

# The DIY automobile project (Level 4)

Worksheet for students

Team:.....

**Aim: Controlling the DIY automobile remotely**

What can be used to control DIY automobile remotely? Think about it and search for information online. Write your answers below.

How Bluetooth works? Search for information online and write your answers below.

Review several scenarios, how DIY-automobile steering remotely can be build, and choose one to demonstrate. Sketch your DIY-automobile and list the crafting materials that you may need.

*Area for sketches*

*Bill of materials*

### Time for circuit making!

During this level, the DIY automobile will be remotely controlled by mobile phone with Android system using Bluetooth technology. First, the Bluetooth module HM-06 should be connected to the Arduino Sensor Shield according to the table 1.

HM-06 module	Arduino Sensor Shield
RXD	3
TXD	2
GND	G
VCC	V

Table 1: Connection of a HM-06 Bluetooth module with an Arduino Sensor Shield (pinout).

### Time for hands-on practice!

Let's create the circuit using your Arduino board and the corresponding electrical components.

Next, the RemoteXY library should be installed on your computer. Follow the instructions below to install the library:

1. Download the RemoteXY library from website <https://remotexy.com/en/library/>
2. Unzip the archive with RemoteXY library inside *Arduino*→*libraries* folder.
3. Run the Arduino IDE software.

The RemoteXY library allows to create an user interface on your mobile phone. You can design interface by dragging and dropping different kinds of buttons from *Elements* section to the virtual mobile phone on website <https://remotexy.com/en/editor/> as it is shown in figure 1. Every button has its own variable, which name can be modified in *Properties* section. You can also change the color of a button and the text on a button.



Figure 1: The RemoteXY website editor.

## Time for programming!

Connect your Arduino to USB.

When you finish the design of the user interface, click on *Get source code* button. Then, you will see the source code of your user interface. This code will be integrated with your code from level 1. If you write code using mBlock, please copy the solution from Arduino IDE from attached document.

First, you can see the user interface configuration between *RemoteXY include library* and *END RemoteXY include* comments in the source code. Copy this part of code to your code from level 1 at the beginning (after including libraries). The *SoftwareSerial* library is included twice, therefore remove one of them, e.g.:

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h> Remove the repeating library

////////////////////////////////////
// RemoteXY include library //
////////////////////////////////////

#define REMOTEXY_MODE_SOFTSERIAL
#include <SoftwareSerial.h>

#include <RemoteXY.h>

// RemoteXY connection settings
#define REMOTEXY_SERIALRX 2
#define REMOTEXY_SERIALETX 3
#define REMOTEXY_SERIALSPEED 9600

// RemoteXY configurate
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =
{ 255,4,0,0,0,43,0,10,13,0,
  1,0,43,11,12,12,2,31,87,0,
  1,0,43,27,12,12,2,31,83,0,
  1,0,23,27,12,12,2,31,65,0,
  1,0,63,27,12,12,2,31,68,0 };

// this structure defines all the variables and events of your control interface
struct {

// input variables
uint8_t W; // =1 if button pressed, else =0
uint8_t S; // =1 if button pressed, else =0
uint8_t A; // =1 if button pressed, else =0
uint8_t D; // =1 if button pressed, else =0

// other variable
uint8_t connect_flag; // =1 if wire connected, else =0
```

```

} RemoteXY;
#pragma pack(pop)

////////////////////////////////////
//                               END RemoteXY include                               //
////////////////////////////////////

void Motors_Forward ()
{
  Left_Motor_Forward ();
  Right_Motor_Forward ();
}

void Motors_Backward ()
{
  Left_Motor_Backward ();
  Right_Motor_Backward ();
}

void Left_Motor_Forward ()
{
  analogWrite (5,200);
  digitalWrite (6,1);
  digitalWrite (7,0);
}

void Left_Motor_Backward ()
{
  analogWrite (5,200);
  digitalWrite (6,0);
  digitalWrite (7,1);
}

void Left_motor_OFF ()
{
  analogWrite (5,0);
  digitalWrite (6,0);
  digitalWrite (7,0);
}

void Right_Motor_Forward ()
{
  analogWrite (11,200);
  digitalWrite (12,1);
  digitalWrite (13,0);
}

void Right_Motor_Backward ()
{
  analogWrite (11,200);
  digitalWrite (12,0);
  digitalWrite (13,1);
}

```

```

void Right_Motor_OFF ()
{
  analogWrite(11,0);
  digitalWrite(12,0);
  digitalWrite(13,0);
}

void Motors_OFF ()
{
  Left_motor_OFF ();
  Right_Motor_OFF ();
}

void turn_left ()
{
  Right_Motor_Forward ();
  Left_motor_OFF ();
}

void turn_right ()
{
  Left_Motor_Forward ();
  Right_Motor_OFF ();
}

void _delay(float seconds)
{
  long endTime = millis() + seconds * 1000;
  while(millis() < endTime) _loop ();
}

void setup()
{
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(13,OUTPUT);
}

void _loop()
{
}

void loop()
{
  _delay(1);
  Motors_Forward ();
  _delay(1);
  Motors_Backward ();
  _delay(1);
  Motors_OFF ();
  _loop ();
}

```

Then, copy rest of the commands:

- *RemoteXY\_Init()* to *setup* function. This function initializes user interface.
- *RemoteXY\_Handler()* to *loop* function, which checks state of the buttons, if they were pressed or not.

All the buttons have their own variables (see part of the source code under *input variables* comment), which are equal to 1 when a button is pressed. In this case, the variables are called W, S, A and D. The source code below checks the value of each variable. If the button is pressed, the appropriate command runs:

```
...  
  
void setup()  
{  
  pinMode(5,OUTPUT);  
  pinMode(6,OUTPUT);  
  pinMode(7,OUTPUT);  
  pinMode(11,OUTPUT);  
  pinMode(12,OUTPUT);  
  pinMode(13,OUTPUT);  
  
  RemoteXY_Init();  
}  
  
void _loop()  
{  
}  
  
void loop()  
{  
  RemoteXY_Handler();  
  
  if(RemoteXY.W) Motors_Forward();  
  if(RemoteXY.S) Motors_Backward();  
  if(RemoteXY.A) turn_left();  
  if(RemoteXY.D) turn_right();  
  
  _delay(1);  
  Motors_OFF();  
  
  _loop();  
}
```



#### Useful functions:

- `#include "name_of_library"` - this line allows to add library header to the source code,
- `analogWrite(pin, duty cycle)` - this function allows to generate the PWM (*Pulse Width Modulation*) wave on chosen pin number and with chosen duty cycle,
- `digitalWrite(pin,value)` - this function sets the chosen value (*HIGH* and *LOW* or 0 and 1) on digital pin,
- `millis()` - return the passed time after running the code in milliseconds.

#### Time for crafting!

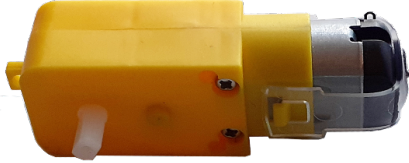
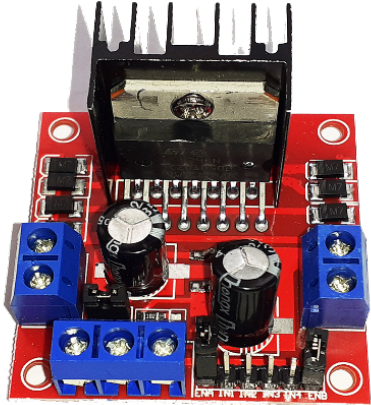
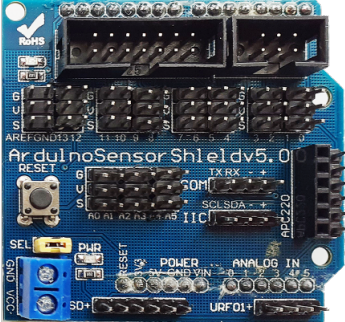
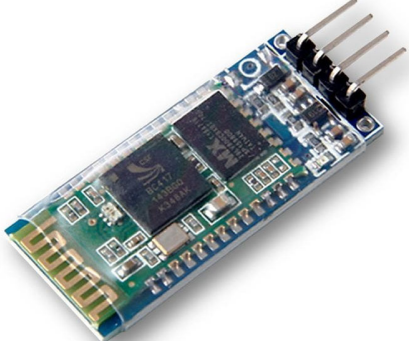
Start working on the design of the DIY-automobile using the available crafting materials.

Now, you can run your code on your DIY automobile. The last step is installation of the RemoteXY application on your mobile phone. You can find this application in *Google Play Shop*. Then pair your mobile phone with the Bluetooth module, which will be called *HC-06*. The password is *1234*.

Next, open the RemoteXY application on your mobile phone. Click a plus sign in the right, top corner, choose *Connect to Bluetooth device*, and select the *HC-06* device as follows. Then click on added device and now you can control your DIY automobile remotely. Enjoy!

## Electrical components

The following table is an index containing all the components that need to be implemented for accomplishing the present activity.

	DC Motor
	L298n driver
	Arduino Sensor Shield
	Bluetooth module HC-06

## **ROBOSCIENTISTS PROJECT**

Motivating secondary school students towards STEM careers through robotic artefact making

**Erasmus+ KA2 2018-1PL01-KA201-051129**

### **Creator**

Angelika Tefelska (WUT)

### **Declaration**

This report has been prepared in the context of the ROBOSCIENTISTS project. Where other published and unpublished source materials have been used, these have been acknowledged.

### **Copyright**

© Copyright 2018 - 2021 the Roboscientists Consortium  
All rights reserved.



This document is licensed to the public under a Creative Commons Attribution- Noncommercial-ShareAlike 4.0 International License.

### **Funding Disclaimer**

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.