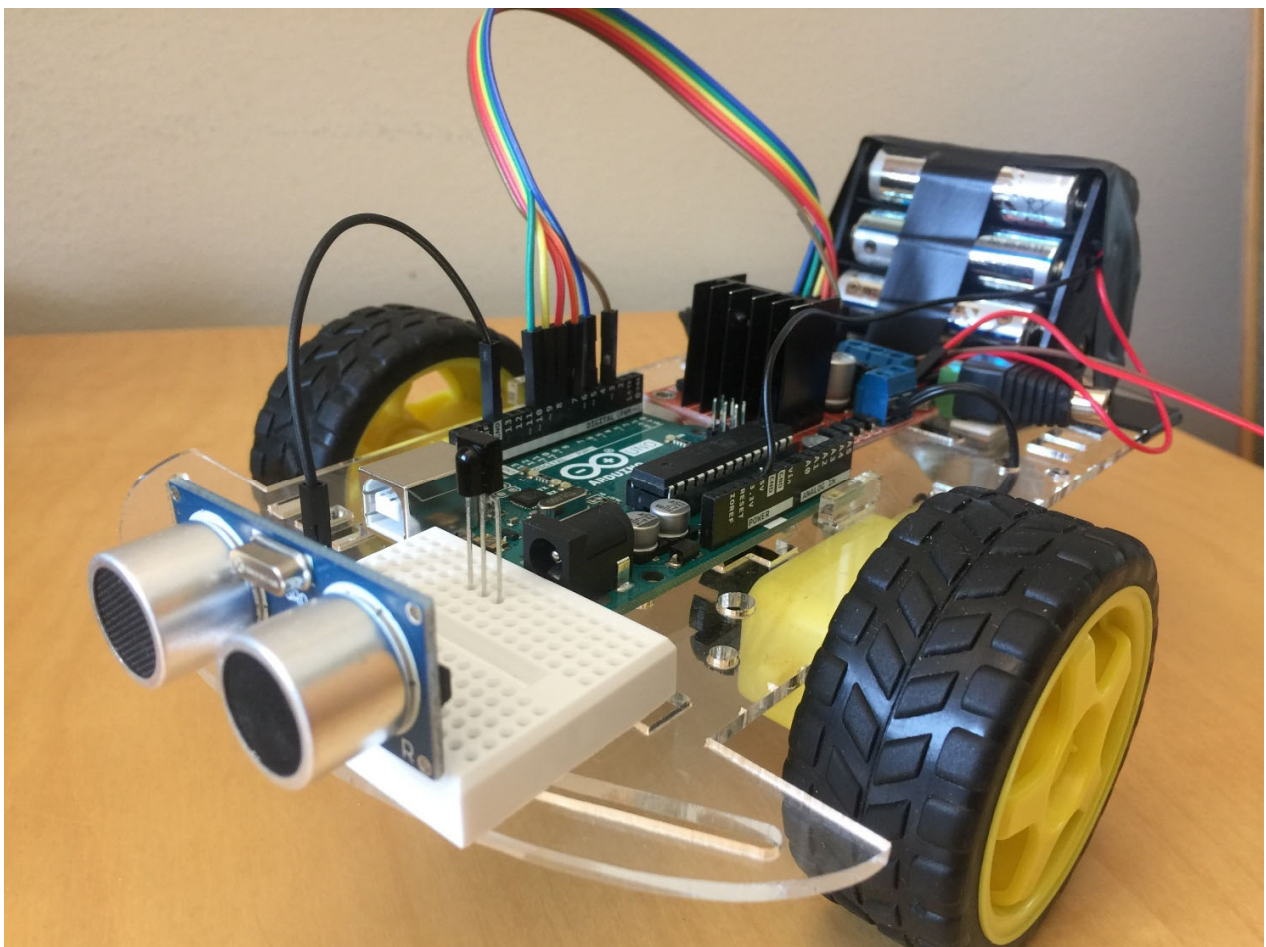


# Lab 5: Develop a Remote Controlled Car

Farhang Nemati



In this lab you will develop a simple car. You will practice towards development of a complex real-time embedded application. You can reuse the concepts, code, and practices that you have achieved in previous labs whenever it is possible.

## A. Controlling Motors

A Motor Driver Board (MDB) is installed on the car platform that can control two motors independently (Figure 1). The MDB has an input for power supply (5V or 12V, and GND – in Figure 1, the blue part in the front), 2 parts where each of them controls one motor (2 blue parts on the sides).

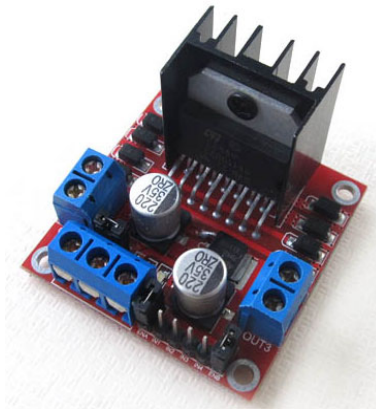
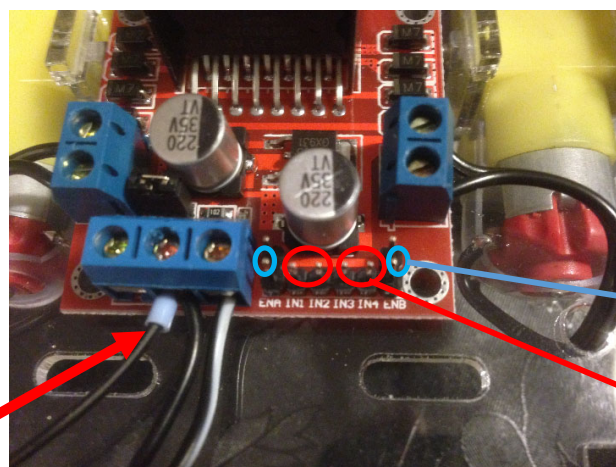


Figure 1

The motor driver has an input port which consists of 6 digital pins; 3 pins for each motor (Figure 2). The 2 pins at two ends (the left most and right most pins) are for controlling the speed of their corresponding motors. These pins have to be connected to an output PWM pin. Next to each PWM pin there are 2 pins that are used for setting the direction of their corresponding motor. To run a motor in a direction one of its 2 direction pins has to be high and the other one has to be low. The motor runs in different directions depending on which of its direction pins has high value. You have to connect the ground of MDB to a ground pin of Arduino



To a ground pin of Arduino

Figure 2

PWM Pin for a motor

Direction Pins for the motor

## B. Distance Sensor

In this section you will learn to work with an ultra-sonic sensor to measure distance to an object. A HC-SR04 sensor that is capable to measure distances in range 2cm to 400cm.

As shown in Figure 3, it has only 4 pins (Vcc, Trig, Echo, and Gnd). Vcc pin has to be connected to 5volt supply, Gnd will be connected to the ground pin of Arduino. Pins Trig and Echo have to be connected to output and input digital Arduino pins respectively.

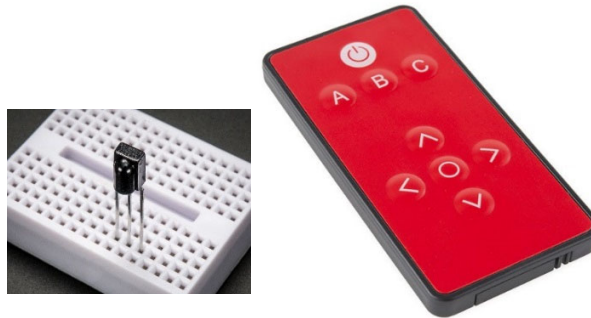
To measure the distance a pulse with the length of at least 10 microseconds (us) has to be sent out on Trig pin. This means that trig pin has to go from LOW to HIGH stay at HIGH for at least 10us and then go back to LOW. Use `delayMicroseconds (unsigned int us)` to delay in us microseconds. After sending the pulse on Trig, you have to measure the length of incoming pulse on Echo. Use `pulseIn(pin, value)` to measure a pulse arrived on pin. Parameter *value* can be HIGH or LOW, e.g., if HIGH it will return the length of the pulse during which the pulse arrived on Echo was HIGH. The returned value is the length of the pulse in microseconds. The returned value is the time during which the sound travels from sensor to an object and back to the sensor. Considering that the velocity of sound is 340 m/s you can measure the distance to the object in cm (centimeters).

Implement a task that periodically measures the distance and puts the calculated distance into a global variable. Later you will use this variable in the complete program. To test the measured distance, print out the distances on Serial port and observe if it measures distances correctly.



Figure 3: Distance sensor

## C. Infrared (IR) Receiver and IR Remote Control



In this section you will learn to work with an IR remote control (COM-14865) and an IR receiver (TSOP38238) to be able to send commands to Arduino for controlling the car remotely. The IR receiver and IR remote control are shown in Figure 4. To be able to work with IR receiver in Arduino IDE you have to first install *IRremote* library. To do this open “Library Management” from “Tools ->Manage Libraries”. In the filter box search for IRremote and install the library. In your program include IRremote.h header file.

**Notice:** You must install the library with version 2.6.1 or OLDER, preferably version 2.2.3. The newer versions are not appropriate for the Arduino UNO with limited resources.

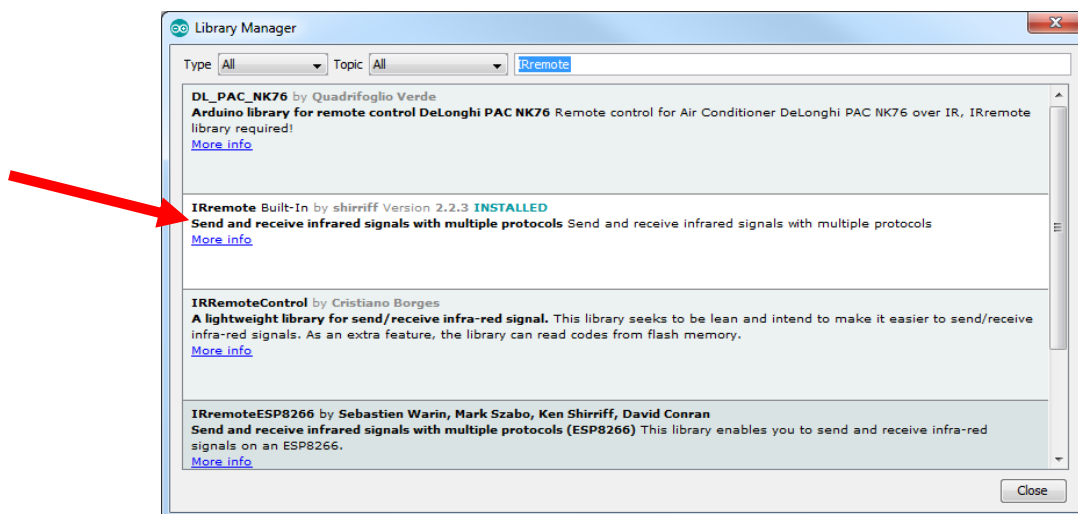


Figure 5: Install IRremote library.

The IR receiver has 3 pins (Figure 6), pin 2 should be connected to ground, and pin 3 should be connected to 2.5 to 5volt supply. Pin 1 should be connected to a digital pin of Arduino where the program reads from.

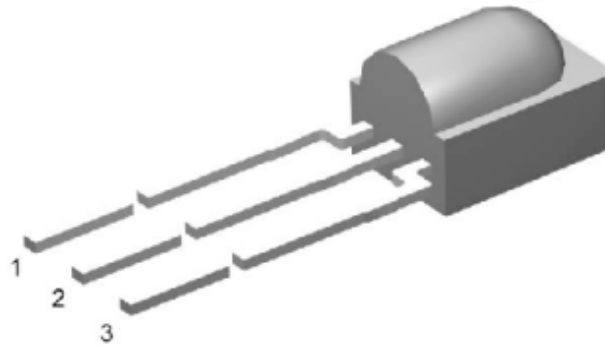


Figure 6: IR Receiver pin: 1 = OUT, 2 = GND, 3 = VS

In your code you have to create a receiver object first:

```
IRrecv receiver(RECV_PIN);
```

Where RECV\_PIN is the digital pin number on Arduino.

Then you should start the receiver:

```
receiver.enableIRIn();
```

You should also declare a variable of type *decode\_results* to store the received command code in it:

```
decode_results results;
```

Now the receiver is ready to get data. If it receives any infrared commands the following command will return true and put the code of the received command in parameter *results*:

```
receiver.decode(&results);
```

By checking the code (**results.value**) you can find out what was the command. For example, to test the received code you can print it out in hexadecimal format:

```
Serial.println(results.value, HEX);
```

After reading the command, call the following function to resume the receiver:

```
receiver.resume();
```

If you are using your own IR Remote control, using the instructions mentioned above, you can find out what codes are sent by pressing each of the buttons on your remote control. The remote control that you have got with the lab kit will send the codes shown in Figure 7 (and shown in the following programming piece of code). When you press a button on the remote control it will send the button's code and if you hold button after sending its code it will keep sending **0xFFFFFFFF** until you release the button.

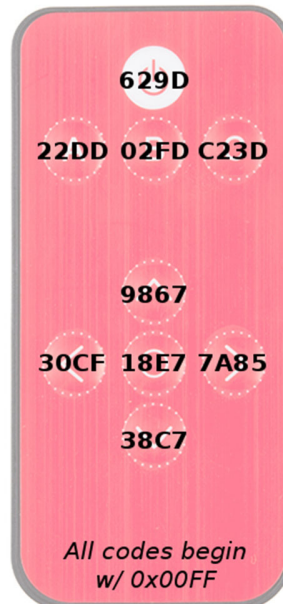


Figure 7: Remote control button codes.





```
#define POWER 0x00FF629D
#define A 0x00FF22DD
#define B 0x00FF02FD
#define C 0x00FFC23D
#define UP 0x00FF9867
#define DOWN 0x00FF38C7
#define LEFT 0x00FF30CF
#define RIGHT 0x00FF7A85
#define SELECT 0x00FF18E7
```

**Notice:** With the current settings of IRremote library (the timer settings) it has conflicts with PWM pin 11 on Arduino Uno. Try not to use pin 11 as PWM while using IR receiver.



## D. The Assignment: Put everything together

In this part you will write a complete program for your Arduino that using the remote control controls the car with different speeds and directions. Connect the PWM and direction pins of each motor (if they are not already connected). You should also connect the pins of the distance sensor and IR sensor as explained in previous sections.

Use the arrow buttons (     ) on the remote control to move the car forward, backward, turn left, and turn right. Use buttons A, B, and C to set different speed profiles for the motors. If the button A is pressed the speed of the motors has to be set to around 50% of full speed. Button B sets the speed to 70%, and button C should set the speed to 100% (full speed). Depending on the speed profile (set by A, B, C buttons), the arrow buttons will move the car with the current speed profile.

The distance sensor continuously measures the distance to the object in front of the car. If distance between the car and the objects drops below a **critical distance**, the car must stop so that it does not hit the object. This means that the command that moves the car forward must be ignored as long as the distance to the object is below the critical distance. The critical distance differs for different speed profiles. For each speed profile find out the critical distance. For a speed profile, you can find the critical distance by trying different distances until you find a good enough distance for which the car does not hit the object. Then you can set your results as the critical distances for speed profiles.

The distance sensor and IR sensor should run in their own separate tasks. Both motors can be controlled using one and the same task. Depending on your design, feel free to have extra helping tasks. However, remember more tasks means introducing more complexity into the system. Thus use just enough number of tasks. Every resource (e.g., variables) shared by more than one task **MUST** be protected. It's recommended but not required to use queues for your system.

Enjoy!

## Examination

To pass the lab, you have to hand in your code on Blackboard and demonstrate the working program to the lab assistant. You have to be prepared to answer questions asked by the lab assistant related to the lab.