Official Arduino Robot Information

Review:

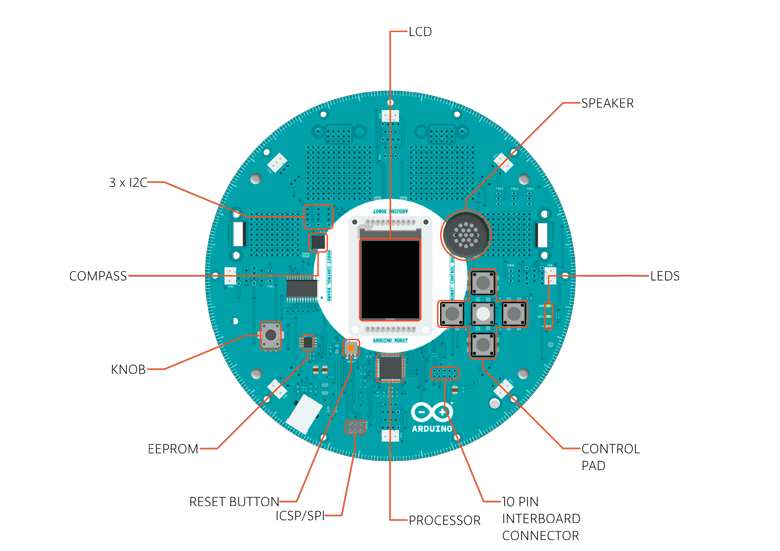
<http://robotshop.com/letsmakerobots/arduino-robot-review>

<https://www.arduino.cc/en/main/robot>

Technical specs

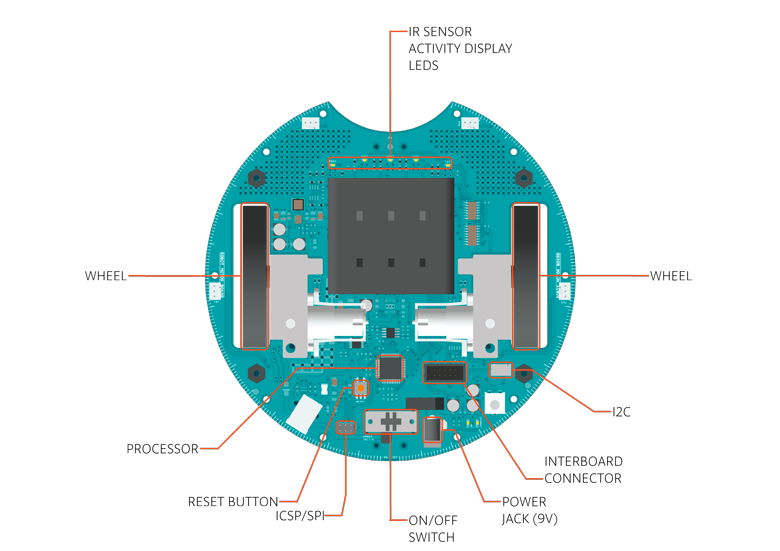
Control Board Summary

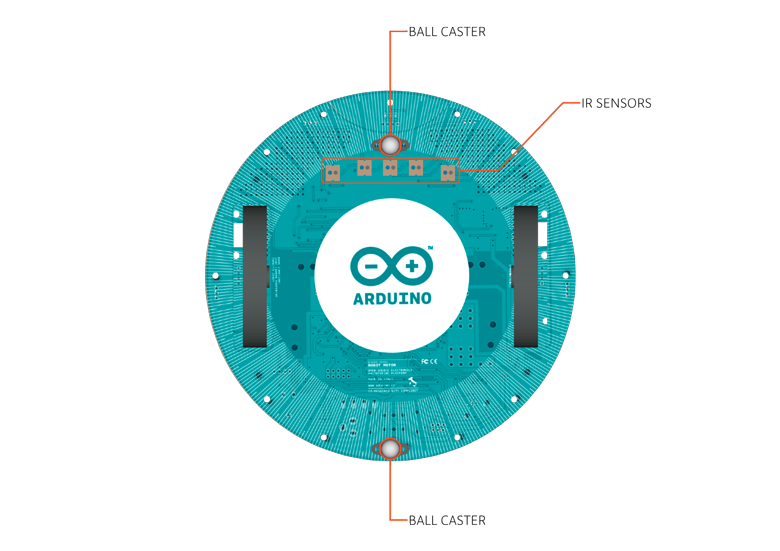
| Microcontroller | [ATmega32u4](http://www.atmel.com/Images/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf) |
| --- | --- |
| Operating Voltage | 5V |
| Input Voltage | 5V through flat cable |
| Digital I/O Pins | 5 |
| PWM Channels | 6 |
| Analog Input Channels | 4 (of the Digital I/O pins) |
| Analog Input Channels (multiplexed) | 8 |
| DC Current per I/O Pin | 40 mA |
| Flash Memory | 32 KB (ATmega32u4) of which 4 KB used by bootloader |
| SRAM | 2.5 KB (ATmega32u4) |
| EEPROM (internal) | 1 KB (ATmega32u4) |
| EEPROM (external) | 512 Kbit (I2C) |
| Clock Speed | 16 MHz |
| Keypad | 5 keys |
| Knob | potentiomenter attached to analog pin |
| Full color LCD | over SPI communication |
| SD card reader | for FAT16 formatted cards |
| Speaker | 8 Ohm |
| Digital Compass | provides deviation from the geographical north in degrees |
| I2C soldering ports | 3 |
| Prototyping areas | 4 |
| Radius | 185 mm |
| Heigth | 85 mm |



Motor Board Summary

| Microcontroller | ATmega32u4 |
| --- | --- |
| Operating Voltage | 5V |
| Input Voltage | 9V to battery charger |
| AA battery slot | 4 alkaline or NiMh rechargeable batteries |
| Digital I/O Pins | 4 |
| PWM Channels | 1 |
| Analog Input Channles | 4 (same as the Digital I/O pins) |
| DC Current per I/O Pin | 40 mA |
| DC-DC converter | generates 5V to power up the whole robot |
| Flash Memory | 32 KB (ATmega32u4) of which 4 KB used by bootloader |
| SRAM | 2.5 KB (ATmega32u4) |
| EEPROM | 1 KB (ATmega32u4) |
| Clock Speed | 16 MHz |
| Trimmer | for movement calibration |
| IR line following sensors | 5 |
| I2C soldering ports | 1 |
| Prototyping areas | 2 |





Power

The Arduino Robot can be powered via the USB connection or with 4 AA batteries. The power source is selected automatically. The battery holder holds 4 rechargeable NiMh AA batteries.

NB : Do not use non-rechargeable batteries with the robot

For safety purposes, the motors are disabled when the robot is powered from the USB connection. The robot has an on-board battery charger that requires 9V external power coming from an AC-to-DC adapter (wall-wart). The adapter can be connected by plugging a 2.1mm center-positive plug into the Motor Board's power jack. The charger will not operate if powered by USB. The Control Board is powered by the power supply on the Motor Board.

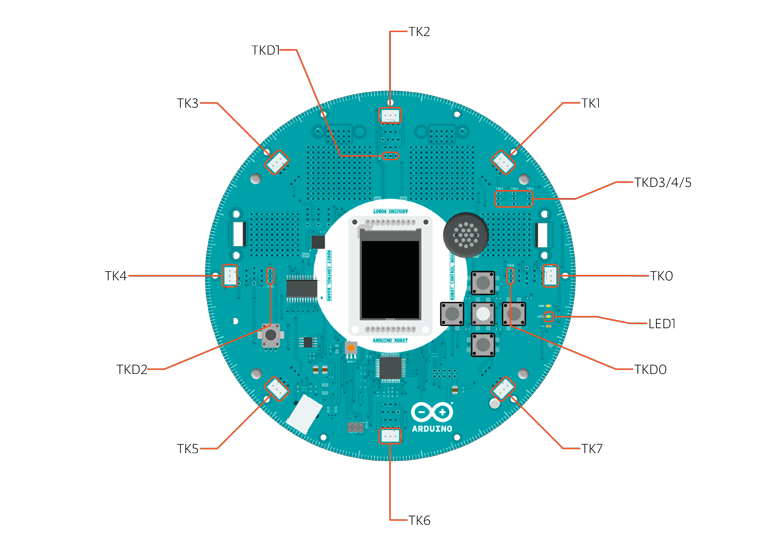
Memory

The ATmega32u4 has 32 KB (with 4 KB used for the bootloader). It also has 2.5 KB of SRAM and 1 KB of EEPROM (which can be read and written with the [EEPROM library](http://www.arduino.cc/en/Reference/EEPROM)). The Control Board has an extra 512 Kbit EEPROM that can be accessed via I2C. There is an external SD card reader attached to the GTFT screen that can be accessed by the Control Board's processor for additional storage.

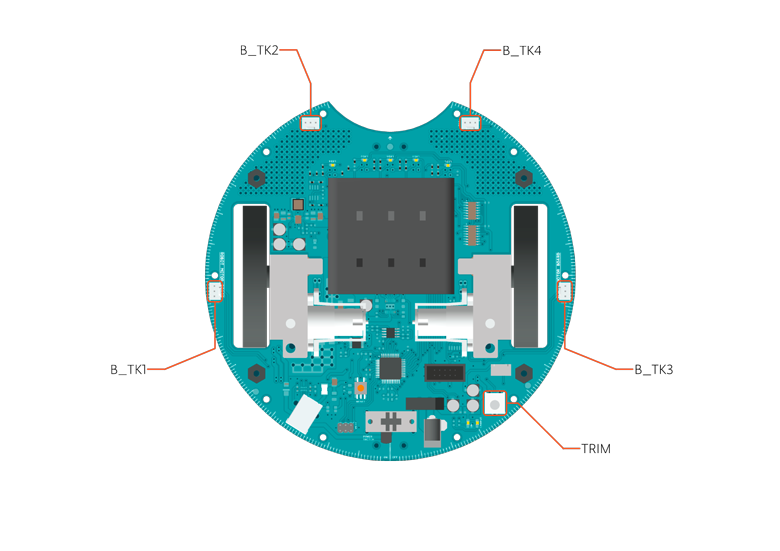
Input and Output

The Robot comes with a series of pre-soldered connectors. There are a number of additional spots for you to install additional parts if needed. All the connectors are labelled on the boards and mapped to named ports through the [Robot library](https://www.arduino.cc/en/Reference/RobotLibrary) allowing access to standard Arduino functions. Each pin can provide or receive a maximum of 40mA at 5V. Some pins have specialized functions :

* Control Board TK0 to TK7: these pins are multiplexed to a single analog pin on theControl Board's microprocessor. They can be used as analog inputs for sensors like distance sensors, analog ultrasound sensors, or mechanical switches to detect collisions.
* Control Board TKD0 to TKD5: these are digital I/O pins directly connected to the processor, addressed using [Robot.digitalRead()](https://www.arduino.cc/en/Reference/RobotDigitalRead) and [Robot.digitalWrite)](https://www.arduino.cc/en/Reference/RobotDigitalWrite) functions. Pins TKD0 to TKD3can also be used as analog inputs with [Robot.analogRead()](https://www.arduino.cc/en/Reference/RobotAnalogRead)
* Note: if you have one of the first generation robots, you will see that the TKD\* pins are named TDK\* on the Robot's silkscreen. TKD\* is the proper name for them and is how we address them on the software.



* Motor Board TK1 to TK4: these pins are named in software as B\_TK1 to B\_TK4, they can be digital or analog input pins, and support [Robot.digitalRead()](https://www.arduino.cc/en/Reference/RobotDigitalRead), [Robot.digitalWrite)](https://www.arduino.cc/en/Reference/RobotDigitalWrite) and [Robot.analogRead()](https://www.arduino.cc/en/Reference/RobotAnalogRead).



* Serial Communication: The boards communicate with each other using the processors' serial port. A 10-pin connector connects both boards carries the serial communication, as well as power and additional information like the battery's current charge.
* Control Board SPI: SPI is used to control the GTFT and SD card. If you want to flash the processor using an external programmer, you need to disconnect the screen first.
* Control Board LEDs: the Control Board has three on-board LEDs. One indicates the board is powered (PWR). The other two indicate communication over the USB port (LED1/RX and TX). LED1 is also accessible via software.
* Both boards have I2C connectors available: 3 on the Control Board and 1 on the Motor Board.

Control Board Pin Mapping

| ARDUINO LEONARDO | ARDUINO ROBOT CONTROL | ATMEGA 32U4 | FUNCTION | REGISTER |
| --- | --- | --- | --- | --- |
| D0 | RX | PD2 | RX | RXD1/INT2 |
| D1 | TX | PD3 | TX | TXD1/INT3 |
| D2 | SDA | PD1 | SDA | SDA/INT1 |
| D3# | SCL | PD0 | PWM8/SCL | OC0B/SCL/INT0 |
| D4 | MUX\_IN A6 | PD4 |  | ADC8 |
| D5# | BUZZ | PC6 | ??? | OC3A/#OC4A |
| D6# | MUXA/TKD4A7 | PD7 | FastPWM | #OC4D/ADC10 |
| D7 | RST\_LCD | PE6 |  | INT6/AIN0 |
| D8 | CARD\_CS A8 | PB4 |  | ADC11/PCINT4 |
| D9# | LCD\_CS A9 | PB5 | PWM16 | OC1A/#OC4B/ADC12/PCINT5 |
| D10# | DC\_LCD A10 | PB6 | PWM16 | OC1B/0c4B/ADC13/PCINT6 |
| D11# | MUXB | PB7 | PWM8/16 | 0C0A/OC1C/#RTS/PCINT7 |
| D12 | MUXC/TKD5A11 | PD6 |  | T1/#OC4D/ADC9 |
| D13# | MUXD | PC7 | PWM10 | CLK0/OC4A |
| A0 | KEY D18 | PF7 |  | ADC7 |
| A1 | TKD0 D19 | PF6 |  | ADC6 |
| A2 | TKD1 D20 | PF5 |  | ADC5 |
| A3 | TKD2 D21 | PF4 |  | ADC4 |
| A4 | TKD3 D22 | PF1 |  | ADC1 |
| A5 | POT D23 | PF0 |  | ADC0 |
| MISO | MISO D14 | PB3 |  | MISO,PCINT3 |
| SCK | SCK D15 | PB1 |  | SCK,PCINT1 |
| MOSI | MOSI D16 | PB2 |  | MOSI,PCINT2 |
| SS | RX\_LED D17 | PB0 |  | RXLED,SS/PCINT0 |
| TXLED | TX\_LED | PD5 |  |  |
| HWB |  | PE2 |  | HWB |

Motor Board Pin Mapping

| ARDUINO LEONARDO | ARDUINO ROBOT CONTROL | ATMEGA 32U4 | FUNCTION | REGISTER |
| --- | --- | --- | --- | --- |
| D0 | RX | PD2 | RX | RXD1/INT2 |
| D1 | TX | PD3 | TX | TXD1/INT3 |
| D2 | SDA | PD1 | SDA | SDA/INT1 |
| D3# | SCL | PD0 | PWM8/SCL | OC0B/SCL/INT0 |
| D4 | TK3 A6 | PD4 |  | ADC8 |
| D5# | INA2 | PC6 | ??? | OC3A/#OC4A |
| D6# | INA1 A7 | PD7 | FastPWM | #OC4D/ADC10 |
| D7 | MUXA | PE6 |  | INT6/AIN0 |
| D8 | MUXB A8 | PB4 |  | ADC11/PCINT4 |
| D9# | INB2 A9 | PB5 | PWM16 | OC1A/#OC4B/ADC12/PCINT5 |
| D10# | INB1 A10 | PB6 | PWM16 | OC1B/0c4B/ADC13/PCINT6 |
| D11# | MUXC | PB7 | PWM8/16 | 0C0A/OC1C/#RTS/PCINT7 |
| D12 | TK4 A11 | PD6 |  | T1/#OC4D/ADC9 |
| D13# | MUXI | PC7 | PWM10 | CLK0/OC4A |
| A0 | TK1 D18 | PF7 |  | ADC7 |
| A1 | TK2 D19 | PF6 |  | ADC6 |
| A2 | MUX\_IN D20 | PF5 |  | ADC5 |
| A3 | TRIM D21 | PF4 |  | ADC4 |
| A4 | SENSE\_A D22 | PF1 |  | ADC1 |
| A5 | SENSE\_B D23 | PF0 |  | ADC0 |
| MISO | MISO D14 | PB3 |  | MISO,PCINT3 |
| SCK | SCK D15 | PB1 |  | SCK,PCINT1 |
| MOSI | MOSI D16 | PB2 |  | MOSI,PCINT2 |
| SS | RX\_LED D17 | PB0 |  | RXLED,SS/PCINT0 |
| TXLED | TX\_LED | PD5 |  |  |
| HWB |  | PE2 |  | HWB |

Communication

The Robot has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega32U4 provides UART TTL (5V) serial communication, which is available on digital the 10-pin board-to-board connector. The 32U4 also allows for serial (CDC) communication over USB and appears as a virtual com port to software on the computer. The chip also acts as a full speed USB 2.0 device, using standard USB COM drivers. [On Windows, a .inf file is required](http://arduino.cc/en/Guide/Windows#toc4). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Robot board. The RX (LED1) and TX LEDs on the board will flash when data is being transmitted via the USB connection to the computer (but not for serial communication between boards). Each one of the boards has a separate USB product identifier and will show up as different ports on you IDE. Make sure you choose the right one when programming. The ATmega32U4 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the [documentation](https://www.arduino.cc/en/Reference/Wire) for details. For SPI communication, use the [SPI library](https://www.arduino.cc/en/Reference/SPI).

Programming

The Robot can be programmed with the Arduino software ([download](https://www.arduino.cc/en/Main/Software)). Select "Arduino Robot Control Board" or "Arduino Robot Motor Board" from the Tools > Board menu. For details, see the [getting started page](https://www.arduino.cc/en/Guide/Robot) and [tutorials](https://www.arduino.cc/en/Tutorial/HomePage).

The ATmega32U4 processors on the Arduino Robot come preburned with a [bootloader](https://www.arduino.cc/en/Tutorial/Bootloader) that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the AVR109 protocol. You can bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see [these instructions](https://www.arduino.cc/en/Hacking/Programmer) for details.

Automatic (Software) Reset and Bootloader initiation

Rather than requiring a physical press of the reset button before an upload, the Robot is designed in a way that allows it to be reset by software running on a connected computer. The reset is triggered when the Robot's virtual (CDC) serial / COM port is opened at 1200 baud and then closed. When this happens, the processor will reset, breaking the USB connection to the computer (meaning that the virtual serial / COM port will disappear). After the processor resets, the bootloader starts, remaining active for about 8 seconds. The bootloader can also be initiated by double-pressing the reset button on the Robot. Note that when the board first powers up, it will jump straight to the user sketch, if present, rather than initiating the bootloader.

Because of the way the Robot handles reset it's best to let the Arduino software try to initiate the reset before uploading, especially if you are in the habit of pressing the reset button before uploading on other boards. If the software can't reset the board you can always start the bootloader by double-pressing the reset button on the board. A single press on the reset will restart the user sketch, a double press will initiate the bootloader.

USB Overcurrent Protection

Both of the Robot boards have a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The Robot is 19cm in diameter. Including wheels, GTFT screen and other connectors it can be up to 10cm tall.

To Learn More

To calibrate the compass module, if you're using an old model of the robot (which uses Honeywell HMC 6352), refer to this tutorial: [Calibrate Compass](https://www.arduino.cc/en/Tutorial/RobotCompassCalibration) See also: [getting started with the Arduino Robot](https://www.arduino.cc/en/Guide/Robot) and [the Robot's library pages](https://www.arduino.cc/en/Reference/RobotLibrary).

Getting Started with the Arduino Robot

<https://www.arduino.cc/en/Guide/Robot>

Arduino Examples from Libraries (Robot)

<https://www.arduino.cc/en/Tutorial/LibraryExamples>

### Logo

This sketch shows you basic movement with the Robot. When the sketch starts, press the buttons ont he control board to indicate the direction you want the robot to move. After you've keyed in your sequence (up to 20 steps at a time), press the middle button to record the steps to memory. Put your robot on the floor for it to follow the directions you programmed.

<https://www.arduino.cc/en/Tutorial/RobotLogo>

### Line Following

Take a large piece of paper (or tape smaller pieces together to make a big one) and draw a shape on it using a thick black marker. This shape will be your racing track.

Put the robot on top of the line and turn it on. It will scan the floor to detect the line. When it finds the line, it will start following it the path.

Compete with other robots to see which one makes it faster!

<https://www.arduino.cc/en/Tutorial/RobotLineFollowing>

### Disco bot

This sketch demonstrates how to play a melody with the robot, by playing back some sound files.

You can choose between three melodies specially composed for the Arduino Robot using the buttons on the control board. Once you get the music you like, put the robot on the floor and it will dance for you.

<https://www.arduino.cc/en/Tutorial/RobotDiscoBot>

### Compass

The robot has a compass module, which it uses to find its direction. This sketch will make sure the robot goes towards a certain direction.

When you hold the robot in your hands and rotate, you will the screen change, indicating direction.

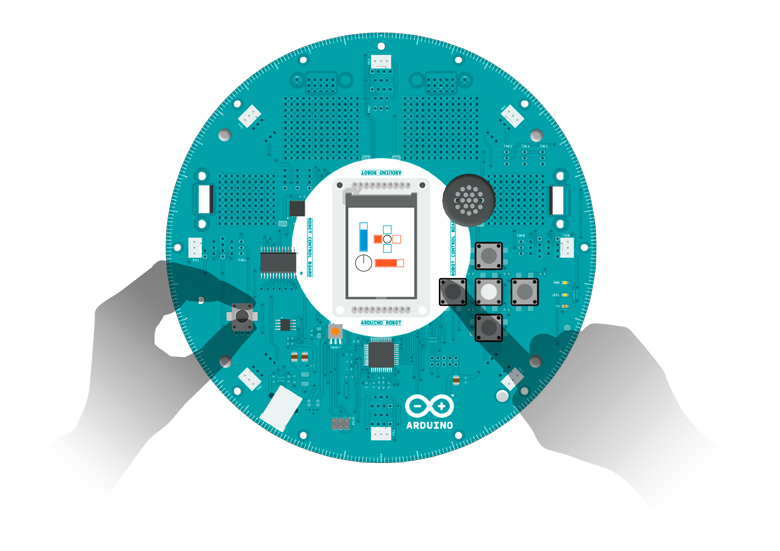
NB : magnets will interfere with the compass. If you're getting unexpected results, check to make sure there are none around.

<https://www.arduino.cc/en/Tutorial/RobotCompass>

### Inputs

This sketch shows you how to use the control board potentiometer and buttons as a keyboard. It turns the robot into a mobile music machine, have some fun with the music by pressing different buttons.

<https://www.arduino.cc/en/Tutorial/RobotInputs>

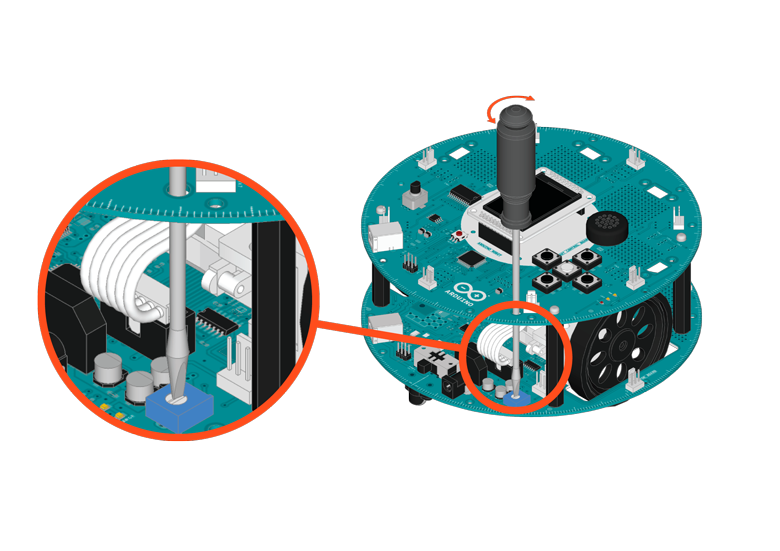


### Wheel Calibration

Use this sketch to calibrate the wheels of your robot. Your robot should move in a straight line when putting both motors at the same speed.

Run the software and follow the on-screen instructions. Use the potentiometer on the bottom board to adjust the calibration.

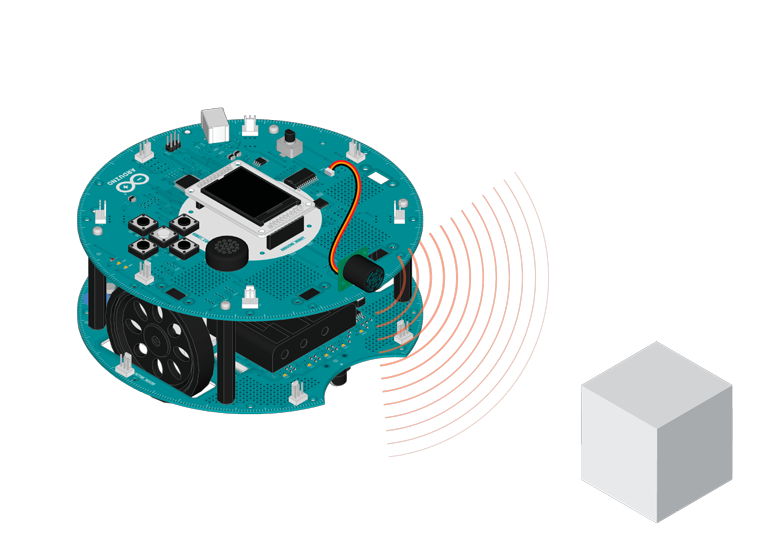
<https://www.arduino.cc/en/Tutorial/RobotWheelCalibration>



### Runaway Robot

Is your robot bumping into walls? This quickly solves that problem. By attaching a ultrasonic rangefinder, the robot can find out if it's too close to obstacles, and turn around to avoid collision.

<https://www.arduino.cc/en/Tutorial/RobotRunawayRobot>



### Remote Control

*This example is considered experimental, you will need to install the IR-Remote library by Ken Shirriff on your IDE for it to run, read the code for more information*

If you connect a IR remote receiver to the robot, you can control it like controlling your TV set. Take a Sony compatible remote controller, map some buttons to different actions, and you can make the robot move around without touching it!

<https://www.arduino.cc/en/Tutorial/RobotRemoteControl>

### Picture Browser

You can make your own gallery/picture show with help of the Robot. Put some pictures in the SD card, start the sketch, and you can see them showing up on the LCD. Use left and right button on the D-pad to go the previous or next image. Press up or down to enter the "tilt navigation" mode.

<https://www.arduino.cc/en/Tutorial/RobotPictureBrowser>

### Rescue

The Rescue challenge in robotics consists of getting your robot to follow a line until it reaches a location where it will perform a task. In this case, the robot makes it to an obstacle, pushes it out of the way, and continues along.

<https://www.arduino.cc/en/Tutorial/RobotRescue>

### Hardware Required

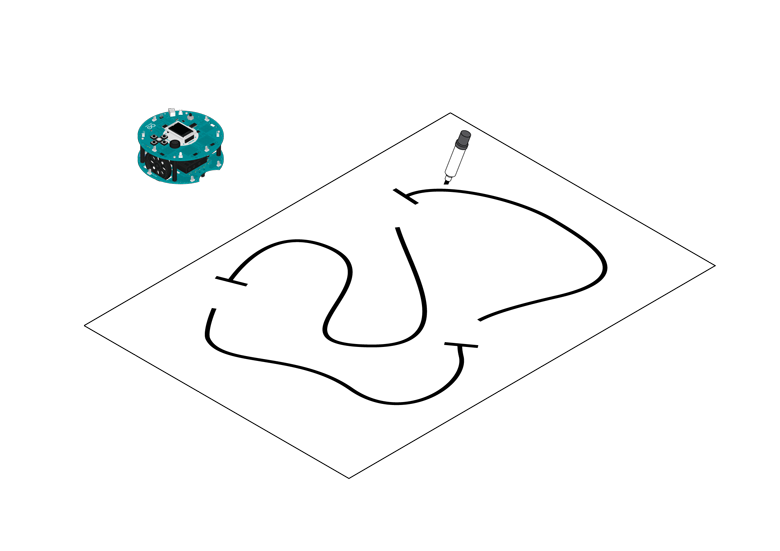
* Arduino Robot
* Large paper
* Thick black marker
* Obstacles, like empty cans, or not too-heavy toys
* "victim" objects

### Instruction

1. To prepare the track, follow the instructions in [line following example](https://www.arduino.cc/en/Tutorial/RobotLineFollowing). There are a few differences:
   1. Add some end lines for the robot to stop. Create a gap in the line, and draw a rectangle about the size of the robot in the gap.
   2. Put the "victim" object inside the rectangle.
2. Upload the example, unplug USB and turn on power.
3. Put robot on the track, on the line.
4. The robot will start following the line, and push the victim out of its location
5. If the robot does not follow the line well, see [lineFollowConfig](https://www.arduino.cc/en/Reference/RobotLineFollowConfig)() for details on calibration.

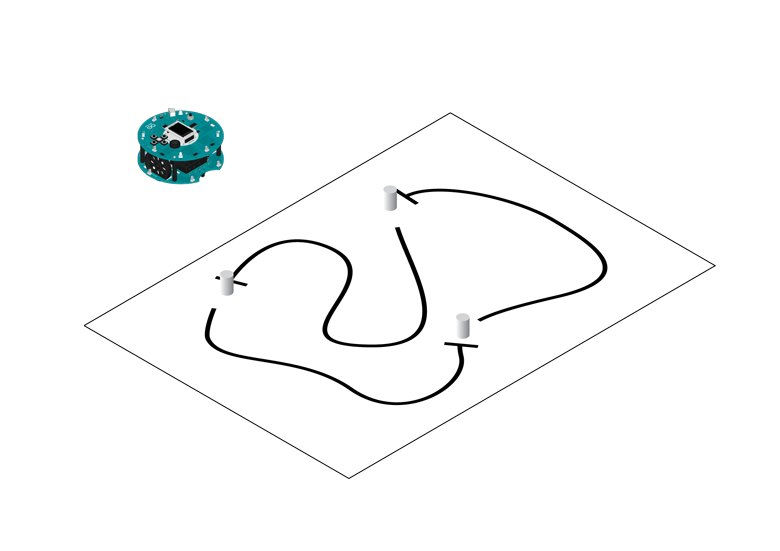
### Try it out

First, draw your racing track, adding the stops (perpendicular lines) for the robot to know where the obstacles will be.



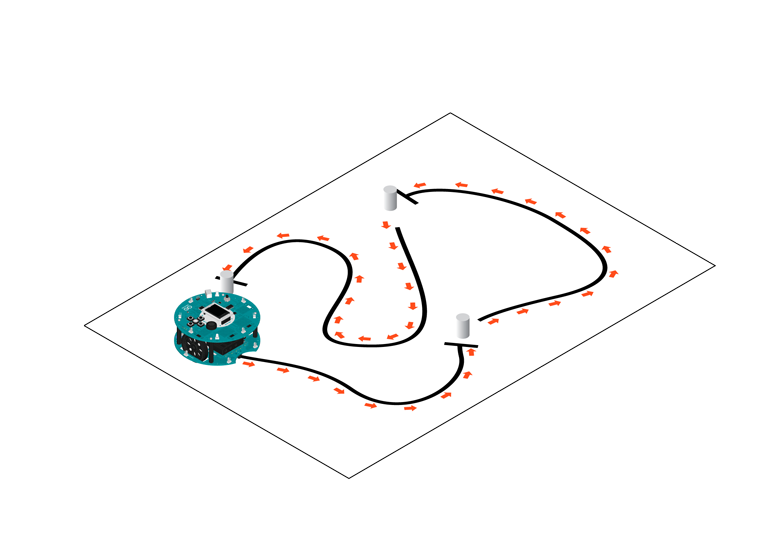
Use a marker to design your racing track

Place your obstacles on the track. The robot can push some weight, like empty soda cans or small plastic toys make for good obstacles.



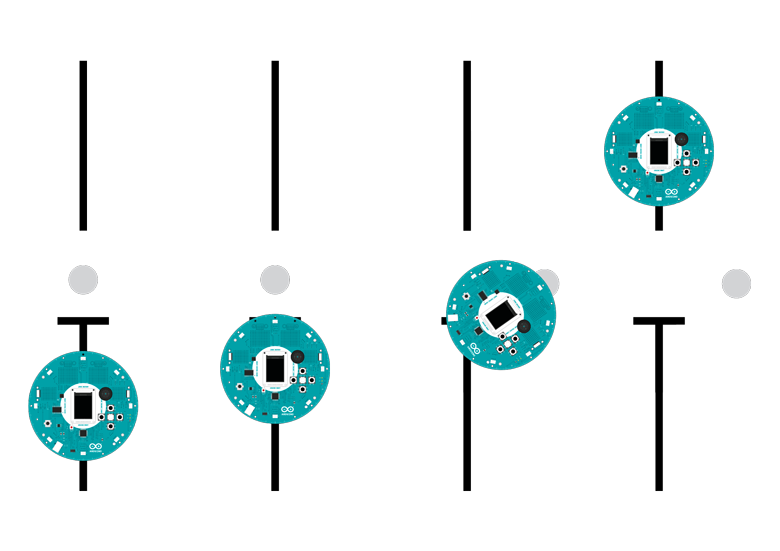
Place the obstacles

Put the robot on the track and turn it on to see if it can solve the challenge.



Put the robot to run

In front of an obstacle, the robot slows down, starts pushing the object, backs up and starts racing again.

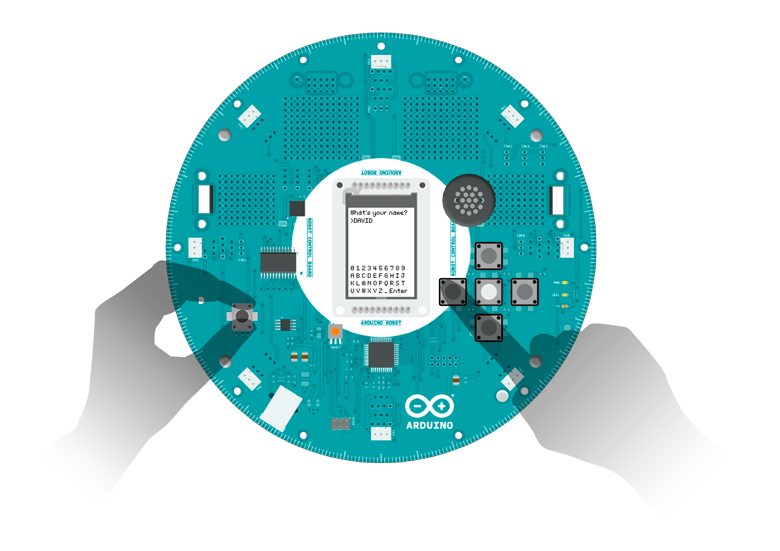


Robot pushing the obstacle away

### Hello User

This sketch is the first thing you see when starting the robot. It gives you a warm welcome, shows you some of its really amazing features, and lets you personalize some data.

<https://www.arduino.cc/en/Tutorial/RobotHelloUser>



Calibrate Compass (Tutorial)

<https://www.arduino.cc/en/Tutorial/RobotCompassCalibration>

Robot Library Reference

<https://www.arduino.cc/en/Reference/RobotLibrary>

#### RobotControl class

This class commands the Control Board as well as all the I/Os and motors on the Motor Board when it has the stock firmware installed.

* [RobotControl](https://www.arduino.cc/en/Reference/RobotConstructor) Constructor
* [begin](https://www.arduino.cc/en/Reference/RobotBegin)()
* [setMode](https://www.arduino.cc/en/Reference/RobotSetMode)()
* [pauseMode](https://www.arduino.cc/en/Reference/RobotPauseMode)()
* [isActionDone](https://www.arduino.cc/en/Reference/RobotIsActionDone)()
* [lineFollowConfig](https://www.arduino.cc/en/Reference/RobotLineFollowConfig)()
* [digitalRead()](https://www.arduino.cc/en/Reference/RobotDigitalRead)
* [digitalWrite()](https://www.arduino.cc/en/Reference/RobotDigitalWrite)
* [analogRead()](https://www.arduino.cc/en/Reference/RobotAnalogRead)
* [analogWrite()](https://www.arduino.cc/en/Reference/RobotAnalogWrite)
* [updateIR()](https://www.arduino.cc/en/Reference/RobotUpdateIR)
* [knobRead](https://www.arduino.cc/en/Reference/RobotKnobRead)()
* [compassRead](https://www.arduino.cc/en/Reference/RobotCompassRead)()
* [keyboardRead](https://www.arduino.cc/en/Reference/RobotKeyboardRead)()
* [waitContinue](https://www.arduino.cc/en/Reference/RobotWaitContinue)()
* [motorsWrite](https://www.arduino.cc/en/Reference/RobotMotorsWrite)()
* [motorsStop](https://www.arduino.cc/en/Reference/RobotMotorsStop)()
* [turn](https://www.arduino.cc/en/Reference/RobotTurn)()
* [pointTo](https://www.arduino.cc/en/Reference/RobotPointTo)()
* [beginSpeaker](https://www.arduino.cc/en/Reference/RobotBeginSpeaker)()
* [playMelody](https://www.arduino.cc/en/Reference/RobotPlayMelody)()
* [beep](https://www.arduino.cc/en/Reference/RobotBeep)()
* [playFile](https://www.arduino.cc/en/Reference/RobotPlayFile)()
* [tuneWrite](https://www.arduino.cc/en/Reference/RobotTuneWrite)()
* [tempoWrite](https://www.arduino.cc/en/Reference/RobotTempoWrite)()
* [beginTFT](https://www.arduino.cc/en/Reference/RobotBeginLCD)()
* [text](https://www.arduino.cc/en/Reference/RobotText)()
* [drawBMP](https://www.arduino.cc/en/Reference/RobotDrawBMP)()
* [debugPrint](https://www.arduino.cc/en/Reference/RobotDebugPrint)()
* [clearScreen](https://www.arduino.cc/en/Reference/RobotClearScreen)()
* [displayLogos](https://www.arduino.cc/en/Reference/RobotDisplayLogos)()
* [drawCompass](https://www.arduino.cc/en/Reference/RobotDrawCompass)()
* [beginSD](https://www.arduino.cc/en/Reference/RobotBeginSD)()
* [userNameRead](https://www.arduino.cc/en/Reference/RobotUserNameRead)()
* [userNameWrite](https://www.arduino.cc/en/Reference/RobotUserNameWrite)()
* [robotNameRead](https://www.arduino.cc/en/Reference/RobotRobotNameRead)()
* [robotNameWrite](https://www.arduino.cc/en/Reference/RobotRobotNameWrite)()
* [cityNameRead](https://www.arduino.cc/en/Reference/RobotCityNameRead)()
* [cityNameWrite](https://www.arduino.cc/en/Reference/RobotCityNameWrite)()
* [countryNameRead](https://www.arduino.cc/en/Reference/RobotCountryNameRead)()
* [countryNameWrite](https://www.arduino.cc/en/Reference/RobotCountryNameWrite)()

#### RobotMotor class

Use this to make your own firmware for the Motor Board.

* [RobotMotor](https://www.arduino.cc/en/Reference/RobotMotorConstructor) Constructor
* [begin](https://www.arduino.cc/en/Reference/RobotMotorBegin)()
* [process](https://www.arduino.cc/en/Reference/RobotMotorProcess)()
* [parseCommand](https://www.arduino.cc/en/Reference/RobotMotorParseCommand)()
* [motorsWrite](https://www.arduino.cc/en/Reference/RobotMotorMotorsWrite)()
* [IRread](https://www.arduino.cc/en/Reference/RobotMotorIRread)()

Motor Control board (Write your own firmware Information)

<https://www.arduino.cc/en/Reference/RobotMotorWriteYourOwnFirmware>

Arduino. ORG Robot Tutorial Page (More or less the same as .CC)

<http://www.arduino.org/learning/tutorials/boards-tutorials/content/arduino-robot-tutorial>

Arduino.org Robot Library Information Page: (more or less the same as .cc, but seems organized a bit better)

<http://www.arduino.org/learning/reference/robot>

Customizable Line Following Printable maps:

<http://robotsquare.com/2012/11/28/line-following/>