Design and Implementation of Car Sensing System (Obstacle Avoiding Robot)

Thesis · March 2016

DOI: 10.13140/RG.2.2.15459.66082

Ministry of higher education
&scientific research
Al –Mustaqbal University College
Computer Techniques Engineering
Department.

Design and Implementation of Car Sensing System
(Obstacle Avoiding Robot)

By
Mohammad AbdulHamza
Zaid Sadiq Hameed
Mustafa AbdulKadhim

Supervised by
MS.c. Mohammed Azher Therib

2015 – 2016

بسم الله الرحمن الرحيم

ألم نَشرَحْ لَكَ صَدرَكَ (1) وَوضَعنا عَنْك وِزرَكَ (2) ألذيْ أنْقَضَ ظَهْرَكَ (3) وَرفعْنا لَكَ ذِكرَكَ (4) فانَّ مَعَ ألعُسرِ يُسْراً (5) اِنَّ مَعَ ألعُسرِ يُسْرْاً (6) فاذا فَرَغْتَ فأنْصَبْ (7) واِلَىْ رَبِكَ فأرْغَبْ (8).

صَدَقَ ألله العَليّ ألعَظيْمْ

(سورة الانشراح)

(الإهـــــــــــداء)

لو ان عطايا الملوك تهدى لأهديناها لوجهك الكريم ولكنها عطاياك لا نحصي ثناءاً عليك فأنت كما أثنيت على نفسك

الى من دنى فتدلى فكان قاب قوسين أو أدنى، سيد الكونين وحبيب إله العالمين ... المصطفى محمد (صلى الله عليه واله وسلم)

الى من ضرب بسيفين وطعن برمحين وبايع البيعتين وهاجر الهجرتين وقاتل ببدرِ وحنين... المرتضى علي (عليه السلام)

الى المدخر لإقامة الأمتِ والعِوَج، أمل المستضعفين في الأرض حجة الله في أرضه... المهدي (عجل الله تعالى فرجه)

الى من ذلل لي المصاعب وهون عليَّ المتاعب من كان لي نِعمَ الرفيق أنارَ لي الطريقْ... أبي العزيز

الى من سهرت الليالي ورخصت لي الغوالي ،الى القلب الفسيح والوجه السميح ،من حصنتني بالدعاء وعانت من أجلي الشقاء... أمي الغالية

الى من علمني حرفا فملكني عبدا ،الى من رفدني بالعلم والمعرفة وبذلَ من أجلي الجهد الوفير... أستاذي الفاضل

أهدي إليهم ثمرة جهدي المتواضع

Contents

Chapter One

Introduction

Arduino is a software company, project, and user community that designs and manufactures computer open-source hardware, open-source software, and microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices.

The project is based on microcontroller board designs, produced by several vendors, using various microcontrollers. These systems provide sets of digital and analog I/O pins that can interface to various expansion boards (termed shields) and other circuits. The boards feature serial communication interfaces, including Universal Serial Bus (USB) on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino project provides an integrated development environment (IDE) based on a programming language named Processing, which also supports the languages C and C++.

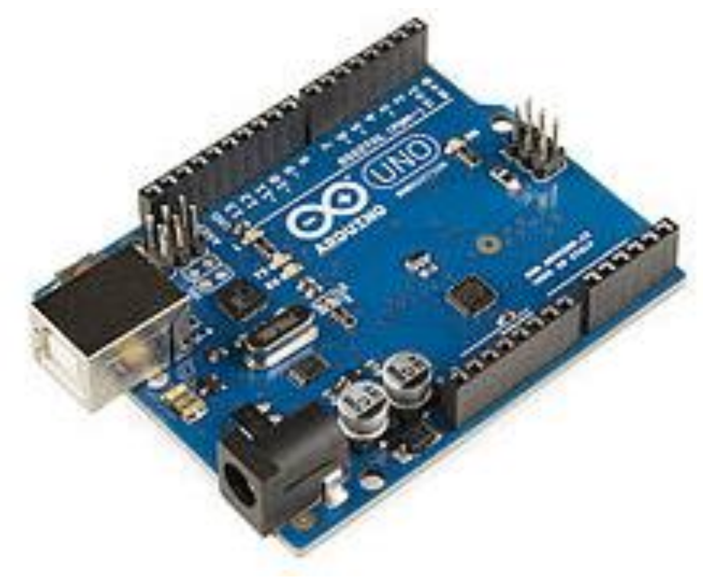


Figure 1: Arduino Board.

The first Arduino was introduced in 2005, aiming to provide a low cost, easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

Arduino boards (as shown in figure 1) are available commercially in preassembled form, or as do-it-yourself kits. The hardware design specifications are openly available, allowing the Arduino boards to be produced by anyone. Adafruit Industries estimated in mid-2011 that over 300,000 official Arduinos had been commercially produced, and in 2013 that 700,000 official boards were in users' hands.

History of Arduino:

Colombian student Hernando Barragán created the development platform Wiring as his Master's thesis project in 2004 at the Interaction Design Institute Ivrea in Ivrea, Italy. Massimo Banzi and Casey Reas (known for his work on Processing) were supervisors for his thesis. The goal was to create low cost, simple tools for non-engineers to create digital projects. The Wiring platform consisted of a hardware PCB with an ATmega128 microcontroller, an integrated development environment (IDE) based on Processing and library functions to easily program the microcontroller.

In 2005, Massimo Banzi, with David Mellis (then an IDII student) and David Cuartielles, added support for the cheaper ATmega8 microcontroller to Wiring. But instead of continuing the work on Wiring, they forked (or copied) the Wiring source code and started running it as a separate project, called Arduino.

The Arduino's initial core team consisted of Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis.

The name Arduino comes from a bar in Ivrea, where some of the founders of the project used to meet. The bar was named after Arduin of Ivrea, who was the margrave of the March of Ivrea and King of Italy.

Following the completion of the Wiring platform, its lighter, lower cost versionswere created and made available to the open-source community. Associated researchers, including David Cuartielles, promoted the idea.

Development:

Arduino-compatible R3 UNO board made in China with no Arduino logo, but with identical markings, including "Made in Italy" text

Arduino is an open-source hardware. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino website. Layout and production files for some versions of the hardware are also available. The source code for the IDE is released under the GNU General Public License, version 2.

Although the hardware and software designs are freely available under copyleft licenses, the developers have requested that the name "Arduino" be exclusive to the official product and not be used for derived works without permission. The official policy document on use of the Arduino name emphasizes that the project is open to incorporating work by others into the official product. Several Arduino-compatible products commercially released have avoided the Arduino name by using -duino name variants.

Applications of Arduino:

The arduino have many applications like:

- Xoscillo, an open-source oscilloscope.
- Scientific equipment such as the Chemduino.
- Arduinome, a MIDI controller device that mimics the Monome.
- OBDuino, a trip computer that uses the on-board diagnostics interface found in most modern cars.
- Ardupilot, drone software and hardware.
- Arduino Phone, a do-it-yourself cellphone.
- GertDuino, an Arduino mate for the Raspberry Pi.
- Water quality testing platform.
- Homemade CNC using Arduino and DC motors with close loop control by Homofaciens.
- DC motor control using Arduino and H-Bridge.

Obstacle Avoidance Applications:

In robotics, obstacle avoidance is the task of satisfying some control objective subject to non-intersection or non-collision position constraints. In unmanned air vehicles, it is a hot topic. What is critical about obstacle avoidance concept in this area is the growing need of usage of unmanned aerial vehicles in urban areas for especially military applications where it can be very useful in city wars. Normally obstacle avoidance is considered to be distinct from path planning in that one is usually implemented as a reactive control law while the other involves the pre-computation of an obstacle-free path which a controller will then guide a robot along.

An OBSTACLE AVOIDING ROBOT is one which can avoid an obstacle by using ultrasound sensor and navigate in its own path. With a breadboard attached to the robot you can play fun within a short period of time. One such is what we are going to discuss here. This project can teach you how a sensor can be used to process some data.

Chapter Two

Arduino and its Accessories

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

Arduino Facilities:

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Hardware:

An Arduino board historically consists of an Atmel 8-, 16- or 32-bit AVR microcontroller (although since 2015 other makers' microcontrollers have been used) with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which let users connect the CPU board to a variety of interchangeable add-on modules termed shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I²C serial bus—so many shields can be stacked and used in parallel. Before 2015, Official Arduinos had used the Atmel megaAVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. In 2015, units by other producers were added. A handful of other processors have also been used by Arduino compatible devices. Most boards include a 5 V linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external programmer. This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer. Currently, optiboot bootloader is the default bootloader installed on Arduino UNO.

At a conceptual level, when using the Arduino integrated development environment, all boards are programmed over a serial connection. Its implementation varies with the hardware version. Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistor–transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Uno boards shown in figure 2 below, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods, when used with traditional microcontroller tools instead of the Arduino IDE, standard AVR in-system programming (ISP) programming is used.

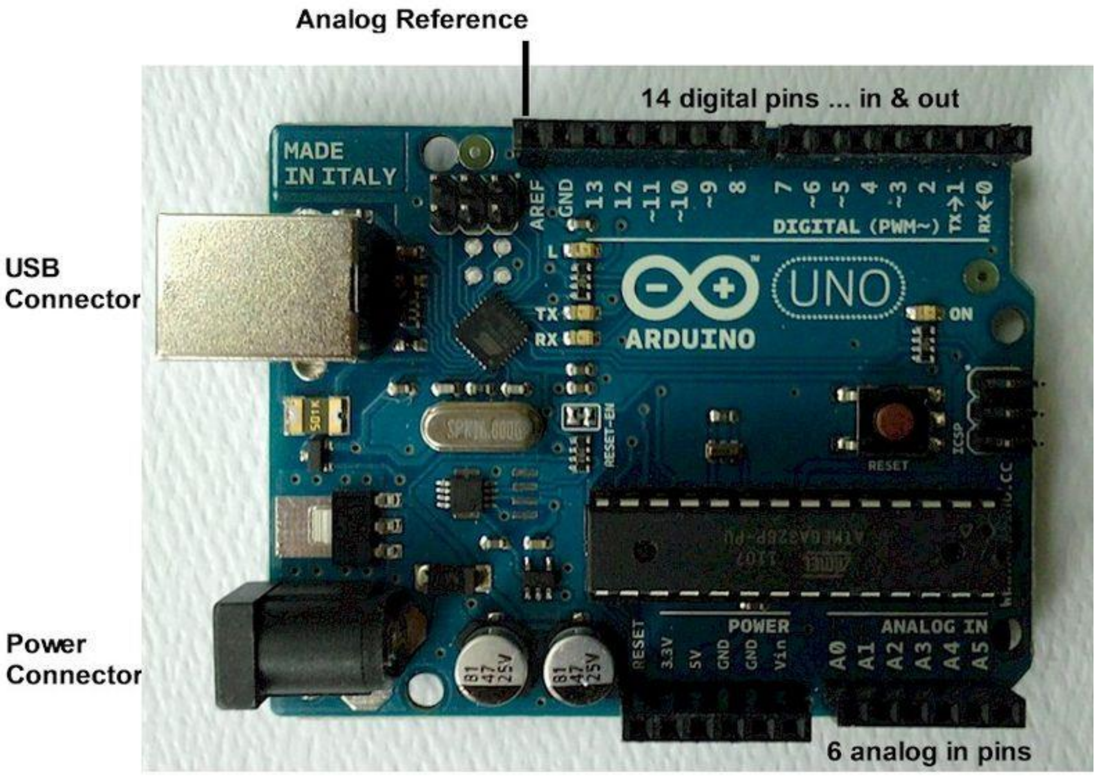


Figure 2: An official Arduino Uno Revision 2 with descriptions of the I/O locations

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, Duemilanove, and current Uno provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into solderless breadboards.

Many Arduino-compatible and Arduino-derived boards exist. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education, to simplify making buggies and small robots. Others are electrically equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not. Some variants use different processors, of varying compatibility.

Software

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages *Processing* and *Wiring*. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and provides simple one-click mechanism to compile and load programs to an Arduino board. A program written with the IDE for Arduino is called a "sketch".

The Arduino IDE supports the languages C and C++ using special rules to organize code. The Arduino IDE supplies a software library called Wiring from the Wiring project, which provides many common input and output procedures. A typical Arduino C/C++ sketch consist of two functions that are compiled and linked with a program stub *main()* into an executable cyclic executive program:

- *Setup( )*: a function that runs once at the start of a program and that can initialize settings.
- *loop( )*: a function called repeatedly until the board powers off.

After compiling and linking with the GNU toolchain, also included with the IDE distribution, the Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal coding that is loaded into the Arduino board by a loader program in the board's firmware.

Arduino Types:

There are many types of Arduino as shown in figure 3 below like:

- Arduino Diecimila in Stoicheia
- Arduino Duemilanove (rev 2009b)
- Arduino UNO
- Arduino Leonardo
- Arduino Mega
- Arduino MEGA 2560 R3 (front side)
- Arduino MEGA 2560 R3 (back side)
- Arduino Nano

- Arduino Due (ARM Cortex-M3 core)
- LilyPad Arduino (rev 2007)
- Arduino Yun

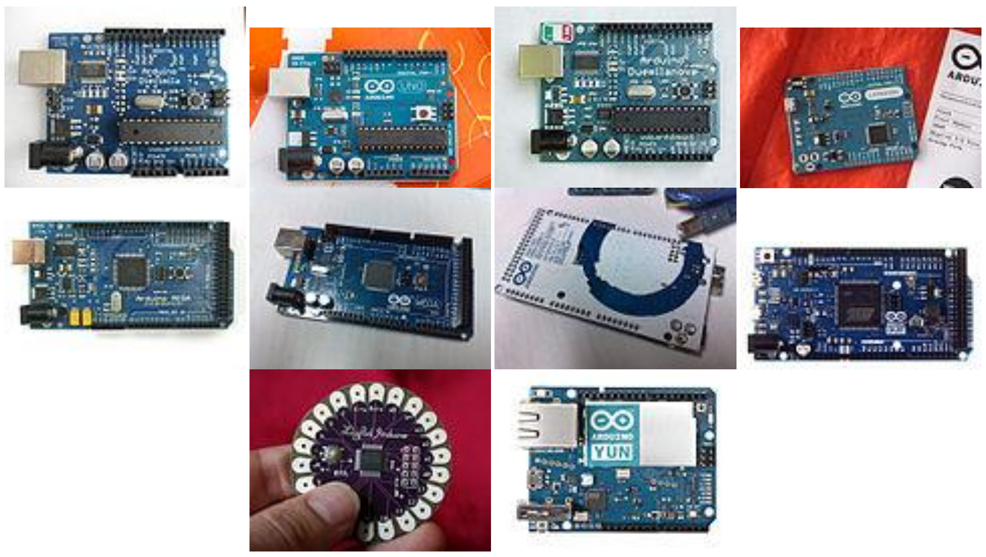


Figure 3: Arduino Types.

Sensors:

In the broadest definition, a sensor is an object whose purpose is to detect events or changes in its environment, and then provide a corresponding output. A sensor is a type of transducer; sensors may provide various types of output, but typically use electrical or optical signals. For example, a thermocouple generates a known voltage (the output) in response to its temperature (the environment). A mercury-in-glass thermometer, similarly, converts measured temperature into expansion and contraction of a liquid, which can be read on a calibrated glass tube.

Sensors are used in everyday objects such as touch-sensitive elevator buttons (tactile sensor) and lamps which dim or brighten by touching the base, besides innumerable applications of which most people are never aware. With advances in micro-machinery and easy-to-use micro controller platforms, the uses of sensors have expanded beyond the most traditional fields of temperature, pressure or flow measurement, for example into MARG sensors. Moreover, analog sensors such as potentiometers and force-sensing resistors are still widely used. Applications include manufacturing and machinery, airplanes and aerospace, cars, medicine, and robotics. It is also included in our day-to-day life.

A sensor's sensitivity indicates how much the sensor's output changes when the input quantity being measured changes. For instance, if the mercury in a thermometer moves 1 cm when the temperature changes by 1 °C, the sensitivity is 1 cm/°C (it is basically the slope Dy/Dx assuming a linear characteristic). Some sensors can also have an impact on what they measure; for instance, a room temperature thermometer inserted into a hot cup of liquid cools the liquid while the liquid heats the thermometer. Sensors need to be designed to have a small effect on what is measured; making the sensor smaller often improves this and may introduce other advantages. Technological progress allows more and more sensors to be manufactured on a microscopic scale as microsensors using MEMS technology. In most cases, a microsensor reaches a significantly higher speed and sensitivity compared with macroscopic approaches.

UltraSonic Distance Measurement:

Small low-cost ultrasonic distance measurement modules like SRF-06 are an effective way to sense the presence of nearby objects and the distance to them. Often Robots use these to sense objects or collisions and take appropriate action.

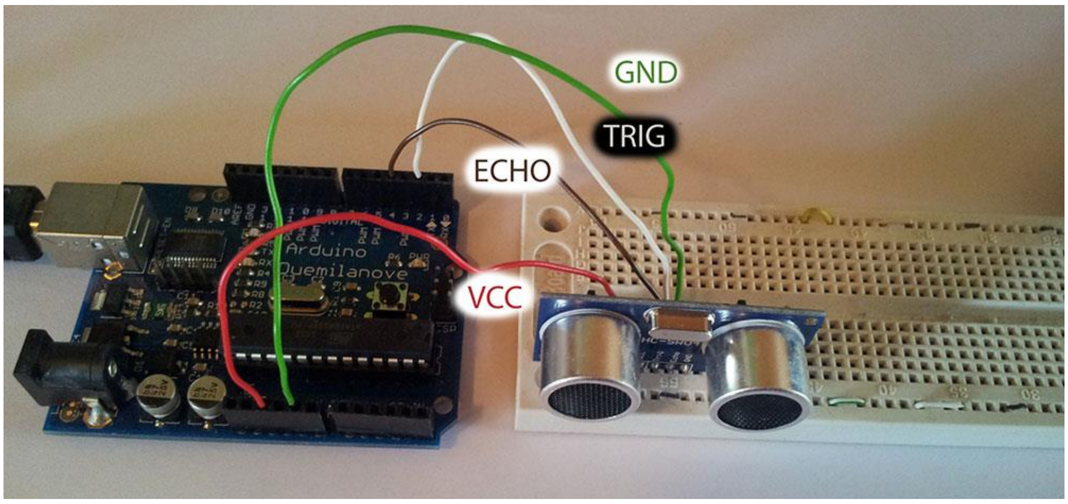


Figure 4: UltraSonic Sensor.

They have two transducers, basically a speaker and a microphone. Ultrasound is a high frequency sound (typically 40 KHz is used). A short burst of sound waves (often only 8 cycles) is sent out the "Transmit" transducer (left, above). Then the "Receive" transducer listens for an echo. Thus, the principle of ultrasonic distance measurement is the same as with Radio-based radar.

Speed of sound in air velocity is affected by the air density, and for high accuracy the temperature must be taken into account, either within the module electronics (In the SRF-06 module we have) or in the Arduino software.

The module in our example shown in figure 4 below has 4 pins:

1. Vcc Operating voltage: 5.0V
2. Trig the transmit signal pin
3. Echo the received echo pin
4. Gnd Ground

It emits an ultrasound at 40 000 Hz which travels through the air as shown in figure 5 and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.

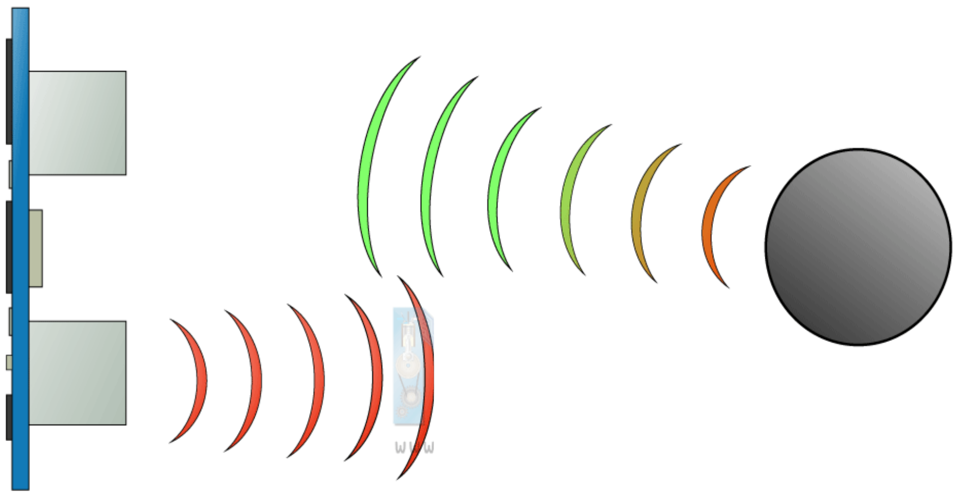


Figure 5: UltraSonic Sensor work.

The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo. The Ground and the VCC pins of the module needs to be connected to the Ground and the 5 volts pins on the Arduino Board respectively and the trig and echo pins to any Digital I/O pin on the Arduino Board. In order to generate the ultrasound you need to set the Trig on a High State for 10 µs as shown in figure 6. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo pin. The Echo pin will output the time in microseconds the sound wave traveled.

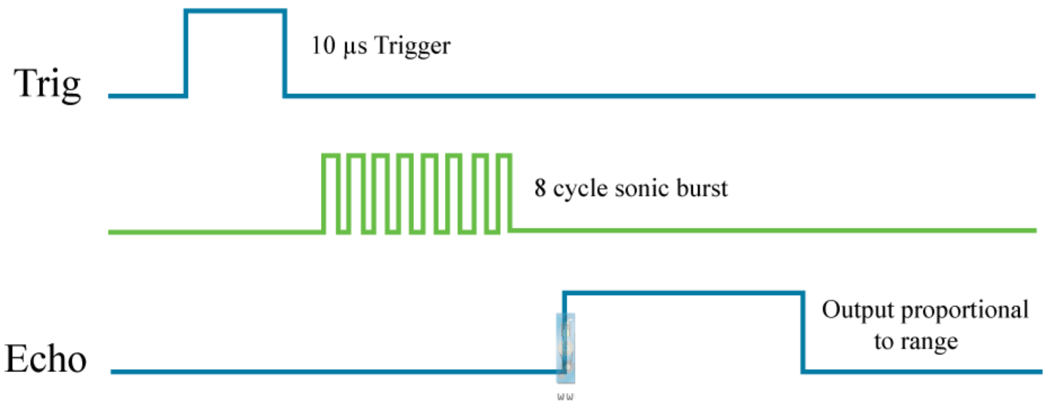


Figure 6: Trigger and Echo signals.

For example, if the object is 10 cm away from the sensor, and the speed of the sound is 340 m/s or 0.034 cm/µs the sound wave will need to travel about 294 u seconds as shown in figure 7. But what you will get from the Echo pin will be double that number because the sound wave needs to travel forward and bounce backward. So in order to get the distance in cm we need to multiply the received travel time value from the echo pin by 0.034 and divide it by 2.

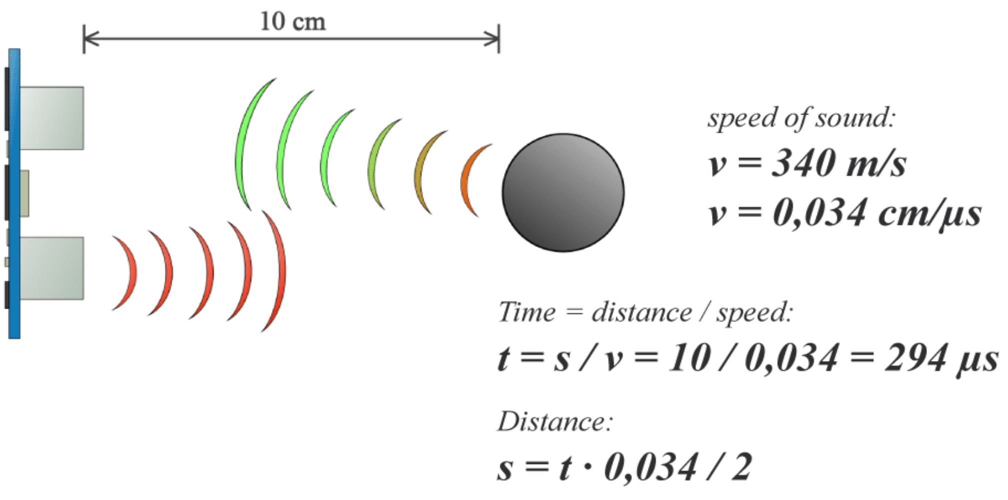


Figure 7: Example for UltraSonic Sensor.

Motors:

A DC motor has a two wire connection. All drive power is supplied over these two wires think of a light bulb. When you turn on a DC motor, it just starts spinning round and round. Most DC motors are pretty fast, about 5000 RPM (revolutions per minute).

With the DC motor, its speed (or more accurately, its power level) is controlled using a technique named *pulse width modulation*, or simply *PWM*. This is idea of controlling the motor's power level by *strobing the power on and off*. The key concept here is *duty cycle* which is the percentage of "on time" versus" off time." If the power is on only 1/2 of the time, the motor runs with 1/2 the power of its full-on operation.

If you switch the power on and off fast enough, then it just seems like the motor is running weaker—there's no stuttering. This is what PWM means when referring to DC motors. The Handy Board's DC motor power drive circuits simply switch on and off, and the motor runs more slowly because it's only receiving power for 25%, 50%, or some other fractional percentage of the time.

A servo motor is an entirely different story. The servo motor is actually an assembly of four things: a normal DC motor, a gear reduction unit, a position-sensing device (usually a potentiometer a volume control knob), and a control circuit.

DC motor   Servo motor

Figure 8: DC and Servo motors.

The function of the servo is to receive a control signal that represents a *desired output position of the servo shaft*, and apply power to its DC motor until its shaft turns to that position. It uses the position-sensing device to determine the rotational position of the shaft, so it knows which way the motor must turn to move the shaft to the commanded position. The shaft typically does *not* rotate freely round and round like a DC motor, but rather can only turn 200 degrees or so back and forth.

The servo has a 3 wire connection: power, ground, and control. The power source must be constantly applied; the servo has its own drive electronics that draw current from the power lead to drive the motor.

The control signal is pulse width modulated (PWM), but here the **duration** of the positive-going pulse determines the **position** of the servo shaft as shown in figure 9. For instance, a 1.520 millisecond pulse is the center position for a Futaba S148 servo. A longer pulse makes the servo turn to a clockwise-from-center position, and a shorter pulse makes the servo turn to a counter-clockwise-from-center position.

The servo control pulse is repeated every 20 milliseconds. In essence, every 20 milliseconds you are telling the servo, “go here”.

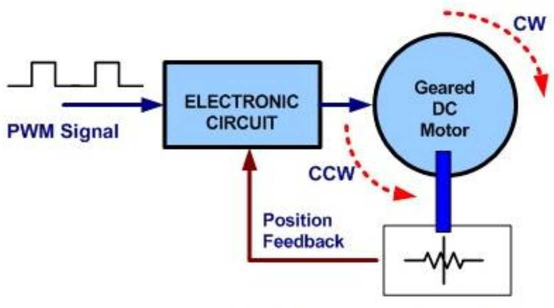


Figure 9: The servo motor work.

To recap, there are two important differences between the control pulse of the servo motor versus the DC motor. First, on the servo motor, duty cycle (on-time vs. off-time) has *no meaning* whatsoever—all that matters is the absolute duration of the positive-going pulse, which corresponds to a commanded output position of the servo shaft. Second, the servo has its own power electronics, so *very little power* flows over the control signal. All power is draw from its power lead, which must be simply hooked up to a high-current source of 5 volts.

Contrast this to the DC motor. On the Handy Board, there are specific motor driver circuits for four DC motors. Remember, a DC motor is like a light bulb; it has no electronics of its own and it requires a large amount of drive current to be supplied to it. This is the function of the L293D chips on the Handy Board, to act as large current switches for operating DC motors.

Plans and software drivers are given to operate two servo motors from the HB. This is done simply by taking spare digital outputs, which are used to generate the precise timing waveform that the servo uses as a control input. Very little current flows over these servo control signals, because the servo has its own internal drive electronics for running its built-in motors.

Relays:

A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal. The first relays were used in long distance telegraph circuits as amplifiers: they repeated the signal coming in from one circuit and re-transmitted it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Solid-state relays control power circuits with no moving parts, instead using a semiconductor device to perform switching. Relays with calibrated operating characteristics and sometimes multiple operating coils are used to protect electrical circuits from overload or faults; in modern electric power systems these functions are performed by digital instruments still called "protective relays".

Magnetic latching relays require one pulse of coil power to move their contacts in one direction, and another, redirected pulse to move them back. Repeated pulses from the same input have no effect. Magnetic latching relays are useful in applications where interrupted power should not be able to transition the contacts.

Magnetic latching relays can have either single or dual coils. On a single coil device, the relay will operate in one direction when power is applied with one polarity, and will reset when the polarity is reversed. On a dual coil device, when polarized voltage is applied to the reset coil the contacts will transition. AC controlled magnetic latch relays have single coils that employ steering diodes to differentiate between operate and reset commands.

To connect the 4 Relay board to an Arduino as shown in figure 10 below is very easy and allows you to turn on and off an wide range of devices, both AC and DC. The first to connections are the ground and power pins, You need to connect the Arduino +5v to the 4 Relay board VCC pin and the Arduino ground to the 4 Relay board GND pin. Then it's a only a matter of just connecting the communication pins, labeled IN1, IN2, IN3 and IN4, two 4 data pins on the Arduino.

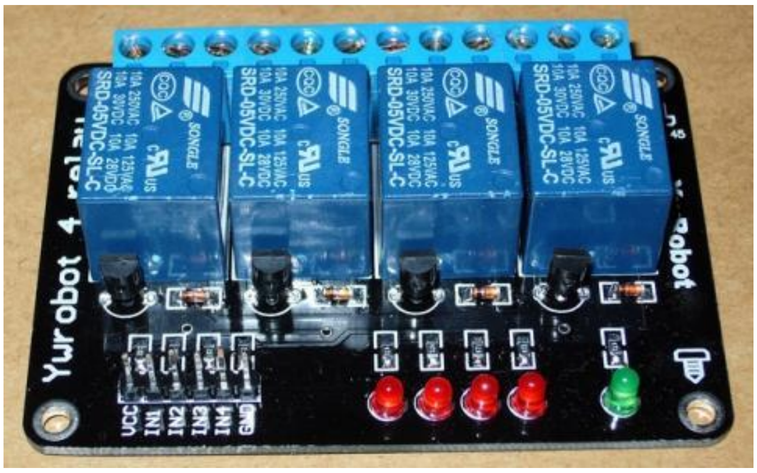


Figure 10: Relays with Arduino.

The default state of the relay when the power is off for COMM (power) to be connected to NC (normally closed), this is the equivalent of setting the 4 Relay boards IN pin to HIGH (has +5v sent to it) It is a safety feature to not use the NC connector in-case you Arduino looses power it will automatically turns off all the devices connected to the relay as shown in figure 11 below.

When you have something connected to the relays NO (Normally Open) connector and you set the corresponding IN pin to LOW (0v), power will flow in from the COMM connector and out of the NO connector powering your device.

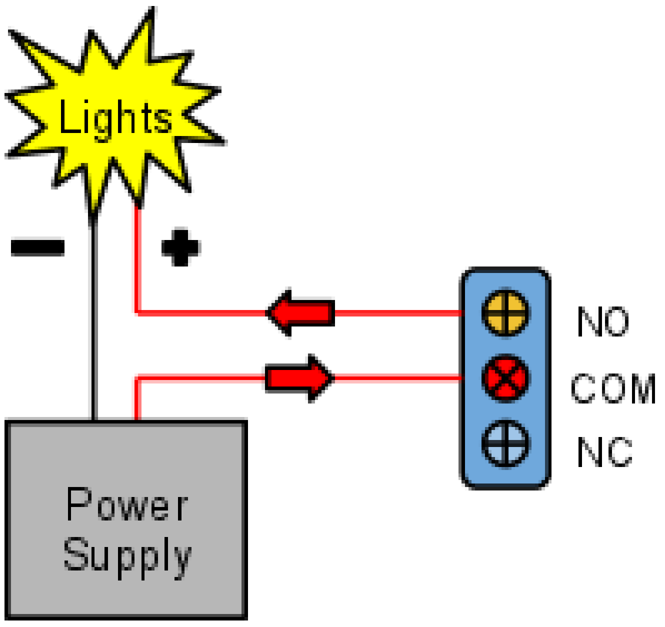


Figure 11: Relays Connection.

In the next chapter the proposed car sensing system (Obstacle Avoiding Robot) will be explained with all details.

Chapter Three
Proposed Car Sensing System
(Obstacle Avoiding Robot)

In robotics, obstacle avoidance is the task of satisfying some control objective subject to non-intersection or non-collision position constraints. In unmanned air vehicles, it is a hot topic. What is critical about obstacle avoidance concept in this area is the growing need of usage of unmanned aerial vehicles in urban areas for especially military applications where it can be very useful in city wars. Normally obstacle avoidance is considered to be distinct from path planning in that one is usually implemented as a reactive control law while the other involves the pre-computation of an obstacle-free path which a controller will then guide a robot along.

Proposed System:

The proposed car sensing system known as "Obstacle Avoiding System" is shown in the flowchart in figure 12 below. While the car moving forward, the ultra-sonic sensor on the car measure the distance from nearest obstacle. When this distance become less than 50 Cm, the car stop and move backward for 5 minutes and the servo motor also move along 0,45,135,180 degrees and the ultra-sonic sensor measure the largest distance and turns the car through the direction of this angle and move forward and then the overall process returned.

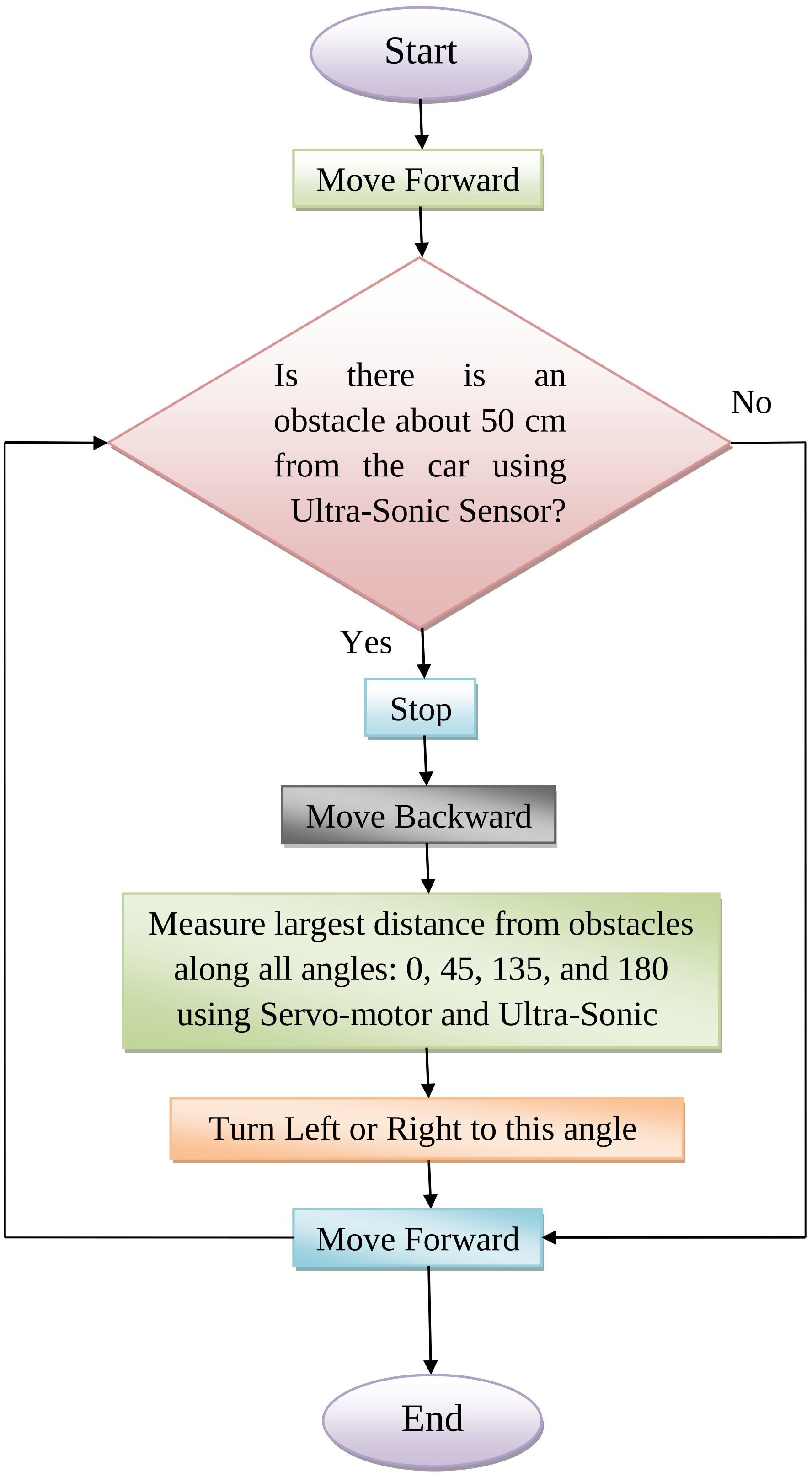

Start
Move Forward
Is there is an obstacle about 50 cm from the car using Ultra-Sonic Sensor?
No
Yes
Stop
Move Backward
Measure largest distance from obstacles along all angles: 0, 45, 135, and 180 using Servo-motor and Ultra-Sonic
Turn Left or Right to this angle
Move Forward
End

Figure 12: The flowchart of the proposed system.

The car in this proposal is shown in figure 13 below:

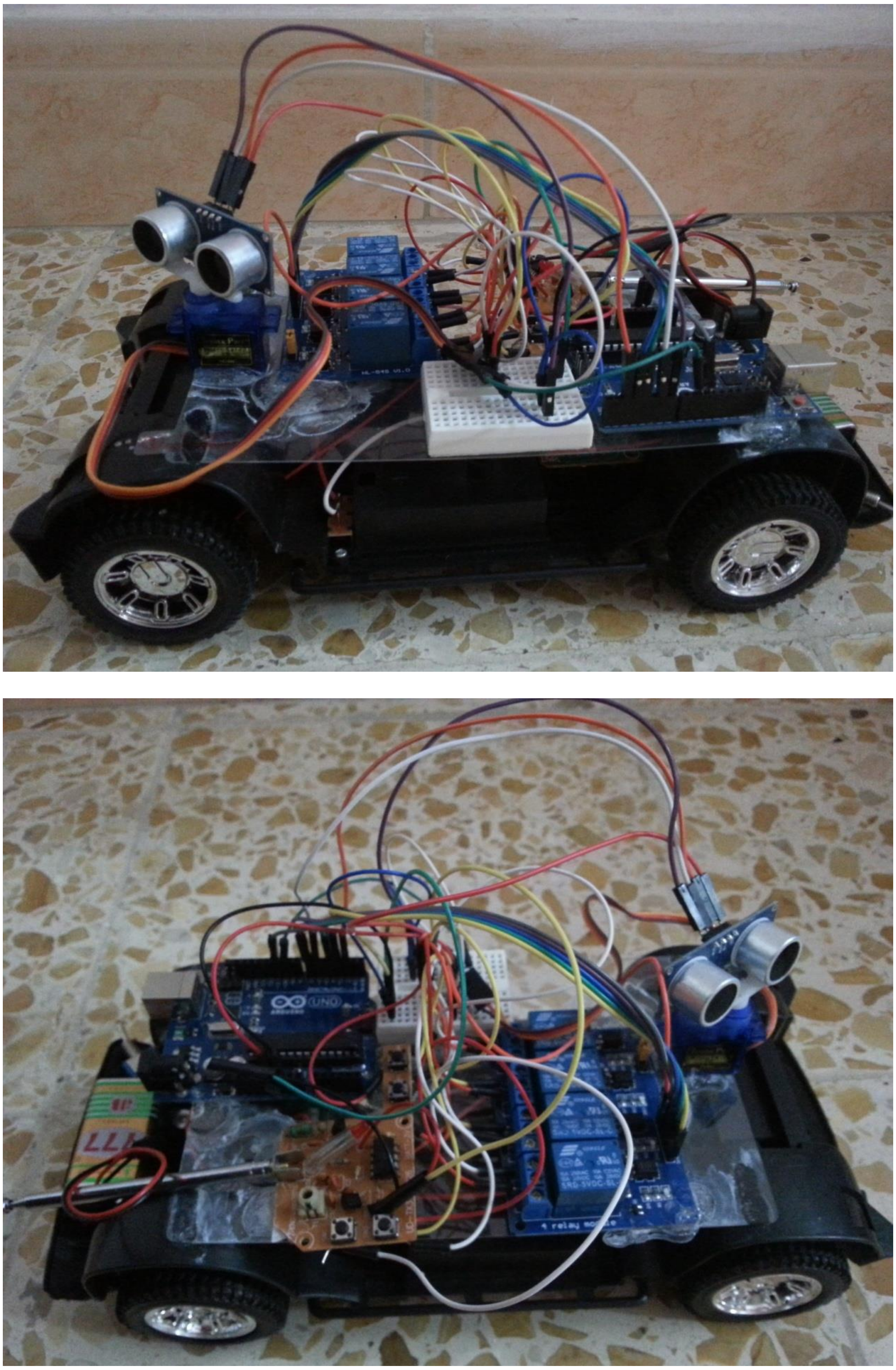


Figure 13: The proposed car.

The proposal uses the following parts:

A) Hardware Part:

The proposal hardware consists from the following:

1) Uno Arduino Board:

The Arduino board (shown in figure 14 below) is used to receive the Ultra-Sonic signal and give an a suitable order to relays that control four push-buttons that are used to control the double motors of the car that turns the car to the suitable direction.

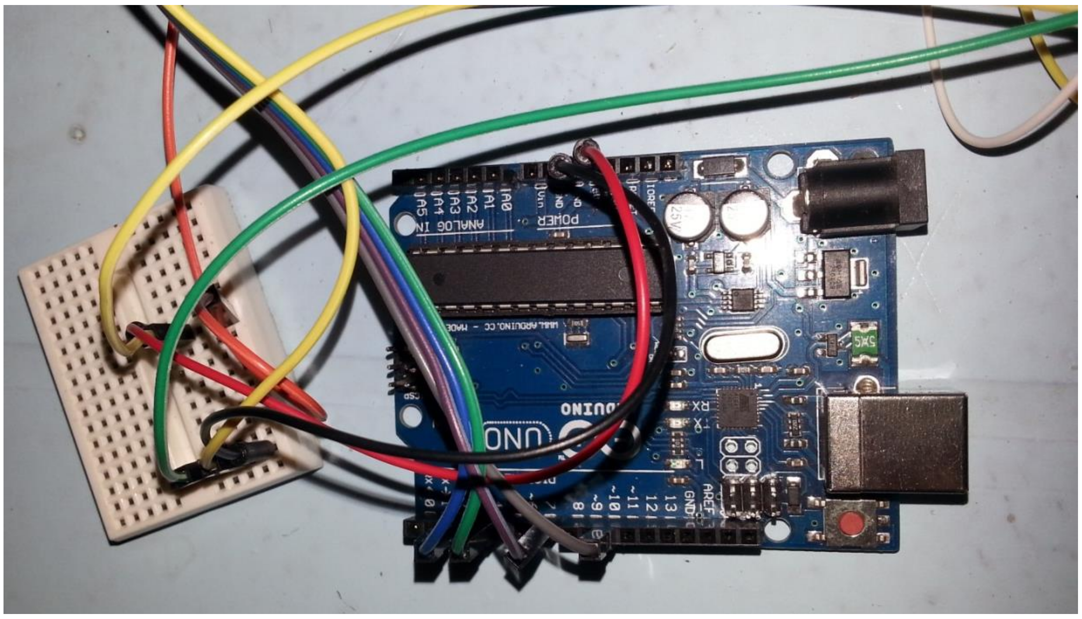


Figure 14: The Used Uno-Arduino Board.

2) The car with two motors:

The small car with two DC motors (shown in figure 15) is used to do this proposed system.

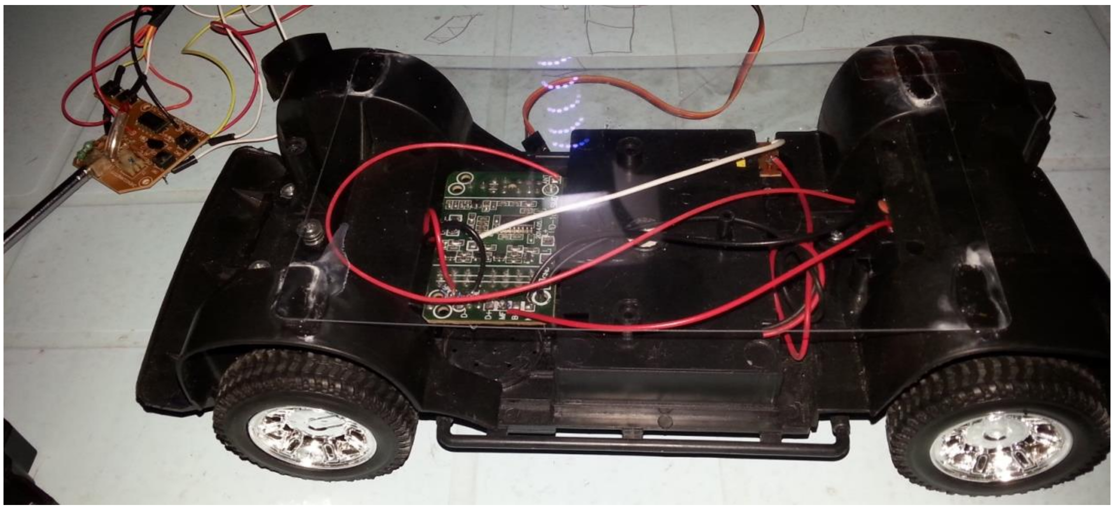


Figure 15: The Used Car.

3) Ultra-Sonic Sensor:

The Ultra-Sonic sensor (shown in figure 16) is used in this proposal to measure the distance from any obstacle in order to take the decision to stop the car and move it backward then turn the car left or right according to the sensor reading.

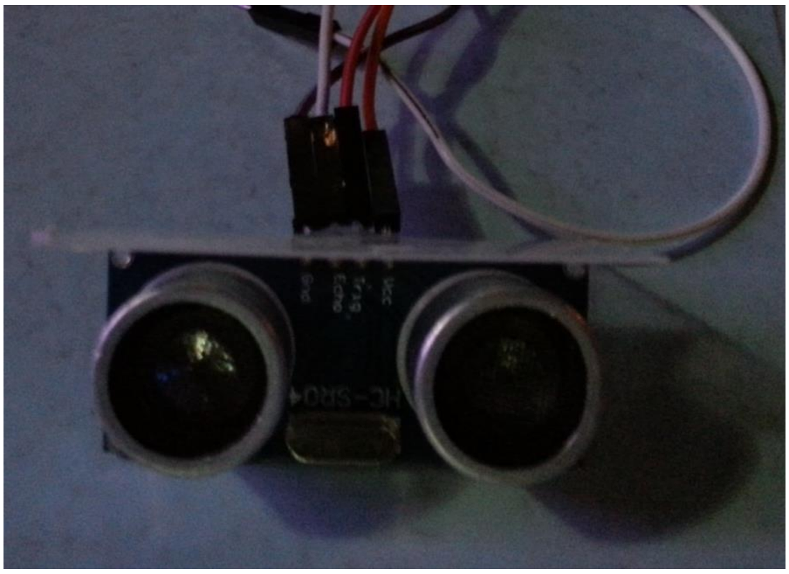


Figure 16: Used Ultra-Sonic sensor.

4) Four Relays Board:

In order to turn the double DC car motors forward, backward, left or right according to the Arduino order due to the Ultra-Sonic sensor reading, four relays (shown in figure 17) are used in this proposal.

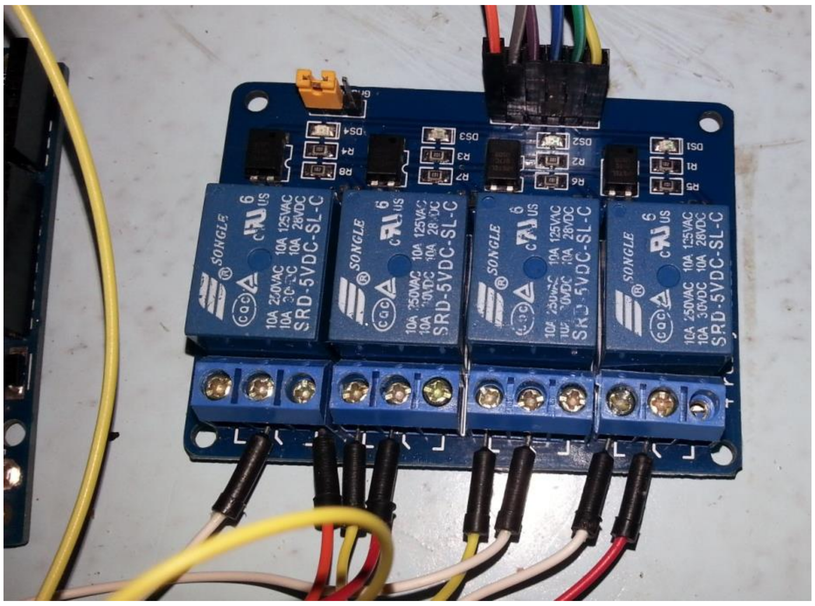


Figure 17: Used Four Relays Board.

5) Transceiver Remote Control Board:

The remote control transceiver with antenna (shown in figure 18) is used in order to send the orders wireless between the relays and the car.

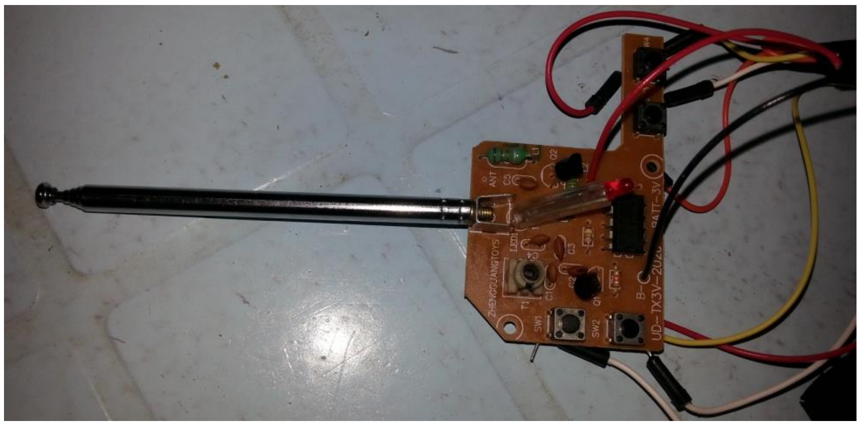


Figure 18: Used Transceiver.

6) Servo-Motor:

The Servo-motor (shown in figure 19) is used to turn the sensor around the four angles to select the suitable distance and turns the car towards it.

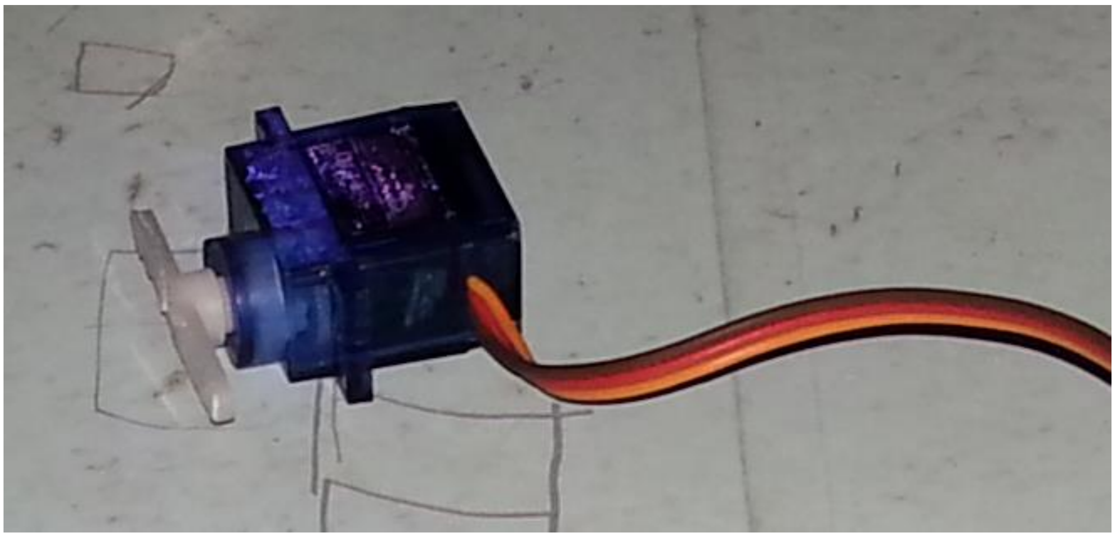
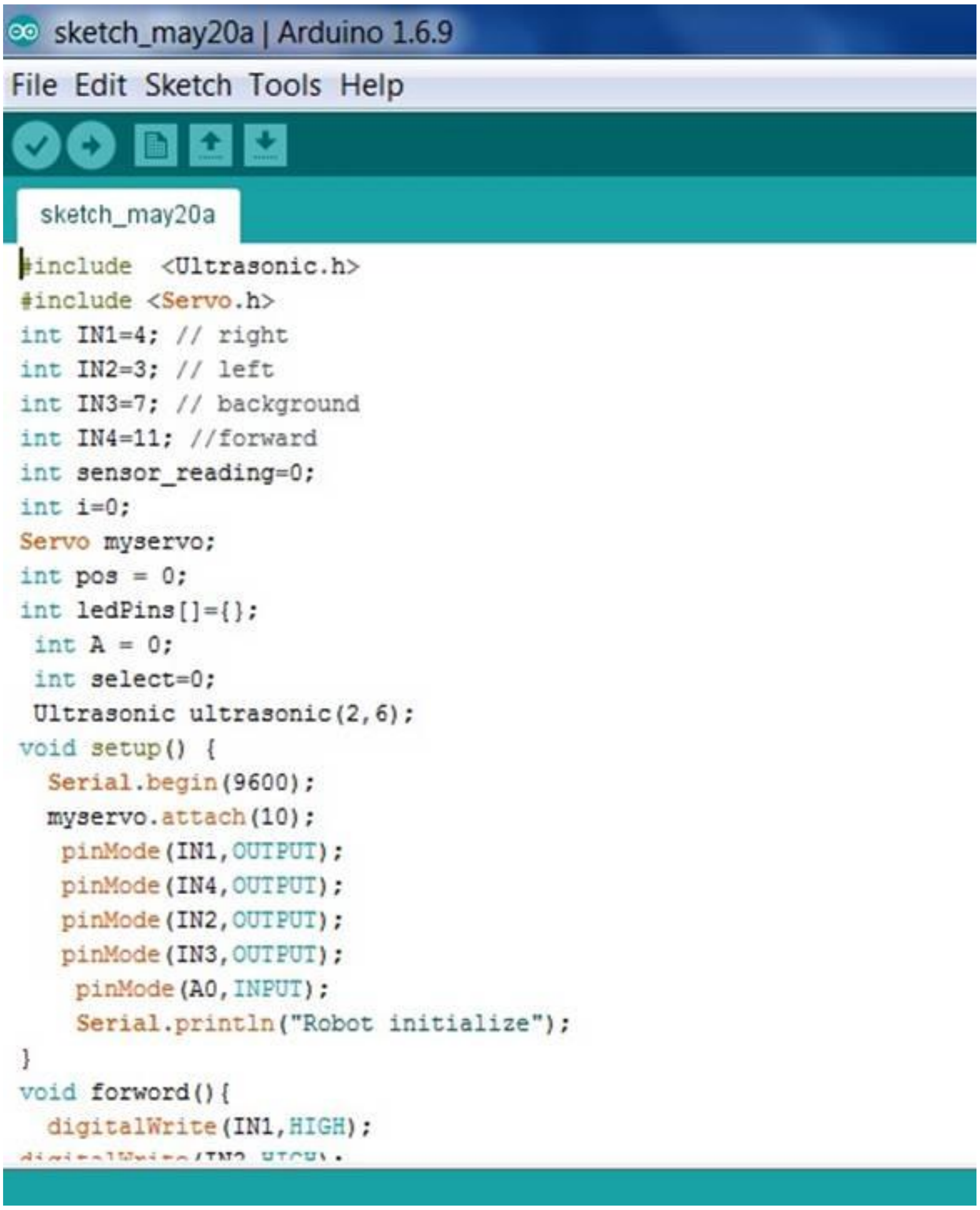


Figure 19: Used Servo-Motor.

B) Software Part:

The Used Arduino need Arduino C language to program the proposed system, therefore; Appendix in the following shows the used Arduino C code which is shown in figure 20 below:



```
#include  <Ultrasonic.h>
#include <Servo.h>
int IN1=4; // right
int IN2=3; // left
int IN3=7; // background
int IN4=11; //forward
int sensor_reading=0;
int i=0;
Servo myservo;
int pos = 0;
int ledPins[]={};
 int A = 0;
 int select=0;
 Ultrasonic ultrasonic(2,6);
void setup() {
   Serial.begin(9600);
  myservo.attach(10);
    pinMode(IN1,OUTPUT);
    pinMode(IN4,OUTPUT);
    pinMode(IN2,OUTPUT);
    pinMode(IN3,OUTPUT);
      pinMode(A0,INPUT);
      Serial.println("Robot initialize");
}
void forword(){
   digitalWrite(IN1,HIGH);
```

Figure 20: Arduino C Software code.

Conclusions:

1- According to the rapid development in the world, the modern automatic control systems was proposed to make the future life more easy.

2- The new technologies are used now and in the future according to the personal cars to make it more easy.

3- The proposed system used in this work uses a new technology to facilitate the driving modern cars.

4- In the future, GPS will be used with this technology to control the car automatically without using personal tools.