HIGH ( for 5V ) . For some boards the HIGH value could be 3.3V, but for our Arduino Nano328P the HIGH value is 5V.  
 The digital pins are D0 to D13 and A0 to A7. The analog pins could be used as digital as the native digital ones are.  
 To set the state of this pins we will use the digitalWrite function. It takes two parameters:  
 digitalWrite(pin,state)

-pin: the pin to write to;  
 -state: the state of the pin, HIGH (5V) or LOW (0V);

**Power:** The Arduino Nano328P works with an imput voltage from 7 to 12 V and the microcontroller works with 5V . The stable 5V is granted bt the AMS1117 Voltage Regulator. Also the AMS1117 gives us the 5V (VCC) pin on Arduino Nano328P which will be used by our sensors.  
 The power consumption of the Nano328P is around 19mA in use. This could be improved with a “sleep function” , but for our purpose this is not the case since the Nano328P is processing data continously.

**Size and weight:** The Arduino Nano328P is a small and powerfull device for small devices like this one, so this is the best choice for a prototype. The overall size of the board is 18x45mm and it weights only 7g.

**Sensors:**



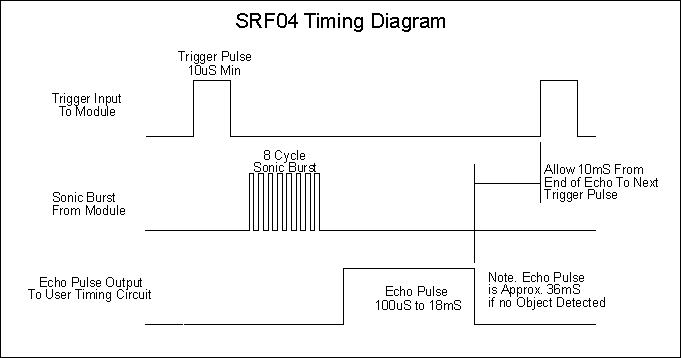
\*Image generated with the help of Fritzing – open source project prototyping software

To sense the distance and the objects that are facing our blind person, we need to use some sensors. The one we will use in this device is the HC-SR04 UltraSonic Ranging Module.

The HC-SR04 is a cheap,easy to use and powerful sensor that could sense ranges between 2-400cm with the accuracy of +/- 3mm. In our case we will need to determine the estimated distance, as 3mm error will never affect our device’s working state.

The basic working of the HC-SR04 is the following:   
 - HIGH signal from our I/O for at least 10uS  
 - The module sends eight 40kHz pulses and detects whether there is a pulse signal back.   
 - If a pulse comes back, through high level, time of high output is the time from sending ultrasonic to returning one.

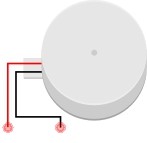
This time helps us determine the distance between the sensor and the object that reflected the ultrasonic signal.

**Timing:** The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .We can calculate the range through the time interval between sending trigger signal and receiving echo signal.   
 

**Breakout Board wiring:** The breakout board that contains the sensors has 4 pins, VCC, GND , TRIG , ECHO.

VCC – we will supply 5V here  
 GND – to GND of the Nano328P  
 TRIG – as we will use two sensors, the TRIG1 on D9 and TRIG2 on D7  
 ECHO – as we will use two sensors, the ECHO1 on D8 and ECHO2 on D6

**Vibration Motors**



\*Image generated with the help of Fritzing – open source project prototyping software

In order to inform the user of the device that there is an obstacle, we need to use any non-visual stimuli. In this case a vibration motor could be the best choice, being fairly quiet but effective.

The vibration motor is based on a normal motor, but at the shaft is attached a non-balanced weight , as when it spins the motor tends to vibrate due centrifugal force induced by the weight.

The motor we will use is rated at 12.000 RPM (rotations per minute) at 5V which will be very noticeable in any condition. Also, it has different current draws and RPMs depending on the supply voltaje, as seen in the table:

|  |  |  |
| --- | --- | --- |
| **Supply voltage** | **Current draw** | **RPM** |
| 5Vcc (100% duty cycle) | 80mA | 12000 |
| 4Vcc (80% duty cycle) | 64mA | 9600 |
| 3Vcc (60% duty cycle) | 48mA | 7200 |
| 2Vcc (40% duty cycle) | 32mA | 4800 |

In this case, for power consumption optimisation we will use only the range [30%,80%] duty cycle, where 30% is “long distance to object” and 80% is “very short distance to object” . The motor is a simple DC one, it does not need a driver and the board can face that draw for short period.

**BATTERY PACK**



The battery pack that we will use to power up the device is made from Samsung ICR18650 Li-Ion Cells. The reason for choosing the 18650 cells from Samsung is that we need a high quality, high power and long lasting battery pack. In this case, the Samsung ICR18650 offers us a 2200mAh capacity per cell at a nominal voltage of 3.6V.

The cells are rated at 4.2V fully charged and 2.75V fully discharged, having a maximul discharge current of 4400mAh. Of course, we will only need 100mAh for our device. This will define a very good battery life of our device over time. The Samsung cells offers us the posibility to charge them at “Standard Rate” which is 1100mAh Charging capacity, or “Rapid Rate” with 2200mAh Charging capacity. For this project we will need to use the Standard Rate, meaning that the device will be fully charged in only 3 hours.

The dimensions of the cells are Diameter(max.) : Φ18.4 mm and Height(max.) : 65.0 mm,and our battery pack will be made from 3 Series Cells.

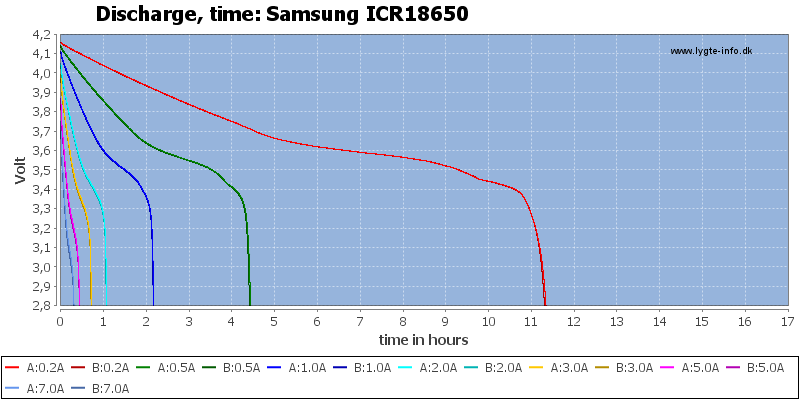
The biggest advantage of our choice ( the Li-Ion 18650 cells ) is that it provides a very good charge cycle durability ( 400-1200 cycles ) , making our device’s battery pack lifetime longer than the others. This could make the difference for a blind person that do not want to pay attention about the capacity loss of the device in time, ensuring an easy,long lasting experience .

Also, being a wearable device it must be very stable and safe. The 18650 Li-Ion cells from Samsung are one of the safest cells on the market, having a very good insulation, high quality metalic contacts and very good heat distribution.

The mechanical and safety characteristics can be seen in the next table:

|  |  |  |
| --- | --- | --- |
| **TEST** | **TEST METHOD** | **CRITERIA** |
| Drop Test | Cell(as of shipment or full charged) drop onto the oak-board (thickness:≥30mm) from 4feet height at a random direction 6 times | No leakage |
| Vibration Test | Cell(as of shipment) is vibrated along 2 mutually perpendicular axes with total excursion of 1.6mm and with frequency cycling between 10Hz and 55Hz by 1Hz/min. | No leakage |
| Overcharge Test | To charge the standard charged cell with 12V and 2200mA at 25℃ for 2.5 hours | No fire, and no explosion |
| External Short-circuit Test | To short-circuit the standard charged cell by connecting positive and negative terminal by less than 50mΩ wire for 3 hours | No fire, and no explosion |
| Reverse Charge Test | Test method: To charge reversely the standard charged cell with charge current 2200mAh for 2.5 hours | No fire, and no explosion |
| Heating Test | To heat the standard charged cell at heating rate of 5℃ per minute up to 130℃ and keep the cell in oven for 60 minutes | No fire, and no explosion |

\* The marked with RED COLOR characteristics are the most important ones. Other tests are not applied in our device as our pack will be protected against drop,vibration. Also the BMS will take care of Overcharge,Reverse Charge or External Short circuit.  
 \*\* The data from the table is collected from Samsung ICR18650 Datasheet

To get a clear image of how the cell behaives, we will look at the discharge,time plot:  
  
   
 \* image courtesy of lygte-info.dk  
  
 As we can see, the battery capacity depends on the amount of current we draw from it. As our device will be drawing around 100mA on normal use and 200mA on high use,including the vibration motors and the electronics, we can expect a battery life of more than 10 hours.

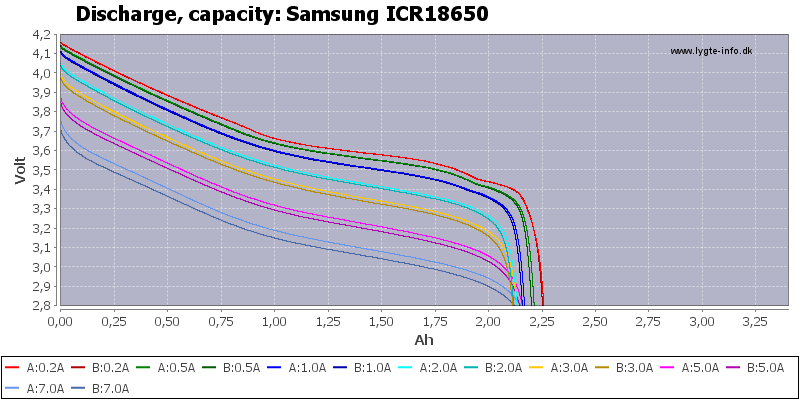
To calculate the power consumtion of our device, we will look in the next table:  
  
\* Maximul ratings available:

|  |  |  |
| --- | --- | --- |
| **Device to power** | **Device usage** | **Device current draw** |
| ATmega328P | 100% | 2.5mA @ 5V |
| CH340 USB Chip | 50% | 5mA @ 50% |
| Arduino Nano LEDs | 100% | 8mA |
| AMS1117 – power loss due heat | 5-10% | 10mA |
| HC-SR04 Ultrasonic Sensor | 100% | 30mA @ 100% |
| Vibration Motors | 80% | 128mA @ 80% |

**TOTAL: 184mA**

Considering that all the devices listed before will be used at full power, the total current draw will be less than **200mA** , this gives us a maximum power consumption of **1W** .

Our battery pack provides us 2200mAh of capacity while the device will request only 200mAh at its maximum working power. This will give us a full 11 Hour battery life.  
  
  
To understand better how the Samsung ICR18650 discharge depending on current draw, we will look at the following plot:



\* image courtesy of lygte-info.dk

We are looking for a linear discharge,capacity plot , but due different factors, this can not be achieved. As our device will use about 0.2A at full power, the red drawing will represent our voltage drop and the capacity.

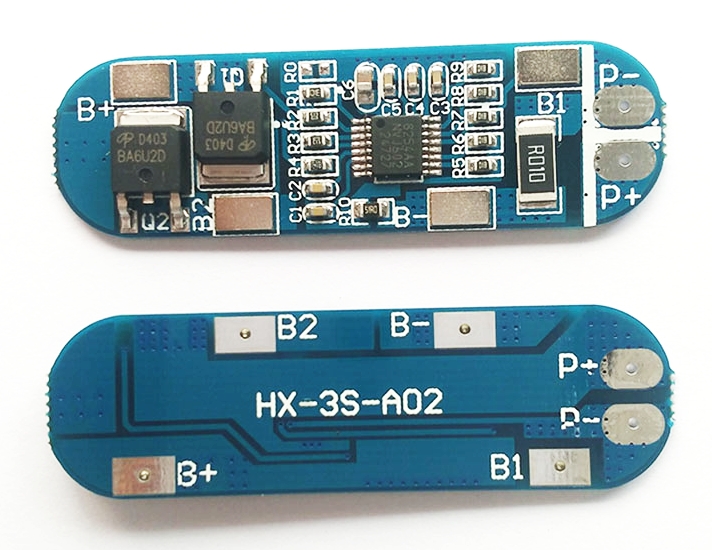
Our battery pack will be made from 3 Series 18650 Cells: The battery pack will have the following characteristics. The package has a 2C characteristic.

|  |  |
| --- | --- |
| Nominal voltage | 10.8V |
| Fully charged voltage | 12.6V |
| Fully discharged voltage | 8.25V |
| Capacity | 2200mAh |
| Maximul charge current | 2200mA |
| Maximum discharge current | 4400mA |
| Working Temperature | -10°C - 150°C |
| Configuration | 3 Cells in series |
| Battery Management System | HX-3S-D02 |

The final dimension of our battery pack will be :

|  |  |
| --- | --- |
| Width | 56 mm |
| Height | 70mm |
| Lenght | 18.3mm |

**BATTERY MANAGEMENT SYSTEM**



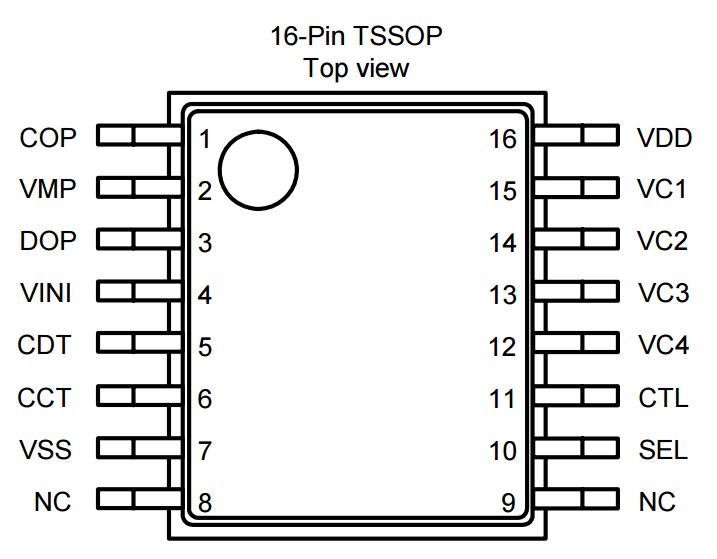
As we will use a battery pack, we need to make sure that our cells are well balanced. Balancing a battery pack means that all the cells are charged to the same level and discharged to the same level. Having a battery management system helps making the battery life better and the cell life longer.

The BMS module is designed for 3 cells connected in series, so we will use 3xSamsung ICR18650 cells with this module.

The module is based on the Seiko Instruments S-8524AA battery protection IC, which is designed to be used with 3S or 4S lithium-ion or lithium-polymer rechargeable batteries.

The IC has a fast and high-accuracy voltage detector and delay circuit. It also provides a posibility to select between 3S or 4S configuration, throught SEL pin, the posibility to select the overcharge voltage,release voltage , overdischarge detection voltage and overdischarge release voltage. Also, it has a three level over-current protection.

The S-8524AA is a very good choice for our device since it has a very low power consumption (only 30uA at +25°C).

The **S-8524AA** has the following pinout

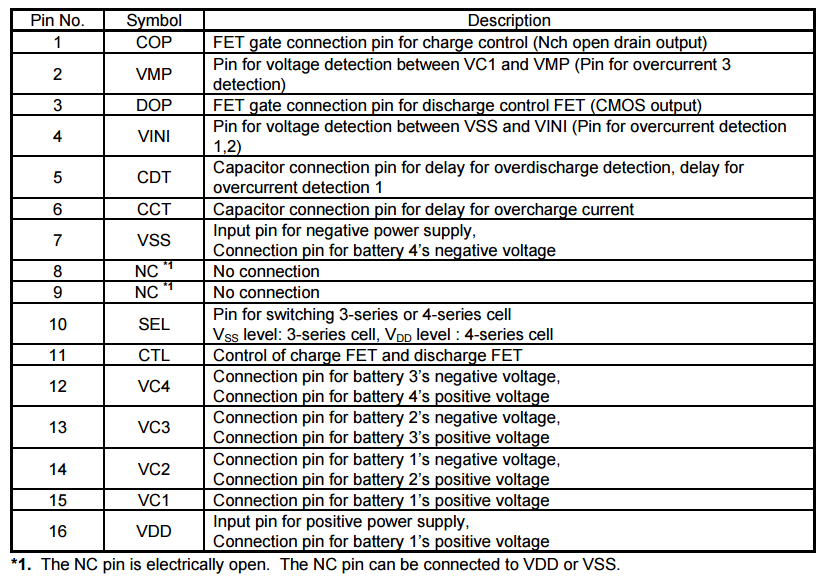
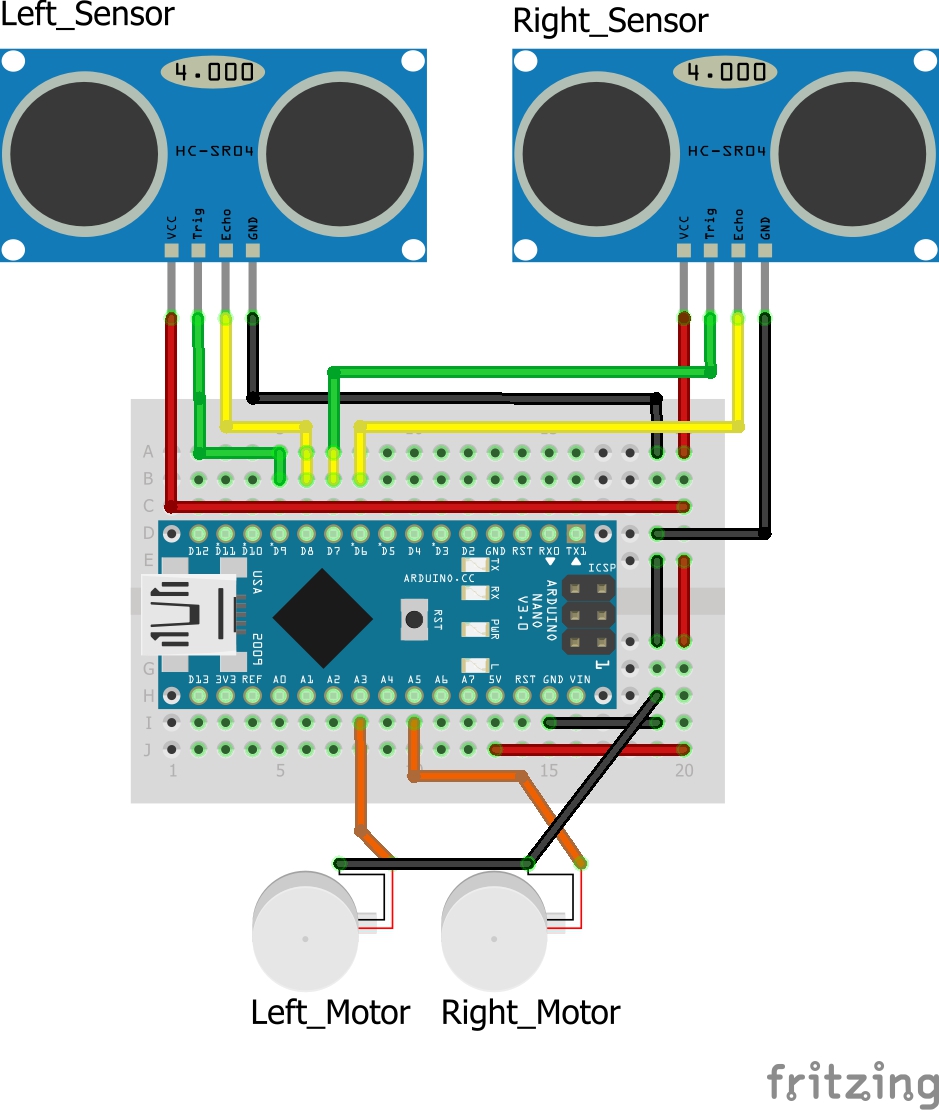


image source: S-8524AA datasheet.

More information about the break-out board can be found in the next table:

|  |  |
| --- | --- |
| **Model** | **S-8524AA** |
| **Size** | 53mm x 16mm x 4mm |
| **Overcharge Voltage Range** | 4.25V – 4.35V +/- 0.05V |
| **Overdischarge Voltage Range** | 2.3V – 3.V +/- 0.05V |
| **Maximum operating current** | 9A |
| **Maximum transient current** | 10-15A |
| **Quiescent current** | < 50uA |
| **Internal Resistance** | < 100mOhm |
| **Working Temperature** | -40°C -> +50°C |
| **Effective life** | > 30000 hours |
| **Short Circuit Protection** | Yes, delayed self recovery |
| **Reverse voltage Protection** | Yes |

**Prototype and testing**

****

\*Image generated with the help of Fritzing – open source project prototyping software

The project started with the prototype breadboard, the final bill of materials is the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Part** | **Qty** | **Shop** | **Price** |
| Arduino Nano328P | 1 | [UNDA Tech](https://magazin.unda.tech/ro/placi-de-dezvoltare/16-placa-de-dezvoltare-arduino-nano-328p.html?search_query=nano&results=4) | 65 RON |
| Ultrasonic Sensor | 2 | [UNDA Tech](https://magazin.unda.tech/ro/senzori/193-hc-sr04-ultrasonic.html?search_query=ultrasonic&results=1) | 16 RON |
| Vibration Motor | 2 | [Optimus Digital](https://www.optimusdigital.ro/motoare-motoare-cu-vibratii/2319-motor-cu-vibraii-in-miniatura-de-4-mm.html?search_query=motor&results=268) | 16 RON |
| 18650 Battery | 3 | [UNDA Tech](https://magazin.unda.tech/ro/acumulatori-baterii-si-suporti/625-acumulator-li-ion-samsung-2200-mah-18650.html?search_query=18650&results=4) | 75 RON |
| HX-3S-A02 BMS | 1 | UNDA Tech | 30 RON |
| 12V 1A Charger | 1 | [UNDA Tech](https://magazin.unda.tech/ro/transformatoare/240-transformator-9v-1a.html) | 15 RON |
| Various Wires | 1 | [UNDA Tech](https://magazin.unda.tech/ro/cablu-banda-si-fire/290-cablu-dupont-70cm-4p.html?search_query=fire&results=12) | 10 RON |

Prototype total cost: 227 RON

\* The cost is calculated for only one device fabricated. Taking into consideration that the quantity will drop this price, the device is for sure a very cheap accesory that can help blind people travel and move easier.

The prototype proved to be a succes, so we will continue to optimise the functionality of the device. The known issues that came out during testing were about the time of measurement. The measuring time was dependent on the distance of the object,sometimes taking a full second to measure a distance. Knowing that the device will be worn while moving, the responce time should be as little as possible. This issue will be solved in the device software algorithm.

**Accuracy**

The device proved to be very accurate, even with small and moving objects. The SR-HC04 manufacturer recommended a minimum size of 40 squere centimeters of material as a reference measurement object, but even a small object like a hand was used to determine the accuracy of the device. The device measures the distance in centimeters, meaning that the aproximation errors could reach a 100% percent when measuring less than a centimeter of distance. This will happen just when the device measures a value of half of a centimeter. The functionality of the device is not affected by this error because the motors will be cotrolled by a interval of distances, not by exact values.

**Efficiency**

To see the efficiency and power consumption of the device, we tested out in a controlled environment . The device passed all the test without any problems, being even more efficient in terms of power consumption. The maximul current draw of our device was 150mA, which is lower than our previously calculated 200mA . The two motors were powered on with the two analog pins that we chose earlier, having a “analogWrite” value of 255 (100% duty cycle – constant 5V supply) .

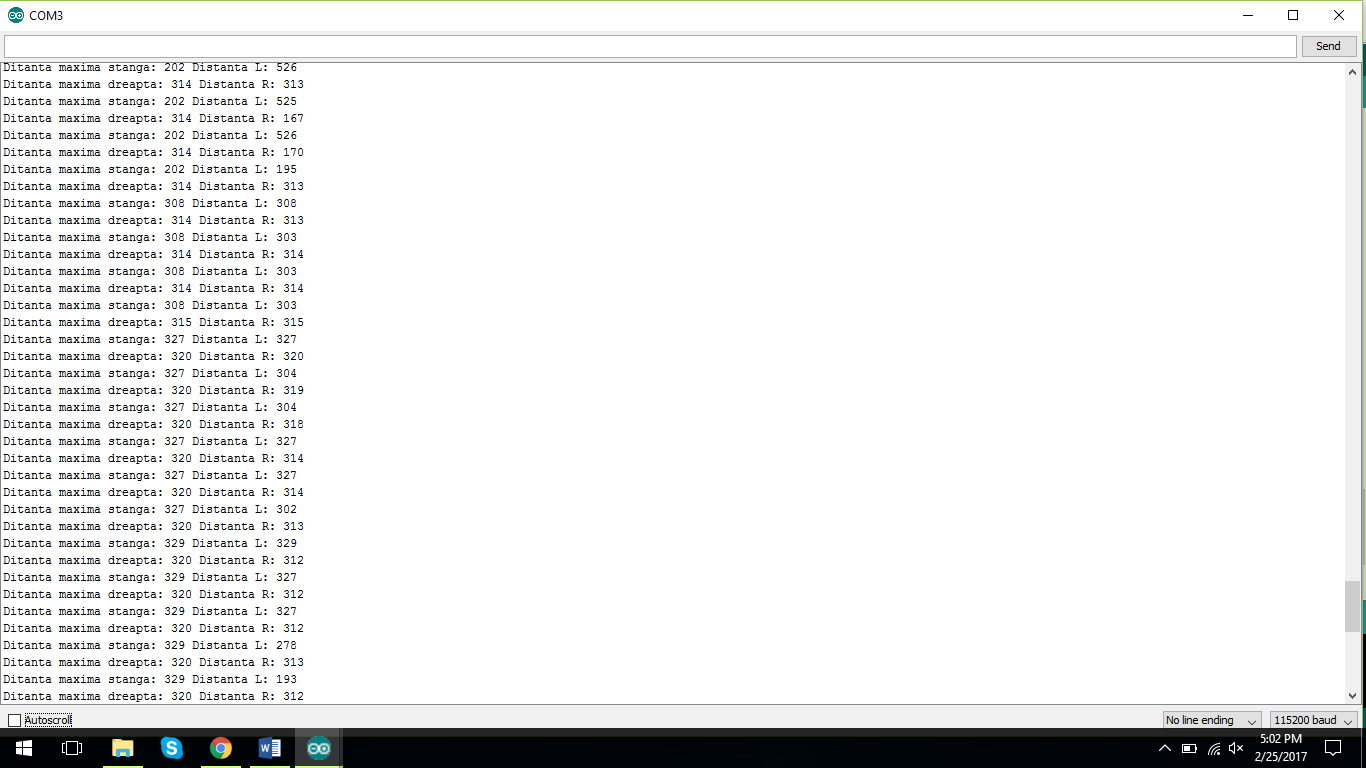
The motors handled this voltage well, the averege working temperature of them was 30°C in a 24-26°C environment.

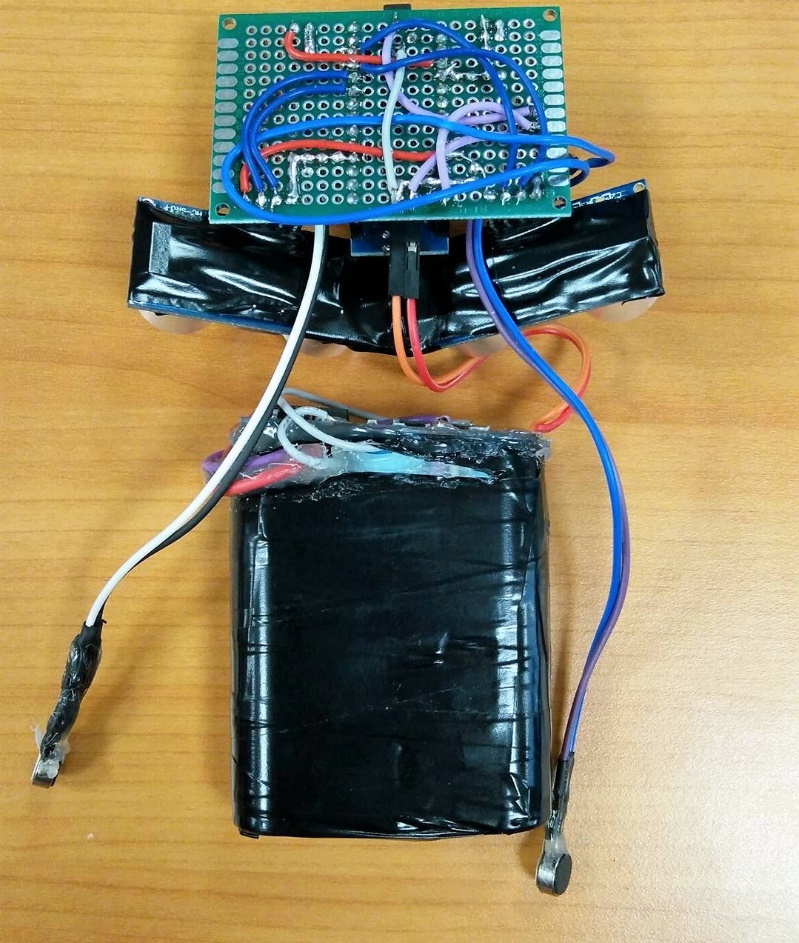
A simple multimeter test resulted in a 0.15A measurement in series with the battery pack.

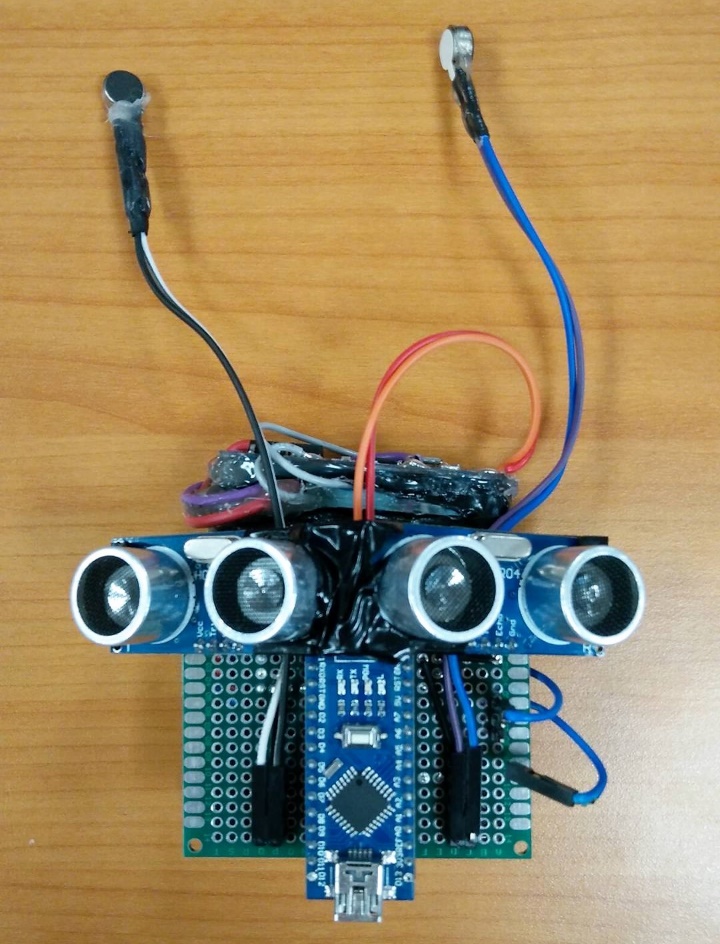


**Range test**

As we tested the device in an open space, the farest object was detected at 329 cm. The manufacturer recommended a measure distance of maximum 400cm.



** Final board and assembly**

****

**SOFTWARE**

**Integrated Development environment**

To link all the components in a manner that we want it to be linked, we need to use an IDE that will help us compile and upload a code to our ATmega328P microcontroller. To make this happen, the easies way to program our Arduino Nano is to use the Arduino IDE.

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

The Arduino IDE is using a C-based programming language, having all the basic and necesary algorithms to help us transforming our idea to a real device. We will need only the basic functions of Arduino IDE, without using external libraries as we will implement our own way of measuring distance with the hardware we have.

Arduino programs can be divided in three main parts: structure, values (variables and constants), and functions

The structure is divided in two parts: the setup() and the loop() functions.

- The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each powerup or reset of the Arduino board.

- After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

**The Arduino IDE Code for our device**

First of all, we need to define the pins that we will use with our sensors/motors. Here we are defining a const int pin number for L\_TRIG,L\_ECHO (Trigger and echo for the left sensor) , R\_TRIG,R\_ECHO (Trigger and echo for the right sensor) , Mot\_right\_pin (Positive wire of the Right motor) and Mot\_left\_pin (Positive wire of the Left motor) .

// Sensor and motors pins definition  
 const int L\_TRIG = 9;  
 const int L\_ECHO = 8;

const int R\_TRIG = 7;  
 const int R\_ECHO = 6;

const int Mot\_right\_pin = A1;  
 const int Mot\_left\_pin = A2;  
// End of sensor and motors pin definition

Then we will define some distance constraints that will help us improve the quality of the measurement and the accuracy of the stimuli representation. Here we are defining the “duration” long variable, which will save the travel time of a sonic wave generated by our Ultrasonic Sensors. The distance will save the calculated distance that our function will generate and thus in RightSensor and LeftSensor we will save the final distances in both directions.

The dist\_max\_left and dist\_max\_right variables will save the maximum distance that was measured during a working cycle, to have a reference point for our algorithm.  
 The Mot\_left and Mot\_right will save the analog value that will be written in the Mot\_right\_pin and Mot\_left\_pin. This value represent a duty cycle (0 -> 0% duty cycle and 255 -> 100% duty cycle)  
 The maximul\_measure\_distance is the software defined sensor limit of the measurements, ignoring anything bigger than that because it could be Ultrasonic Sound noise that could ruin our measurements.

// Distance constraints

long duration, distance, RightSensor,LeftSensor;

long dist\_max\_left=0;  
 long dist\_max\_right=0;

float Mot\_left,Mot\_right;

float maximum\_measure\_distance=400;

// End of distance constraints

Then we will define our setup() function:

This function will hold the initialization data which will run when the sketch starts (when the arduino is powered on).   
The pinMode(pin,type) function defines the type of the pins we will interact with. The input type is used to read data and the output type is used to write data.

We will define the TRIG pins as OUTPUT (from there we will send the sonic waves) and the ECHO pins as INPUT (here we will wait for the soni waves to come).  
  
The motor pins will be defined as OUTPUT pins as we will write an analogic value to them when we need to turn them on/off depending on the measurements.

void setup()

{

// Initialize the serial port to see the measured distances

Serial.begin(115200);

// Initialize the pins as input/output

pinMode(L\_TRIG, OUTPUT);  
 pinMode(L\_ECHO, INPUT);

pinMode(R\_TRIG, OUTPUT);  
 pinMode(R\_ECHO, INPUT);

pinMode(Mot\_left\_pin, OUTPUT);  
 pinMode(Mot\_right\_pin, OUTPUT);

// End of pins initialization

}

Next we will look at the loop() function, which will contain all the measuring and processing technuques. The further code will be explained after each important pack of instructions. (in /\* \*/)

void loop()  
{  
 // Left sensor distance measurement

SonarSensor(L\_TRIG, L\_ECHO);   
 LeftSensor = distance;

/\*The sonarSensor functon will modify the “distance” variable, making it the measured value. We will save this value in LeftSensor variable. \*/

if(dist\_max\_left<LeftSensor)  
 dist\_max\_left=LeftSensor;  
 /\* We verify if the distante we measured is higher than the maximul value we saved . If it is, we overwrite this value with the new one. This will save the maximul value after every loop() \*/

// End of left sensor distance measurement

// Right sensor distance measurement

SonarSensor(R\_TRIG, R\_ECHO);  
 RightSensor = distance;  
  
 /\*The sonarSensor functon will modify the “distance” variable, making it the measured value. We will save this value in RightSensor variable. \*/

if(dist\_max\_right<RightSensor)  
 dist\_max\_right=RightSensor;  
 /\* We verify if the distante we measured is higher than the maximul value we saved . If it is, we overwrite this value with the new one. This will save the maximul value after every loop() \*/

// End of right sensor distance measurement

// Verifying the range limit

/\* We will verify if any of the distances we measured earlier exceeds the maximum measurable distance. If not,the loop function will be called again.\*/

if(RightSensor < maximum\_measure\_distance || LeftSensor < maximum\_measure\_distance)

{

// Right sensor serial printing

Serial.print("Ditanta maxima dreapta: ");  
 Serial.print(dist\_max\_right);  
 Serial.print(" ");  
 Serial.print("Distanta R: ");  
 Serial.println(RightSensor);

/\* To see in the serial monitor the actual distance, we are using the “Serial.print()” function which will print the maximum distance and the measured distance. \*/

// End of right sensor serial printing

// Right motor driving algorithm  
 // All the distances are in cm (CENTIMETERS)

/\* We will define 4 intervals in which the motors will behave different, representing the distance to the obstacle that is eighter in the right or the left side of the device \*/

if(RightSensor<250&&RightSensor>150)  
 {  
 analogWrite(Mot\_right\_pin,255);  
 delay(300);  
 analogWrite(Mot\_right\_pin,0);  
 delay(600);  
 }  
 /\* If the distance is anywhere between 150 and 250, the motor will vibrate for 300ms and will wait for 600ms \*/

if(RightSensor<150&&RightSensor>75)  
 {   
 analogWrite(Mot\_right\_pin,255);  
 delay(300);  
 analogWrite(Mot\_right\_pin,0);  
 delay(300);  
 }

/\* If the distance is anywhere between 150 and 75, the motor will vibrate for 300ms and will wait for 300ms \*/

if(RightSensor<75&&RightSensor>35)  
 {  
 analogWrite(Mot\_right\_pin,255);  
 delay(300);  
 analogWrite(Mot\_right\_pin,0);  
 delay(100);  
 }

/\* If the distance is anywhere between 75 and 35, the motor will vibrate for 300ms and will wait for 100ms \*/

if(RightSensor<35)  
 analogWrite(Mot\_right\_pin,255);

/\* If the distance lower than 35, the motor will vibrate until the next measurement returns a distance bigger than 35 \*/

if(RightSensor>250)  
 analogWrite(Mot\_right\_pin,0);

/\* If the distance is higher than 250, the motor will not vibrate until the next measurement returns a distance lower than 35 \*/

// End of right motor driving algorithm

/\* The left sensor works exactly the same as the right sensor . \*/

// Left sensor Serial printing  
 Serial.print("Ditanta maxima stanga: ");  
 Serial.print(dist\_max\_left);  
 Serial.print(" ");  
 Serial.print("Distanta L: ");  
 Serial.println(LeftSensor);

// End of left sensor Serial printing

// Left motor driving algorithm  
 if(LeftSensor<250&&LeftSensor>150)  
 {  
 analogWrite(Mot\_left\_pin,255);  
 delay(300);  
 analogWrite(Mot\_left\_pin,0);  
 delay(600);  
 }

if(LeftSensor<150&&LeftSensor>75)  
 {  
 analogWrite(Mot\_left\_pin,255);  
 delay(300);  
 analogWrite(Mot\_left\_pin,0);  
 delay(300);  
 }

if(LeftSensor<75&&LeftSensor>35)  
 {  
 analogWrite(Mot\_left\_pin,255);  
 delay(300);  
 analogWrite(Mot\_left\_pin,0);  
 delay(100);  
 }

if(LeftSensor<35)  
 analogWrite(Mot\_left\_pin,255);

if(LeftSensor>250)  
 analogWrite(Mot\_left\_pin,0);

// End of left motor driving algorithm   
}  
 // End of veryfing the range limit  
}

The function that modifies the “distance” variable is the SonarSensor, which is defined as :

void SonarSensor(int trigPin,int echoPin)

{

digitalWrite(trigPin, LOW);  
 delayMicroseconds(2);

/\* Make sure the trigPin is LOW and wait 2us \*/

digitalWrite(trigPin, HIGH);  
 delayMicroseconds(10);

/\* Make the trigPin HIGH and wait for 10uS . This will start a “burst-fire” of 8 ultrasonic sound waves at 40kHz . This is the SR-HC04 Routine. For more information visit the Sensors part \*/

digitalWrite(trigPin, LOW);  
 delayMicroseconds(20);

/\* Make the trigPin Low and wait for 20uS. This is used to make sure that the waves have enough time to bounce back and forth to the sensor. \*/

duration = pulseIn(echoPin, HIGH);  
/\* the pulseIn(pin,state) will save the duration (in ms) until the waves come back to the echoPin. This will help us determine the distance knowing the speed of sound. \*/

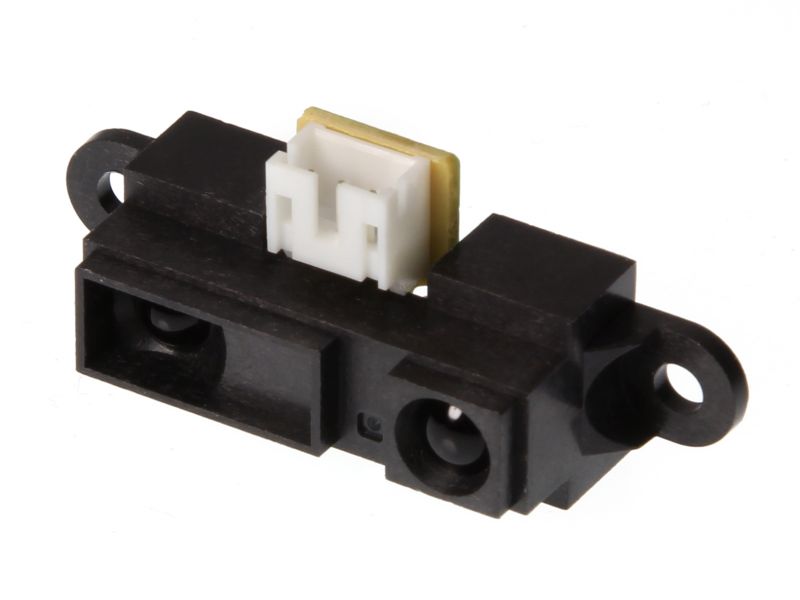
/\* Considering an averege temperature of 20°C in Romania, the speed of sound is 343,4m/s . The duration of the wave travel is twice the duration to the object, meaning that we need to divide the duration by two to have a correct reading of distance \*/   
  
 distance = duration \* 0.034 / 2;

}

**Further improvements:**

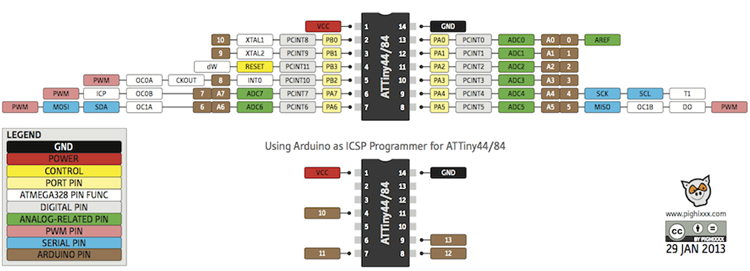
**Sensors**

The first major improvement would be a better sensor set. The HC-SR04 ultrasonic sensor has some issues with the fast measurements, being slow at long distance as it has to wait until the sonic wave bounces back an forth to the object. A better alternative would be a Sharp sensor, featuring a IR distance measuring algorithm which is faster ( Light speed is faster than sound speed ) .



**Hardware & Microcontroller**

The hardware assembly of the device is not very complicated, i uses only 6 pins of the Arduino Nano board, 4 of them being digital I/O pins and 2 analog I/O pins. The same functionality could be obtined using a ATtiny44/84 microcontroller and a custom PCB.



**Battery**

A smaller alternative for our battery pack could be made from 15266 Li-Ion Cells. The 15266 Li-Ion cells are smaller than the 18650, but they also provide less capacity. If we are able to decrease the power consumption of our device, which is very possible, we could be able to use the small 15266 li-ion cells.  
The 15266 li-ion cells provide a maximum capacity of 1000-1200 mAh, meaning under a half of our battery pack capacity. Also, the averege price/capacity is higher with the 15266 li-ion cells, but the cost could be reduced at the hardware and still obtain a cheap and efficient device.

