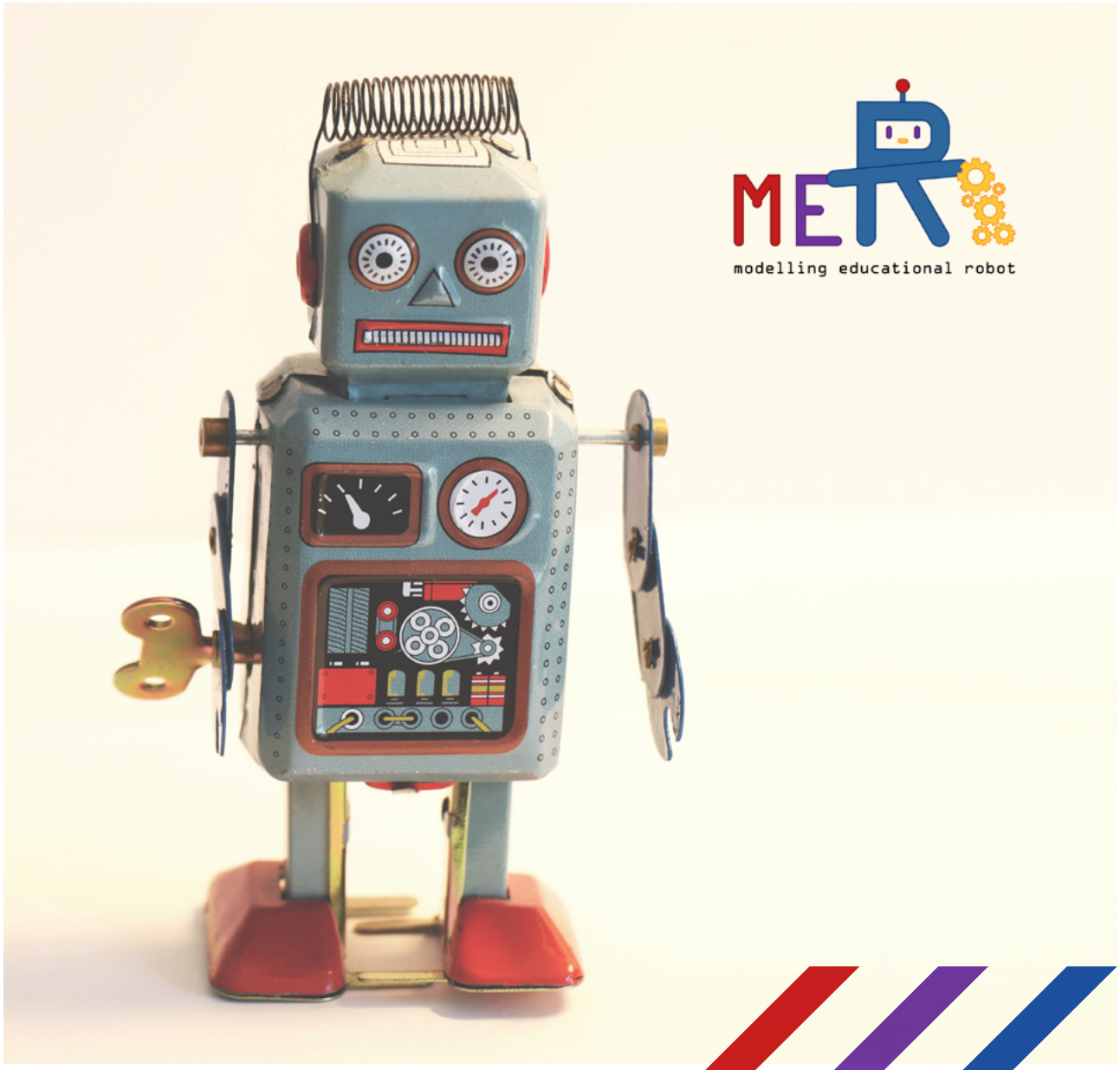


modelling educational robot

Modeling Educational Robot

COURSE BOOK

WWW.MERPROJECT.NET

Co-funded by the Erasmus+ Programme of the European Union

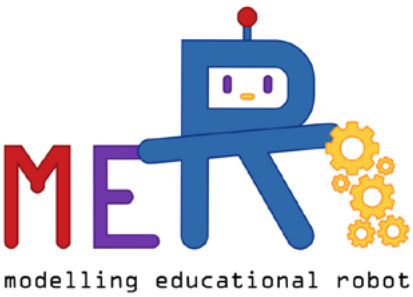
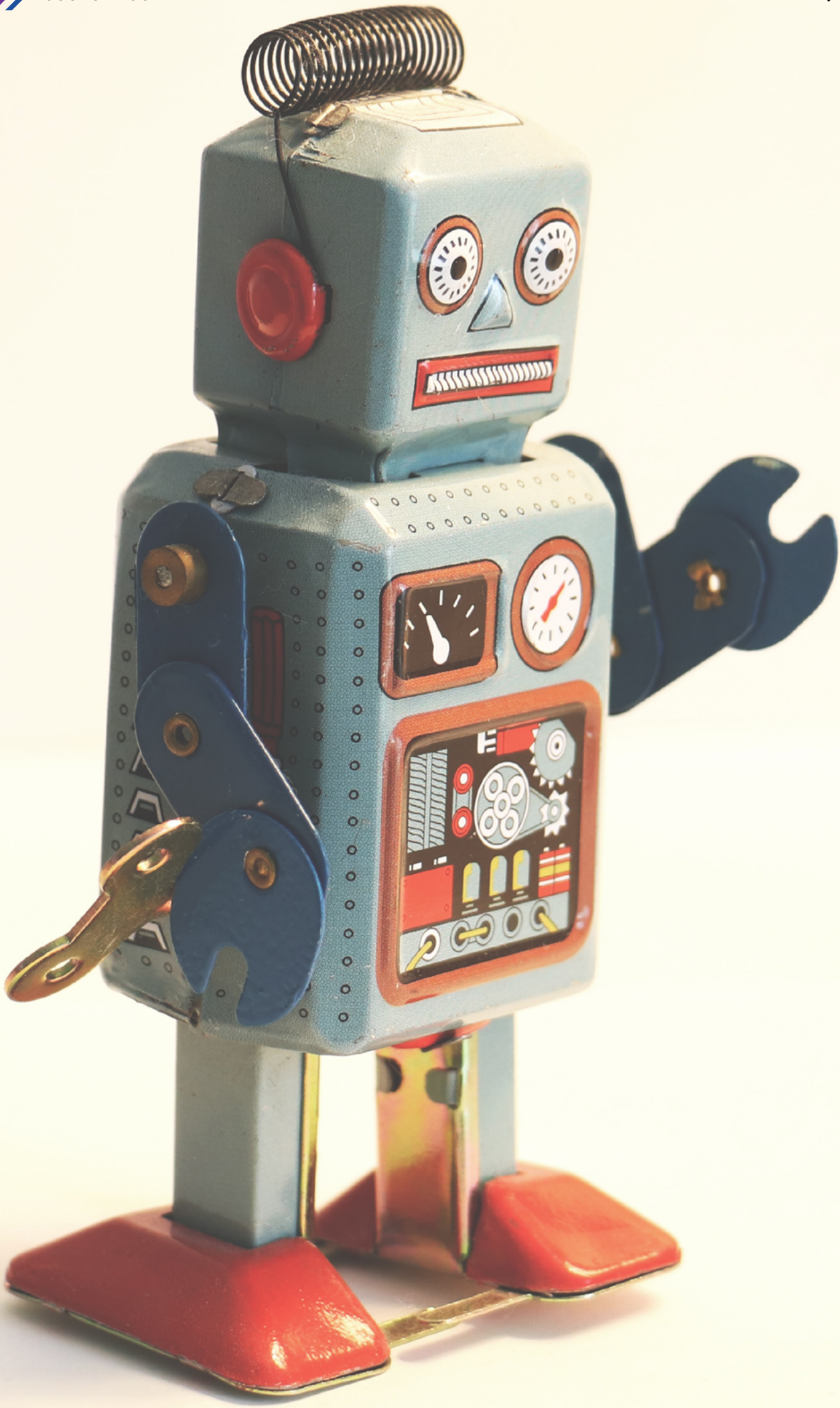


Image credits: Unsplash, Freepik

December 2021



Contents

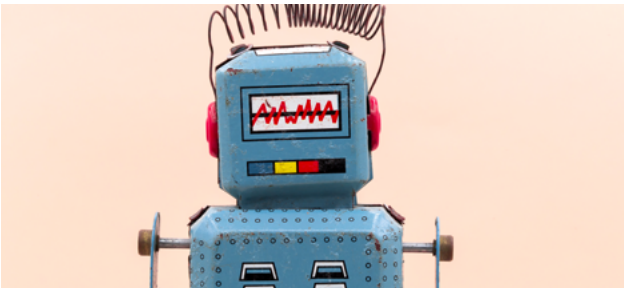
Introduction

The main question is not so much how the new technology can help students learn. Rather, it´s what will they do with what they learn? Will they use their knowledge to build …. or will they use it to destroy? Only real human beings can help them know the difference – regardless of the medium or the technology used in communication. - Fred Rogers, 1996

This book is a result of an international project with participants from Italy, Slovenia, Spain, Turkey and, Bosnia and Herzegovina realized during one and a half-year of cooperative works. A large portion of the work has been done during the COVID-19 pandemic, a period in which the world scientific community showed its ability to solve the complex problem of identifying the virus and developing a vaccine. During this pandemic, the role of information technology, automation, and robotics in the world economy had been emphasized. The information technology allowed, at least in some countries, an educated control of the lockdowns and other measures and at the same time work from the distance. The factories full of automated machines and robots sustained, to a large extent, world industrial production. Thanks to mechanization and automation the global food production did not suffer a large setback and international movement of goods did not suffer large interruptions.

The pandemic demonstrated how much our modern society is dependent on technological developments and of the ability to innovate and use technology wisely and effectively. That brings us to the core subject this book is discussing – building a capacity in our schools to deal with the increasing complexities of educational needs for twenty-first-century learners. It is not only logical as shown by the current developments, but the educational institutions must promote STEAM education beyond what we need as the users or consumers, beyond technology literacy. The need of generating new technologies and educate new technology inventors and producers should be a new emphasis in our education to pass from simple users towards technology inventors and reaching wider technology fluency. It is the question ".. how to give the best possible education for a future powered by technology?" that needs to be addressed by all and better sooner than later. Will our current students be educated to influence the future of technology or simply be the technology consumers will they be technology fluent or technology literate is in hands of education that is offered to learners in the school but also at home. Technology fluency is asking for understanding limits and possibilities embedded in machines and computers and learners becoming innovators capable of technology innovation rather than just its usage.

While working on this project in different countries and with students of different ages and experiences it becomes apparent that infusing robotics into education helps educate students that will grow beyond the usage of the technology and will be able to contribute to technology development - they could go from technology literacy to technology fluency. Robotics in the classroom facilitates a blending of the different subjects and maybe, more importantly, it opens the door for learners' exploration (let us not use the word research even though in reality it is research). That way robotics in the classroom could facilitate learning by doing, supports critical thinking, problem-solving and, increased comprehension of complex concepts and procedures. That enables the usage of innovative learning methods so that students can develop creative thinking, digital competence and go beyond usage to the level of creation of technology. These learning methods have to be able to prepare the learners for work with technology and use the opportunities created by technology.



For technology to succeed in an educational environment it is not enough that it helps teachers – it must attract learners and motivate them to explore and learn. Accumulated experience shows that robots and robotics-related subjects can play the role of the "curiosity drivers" for many learners. What is a robot and why it has such an attraction for the learners? The definition of a robot is still an evasive subject with many researchers having, and using, their own. On a basic level, we see robots as a combination of different components (sensors- controllers-motors) that together act as one system. P.W. Singer (2009) defined robots as: " … machines that are built upon what researchers call the "sense-think-act" paradigm. That is they are man-made devices with three key components: "sensors" that monitor the environment and detect changes in it, "processors" or artificial intelligence" that decides how to respond, and "effectors" that act upon the environment in a manner that reflects decisions, creating some sort of the change in the world around a robot. When these three parts act together, a robot gains the functionality of an artificial organism."

This definition seems to blend in all key ingredients – the attractiveness of a robot comes from its ability to act upon the environment and learners can witness the results of the action. That brings in the cycle of innovations – add a new sensor and you could add new functions to your robot, change the ways you change actions based on the sensor's information and the whole new world of actions appears. Creating new robot behavior becomes a challenge, a cognitive exercise with a large space of possibilities for different solutions and, why not, competition among learners-students. Reflection of ever-changing technology by allowing students to learn while playing with robots becomes a part of the learning process. The robot within its sensing - decision making - actuation paradigm allows many different emphases in using robots in education. On one side it is a skill of building a mechanical system that is supposed to deliver desired capabilities – a process of transformation of an idea to a 3D artifact. For more advanced students the process of building a robot may be an opportunity to learn the operation of some machine tools and further interact with technology.

The process of making a sense of the data that are coming from sensors brings another level of the learning experience – not only with the technology upon which sensors are built but, maybe more importantly, on a notion of the information content of measurement and the abstraction of the data obtained from sensors and their usage in the decision making – the formulation of the control algorithms that links sensor data with desired robot behavior and the actuators. The process of measurement data usage brings into learners' attention the link data between the real and the virtual world – a perception of the environment expressed in a set of data that could be integrated into decision making. Perception - the ability to collect and interpret information from sensors - cameras, sonar rangefinders, radar, light sensors, and many others - seems magical because a product we use becomes something that interacts with us in many different ways. Robots make sense of their surroundings make a decision and act back on the world and make a change.

For students, robots are a new form of link between our real world and the digital universe. For that, we need to add a cognitive-control ability to a robot based on the robot being connected to the real world by sensing – a perception of environment – and the ability to act upon the environment – actuation. Cognitive ability is about deciding on what to do. It is a glue connecting perception and action – just like our brains. Robot brains (computers) have one more ability – to be interconnected to the vast digital world (digital cloud) with an abundance of information within. And that differs robots from natural animals. With the Internet and radio robots have effortless access to the almost unlimited source of information – data and computing power – so that their computational power may be rather limited. And that brings through a robot into a learner´s reach a waste amount of information and possibility of exploring both real and virtual world complexities and how to think about a pattern of information – how to relate available information into a meaningful whole and see patterns instead of dots.

This book offers a set of learning materials that can guide interested learners from basic learning about robots to a level of abstraction and integration of robots in an unstructured world.

You can find more material, course lessons, and source code examples at **https://www.merproject.net**.

We wish you happy learning!

MER Project Partners

Basic Level Robot

What is Arduino?

The need for electricity and electrical devices today is immense. Each electrical device nowadays requires some kind of programming in order to work properly. As we know, programming is not quite interesting for everyone, and it is not so easy.

In order to make programming easier and more available to daily consumers, the engineers have made the “Arduino”. The main idea behind “Arduino” is to bring closer electrical devices and programming to non – engineers, because it doesn’t require big knowledge in programming.

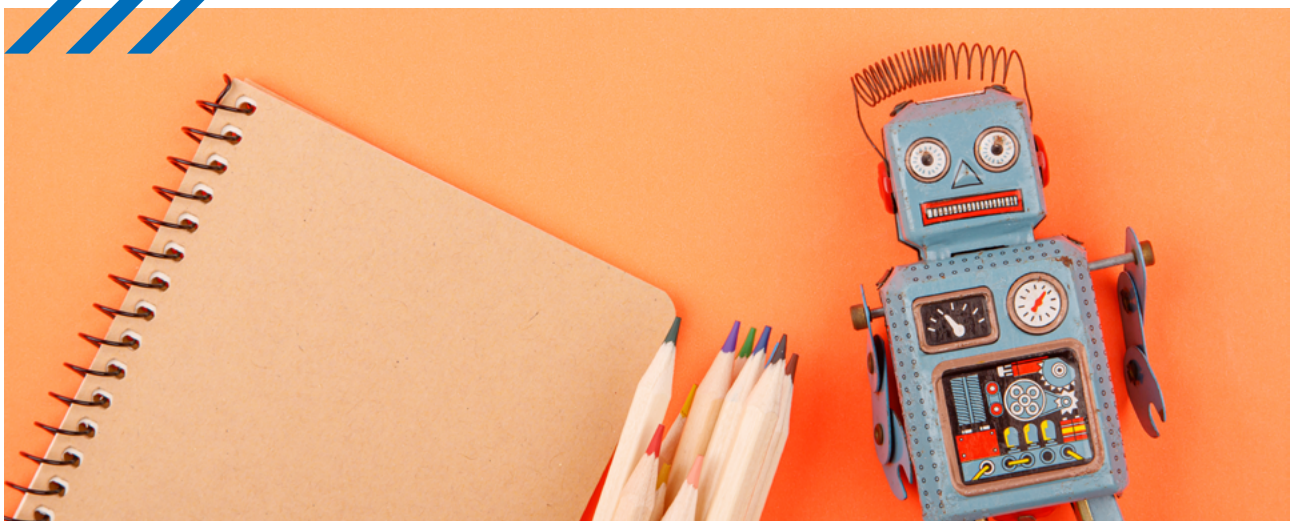
History

The very beginning of the “Arduino” is connected to the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy. Microcontrollers that were used in that time were BASIC Stamp microcontrollers and they were quite expensive for the students to afford it.

Herardo Barragan was the initiator of the development project called Wiring, which actually was his master thesis. The project initiate in 2003, and the main goal was to create simple, low-cost tools for creating digital projects by non – engineers. The platform was based on a printed circuit board (PCB) with an ATmega168 microcontroller and library functions to easily program the microcontroller.

The year 2003, was actually a breakthrough year for the science and usage of Arduino, and in that year we meet with the name of “Arduino”. The main people are Masimo Banzi, David Cuartielles, Tom Igoe, and some other students and professors from the IDII. The most interesting thing is that the student who had initiated this project to the professors weren’t involved in the development of the project.

This open-source software/hardware has continued with growing and by the year 2013, they had almost 700 000 commercially produced units. Today, “Arduino” engaged in a partnership with ARM Holdings which will provide a safer and more stable future for “Arduino”.



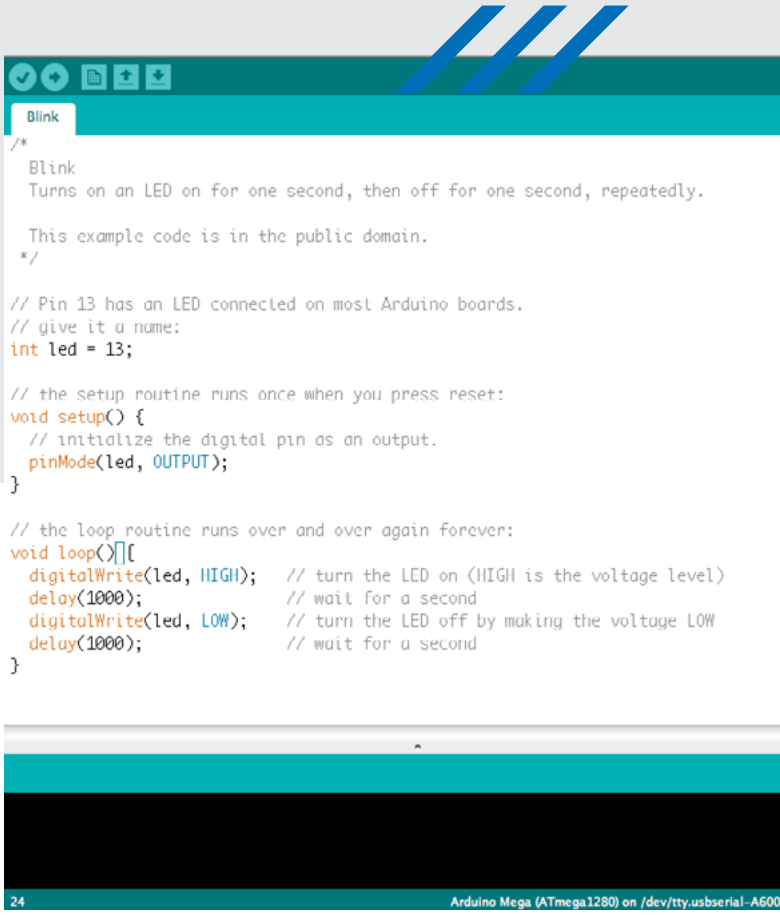
Parts and Characteristics

Arduino is an open-source platform for building electronics projects. It consists of both hardware (programmable circuit board, known as a microcontroller) and software (integrated Development Environment – IDE). The software is simply running on our computer and it is used for writing and uploading the code on the circuit board.

The Arduino software is based on the simpler version of the C++ program language, which makes him easy to learn. The other thing that makes "Arduino" so simple is the connection between the hardware and software parts. We can simply use a USB cable and connect the circuit board with our PC and in that way, the software has a full connection with the hardware and is ready to operate.

The Arduino software can be simply and freely downloaded from the main web page https://www.arduino.cc/. After the download, it can be installed in a few minutes and it is ready to be used. The interface is very simple, so people who are not so good at programming may not have bigger problems. The software itself has a library in which we can find some programming codes for specific problems.

In the figure given below, we can see the simple code of 10 lines. The written code will give us blinking of the LED diode

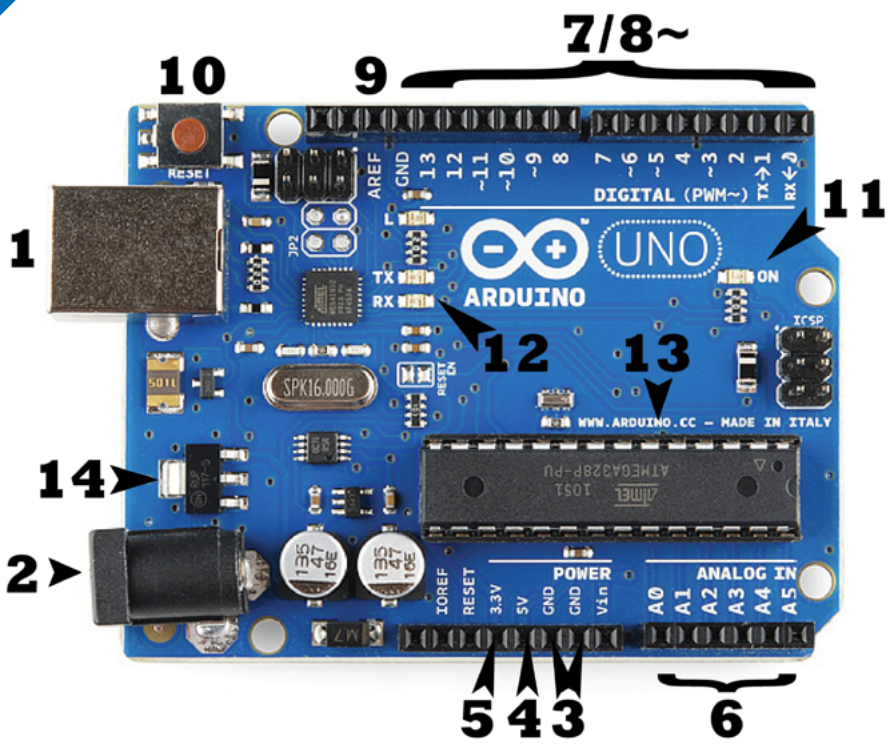


on the circuit board as an output, which actually shows how simple it is. One of the most important things in programming is that we have to choose the proper hardware since the software enables us to pick the board at which we want to have an output.

The hardware consists of several parts, but the main parts are the same for all types of Arduinos, other parts depend basically on the type of the hardware. Here, we will describe parts (components) of the Arduino Uno, since it is the most commonly used Arduino around the world.

The main parts of the Arduino

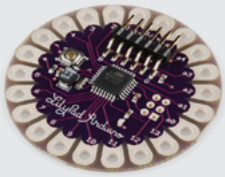
1. Each Arduino must be powered by electricity and it can be done by using the USB port, which will connect the Arduino with our PC. Also, the USB port is used to transfer the programming code from software to the hardware.
2. The other way of powering Arduino can be by using the barrel jack, which is connecting a wall power supply to Arduino.
3. GND (ground) pins are used to ground our circuit, there are several pins on the board.
4. 5V pin supplies 5 volts of power.
5. 3.3V pin supplies 3 volts of power. The simple components can run without problem on any of these two pins.
6. Analog In pins are used to read the signal from an analog sensor, e.g. temperature sensor, and then convert it into a digital value that can be read by the consumer. There are several Analog In pins labeled as A0 to A5.
7. Digital pins are used for both, input and output. The input might be, telling if the button is pushed, and the output might be the powering of the LED diode. Three pins are placed across the Analog In pins.
8. PWM (Pulse with Modulation) pins can act as a digital one, but the main difference (characteristic) is that they can simulate an analog output.
9. AREF (Analog Reference) pins can be left alone. In some cases, it might be used to set an external reference voltage.
10. Reset Button enables us to temporarily connect the reset pin to the ground and restart any code that is loaded on the Arduino.
11. Power LED Indicator is the tiny LED that gives us information if the Arduino is plugged into the power source.
12. TX RX LEDs (transmit and receive led) give us visual information whenever our Arduino is transmitting or receiving data.
13. Main IC (integrated circuit) is what we call the brain of Arduino. The most used ICs are ATmega lines from the ATMEL company.
14. Voltage Regulator does what it says, it controls the amount of voltage that enters the Arduino. It will turn away any extra voltage that may harm our board. The regulator has its limits so it is not recommended to go further than 20 volts.



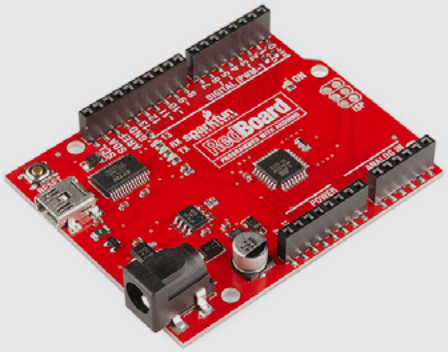
Types and Applications

In the Arduino world, we can find several different boards, with different capabilities. Since the Arduino is open-source hardware, people around the world can modify them so we can find a big variety of the same board.
In the future text, we will describe some of the most standard types.

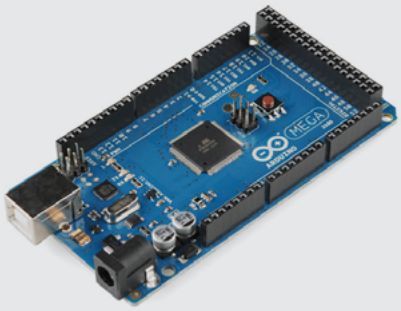
For beginners, the best choice is **Arduino Uno.** The main parts are already described, and it has everything needed to support the microcontroller.



LilyPad Arduino is a wearable e-textile technology. It is creatively designed with large connecting pads and a flat back to allow them to be sewn into clothing with conductive thread. The LilyPad has all the input, output, power, and all other sensors and pins that were specifically designed for e-textiles, and they are washable.



RedBoard Arduino is the combination of different features that are chosen from the creators. It can be programmed over a USB Mini – B cable using the Arduino IDE. It is more stable than the Arduino Uno, even if they have the same plug-in. The RedBoard is flat on the back and it can easily be embedded in projects. The on–power regulator can handle anything from 7 – 15 volts.



Arduino Mega is like a big brother of Arduino Uno. It has lots of digital input and output pins, 16 analog inputs, a USB connection, a power jack and a reset button. It has everything needed to support the microcontroller same as the Arduino Uno, but a huge number of pins make this board very useful for projects that require a lot of digital inputs and output.

Arduino Leonardo is the first Arduino product with a built-in USB. Since the board is handling the USB directly, code libraries are available which allow the board to imitate a computer keyboard, mouse, etc.

Arduino is widely used in modern society, but somehow the most interesting application is actually the ArduPilot. ArduPilot is a professional grade open source, unmanned vehicle Autopilot Software suite, capable of controlling autonomous multirotor drones, model helicopters, ground rovers, model submarines, etc. It is the most advanced, full-featured, and reliable open-source software available.

In addition to hobbyists, ArduPilot is used by a large number of leading companies around the world. The ArduPilot software suite consists of navigation software running on the vehicle, along with ground station controlling software including Mission Planner, APM Planner, QGroundControl, MavProxy, Tower, and others.

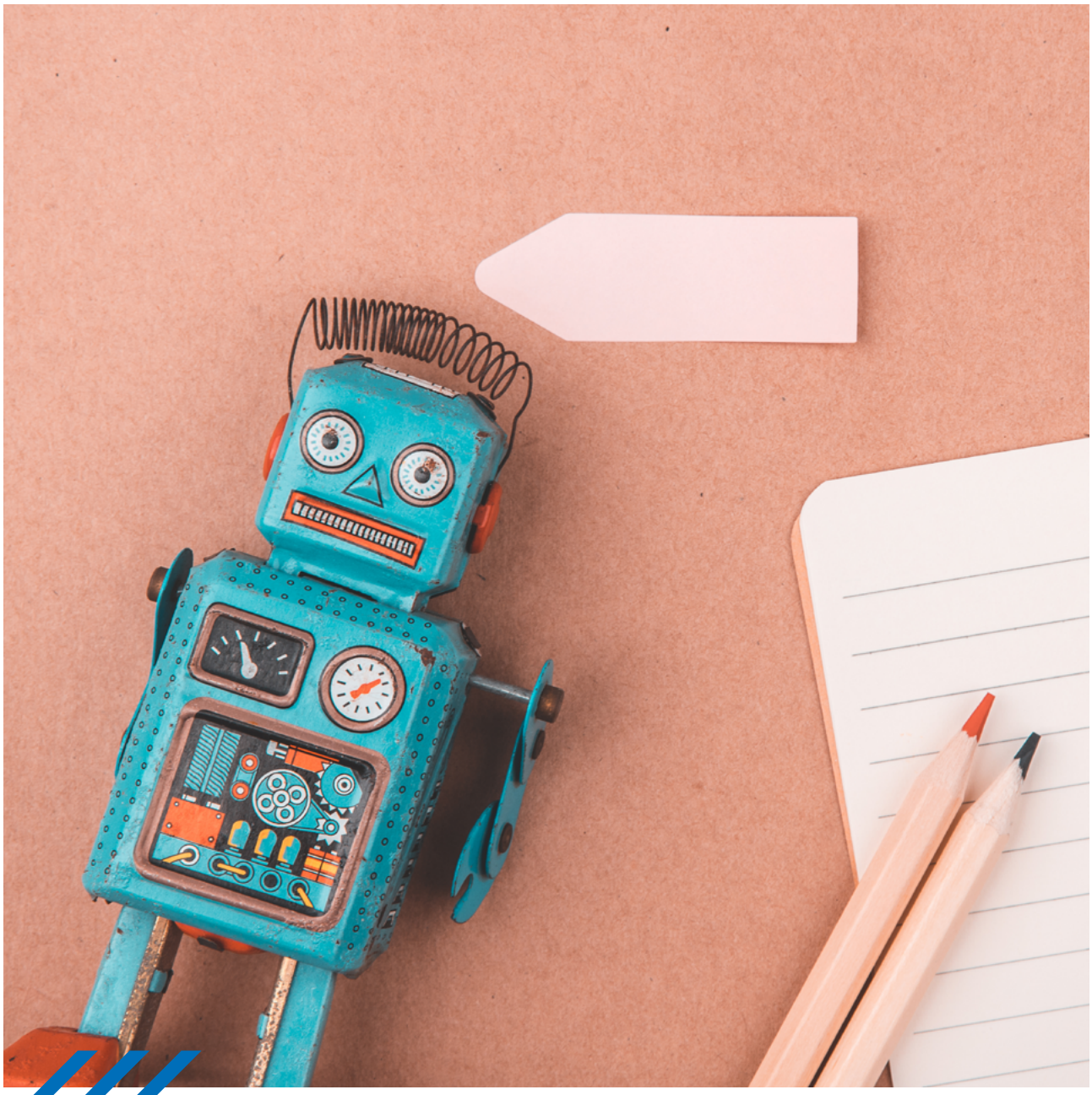
Copters and other vehicles have software that runs on a variety of embedded hardware consisting of one or more microcontrollers or microprocessors connected to the sensors used for navigation.

Some of the hardware platforms that ArduPilot is running are Intel Aero, APM 2.X, BeagleBone, The Cube, Navio2, Parrot, Pixhawk2, Qualcomm SnapDragon, etc.

The main reason for such a wide usage of ArduPilot is the number of features that this hardware/software is giving (providing).

Some of the main features

1. Fully autonomous, semi-autonomous, and fully manual flight modes.
2. Stabilization options (denial of the need of third-party co-pilot).
3. Simulation with a variety of simulators.
4. Large number navigation sensors, including several GSPs, barometers, laser, and sonar range finders, magnetometers, optical flow, etc.
5. Support for brushless and brushed motors.
6. Photographic and video gimbal support and integration.
7. Auto tuning and other features vary with the type of vehicle.



Middle Level Robot

In this module, we will deal with the mobility of a robot that, in order to move a robotic arm, performs movements and maneuvers. We want to build the track of the vehicle that moves in the plane and interacts with the objects of the surrounding environment.

The movement of the robot is based on 2 independent motors. The motors work in direct current and have been connected to a mechanical system for reducing the number of revolutions. This allows for a high traction torque and a feed speed suitable for use as a robot.

How to move a robot

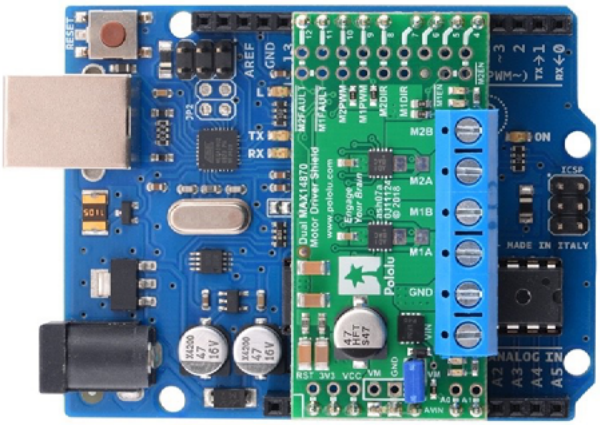
The movement of the robot through the motors

The movement of the robot is based on 2 independent motors. The motors are composed of DC motors (direct current), connected to a mechanical system for reducing the number of revolutions, which allows a high traction torque and a speed of advancement suitable for use as a robot. These two motors are driven by a power circuit that bears the principle of the H-bridge, which makes it possible to drive in PWM mode, and above all allows the inversion of the direction of rotation.

This motor driving technique referred to as P (pulse) W (Width) M (modulation), i.e. driving the motors through impulse modulation, allows us to increase or decrease the power to the motors which translate into the higher or lower speed of the motors. PWM is used to generate analog signals through a digital output for example of a microcontroller.

We used a Dual MAX14870 Motor Driver Shield for Arduino which makes it easy to control a pair of brushed bi-directional DC motors with the Arduino board or a compatible board.

The board features a pair of Maxim's MAX 14870 H-bridge ICs for the motor driver. The board features a pair of Maxim's MAX14870 H-bridge ICs for the motor driver, allowing the motor driver to operate from 4.5V to 36V and making it suitable for driving high voltage motors such as our 12 V 20 mm diameter gear motors. The shield can deliver 1.7 A continuous per channel and tolerate peak currents up to 2.5 A per channel for a few seconds. It comes with its SMD components, including MAX14870 drivers and a FET for



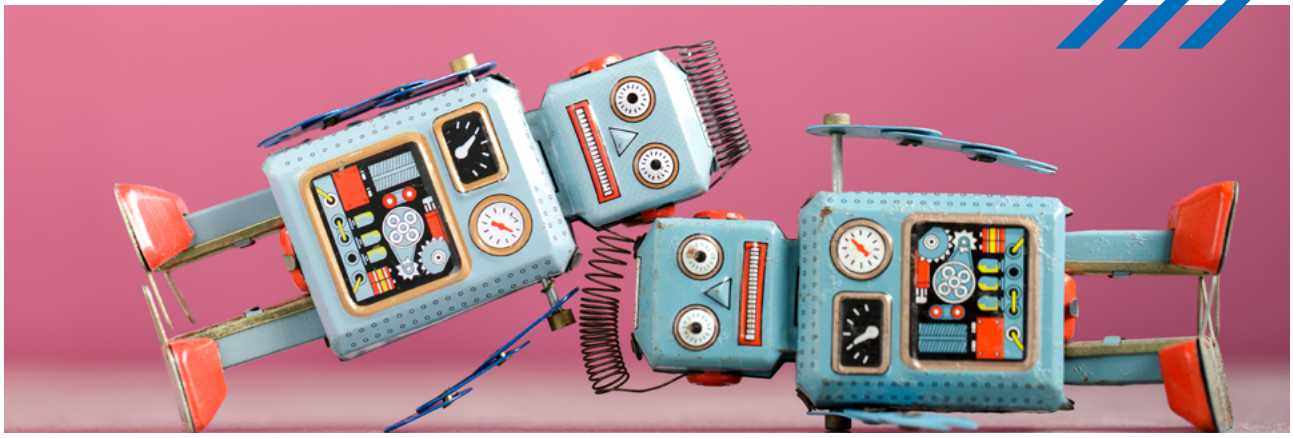
reverse voltage protection; header pins for interfacing with an Arduino and terminal blocks for connecting motors and power are included but not soldered (see Assembling with Included Hardware section below).

The shield uses digital pins 4, 7, 8, 9, 10, and 12 for its control lines, though the control pin mappings can be customized if the defaults are not convenient. It should be compatible with any board that has a standard Arduino pin arrangement and the ability to generate PWM signals on pins 9 and 10. Compatible control boards include:

- Arduino Uno,
- Arduino Mega 2560,

and others.

This shield is intended to provide a low-cost, basic motor driver option for Arduinos, so it is much smaller than typical Arduino shields and does not include pass-through, stackable headers.



How the robot manages to detect the environment

The robot follows a line

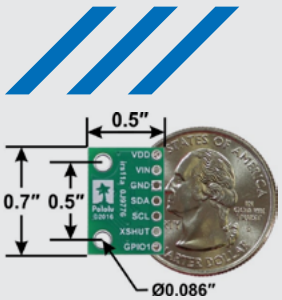
The robot is equipped with at least three sensors, which recognize the light or dark color.

The three sensors in question are composed of an infrared emitter and an infrared receiver. The receiver is hit by the reflected infrared light, so the reflection coming from the ground will be different depending on whether the ground is colored light or dark.

The optical signal will be converted into an electrical signal, which will be sent to the CPU which, through adequate programming, will give the wheels the right command to make sure they follow the line.

Three sensors are used because, when the control unit is active, the CPU understands that the robot's wheels are moving over the dark stripe on the ground. When the central sensor turns off and a side sensor is activated (right or left), the CPU understands that the line has curved towards the side in question, so it only instructs the wheel of the side whose sensor is activated to proceed to a reduced speed, or even stopping, until only the central sensor becomes active again.

Therefore, following this principle, our robot is able to follow a line drawn on the ground, even of considerable difficulty.



The robot measures a distance to the wall

We have used the VL53L1X laser-ranging sensor. This sensor is a laser distance sensor with a range of up to four meters which through the trajectory time (time of flight - ToF) of invisible, eye-safe laser pulses allows measuring absolute distances regardless of the ambient lighting conditions and the characteristics of the target such as color, shape, and texture (although these things affect the maximum flow rate).

The VL53L1X also features a programmable region of interest (ROI), so the entire field of view can be reduced or divided into multiple zones. Distance measurements can be read via an I²C digital interface. The board includes a 2.8V linear regulator and level shifters that allow it to operate over an input voltage range from 2.6V to 5.5V and the 0.1" pin spacing makes it easy for use with a standard solderless breadboard and 0.1" perf board.

The VL53L1X from ST Microelectronics is a long-distance-ranging time-of-flight (TOF) sensor integrated into a compact module. This board is a carrier for the VL53L1X, so we recommend careful reading of the VL53L1X datasheet before using this product.

Overview

The VL53L1X is effectively a tiny, self-contained lidar system featuring an integrated 940 nm Class 1 laser, which is invisible and eye-safe. Unlike conventional IR sensors that use the intensity of reflected light to estimate the distance to an object, the VL53L1X uses ST's FlightSense technology to precisely measure how long it takes for emitted pulses of infrared laser light to reach the nearest object and be reflected back to a detector. This approach ensures absolute distance measurements independent of ambient lighting conditions and target characteristics (e.g. color, shape, texture, and reflectivity), though these external conditions do affect the maximum range of the sensor, as do the sensor configuration settings.

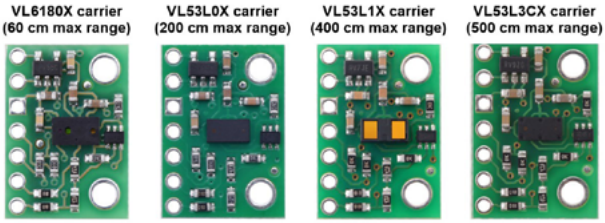
Under favorable conditions, such as low ambient light with a high-reflectivity target, the sensor can report distances up to 4 m (13 ft) with 1 mm resolution. See the datasheet for more information on how various external conditions and sensor configurations affect things like maximum range, repeatability, and ranging error. The minimum ranging distance is 4 cm; inside of this range, the sensor will still detect a target, but the measurement will not be accurate. Ranging measurements are available through the sensor's I²C (TWI) interface, which is also used to configure sensor settings, and the sensor provides two additional pins: a shutdown input and an interrupt output.

The VL53L1X offers three distance modes: short, medium, and long. Long-distance mode allows the longest possible ranging distance of 4 m, but the maximum range is significantly affected by ambient light. Short distance mode is mostly immune to ambient light, but the maximum ranging distance is typically limited to 1.3 m (4.4 ft). The maximum sampling rate in short distance mode is 50 Hz while the maximum sampling rate for medium and long-distance modes is 30 Hz. Performance can be improved in all modes by using lower sampling rates and longer timing budgets.

For advanced applications, the VL53L1X supports configurable thresholds that can be used to trigger interrupts when a target is detected below a certain distance, beyond a certain distance, outside of a range, or within a range. It also supports an alternate detection mode that generates an interrupt when no target is present. Additionally, unlike its predecessors, the VL53L1X supports a configurable region of interest (ROI) within its full 16×16 sensing array, allowing you to reduce the field of view (FoV). With all 265 detection elements enabled, the FoV is 27°. An "Autonomous Low Power" mode that is specially tuned for advanced presence detection is available. This mode allows for significant system power saving by switching off or waking up the host automatically when a human or object is detected within the configured distance thresholds in the region of interest.

The VL53L1X is a great IC, but its small, leadless, LGA package makes it difficult for the typical student or hobbyist to use. It also operates at a recommended voltage of 2.8 V, which can make interfacing difficult for microcontrollers operating at 3.3 V or 5 V. Our breakout board addresses these issues, making it easier to get started using the sensor while keeping the overall size as small as possible.

The carrier board includes a low-dropout linear voltage regulator that provides the 2.8 V required by the VL53L1X and allows the sensor to be powered from a 2.6 V to 5.5 V supply. The regulator output is available on the VDD pin and can supply almost 150 mA to external devices. The breakout board also includes a circuit that shifts the I²C clock and data lines to the same logic voltage level as the supplied VIN, making it simple to interface the board with 3.3 V or 5 V systems, and the board's 0.1" pin spacing makes it easy to use with standard solderless breadboards and 0.1" perfboards. The board ships are fully populated with its SMD components, including the VL53L1X.



For similar but shorter-range sensors, see our 200 cm VL53L0X carrier and 60 cm VL6180X carrier. Both of these are physical drop-in replacements for the VL53L1X carrier, but they have different APIs, so software for the VL53L1X will need to be rewritten to work with the VL53L0X or VL6180X.

Measurement of the Earth's magnetic field

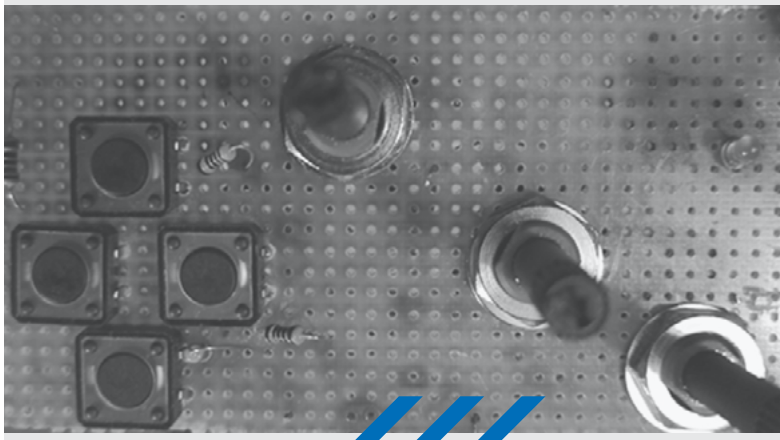
HMC5883L is a magnetometer module developed by Honeywell using anisotropic magnetoresistive technology. It is a multichip module that behaves as a digital compass IC to find the direction and measures the magnitude and direction of the magnetic field along the X, Y, and Z-axis. HMC5883L module converts the magnetic field into differential voltage output on 3 axis pins.

The magnetic field affects these sensors by somewhat modifying the current flowing through them. By applying a scale to this current, we can know the magnetic force (expressed in Gauss) exerted on each sensor.

The HMC5883L component communicates with Arduino through the I2C protocol, a very simple protocol to use.

Remote robot wire control

We use wire control to remotely control the robot's movement.

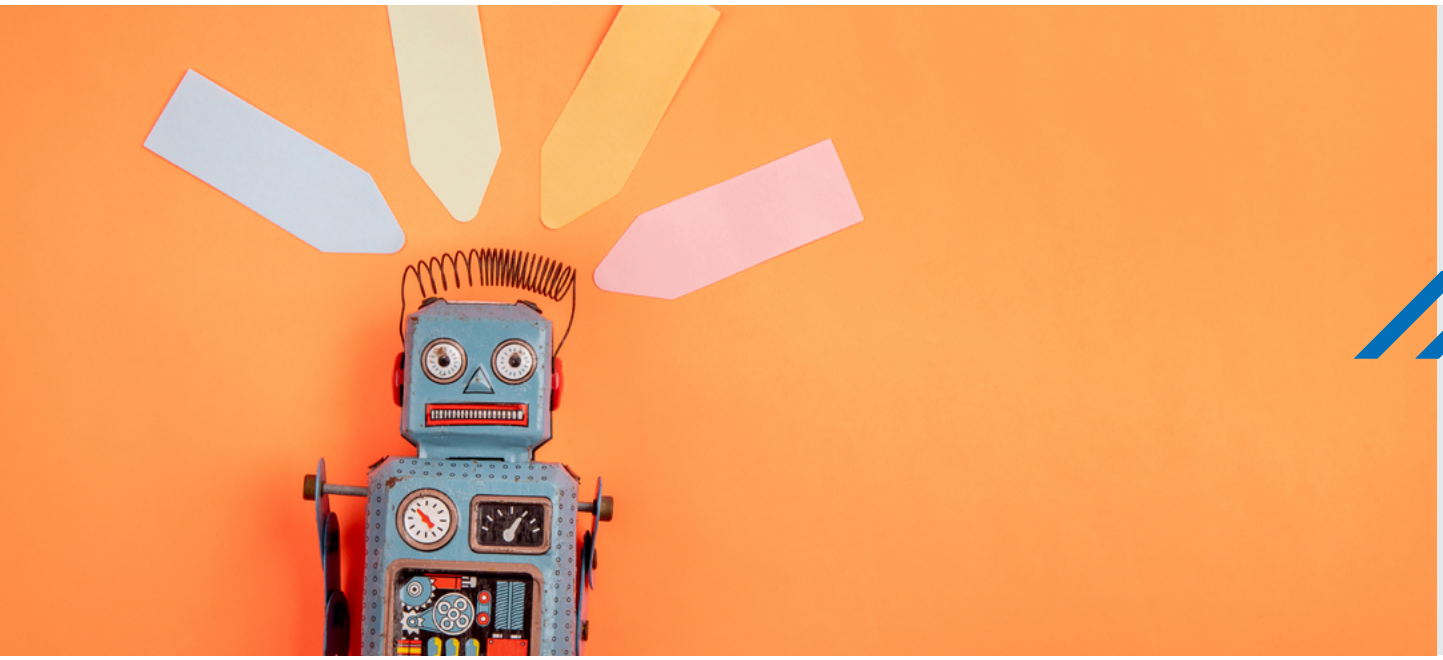


High Level Robot

Benefits of upgrade to high-level robot

In this lesson, we explore the core differences between mid-level and high-level robots, complete with functionality differences and benefits of the upgrade.

The lesson is designed as a comparison of the two levels. The purpose of this lesson is to give students an insight into what they can expect from the upgrade and how the core approach to building and coding will change.
The end goal of the lesson is to give students a grasp of the new functions of the upgraded robot and how they can be used in real-life situations.



Basic differences between mid-level and high-level robot

As we explore the upgrades required to move from mid-level to high level, we can classify them into distinctive classes:

- dynamic controls
- advanced sensors
- fusion of sensory data
- telemetry
- gripper module
- object handling

Controls of the robot

For the control of the robot, the high level has multiple options of connectivity. We can connect via blue tooth, wifi or nRF.
Bluetooth protocol, an affordable communication method in PAN network, with a maximum data rate of 1Mb/S, working in a nominal range of 100 meters using 2.4 G frequency, is a common wireless communication method.
HC05 module is a Bluetooth module using serial communication, mostly used in electronics projects.
HC05 Bluetooth module important specifications

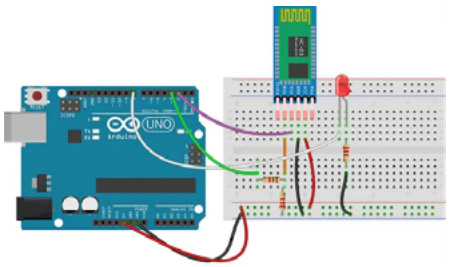
Working voltage: 3.6V – 5V
Internal antenna: Yes
Automatic connection to the last device: Yes

Sending Data to Arduino via Bluetooth

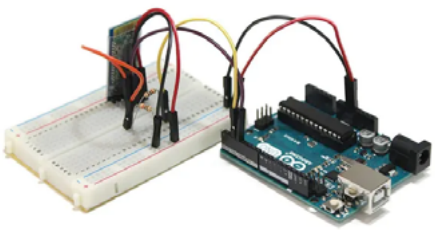
HC05 module has an internal 3.3v regulator, and that is why you can connect it to 5v voltage. But we strongly recommend 3.3V voltage since the logic of HC05 serial communication pins is 3.3V. Supplying 5V to the module can cause damage to the module.

To prevent the module from damaging and make it work properly, you should use a resistance division circuit (5v to 3.3v) between the Arduino TX pin and module RX pin.

When master and slave are connected, blue and red LEDs on the board blink every 2 seconds. If they aren't connected, only the blue one blinks every 2 seconds.

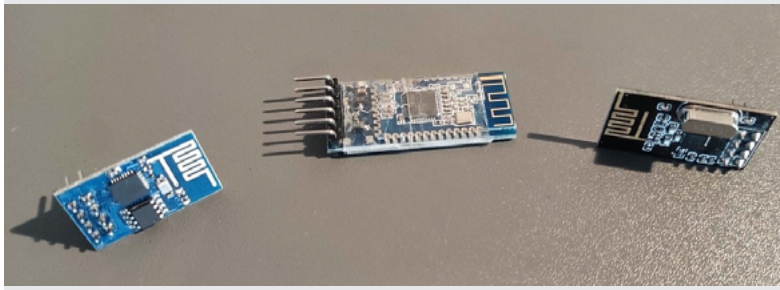


To communicate with HC05 using Bluetooth, you need a Bluetooth terminal application on your phone. You can use this one. To start transferring data, upload this code on your Arduino and connect HC05 using the app you have just installed. The communication name is HC05, the password is 1234 or 0000, and the transfer baud rate is 9600 by default.



By pressing and holding the button, the module switches into AT-command mode. Otherwise, it works in the communication mode. Some modules have a push button in their packages, and there is no need to add one anymore. The default baud rate to enter At-command mode is 38400. Now upload this code on your board and set commands using Serial Monitor.

You will receive the RESPONSE by sending a COMMAND to the module. Here are some of the most important AT commands:



Description	Response	Command
Just test	OK	AT
Reset module	OK	AT+RESET
Firmware version of module	+VERSION: OK	AT+VERSION?
Restore to default	OK	AT+ORGL
Module address	+ADDR: OK	AT+ADDR?
Module name	+NAME: OK	AT+NAME?
Change module name	OK	AT+NAME=your name\r\n
Bluetooth device name	+NAME: OK	AT+RNAME?param1\r\n
Module role(0 is slave & 1 is master)	+ROLE:	AT+ROLE?
Change module role	OK	AT+ROLE=(0 as slave,1 as master)\n\r
Module password	PSWD: OK	AT+PSWD?
Change module password	OK	AT+PSWD=your pass\r\n
P1: Baud rate	+UART=	
P2: Stop bit	OK	AT+UART?
P3: Parity		

Goals behind the upgrade

The main goal of the upgrade is the added ability to manipulate objects in the robot's environment. With the addition of the gripper module, the high-level robot can interact with its surroundings beyond that of reading the sensory data. This is also the change driving the rest of the upgrades as the manipulations of objects require upgrades to robots controls and additional sensors.

Benefits of the upgraded robot

The main benefits are:

- manipulation of objects
- examples of testing
- further discussion

Benefits of the upgraded robot

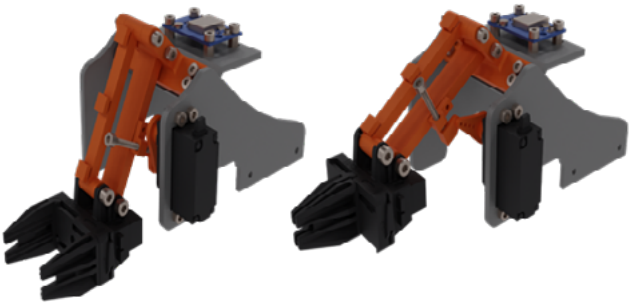
The main benefits are:

- manipulation of objects
- examples of testing
- real life application

Manipulation of objects

As stated before, the main benefit of the upgrade is the added ability to move objects in the robot's surroundings. This is achieved by introducing a gripper arm to the robot and adding a GPS module.

Two servo motors power gripper. While one lifts the arm, while the structure itself keeps the gripper level, hence removing the need for a third leveling servo, the other operates the arm's grip. This is pictured in the renders below. Note the position of the lifting servo that can be further tuned by the screw, proving higher or lower resting arm, depending on the robot's current task.



Examples of testing

We propose moving simple 3D printed parts while controlling the robot through one of the wireless modules described earlier and in the functionality testing lesson toward the end of this chapter to test the gripper itself.

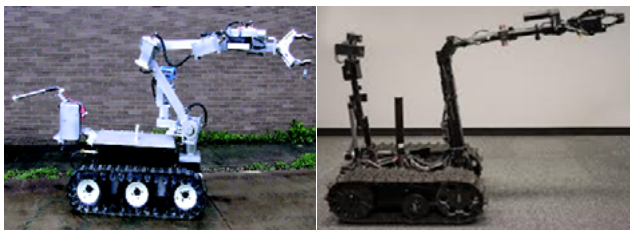
Another method is to program the robot in Arduino IDE software to do simple repetitive tasks, like moving the box in the predefined location to a new location or stacking boxes.

Benefits in terms of real-life application

Let us consider the real-life application of a semi-autonomous tracked robot equipped with a gripper arm. On a bigger scale, the same principles demonstrated with the MER high-level robot could be applied in various search and rescue scenarios. Further on equipping the robot with a camera module, like for instance ESPcam (pictured below) and using its GPS module to provide the exact location of victims, would give its operators a way to locate victims in collapsed buildings for example, while clearing the route for first responders, thus reducing the exposure of first responders personnel to hazards. The camera module should be swapped with an IR camera to increase visibility in certain conditions.

It is important to note that for any real-world search and rescue applications, the systems of MER robots should only be used as technology demonstrators, as they are not designed with the durability and redundancies required for such work.

Such robots in more or less autonomous states are already making grounds in search and rescue operations.



Note the similarities in design between these examples and MER high-level robot.

Robot assembly and wiring

As an overview lesson in this module, we will look at the assembly of a high-level robot. As we start from scratch, we will list and discuss all 3D printed parts and every other part or sensor that needs to be fitted.

We designed this lesson as a step-by-step guide to building the robot, which can overlap with the previous lessons, all the while supposing the students are familiar with the themes of levels 1 and 2.

Similarly, we explore the wiring of the sensors and microcontrollers of the robot and line out core code that we continue to explore in the next lesson.

Complete parts list required

A robot is made of:

- chassis
- the platform for Arduino with shield and sensors
- extension for IR and RBG sensor array
- gripper and arm module

For building the chassis for our robot, we will need:

- plate for mounting: 3D printed
- motors with gearbox: Polulo Tamiya 70168 Double Gearbox Kit
- tracks or wheels: Tamiya 70157 Universal Plate Set

Then we need to build the platform that contains the Arduino's brain, its shield, and sensors.

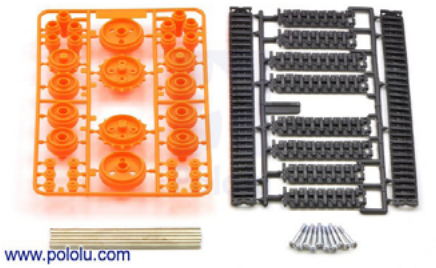
3D printing

Building and assembly - step by step

For building the chassis for our robot, we will need:

- plate for mounting
- motors with gearbox
- tracks or wheels

For the gearbox, we recommend using Polulo Tamiya 70168 Double Gearbox Kit. It is simple to use, and you can change the gearbox ratio yourself.

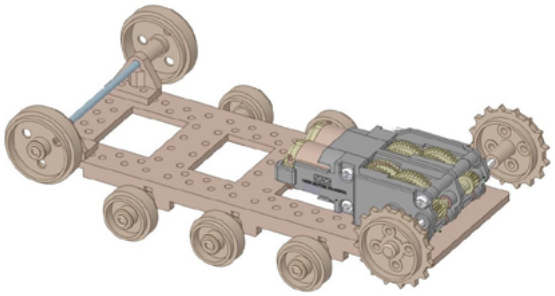


We are using Tamiya 70100 Track and Wheel Set.

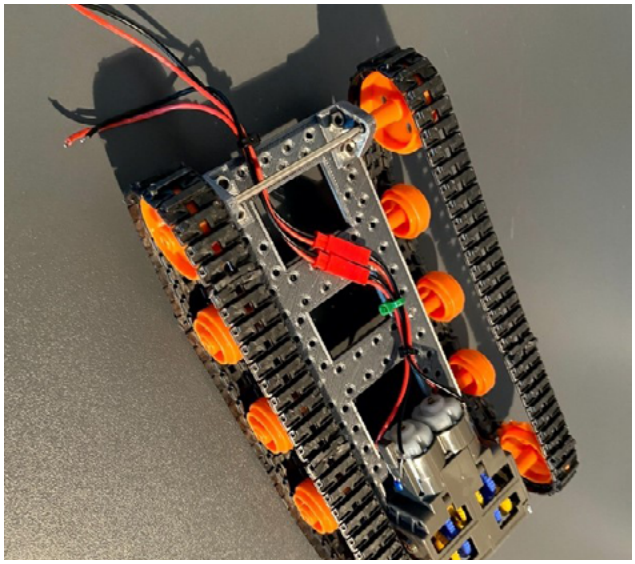
And for the mounting plate, you can use Tamiya 70157 Universal Plate Set or use our variant of the plate and 3D print it yourself.

3D models are available for download.

Assembling the mentioned parts is straightforward.



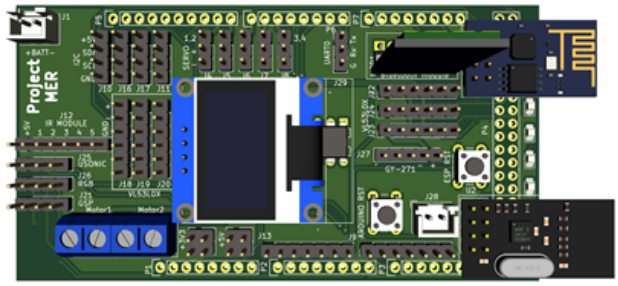
Here are the pictures of the assembled chassis.



Wiring the robot - step by step

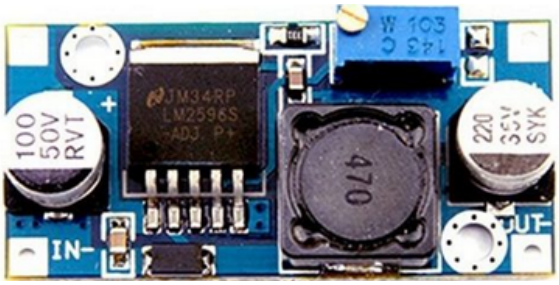
Preparations for robot wiring

Before we start wiring the robot itself, we need to ensure that all electronic components, including connectors, are present on the printed circuit board. Attach the printed circuit board with the Arduino development board to the robot chassis. An additional printed circuit board with IR and RGB sensors, a robot switch, a battery housing, and a distance sensor are also attached to the chassis.

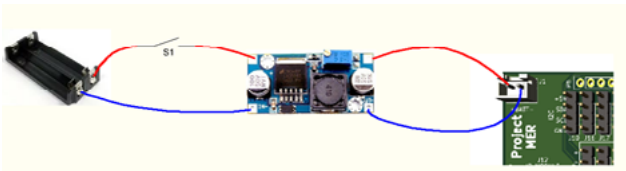


Electronics power supply

The robot is powered by two li-ion batteries connected in series. Because the voltage of the locked batteries is too high (8.4 V), lower it with the DC / DC converter down. An LM2596 integrated circuit module with a voltage selection potentiometer is selected for the inverter. Turn the potentiometer until 5 V is reached.



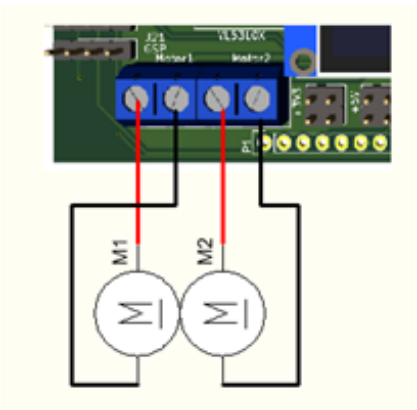
The set voltage is then applied to the connector on the printed circuit board.

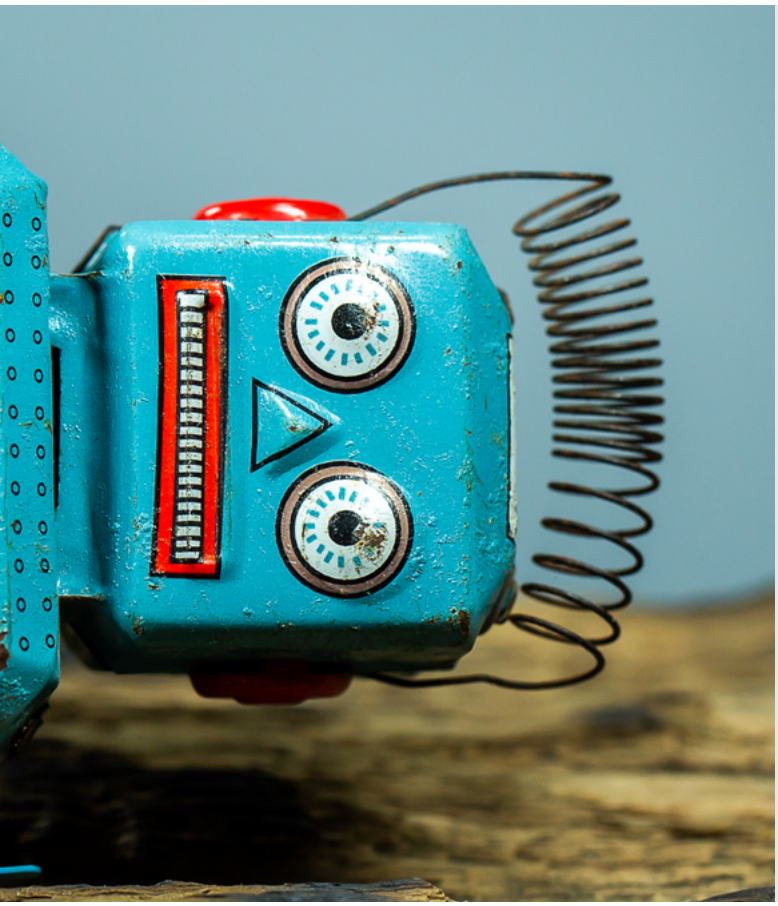


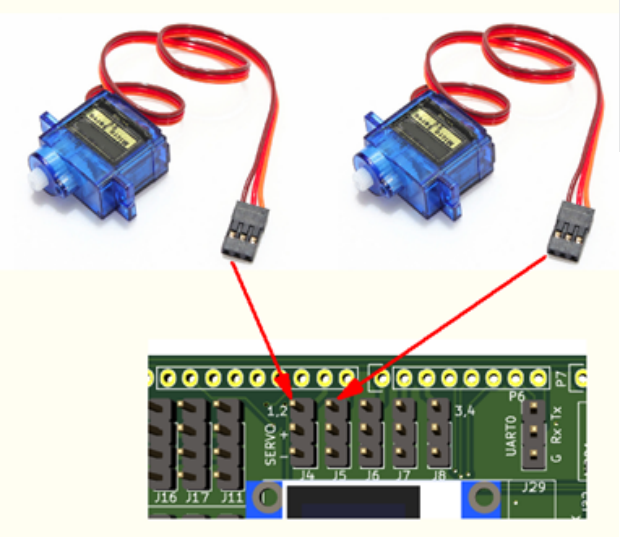
Connection of drive motors

The main function of the printed circuit board is to control the two motors that drive the tracks. The connection is provided on the side of the board, as we want the motor wires to travel as far as possible from the circuit, as high current flows through them, thus emitting an electromagnetic wave, which is a disturbance.

The motors have colored wires that emphasize the polarity of the motor.

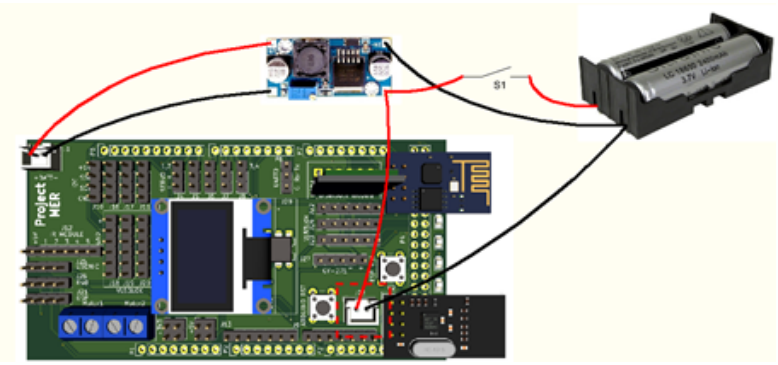






Servo motor connection

The circuit also has connectors for servo motors, which control the robotic arm. The connectors are connected to the so-called hardware PWM outputs of the Arduino, which means that the PWM signal is generated using the peripherals of the microprocessor and not programmatically.



Battery capacity connection

The same connector is available for measuring the battery and powering the circuit. It is recommended that the battery capacity measurement be performed behind the power switch. To this end, we will protect the batteries from being discharged when the robot is not working. We must also be careful not to measure the DC / DC converter, as the measurement will be incorrect (changed voltage).

Adding sensors and actuators

The first practical lesson of the course will be an overview of additional sensors and actuators needed to perform the upgrade.

We will list and explain the additional sensors, their inner workings, and wiring needed to add functionality to the high-level robot.

This lesson will focus on all the parts added to the main chassis and ignore the gripper arm, as that is a lesson on its own.

Additional sensors and actuators

As we upgrade the robot from mid to high level, we add several new sensors. These sensors are used in sensory data fusion to ensure greater awareness of the surroundings and give the operator more information. We use the telemetry functions built in the robot's code.

Added sensors :

- communication modules introduced in the previous lesson
- RGB sensors for determining the color of the ground
- gyro and accelerometer, which adds positional information to the telemetry
- GPS module for coordinates, which is used with telemetry data
- ToF camera for measuring distances from other objects as well as helping with positioning
- temperature and LDR sensor, both of which are used to determine the environmental conditions the robot operates in

Adding additional sensors and actuators

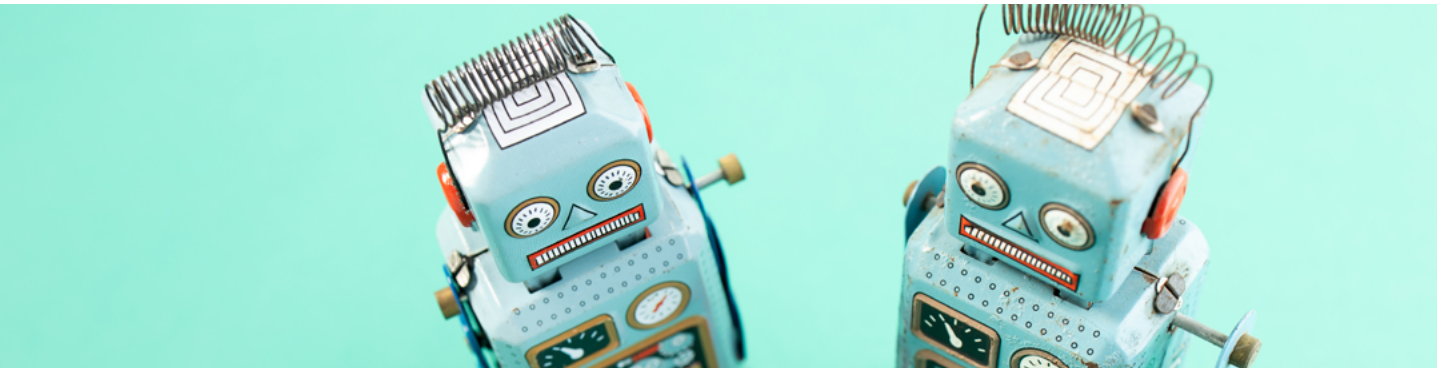
How do Colour Sensors work?

Colour sensors are developed based on diffuse technology that can detect various colors. The combination of color-sensitive filters and sensors array performs color sense, further used to analyze the color present in an image or a specified object. The color measurement process involves a light source to illuminate the surface, the target surface, and a receiver that measures the reflected wavelengths. A white light emitter is used to illuminate the surface. The sensor then activates three filters with three-wavelength sensitivities to measure RGB colors' wavelengths. Based on these three colors, the color of the material is determined.

Industrial Applications of Colour Sensors

Colour sensors are majorly used to grade colored products, distinguish coded markings, detect the presence of adhesives or data codes on a package. The technology has a wide range of applications in various industries such as textile, automation, automotive, food, printing, pharmaceutical, and many more. These sensors can also be programmed to identify anyone color or multiple colors for sorting applications, based on the level of sophistication required in the color measurement process.

In most industrial applications, color sensors are mainly used as visual inspection tools in the quality control stage. For example, in the food industry, color sensors monitor color changes in the plastic in which the meat product is wrapped to detect whether the meat quality has deteriorated. It can also be used for process control, such as controlling the temperature to achieve ideal roasting time for coffee beans.



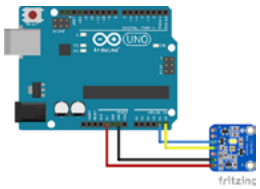
Adafruit TCS34725 specifications

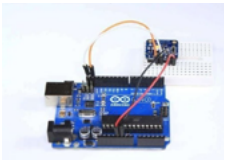
Weight: 3.23g

Dimensions: 20.44mm x 20.28mm

Board uses I2C 7-bit address 0x29.

Wiring the sensor





To connect and test the RGB sensor, we run either 3V or 5V power to VDD and common ground to Ground pin on the breakout board. SCL is wired to I2C Clock and SDA to I2C Data.

To test, there are several examples available along with the library that needs to be downloaded and imported into Arduino IDE.

Gyro and accelerometer

We will use the combined module with gyro and accelerometer for our purposes, thus reducing wiring and simplifying the design. MPU 6050 is a six-axis Gyroscope and accelerometer.

A gyroscope is a device that takes the help of the earth's gravity in determining its orientation. It is a type of sensor which we find inside IMU (Inertial Measurement Unit). A gyroscope can be used to measure the rotation on a particular axis. The device consists of a rotor which is nothing but a freely rotating disc. The rotor is mounted on a spinning axis present in the center of another larger wheel.

An accelerometer works using an electromechanical sensor designed to measure either static or dynamic acceleration. Static acceleration is the constant force acting on a body, like gravity or friction. These forces are predictable and uniform to a large extend. For example, the acceleration due to gravity is constant at 9.8m/s, and the gravitation force is almost the same at every point on earth.

Dynamic acceleration forces are non-uniform, and the best example is vibration or shock. A car crash is an excellent example of dynamic acceleration. Here, the acceleration change is sudden when compared to its previous state. The theory behind accelerometers is that they can detect acceleration and convert it into measurable quantities like electrical signals.

Technical specifications

Chip: MPU-6050

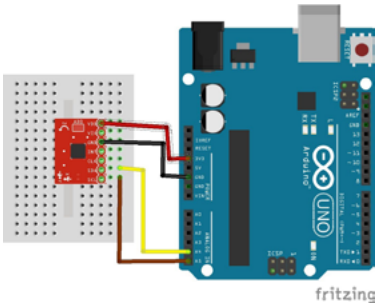
Supply voltage: 3V-5V

Chip built-in 16bit AD converter, 16-bit data output

Communication modes: standard IIC communications protocol

Wiring the sensor

We will connect VCC to 5V and GND to a common ground to test the sensor. SCL to A5 (analog input 5) and SDA to A4 (analog input 4).



GPS module

GPS stands for Global Positioning System, by which anyone can always obtain the position information anywhere in the world.

GPS works with three distinct segments:

1. GPS satellites
2. Ground control stations
3. GPS receivers

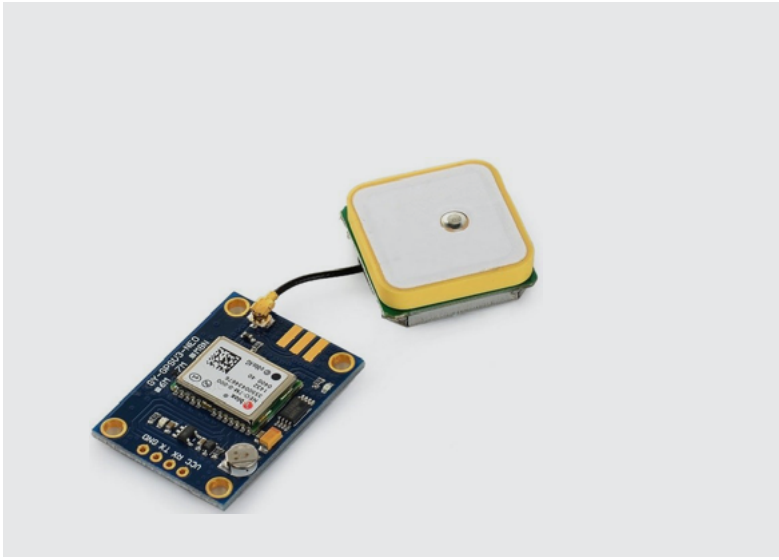
GPS first sends a signal of time from a GPS satellite as the receiver receives the signal. The difference between time sent and time received is used to calculate the distance between satellite and receiver. As we do this with multiple available satellites, we can calculate the receiver's position. However, the position generated using this method is not accurate. An error is calculated distance between satellites and a GPS receiver arising from a clock's time error incorporated into a GPS receiver. The atomic clock is incorporated for a satellite to generate on-the-spot time information. Still, the time generated by clocks incorporated into GPS receivers is not as precise as the time generated by atomic clocks on satellites.

Here, the fourth satellite comes to play its role: the distance from the fourth satellite to the receiver can be used to compute the position concerning the position data generated by the distance between three satellites and the receiver, hence reducing the margin of error in position accuracy.

NEO-6M module

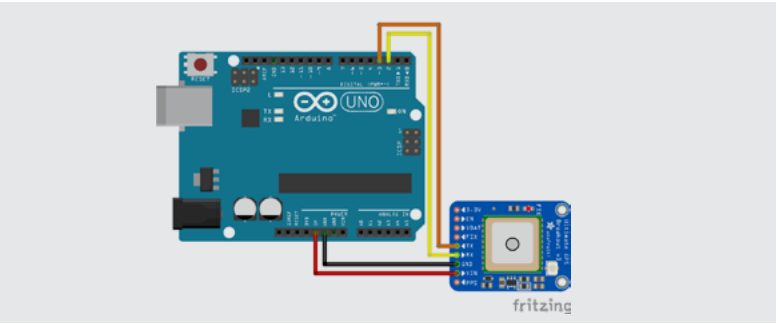
Technical specs:

- Receiver type: 50 Channels GPS L1 frequency, C/A Code SBAS: WAAS, EGNOS, MSAS
- Time-To-First-Fix with cold start : 27 s
- Sensitivity: -161 dBm
- Horizontal position accuracy GPS: 2,5m
- Velocity and heading accuracy: 0,1 m/s 0,5deg
- Interface: RS232 TTL
- Power supply: 3V to 5V
- Default baudrate: 9600 bps
- Works with standard NMEA sentences



Wiring the sensor:

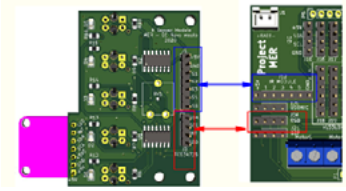
We will wire it as follows: VCC pins on NEO-6M to 5V power supply on Arduino and GND pin to common ground to test the sensor. RX and TX pins on NEO are connected to TX and RX pins on Arduino as defined in the software serial. In our case, we use pins 2 and 3 for serial.



Wiring of the sensors

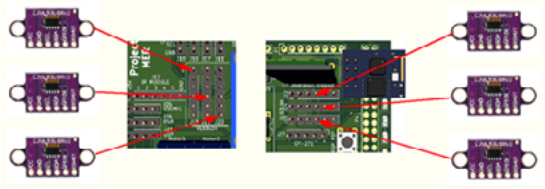
PCB connection with IR and RGB sensor

An additional circuit must be installed on the bottom of the chassis before wiring. The circuit has a ready output connector connected to the main printed circuit board according to the markings.



Connecting distance meters

VL53L0x modules are used to measure the distance, which uses a laser to determine the distance up to 2 m very accurately. They use I2C for communication, which means they can connect more to the same bus. For this purpose, the printed circuit board has three connectors on both sides for the given modules.

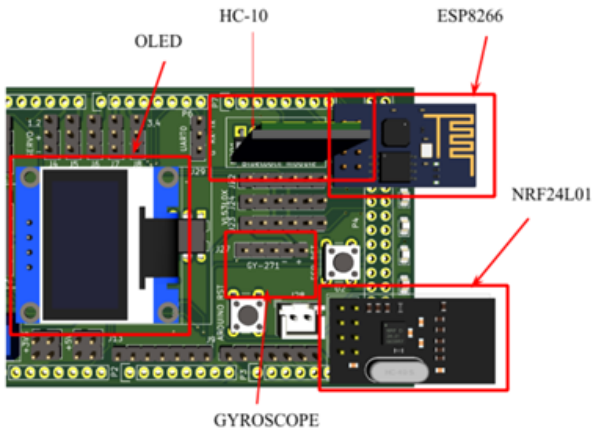


Connection of additional modules

There are connectors on the printed circuit board designed to connect modules. The module is already an ideal printed circuit board, making it easier for the user to use an individual integrated circuit.

The following modules connect directly to the PCB:

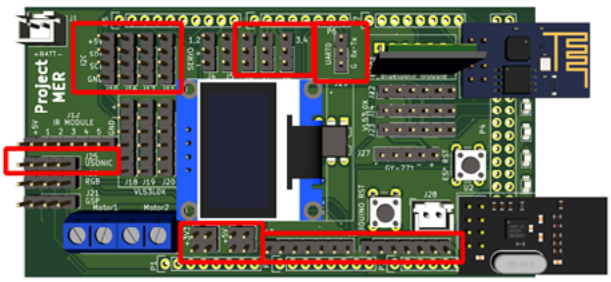
· 0.96" OLED display with the I2C communication,
· ESP8266 WiFi modul,
· NRF24L01 2.4 GHz transciver,
· HC-10 Bluetooth modul,
· Gyroskope GY-271.



A special feature appears in the OLED display. As there are several different providers of the same modules on the market, there may be a difference in the order of the connectors, most of which are Vcc and GND. For this purpose, two connectors are present on the printed circuit board. When connecting, we must be careful which one we use because the display will be destroyed if the connection is incorrect.

Free connections

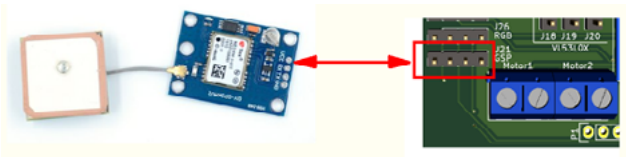
The connectors on the printed circuit board have the same layout as the connectors on the modules. The circuit board is full of unused connectors that the user can fill in himself. Some of these are connected to certain communications (I2C, UART,...), some are intended for modules that are not installed (ultrasonic distance meter, additional connectors for servo motors), and others are intended for general use.

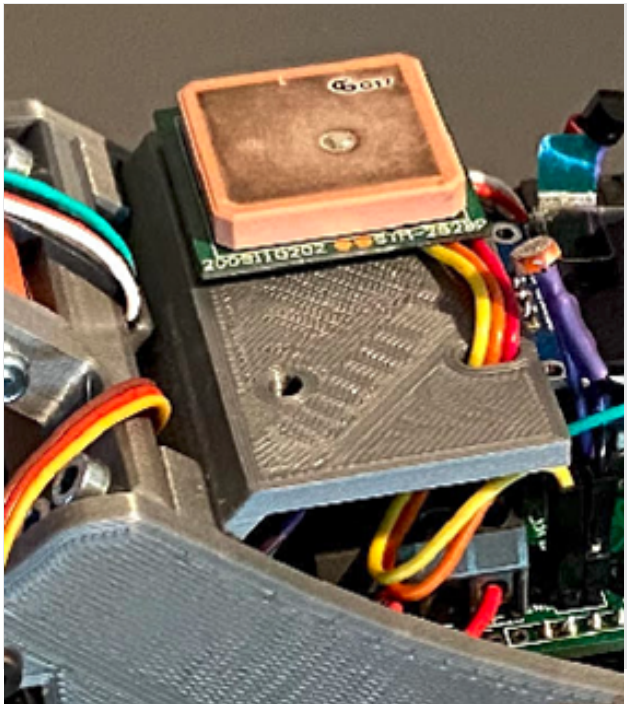


Basic wiring of the GPS module

Using the list of components below and using the lesson about GPS module, make a basic circuit using Arduino board and GPS module.

The printed circuit board also has a connector for a GPS module. It is advisable to connect the latter, as the module is mounted on a plastic housing located on the printed circuit board. The module has only four connectors, two for power and 2 for serial communication UART. The terminal layout of the module matches the layout of the PCB connectors.





List of components:

- Arduino Nano or Uno
- Neo 6M GPS module

Adding the gripper module

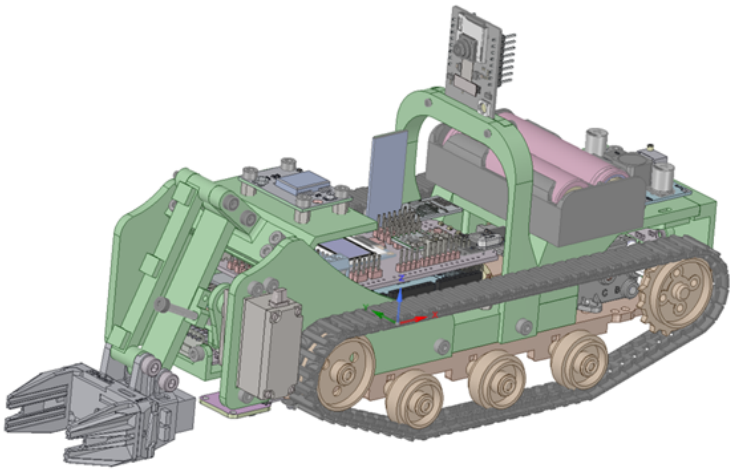
During this lesson, we will discuss what functionalities we can expect from our gripper module, how to apply it to our robot.
We will learn which parts we need to 3D print and their function. We will also list the additional parts needed.
Lastly, we will look at servo motors, their core principles, and their usage in our project.

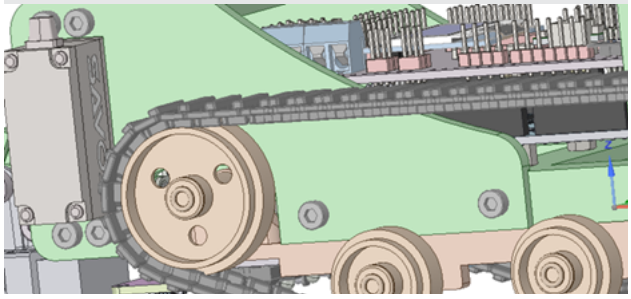
List of parts needed

- 1 x Pololu gripper: https://www.pololu.com/product/3551
- 1 x Servo Motor: https://www.savox-servo.com/Import/Savox-Servo-SH-1250MG-Digital-Coreless-Motor-Metal-Gear/
- 1 x 3D printed parts
- 6 x M3x6 mm brass nut: https://studia3d.com/sl/latunnaya-gajka-vstavnaya-vtulka-s-rezboj-m1-6-dlina-2-mm/
- 8 x screws M3x16
- 4 x screws M3x30
- 2 x screws M4x25
- 2 x screws M2x10
- 15 x nut M3
- 2 x nut M4

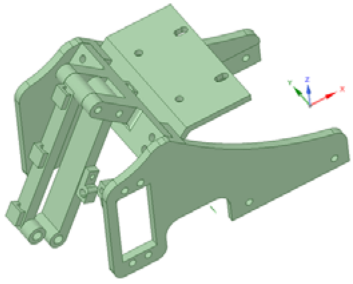
Assembly of the gripper module

The robot arm is designed to be screwed to the robot chassis with four M3 screws. Brass inserts with M3 thread are inserted into the robot chassis.

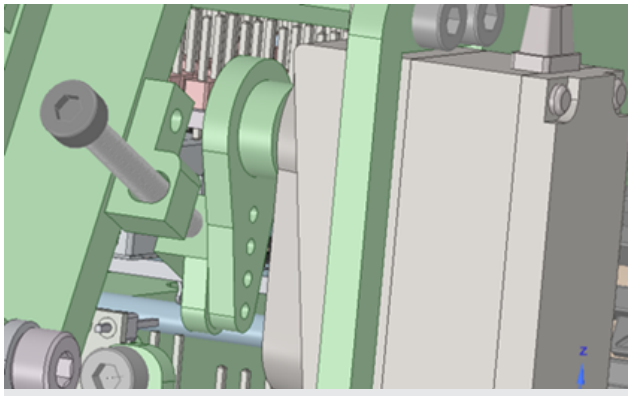


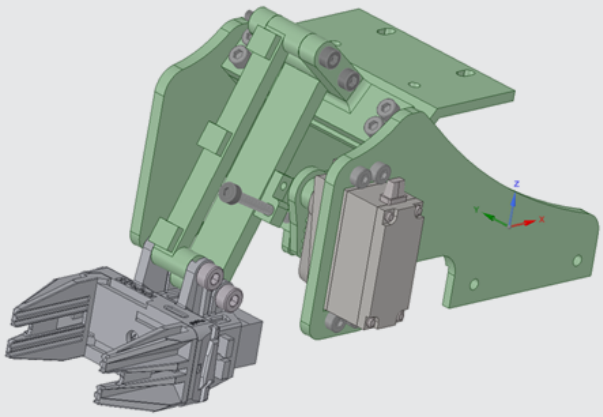


All parts printed on a 3D printer are designed so that they can be printed without support.



The robotic arm is assembled with M3 screws of suitable length. Only the gripper at the end of the robot arm is attached with M4 screws. The M3x30 screw is used to connect the servomotor to the lifting mechanism. The height of the gripper in the lower position can be adjusted on this screw.





Functionality testing

The final lesson of the course provides the cumulation of what we did in all previous lessons.

We will learn how to write and upload basic code that will help us test all the robot's functions.

We will also explore some code examples that will give students the foundations needed to develop their code and the robot's functionalities.

At the end of the lesson, the students should have a working robot coded in basic functionalities.

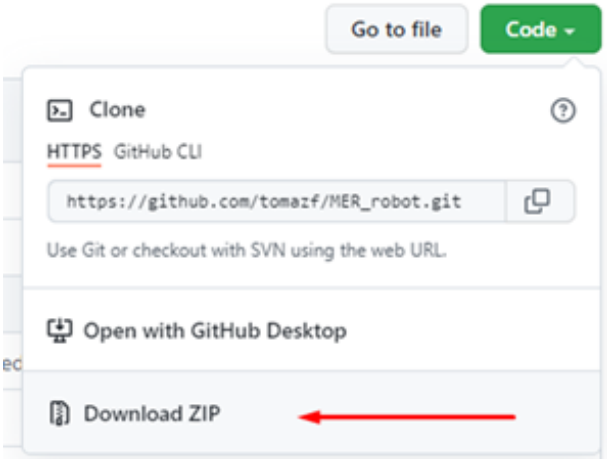
All the available code for the MER robot is available on GitHub. GitHub is a website and cloud-based service that helps developers store and manage their code and track and control changes to their code. All other users can download and browse the code and write and add their own code.

The MER repository is available at the following address: https://github.com/tomazf/MER_robot

Exploring the core code for high level functions

Exploring the code is quite simple – browse the page and view the code. The Readme file is presented on the first page, which gives basic info about the project.

To download the project code to test or develop new robot functionalities, browse the GitHub address mentioned before and click Code / Download.



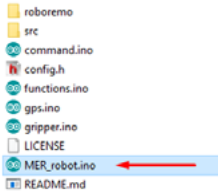
All the project files will be downloaded to your local computer. You need to decompress the contents to your local drive, i.e. Desktop. The folder name must be the same as in the ZIP – Mer_robot to function properly.

Most of the needed files for the project and source compilation will be in the extracted folder. Some libraries must be added manually because they reside outside of the project.

All the source is written in C++ programming language with Arduino IDE. This provides the easiest way to develop the code. There are other options for coding (Visual Studio Code, Sublime Text, Brackets etc.), which can be further used with more experienced users.

Users should install the latest version of Arduino IDE (Integrated Development Environment) from https://www.arduino.cc/en/software. The IDE compiles (converts) the code you write into instructions the Arduino can understand. The code is typed (or copied/pasted) into the IDE and sent to the Arduino via a USB cable.

Once the IDE is installed, browse the extracted source code, and double click the Mer_robot.ino file.



The IDE should open and present you with the main source file. On the upper row, you can see project files needed for compilation.

You should have:

- Mer_robot (main project file and loop)
- command (robot serial command parser)
- config.h (HW connections and some static variables, defines)
- functions (supported robot functions and functionalities)
- GPS (code for GPS sensor)
- gripper (code for gripper manipulation)

At the beginning of the source, there is a lot of useful info regarding used HW and libraries. Before compilation, some libraries should be installed manually in the Arduino environment. The process is simple – find the required libs, download them and extract them to the Arduino lib folder.

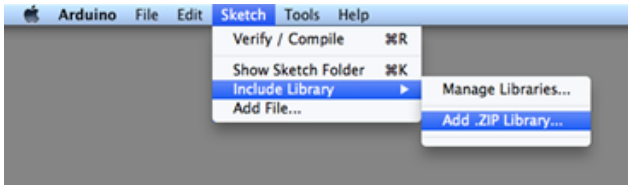
Library list:

- compass: https://github.com/helscream/HMC5883L_Header_Arduino_Auto_calibration/tree/master/Core/Compass_header_example_ver_0_2
- softI2C: https://github.com/Fire7/Adafruit_TCS34725_SoftI2C for TCS34725
- TCS_int: https://github.com/adafruit/Adafruit_TCS34725/blob/master/examples/interrupt/interrupt.ino
- command: https://github.com/ppedro74/Arduino-SerialCommands
- StateMAchine: https://github.com/jrullan/StateMachine
- LinkedList: https://github.com/ivanseidel/LinkedList (delete test.cpp file!)
- GPS: https://github.com/stevemarple/MicroNMEA
- freeMEM: https://github.com/maniacbug/MemoryFree
- neotimer: https://github.com/jrullan/neotimer

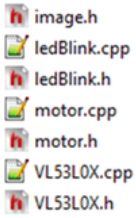
Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in LiquidCrystal library makes talking to character LCDs easy. There are hundreds of additional libraries available on the Internet for download. The built-in and some of these additional libraries are included during the install process. To use the additional libraries, you will need to install them.

These libraries should be installed/copied to the Arduino lib folder or installed with the IDE. Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside the folder will be a .cpp file, a .h file, and often a keywords.txt file, examples folder, and other files required by the library. Starting with version 1.0.5, you can install 3rd party libraries in the IDE. Do not unzip the downloaded library; leave it as is.

In the Arduino IDE, navigate to Sketch > Include Library > Add .ZIP Library. At the top of the drop-down list, select the option to "Add .ZIP Library''.



You will be prompted to select the library you would like to add. Please navigate to the .zip file's location and open it. Return to the Sketch> Include Library menu. Menu. You should now see the library at the bottom of the drop-down menu. It is ready to be used in your Sketch. The zip file will have been expanded in the libraries folder in your Arduino sketches directory.



Some custom-written libraries are used in this project – for VL53X sensors, motor manipulation, and timers. All these can be found in the src subfolder of the Mer_robot project.

For the above libraries, installation is not needed because the compiler will find them in the src subfolder.

Once the code is in place and installed all the libraries, you can compile the project and upload the initial code to the robot. Before uploading, it is suggested to browse the code and see how it is written. In the beginning, there are some includes, definitions, and objects.

All Arduino programs have a setup function executed at the beginning. Here we initialize hardware components used (sensors, motors, etc.) and write some debug data to serial console or/and onboard OLED module.

```
// ---------------------------------------------------------------------
// ---------------------------------------------------------------------
// SETUP
//
void setup() {

  // serial baud setup
  Serial.begin(115200);
  Serial2.begin(9600);              // BT module
#ifdef USE_GPS
  GPS_PORT.begin(GPS_BAUD);         // set GPS baud
#endif
  delay(10);                        // wait until serial port opens for native USB devices

  // start I2C
  Wire.begin();

  Serial.println("INIT...");
```

After the setup is complete, the main loop starts. The main function integrates core functions to manipulate robot actuators, sensors, and commands. Code uses compiler macros, so if some part of the code is not used, it does not compile it. The result is a smaller code and memory footprint. All the functions are written in the functions.ino file. Study them carefully.

```
// ---------------------------------------------------------------------
// MAIN LOOP
//
void loop() {
  if (I2C_scanner) i2c_scanner();
  else
  {

    // commands parse
    serial_commands_.ReadSerial();

    // for debug time-loop only - not for production
    //Serial.println(millis());

    // we must do better here!
    if (i2c_device[1][5] == 1) read_Compass();
    read_Batt();
    read_RGB();
    read_VL53x();
    read_IR();

#if defined(USE_GRIPPER)
    // do gripper and arm stuff
    gripper_run();
    arm_run();
#endif
```

Uploading the code to the robot

The robot can be controlled in a few ways:

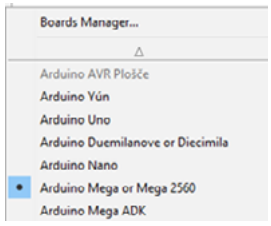
- directly using serial console over USB cable or/and,
- over Bluetooth or/and,
- over ESP8266 module or WiFi or/and,
- over nRF24L01+ transceiver module (no code written yet).

Regardless of the above selection, all the commands are the same. So whichever method you choose, the commands stay the same. The robot is designed to follow serial commands. All the available commands can be found in the commands.ino file.

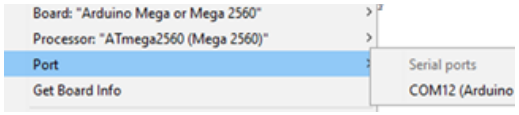
You should browse through the code to enable or disable appropriate HW used. You can also use two or three methods of controlling the robot simultaneously.

Once the code is set and the config.h file is suited for working HW; you can start compiling and uploading the code to the robot.

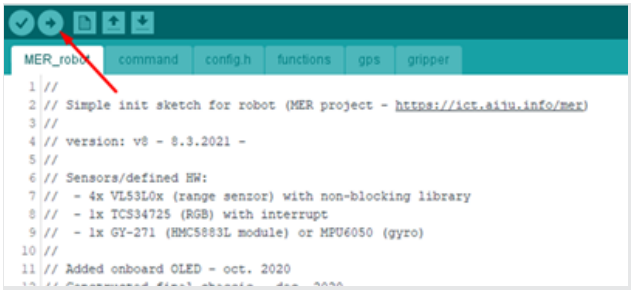
First, we must select the appropriate Arduino board. The project uses Arduino Mega, so we select the board.



After the board selection, we must select the available serial COM port the Arduino board uses.



Then we can press and try to compile the code.

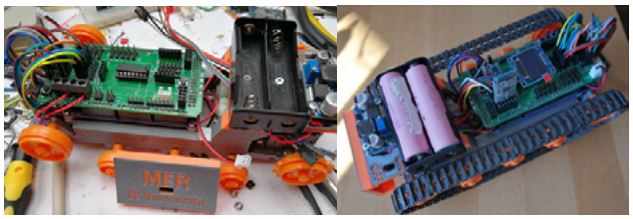


If all goes well, no errors are printed, code is compiled and uploaded to the robot.

When the robot boots (initializes), some debug info and MER project logo are displayed on OLED display. Also, motors are turned on for a few seconds – no need to be alarmed. This indicates that the HW is functioning properly.

Testing the robot functions

Now we can test the robot. It is suggested to test each robot HW separately when you build and connect HW to the main PCB board. Connect and test the motors, then sensors, then actuators, and at the end serial commands and wireless connections.



COMMAND reference

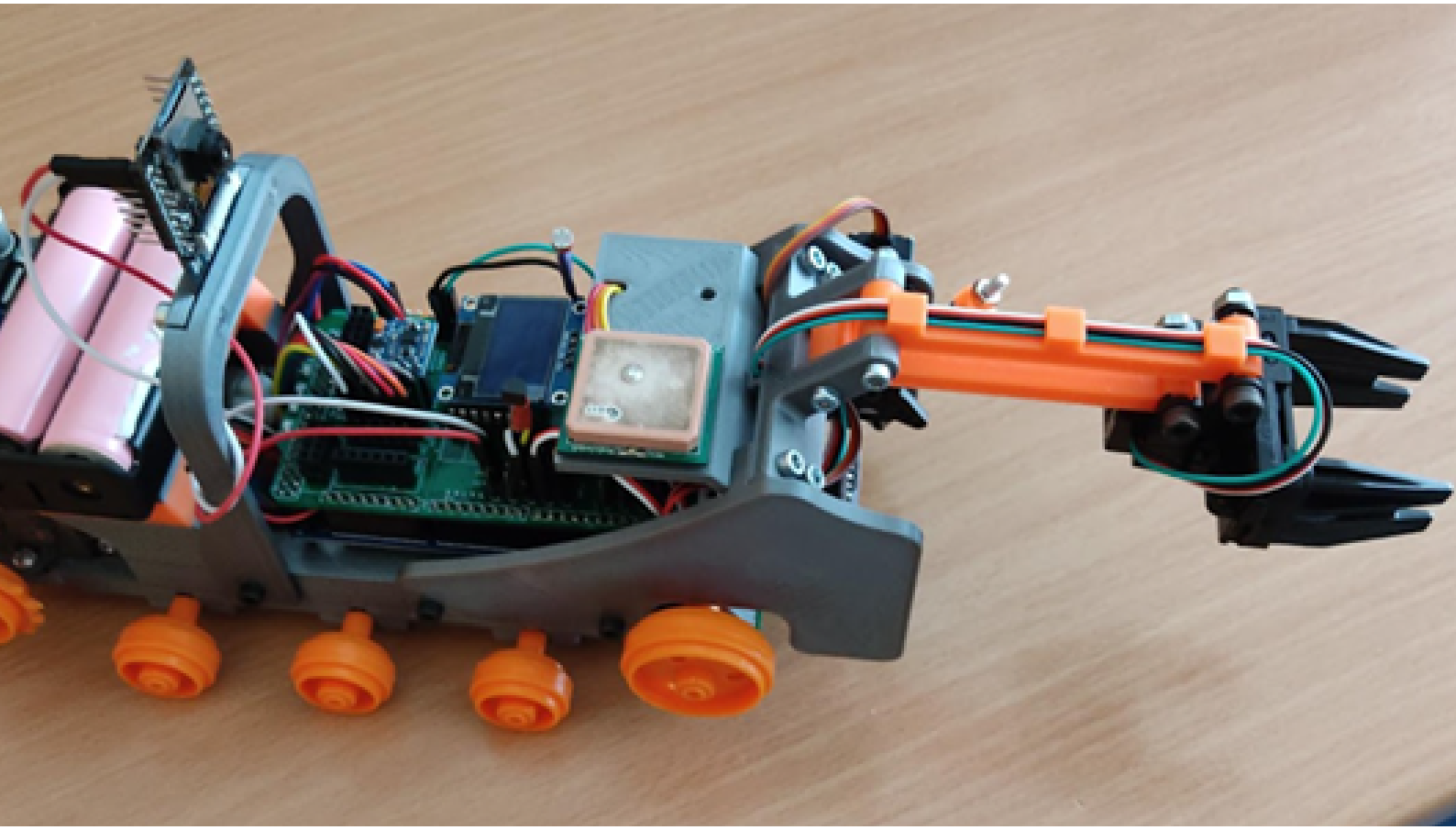
This section briefly describes the serial command parser for the MER robot. Commands for controlling or getting status from the robot are processed on Arduino UART with 115200 baud. Serial ports for communication are defined in software (can be multiple). Commands are parsed in the command.ino file.

All commands must be sent with CR and LF line endings (this can be modified in the source code). All commands receive ACK messages. If unknown command is sent/received, the sender is notified. Commands can have multiple parameters, as described. To test – opens serial monitor and enter some commands. The robot will respond.

Commands are divided into 3 groups (the first character defines group):

- L (LED commands) for manipulating onboard LEDs
- F (FUNCTION mode commands) for setting different modes and variables of a robot
- G (GET data commands) for getting sensor data and updates if defined in software

As mentioned in previous chapters, the main loop is responsible for all the repeating tasks. All mounted sensors are read using asynchronous code, so there is no time waiting for functions, and separate tasks are done in almost a minimum amount of time (less than 1ms) . The OLED screen is refreshed once every second, but measurements still get their data. This greatly improves the functionality and the ocessing speed of the robot.



Where here to go next

In the picture above, we can see some of the HW used. There is also a mounted camera at the back for autonomous level, which will be presented in the final chapter of this tutorial. The camera acts as a separate device connected to a PC. The robot still receives its commands over the serial channel. Computer vision is implemented, and objects can be recognized.

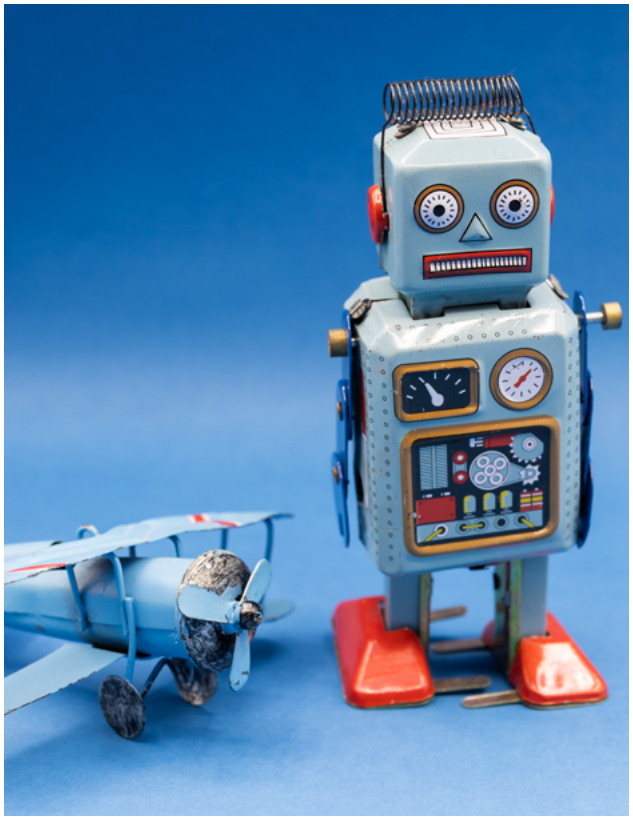
The way robot hardware is designed makes it easier for further software development and hardware upgrades. One can learn and build just the basic parts and use some sensors or add a full pack of features and extend it even further.

Autonomous Level Robot

Introduction

As a natural development and upgrade to a human-controlled or task-controlled robot, like the ones studied previously, the next step would be a robot that can perform certain tasks autonomously. In other words, it can decide on its own, based on collected data from sensors.

So, we will call this group of lessons autonomous robots.



Lessons

The following sets of lessons are organized as follows:

- Lesson one talks about vision, sensors fusion, and other important sensor and image processing aspects related to autonomous robots.
- Lesson two talks about a robot that goes to one point and comes back in different modes.
- Lesson three talks about mapping the environment using images and/or other sensors.
- Lesson four uses the concepts of lessons one, two, and three to perform a rescue a task.

What is it?

It is a robot that can make decisions based on data collected from sensors to perform a certain task or tasks. Examples of popular tasks include vacuum cleaner robots, mars rovers, choosing the best route for cars, and moving accordingly.

Why do we need autonomous robots? We could associate the need or the use of autonomous robots to the applications they are involved in. Some of but not all the reasons are:

Safety

In some applications, humans or human-controlled robots can have accidents and cause injuries due to the subjective decision-making process of human beings. As an example, parking a car with side mirrors only. This might cause the car to hit another car accident or a wall or part of the pavement, as side mirrors reflection is approximate and it is subjective to human eye side and feeling

Reachability

There are applications in which humans or human-controlled robots cannot reach; the best example is the research done on Mars or in very deep waters.

Can you think of other applications where you might need the help of a robot that can make decisions?

Is it physically different from previous robots?

Robots, like other machines, have different physical parts. The autonomous robot differs mainly from other robots in its controller algorithm and sensors interaction. In other words, any robot can be made an autonomous robot for certain tasks or tasks with the right sensors and the right algorithm (software).

Vision and sensors fusion

This topic, which talks about computer vision concepts and sensors fusion for robot application, we have five smaller lessons.

The general concepts are presented abstractly, and eventually, an application is shown as an example.

What is an image?

We can think of an image from a computer (robot) understanding as a function of space. The Independent variable is two-dimensional space: x-dimension and y-dimension. The units are pixels, while the dependent variable is the color intensity at each pixel or intersection of x and y.

That will make the image looks like a matrix. Images can be categorized into binary (black and white), greyscale images, and colored images. The higher the resolution of an image, the more pixels it has in the same space. The file size of an image can take a large amount of memory in case higher resolutions are used.

The values of pixels intensity in binary images are 0 or 1. In contrast, the values of greyscale images of pixels intensity vary from 0 to 255, and the colored images are multi-dimensional. Each colored image is three matrices of Red, Green, and Blue light intensities, hence, RGB levels.

The following figures show images and snap of pixels around them.

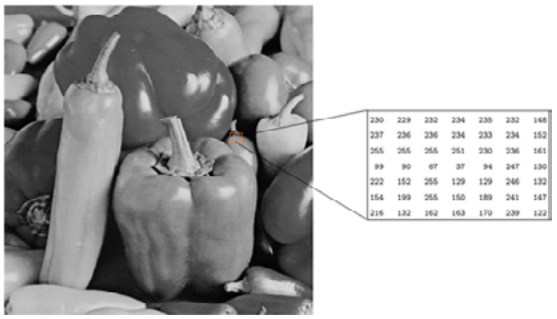


Image manipulation concepts

Suppose we understand that an image is a matrix of numbers. In that case, the manipulation of an image, can be thought of as manipulation of numbers, while keeping in mind the boundaries of those numbers. This is not an image processing course, so we will give some small tips on what can be done.

Flipping the zeroes to ones in binary images

For the same image shown, we can flip the zeros to ones and ones to zeros, and we see the difference in the shown image.

Background subtraction

In many cases, we can have a background image that is fixed, and when objects appear in the image, we can detect if an object is thereby subtracting the original value of the background. For example, we the given background, and the person or an object appears, and we remove the background, then we have a white spot, this tells us that something appeared in the frame (image).

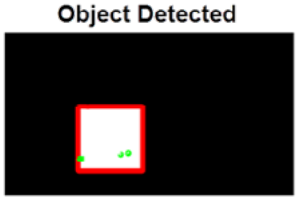


Image conversion

Images can be converted from colored to grayscale or binary, this is sometimes useful in saving space and speeding the processing.

Many other manipulation ideas exist in the literature, but this is out of the scope of this course.

Sensors fusion

Sensor fusion is the ability to use multiple sensors' data to better understand the environment. This is an advanced task and requires an understanding of environment models. But we felt it was good to mention it under the autonomous robot topic. The concept is important in autonomous cars, tracking, and other application.

In an autonomous car, the data from the radar, Lidars, and Cameras can be used to model or image the environment.

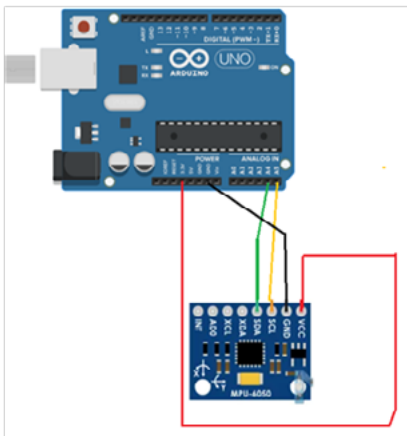
In a simple Arduino application, the data from an accelerometer and a gyroscope could be used to determine the exact location of the Arduino at a certain point. So an accelerometer measures the acceleration along a certain axis or direction, x, y, or z. The gyroscope measures the angular acceleration on one axis, i.e., rotation.

After this theoretical and long introduction, let us do an experiment.

An experiment:

The connection is shown below to test the MPU-6050 3 Axis Gyroscope and Accelerometer, and a testing code is provided under the graph.

The wire library, which allows or serial communication of Arduino with other devices, is called and used.



Real-time vs server-side processing

The Arduino platform is not an industrial-level platform or high-speed real-time microcontroller. So, we have two options for processing data collected by a microcontroller; real-time or batch processing.

The first type requires fast and quick calculation and response capabilities, while in batch processing, data can be collected and sent in batches to software for processing.

Some of the differences between the two approaches:

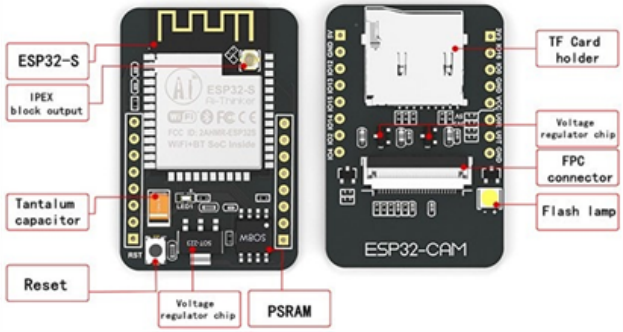
Real-time	Batch processing
External events are accepted and processed immediately	External events are collected and sent for processing
Time to complete the task is very critical	Completion time is not critical
Complex and costly processing requires unique hardware and software	It provides the most economical and simplest processing method

This is important to bring to the table, as many complex applications cannot be real-time implemented on Arduino due to speed and memory restrictions. So a batch processing should be considered, and that requires the Arduino to be connected to a PC or a server, where the processing is done there.

Application example

Streaming video to PC

The ESP32-CAM is a camera module based on an ESP32-S2 chip, a low-power microcontroller (It is a camera, with a microcontroller), or (microcontroller with a camera). It has Bluetooth capabilities, Wi-Fi, and a limited number of input and output pins.



Pulse-Width Modulation

Pulse-Width Modulation for motor speed control

The pulse width modulation is one of the interesting methods used to control the speed of DC motors and other applications where the control of the applied voltage is needed. Examples include control of LED brightness, loudspeakers voice level, and others.

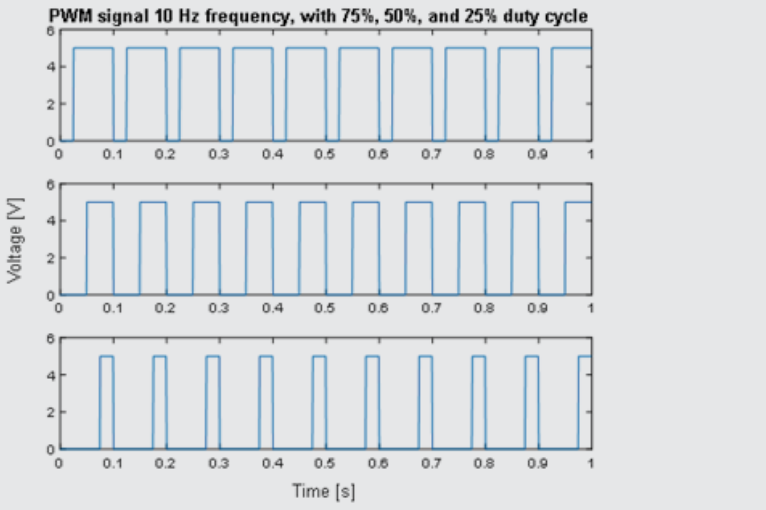
The basic idea of PWM is that the applied voltage is applied in pulses; these pulses are applied fast enough, such that the load's inertia is enough not to allow the load to change state until the next pulse is applied.

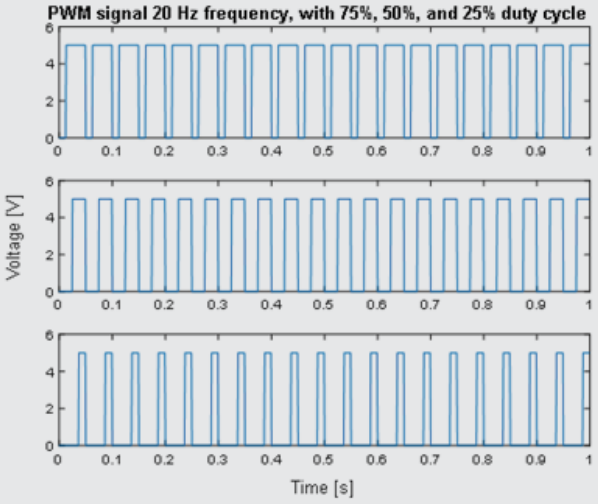
The lesson is supposed to show the students how to use PWM to control the dimming of LED, or RGB LED, and control a DC motor's speed.

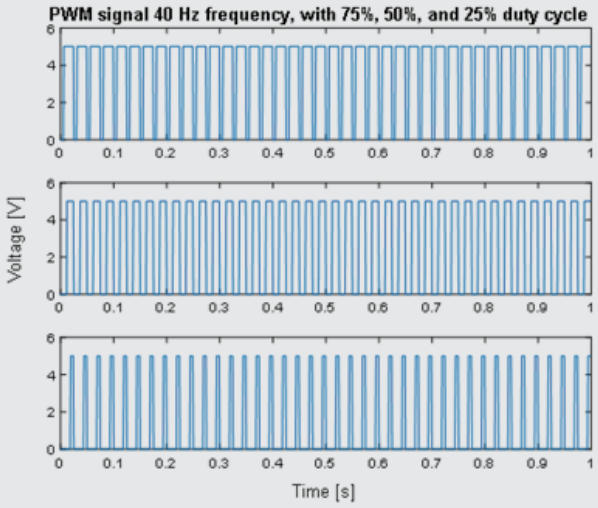
Frequency and duty cycle

The simple version of PWM for control applications is to switch a signal on/off at certain periods. The number of pulses made in a second is called the frequency, while the time for both parts of a pulse (on/off) parts, is called the period. The on-time of the period is referred to as the duty cycle.

The first figure shows a PWM signal of frequency 10 Hz, with duty cycles of 25%, 50%, and 75%. In the second figure, we see a PWM signal of frequency 20 Hz, with the same duty cycles signals in the third figure.

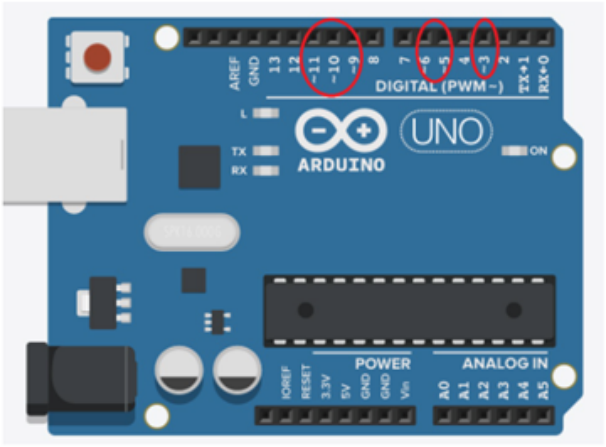






Default PWM using Arduino

Many Arduino boards have built-in PWM pins, which are indicated with the tilde (~) sign on the digital pins. As indicated in the Uno shown board in figure 4, those can be used as PWM pins.



The common frequency of the PWM signal in Arduino is 490 Hz, but each board can have some pins with higher frequencies. It's recommended to check the board specification sheet for that. In the case of the Uno board, in figure 4, pins 5 and 6 have a PWM frequency of 980 Hz, almost 1 kHz.

As for the duty cycle of these pulses, its determined by choosing a number from 0-255, so if we want a 50% duty cycle, we use round(0.5(255)) = 128.

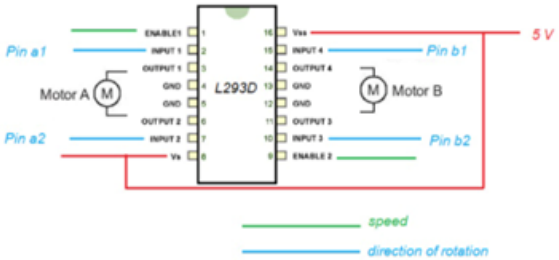
PWM using digital output

An interface integrated circuit motor driver is used to ensure that a DC motor is driven correctly. It provides protection, efficiency, and a built-in H-Bridge that s needed to control the direction of rotation of the motor. An example of this interface circuit is the L293D H-bridge motor driver shown in figure.

This is a 16 pin IC, where the eight pins on the side can control one motor, and the other pins on the other side control another motor. So, two DC motors simultaneously can be controlled. There are two inputs, two output pins, and an enable pin for each motor. The input pin will determine the rotation direction of the motor.

The enable pin determines if the motor will work or not. One way of controlling the speed of the motor is to use PWM with the enable pin. It will make the motor on/off within the determined duty cycle, which results in controlling the speed of the motor. (This may not be the best way to control the speed, but it is simple).

Finally, there are four ground pins, 2 for each motor, a power pin for the IC, and a power pin for the motors. Figure 6 shows the pin-out of the IC, and figure shows its connection to the actual motor set.



Move to a certain point and come back

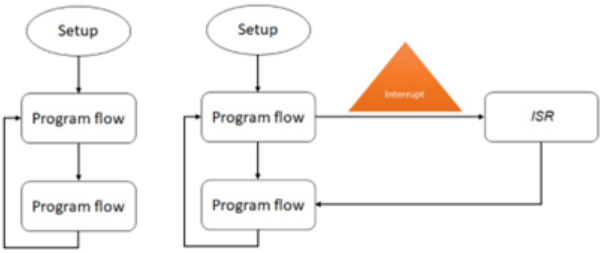
We discuss four practical aspects in this practical example of a robot going to a certain point and coming back. The practical aspects are approached generically; an example from a certain robot implementation is described after each generic topic.

Distance measuring using wheels rotations

In this lesson, we will do two experiments to measure the distance traveled. We will assume that the batteries are in good shape, and the robot will not work for a long time. Hence, the performance is going to be consistent within the work time.

The first thing we need to learn for this lesson is the concept of interrupts. An interrupt is a hardware or software activity that stops the regular flow and executes a certain routine before going back and proceeding with the program.

The first figure shows a regular Arduino code workflow, while the second figure shows the workflow with interrupts. Simply, the program flow stops when an interrupt is generated, and an interrupt service routine ISR is called and executed. The program goes back to workflow until another interrupt happens when it's done.



Arduino platform supports hardware interrupts only. They are divided into internal interrupts for the internal timers and external interrupts, resulting from interaction with other hardware devices connected to Arduino.

An interrupt happens when a pin state changes from 0 to 1 or 1 to 0. Different Arduino boards have different allocated pins for interrupts. The Uno we are using has pins 2 and 3 allocated for external hardware interrupts.

Interrupts have the internal labels INT0, INT1, … INT(n), and since we have only 2 interrupts on the Uno, we have INT0 and INT1.

The ISR is a code that deals with interrupts, it stops some functions from working, so it must be short and fast. Otherwise, the program will not work smoothly, and we might not catch all interruptions that occur while executing a long ISR.

To make the ISR run when interrupts happen, the attachInterrupt() command is used, and the format would be: attachInterrupt(digitalPinToInterrupt(pin), ISR, mode) (recommended) interrupt: the number of the interrupt. Allowed data types: int.

pin: the Arduino PIN.

ISR: the ISR to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.

mode: defines when the interrupt should be triggered.

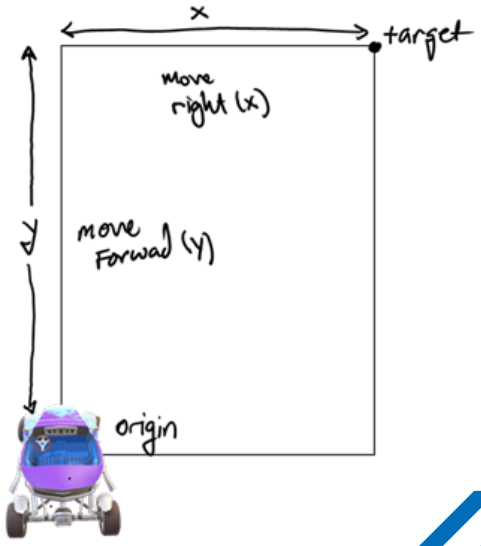
Four constants are predefined as valid values:

- **LOW** to trigger the interrupt whenever the pin is low,
- **CHANGE** to trigger the interrupt whenever the pin changes the value
- **RISING** to trigger when the pin goes from low to high,
- **FALLING** for when the pin goes from high to low.

Moving to a preset point

To move to a preset point, there are different methods, one of the simple and effective ways is to assume that the robot is in a relative origin point, and the preset point is relative to that origin point.

Let us say that we want to move to the point 50 cm in front and 75 cm to write, then y = 50 cm and x would be 75 cm.



So you can have functions move forward, move right, move left, move back, and each one with a certain distance. The disks can measure that distance on the wheel from part 1 or by using approximate time.
This is implemented in the section Return to start point.

Saving dynamical data

The Arduino is limited in its ability to save data for processing. It has a relatively small onboard memory for such activity. The speed processing is not fast enough to loop through large sets of data arrays.
So to have a practical sense of dynamical data collected by an Arduino, it should be sent to a server, which is the last topic of this class.
But for operating on a small scale, a pre-allocated vector of a certain number of values can be used to save a limited amount of data. Such as the directions of movements that a robot makes until it reaches its destination, then use those to get back home.
To experiment with arrays and vectors, we can use the public domain example of Arduino, where you can iterate over the number of pins to show the effect of a vector.

Return to start point

Since you know the steps for moving forward, right, left, and backward, those can be kept in an array to return from the target point to the origin.

Let us assume that we want to go to point 2.2 meters then, we go:

- moveForward(2)
- moveRight(2)
- getback
- moveLeft(2)
- moveBack(2)

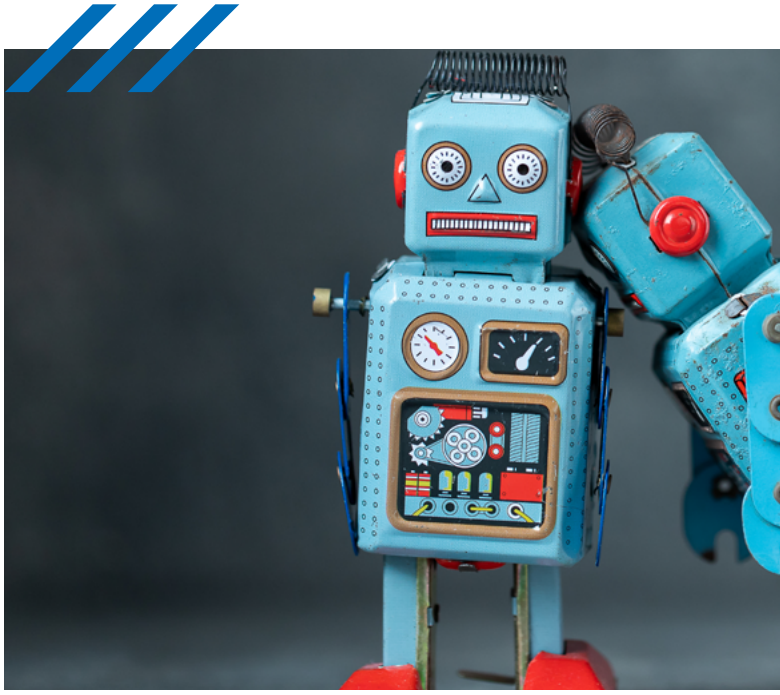
One of the problems you will face is the right orientation of the robot car. So we will leave that for you to experiment with and check how you can make the robot turn the right amount.

Transmit data to server about the way

Since Arduino is not good for saving large data and processing it, transmitting the data to a server makes and allowing access to that data feasible.
A straightforward way is to have an Arduino send the data through wifi to the server, which will use a POST method within a PHP file to send the data to an HTML file. This can be improved by sending the data to a database.
As an example, we could use the interrupt function from the previous lesson, and every time an interrupt happens, the Arduino sends a signal to the server. This would help map the environment. However, we will go with a simpler application for now, where the Arduino will send data to the server quickly.

The structure of the code is as follows:

1. Establish wi fi connection
2. Establish web server connection (could be the local host for testing)
3. Prepare the client (Arduino)
4. The setup function
5. The loop function



Map the environment

In this practical application, the robot will move to a predefined point in two different ways.

Collecting data

The last part of the previous lesson explained how to send Arduino data over wi-fi to an HTML page. In this part, we modify the previous version of the temperature sensing device. Instead of sending data to a file, it would send it to the database and retrieve it on demand.

A popular database that is free and easy to use is MySQL. Now an SQL database needs a server to host it, and the XAMPP server is one of the most popular ones used for that purpose.

So to prepare for data collection, we need to install XAMPP for both Apache serve and MySQL.

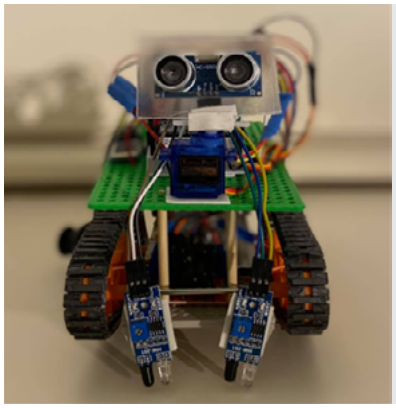
Next, we need to make a database for collecting the data sent by the Arduino, i.e., robot.

The only difference between the code in the previous lesson and this is the PHP file. In the previous lesson, the PHP file showed the data on an HTML file; now, we want it to send it to the database.

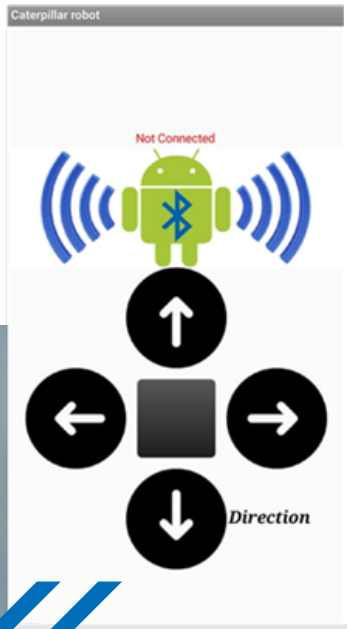
On command run

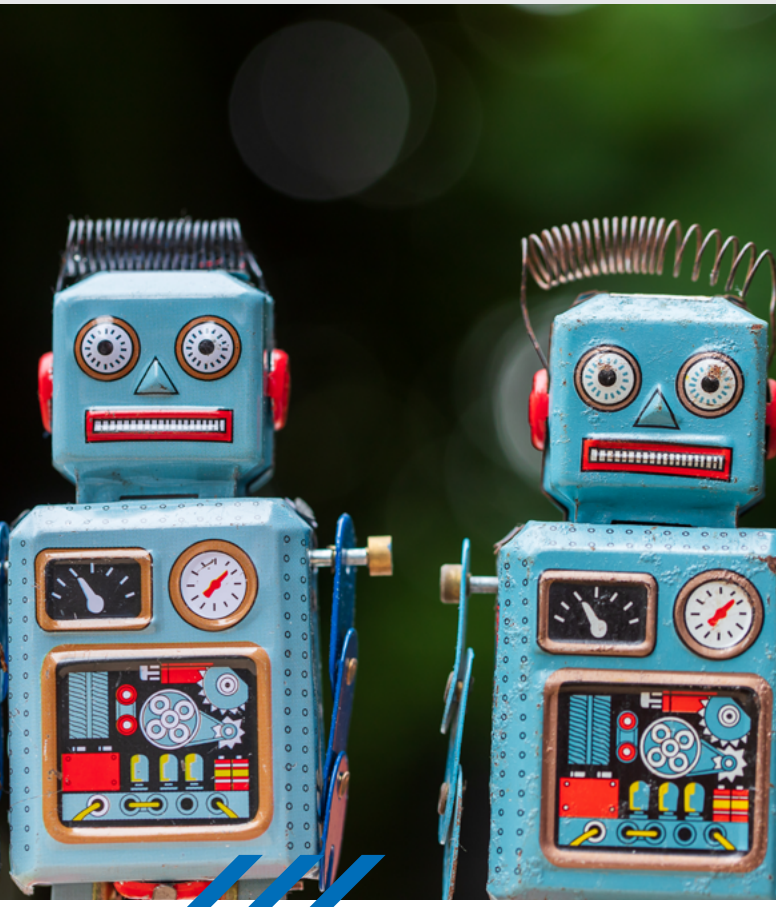
The move-on command is a standard joystick-like application; again, you can implement multiple operations modes on a single car.

This is done with the shown robot:



The on-demand part is done via an android app; the design is similar to the one from a previous class.





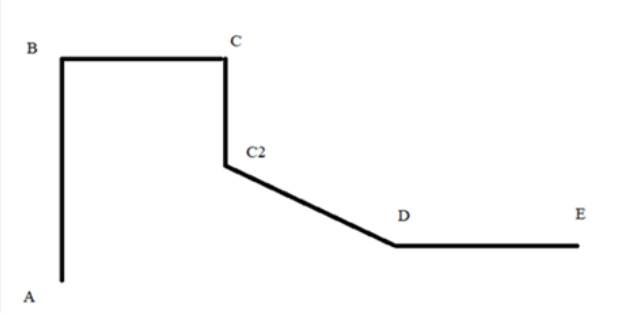
Predefined path

An advantage for the pre-defined path is in case we are not sure of the communication link (Bluetooth).

A pre-defined path can be implemented in different ways:

1. Use a communication link or an app to enter the number of turns, distance, and order
2. Implement the array of these steps before uploading the code to the robot

The first choice seems to be more appropriate.
In a flow chart scheme:



Assume a robot can move along the shown path from point A to point E through B, C, C2, and D.
Using the distance measuring technique from the previous class, we can use the functions:

- moveUp(distance)/ moveForward(distance)
- moveUp(time)/ moveForward(time)
- moveUp(distanceAB)
- turnRight(90 degrees)
- moveUp(distanceBC)
- turnRight(90 degrees)
- moveUp(distanceCC2)
- turnLeft(45 degrees)
- moveUp(distanceC2D)
- turnLeft(45 degrees)
- moveUp(distanceDE)

The speed of one motor can determine the robot's turning concerning the other. If one motor is held still and the other is turning, the robot will turn around the axis of the hold motor.
The angles can be determined experimentally or by some complicated mathematics.

Expect problems

The distance, or the time the robot has traveled, can be recorded and sent to a file or database. This would allow the mapping of the environment which the robot has visited.
This could be combined with images or video streams, giving a better perception of the surroundings.
So, to simplify the problem, one approach would be that before every turn, the robot would report the previous distance and send an image of the spot it stands at.
Due to limitations, time constraints, and implementation of this idea, it would be uploaded to the platform in the future.

Perform a task

In this practical part of the implementation, we want the robot to go to a certain point, count how many red balls are in a box, use the gripper to catch a black ball, then come back and inform them at the starting point on the number of red balls by the buzzer.

Choose a task

An Arduino robot involving image processing is not easy to do because of the limitations we mentioned before (Speed and memory).

But still, we can try to implement two different tasks by combining all the experiments we performed before.

We start with a simple gripper application and straight-line target.

The robot should move from point A to point B. Use an IR sensor for color detection and move a gripper to catch a target.

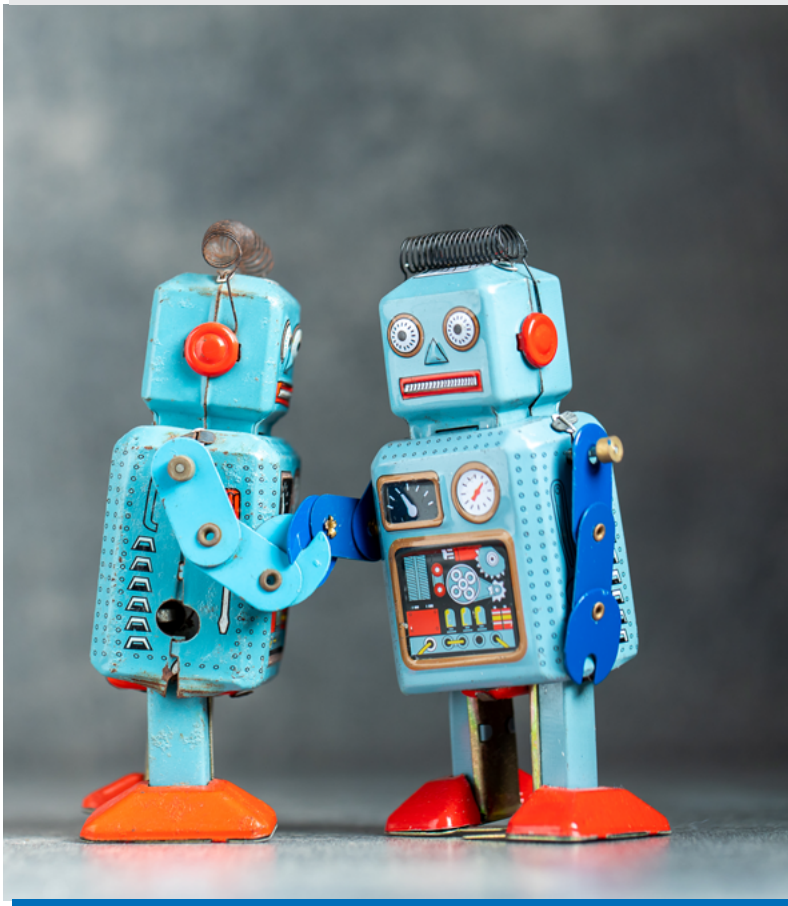
This can be modified by having red balls in a pot, moving the robot from point A to point B where the pot is, using simple image manipulation from lesson one, subtracting the background, and checking if there is a red color in the pot. Then the robot comes back and signals that there is a red color in the pot. Both tasks require involved thinking of the problem but can be done.

Expect problems

The Camera module has a small number of free GPIOs; they can and should be used for making decisions on the main Arduino board. However, the server-side processing of camera images might be a demanding task. So this is something to be aware of.

Choosing the right target for the gripper is important in sensing and holding on to it. Calibration of the IR or sonar sensor should be done and help make the task feasible. But it requires some work.

Other platforms are more suitable for working with images, like the Raspberry pi. But that is out of the scope of this lesson.



Notes

Credits:

Univerza v Ljubljani
(Slovenia)

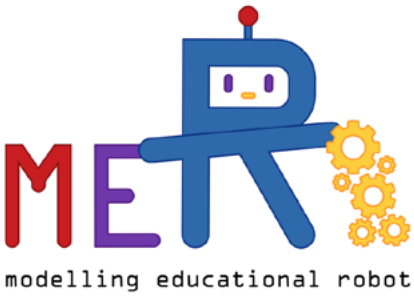
AIJU Institute Ibi
(Spain)

Šolski center Novo mesto
(Slovenia)

Muratpasa Mesleki ve Teknik Antalya
(Turkey)

Internacionalni Univerzitet u Sarajevu
(Bosnia and Herzegovina)

Polotecnico Fermi Gada Napoli
(Italy)



Co-funded by the
Erasmus+ Programme
of the European Union

WWW.MERPROJECT.NET