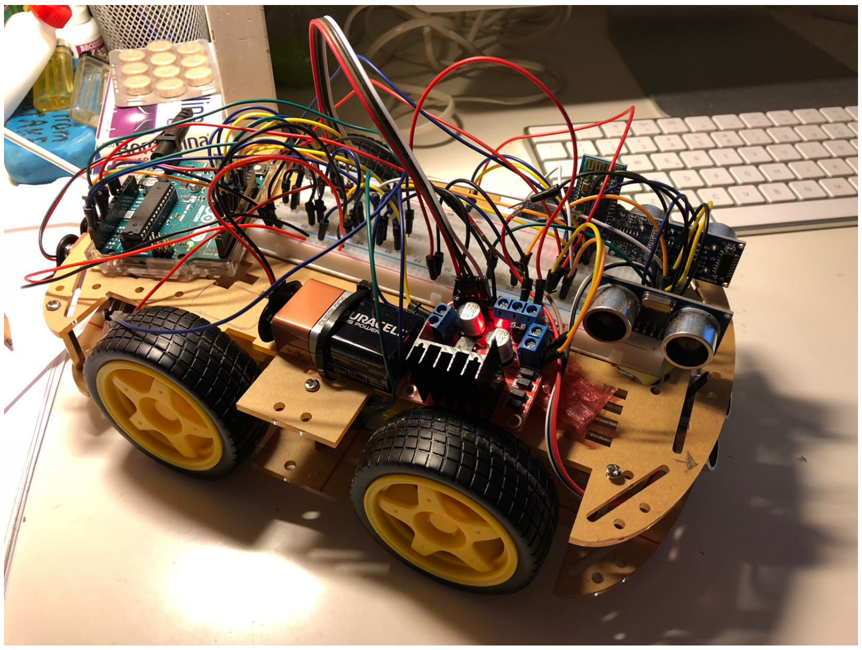
Lamborguino

Giulio Candreva, Filippo Malgarini, Mattia Pagetti

Febbraio 2018

Indice

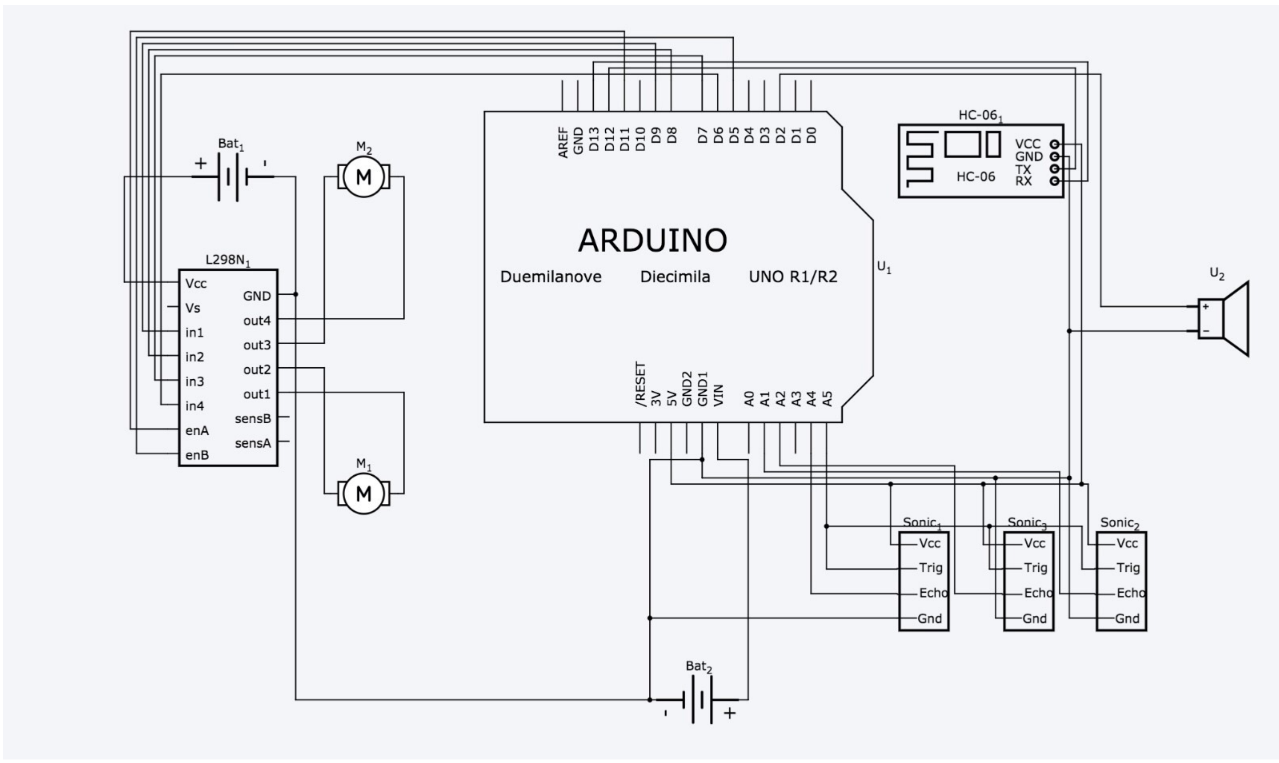


1 Introduction

Our project is a machine which can be driven by bluetooth protocol (or anyway controlled). We have implemented three possible way to use it: allowing the user to directly drive it, setting a self drive mode or use it as an alarm clock (and thus by communicating the hour at which the self drive mode, jointly with a buzzer, should automatically start).
In the first mode the machine movement ("avanti", "indietro", "curva a destra", "curva a sinistra", "stop") will follow the instruction given by the user via bluetooth. In the self drive mode, always activated by the user through bluetooth, three sensors are used in order to detect obstacles in the machine's path: given this informations, the machine will automatically choose which movement will be the next one to be actuate.
The alarm mode, as told before, is a combination between the self drive mode and a buzzer. To quit the self drive mode and the alarm mode we'll have to reset Arduino.
Of course, the main point if the problem was to use an Arduino board - writing in its environment the code which allows the machine to actually work.

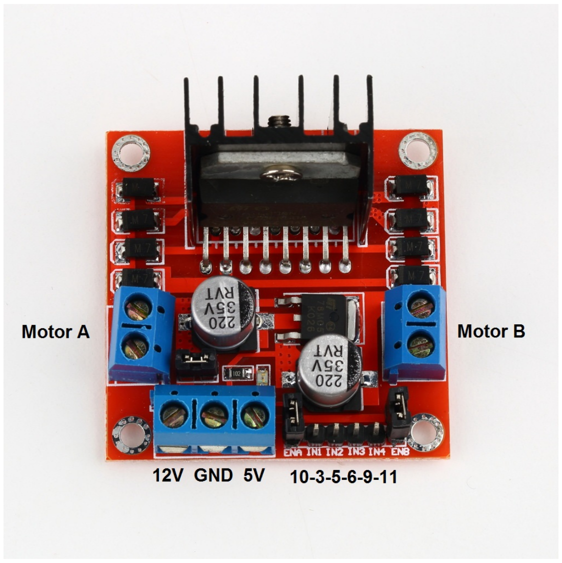
2 Comments

All the codes that we have written and used are available on Github.
Here is also present a README file, which is a recap of the functions purpose.
In first place, our idea was to implement the alarm clock mode only. The first thing we've done was to write the program necessary to move the wheels, and thus our first problem was to understand how to interface the engine L298N to Arduino and to the wheels' engines. We had some problem while dealing with it: on the board on which all the engine components were weldt, in fact, there were defective contacts. That has been particularly tedious since the problem itself can be mistaken with a software problem, a bug. This brought us, more than one time, to search for a problem in the code when the malfunction of the car was due to the defective engine hardware. In order to test our code, anyway, we relied on some simple test functions, which can be found on github as "MachineFunctions.ino".
The next step was to implement the bluetooth module. The function used to test it undergoes the name of "BT-HC-06-test" on github. After that, contemplate the car to be directly driven by the user has been a spontaneous evolution. We only had to make accessible to the user functions that we'd had to write in any case.
The sensors have been in a certain way challenging. We had some serious difficulties not in the wiring or in the code development, but in the precision which they allow us to reach. The sensors was cheap models, and thus not so much reliable. Many times we obtained very small distances (and in particular small enough to trigger the alert signals occurring in the code) even if no obstacle was present on the path of the car. To remedy to this problem, we wrote the code in such a way that the alert signals are triggered only when three small distances in a row (below 20cm) are obtained. The function "ping-distance-test" has been written to test the sensors' code.
Again, since the self drive mode was a piece of our original idea of an alarm clock machine, to make the user able to activate it trough bluetooth was just a question of write few lines of code; and that's what we have done.
Remarkable has also been the power consume of the machine. Eventually, we needed to guarantee it three batteries, 9V each.



3 Components, Wiring, Programs

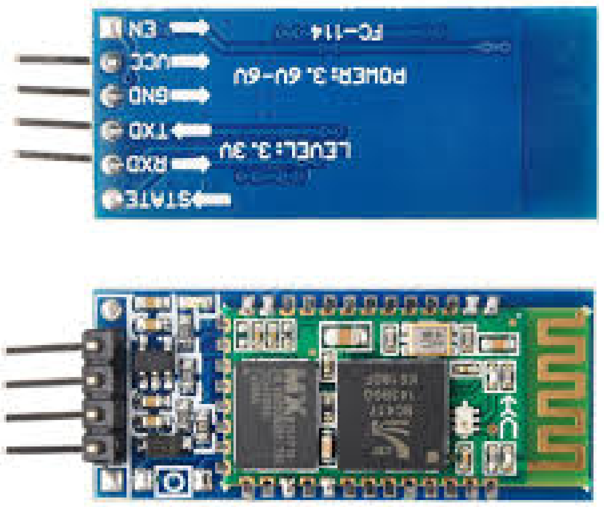
Above we have the wiring scheme. Now we're going to introduce each components in a dedicated subsection.



3.1 Engine L298N Dual H Bridge DC

The engine was absolutely necessary in order to handle the wheels motion at our will. The engine has six signals as input: four to handle the direction of the motion of the wheels (forward or backward) and two to set their velocity. As output, it has two couples of signals (every couple has to be linked to the upper or lower attaches of the wheels' engines): thus imposed us to control two couples of wheels (that is to handle jointly two wheels per time). To make it easy: we right wheels had to move in the same way, such as the left ones. The movement "curva a sinistra" is obtained by moving the right wheels forward and the left wheels backward; "curva a destra" is obtained specularly.
Pins 6,7,8 and 9 have been used to set the motion direction, pins 5 and 11 to set the velocity. As said before, the test function "MachineFunctions.ino" has been written on the purpouse to verify the if the correct motion was executed.

.



3.2 Bluetooth DSD Tech HC-05

This component has two pins for the alimentation (one lniked to the ground and one linked to a 5V source) and two pins to obtain the signal: one for transmission (TX) and one for reception (RX). Obviously, on Arduino we'll have the same pin, placed symmetrically: RX talks to TX and the other way around. We use pin 12 for RX and 13 for TX. According to the word sent to Arduino through bleutooth, the user will activate a certain mode (or will give indications about the direction which the machine has to take). It's thus a problem of identifying certain strings of letters. Also in this case we have written a test function.

.

3.3 Aukru HC-SR04

We used three of these components, one for every direction in which we were interest to detect the presence of obstacles. Basically, they are supersonic sensors: when they receive a trigger signal they create a sonorous signal and an high signal outogoing from the ECHO pin. When they receive the return of the sonorous signal, the high signal outgoing from ECHO becomes low again. What we do is to take note of the interval of time in which the ECHO signal has been high and to divide it for the velocity of the sound and for a factor 2. In this way we obtain the distance of the closer object, and this is what we are interested to receive when we call the function which activates the sensors. Of course, to the bluetooth module we'll also have to connect the 5V source and the ground. The Arduino pin used to generate the trigger signal is the A5, and is the same for every sensor. To receive the signals we instead use pins A1, A2, A4.

.

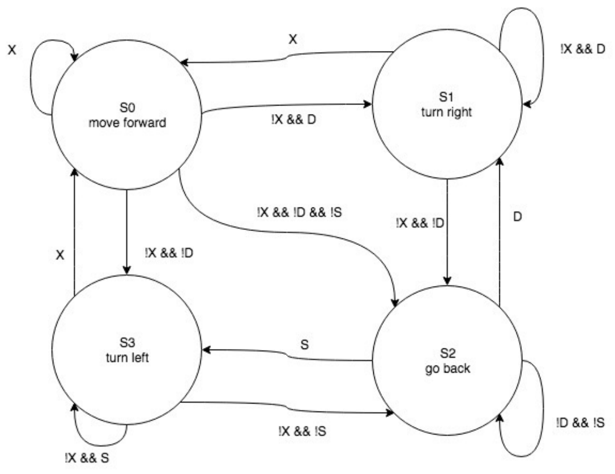


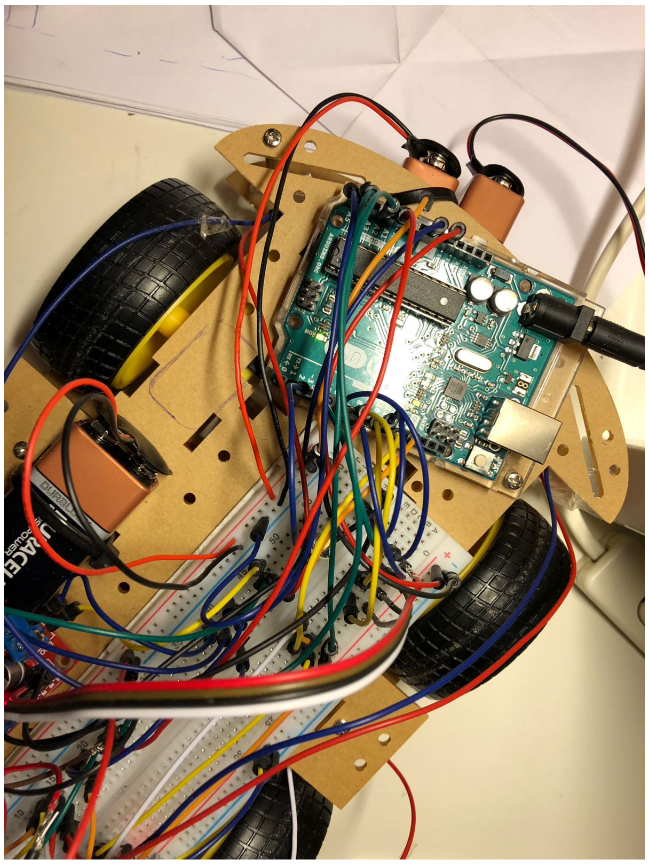
Figura 1: FSM

3.4 Arduino and Main Program

The first task of the program is to periodically control the communication trough the bluetooth module to identify the mode in which the user is interested to use the machine in (this doesn't happen if we are already in self drive or alarm clock mode). This is implemented by the call of the function "parsCommand" whenever the user put "!" at the end of a string.
If the command is "Avanti!", "Indietro!", "destra!", "sinistra!" or "stop!", the system simply execute the function associated to that command until a new instruction is received.

When the received instruction is "drive!", th car enters the self drive mode. The self drive logic is implemented using a finite state machine model, with four possible states and three inputs: the signals X,S,D. These signals are just boolean variables generated by the function 'getDangers()', which reads tha values from the proximity sensors and set the three variables X,S,D to TRUE whenever the respective direction is free from obstacles (the three directions are forward, left, right, respectively).

A direction is considered free from obstacle if the proximity sensor reads a value greater than 30cm. The FSM diagram is shown in figure 1.
To each state is associated a movement (forward, backward, to the left, to the right), which is performed through a function ("call"), which has the current state as only input.
The implementation of the FSM is obtained by a function, nextState, which have as input the actual state and the actual X,S and D signals, and as output the next state.



The alarm mode clock is called when we receive, through bluetooth, a command in the form "sveglia 00 50", where the numbers are the minutes and the hours which the user desire to let pass before the activation of the alarm clock. This mode consists in the activation of a buzzer and in the automatic selection of the self drive mode. The time is reckoned by "for" cycles endued with 1000ms delays.