



STEM

FOR
YOUTH

ENJOY. SCIENCE TECHNOLOGY ENGINEERING MATHEMATICS.

OBSTACLE AVOIDING ARDUINO ROBOT

BUILD AND PROGRAM A ROBOTIC VEHICLE
THAT AVOIDS OBSTACLES

ROBOTICS

ENGINEERING FOR SECONDARY
SCHOOL STUDENTS



PROJECT DETAILS

PROJECT ACRONYM	STEM4YOU(th)
PROJECT TITLE	Promotion of STEM education by key scientific challenges and their impact on our life and career perspectives
GRANT AGREEMENT	710577
START DATE	1 May 2016
THEME	SWAFS / H2020

DELIVERABLE DETAILS

WORK PACKAGE NO. AND TITLE	WP5 – CONTENT CREATION, TOOLS AND LEARNING METHODOLOGY DEVELOPMENT
DELIVERABLE NO. and TITLE	D5.1 MULTIDISCIPLINARY COURSE-ENGINEERING SUB-COURSE
NATURE OF DELIVERABLE AS PER DOW	R=Report
DISSEMINATION LEVEL AS PER DOW	PU=Public
VERSION	FINAL
DATE	JULY 2018
AUTHORS	EUGENIDES FOUNDATION



INDEX

INDRODUCTION.....	4
Activity 0-What is engineering?	5
Activity 1-Identifying the problem (what is the engineering problem?).....	14
Activity 2 - Divide into sub-problems	16
Activity 3-Explore the science	17
Activity 4 – Solve sub-problems	28
Activity 5 – Combine sub-solutions, test and improve.....	47
Activity 6 – Present Final Solution.....	50
Science Carriers and Your Future	51
List of Materials	52
References.....	53



INDRODUCTION

Nature is the source of inspiration for the field of robotics and engineers are trying to mimic its ways in many cases. In this challenge, students construct and program a tricycle robot to navigate in space avoiding obstacles that appear along its course. The robot mimics the way that the animals such as bats (or dolphins etc.) "see" in dark natural environments. The robot is built on the open arduino platform and includes the appropriate distance sensors (ultrasonic sensors) in order to "see" the obstacles ... An impressive conclusion that children will make is that the robot "sees" ... by listening.

The field of Robotics integrates all STEM fields in a way no other subject can cover. In fact, the field of robotics integrates mechanical, electrical, electronics, control engineering, computer science, technology, mathematics, physics and biology.

The challenge of designing and programming a robot that avoids obstacles can be implemented from schools to science museum and science fair workshops. The primary aim of this activity is to motivate students and young people to become interested in science and engineering.

In general, robotics can trigger and develop young people's curiosity while making the process of learning about STEM fields much more attractive.

Overview of the challenge:

<u>Participant age:</u> 14-18	<u>Number of participants:</u> Groups (3-4 students)	<u>Module length:</u> App. 1.5 hours to 4 hours
<u>Level of knowledge:</u> intermediate, advanced	<u>No. and type of personnel:</u> teacher / external science experts/science museum / staff/students	<u>Type of venue:</u> Classroom / outdoors/science museum
<u>Technological needs:</u> internet / computer / tablet /	<u>Topic as per formal curricula:</u> Programming, sensors, navigation	<u>Estimated cost:</u> low / intermediate / high (specify) Low (150 € per 5 teams) All the materials are reusable.
<u>Specify learning methodology (D3.1):</u> Engineering Design Process (EDP) Inquiry Based Learning (IBSE)	<u>Engineering Field:</u> electrical, mechanical, computer, electronic, robotics	<u>Type of activity:</u> Hands on activity

General Objectives: In this hands on activity students will

- understand the principal role of the materials and their properties in engineering solutions
- get interested in phenomena found in daily life
- develop the ability to predict and verify results
- realize the difference between natural and man-made objects
- conceive that goals are achieved by collaboration among scientists and engineers
- experience the importance of teamwork as well as individual responsibility as a member of the team
- experience the satisfaction of success
- discover and experience the relationship between theory and practice
- develop a spirit of inquiry
- develop the ability to accomplish a task from start to finish
- develop design skills
- develop the ability to turn designs into reality
- acquire technical skills on using tools properly and safely
- get familiar with the process of finding means to overcome difficulties and problems

Activity 0-What is engineering?

Duration: 40 minutes (max)

Objectives: In this activity students will

- discover the differences between engineering and technology
- associate things, activities or other terms with engineering and technology
- familiarize with different engineering fields
- apply the Engineering Design Process in order to design and construct a paper table

General Context

This first activity is intended to encourage thinking about what engineering and technology are and to challenge the misconceptions that students may have about the field of engineering or the work of an engineer. This activity aims to disentangle the concepts of engineering and technology and develop the understanding that manmade objects are designed for a purpose and that technology, in a very broad sense, refers to any object, system or process that has been designed, constructed, modified in order to solve a problem or to meet a



certain need. Finally, in this first activity, students are introduced to the process that engineers follow in order to find solutions to the problems they are dealing with. Student teams try to find and construct a solution to a simple problem following the same process as engineers do.

❖ **Small groups**

Teacher arranges students into of 3-4 person groups, preferable mixed gender and aptitude (teams should be kept the same through the entire challenge). Each group is asked to discuss and interpret the concepts of engineering and technology and try to associate things, activities or other terms with these concepts. Students are asked to answer to the following questions and write their answers down:

- i. What is engineering?
- ii. What is the work of an engineer?
- iii. Can you give some every day examples of engineering and technology?
- iv. What is the difference between engineering and technology?

After that, the teacher writes student team's answers on the board and initiates a discussion about engineering and technology. He/she seizes the opportunity to introduce the Engineering Design Process (EDP) steps and initiate a quick discussion about each individual step. Finally, the teacher asks student teams to construct a laptop table out of paper, by applying the EDP.

What is engineering?

The word engineering is of Latin origin; its root is "ingeniere" which means to design or to devise.

Engineering is the application of scientific knowledge (natural sciences, mathematics, economic and social), practical knowledge and empirical evidence in order to solve everyday life problems. More specific, the purpose of engineering is to invent, innovate, design, build, research and improve structures, machines, tools, systems, components, materials, processes and organizations under specific constraints. The field of engineering is very broad and encompasses a great range of more specialized fields [1], [2] such as:

- Agricultural Engineering
- Architectural Engineering
- Biochemical Engineering
- Biological Engineering
- Biomedical Engineering
- Chemical Engineering
- Civil Engineering
- Computer Engineering



- Electrical Engineering
- Environmental Engineering
- Geoscience Engineering
- Industrial Engineering
- Marine Engineering
- Materials Engineering
- Mechanical Engineering
- Metallurgical Engineering
- Ocean Engineering
- Petroleum Engineering

What is the work of an engineer?

Engineers identify a problem, and come up with a solution – often creating something completely new in the process.

“Scientists investigate that which already is; engineers create that which has never been.” (Albert Einstein)

The most famous engineering fields in more detail [1], [2], are the following:

Aerospace engineering: the field of engineering concerned with the development of aircraft and spacecraft. Aerospace engineers design, develop, test, and supervise the construction of aerospace vehicle systems. Such systems are aircrafts, helicopters, space vehicles and launching systems.

Architectural engineering: the field of engineering that uses engineering principles to the construction, planning and designing of buildings and other structures. Architectural engineers work in several areas such as: the structural integrity of buildings, the design and analysis of light, heating and ventilation of buildings, energy conservation issues.

Biological engineering (bio-engineering): the field that applies concepts and methods of biology, physics, chemistry, mathematics and computer science to solve problems which are related to life sciences. Bioengineers solve problems in biology and medicine by applying principles of physical sciences and engineering while applying biological principles to create devices such as diagnostic equipment, biocompatible materials, medical devices etc. In general, bioengineers try to mimic biological systems in order to create products or modify and control biological systems.

Chemical engineering: the field of engineering that applies physics, chemistry, microbiology and biochemistry along with applied mathematics and economy in order to transform, transport and use chemicals, materials and energy. Traditionally chemical engineering was linked to fuel combustion and energy systems, but today’s chemical engineers work in medicine, biotechnology, microelectronics, advanced materials, energy and nanotechnology.



Civil engineering: the engineering field that deals with design, construction and maintenance of constructions such as roads, bridges, dams, buildings, tunnels. Civil engineering is probably the oldest engineering discipline which deals with the built environment. Civil engineers use their knowledge on physics and mathematics to solve society problems.

Computer engineering: the discipline that integrates electrical and electronic engineering and computer science in order to design and develop hardware, software, computer systems and other technological devices. Computer engineers embed computers in other machines and systems, build networks to transfer data and develop ways to make computers faster and smaller. Furthermore, computer engineers have expertise in a variety of areas such as software design and coding and are trained to design software and perform and integrate that software with hardware components.

Electrical engineering: the field of engineering that deals with the study and application of electricity, electronics and electromagnetism. Electrical engineers conceive, design and develop circuits, devices, algorithms, systems and components that can be used to sense, analyze and communicate data. Electrical engineers work on a variety of projects, such as computers, robots, cell phones, radars, navigation systems and all other kinds of electrical systems.

Materials engineering: the field that involves the discovery and design of new materials. Material engineering incorporates physics, chemistry, mathematics and engineering. Materials engineers develop, process and test materials used to create a wide range of products such as computer chips, medical devices, aircraft components etc. Materials engineers are concerned with the structure and properties of materials used in modern technology so they study the properties and structures of metals, ceramics, plastics, nanomaterials and other substances in order to create new materials that meet certain mechanical, electrical or chemical needs.

Mechanical engineering: the engineering discipline which applies the principles of engineering, physics and mathematics for designing analyzing manufacturing and maintaining mechanical systems. Mechanical engineers create machines used in manufacturing, mechanical components of electronics, engines and power-generating equipment, vehicles and their components, artificial components for the human body, and many other products.

Ocean (Marine) engineering: the branch of engineering study that deals with the design and operations of manmade systems in the ocean and other marine environments. Ocean engineering includes the engineering of boats, ships, oil rigs and any other marine vessel or structure. Ocean engineers apply their engineering (mechanical, electrical, electronic engineering) and scientific knowledge in order to design and develop systems and structures in marine environments. An ideal ocean engineer has to achieve a proper tandem between the marine eco-system and the developed human world.

Robotics: the interdisciplinary branch of engineering and science that deals with designing, constructing, programming, controlling, operating and using robots. Robots are used in a wide range of applications which include industrial, military, agricultural, medicine robots etc.

- Industrial robots - take over work that is dirty, dangerous and degrading to the human spirit (e.g. arc welding, grinding, sanding, polishing and buffing, palletizing etc). Typically, these robots are articulated arms particularly created for applications like- material handling, painting, welding and others.
 - Medical robots – robots that are employed in medicine and medicinal institutes such as surgical robots, rehabilitation robots and biorobots.
 - Domestic or household robots – These types of robots are used at home and consist of robotic pool cleaners, robotic sweepers or robotic vacuum cleaners.
 - Military robots – These types of robots are used for offensive or defensive purposes and include bomb discarding robots, ballistic shield robots, inspection robots, attacking drones etc.
 - Space robots - Robotic devices used to aid, augment, or substitute astronauts in order to do difficult or rote tasks such as exploration or repairs in dangerous environments (e.g. space station robotic arms, Mars rovers Spirit and Opportunity).
 - Deep Sea robots – Robots that have long-term presence in the deep ocean and carry equipment to measure various parameters that scientists are interested in (e.g. Benthic Rover).
- Engineering Misconceptions
- Plumber
 - Electrician
 - Carpenter
 - Auto Mechanic
 - PC (Personal Computer) Technician
 - Welder
 - Machinist

What is technology?

Engineering and technology are intertwined terms in society. In order to disentangle the two terms, one needs to understand what their meaning is. Engineering is both a field of study as well as application of scientific knowledge to create or produce something. On the other hand, technology is the collection of techniques, skills, methods and processes used in the production of goods or services or in the accomplishment of objectives, such as scientific investigation. Technology can be the knowledge of techniques and processes, or

it can be embedded in machines, computers, devices, and factories, which can be operated by individuals without detailed knowledge of the workings of such things.

Engineering Design Process

The teacher introduces the EDP steps to the students. A short description of the Engineering Design Process follows.

The Engineering Design Process (EDP) is a series of steps that engineers follow when they are trying to solve a problem they are facing and consists of a methodical approach. However, there is no single design process which is universally accepted. In general, each individual design process begins with identifying the problem and its requirements and ends up with a proposed solution. The intermediate steps, however, can vary. It is very important to point out that EDP is not a linear process. Since, engineering problems can have numerous correct answers; the process may require backtracking and iteration. The solution to an engineering problem is usually subject to unexpected complications and changes as it develops. In this project we propose a series of steps which are described below.

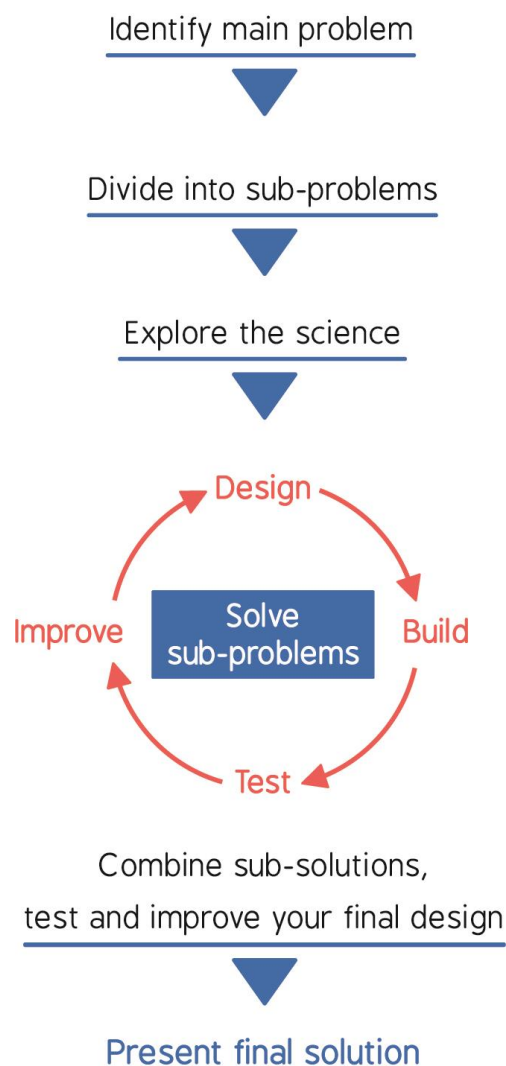


Figure 1: EDP steps

1. Identify the problem

Engineers ask critical questions about the problem and what they want to create, whether this is a space station, a skyscraper, a car or a computer. These questions include:

- *What is the problem?*
- *Define the problem in specific terms. Be as specific as possible.*
- *Which are the available materials?*
- *What do we need to know in terms of scientific principles that underlie the problem?*
- *What are the constraints of the problem? (budget, time etc)*
- *Which are the criteria that must be met so that the solution is acceptable?*

2. Divide problem into sub-problems

Usually big problems consist of a series of sub-problems. So, engineers analyze the problem in order to plan their work.

- *Is the solution to the main problem straight forward?*
- *Does the main problem consist of smaller and simpler problems?*
- *Engineers do not attempt to plan the whole thing at once. Large projects have many variables that you do not know and can affect the whole plan.*
- *Engineers set smaller goals. Instead of trying to plan everything from the beginning, they figure out the first obvious step and then move to the next one.*

3. Explore the science

After dividing the main problem to the sub-problems it consists of, engineers investigate the scientific principles that underlie each sub-problem. The fundamental background science is essential for solving sub-problems and designing the optimum solution.

- *What areas of science cover my project?*
- *Which are the scientific principles that underlie each sub-problem?*
- *Research background theory*
- *Perform experiments-tests to understand the theory's applications.*

4. Solve sub-problems

Generate as many solutions as possible by brainstorming and examine the advantages and the disadvantages of each possible solution. Evaluate all the solutions in order to identify the optimum.

- *Design: Design the application of the chosen solution, carefully and with as much detail as possible. Draw a diagram of the solution and make a list of materials you need.*
- *Build: Follow your design and develop your solution of each one of the sub-problems*
- *Test: Test whether the solutions of individual sub-problems are compatible with each other*
- *Improve: Make the necessary corrections and improvements*

5. Combine sub-solutions, test and improve

Combine the different components that will provide you the final, integrated solution to the main problem.

Test and if necessary improve your final design

- *Does it work?*
- *Does it solve the need?*
- *Does the final design meet the criteria set?*
- *Analyze and talk about what works, what doesn't and what could be improved.*
- *Discuss how you can improve your solution*

6. Present final solution

Review and evaluate your work and present your final solution in front of an audience.

Preparatory activity - Strong Paper Table

This activity is designed as a way to introduce students to the EDP in order to have a common understanding of how it works and help teachers who are not familiar with engineering and technology in their classrooms.

Can you build a newspaper table that won't collapse under the weight of a laptop?

Student teams are asked to follow the design process to build a sturdy and steady laptop table out of paper. Find a way to make paper support weight and prevent the legs of the table from buckling (see Fig. 2 for possible solutions).

Criteria

- The table must withstand a weight of 2-3 kg.
- The table must be sturdy and stable.
- The table's surface must be inclined to make the use of keyboard easier.
- The table's surface must be ventilated, to prevent laptop from overheating.

Constraints

- The available materials are 5 newspapers and 50 A4 sheets of paper.
- The available tools are duct tape and a pair of scissors.
- The available time is 30 minutes

-Tip: From the criteria the main problem can be divided into sub-problems

- *Stability and durability of the table*
- *Inclination*
- *Ventilation*





Figure 2: Possible Solutions

Activity 1-Identifying the problem (what is the engineering problem?)

Duration: 20 minutes

Objectives: In this activity students will

- familiarize with materials and tools such as pliers, screw drivers, screws, soldering iron etc.
- reflect the role of materials in designing a solution to their problem

General Context

In this activity the teacher sets the engineering problem that students have to face. Student teams ask questions concerning the problem they are facing and discuss with their teacher the criteria that their solution must meet as well as the constraints they have. Afterwards, each team prepares a problem statement i.e. a brief description of the issues that need to be addressed by a problem solving team and should be presented to them (or created by them) before they try to solve a problem. Finally, they discuss with their teacher about materials and components that could be appropriate for their challenge. The robotic kit and available tools are provided to the teams. Each team spends some time to become familiar with the different kit components and tools.

❖ Working in groups

The teacher briefly introduces the Engineering Challenge: “Each team has to build a tricycle robotic vehicle and program it to avoid obstacles while moving through space”. The teacher then states that engineers who face and deal with problems such as the one under study are called Robotic Engineers.

The teams are encouraged to ask questions concerning the problem. The basic questions that should be investigated are:

- What is the problem or need?
- Which are the criteria that their solution must meet?
- Which are the constraints of the problem?
- Which are the available materials, tools, resources, technologies?
- Which are the scientific-engineering principles behind the problem?

Each team is asked to prepare a problem statement. A good problem statement should answer these questions:

1. What is the problem? This should explain why the team is needed.
2. Who has the problem or who is the client/customer? This should explain who needs the solution and who will decide the problem has been solved.
3. What form can the resolution be? What is the scope and limitations (in time, money, resources, technologies) that can be used to solve the problem?

The problem must be specific enough to allow each team to design a solution.

Constraints

- Available materials, technologies
- Available tools
- Available time
- The robot's size
- Cost
- Security Issues

Criteria

- The vehicle must be able to move forwards-backwards and left-right
- The vehicle must be able to rotate around its own axis (perform axial turn)
- The vehicle must be able to identify obstacles from a distance.
- The vehicle must be able to avoid obstacles while navigating through a restricted area

After defining the problem and setting constraints and criteria, provide student teams with the robotic vehicle kit and tools. Give them some time to become familiar with all different tools and kit's components. Ask them to propose possible uses for both tools and kit's components.

In this activity student teams get in touch with the first step of EDP and are gradually introduced to the process of investigating the basic principles of robotic engineering, which follow next.



Activity 2 - Divide into sub-problems

Duration: 15 minutes

Objectives: In this activity students will

- break the main problem to simpler problems
- organize their goals
- schedule their work and set time limits
- draft a plan how they will work

General Context

In this activity, student teams move to the second step of the Engineering Design Process which is to divide the main problem into sub-problems. Student teams try to analyze and divide the bigger problem to smaller and easier to handle sub-problems. Student teams write down and justify their thoughts.

❖ Working in groups

Teacher initiates a discussion about the fact that an easy way to deal with a large project is to break it into smaller tasks which are more manageable and easier to face. However, he/she should point out that the task of getting a large goal divided into smaller and achievable ones is not very easy and in fact it can be something quite hard to do. The teacher can propose some simple guidelines that if followed can make the process of breaking the problem, easier. After that student teams should be prompted to propose possible sub-problems.

Guidelines

- Do not attempt to plan the whole thing at once. Large projects have many variables that you do not know and can affect the whole plan.
- Set smaller goals. Instead of trying to plan everything from the beginning, figure out the first obvious step and then move to the next one.
- Do not hesitate to re-divide. If you procrastinate on any of the smaller task, do not hesitate to analyze to simpler ones.
- Set time limits. Usually, when engineers deal with a complex problem, apart from the problem itself they have to face time limitations. So in order to be effective manage your time as good as possible.

The main problem can be divided into four sub-problems:

1. Controlling the motors
2. Programming the motors
3. Identify Obstacles (how sensors work)
4. Avoid Obstacles
5. Combining all the above



Activity 3-Explore the science

Duration: 90 minutes

Objectives: In this activity students will

- explore how motors work and how a tricycle moves
- organize and classify their observations
- predict and verify results
- familiarize with the third step of the Engineering Design Process

General Context

The purpose of this activity is to introduce students to **the process of exploring the scientific-engineering principles that underlie the problem**. Student teams experiment with a simple motor they construct on their own and with the actual motors they are going to use on the final design. Furthermore, they experiment with the motion of a tricycle and investigate how such a vehicle moves through space. Student teams propose investigative questions concerning the principles that underlie the function of motors and the motion of a tricycle through space. They experiment by constricting their own motor, by using readymade motors and by manipulating the motors' spin direction in order to make a tricycle perform certain moves.

Student teams are guided through the process of acquiring the necessary knowledge they need for solving the problem. In addition, students organize their observations/answers.

❖ Working in groups

The teacher's goal is to introduce students to the third step (Explore the science) of the EDP and to motivate them start thinking about the knowledge they need to have and to start brainstorming on how this knowledge can be applied by imagining possible solutions to the engineering problem.

Student teams are encouraged to brainstorm and pose questions concerning a robot's architecture, the function of motors and the principles that underlie a vehicle's land navigation.

The key questions, which are important to investigate and are the focus of this activity:

- Which are the main components of a robot?
- How is a robotic vehicle set to motion?
- How does a tricycle robotic vehicle move forward and backward?
- How does a tricycle robotic vehicle turn right, left?
- How a tricycle robotic vehicle performs axial and radial turns?

Discuss with the teams the main components of a robot, the function of each component and purpose it serves (as described below).

Provide each student team with the necessary materials for constructing a simple motor. Have them construct the motor and discuss the physics that underlies the function of motors (Law of Induction).

Provide each team with the robotic kit. Have them construct the vehicle and experiment on the vehicle navigation by using different combinations among wires.

➤ Robot Parts

1. Frame (chassis): The frame supports all the components that make up the robot. The chassis plates have a variety of holes for attaching sensors, controllers, power etc. The chassis also provides protection for some internal parts of the robot.
2. Locomotion System: This system defines how the robot moves (translator motion, rotary motion etc). This system enables the robot move forward, backward, left, right, up, down etc. For this task the robot needs electrical energy which is converted to mechanical energy by certain devices called actuators. The most common actuator is the DC motor.
3. Actuator System: Actuators are components or machines that are responsible for moving or controlling mechanisms or systems. An actuator requires a control signal and a source of energy. There are several types of actuators such as hydraulic (cylinder or fluid motor that uses hydraulic power to facilitate mechanical operation), pneumatic (converts energy formed by vacuum or compressed air at high pressure into either linear or rotary motion), electric (powered by a motor that converts electrical energy into mechanical torque), mechanical (execute movement by converting one kind of motion, such as rotary motion, into another kind, such as linear motion). Inside the main bodies of robots are small motors called actuators which move parts of the robot's body.
4. End-Effectors: In order to interact with their environment and carry out tasks, robots are equipped with tools called end-effectors. These tools vary according to the tasks that the robot is designed to accomplish. Effectors enable robots to take action, to do physical things. Effectors use underlying mechanisms, such as motors which are called actuators and which do the actual work for the robot. For example, robotic factory workers have interchangeable tools such as paint sprayers or welding torches. Mobile robots such as the probes sent to other planets or bomb disposal robots often have universal grippers that mimic the function of the human hand. (<http://sciencing.com/main-parts-robot-7403157.html>)
5. Sensor System: Sensors are the physical devices that enable the robot to perceive its physical environment in order to get information about itself and its surroundings. In general, sensors are electronic components whose

purpose are to detect events or changes in their environment and send the information to other electronic systems such as a computer processor. There are several types of sensors which can measure physical parameters like temperature, heat, radio waves, infrared waves, light, ultrasounds, speed, acceleration etc.

6. Power Supply System: In order to be functional a robot needs power. Most robots get their energy form electricity. Stationary robotic arms, like the ones used in industry are plugged in like other devices. On the other hand mobile robots are usually powered by batteries.
7. Microcontroller¹: A microcontroller is a computing device that can execute a program (i.e. a sequence of instructions) and is referred to as the “brain” of a robot as it is responsible for all computations, decision making and communications. Each microcontroller possesses a series of pins (electrical signal connections) that can be turned on or off, via which it interacts with the outside world. These pins are controlled through programming instructions. Microcontrollers can be used to control other electrical devices such as actuators, storage devices, WiFi or Bluetooth interfaces. For instance TV sets, washing machines, remote controls, telephones, watches, microwave ovens, and now robots require these little devices to operate. Unlike microprocessors (e.g. the CPU in personal computers), a microcontroller does not require peripherals such as external RAM or external storage devices to operate. This means that although microcontrollers can be less powerful than their PC counterpart, developing circuits and products based on microcontrollers is much simpler and less expensive since very few additional hardware components are required.

➤ Make a simple electric motor

Student teams are asked to construct a simple electric motor using simple materials.

- 1.5 meters of magnet wire (24 or 25 gauge, Radio Shack #278-1345)
- 2 ring magnets
- 2 safety pins
- 1 D battery (Do not use any battery rated above 1.5 volts, it will result to the overheating of coil)
- plasticine
- small piece of sandpaper
- electric tape or rubber band

Wind the magnet wire around the battery to form a ring. The coil should have 10 – 15 turns. Too many turns will make the coil too heavy and too few

¹ <http://www.robotshop.com/blog/en/how-to-make-a-robot-lesson-4-understanding-microcontrollers-2-3700>

turns will deteriorate motor functioning. Leave 6 cm of each end extended. Carefully slide the coil of wire off of the battery. Wind the two ends around the coil three times, they will hold it together (make knots like the ones shown in Fig. 3). You should have 2 cm of straight wire sticking out each side of the coil. Hold the loop vertically by placing your thumb through the center of the rotor. Place one of the straight sides of wire in a flat surface and by using a blade strip ONLY the TOP surface of the wire (remove the insulation only from the top surface). Strip the other section of the straight wire completely (see Fig. 4). Make sure the two wires extend from the coil opposite the centre.

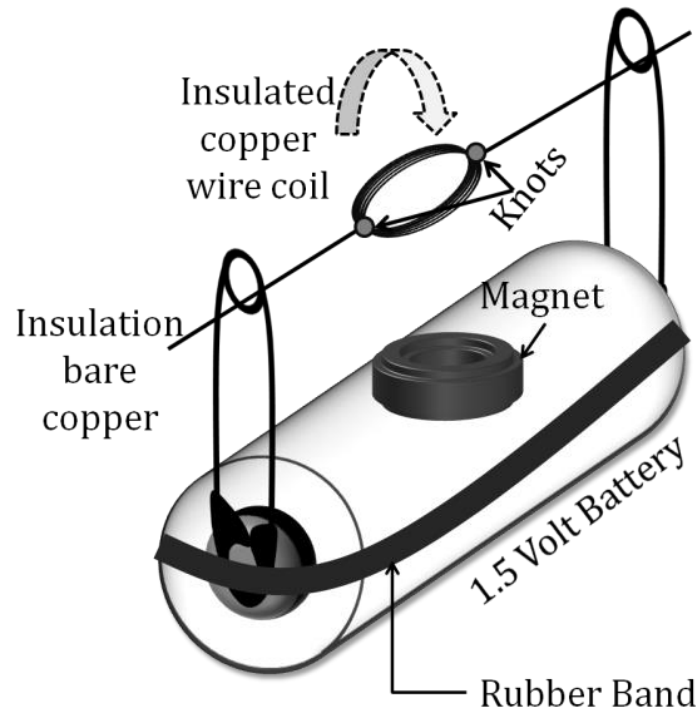


Figure 3: Illustration of a simple motor made from simple materials

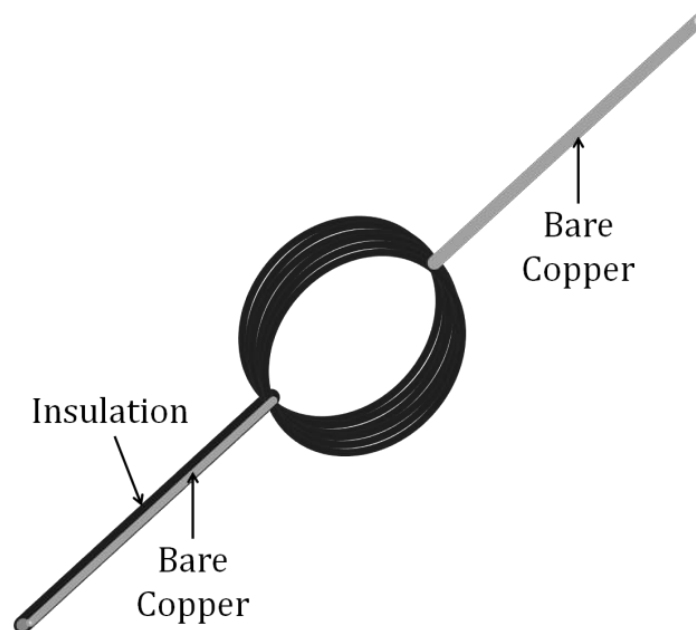


Figure 4: Close up shot of copper wire coil. Pay attention to the straight wire. The one is completely stripped. The other has only its upper part stripped.

The purpose of this activity is to demonstrate how an electric motor works. Though simple, this activity demonstrates how motors convert electrical energy (from a battery or voltage source) into mechanical energy (used to cause rotation).

Explanation [2]

When a wire that carries current is placed in the region of a magnetic field, the wire experiences force. The ring magnet provides the magnetic field. The magnetic field lines of a ring magnet are shown in Figs 5, 6 (out or into the magnet depending on which side of the magnet is exposed). When the coil stands in the safety pins, so that the plane of the coil has vertical orientation, the top and bottom sections of the coil behave as current carrying wires in the region of a magnetic field (we are interested only in the sections of the loop which are perpendicular to the magnetic field lines, because only these experience force).

The direction of the force on a current carrying wire in a magnetic field, and as consequence the direction that the motor turns, is determined by the Right Hand Rule (see Figure 7).

Since the loop experience two forces of different directions with the first one acting on one side of the loop and the other on the opposite side of the loop, the loop experiences a torque and it rotates. The greater the number of loops the greater the torque that the coil experiences.

If the system is left on its own, the rotor would never make a single complete rotation. In fact, the rotor will oscillate back and forth. Firstly, the rotor will turn 180 degrees one way and then 180 degrees the other way and it will never make a full rotation. The reason for this effect is that after the motor has made a 180 degrees turn, the current has changed direction (see Figure 7) and as a result the top of the rotor experiences a force pointing into the plane of the paper while the bottom of the rotor experiences an opposite force.

In order to overcome this problem a simple technique is used. On one of the straight sections of the coil we have removed the insulation. The circuit is complete only when the stripped wire touches the safety pin.

- Initially the rotor is given a small push so that the stripped section of the straight wire comes in touch with the safety pin.
- The circuit is then complete, the current flows and the rotor experiences a torque.
- The rotor makes half a turn (180 degrees) and the circuit is broken as the insulated part of the straight wire touches the safety pin.
- No current flows and as a result there are no opposing forces acting on the rotor and the rotor does not experience a torque on the opposite direction than before.
- The inertia of the rotor carries the rotor until it completes a full turn.
- Once again the stripped section of the straight wire touches the safety pin and the circuit is again complete
- The cycle starts again and as a result we have a fully working motor.

Clockwise Current

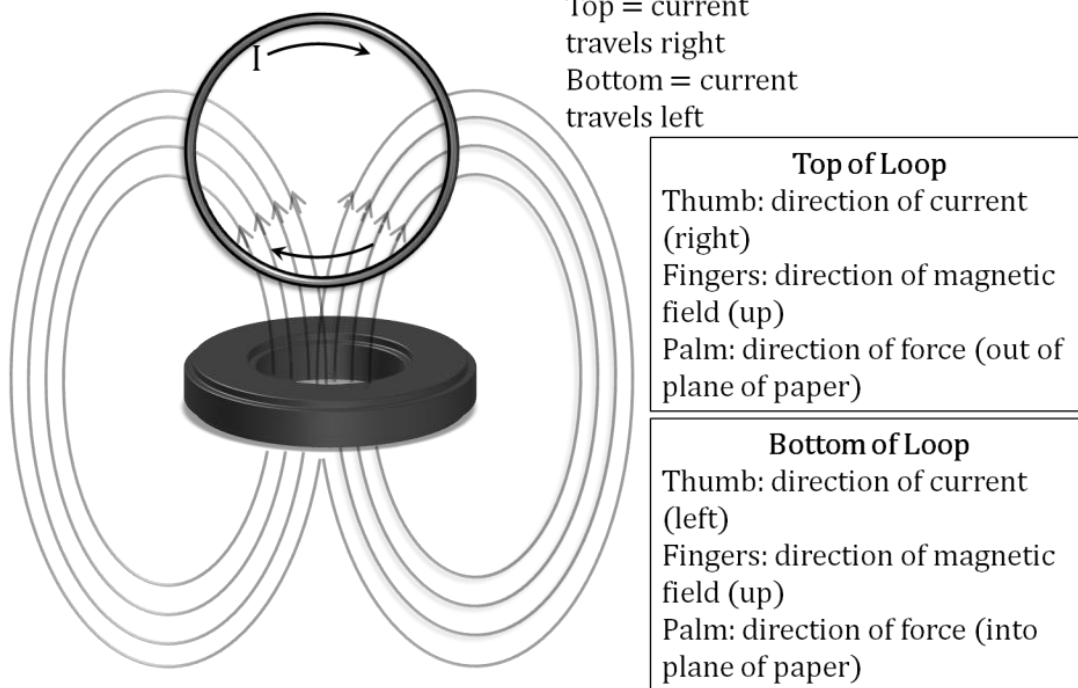


Figure 5: A single copper loop inside the magnetic field of a ring magnet. The figure also illustrates the direction of electric current.

Counterclockwise Current

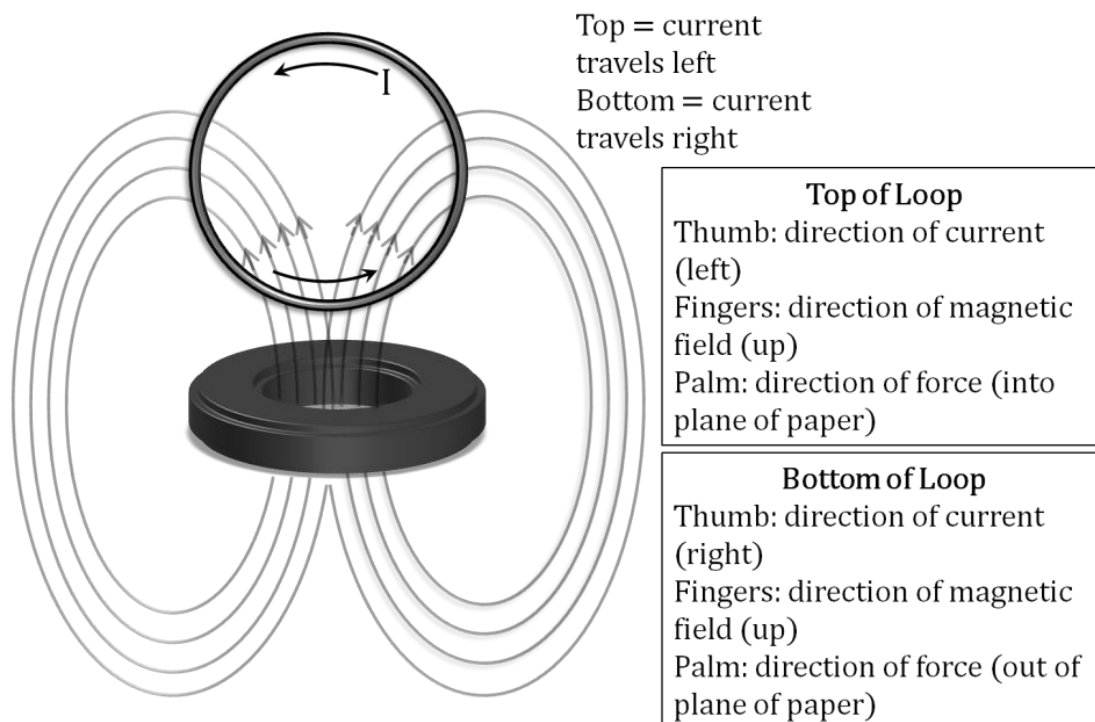


Figure 6: A single copper loop inside the magnetic field of a ring magnet. The figure also illustrates the direction of electric current.

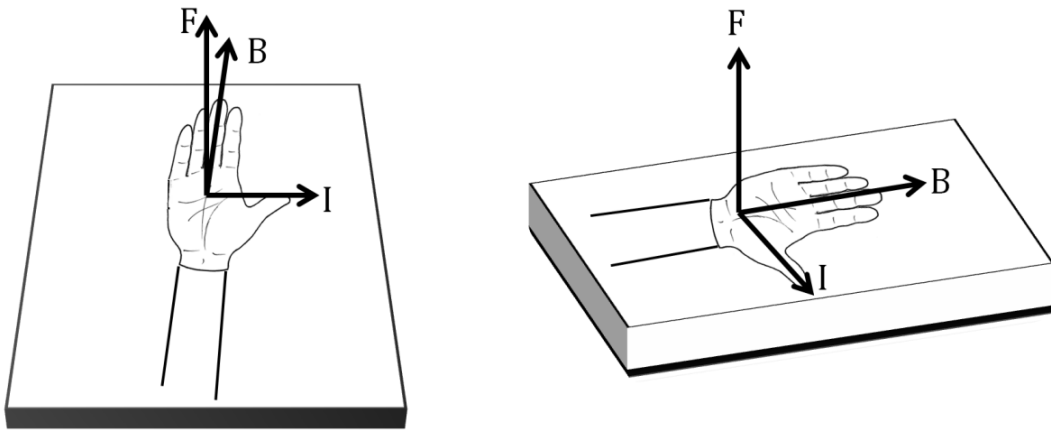


Figure 7: Right Hand Rule. Thumb: current, Fingers: magnetic field, Palm: force

➤ The robotic vehicle

The main body

- Provide each team the robotic kit and the construction instructions.
- Do not focus on the construction process for two reasons: Firstly the construction of the main body of the vehicle is fairly simple and secondly the purpose of this challenge is not the construction process but the process of programming the vehicle in order to accomplish certain tasks.
- Spend time only in the welding of the power supply wires to the motors. Have students use the soldering iron (in order to save time the teacher can have the wires already soldered).
- Have students experiment with the motors before moving to the construction of the main body. Provide each student team with two alligator clip wires, a motor and a 1.5 V (AA) battery. Have them connect the motor's terminals with the battery's poles. Ask them to reverse the polarity and note down their observation (see Fig. 8).
- Ask them to draw a simple wiring diagram which shows when the motor spins clockwise and when it spins anticlockwise.

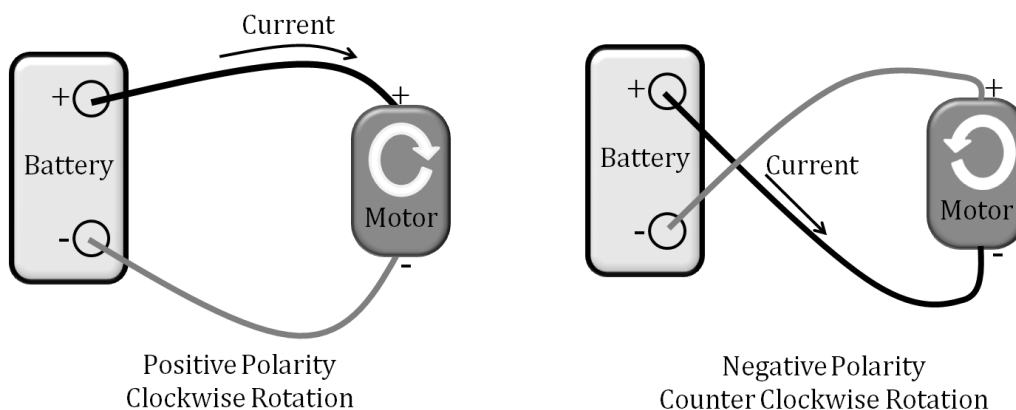


Figure 8

- Proceed to the construction of the robot's main body (Fig. 9).

- Move to the construction of the electric circuit by connecting the power supply to the motors and the on-off switch. Have student teams use the soldering iron and solder to attach the red and black wires to the motors (Fig. 10).

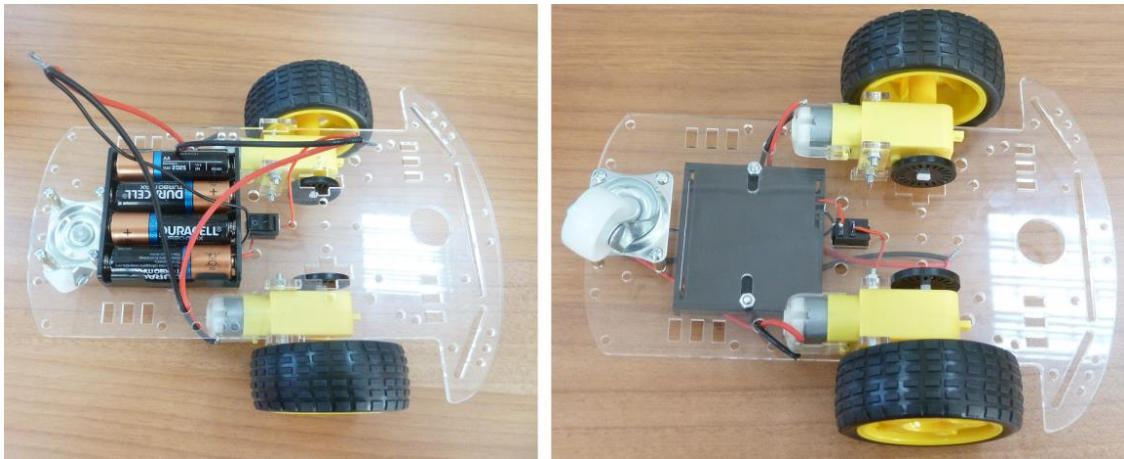
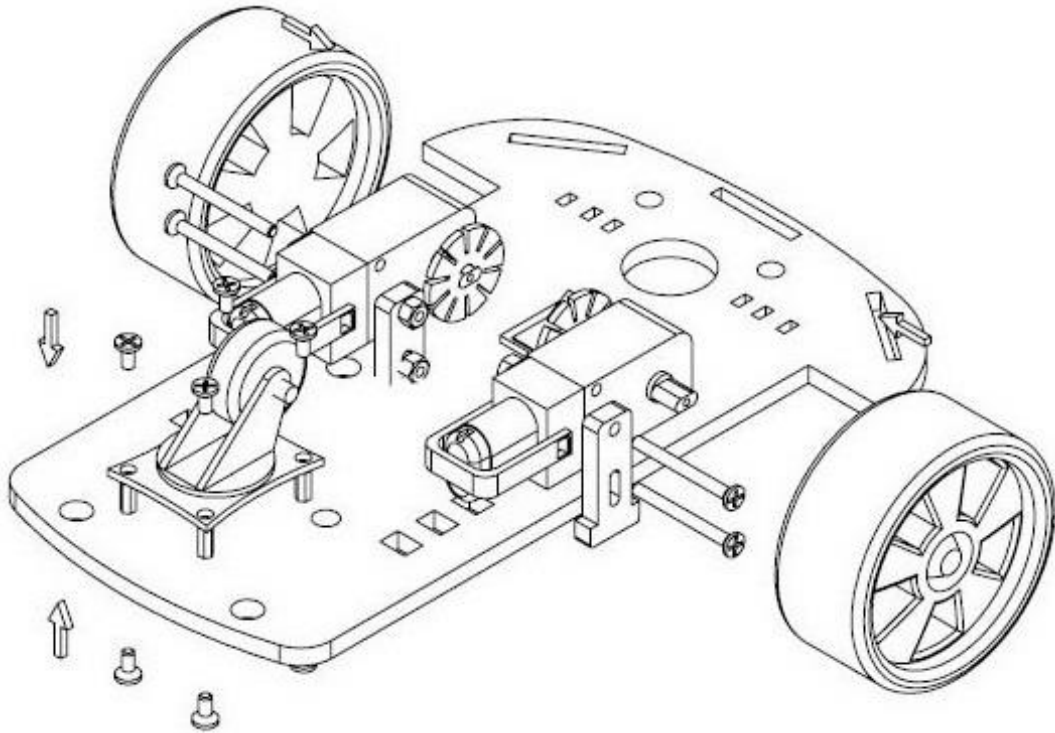


Figure 9: Main body of the robotic vehicle (chassis, fixed wheels, castor wheel, motors, battery holder, on-off switch).

- Have students connect the on – off switch to the battery holder (Fig. 11). The red wire (+) coming from the battery holder must be cut into two parts (Fig. 12). The part coming from the holder is soldered to the one terminal of the switch. The one end of the remaining red wire is soldered to the other terminal of the switch (Fig. 12).

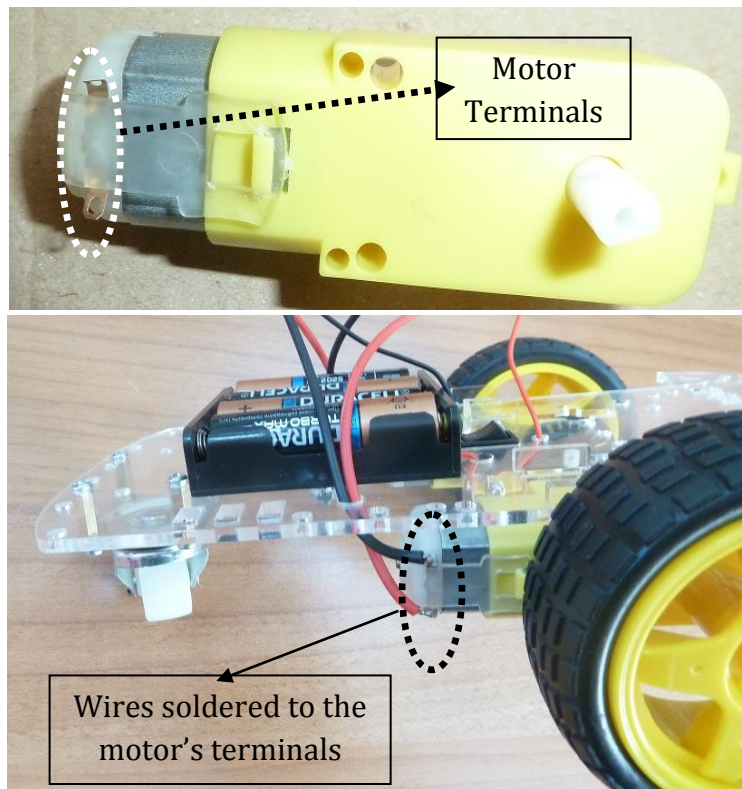


Figure 10

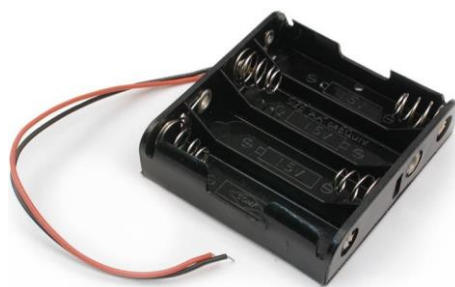


Figure 11: Battery Holder

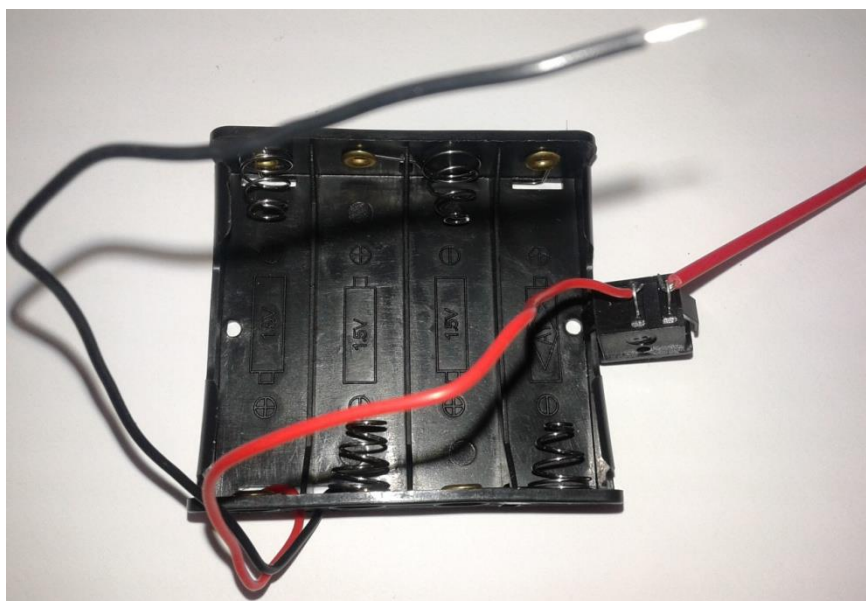


Figure 12

- Once the motor wires are soldered and the red wire of the battery holder is attached to the switch, have students slide all wires from the holes of the chassis so that they all meet on the upper side of the chassis (Figs. 13, 14).

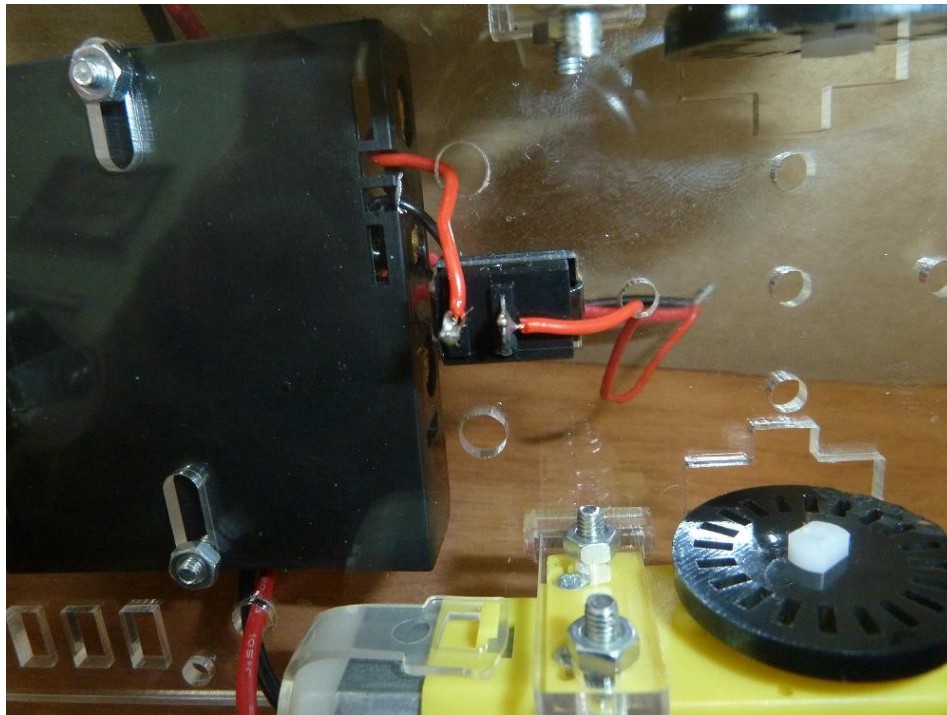


Figure 13

- The robotic vehicle has 6 wires in total (3 black and 3 red). More specific, each motor has 2 wires (1 red and 1 black) and so does the battery holder.

The circuit

Students have already drawn a diagram which explains how they can manipulate the direction of a motor's rotation. Students are called to answer the following questions through arranging different connections between the wires. Have student teams predict what they believe will happen, before they begin testing.

- How will the vehicle move if both motors rotate clockwise?
- How will the vehicle move if both motors rotate anticlockwise?
- How will the vehicle move if the right motor rotates clockwise while the left rotates anticlockwise? What will happen if the reverse takes place?
- How are you going to connect the wires so that the vehicle will move forward?
- How are you going to connect the wires so that the vehicle will move backward?
- How are you going to connect the wires so that the vehicle will rotate about its axis of symmetry (forwards and backward)?
- How are you going to connect the wires so that the vehicle will rotate about the right or the left wheel (backward and forward)?

Right Wheel		Left Wheel	
$\begin{pmatrix} B_r - B_b \\ R_r - R_b \end{pmatrix}$	Clockwise	$\begin{pmatrix} B_l - B_b \\ R_l - R_b \end{pmatrix}$	Counter-clockwise
$\begin{pmatrix} B_r - R_b \\ R_r - B_b \end{pmatrix}$	Counter-clockwise	$\begin{pmatrix} B_l - R_b \\ R_l - B_b \end{pmatrix}$	Clockwise
Forward		Backward	
$\begin{pmatrix} B_r - B_b - B_l \\ R_r - R_b - R_l \end{pmatrix}$	R _{Wheel} : Clockwise L _{Wheel} : Anticlockwise	$\begin{pmatrix} B_r - R_b - B_l \\ R_r - B_b - R_l \end{pmatrix}$	R _{Wheel} : Anticlockwise L _{Wheel} : Clockwise
Rotation about axis of symmetry			
Clockwise Rotation		Anticlockwise Rotation	
$\begin{pmatrix} B_r - R_b \\ R_r - B_b \end{pmatrix} + \begin{pmatrix} B_l - B_b \\ R_l - R_b \end{pmatrix}$		$\begin{pmatrix} B_r - B_b \\ R_r - R_b \end{pmatrix} + \begin{pmatrix} B_l - R_b \\ R_l - B_b \end{pmatrix}$	
Rotation about the right wheel		Rotation about the left wheel	
$\begin{pmatrix} B_l - R_b \\ R_l - B_b \end{pmatrix}$	Backward rotation around right wheel	$\begin{pmatrix} B_r - R_b \\ R_r - B_b \end{pmatrix}$	Backward rotation around left wheel
$\begin{pmatrix} B_l - B_b \\ R_l - R_b \end{pmatrix}$	Forward rotation around right wheel	$\begin{pmatrix} B_r - B_b \\ R_r - R_b \end{pmatrix}$	Forward rotation around left wheel

Where:

B_r: Black wire of right wheel

B_b: Black wire of battery holder

B_l: Black wire of left wheel

R_r: Red wire of right wheel

R_b: Red wire of battery holder

R_l: Red wire of left wheel

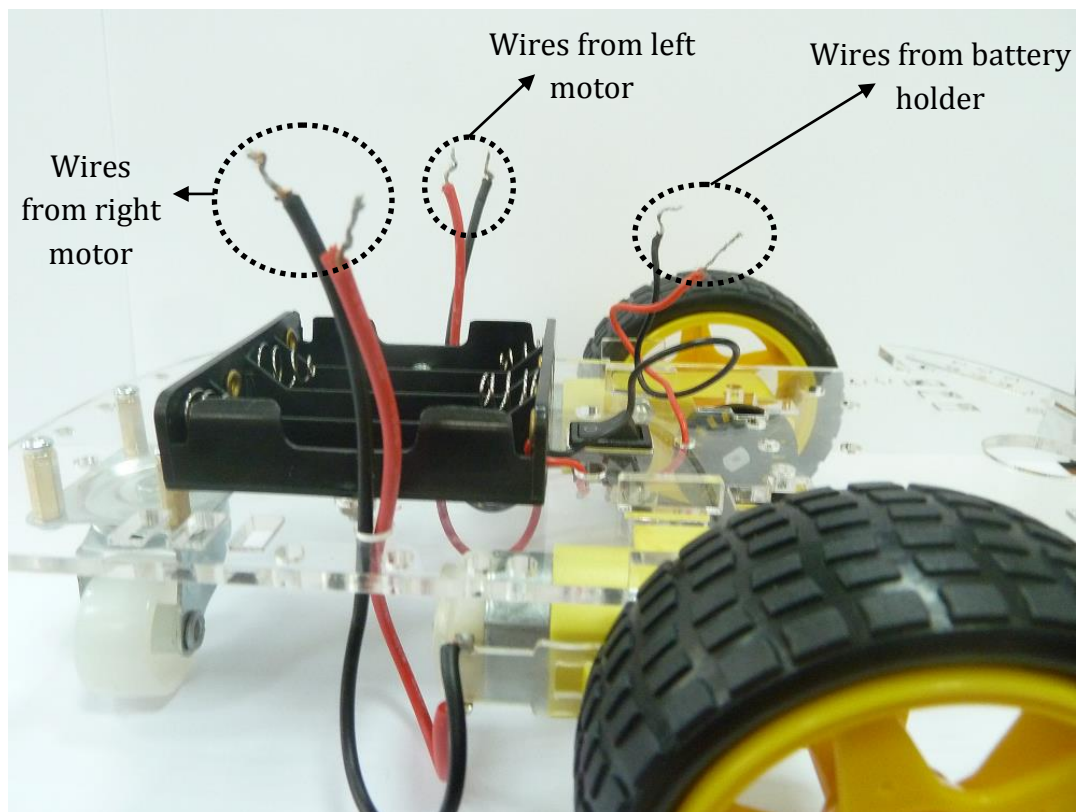


Figure 14

Activity 4 – Solve sub-problems

Duration: 120 minutes

Objectives: In this activity students will

- solve each sub-problem based on their plans
- construct the electronic circuit
- write simple algorithms
- investigate how the robot responds to simple commands
- realize how vehicles navigate through space

General Context

In this activity students are introduced to the core of the Engineering Design Process and apply the corresponding steps of EDP to face their challenge. In order to face and solve each sub-problem they follow the circle: design-build-test-improve. As a part of the whole EDP process students need to recall the scientific knowledge they gained in activity 3.

❖ Working in groups

The teacher summarizes the conclusions of Activities 1, 2 and 3. As student teams have already defined the individual sub-problems, the teacher encourages and guides student teams to gradually solve each one of the sub-problems that the main challenge has been divided into. The teacher encourages the teams to draft a plan of their work.

- Sub-problem 1: Controlling the motors

The microcontroller

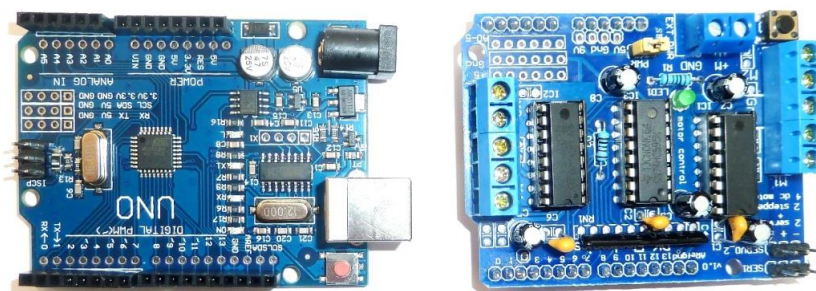


Figure 15: Left: The microcontroller. Right: The motor shield

The microcontroller is a computing device that can execute a program (i.e. a sequence of instructions) and is referred to as the “brain” of the robot as it is responsible for all computations, decision making and communications.

The motor shield is a double full-connect driver, which is used to drive two dc engines with a microcontroller². A motor shield often has its own lines for

² <http://www.igi-global.com/dictionary/motor-shield/55774>

power input, allowing you to not only use significantly higher amounts of current, but a wide range of voltages too. Some DC motors run much better at 6V, 12V, 24V, or any other voltage that's not readily provided by the microcontroller. Another advantage of the shield is that it simplifies the wiring (and allowing features like motor direction reversal) - eliminating the need for a breadboard and reducing or eliminating any soldering necessary³.

Preparing the electronic circuit

- Attach the motor shield to the arduino microcontroller (see Fig. 16).
- Use double sided tape to attach the microcontroller+shield on the vehicles chassis (see Figs. 17, 18).
- Attach the motors' wires and the battery holder's wires to the terminal blocks of the motor shield (see Figs. 18, 19). The motor shield has terminals for four motors (M1, M2, M3, M4, see Fig. 19). It also has a terminal block called GND (ground, see Fig. 19). Attach the right motor's wires to terminal M3 as shown in Fig 14. Attach the left motor's wires to terminal M2 as shown in Fig. 19. Attach the battery holder's wires to the +M and GND terminal as shown in Fig. 19.
- Download and install the Arduino software. The latest version is "*arduino-1.8.2-windows.exe*" and can be found in the following address: <https://www.arduino.cc/en/Main/Software>. In the case you are using the Arduino "Arduino UNO R3 ATmega328P Board" download and install the following driver: "CH341SER" which can be found in the following address: <http://www.arduinoed.eu/ch340g-converter-windows-7-driver-download/>. The zip file contains two folders. In folder "*CH341SER*" execute the "SETUP.EXE" and install the driver.
- Visit <https://learn.adafruit.com/adafruit-motor-shield/library-install> and follow the steps given below in order to install the "*Motor-Shield-library*".

Library Installation

Before you can use the Motor shield, you **must** install the **AF_Motor** Arduino library - this will instruct the Arduino how to talk to the Adafruit Motor shield, and it isn't optional!

1. [First, grab the library from github](#)
2. Uncompress the ZIP file onto your desktop
3. Rename the uncompressed folder **AFMotor**
4. Check that inside **AFMotor** is **AFMotor.cpp** and **AFMotor.h** files. If not, check the steps above
5. Place the **AFMotor** folder into your *arduinofolder/libraries* folder. For Windows, this will probably be something like **MY Documents/Arduino/libraries** for Mac it will be something like **Documents/arduino/libraries**. If this is the first time you are installing a library, you'll need to create the **libraries** folder. Make sure to call it **libraries** exactly, no caps, no other name.

³ <https://www.quora.com/Why-is-it-common-to-use-a-motor-shield-with-an-Arduino>



6. Check that inside the **libraries** folder there is the **AFMotor** folder, and inside **AFMotor** is **AFMotor.cpp** **AFMotor.h** and some other files
7. Quit and restart the IDE. You should now have a submenu called **File->Examples->AFMotor->MotorParty**

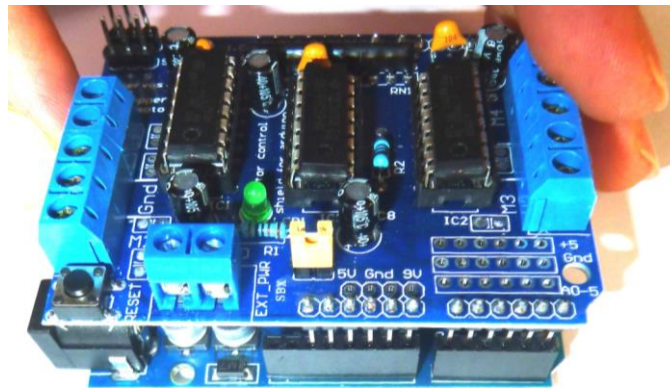


Figure 16: The motor shield attached to the microcontroller

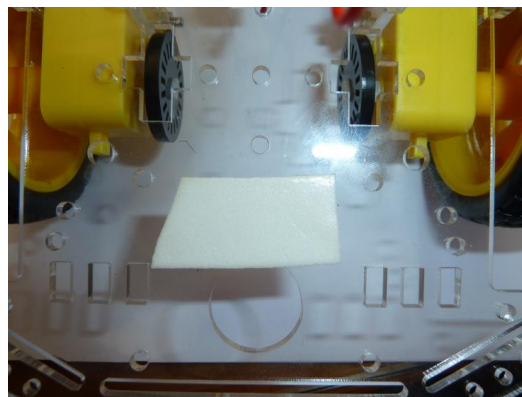


Figure 17: Double sided duct tape placed on the vehicle's chassis

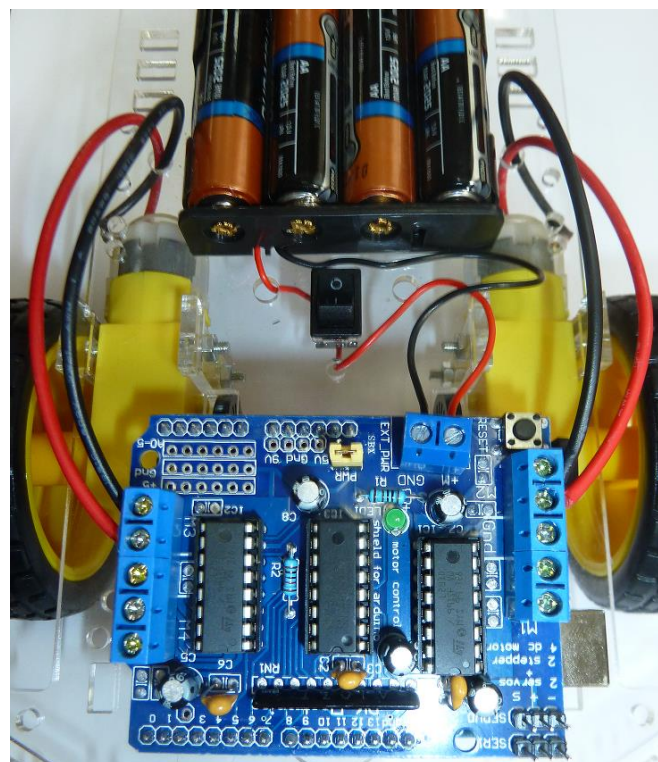


Figure 18: Electronic circuit (microcontroller + motor shield) attached to the vehicle's chassis.

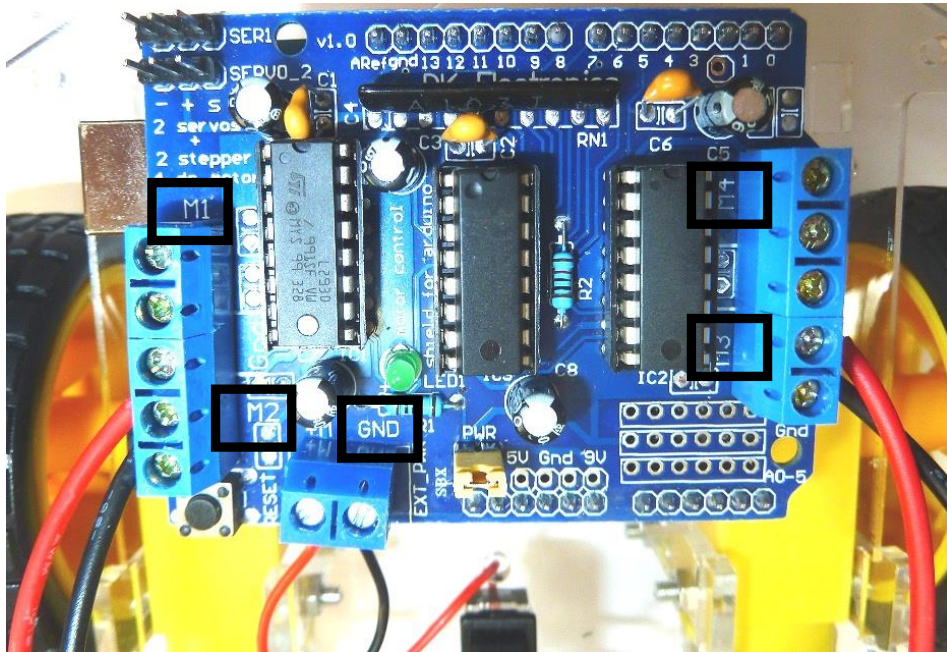


Figure19: Motors' wires attached to M2, M3 shield's terminals and battery holder's wires attached to GND terminal.

Preparing Arduino

- Double click the Arduino icon on your desktop to open the software. You should see something like in Fig. 20.

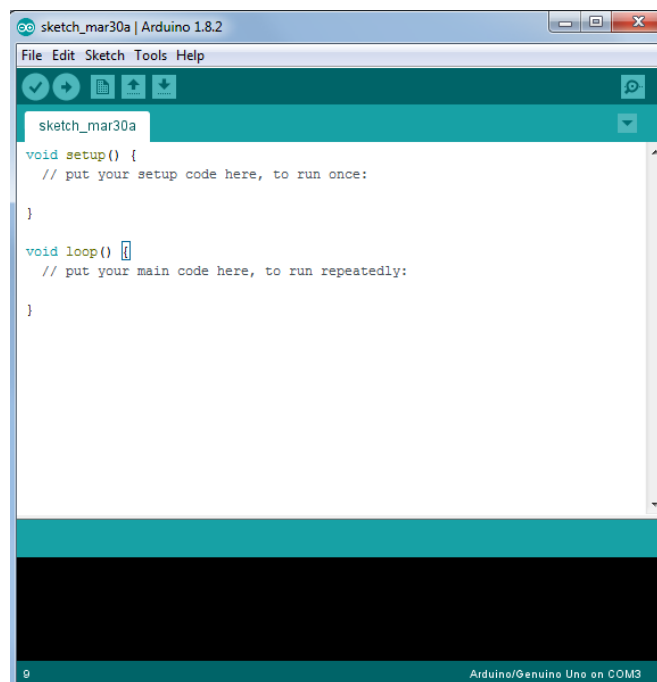


Figure 20: Arduino 1.8.2 Main Window

- Go to Tools->Board and select "Arduino/Genuino Uno"

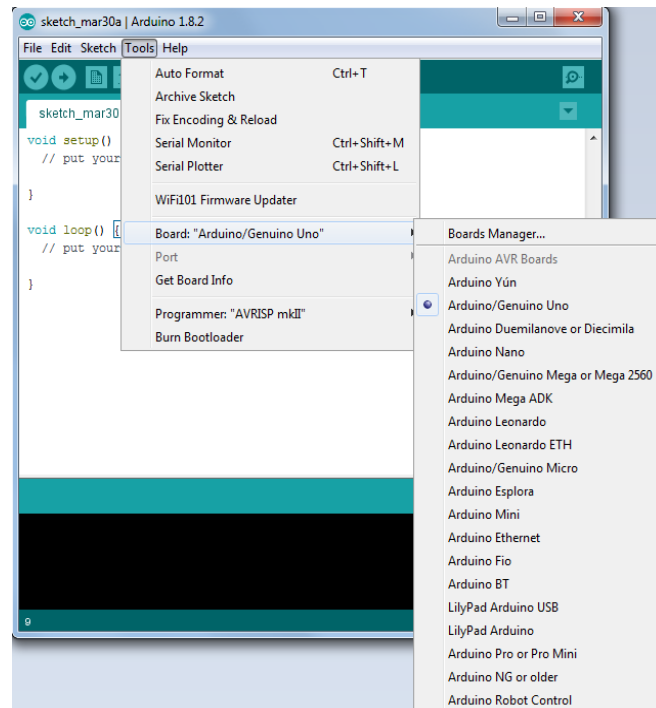


Figure 21

- Use the usb cable to connect the microcontroller to your pc. Go to Tools->Port. If you have a genuine Arduino Uno then under **port** you will find Arduiono Uno COM1 (there is a possibility that you will find Arduiono Uno COM2 or COM3). In the case you are using a clone of Arduino Uno then under **port** you will find either COM1 or COM2 or COM3. In case that there are two indications e.g. COM1 and COM3 you have to determine which port corresponds to the microcontroller. Unplug the microcontroller by removing the usb cable from your pc. Go to Tools->port and check which port disappeared after unplugging the microcontroller. Plug the microcontroller again and choose the right port.

Simple Commands

Experimenting with the code

Provide students with the following code and explain each command line

#include <AFMotor.h>	(loads the AF_Motor library for the motor shield)
AF_DCMotor left_motor(<u>here you put the number of the shield's terminal on which you attached the left motor's cables - in our case we attached the left motor to terminal M2 so we set 2</u>); //create motor #2 AF_DCMotor right_motor(<u>here you put the number of the shield's terminal on which you attached the right motor's cables - in our case we attached the right motor to terminal M3 so we set 3</u>); //create motor #3	(declare the left motor) (declare the right motor)
void setup() { left_motor.setSpeed(set motor speed); //set the speed to 200/255 right_motor.setSpeed(set motor speed); //set the speed to 200/255 }	Set the speed for each motor. Maximum speed is 255.
void loop() { left_motor.run(set motor status); // turn it on going forward right_motor.run(set motor status); // turn it on going forward }	Motor Status: FORWARD, BACKWARD or RELEASE (RELEASE: the motor does not rotate)
Full Code Example	
<pre>#include <AFMotor.h> AF_DCMotor left_motor(2); // create motor #2 AF_DCMotor right_motor(3); // create motor #3 void setup() { left_motor.setSpeed(200); // set the speed to 200/255 right_motor.setSpeed(200); // set the speed to 200/255 } void loop() { left_motor.run(FORWARD); // turn it on going forward right_motor.run(FORWARD); // turn it on going forward }</pre>	

Students are asked to write simple commands that will control each motor individually or both motors simultaneously. Using different combinations, student teams are called to answer the following questions:

- How will the vehicle move forward?
- How will the vehicle move backwards?
- How will you make the vehicle rotate about its axis of symmetry (forwards and backward-axial or spin turn)?
- How will you make the vehicle rotate about the right or the left wheel (backward and forward-radial or pivot turn)?
- How will the robotic vehicle perform smooth right and left turns?

Use the USB AM-BM Printer Cable to connect the Arduino microcontroller to your PC.

[1]. Forward

```
#include <AFMotor.h>
AF_DCMotor left_motor(2); // create motor #2
AF_DCMotor right_motor(3); // create motor #3

void setup() {
  left_motor.setSpeed(200); // set the speed to 200/255
  right_motor.setSpeed(200); // set the speed to 200/255
}

void loop() {
  left_motor.run(FORWARD); // turn it on going forward
  right_motor.run(FORWARD); // turn it on going forward
}
```

[2]. Backward

```
#include <AFMotor.h>
AF_DCMotor left_motor(2); // create motor #2
AF_DCMotor right_motor(3); // create motor #3

void setup() {
  left_motor.setSpeed(200); // set the speed to 200/255
  right_motor.setSpeed(200); // set the speed to 200/255
}

void loop() {
  left_motor.run(BACKWARD); // turn it on going backward
  right_motor.run(BACKWARD); // turn it on going backward
}
```

[3]. Axial left turn.

```
#include <AFMotor.h>
AF_DCMotor left_motor(2); // create motor #2
AF_DCMotor right_motor(3); // create motor #3

void setup() {
  left_motor.setSpeed(200); // set the speed to 200/255
  right_motor.setSpeed(200); // set the speed to 200/255
}

void loop() {
  left_motor.run(BACKWARD); // turn it on going backward
  right_motor.run(FORWARD); // turn it on going forward
}
```



[4]. Axial Right Turn

```
#include <AFMotor.h>
AF_DCMotor left_motor(2); // create motor #2
AF_DCMotor right_motor(3); // create motor #3

void setup() {
  left_motor.setSpeed(200); // set the speed to 200/255
  right_motor.setSpeed(200); // set the speed to 200/255
}

void loop() {
  left_motor.run(FORWARD); // turn it on going forward
  right_motor.run(BACKWARD); // turn it on going backward
}
```

[5]. Radial Left Forward Turn

```
#include <AFMotor.h>
AF_DCMotor left_motor(2); // create motor #2
AF_DCMotor right_motor(3); // create motor #3

void setup() {
  left_motor.setSpeed(200); // set the speed to 200/255
  right_motor.setSpeed(200); // set the speed to 200/255
}

void loop() {
  left_motor.run(RELEASE); // turn it off
  right_motor.run(FORWARD); // turn it on going forward
}
```

[6]. Radial Left Backward Turn

```
#include <AFMotor.h>
AF_DCMotor left_motor(2); // create motor #2
AF_DCMotor right_motor(3); // create motor #3

void setup() {
  left_motor.setSpeed(200); // set the speed to 200/255
  right_motor.setSpeed(200); // set the speed to 200/255
}

void loop() {
  left_motor.run(BACKWARD); // turn it on going forward
  right_motor.run(RELEASE); // turn it off
}
```


[7]. Radial Right Forward Turn

```
#include <AFMotor.h>
AF_DCMotor left_motor(2); // create motor #2
AF_DCMotor right_motor(3); // create motor #3
void setup() {
  left_motor.setSpeed(200); // set the speed to 200/255
  right_motor.setSpeed(200); // set the speed to 200/255
}
void loop() {
  left_motor.run(FORWARD); // turn it on going forward
  right_motor.run(RELEASE); // turn it off
}
```

[8]. Radial Right Backward Turn

```
#include <AFMotor.h>
AF_DCMotor left_motor(2); // create motor #2
AF_DCMotor right_motor(3); // create motor #3

void setup() {
  left_motor.setSpeed(200); // set the speed to 200/255
  right_motor.setSpeed(200); // set the speed to 200/255
}

void loop() {
  left_motor.run(RELEASE); // turn it off
  right_motor.run(BACKWARD); // turn it on going forward
}
```

[9]. Smooth Turn

A smooth turn is performed when both wheels are driving in the same direction but one wheel is turning faster and the faster turning wheel becomes the outside wheel. These turns can be adjusted by changing the relative speeds of the wheels. So by manipulating the motors' speeds one can adjust the smoothness of the turn.

Right Turn

```
#include <AFMotor.h>
AF_DCMotor left_motor(2); // create motor #2
AF_DCMotor right_motor(3); // create motor #3

void setup() {
  left_motor.setSpeed(200); // set the speed to 200/255
  right_motor.setSpeed(160); // set the speed to 160/255
}

void loop() {
  left_motor.run(FORWARD); // turn it on going forward
  right_motor.run(FORWARD); // turn it on going forward
}
```

Left Turn

```
#include <AFMotor.h>
AF_DCMotor left_motor(2); // create motor #2
AF_DCMotor right_motor(3); // create motor #3

void setup() {
  left_motor.setSpeed(160); // set the speed to 160/255
  right_motor.setSpeed(200); // set the speed to 200/255
}

void loop() {
  left_motor.run(FORWARD); // turn it on going forward
  right_motor.run(FORWARD); // turn it on going forward
}
```

The sensors

Attaching the sensors

In order to attach and connect the sensors to the shield you first have to solder some female pin header connectors (see Fig. 22) to the shield. By soldering the pin headers to the shield you can very easily connect and disconnect jumper wires without having to solder them.



Figure 22

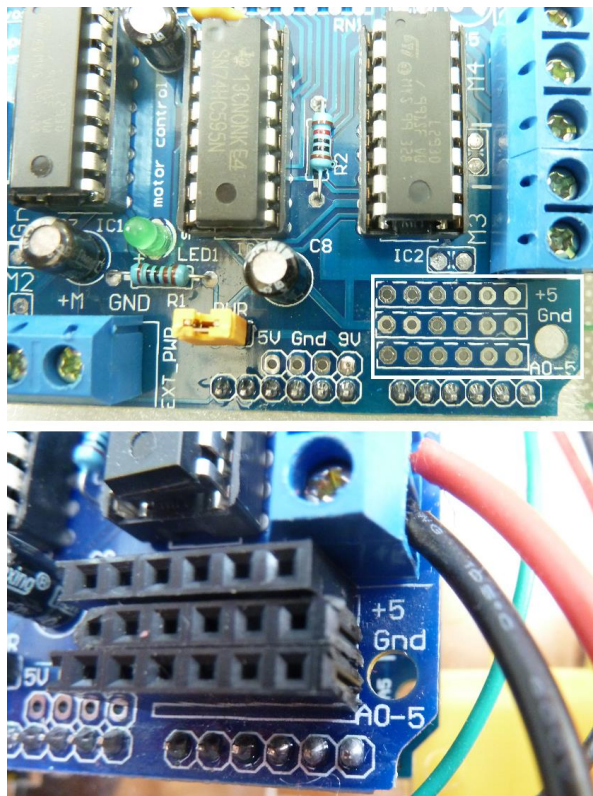


Figure 23

The motor shield has three rows of six pin holes each (see Fig. 23). The first row is “+5” the second is “Gnd” and the third is “A0-5” (which means A0, A1, A2,...,A5). Attach three pin headers (of 6 pins each) to these holes as shown in Fig. 18. Afterwards, turn the motor shield upside down and solder the pins using the soldering iron (see Fig. 24). Finally, reattach the motor shield to the Arduino microprocessor.

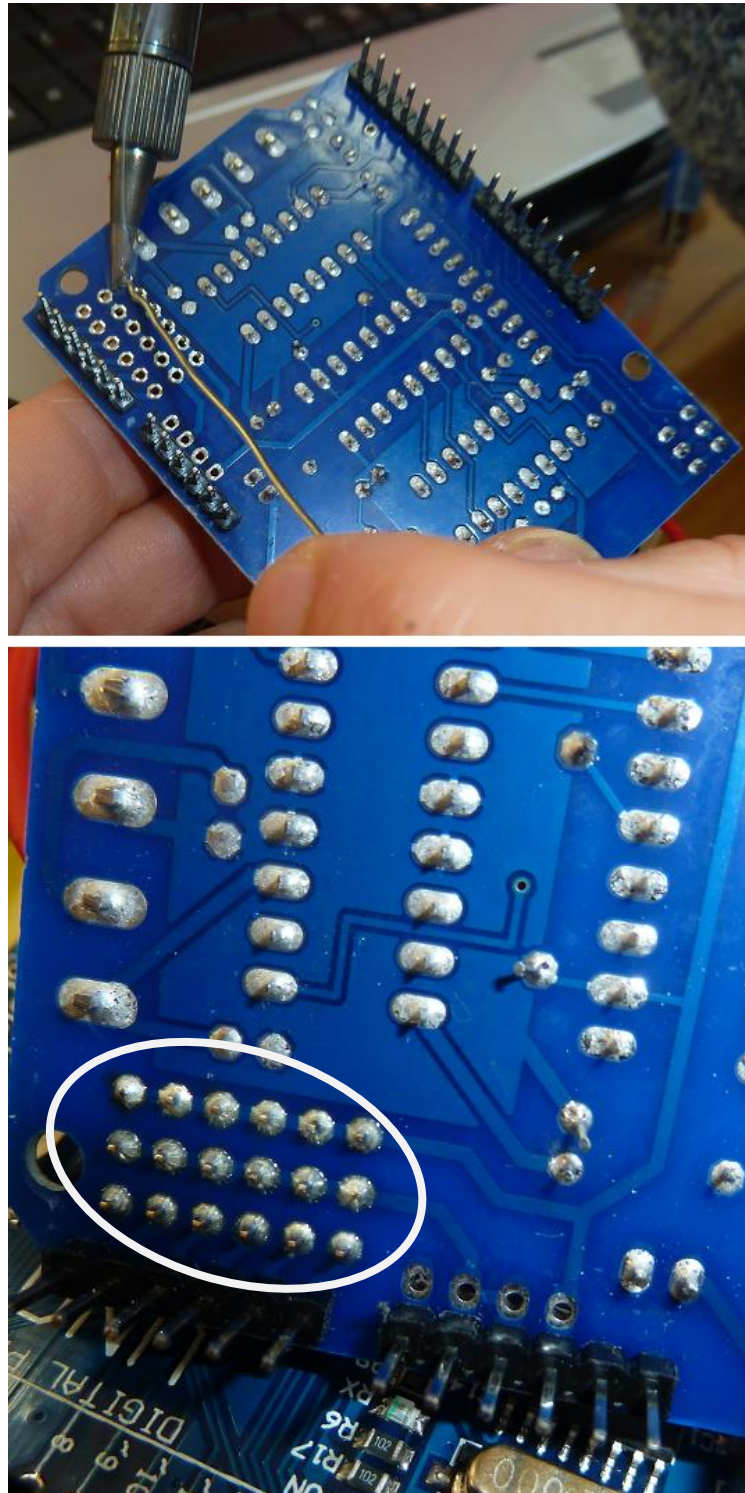


Figure 24

Each sensor has four pins. Two of them (Vcc and GND) go to the power supply. Connect two, female to male, jumper wires to the sensor as shown in Fig. 25. Get

two more female to male, jumper wires and connect them to the sensor as shown in Fig. 26.

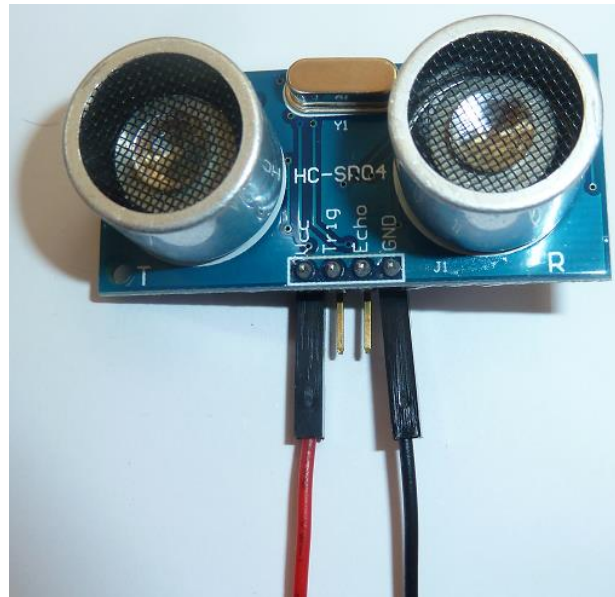


Figure 25

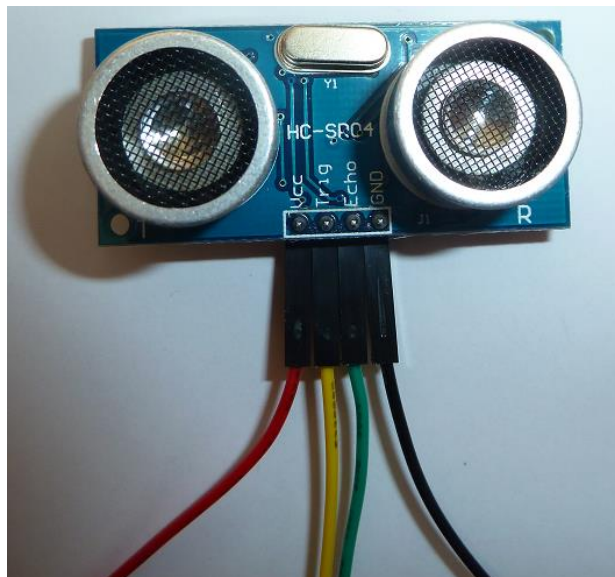


Figure 26

Attach the sensor to the vehicle as shown in Fig. 27. Use some pressure-sensitive adhesive (like blue tack) to glue the sensor as shown in Fig. 28. For the connections do as follows:

- The red wire (from Vcc in Fig. 26) goes to the pin hole “+5” (see Fig. 29)
- The black wire (from GND) goes to the pin hole Gnd (see Fig. 29)
- The yellow wire (from Trig) goes to the pin hole A0 (see Fig. 29)
- The green wire (from Echo) goes to the pin hole A1 (see Fig. 29)

Important Notice!

Arduino microcontroller has 20 ports in total. The 14 of them are digital while the remaining six are analog (these six can also be used as digital ports). When

we attach the motor shield to the Arduino microcontroller we connect the 6 ports A0-5 of the shield with the corresponding analog ports of the Arduino microcontroller. In order to use these 6 analog ports as digital, when writing the code we define ports A0-5 as 14-19 (i.e. A0 =14 while A5 = 19).

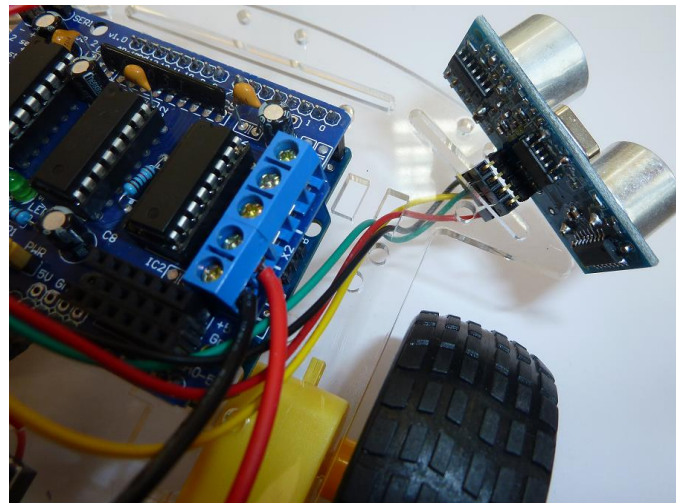


Figure 27

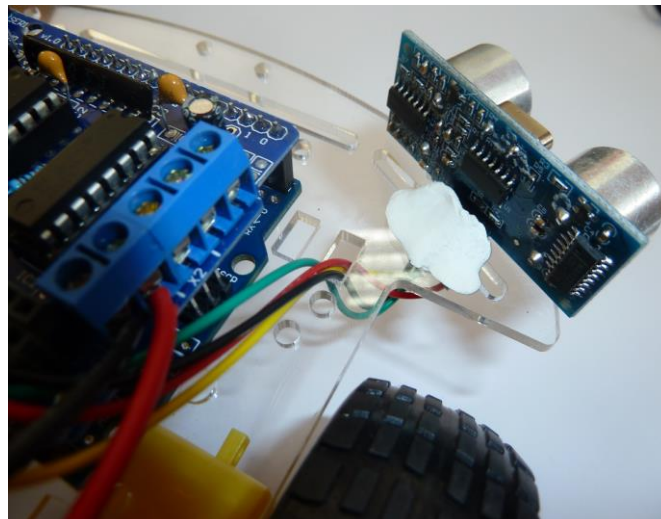


Figure 28

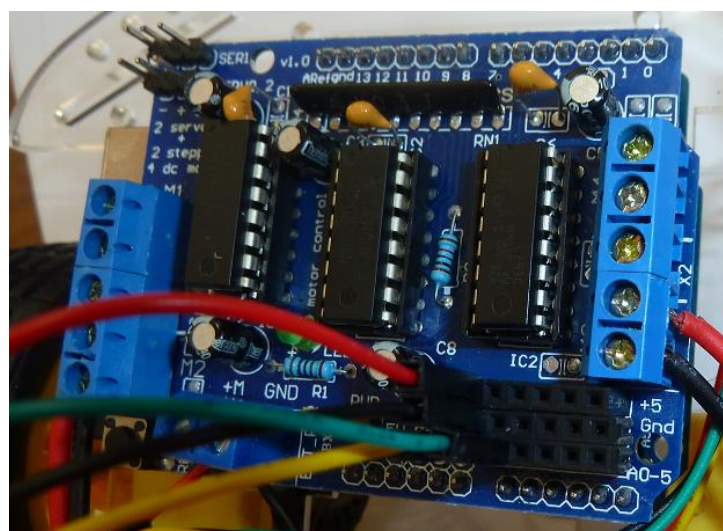


Figure 29

Experimenting with sensors

Provide students with the following code and explain each command line

<pre>#define RightSonar 15 #define RightSonarTrig 14</pre>	<p>Define the Arduino port in which the sensor is connected.</p>
<pre>long ultrasonar(int trigPin, int echoPin){ long duration;</pre>	<p>Define a function with parameters trigPin and echoPin</p>
<pre>pinMode(trigPin, OUTPUT); // turn on output trigger pin digitalWrite(trigPin, LOW); delayMicroseconds(2); digitalWrite(trigPin, HIGH); delayMicroseconds(5); digitalWrite(trigPin, LOW); pinMode(echoPin, INPUT); // Switch signalpin to input digitalWrite(echoPin, HIGH); // Turn on pullup resistor</pre>	<p><u>pinMode()</u> Configures the specified pin to behave either as an input or an output.</p> <p><u>delay()</u> Pauses the program for the amount of time (in milliseconds) specified as parameter.</p> <p><u>digitalWrite()</u> Write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V for HIGH, 0V (ground) for LOW. If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin.</p>
<pre>duration = pulseIn(echoPin, HIGH, 2000);</pre>	<p><u>pulseIn()</u> Reads a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH, pulseIn() waits for the pin to go HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds or 0 if no complete pulse was received within the timeout.</p>
<pre>pinMode(trigPin, INPUT); // shut off pin to avoid noise from other operations</pre>	
<pre> // Speed of sound is 340m/s = 0.034cm/us // Time = distance/speed <=> distance = Time*speed // Due to the fact that we calculate the time the sound takes to travel to the object // and then return back to the sensor, the actual distance is half from that we measure // That is why we divide by 2 return (long) (duration*0.034/2); }</pre>	



Final Code

```
#define RightSonar 15
#define RightSonarTrig 14

long ultrasonar(int trigPin, int echoPin){
  long duration;

  pinMode(trigPin, OUTPUT); // turn on output trigger pin

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT); // Switch signalpin to input
  digitalWrite(echoPin, HIGH); // Turn on pullup resistor

  duration = pulseIn(echoPin, HIGH, 2000);

  pinMode(trigPin, INPUT); // shut off pin to avoid noise from other
  operations

  // Speed of sound is 340m/s = 0.034cm/us
  // Time = distance/speed <=> distance = Time*speed
  // Due to the fact that we calculate the time the sound takes to travel
  to the object
  // and then retrun back to the sensor, the actual distance is half from
  that we measure
  // That is why we divide by 2

  return (long) (duration*0.034/2);
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(57600);
}

void loop() {
  // put your main code here, to run repeatedly:
  long rdist;

  rdist = ultrasonar(RightSonarTrig, RightSonar);

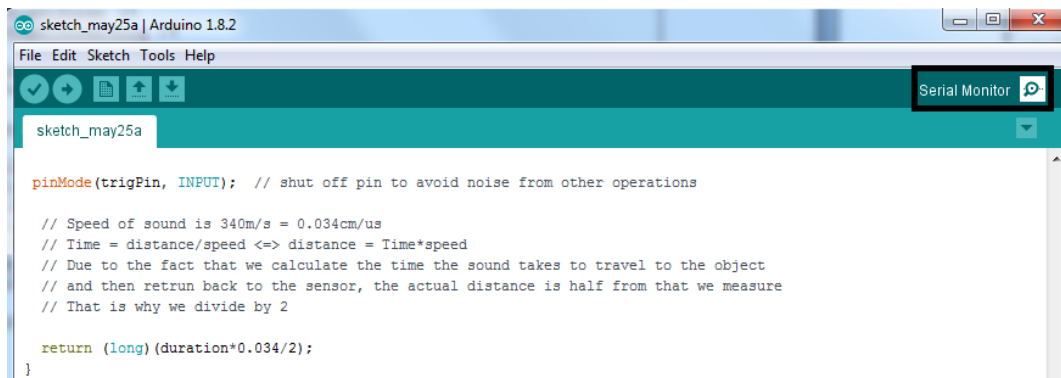
  Serial.println(rdist);
  delay(100);
}
```

Ask student teams to experiment with the code using the following instructions.

- Use the USB AM-BM Printer Cable to connect the Arduino microcontroller to your PC.
- Double click on the Arduino icon. Go to file and open a new project.
- Copy and paste the Final Code.
- Click on the Serial monitor icon on the top right corner of the main window (see Fig. 30). A window will appear (see Fig. 31).
- On the right lower corner of the new window you will find a drop down menu with the indication “baud”: Select a value for the “baud” which is the same as the value in the function “void setup()” (see Fig. 32). This

value determines the communication speed between the Arduino microcontroller and the computer.

- Select upload (see Fig. 33)



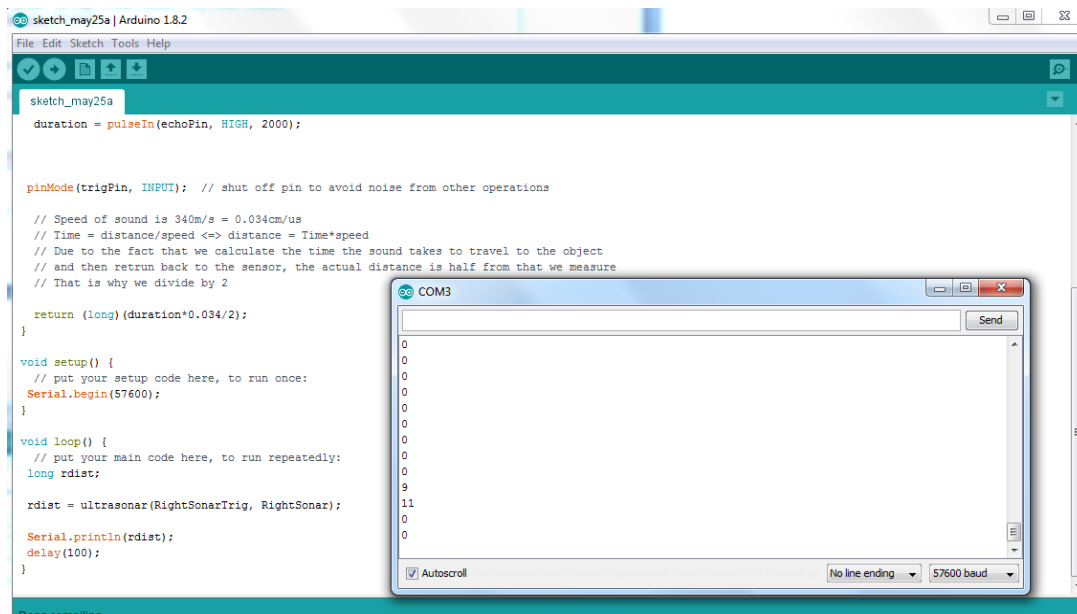
```
sketch_may25a

pinMode(trigPin, INPUT); // shut off pin to avoid noise from other operations

// Speed of sound is 340m/s = 0.034cm/us
// Time = distance/speed <=> distance = Time*speed
// Due to the fact that we calculate the time the sound takes to travel to the object
// and then retrun back to the sensor, the actual distance is half from that we measure
// That is why we divide by 2

return (long) (duration*0.034/2);
}
```

Figure 30



```
sketch_may25a

duration = pulseIn(echoPin, HIGH, 2000);

pinMode(trigPin, INPUT); // shut off pin to avoid noise from other operations

// Speed of sound is 340m/s = 0.034cm/us
// Time = distance/speed <=> distance = Time*speed
// Due to the fact that we calculate the time the sound takes to travel to the object
// and then retrun back to the sensor, the actual distance is half from that we measure
// That is why we divide by 2

return (long) (duration*0.034/2);

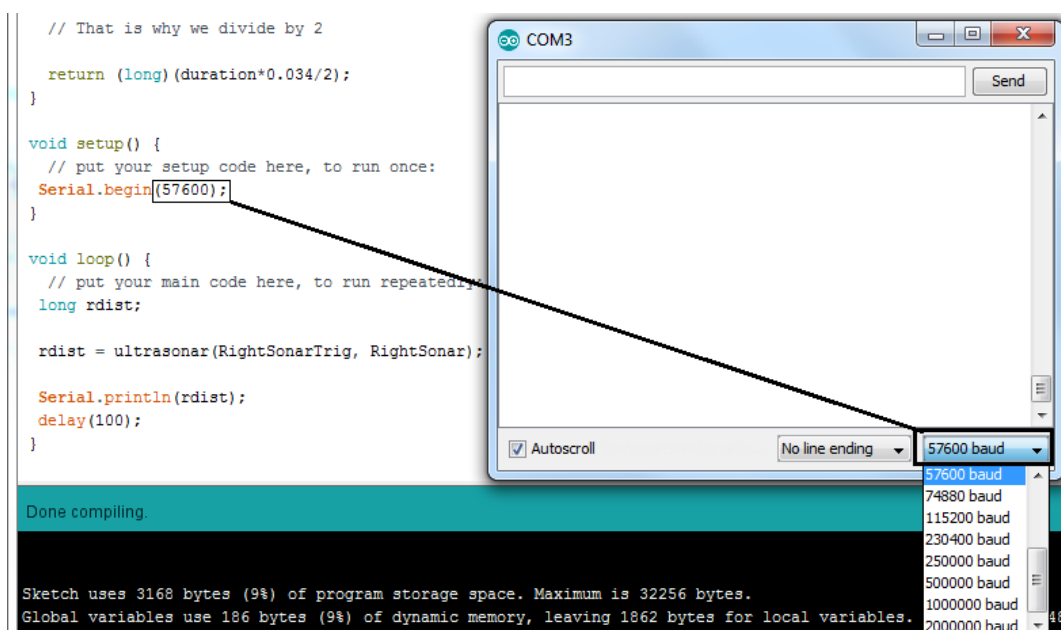
void setup() {
  // put your setup code here, to run once:
  Serial.begin(57600);
}

void loop() {
  // put your main code here, to run repeatedly:
  long rdist;

  rdist = ultrasonar(RightSonarTrig, RightSonar);

  Serial.println(rdist);
  delay(100);
}
```

Figure 31



```
// That is why we divide by 2

return (long) (duration*0.034/2);
}

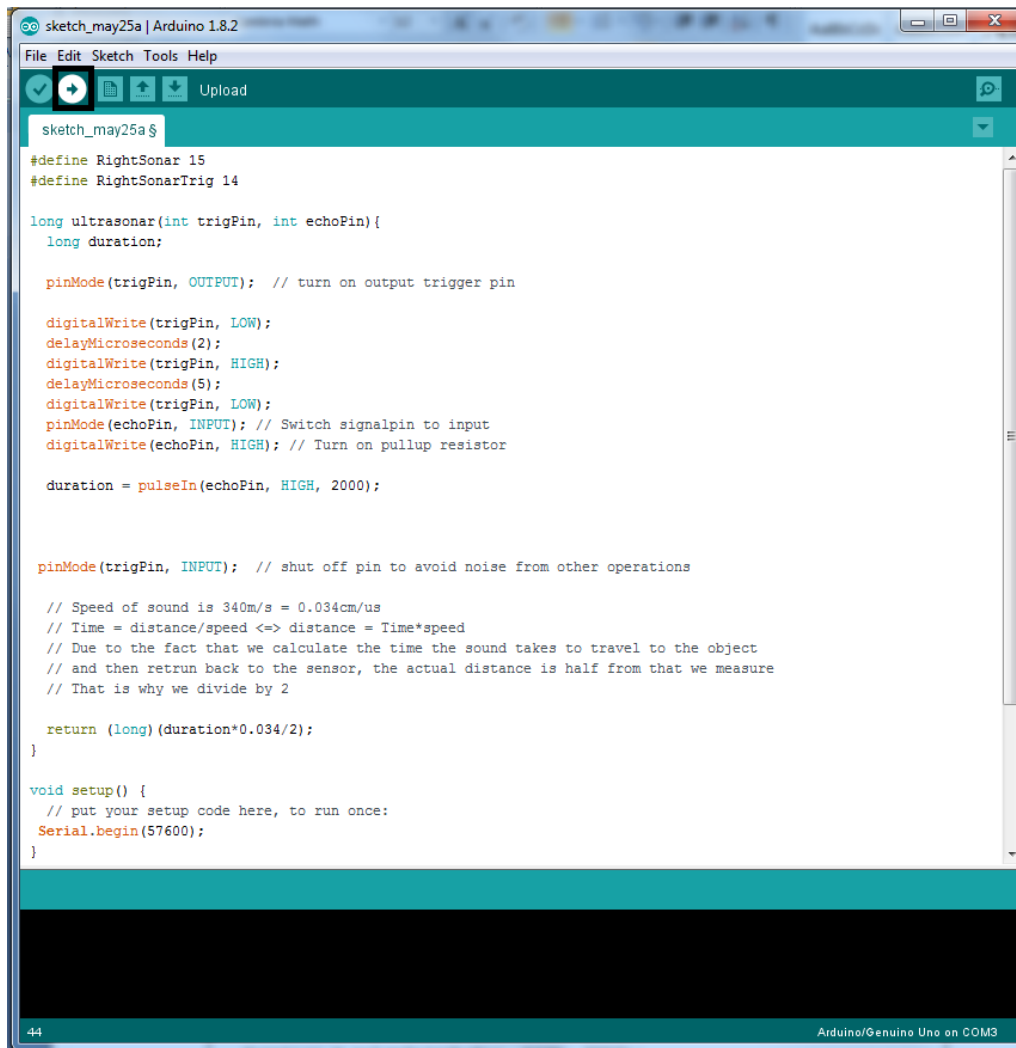
void setup() {
  // put your setup code here, to run once:
  Serial.begin(57600);
}

void loop() {
  // put your main code here, to run repeatedly:
  long rdist;

  rdist = ultrasonar(RightSonarTrig, RightSonar);

  Serial.println(rdist);
  delay(100);
}
```

Figure 32



```
sketch_may25a | Arduino 1.8.2
File Edit Sketch Tools Help
Upload
sketch_may25a $
#define RightSonar 15
#define RightSonarTrig 14

long ultrasonar(int trigPin, int echoPin){
  long duration;

  pinMode(trigPin, OUTPUT); // turn on output trigger pin

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT); // Switch signalpin to input
  digitalWrite(echoPin, HIGH); // Turn on pullup resistor

  duration = pulseIn(echoPin, HIGH, 2000);

  pinMode(trigPin, INPUT); // shut off pin to avoid noise from other operations

  // Speed of sound is 340m/s = 0.034cm/us
  // Time = distance/speed <=> distance = Time*speed
  // Due to the fact that we calculate the time the sound takes to travel to the object
  // and then return back to the sensor, the actual distance is half from that we measure
  // That is why we divide by 2

  return (long) (duration*0.034/2);
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(57600);
}

-44 Arduino/Genuino Uno on COM3
```

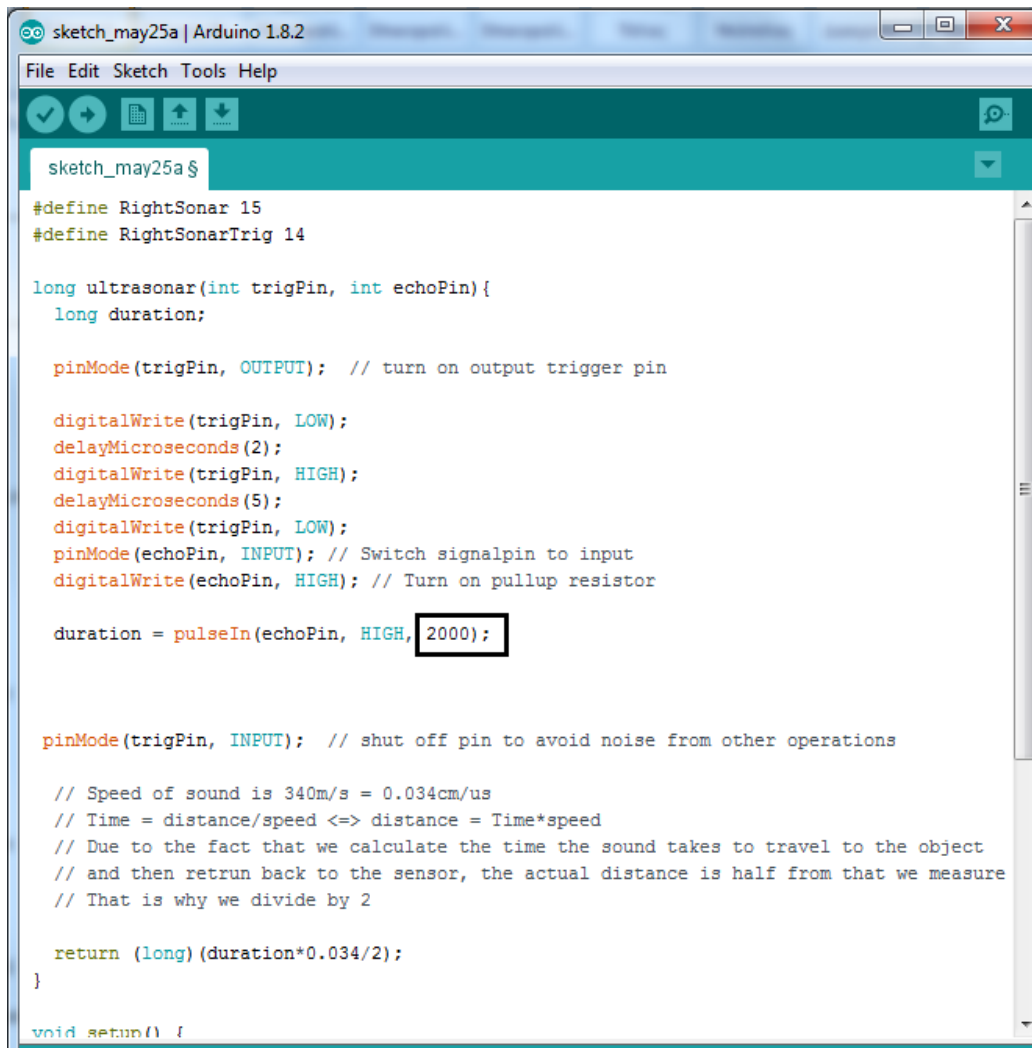
Figure 33

Once everything is ready, have student teams use an obstacle, such as a book, in front of the sensor. Ask them to move the obstacle at different distances from the sensor and observe the different indications that appear in the serial monitor. Have them experiment by varying the value of the `pulseIn()` command. In our example the value is set to 2000 (2000 micro seconds, see Fig. 34). The speed of sound is ~ 340 m/s or 0.034 cm/ μ s. So, from Newton's first law we get:

$$s = u * t$$

$$s = 0.034 * 2000 = 68 \text{ cm}$$

Due to the fact that we calculate the time the sound takes to travel to the object and then return back to the sensor, the actual distance is half from that we measure. As a result, if we use 2000 μ s, the sensor can detect obstacles at a distance of 34 cm (due to the fact that the sensors are not highly sensitive and accurate it is possible that they will detect obstacles at a distance smaller than 34 cm, so by testing you might need to reconsider the value of 2000 μ s).



```
sketch_may25a | Arduino 1.8.2
File Edit Sketch Tools Help
sketch_may25a $
#define RightSonar 15
#define RightSonarTrig 14

long ultrasonar(int trigPin, int echoPin){
  long duration;

  pinMode(trigPin, OUTPUT); // turn on output trigger pin

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT); // Switch signalpin to input
  digitalWrite(echoPin, HIGH); // Turn on pullup resistor

  duration = pulseIn(echoPin, HIGH, 2000);

  pinMode(trigPin, INPUT); // shut off pin to avoid noise from other operations

  // Speed of sound is 340m/s = 0.034cm/us
  // Time = distance/speed <=> distance = Time*speed
  // Due to the fact that we calculate the time the sound takes to travel to the object
  // and then retransmit back to the sensor, the actual distance is half from that we measure
  // That is why we divide by 2

  return (long)(duration*0.034/2);
}

void setup() {
```

Figure 34

Avoiding Obstacles

Before moving to the code that enables the robot to identify and avoid obstacles, connect and attach the second sensor to the chassis. In order to do so, follow the same process as shown in Figs. 25-29. The final result is shown in Fig. 35.

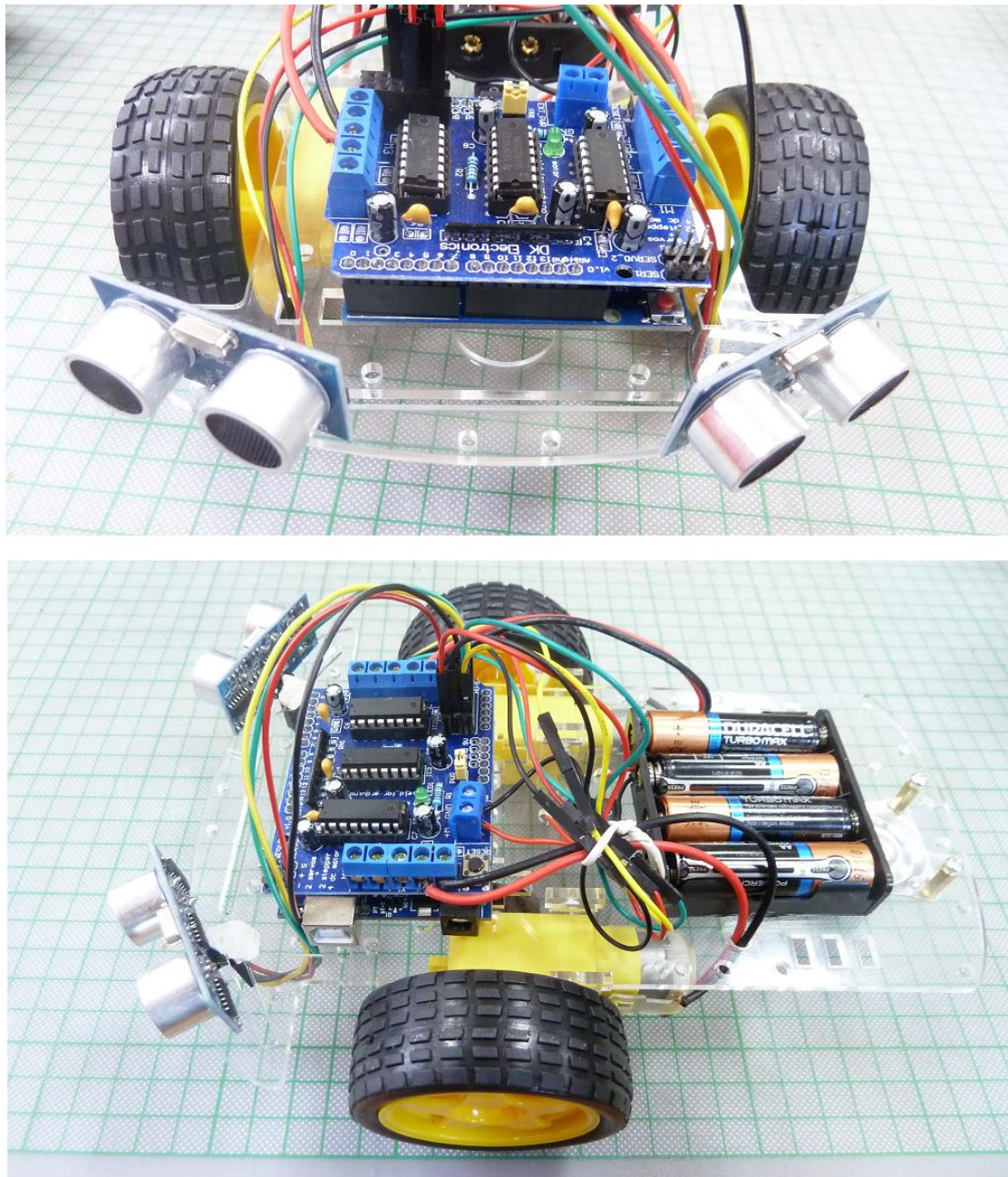


Figure 35

Activity 5 – Combine sub-solutions, test and improve

Duration: 45 minutes

Objectives: In this activity students will

- combine solutions of individual sub-problems to end up with the final design
- use their design to find the optimal settings that enable the vehicle to navigate through space and avoid obstacles
- use their design to probe whether the criteria are met or not
- make all the necessary changes to improve their design
- have fun with their design

General Context

By the end of activity 4, student teams are supposed to have finished with the construction and programming of the motors and sensors. The final step is to write the final code which would enable the vehicle to detect and avoid obstacles while moving. After finishing their design they test it in order to confirm that it is functional and meets the criteria set in previous steps. Student teams, experiment with different speeds as well as different distances that the robot can identify obstacles, writing down their predictions as well as their observations. In the case that the final design has any problem, student teams are encouraged to perform improvements and then test again their design.

❖ Working in groups

The teacher initiates a discussion about the compatibility of the different components of the final design. Student teams write the final code with the help of the teacher. As soon as the robot is ready, student teams move to a testing area with many obstacles where they can test whether the robot can detect and avoid obstacles. The teacher encourages student teams to carefully observe the behavior of the robot and try to find any flaws or mistakes in their design that if fixed will improve their design substantially.

-Tip: From an educational standpoint, it's important to allow the children to participate in setting up/cleaning up the room.

The final code

Provide each student team with the final code and explain its function.

The final code

```
#include <AFMotor.h>
#define THRESS 15

#define RightSonar 15
#define RightSonarTrig 14
#define LeftSonar 17
#define LeftSonarTrig 16

AF_DCMotor left_motor(2); // create motor #2
AF_DCMotor right_motor(3); // create motor #3

long ultrasonar(int trigPin, int echoPin){
    long duration;

    pinMode(trigPin, OUTPUT); // turn on output trigger pin

    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(5);
    digitalWrite(trigPin, LOW);
    pinMode(echoPin, INPUT); // Switch signal pin to input
    digitalWrite(echoPin, HIGH); // Turn on pull-up resistor

    duration = pulseIn(echoPin, HIGH, 2000);

    pinMode(trigPin, INPUT); // shut off pin to avoid noise from other operations

    // Speed of sound is 340m/s = 0.034cm/us
    // Time = distance/speed <=> distance = Time*speed
    // Due to the fact that we calculate the time the sound takes to travel to the
    object
    // and then retransmit back to the sensor, the actual distance is half from that we
    measure
    // That is why we divide by 2
    return (long) (duration*0.034/2);
}

void setup() {
    // put your setup code here, to run once:
    Serial.begin(57600);

    left_motor.setSpeed(255); // set the speed to 255/255
    right_motor.setSpeed(255); // set the speed to 255/255
}

void loop() {
    // put your main code here, to run repeatedly:
    long rdist;
    long ldist;

    rdist = ultrasonar(RightSonarTrig, RightSonar);
    ldist = ultrasonar(LeftSonarTrig, LeftSonar);

    if ( (rdist>0) && (rdist<THRESS) ){

        // Right sensor found an obstacle, so turn left to avoid
        right_motor.run(FORWARD); // turn it on to go forward
        left_motor.run(BACKWARD); // turn it on to go backwards

    }else if ( (ldist>0) && (ldist<THRESS) ){

        // Left sensor found an obstacle, so turn right to avoid
        right_motor.run(BACKWARD); // turn it on to go backwards
        left_motor.run(FORWARD); // turn it on to go forward

    }else{

        // No obstacle found. Go forward
        right_motor.run(FORWARD); // turn it on to go forward
        left_motor.run(FORWARD); // turn it on to go forward

    }
}
```

Using the usb cable, connect the arduino microcontroller to your pc. Double click on your arduino icon on your desktop. Open a new arduino project and copy and paste the final code. Press Upload button (Fig. 36) and your robot is ready to accomplish the tasks it is programmed for.

All you need to do is to test your final design!

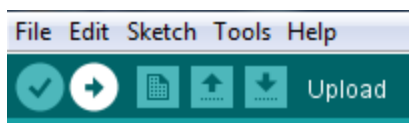


Figure 36

Important Notice

The new commands added are the following:

<pre>#define THRESS 15</pre>	<p>Define a constant that determines the distance (in cm) that when an obstacle is detected the robot will turn in order to avoid it.</p>
<pre>#define LeftSonar 17 #define LeftSonarTrig 16</pre>	<p>Define the Arduino port in which the sensor is connected.</p>
<pre>long rdist; long ldist;</pre>	<p>We declare two variables in order to store the distances in centimeters returned by the ultrasonar function for the right and left sensor accordingly</p>
<pre>rdist = ultrasonar(RightSonarTrig, RightSonar); ldist = ultrasonar(LeftSonarTrig, LeftSonar);</pre>	<p>We call the function ultrasonar for both sensors. The values of distances are stored in rdist and ldist variables</p>
<pre>if ((rdist>0) && (rdist<THRESS)){ // Right sensor found an obstacle, so turn left to avoid right_motor.run(FORWARD); // turn it on to go forward left_motor.run(BACKWARD); // turn it on to go backwards } else if ((ldist>0) && (ldist<THRESS)){ // Left sensor found an obstacle, so turn right to avoid right_motor.run(BACKWARD); // turn it on to go backwards left_motor.run(FORWARD); // turn it on to go forward } else{ // No obstacle found. Go forward right_motor.run(FORWARD); // turn it on to go forward left_motor.run(FORWARD); // turn it on to go forward } }</pre>	<p>If the right sensor detects an obstacle in a distance smaller than the threshold (defined earlier) then the robot performs an axial left turn until it no longer detects the obstacle. We also demand that the distance is >0 because the ultrasonar function returns 0 when the sensor does not detect an obstacle or when the obstacle is at a distance greater than the distance that corresponds to a time out of 2000 μs.</p> <p>The same stands for the ldist variable.</p> <p>In any other case (declared inside the else command) the robot moves forward.</p>

The main idea behind the code is that whenever the right sensor detects an obstacle at a distance smaller than the defined threshold, the robot turns left to avoid it. When the left sensor detects an obstacle at a distance smaller than the defined threshold, the robot turns right in order to avoid it. In any other case the (no obstacle detected or the obstacle is at a distance greater than the defined threshold) the robot moves forward.

Activity 6 – Present Final Solution

Duration: 20 minutes

Objectives: In this activity students will

- organize their presentation as a team
- present their team work in front of an audience

General Context

The purpose of this activity is to help students realize that they used the same process that engineers use in solving problems. Students also realize that they posed questions and investigated the science that underlies a problem and used already existing technology (tools and materials) in order to imagine, design and construct the final solution to their problem. Student teams, prepare a power point which presents the whole process they followed in order to conclude and construct the final design. Finally they present their work in front of other people.

❖ Plenary

The teacher initiates a discussion about how important it is to present your work in front of an audience. It is very important for an engineer to make a clear and comprehensible presentation to an audience who can easily be his/her employer. The teacher should point out that in order to explain something to others you must understand it in depth firstly. Have student teams to prepare a presentation where they explain what they did, how they worked and what the result was. The teacher motivates the audience to put forward questions:

- Did you find any difficulties in applying the Engineering Design Process? What difficulties did you face?
- Was the science background helpful in understanding how water rockets work?
- Did you change your original design? What affect did this/these change(s) have upon the final design?
- Do the suggested materials work properly and safely? What materials you might substitute?
- What changes did you make to your design in order to improve its performance?



- If you had more time what you would add, change, or do differently?

If you can't explain it simply, you don't understand it well enough. (Albert Einstein).

Science Carriers and Your Future

There are numerous scientific, engineering technological elements which are involved in the development of actual robots. Some of them are the following:

- Robotic technology in the healthcare industry: Distribution of medication, surgical procedures, and research
- Robotic technology in farming, and food production industry: harvesting, packaging, or distributing food
- Robotic technology in the manufacturing industry: invention and production of goods
- Robotic technology in the arts and entertainment industry: production and distribution of art, music, video and live entertainment
- Robotic technology in the field of communication: forms of communication utilize or can utilize robotic technology
- Robotics in space exploration: Remotely Operated Vehicles (ROVs) and the Remote Manipulator System (RMS), are both used in space missions. A ROV can be unmanned spacecraft that orbits freely or lands when it makes contact with an outer space surface and explore the terrain. They are used to collect data and visual footage that would never be humanly possible without the assistance of robots. RMS mechanical arms also help astronauts perform very important and difficult tasks during space missions.
- Underwater Exploration: Underwater robots have the ability to dive longer and deeper than any human, and they collect provide data of marine life. These robots are equipped with sensors, high-definition cameras, wheels and other technology to assist scientists when they explore docks, ocean floors, dams, ship bellies and other surfaces.
- Investigating Hazardous Environments: robots have become increasingly important for investigating and researching hazardous and dangerous environments, such as volcanoes, rooms on fire, areas with explosives, earthquake ruins and caves.
- Robots used to assist people with special needs.



List of Materials

<p><u>2 x Distance Measuring Transducer sensors</u> (ElecFreaks HC-SR04 Ultrasonic Module-Distance Measuring Transducer sensor For Arduino) https://www.amazon.com/SainSmart-HC-SR04-Ranging-Detector-Distance/dp/B004U8TOE6/ref=sr_1_6?s=electronics&ie=UTF8&qid=1495620549&sr=1-6&keywords=HC-SR04)</p>	
<p><u>1 x 1 x Arduino uno</u> (Arduino Uno R3 Microcontroller A000066) https://www.amazon.com/Arduino-Uno-R3-Microcontroller-A000066/dp/B008GR7SV6/ref=sr_1_2?s=toys-and-games&ie=UTF8&qid=1495620516&sr=1-2&keywords=arduino+uno)</p>	
<p><u>1 x Robot chassis</u> (DIY 2WD Smart Robot Car Chassis Kit for Arduino) https://www.amazon.com/DIY-Smart-Robot-Chassis-Arduino/dp/B01N7KJIW4)</p>	
<p><u>1 x L293D motor driver shield</u> (Qunqi L293D Motor Drive Shield For Arduino Duemilanove Mega UNO R3 AVR ATMEL) https://www.amazon.com/Qunqi-L293D-Shield-Arduino-Duemilanove/dp/B014KN2898/ref=sr_1_1?s=electronics&ie=UTF8&qid=1495620706&sr=1-1&keywords=L293D+Motor+Driver+Shield)</p>	
<p><u>3 x six pin female headers</u></p>	
<p><u>8 x male to female jumper wires (20 cm length)</u></p>	
<p><u>4 x AA batteries</u></p>	
<p><u>Double sided foam tape</u></p>	
<p><u>1 x USB AM-BM Printer Cable</u></p>	

References

- [1]. Henry Samueli School of Engineering and Applied Science, (2017). What engineers do. UCLA engineering. Available at:
<http://engineering.ucla.edu/descriptions-of-majors-offered/>
- [2]. College Factual. (2017). Engineering Overview. Available at:
<http://www.collegefactual.com/majors/engineering/>,
<http://www.umich.edu/~ptclab/>

