LEGO® MINDSTORMS® Education EV3

Coding Activities

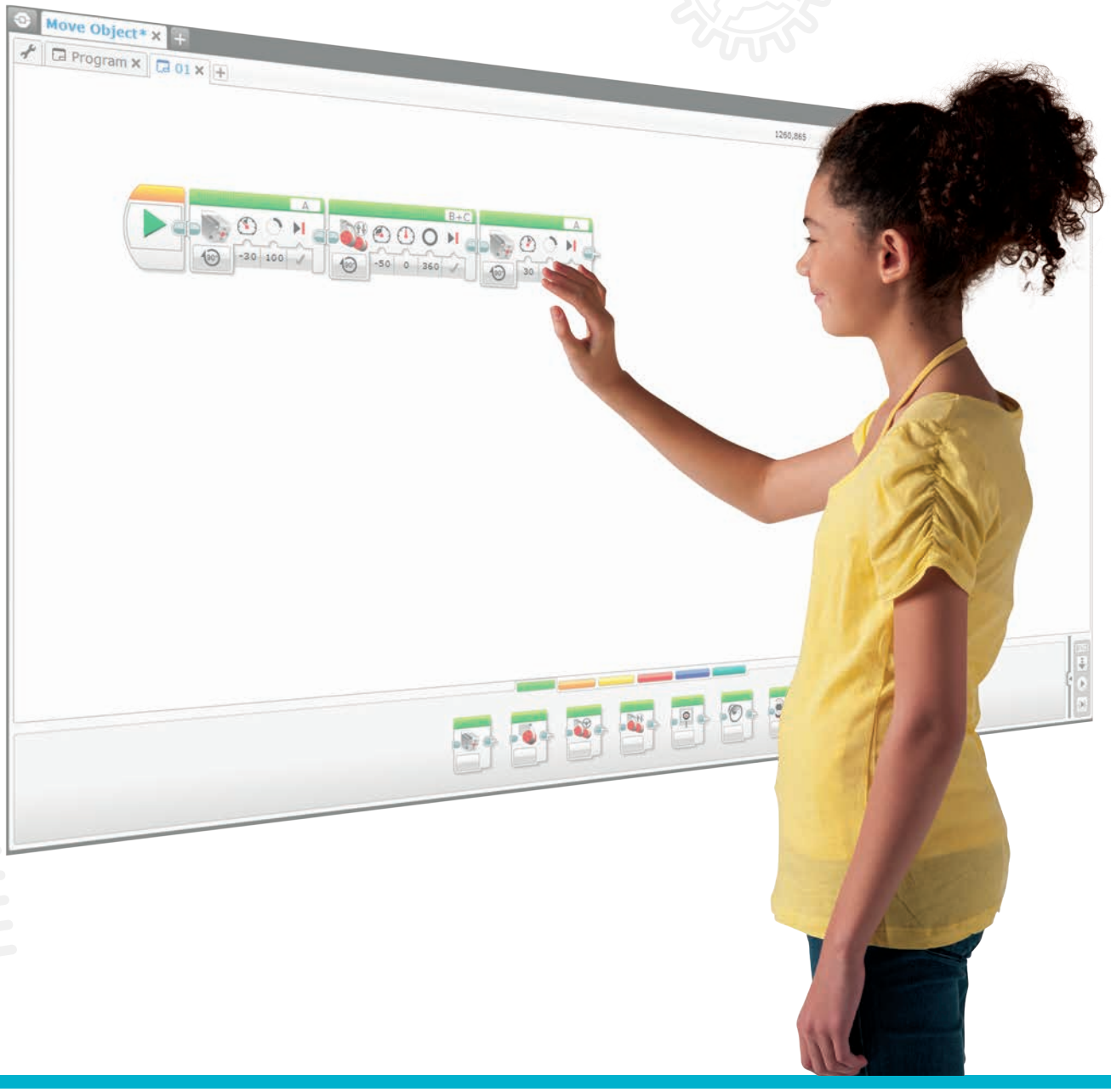
LEGOeducation.com.au

LEGO mindstorms education EV3

Contents

Introduction

LEGO® MINDSTORMS® Education EV3 Coding Activities



LEGO mindstorms education EV3

Introduction

LEGO® MINDSTORMS® Education EV3 Coding Activities

The LEGO® Education team is pleased to present this set of LEGO® MINDSTORMS® Education EV3 Coding Activities for use in grades 5-9. These materials will help teachers facilitate exciting projects designed around relevant technology topics, which will enable students to learn fundamental computing and engineering concepts using a real-world context.

– Background and curriculum links
– Target Group
– Activity Overview
– Organisation of the Activities
– EV3 Programming App
– Robot Educator
– Real Life References
– Text-Based Programming

BACKGROUND AND CURRICULUM LINKS

In 2016-2017, Australian schools implemented the Australian Curriculum: Technologies. The Technologies learning area comprises two subjects: Digital Technologies and Design and Technologies. In Design and Technologies students use design thinking to generate and produce solutions. In Digital Technologies students use computational and systems thinking to define, design and implement digital solutions.

With this in mind, LEGO Education has produced this set of activities for Years 5-9 using MINDSTORMS® Education EV3 to help students tackle these subjects. The guide consists of 12 activities, which, depending on how they are organised, could equate to around 36 hours of classroom-based activity.

Introduction

CURRICULUM GRID

	Technologies Content Descriptors **Years 5-6** ● = addresses Content Descriptor ◐ = partially addresses Content Descriptor	**Activity 1:** Performing a Three Point Turn	**Activity 2:** Written Instructions For a Three Point Turn	**Activity 3:** Reversing the Robot	**Activity 4:** Light the Way	**Activity 5:** Traffic Lights and Automated Navigation	**Activity 6:** Reversing Beeps	**Activity 7:** Keyless Starting of a Car	**Activity 8:** Cruise Control	**Activity 9:** Roaming Robots	**Activity 10:** Final Project – Designing a Driverless, Automated, Wheeled Robot	**Activity 11:** Final Project - Building and Programming a Driverless, Automated, Wheeled Robot	**Activity 12:** Final Project - Reviewing, Revising and Presenting Your Driverless, Automated, Wheeled Robot
Digital Technologies													
Knowledge and Understanding													
Digital Systems													
ACTDIK014	Examine the main components of common digital systems and how they may connect together to form networks to transmit data	●	●	●	●	●	●	●	●	●	●	●	●
Processes and production skills													
Creating digital solutions by:													
Investigating and defining													
ACTDIP017	Define problems in terms of data and functional requirements drawing on previously solved problems	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐
Generating and designing													
ACTDIP019	Design, modify and follow simple algorithms involving sequences of steps, branching, and iteration (repetition)	●	●	●	●	●	●	●	●	●	●	●	●
Producing and implementing													
ACTDIP020	Implement digital solutions as simple visual programs involving branching, iteration (repetition), and user input	●	●	●	●	●	●	●	●	◐	●	●	●
Evaluating													
ACTDIP021	Explain how student solutions and existing information systems are sustainable and meet current and future local community needs	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐
Collaborating and managing													
ACTDIP022	Plan, create and communicate ideas and information, including collaboratively online, applying agreed ethical, social and technical protocols	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●
Design and Technologies													
Knowledge and Understanding													
Technologies and society													
ACTDEK019	Examine how people in design and technologies occupations address competing considerations, including sustainability in the design of products, services, and environments for current and future use	●	●	●	●	●	●	◐	◐	◐	●	●	●
Technologies contexts													
Engineering principles and systems													
ACTDEK020	Investigate how electrical energy can control movement, sound or light in a designed product or system	●	●	●	●	●	●	●	●	●	●	●	●

Introduction

CURRICULUM GRID

	Technologies Content Descriptors **Years 5-6** ● = addresses Content Descriptor ◐ = partially addresses Content Descriptor	**Activity 1:** Performing a Three Point Turn	**Activity 2:** Written Instructions For a Three Point Turn	**Activity 3:** Reversing the Robot	**Activity 4:** Light the Way	**Activity 5:** Traffic Lights and Automated Navigation	**Activity 6:** Reversing Beeps	**Activity 7:** Keyless Starting of a Car	**Activity 8:** Cruise Control	**Activity 9:** Roaming Robots	**Activity 10:** Final Project – Designing a Driverless, Automated, Wheeled Robot	**Activity 11:** Final Project - Building and Programming a Driverless, Automated, Wheeled Robot	**Activity 12:** Final Project - Reviewing, Revising and Presenting Your Driverless, Automated, Wheeled Robot
Engineering principles and systems													
ACTDEK023	Investigate characteristics and properties of a range of materials, systems, components, tools and equipment and evaluate the impact of their use	●	●	◐	◐	●	◐	●	◐	◐	●	●	●
Processes and production skills													
Investigating and defining													
ACTDEP024	Critique needs or opportunities for designing, and investigate materials, components, tools, equipment and processes to achieve intended designed solutions	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●
Generating and designing													
ACTDEP025	Generate, develop and communicate design ideas and processes for audiences using appropriate technical terms and graphical representation techniques	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●
Producing and implementing													
ACTDEP026	Select appropriate materials, components, tools, equipment and techniques and apply safe procedures to make designed solutions	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐
Evaluating													
ACTDEP027	Negotiate criteria for success that include sustainability to evaluate design ideas, processes and solutions	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●
Collaborating and managing													
ACTDEP028	Develop project plans that include consideration of resources when making designed solutions individually and collaboratively	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●

www.australiancurriculum.edu.au/technologies

Introduction

CURRICULUM GRID

	Technologies Content Descriptors **Years 7-8** ● = addresses Content Descriptor ◐ = partially addresses Content Descriptor	**Activity 1:** Performing a Three Point Turn	**Activity 2:** Written Instructions For a Three Point Turn	**Activity 3:** Reversing the Robot	**Activity 4:** Light the Way	**Activity 5:** Traffic Lights and Automated Navigation	**Activity 6:** Reversing Beeps	**Activity 7:** Keyless Starting of a Car	**Activity 8:** Cruise Control	**Activity 9:** Roaming Robots	**Activity 10:** Final Project - Designing a Driverless, Automated, Wheeled Robot	**Activity 11:** Final Project - Building and Programming a Driverless, Automated, Wheeled Robot	**Activity 12:** Final Project - Reviewing, Revising and Presenting Your Driverless, Automated, Wheeled Robot
Digital Technologies													
Processes and production skills													
Investigating and defining													
ACTDIP027	Define and decompose real-world problems taking into account functional requirements and economic, environmental, social, technical and usability constraints	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐
Investigating and defining													
ACTDIP029	Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors	●	●	●	●	●	●	●	●	●	●	●	●
Producing and implementing													
ACTDIP030	Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language	●	●	●	●	●	●	●	●	●	●	●	●
Evaluating													
ACTDIP031	Evaluate how student solutions and existing information systems meet needs, are innovative, and take account of future risks and sustainability	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐
Collaborating and managing													
ACTDIP032	Plan and manage projects that create and communicate ideas and information collaboratively online, taking safety and social contexts into account	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●
Design and Technologies													
Knowledge and Understanding													
Technologies contexts													
Engineering principles and systems													
ACTDEK031	Analyse how motion, force and energy are used to manipulate and control electromechanical systems when designing simple, engineered solutions	●	●	●	●	●	●	●	●	●	●	●	●
Materials and technologies specialisation													
ACTDEK034	Analyse ways to produce designed solutions through selecting and combining characteristics and properties of materials, systems, components, tools and equipment	●	●	◐	◐	●	◐	●	◐	◐	●	●	●
Processes and production skills													
Investigating and defining													
ACTDEP035	Critique needs or opportunities for designing and investigate, analyse and select from a range of materials, components, tools, equipment and processes to develop design ideas	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●

Introduction

CURRICULUM GRID

	Technologies Content Descriptors **Years 7-8** ● = addresses Content Descriptor ◐ = partially addresses Content Descriptor	**Activity 1:** Performing a Three Point Turn	**Activity 2:** Written Instructions For a Three Point Turn	**Activity 3:** Reversing the Robot	**Activity 4:** Light the Way	**Activity 5:** Traffic Lights and Automated Navigation	**Activity 6:** Reversing Beeps	**Activity 7:** Keyless Starting of a Car	**Activity 8:** Cruise Control	**Activity 9:** Roaming Robots	**Activity 10:** Final Project – Designing a Driverless, Automated, Wheeled Robot	**Activity 11:** Final Project - Building and Programming a Driverless, Automated, Wheeled Robot	**Activity 12:** Final Project - Reviewing, Revising and Presenting Your Driverless, Automated, Wheeled Robot
Generating and designing													
ACTDEP036	Generate, develop, test and communicate design ideas, plans and processes for various audiences using appropriate technical terms and technologies including graphical representation techniques	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●
Producing and implementing													
ACTDEP037	Select and justify choices of materials, components, tools, equipment and techniques to effectively and safely make designed solutions	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐
Evaluating													
ACTDEP038	Independently develop criteria for success to evaluate design ideas, processes and solutions and their sustainability	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●
Collaborating and managing													
ACTDEP039	Use project management processes when working individually and collaboratively to coordinate production of designed solutions	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●

Introduction

CURRICULUM GRID

	Technologies Content Descriptors **Years 9-10** ● = addresses Content Descriptor ◐ = partially addresses Content Descriptor	**Activity 1:** Performing a Three Point Turn	**Activity 2:** Written Instructions For a Three Point Turn	**Activity 3:** Reversing the Robot	**Activity 4:** Light the Way	**Activity 5:** Traffic Lights and Automated Navigation	**Activity 6:** Reversing Beeps	**Activity 7:** Keyless Starting of a Car	**Activity 8:** Cruise Control	**Activity 9:** Roaming Robots	**Activity 10:** Final Project – Designing a Driverless, Automated, Wheeled Robot	**Activity 11:** Final Project - Building and Programming a Driverless, Automated, Wheeled Robot	**Activity 12:** Final Project - Reviewing, Revising and Presenting Your Driverless, Automated, Wheeled Robot
Digital Technologies													
Knowledge and Understanding													
Digital Systems													
ACTDIK034	Investigate the role of hardware and software in managing, controlling and securing the movement of and access to data in networked digital systems	●	●	●	●	●	●	●	●	●	●	●	●
Processes and production skills													
Creating digital solutions by:													
Investigating and defining													
ACTDIP038	Define and decompose real-world problems precisely, taking into account functional and non-functional requirements	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐
Generating and designing													
ACTDIP040	Design algorithms represented diagrammatically and in structured English and validate algorithms and programs through tracing and test cases	●	●	●	●	●	●	●	●	●	●	●	●
Producing and implementing													
ACTDIP041	Implement modular programs, applying selected algorithms and data structures including using an object-oriented programming language	●	●	●	●	●	●	●	●	●	●	●	●
Collaborating and managing													
ACTDIP044	Plan and manage projects using an iterative and collaborative approach, identifying risks and considering safety and sustainability	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●
Design and Technologies													
Knowledge and Understanding													
Technologies and society													
ACTDEK041	Explain how products, services and environments evolve with consideration of preferred futures and the impact of emerging technologies on design decisions	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐
Technologies contexts													
Engineering principles and systems													
ACTDEK043	Investigate and make judgments on how the characteristics and properties of materials are combined with force, motion and energy to create engineered solutions	●	●	●	●	●	●	●	●	●	●	●	●

Introduction

CURRICULUM GRID

	Technologies Content Descriptors **Years 9-10** ● = addresses Content Descriptor ◐ = partially addresses Content Descriptor	**Activity 1:** Performing a Three Point Turn	**Activity 2:** Written Instructions For a Three Point Turn	**Activity 3:** Reversing the Robot	**Activity 4:** Light the Way	**Activity 5:** Traffic Lights and Automated Navigation	**Activity 6:** Reversing Beeps	**Activity 7:** Keyless Starting of a Car	**Activity 8:** Cruise Control	**Activity 9:** Roaming Robots	**Activity 10:** Final Project – Designing a Driverless, Automated, Wheeled Robot	**Activity 11:** Final Project - Building and Programming a Driverless, Automated, Wheeled Robot	**Activity 12:** Final Project - Reviewing, Revising and Presenting Your Driverless, Automated, Wheeled Robot
Materials and technologies specialisation													
ACTDEK046	Investigate and make judgments on how the characteristics and properties of materials, systems, components, tools and equipment can be combined to create designed solutions	●	●	◐	◐	●	◐	●	◐	◐	●	●	●
ACTDEK047	Investigate and make judgments, within a range of technologies specialisations, on how technologies can be combined to create designed solutions	●	●	◐	◐	●	◐	●	◐	◐	●	●	●
Processes and production skills													
Investigating and defining													
ACTDEP048	Critique needs or opportunities to develop design briefs and investigate and select an increasingly sophisticated range of materials, systems, components, tools and equipment to develop design ideas	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●
Generating and designing													
ACTDEP049	Develop, modify and communicate design ideas by applying design thinking, creativity, innovation and enterprise skills of increasing sophistication	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●
Producing and implementing													
ACTDEP050	Work flexibly to effectively and safely test, select, justify and use appropriate technologies and processes to make designed solutions	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐
Evaluating													
ACTDEP051	Evaluate design ideas, processes and solutions against comprehensive criteria for success recognising the need for sustainability	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●
Collaborating and managing													
ACTDEP052	Develop project plans using digital technologies to plan and manage projects individually and collaboratively taking into consideration time, cost, risk and production processes	◐	◐	◐	◐	◐	◐	◐	◐	◐	●	●	●

Introduction

TARGET GROUP

These activities will help teachers to inspire their students to think about how important computer programming is in our everyday lives. The students will gain experience in programming through a mixture of direct teaching, experimentation and exploration, and tutorials from the LEGO® MINDSTORMS® Education EV3 Software. Example programming solutions are provided along with this set of activities. The material is aimed at Years 5-9, but can be easily adapted for older students. While this material has primarily been written to address the Technologies curriculum, there are ample cross-curricular opportunities that will have an impact on other subject areas, most notably science and mathematics.

The lesson plans in this set of activities will greatly reduce planning time, and will help teachers to produce further plans as their experience grows.

At the end of each activity there are full page examples of the pictures and programs that you will find in the lesson plans. These can be used as handouts.

Enjoy!

ACTIVITY OVERVIEW

ACTIVITY	TASK	LEGO MINDSTORMS EDUCATION PROGRESSION	EV3 SOFTWARE BLOCKS COVERED
1	– Introduction to Computing – EV3 hardware and software – Simple driving and turning	– Use two or more programming languages to solve a variety of computational problems	– Move Tank – Wait – Ultrasonic Sensor – Sound
2	– Text-based versus visual programming – Write text-based programs based on Activity 1	– Use two or more programming languages to solve a variety of computational problems	– Move Tank – Wait – Ultrasonic Sensor – Sound
3	– Using the Display and Brick Status Light Blocks – Warning symbols on cars	– Understand that algorithms are capable of carrying out a series of instructions in order – Use the Move Steering Block to make a wheeled robot travel in a straight line. – Use the Wait Block in relation to Touch Sensor(s) – Utilise the Brick Status Light and display functions – Extend computational thinking through the creation of more complex algorithms	– Steering – Wait – Touch Sensor

Introduction

ACTIVITY	TASK	LEGO® MINDSTORMS® EDUCATION PROGRESSION	EV3 SOFTWARE BLOCKS COVERED
4	– Colour Sensor – Ambient light settings – Automatic headlights	– Understand several key algorithms that reflect computational thinking – Understand simple Boolean logic (such as AND, OR and NOT) and some of its uses in circuits and programming	– Wait – Colour Sensor – Display – Time – Loop – Touch Sensor – Loop Interrupt
5	– Line following – Automated car	– Understand that algorithms are capable of carrying out a series of instructions in order – Extend understanding of Boolean logic and its uses – Use the Wait Block in relation to the Colour Sensor – Understand that the Colour Sensor has several functions and can measure a range of parameters – Extend the use of the Colour Sensor to recognise LEGO system colours and reflected light intensity – Extend understanding of the Loop Block – Understand the concept of a switch and how to use it for 'true' and 'false' operations	– Wait – Move Steering – Colour Sensor – Loop – Switch – Loop Interrupt
6	– Ultrasonic Sensor – Object detection tool – Car reversing sensors	– Understand that algorithms are capable of carrying out a series of instructions in order – Extend understanding of Boolean logic and its uses – Use the Wait Block in relation to the Colour Sensor – Understand that the Ultrasonic Sensor works by 'bouncing' waves off objects and that it can be programmed to respond to distance – Program the wheeled robot to reverse, emit sound based on distance from an object and stop at a given distance from that object – Extend understanding of the Loop Block – Understand the concept of a switch and how to use it for 'true' and 'false' commands – Understand the Maths Block and its functions – Understand that readings can be taken from one block and sent to another through the use of Data Wires	– Move Steering – Wait – Ultrasonic Sensor – Loop – Math – Sound

Introduction

ACTIVITY	TASK	LEGO® MINDSTORMS® EDUCATION PROGRESSION	EV3 SOFTWARE BLOCKS COVERED
7	– Keyless starting of a car	– Understand several key algorithms that reflect computational thinking – Understand simple Boolean logic (such as AND, OR and NOT) and some of its uses in circuits and programming – Use the Logic Block in conjunction with the Switch Block – Use several sensors in combination to activate a program on the EV3 Brick	– Wait – Touch Sensor – Display – Wait – Time – Ultrasonic Sensor – Brick Buttons – Logic – Switch – Loop – Move Steering
8	– Design cruise control	– Understand several key algorithms that reflect computational thinking – Use the Variable Block to store information – Develop multi-level programs – Create My Blocks	– Wait – Touch Sensor – Loop – Switch – Variable – Math – Move Steering – My Blocks
9	– Arrays: how do they work?	– Make appropriate use of data structures such as lists, tables and arrays – Understand simple Boolean logic (such as AND, OR and NOT) and some of its uses in circuits and programming – Use the brick buttons to control the movement of the wheeled robot – Use the Variable Block to store information – Use the Array Operations Block	– Variable – Wait – Brick Buttons – Loop – Sound – Array Operations – Time – Move Steering – My Blocks
FINAL PROJECT			
10	– Design a driverless, automated, wheeled robot	– Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems	– Any programming blocks can be used from previous weeks
11	– Build and program a driverless car	– Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems	– Any programming blocks can be used from previous weeks
12	– Evaluate the design, make changes and conclude the design effectiveness	– Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems	– Any programming blocks can be used from previous weeks

Introduction

ORGANISATION OF THE ACTIVITIES

There are usually three main tasks to carry out in each activity. Possible solutions to these tasks are contained within the lesson plans and the EV3 programs that are available to download. The activities should include any building that needs to be done. Typically, this is the Robot Educator model. It is a good idea to give the students plenty of building experience during activities 1 to 9, so that they are well prepared to undertake the final project (responding to a design brief and following the engineering process).

Each activity should include a discussion phase with the student about the structure / design of their program.

EV3 PROGRAMMING APP

Many of the challenges found within this set of LEGO® MINDSTORMS® Education EV3 Coding Activities can be solved with the EV3 Programming App, which is an app with basic programming capabilities. The following table indicates which student challenges can be solved using the example code provided with this document. With some creative problem solving, the students can adapt algorithms to expand the total number of challenges solved with the EV3 Programming App.

Activity	1			2			3			4			5			6			7			8			9	
Challenge	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2
EV3 Programming App	x	x	x				x	x	x	x	x		x	x	x	x	x		x							

ROBOT EDUCATOR

Each activity will require the students to have a good understanding of the EV3 Software. This can be achieved by having them complete the Robot Educator tutorials named at the end of each activity plan (under 'Teacher Notes'). This, supported by direct teaching and exploration, will ensure that the students gain the necessary skills and understanding to carry out the challenges set in each activity.

As the activities progress, the Robot Educator tutorials section will be divided into 'New Robot Educator Tutorials' and previously covered tutorials that remain relevant to the current activity.

Introduction

REAL LIFE REFERENCES

In this set of activities, you will find no direct links to search engines or video hosting sites. This is because, over time, links can become obsolete. Also, in certain countries video hosting websites are restricted and in some cases blocked altogether.

To overcome these restrictions, we have provided a number of keywords that you can enter into your preferred search engine in order to find appropriate content.

For example, to start this set of activities we recommend watching an informative video about BMW's driverless car, which was originally broadcast on a BBC television programme called 'Top Gear'. To access this video, type the following into your preferred search engine or video hosting site:

Keywords: **BMW driverless car top gear**

These five words will find the video clip and you can watch a driverless car in action.

This video is not the only one to be found. Driverless cars are a very exciting modern development. Take the time to search and you will find many great online resources for your students.

TEXT-BASED PROGRAMMING

For this set of activities, LEGO® Education has opted to use ROBOTC as its text-based language.

This set of activities is not intended to teach teachers and students how to use a text-based language, but rather how EV3 programs are the visual equivalent to a text-based solution. For everything you need to know about ROBOTC, go to: http://www.robotc.net/

Where possible, the ROBOTC programs will emulate the EV3 programs exactly. However, due to the nature of using two different programming languages there will be slight inherent differences. Any differences that have been identified will be highlighted in the relevant ROBOTC program (using green text).

Activity 1
Performing a Three Point Turn

Activity 1

Performing a Three Point Turn

INTRODUCTION TO COMPUTING, EV3 AND MINDSTORMS®

During this activity, the students will be introduced to computing and the concept of programming. In this and in subsequent activities, the students will learn the importance of programming in everyday life.

Each activity builds on previous knowledge and experience and will equip the students to develop a driverless, automated car:
– a wheeled robot that can get from point A to point B without needing a driver.

In order to contextualise the activities, you may wish to encourage discussion amongst the students. Questions such as:

- Which features do the students think are needed for such a car? Discuss features such as sensors for safety and navigation.
- What other things can the students think of?
- Which automated systems are on their own parents' cars? These could be features such as proximity sensors to detect obstacles, horns/sirens to warn pedestrians, etc.

NB: You could plan a visit to a car factory or dealership nearby to learn more about the automated features on some vehicles.

INTRODUCE THE EV3 AND EV3 SOFTWARE

Time must be spent here on building the Robot Educator base model.

This lesson will be mainly concerned with using motors and writing an algorithm to perform a three point turn.

Explain that three point turns are used by cars to make a 180 degree turn.

You may find online videos displaying three point turns / turning in the road, which may be useful for the students to use as reference material.

Activity 1
Lesson Plan

LESSON PLAN

There are three main tasks to be carried out in this lesson. The solutions to these are contained within the lesson plan / resources section.

OUTCOMES

Students will be able to:

- understand that algorithms are capable of carrying out a series of instructions in order
- understand how the Move Tank Block works to steer their wheeled robot
- build and program a simple wheeled robot using Move Tank Blocks and timing to carry out a 180 degree turn using the three point turn method
- be introduced to the 'wait' command and the Ultrasonic Sensor Block

VOCABULARY

input, output, algorithm, wait, Ultrasonic Sensor, debug

INTRODUCTION

- Explain to the students that during the course of the lesson, they will construct and program a wheeled robot to carry out a three point turn, but that first they will have some time to explore the EV3 Software to make their models move.
- Show the students a complete EV3 base model and tell them that they are going to use the built-in instructions from the EV3 Software to construct the same model. This can be found in the Robot Educator section of the software.
- Demonstrate to the students how to access the EV3 Software and how to use the Move Tank Block to make their wheeled robot move.

Image 1

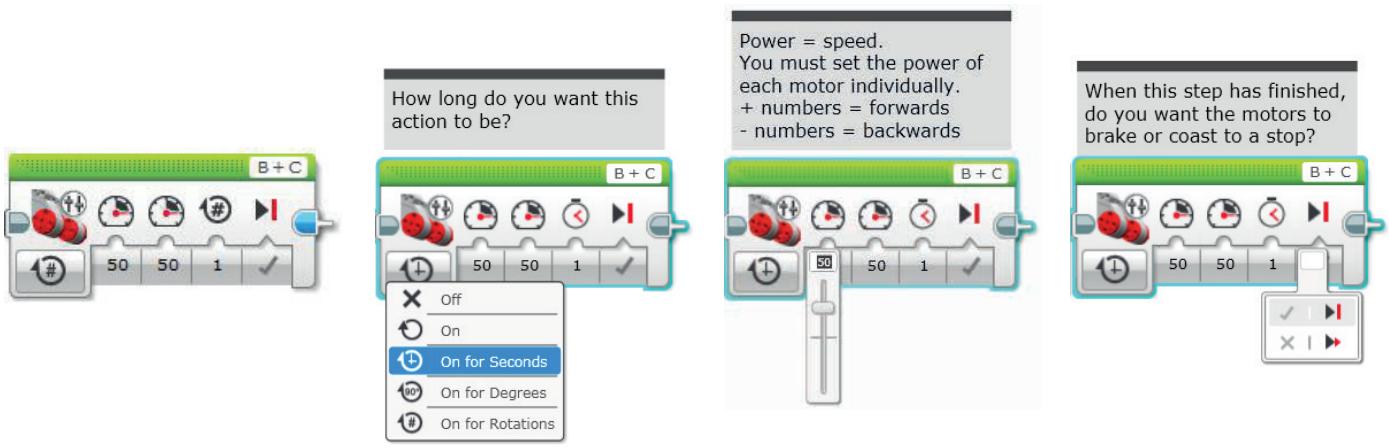
Image 2


Image 3

Activity 1
Lesson Plan

INTRODUCTION: EXPLORATION TASK

- The students will work in pairs or threes to construct the base model.
- Once made, they will begin a new project and experiment with the Move Tank Block.
- Encourage the students to explore different ways of turning the wheeled robot and the effect that altering the power on each motor has.

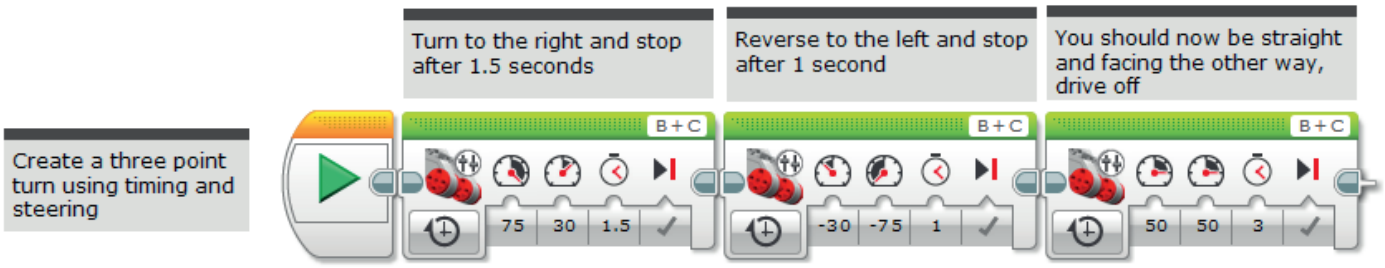


MAIN CHALLENGE 1

- Get some feedback from the students about the different methods they have used to make their wheeled robot turn using the Move Tank Block. Ask them to explain how they arrived at that point.
- Explain to the students that they will need to use what they have discovered about turning the model to simulate a three point turn.
- Consider showing the students an online video of how to perform a three point turn.
- The students will then use the EV3 Software to create a three point turn program using different power settings in each motor to steer the robot.
- Encourage the students to constantly evaluate and debug their program in order to ensure that their wheeled robot turns 180 degrees.
- NB: You may wish to 'tape out' road markings to give the students a constrained area in which to turn their wheeled robot.
- NB: You will see in the possible solution programs that the wheeled robot 'should be straight' before the final stage and driving off. It is worth noting that in a real situation this is not always the case, and that students could include a curve here before the straight line. Encourage the students to appreciate that there is no wrong way of solving these challenges.

Activity 1
Lesson Plan

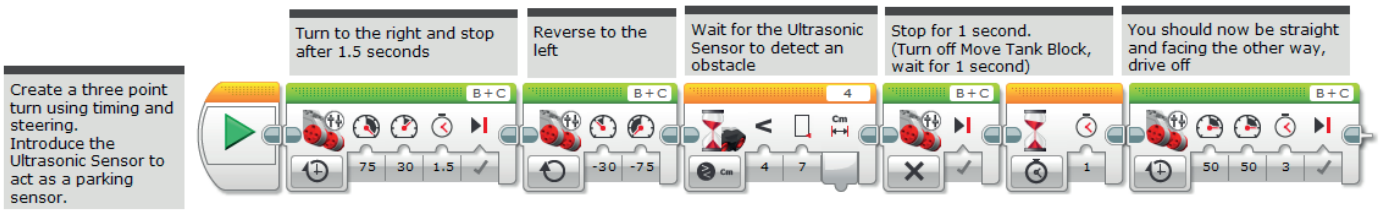
POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 1
TAB: MAIN 1



MAIN CHALLENGE 2

- Introduce the simple use of the Ultrasonic Sensor.
- Ask the students how we could make the program more autonomous and be able to account for any obstacles that might appear while the wheeled robot is reversing.
- Demonstrate the Wait Block and how to use it with the Ultrasonic Sensor.
- The students will extend their programming to ensure that the wheeled robot stops at a given point in reaction to the Ultrasonic Sensor.
- Point out that their programs will simulate a driver applying the brakes when reversing close to an obstacle.

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 1
TAB: MAIN 2

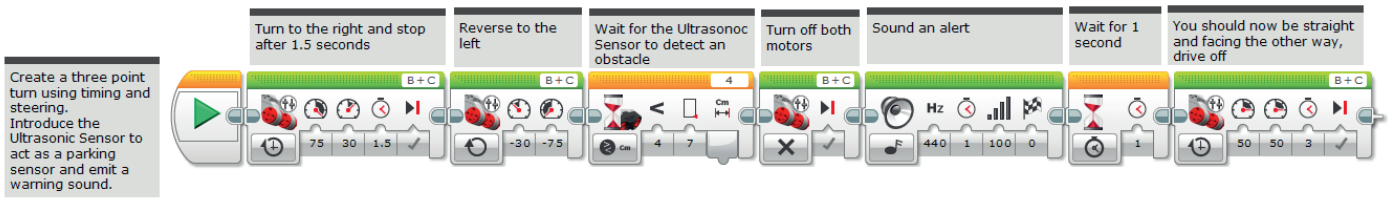


Activity 1
Lesson Plan

MAIN CHALLENGE 3

- Use the Ultrasonic Sensor to add safety features (such as warning sounds) to the wheeled robot.
- Ask the students what happens in a car when it is reversing and it approaches an obstacle – a warning sound can be heard.
- Introduce the Sound Block to add a warning sound after the wheeled robot has stopped. Demonstrate to the students how the Sound Block works and where it should appear in their program.
- The students must adapt their existing program to incorporate the Sound Block to emit a warning sound when the wheeled robot is a certain distance from an obstacle.
- Encourage the students to constantly debug their programs so that the sound and stopping distance work effectively.

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 1
TAB: MAIN 3



CLASS DISCUSSION

- Recap on what the students have learnt in this activity.
- Make sure that the students use and understand the vocabulary used throughout the activity - recap on what the key words mean.
- Ask one or two groups to demonstrate their programs. Discuss what works well and what could be improved.
- Tell the students that next week they will be carrying out the same tasks, but instead of using the EV3 Software, they will be using a text-based programming solution.

Activity 1
Student Worksheets

CHALLENGES FOR TODAY

Today is designed to introduce you to and get you started with the LEGO® MINDSTORMS® Education EV3 Software.

You will already have had some time to experiment with the Move Tank Block to get your wheeled robot moving around the room. Now you will need to hone those skills in order to carry out three challenges.

Good luck!

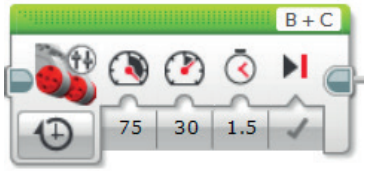
CHALLENGE 1

Program your wheeled robot to perform a three point turn.

You will need to turn your wheeled robot while going forwards, then reverse it before driving forwards again.

Watch the online video clip again to remind you of what it looks like, and make that sure you don't cross the road markings!

Blocks to Consider



Plan your program first. Write it in pseudo-code below:

Activity 1
Student Worksheets

CHALLENGE 2

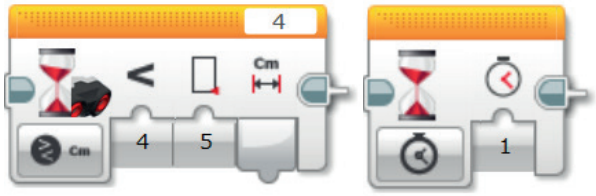
You are now going to experiment with one of the EV3 sensors – the Ultrasonic Sensor.

Program your wheeled robot to perform a three point turn and use the Ultrasonic Sensor as a 'reversing parking sensor' so that your wheeled robot stops at a given distance from an obstacle when it is reversing.

Can your wheeled robot 'put the brakes on' before it drives off again?

You will need to use your knowledge of the Wait Block here, and attach the Ultrasonic Sensor to the rear of your wheeled robot.

Blocks to Consider



Plan your program first. Write it in pseudo-code below:

Activity 1
Student Worksheets

CHALLENGE 3

You will now simulate warning sounds.

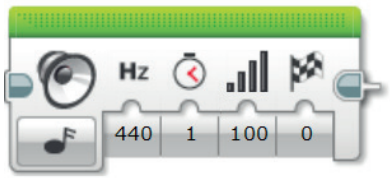
What often happens when a car is reversing and approaches an obstacle?

Now that your wheeled robot stops in response to the Ultrasonic 'parking sensor', can you extend your program so that your wheeled robot emits a warning sound just after the brakes are applied when reversing?

You will need to constantly debug your program so that the warning sound stops at the same time as your wheeled robot. Which parts of your program will need to change?

Blocks to Consider

Use the same blocks that you used in programming tasks 1 and 2, but also consider using the following:



Plan your program first. Write it in pseudo-code below:

Activity 1
Student Worksheets

After a programming activity it is important to note down your thoughts and observations.

Consider the following points and then in the box below record how the activity went.

- How could you improve your program?
- Could your program have been more streamlined? Have you used too many blocks? Is there a more efficient way of building your program?
- What examples of real-world applications could you see your program being used in?

Thoughts and Observations

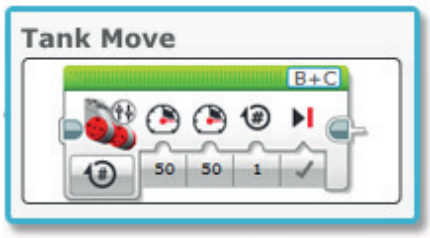
Activity 1
Teacher Notes

ROBOT EDUCATOR TUTORIALS

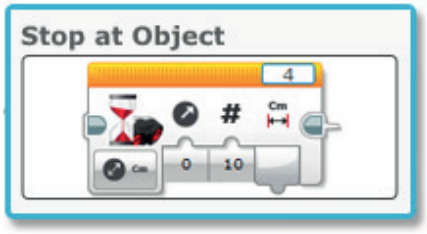
The following Robot Educator Tutorials will help teachers and their students to solve the challenges.

NEW ROBOT EDUCATOR TUTORIALS

Basics > Tank Move



Basics > Stop at Object



Activity 1
Appendix

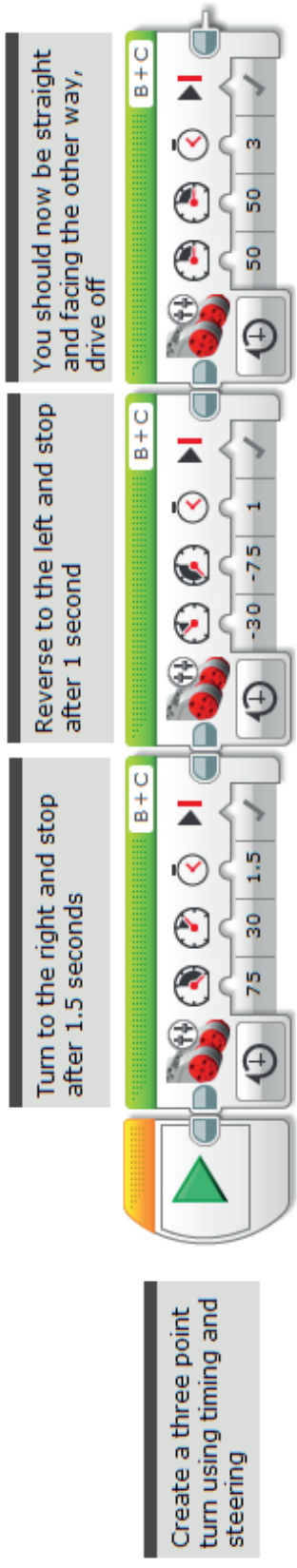
APPENDIX FOR ACTIVITY 1: LARGE IMAGES AND PROGRAMS

Activity 1
Appendix

Activity 1
Appendix

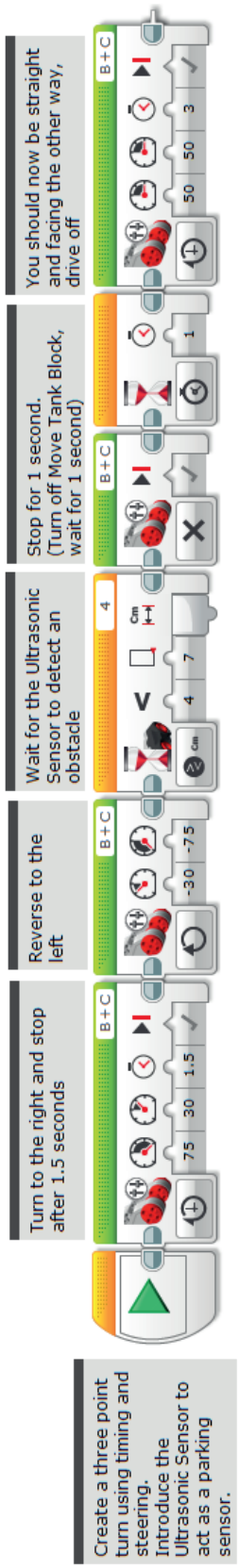
Activity 1
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 1
TAB: MAIN 1



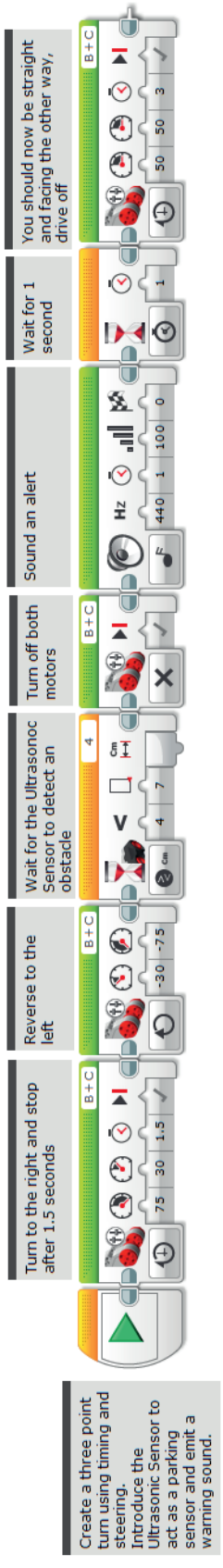
Activity 1
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 1
TAB: MAIN 2



Activity 1
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 1
TAB: MAIN 3



Activity 1
Appendix

Possible ROBOTC Solution
FILENAME: Activity1_1.c

Activity1_1.c

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard!!*//
3
4  /*
5  Create a three point turn using timing and steering.
6  */
7
8  task main()
9  {
10   //Turn to the right and stop after 1.5 seconds.
11   setMotorSpeed(motorB, 75);
12   setMotorSpeed(motorC, 30);
13   sleep(1500);
14
15   //Reverse to the left and stop after 1 second.
16   setMotorSpeed(motorB, -30);
17   setMotorSpeed(motorC, -75);
18   sleep(1000);
19
20   //You should now be straight and facing the other way. Drive off.
21   setMotorSpeed(motorB, 50);
22   setMotorSpeed(motorC, 50);
23   sleep(3000);
24 }
25
```

This code may be subject to change as ROBOTC is updated periodically.

Activity 1
Appendix

Possible ROBOTC Solution
FILENAME: Activity1_2c

Activity1_2.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

/*
Create a three point turn using timing and steering.
Introducing the Ultrasonic Sensor to act as parking sensors.
*/

task main()
{
  //Turn to the right and stop after 1.5 seconds.
  setMotorSpeed(motorB, 75);
  setMotorSpeed(motorC, 30);
  sleep(1500);

  //Reverse to the left while the Ultrasonic Sensor sees a value greater than or
  setMotorSpeed(motorB, -30);
  setMotorSpeed(motorC, -75);
  while(getUSDistance(sonarSensor) >= 7)
  {
    //Keep reversing while the Ultrasonic Sensor sees a value greater than or eq
  }

  //Once the Ultrasonic Sensor sees a value less than 7cm.
  //Stop the robot for 1 second.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  sleep(1000);

  //You should now be straight and facing the other way. Drive off.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);
  sleep(3000);
}
```

This code may be subject to change as ROBOTC is updated periodically.

Activity 1
Appendix

Possible ROBOTC Solution
FILENAME: Activity1_3c

Activity1_3.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

/*
Create a three point turn using timing and steering.
Introducing the Ultrasonic Sensor to act as a parking sensor.
Play an alert before moving off.
*/

task main()
{
  //Turn to the right and stop after 1.5 seconds
  setMotorSpeed(motorB, 75);
  setMotorSpeed(motorC, 30);
  sleep(1500);

  //Reverse to the left while the Ultrasonic Sensor sees a value greater than or equal to 7cm.
  setMotorSpeed(motorB, -30);
  setMotorSpeed(motorC, -75);
  while(getUSDistance(sonarSensor) >= 7)
  {
    //Keep reversing while the Ultrasonic Sensor sees a value greater than or equal to 7cm.
    sleep(10);
  }

  //Once the Ultrasonic Sensor sees a value less than 7cm.
  //Play sound alert.
  //Wait for 1 second.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  playTone(440, 100);

  // Wait for the tone to be done playing
  while(bSoundActive)
  {
    sleep(10);
  }

  sleep(1000);

  //You should now be straight and facing the other way. Drive off.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);
  sleep(3000);
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 2

Written Instructions for a Three Point Turn (Text-Based Programming)

Activity 2

Written Instructions for a Three Point Turn

This activity will be a repeat of Activity 1, but this time, the students will be moving from visual to text-based programming.

Discuss with the students the different programming environments and the reasons we have different environments.

Explore the concept that all people are different and we learn in different ways. Some people prefer text-based languages, others visual. Point out that during the coming weeks the students will be asked to write text-based solutions to match their EV3 programming.

Ask the students to complete some simple driving tasks so they can fully understand how the various move / motor blocks work.

INTRODUCE THE STUDENTS TO THE ROBOTC SOFTWARE

Point out that the ROBOTC software is different in many ways. For example, the need to tell ROBOTC which sensors and motors are being used.

This lesson will be mainly concerned with repeating the activities in Activity 1. That means writing an algorithm to perform a three point turn.

Activity 2
Lesson Plan

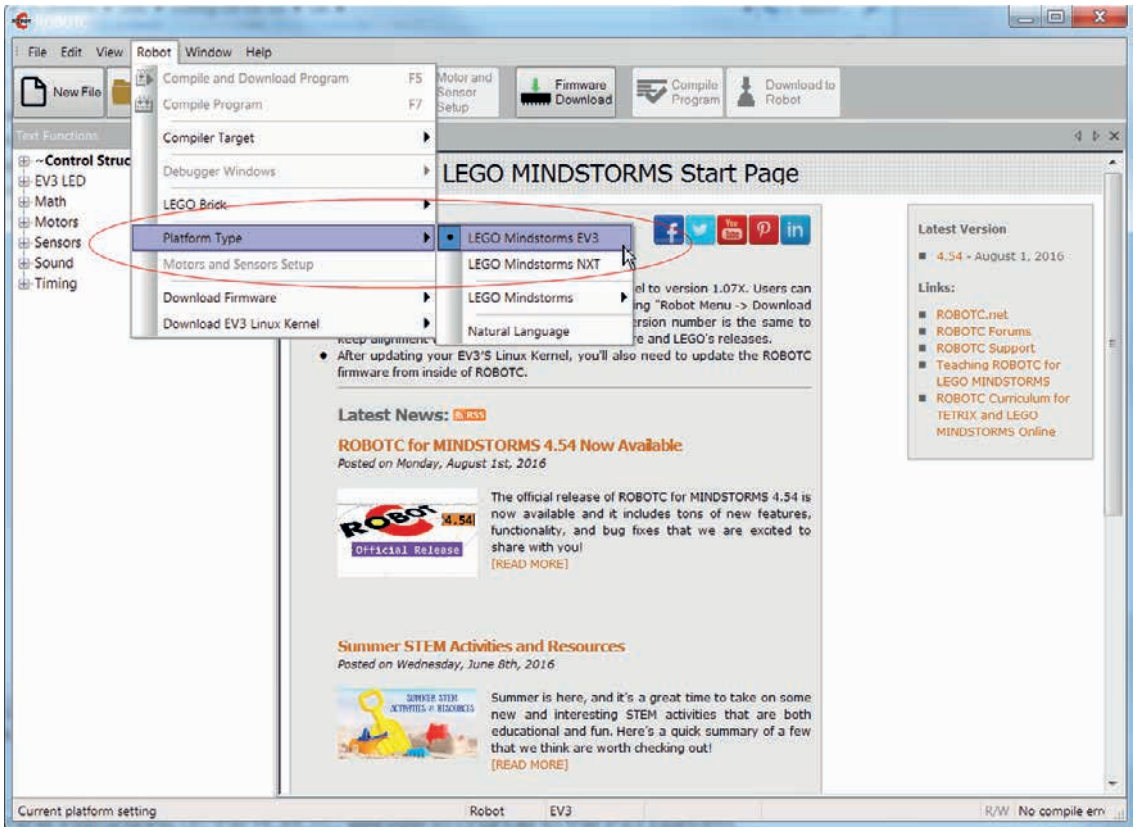
OUTCOMES

Students will be able to:

- use the ROBOTC program to create text programs of the EV3 programs created in Activity 1
- understand the importance of syntax, i.e. semi colons, brackets etc.
- set up a ROBOTC activity
- understand that algorithms are capable of carrying out a series of instructions in order
- build and program a simple wheeled robot to carry out a 180 degree turn using the three point turn method
- be introduced to the 'setMotorSpeed' and 'sleep' commands

INTRODUCTION

- Remind the students about what they did in the last activity. They had to construct a wheeled robot. Use the hardcopy of the building instructions included in the EV3 base set to construct the base model if they have not already done so.
- Demonstrate to the students how they can access the ROBOTC software. Show where ROBOTC highlights the platform to be used. Tell the students that some companies who build robots and automate things use this software.

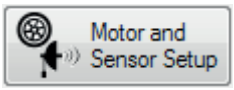


Activity 2
Lesson Plan

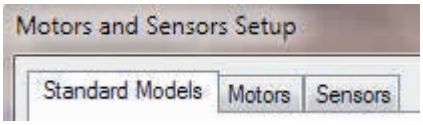
- Point out where the software tells the user which platform is being used as circled above. If the platform is different, follow the online instructions on the ROBOTC website to change to the EV3 platform.
  You will need to do the following:
  1. Change the platform type to LEGO® MINDSTORMS® EV3.
  2. Update the EV3 Firmware (Operating System) by clicking on 'Download EV3 Linux Kernel' and then 'Standard File'.
  3. Install the ROBOTC Firmware by clicking on 'Download Firmware and Standard File'.

SETTING UP ROBOTC FOR PROGRAMMING

- Explain that, unlike the EV3 Software, we have to tell the ROBOTC software which sensors and motors we are going to be using for the activity. There are two ways in which users need to set up the software to match the hardware. These are explained below:
  4. Using the Preset models
  5. Manually configuring the software
- The first thing to do is to click on the 'Motor and Sensor Setup' button.

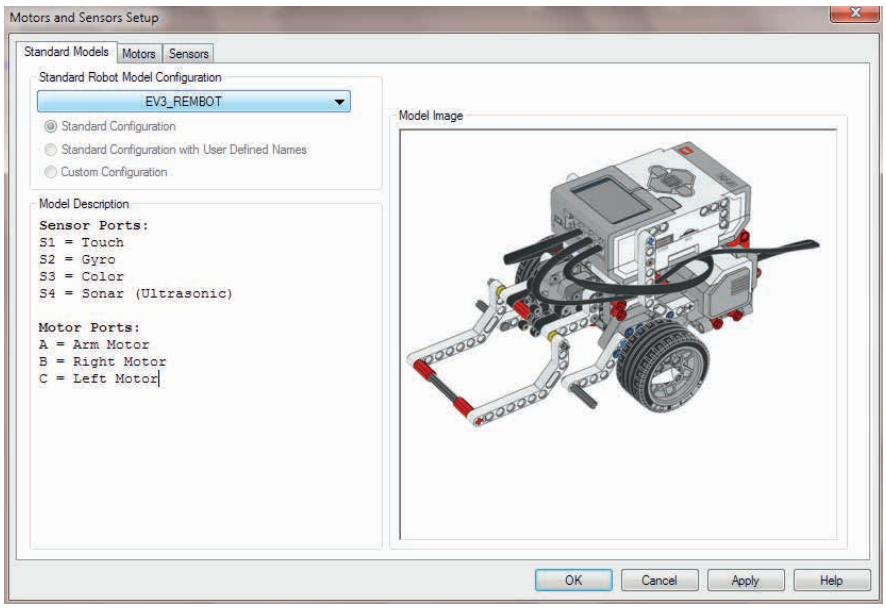


- The user now has the choice of selecting a Standard Model or to configure their own robots using the 'Motors' and 'Sensors' tabs.



USING STANDARD MODELS

- Click on the tab called 'Standard Models'.
- Then click on the drop-down menu and choose EV3__REMBOT. This is the standard Robot Educator model. An image of the Robot Educator model will appear, as well as all of the sensor and motor ports.



Activity 2
Lesson Plan

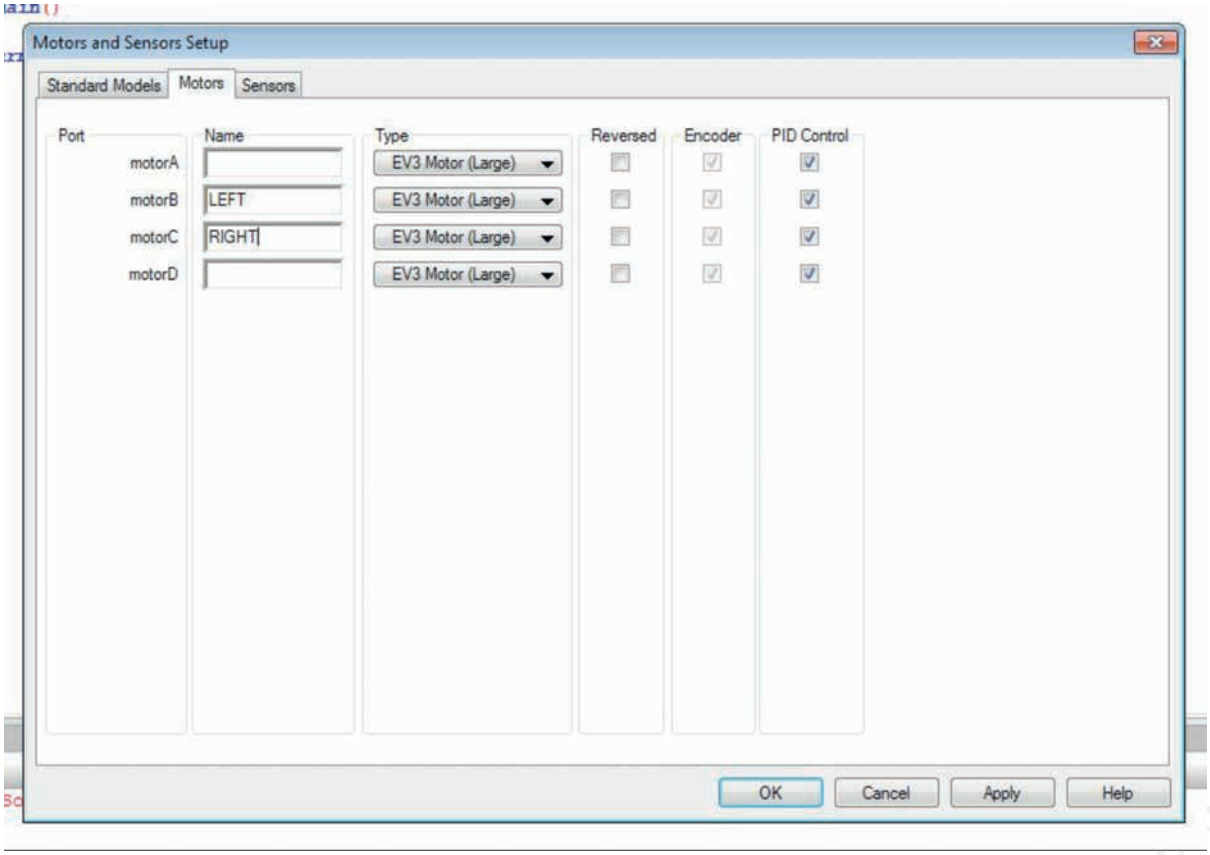
- Notice that at the top of the programming screen, the #pragma config displays *EV3_REMBOT.*

LEGO Start Page | SourceFile002.c*

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
3
4  task main()
5  {
6
7
8
9  }
```

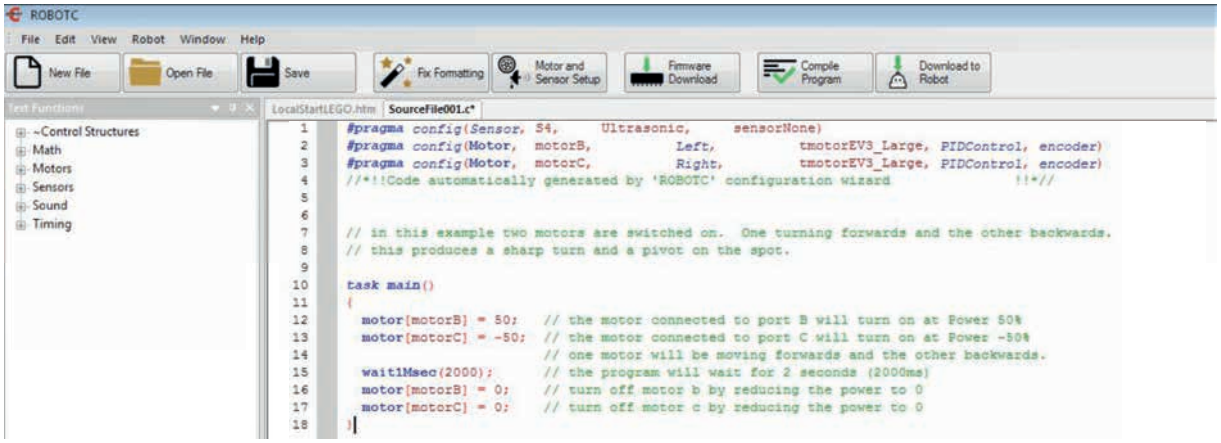
MANUALLY SETTING UP THE MOTORS AND SENSORS

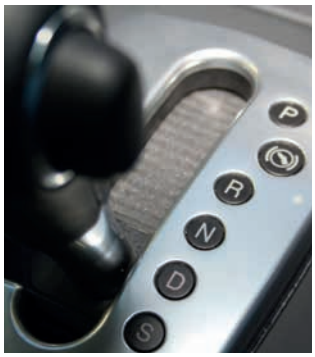
- Click on the 'Motor' tab.
- Name motor B 'LEFT' and motor C 'RIGHT' as shown below. Then click on OK to confirm the set up.



Activity 2
Lesson Plan

- Now demonstrate to the students how to create a simple turn using the text-based language.
- Then point out that they can make annotations to their program by simply adding two backslashes, as shown below. It is important for the students to understand that each part of the text-based program is just like the text boxes in the EV3 Software.





Activity 2
Lesson Plan

INTRODUCTION: EXPLORATION TASK

- The students will work in pairs or threes to construct the base model. This may already have been done in Activity 1.
- Once the base model has been made, they will begin a new program and experiment with different ways of turning. There are three examples below.

LEGO Start Page | **Explore1.c** | explore2.c | explore3.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//

task main()
{
  //motor c reverses at power 50
  //motor b drives forward at power 50
  //robot will pivot centrally for 1 second (1000ms).
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, -50);
  sleep(1000);
}
```

Example 1: Pivoting Centrally

LEGO Start Page | Explore1.c | **explore2.c** | explore3.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//

task main()
{
  //motor c drives at power 50
  //robot will pivot on stationary wheel for 1 second (1000ms).
  setMotorSpeed(motorC, 50);
  sleep(1000);
}
```

Example 2: Pivoting on One Wheel

LEGO Start Page | Explore1.c | explore2.c | **explore3.c**

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//

task main()
{
  //motor b drives forward at power 75
  //motor c drives forward at power 50
  //robot will turn gently for 1 second (1000ms).
  setMotorSpeed(motorB, 75);
  setMotorSpeed(motorC, 50);
  sleep(1000);
}
```

Example 3: Slow Curved Turn

Activity 2
Lesson Plan

MAIN CHALLENGE 1

- Get some feedback from the students about the different methods they have used in order to make their wheeled robot turn. Ask them to explain how they arrived at that point.
- Explain to the students that they will need to use what they have discovered about turning the model in order to simulate a three point turn.
- Consider showing the students an online video of how to perform a three point turn.
- Students will then use ROBOTC to create a three point turn program using different power settings in each motor in order to simulate steering.
- Encourage the students to constantly evaluate and debug their program in order to ensure that their wheeled robot turns 180 degrees.
- NB: You may wish to 'tape out' road markings in order to give the students a constrained area in which to turn their wheeled robot.
- NB: You will see in the possible solution programs that the wheeled robot 'should be straight' before the final stage and driving off. It is worth noting that in a real situation this is not always the case, and that the students could include a curve here before the straight line. Encourage the students to appreciate that there is no 'wrong' way of solving these challenges.
- Ask the students to compare the two approaches to the same task. Which was the easiest and most efficient?

Activity 2
Lesson Plan

POSSIBLE SOLUTION
FILENAME: Activity2_1.c

Activity2_1.c

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard!!*//
3
4  /*
5  Create a three point turn using timing and steering.
6  */
7
8  task main()
9  {
10   //Turn to the right and stop after 1.5 seconds.
11   setMotorSpeed(motorB, 75);
12   setMotorSpeed(motorC, 30);
13   sleep(1500);
14
15   //Reverse to the left and stop after 1 second.
16   setMotorSpeed(motorB, -30);
17   setMotorSpeed(motorC, -75);
18   sleep(1000);
19
20   //You should now be straight and facing the other way. Drive off.
21   setMotorSpeed(motorB, 50);
22   setMotorSpeed(motorC, 50);
23   sleep(3000);
24 }
```

This code may be subject to change as ROBOTC is updated periodically.

Activity 2
Lesson Plan

MAIN CHALLENGE 2

- Introduce the use of the Ultrasonic Sensor. Demonstrate to the students how to add another sensor to the configuration. Simply enter the motor/sensor set up to add more.
- Ask the students how they could make the program more autonomous and able to account for any obstacles that might appear while the wheeled robot is reversing.
- Demonstrate the 'while' command and how to use it with the Ultrasonic Sensor. Point out the motor settings inside of the 'while' loop and the motor settings outside of the loop.
- The students will extend their programming to ensure that their wheeled robot stops at a given point in reaction to the Ultrasonic Sensor.
- Point out that their programs will simulate the driver applying the brakes when reversing close to an obstacle.
- Ask the students to compare the two different approaches to the same task. Which was the easiest and most efficient?

Activity 2
Lesson Plan

POSSIBLE SOLUTION
FILENAME: Activity2__2.c

Activity2_2.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

/*
Create a three point turn using timing and steering.
Introducing the Ultrasonic Sensor to act as parking sensors.
*/

task main()
{
  //Turn to the right and stop after 1.5 seconds.
  setMotorSpeed(motorB, 75);
  setMotorSpeed(motorC, 30);
  sleep(1500);

  //Reverse to the left while the Ultrasonic Sensor sees a value greater than or equal to 7cm.
  setMotorSpeed(motorB, -30);
  setMotorSpeed(motorC, -75);
  while(getUSDistance(sonarSensor) >= 7)
  {
    //Keep reversing while the Ultrasonic Sensor sees a value greater than or equal to 7cm.
  }

  //Once the Ultrasonic Sensor sees a value less than 7cm.
  //Stop the robot for 1 second.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  sleep(1000);

  //You should now be straight and facing the other way. Drive off.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);
  sleep(3000);
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 2
Lesson Plan

MAIN CHALLENGE 3

- Use the Ultrasonic Sensor to add safety features (such as warning sounds) to the wheeled robot.
- Ask the students what happens in a car when it is reversing and it approaches an obstacle – a warning sound can be heard.
- Introduce the 'sound' command to add a warning sound prior to the wheeled robot stopping. Demonstrate to the students how the 'sound' command works and where it should appear in their program.
- The students must adapt their existing program to incorporate the 'sound' command to emit a warning sound when the wheeled robot is a certain distance from an obstacle.
- Encourage the students to constantly debug their programs so that the sound and stopping distance work efficiently.

Activity 2
Lesson Plan

POSSIBLE SOLUTION
FILENAME: Activity2__3.c

Activity2_3.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

/*
Create a three point turn using timing and steering.
Introducing the Ultrasonic Sensor to act as a parking sensor.
Play an alert before moving off.
*/

task main()
{
  //Turn to the right and stop after 1.5 seconds
  setMotorSpeed(motorB, 75);
  setMotorSpeed(motorC, 30);
  sleep(1500);

  //Reverse to the left while the Ultrasonic Sensor sees a value greater than or equal to 7cm.
  setMotorSpeed(motorB, -30);
  setMotorSpeed(motorC, -75);
  while(getUSDistance(sonarSensor) >= 7)
  {
    //Keep reversing while the Ultrasonic Sensor sees a value greater than or equal to 7cm.
    sleep(10);
  }

  //Once the Ultrasonic Sensor sees a value less than 7cm.
  //Play sound alert.
  //Wait for 1 second.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  playTone(440, 100);

  // Wait for the tone to be done playing
  while(bSoundActive)
  {
    sleep(10);
  }

  sleep(1000);

  //You should now be straight and facing the other way. Drive off.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);
  sleep(3000);
}
```

This code may be subject to change as ROBOTC is updated periodically.

CLASS DISCUSSION

- Recap what the students have learnt in this activity.
- Make sure that the students use and understand the vocabulary that is used throughout the activity – recap what the key words mean.
- Ask one or two groups to demonstrate their program. Discuss what works well and what could be improved.
- Ask the students to evaluate the text-based program. How many of the students would like to use text-based programming as their main language?

Activity 2
Student Worksheets

CHALLENGES FOR TODAY

Today is designed to introduce you to and get you started with the ROBOTC software.

You will already have had some time to experiment with the setMotorSpeed command to get your wheeled robot to move around the room. Now you will need to hone those skills in order to carry out three challenges.

Good luck!

CHALLENGE 1

Program your wheeled robot to perform a three point turn.

You will need to turn your wheeled robot while going forwards, then reverse it before driving forwards again.

Watch the online video clip again to remind you of what it looks like, and make sure that you don't cross the road markings!

Programming Commands to Consider

setMotorSpeed **sleep**

Plan your program first. Write it in pseudo-code below:

Activity 2
Student Worksheets

CHALLENGE 2

You are now going to experiment with one of the EV3 sensors – the Ultrasonic Sensor.

Program your wheeled robot to perform a three point turn and use the Ultrasonic Sensor as a 'reverse parking sensor', so that your wheeled robot stops at a given distance from an obstacle when it is reversing.

Can your wheeled robot 'put the brakes on' before it drives forwards again?

You will need to use your knowledge of the 'while' command here, and attach the Ultrasonic Sensor to the rear of your wheeled robot.

Programming Commands to Consider

setMotorSpeed **sleep** **while** **getUSDistance**

Plan your program first. Write it in pseudo-code below:

Activity 2
Student Worksheets

CHALLENGE 3

You will now simulate warning sounds.

What often happens when a car is reversing and approaches an obstacle?

Now that your wheeled robot stops in response to the Ultrasonic 'parking sensor', can you extend your program so that your wheeled robot emits a warning sound just before the brakes are applied when reversing?

You will need to constantly debug your program so that the warning sound stops at the same time as the wheeled robot. Which parts of your program will need to change?

Programming Commands to Consider

setMotorSpeed **sleep** **while** **getUSDistance** **playTone**

Plan your program first. Write it in pseudo-code below:

Activity 2
Student Worksheets

After a programming activity, it is important to note down your thoughts and observations. Consider the following points and then in the box below record how the activity went.

- How could you improve your program?
- Could your program have been more streamlined? Have you used too many commands? Is there a more efficient way of building your program?
- What examples of real-world applications could you see your program being used in?

Thoughts and Observations

Activity 2
Appendix

APPENDIX FOR ACTIVITY 2: LARGE IMAGES AND PROGRAMS

Activity 2
Appendix

Activity 2
Appendix

Activity 2
Appendix

Possible ROBOTC Solution
FILENAME: Activity2_1.c

Activity2_1.c

```
1  #pragma config(StandardModel, "EV3_REMBOT")
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard!!*//
3
4  /*
5  Create a three point turn using timing and steering.
6  */
7
8  task main()
9  {
10   //Turn to the right and stop after 1.5 seconds.
11   setMotorSpeed(motorB, 75);
12   setMotorSpeed(motorC, 30);
13   sleep(1500);
14
15   //Reverse to the left and stop after 1 second.
16   setMotorSpeed(motorB, -30);
17   setMotorSpeed(motorC, -75);
18   sleep(1000);
19
20   //You should now be straight and facing the other way. Drive off.
21   setMotorSpeed(motorB, 50);
22   setMotorSpeed(motorC, 50);
23   sleep(3000);
24 }
```

This code may be subject to change as ROBOTC is updated periodically.

Activity 2
Appendix

Possible ROBOTC Solution
FILENAME: Activity2_2.c

Activity2_2.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

/*
Create a three point turn using timing and steering.
Introducing the Ultrasonic Sensor to act as parking sensors.
*/

task main()
{
  //Turn to the right and stop after 1.5 seconds.
  setMotorSpeed(motorB, 75);
  setMotorSpeed(motorC, 30);
  sleep(1500);

  //Reverse to the left while the Ultrasonic Sensor sees a value greater than or equal to 7cm.
  setMotorSpeed(motorB, -30);
  setMotorSpeed(motorC, -75);
  while(getUSDistance(sonarSensor) >= 7)
  {
    //Keep reversing while the Ultrasonic Sensor sees a value greater than or equal to 7cm.
  }

  //Once the Ultrasonic Sensor sees a value less than 7cm.
  //Stop the robot for 1 second.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  sleep(1000);

  //You should now be straight and facing the other way. Drive off.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);
  sleep(3000);
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 2
Appendix

Possible ROBOTC Solution
FILENAME: Activity2_3.c

Activity2_3.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

/*
Create a three point turn using timing and steering.
Introducing the Ultrasonic Sensor to act as a parking sensor.
Play an alert before moving off.
*/

task main()
{
  //Turn to the right and stop after 1.5 seconds
  setMotorSpeed(motorB, 75);
  setMotorSpeed(motorC, 30);
  sleep(1500);

  //Reverse to the left while the Ultrasonic Sensor sees a value greater than or equal to 7cm.
  setMotorSpeed(motorB, -30);
  setMotorSpeed(motorC, -75);
  while(getUSDistance(sonarSensor) >= 7)
  {
    //Keep reversing while the Ultrasonic Sensor sees a value greater than or equal to 7cm.
    sleep(10);
  }

  //Once the Ultrasonic Sensor sees a value less than 7cm.
  //Play sound alert.
  //Wait for 1 second.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  playTone(440, 100);

  // Wait for the tone to be done playing
  while(bSoundActive)
  {
    sleep(10);
  }

  sleep(1000);

  //You should now be straight and facing the other way. Drive off.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);
  sleep(3000);
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 3
Reversing the Robot

Activity 3

Reversing the Robot

In this activity, the students will program their wheeled robot to simulate visual indicators for pedestrians, other car drivers and passengers when reversing.

Using the Brick Status Light and Display Block, the students will create programs that will visually communicate what their wheeled robot is doing.

The students will also use the Touch Sensor to simulate forward and reverse gears.

Activity 3
Lesson Plan

OUTCOMES

Students will be able to:

- understand that algorithms are capable of carrying out a series of instructions in order.
- use the Move Steering Block to make their wheeled robot travel in a straight line.
- use the Wait Block in relation to Touch Sensor(s).
- utilise the on-brick status light and display functions.
- extend their computational thinking through the creation of more complex algorithms.

VOCABULARY

input, output, algorithm, wait, Touch Sensor, debug, Move Steering Block

INTRODUCTION

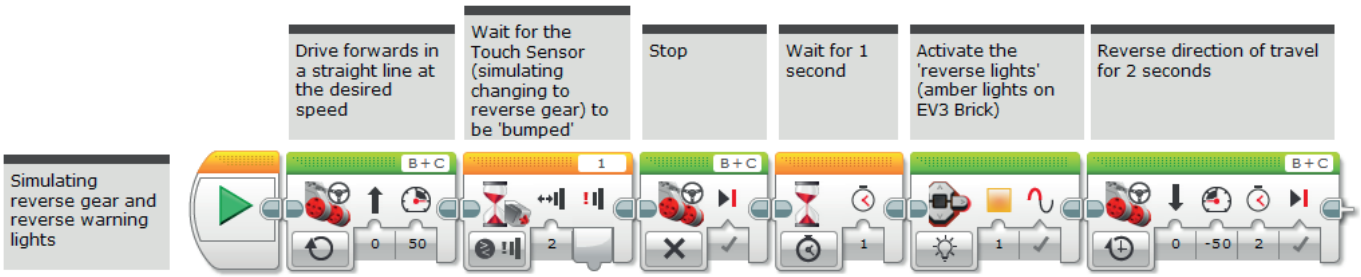
- Explain to the students that during the course of the lesson, they will program their wheeled robot to simulate reversing, including lights and dashboard indicators, and that they will be introduced to another sensor.
- Ask them to discuss briefly what happens when cars reverse. Make sure that the students realise that there are reverse lights to warn other motorists and pedestrians, and that there may be indicators on the dashboard showing which gear the car is in and indicating the direction of travel.
- Demonstrate, using the EV3 software, the three different modes of the Touch Sensor (pressed, released, bumped) and the Brick Status Light Block.

Activity 3
Lesson Plan

MAIN CHALLENGE 1

- Explain to the students that during this activity they will be creating a program that simulates putting a car into reverse gear while displaying appropriate warning lights.
- The students are to create a program that will drive their wheeled robot forwards. When the 'reverse' gear is activated (by using a Touch Sensor) the wheeled robot will travel backwards and display the reverse warning lights using the EV3 Brick Status Light.
- You may wish to give the students some time prior to this activity in order to explore the Touch Sensor and the Brick Status Light Block.
- NB: The students could use the Move Tank Block to drive the wheeled robot, but it may be more beneficial to use the Move Steering Block.

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 3
TAB: MAIN 1

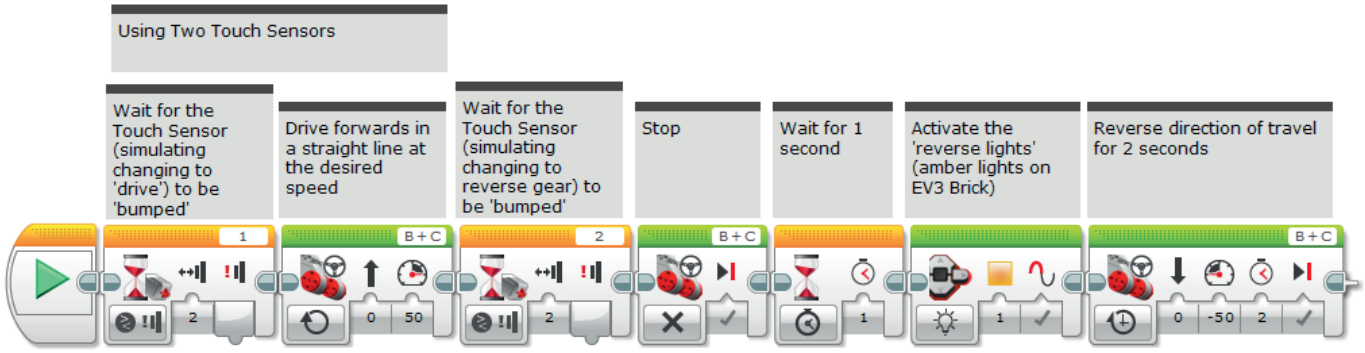


MAIN CHALLENGE 2

- Introduce a second Touch Sensor to act as the 'drive' gear.
- Show the students that they can use an additional Touch Sensor and that the EV3 Software will detect which input it is connected to.
- The students will build on their previous learning by creating programs which simulate forward and reverse gears. The second Touch Sensor (the 'drive' gear) will trigger the start of the program.

Activity 3
Lesson Plan

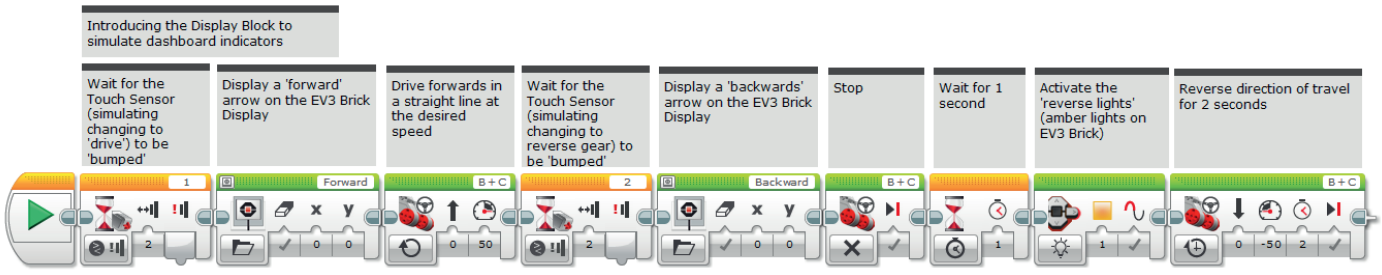
POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 3
TAB: MAIN 2



MAIN CHALLENGE 3

- Utilise the EV3 Brick screen as a dashboard indicator.
- The students will further develop their programs to incorporate the EV3 Brick screen using the Display Block. The forward and reverse motion of the wheeled robot will trigger the screen to display arrows that indicate the direction of travel.
- Demonstrate the Display Block to the students.
- The students may wish to explore the options in the Display Block and customise their screens to suit their programs.
- What do the students think that the screen could be used for within the context of an automated car? Make sure that the students realise that the screen could be used to relay important information to the passengers of the driverless car.

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 3
TAB: MAIN 3



Activity 3
Lesson Plan

CLASS DISCUSSION

- Ask one or two groups to demonstrate their programs and how their wheeled robot moves.
- How many variations did the group as a whole come up with? This is a useful exercise and reinforces the concept that there are many possible solutions to any given problem.
- Explain to the students that next time they will experiment with the Colour Sensor in order to continue the development of their fully automated wheeled robot.

Activity 3
Student Worksheets

CHALLENGES FOR TODAY

Today's challenges will require you to build on what you have already learnt about programming. You will be using another sensor (the Touch Sensor) and also using the functions of the EV3 Brick. You will program your brick to activate the on-brick lights and use the screen as a visual indicator.

By the end of the third challenge, your programs will make your wheeled robot simulate forward and reverse gears, reverse lights and a dashboard indicator.

CHALLENGE 1

Can you write a program that will drive your wheeled robot forwards and then put it into reverse when you press the Touch Sensor?

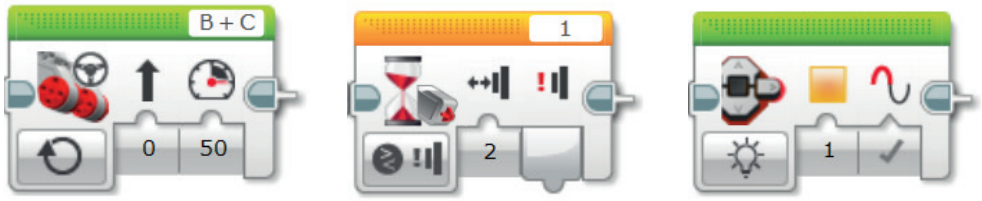
Try this first and then extend your program:

What happens on the outside of vehicles when they are reversing in order to let pedestrians and other road users know what is happening?

Your wheeled robot should display reverse warning lights.

Use the EV3 Brick and the Brick Status Light to simulate reverse lights.

Blocks to Consider



Plan your program first. Write it in pseudo-code below:

Activity 3
Student Worksheets

CHALLENGE 2

Can you extend your program so that your wheeled robot has two gears (forward and reverse)?

Your wheeled robot should 'start' (drive forwards) when the 'drive' gear is activated.

Tip: You will need a second Touch Sensor.

Blocks to Consider

Use the same blocks that you used in programming task 1, but also consider using the following:

Plan your program first. Write it in pseudo-code below:

Activity 3
Student Worksheets

CHALLENGE 3

What happens inside of a car when it is in different gears?

There is often an indicator / image on the dashboard to let the driver know which gear the car is in.

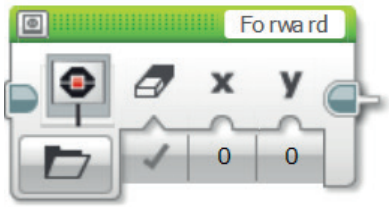
Can you simulate this indicator in your program by using the Display Block?

You may wish to explore the Display Block in order to find suitable images for indicating forward and reverse motions.

Your program should be a further extension of what you have done before and it should still include the reverse lights!

Blocks to Consider

Use the same blocks that you used in programming tasks 1 and 2, but also consider using the following:



Plan your program first. Write it in pseudo-code below:

Activity 3
Student Worksheets

After a programming activity, it is important to note down your thoughts and observations. Consider the following points and then in the box below record how the activity went.

- How could you improve your program?
- Could your program have been more streamlined? Have you used too many blocks? Is there a more efficient way of building your program?
- What examples of real-world applications could you see your program being used in?

Thoughts and Observations

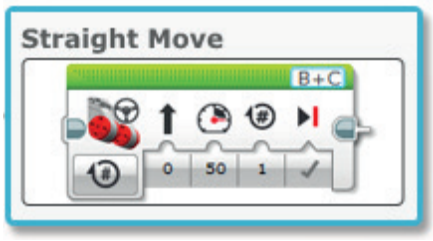
Activity 3
Teacher Notes

ROBOT EDUCATOR TUTORIALS

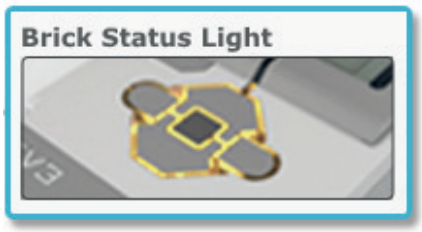
The following Robot Educator Tutorials will help teachers and their students to solve the challenges.

NEW ROBOT EDUCATOR TUTORIALS

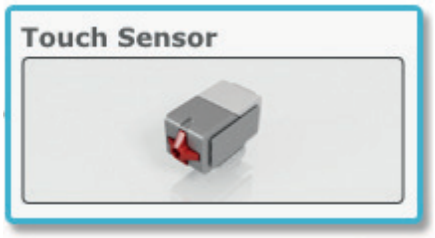
Basics > Straight Move



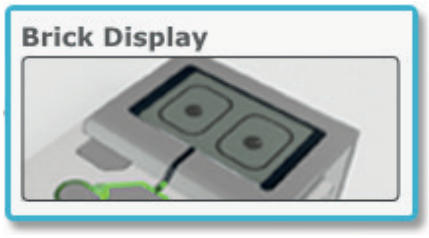
Hardware > Brick Status Light



Hardware > Touch Sensor

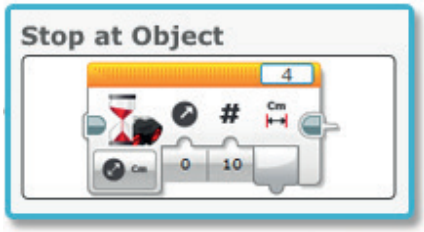


Hardware > Brick Display



ROBOT EDUCATOR TUTORIALS PREVIOUSLY COVERED

Basics > Stop at Object



Activity 3
Appendix

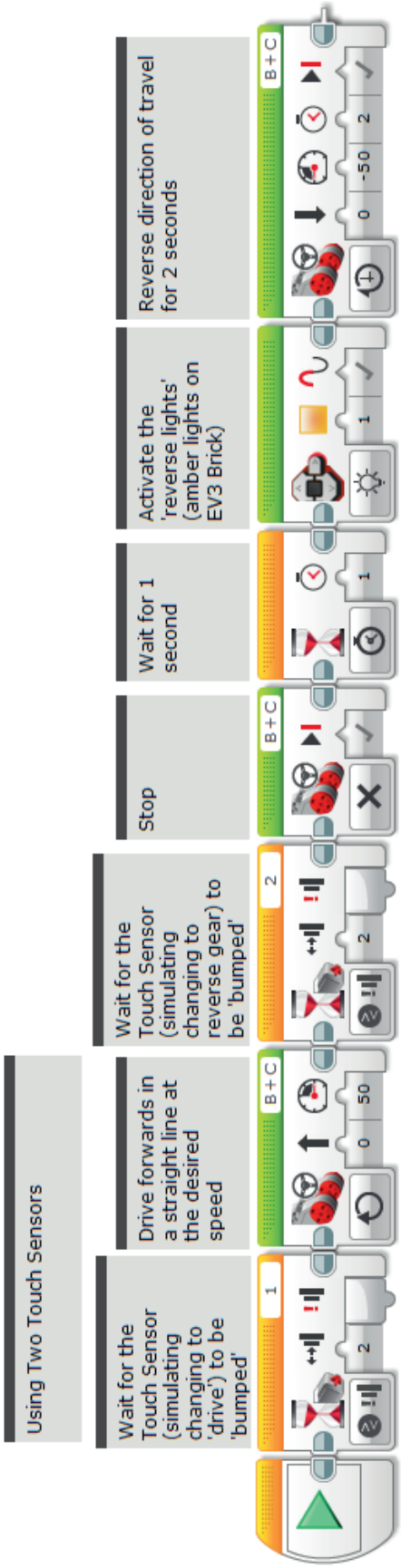
APPENDIX FOR ACTIVITY 3: LARGE IMAGES AND PROGRAMS

Activity 3
Appendix

Activity 3
Appendix

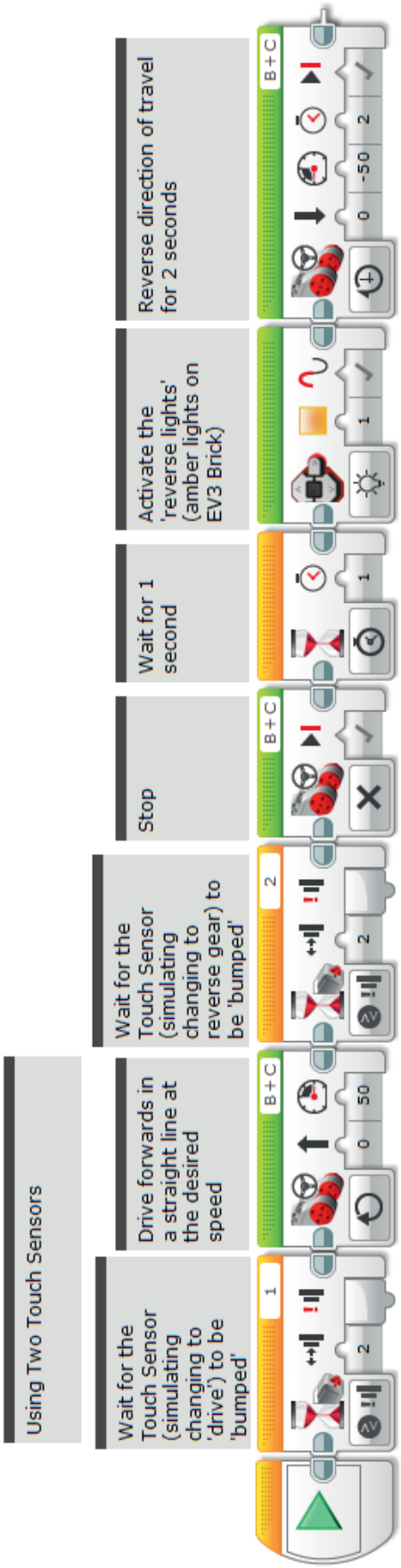
Activity 3
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 3
TAB: MAIN 1



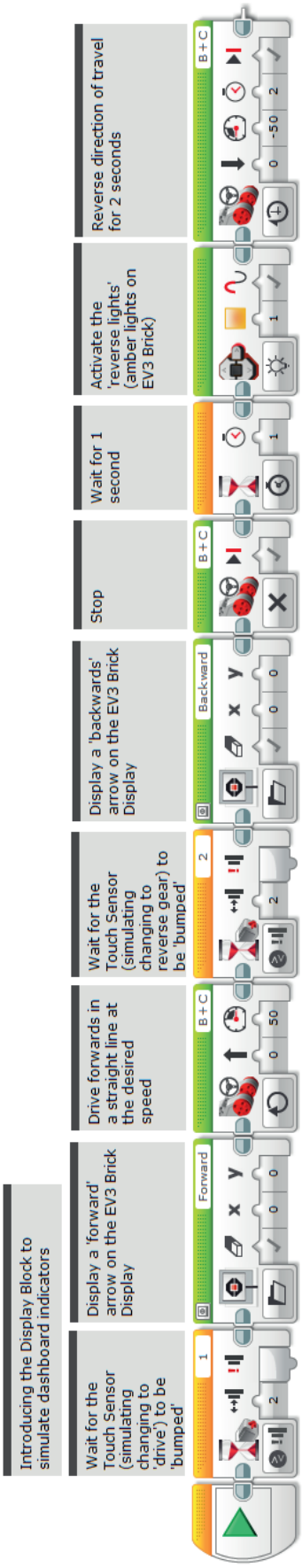
Activity 3
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 3
TAB: MAIN 2



Activity 3
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 3
TAB: MAIN 3



Activity 3
Appendix

Possible ROBOTC Solution
FILENAME: Activity3_1.c

Activity3_1.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Create a program where the robot drives forward until the Touch Sensor is pressed.
The robot then reverses flashing an orange LED.
*/

task main()
{
  //Set the MotorSpeed to 50%.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);

  //Wait for touch sensor to be pressed.
  while(getTouchValue(touchSensor) ==0)
  {
    sleep(10);
  }

  //Set the MotorSpeed to 0% or off. Wait 1 second.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  sleep(1000);

  //Flash the EV3 LED Orange.
  setLEDColor(ledOrangeFlash);

  //Set the MotorSpeed to -50%(reverse). Wait 2 seconds.
  setMotorSpeed(motorB, -50);
  setMotorSpeed(motorC, -50);
  sleep(2000);
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 3
Appendix

Possible ROBOTC Solution
FILENAME: Activity3_2.c

Activity3_2.c

```
#pragma config(Sensor, S1,     touchSensor1,    sensorEV3_Touch)
#pragma config(Sensor, S2,     touchSensor2,    sensorEV3_Touch)
#pragma config(Sensor, S3,     colorSensor,     sensorEV3_Color)
#pragma config(Sensor, S4,     sonarSensor,     sensorEV3_Ultrasonic)
#pragma config(Motor,  motorA,          armMotor,      tmotorEV3_Large, PIDControl, encoder)
#pragma config(Motor,  motorB,          leftMotor,     tmotorEV3_Large, PIDControl, driveLeft, encoder)
#pragma config(Motor,  motorC,          rightMotor,    tmotorEV3_Large, PIDControl, driveRight, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Create a program that starts the robot on the press of the Touch Sensor. The robot
then drives forward until a second Touch Sensor is pressed touched. The robot then reverses
flashing an orange LED.
*/

task main()
{
  //Wait for Touch Sensor 1 to be pressed.
  while(getTouchValue(touchSensor1) ==0)
  {
    sleep(10);
  }

  //Set the MotorSpeed to 50%.
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);

  //Wait for Touch Sensor to be pressed.
  while(getTouchValue(touchSensor2) ==0)
  {
    sleep(10);
  }

  //Set the MotorSpeed to 0% or off. Wait 1 second.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  sleep(1000);

  //Flash the EV3 LED Orange.
  setLEDColor(ledOrangeFlash);

  //Set the MotorSpeed to -50%(reverse). Wait 2 seconds.
  setMotorSpeed(motorB, -50);
  setMotorSpeed(motorC, -50);
  sleep(2000);
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 3
Appendix

Possible ROBOTC Solution
FILENAME: Activity3_3.c

Activity3_3.c

```
#pragma config(Sensor, S1,     touchSensor1,    sensorEV3_Touch)
#pragma config(Sensor, S2,     touchSensor2,    sensorEV3_Touch)
#pragma config(Sensor, S3,     colorSensor,     sensorEV3_Color)
#pragma config(Sensor, S4,     sonarSensor,     sensorEV3_Ultrasonic)
#pragma config(Motor,  motorA,          armMotor,      tmotorEV3_Large, PIDControl, encoder)
#pragma config(Motor,  motorB,          leftMotor,     tmotorEV3_Large, PIDControl, driveLeft, encoder)
#pragma config(Motor,  motorC,          rightMotor,    tmotorEV3_Large, PIDControl, driveRight, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Create a program that communicates messages via the EV3 screen.
Users develop the program from the previous actvity.
NOTE: ***THIS PROGRAM USES TEXT INSTEAD OF IMAGES***
*/

task main()
{

  //Wait for Touch Sensor 1 to be pressed.
  while(getTouchValue(touchSensor1) ==0)
  {
    sleep(10);
  }

  //Display the word "forward" on the EV3 screen.
  displayCenteredBigTextLine(4, "forward");

  //Set the MotorSpeed to 50%
  setMotorSpeed(motorB, 50);
  setMotorSpeed(motorC, 50);

  //Wait for Touch Sensor 2 to be pressed.
  while(getTouchValue(touchSensor2) ==0)
  {
    sleep(10);
  }

  //Set the MotorSpeed to 0% or off. Wait 1 second.
  //Display the word "reverse" on the EV3 screen.
  displayCenteredBigTextLine(4, "reverse");
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);

  sleep(1000);

  //Flash the EV3 LED Orange.
  setLEDColor(ledOrangeFlash);

  //Set the MotorSpeed to -50%(reverse). Wait 2 seconds.
  setMotorSpeed(motorB, -50);
  setMotorSpeed(motorC, -50);
  sleep(2000);
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 4
Light the Way

LEGOeducation.com.au

LEGO mindstorms education EV3

Activity 4

Light the Way

In this activity, the students will explore the Colour Sensor and how they can use its different modes (sensing colour and light) to simulate various automated functions of a car.

The students' programs will become more sophisticated through the use of sensors which respond to environmental changes.

The students will create programs that will simulate automatic headlights, and they will extend their programs to include a simulation of a manual headlight override function.

Activity 4
Lesson Plan

OUTCOMES

Students will be able to:

- understand that algorithms are capable of carrying out a series of instructions in order
- understand simple Boolean logic and some of its uses
- use the Wait Block in relation to the Colour Sensor
- understand that the Colour Sensor has several functions and that it can measure and react to a range of parameters
- extend their use and understanding of the Display Block
- understand the Loop Block and multi-tasking / parallel programming

VOCABULARY

input, output, algorithm, wait, Colour Sensor, debug, ambient light, loop, Boolean logic, parallel programming

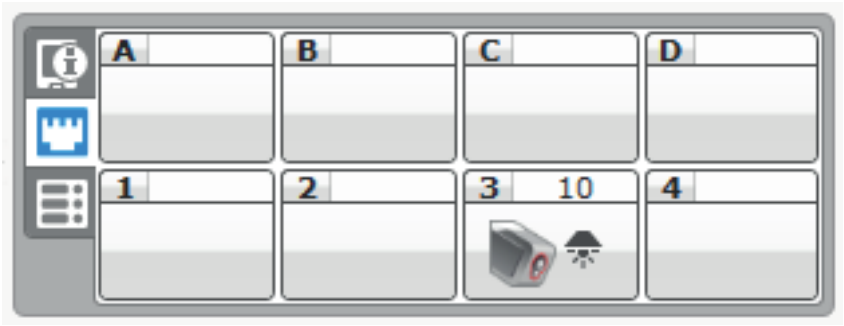
INTRODUCTION

- Explain to the students that during the course of the next two lessons they will be exploring the different functions of the Colour Sensor, and programming their wheeled robot in order to simulate automatic lights. They will also use the Colour Sensor in order to make their wheeled robot more autonomous.
- Tell the students that during this activity they will be exploring how changes in ambient light can trigger reactions in programs and that this is an example of Boolean logic (control of a program through a decision).
- Ask them what 'real life' situations they can think of in which changes in ambient light trigger events (get them to mention examples such as automatic car lights and street lights).
- Explain that they will be exploring how loops work and how to incorporate these into their programs.

Activity 4
Lesson Plan

MAIN CHALLENGE 1

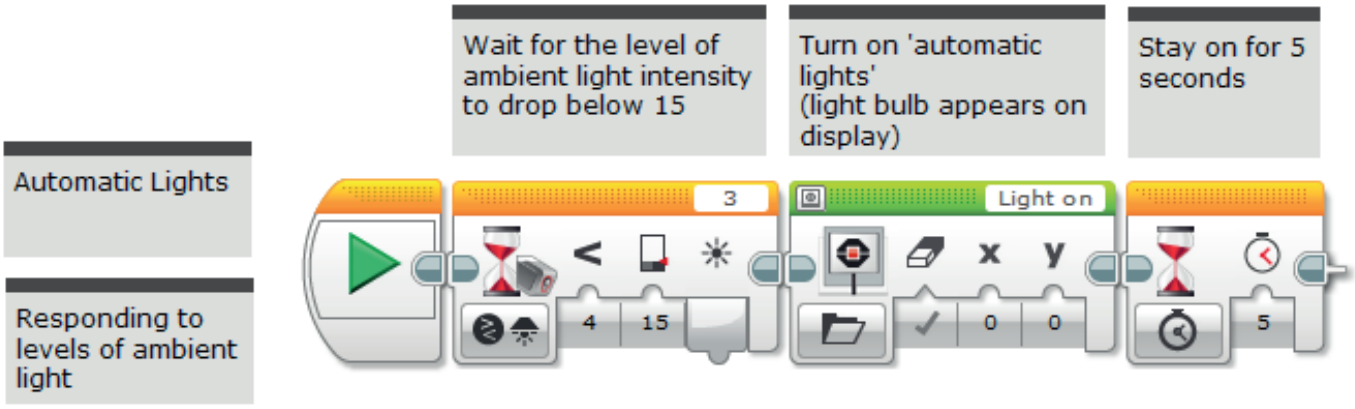
- Introduce the Colour Sensor and demonstrate to the students that it has three modes:
  - Colour, which detects LEGO® system colours
  - Ambient Light Intensity, which detects light levels
  - Reflected Light Intensity, which allows robots to follow given paths
- Ask the students which features of a car might respond to changes in ambient light (automatic lights). When ambient levels drop too low, the headlights turn on.
- The students will write a program in order to simulate this using the Colour Sensor in 'Compare Ambient Light Intensity' mode, along with the EV3 Brick Display in order to simulate headlights (they may use an image of a light bulb, for example).
- Show the students how to use the Port View in the MINDSTORMS® EV3 Software in order to take readings of ambient light.
- The students will need to take measurements of ambient light in the room and make decisions as to what level of light will trigger the display.
- If the students are having difficulties attaching the Colour Sensor, you may wish to point them towards the Robot Educator Building Instructions: *Colour Sensor Forward – Driving Base*. They can also use their own design.



Port View showing ambient light intensity (port 3)

Activity 4
Lesson Plan

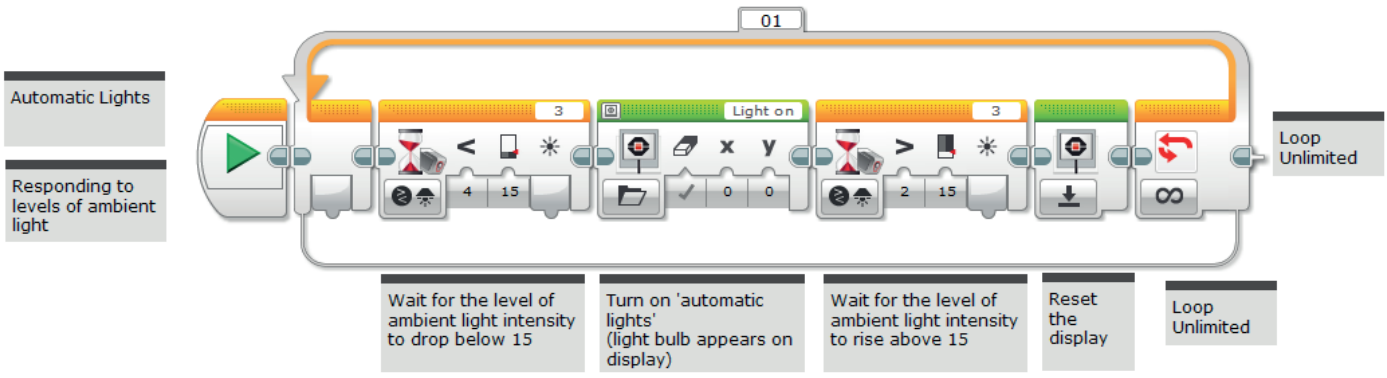
POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 4
TAB: MAIN 1



MAIN CHALLENGE 2

- The students will create a program that reacts to changes in ambient light intensity and that simulates automatic lights more closely. The programs will need to: turn the 'lights' on when it becomes 'dark' and turn the 'lights' off when it becomes 'light' again.
- Introduce the Loop Block to the students and demonstrate how it works while using the EV3 Software.
- Explain the concept of a loop and how it can be used in order to make programs run indefinitely until they are manually stopped.
- Highlight that the ambient light intensity settings in their programs will need to be constantly monitored and debugged in order to ensure that the program is operating correctly.

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 4
TAB: MAIN 2

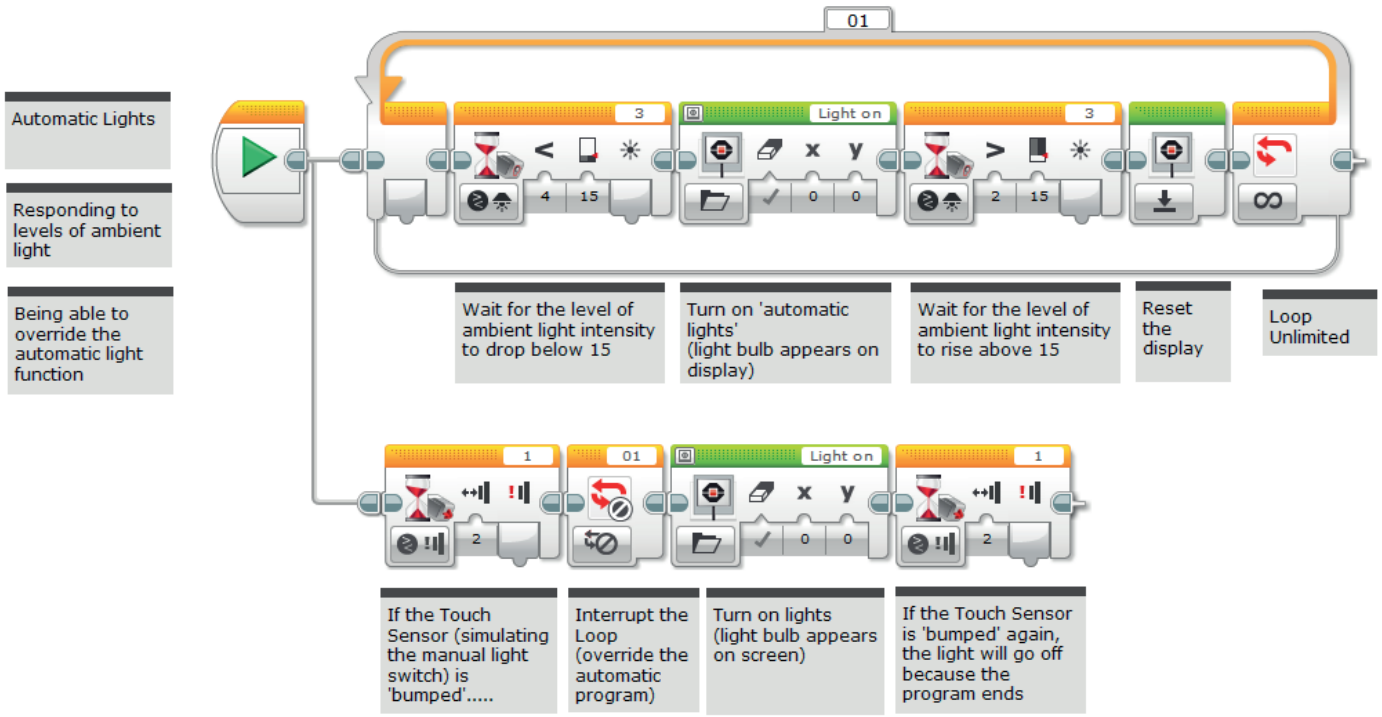


Activity 4
Lesson Plan

MAIN CHALLENGE 3

- Use multitasking / parallel programming to simulate automatic and manual light switches.
- The students will further develop their programs to incorporate parallel programming. They will create a program that expands their automatic lights program in order to give the 'driver' the option of overriding the automatic program and turning on the lights manually.
- They will need to use a Touch Sensor in order to simulate the manual switch.
- This is a further development of the use of Boolean logic.
- Demonstrate to the students how to drag a 'wire' from the Start Block in order to create a parallel program.
- Show the students the Loop Interrupt Block and explain that they can use it to stop whatever is happening within the loop in the parallel program.

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 4
TAB: MAIN 3



Activity 4
Lesson Plan

CLASS DISCUSSION

- NB: Students could explore replacing the Display Block with the Brick Status Light Block so that the wheeled robot simulates real life more closely. They could also combine the two.
- This activity has covered a lot of new concepts and has introduced several new blocks. Use this time to recap on these elements and to ensure that the students understand how they work.
- Ask one or two groups to present their programming solutions.
- Inform the students that in the next activity they will be exploring another use for the Colour Sensor, in order to make their vehicle more autonomous.

Activity 4
Student Worksheets

CHALLENGES FOR TODAY

Today you are going to explore one of the functions of the Colour Sensor – its ability to measure and respond to changes in ambient light intensity.

Automatic lights on cars measure the amount of ambient light that is available and respond accordingly (they will turn on and off automatically).

You will also learn how to use parallel programming (multitasking) to give your wheeled robots two instructions at once.

CHALLENGE 1

What happens on certain cars when it becomes dark? The lights come on.

Can you write a program that will simulate the automatic lights on a car?

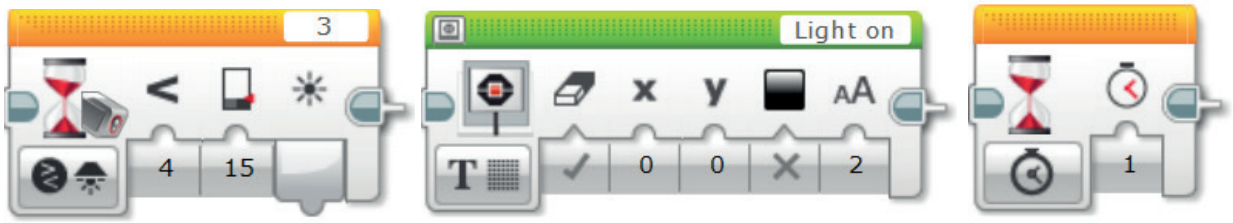
Can you find a 'light bulb' image for the EV3 Brick Display that you can incorporate into your program?

You will need to use the Colour Sensor in order to trigger your light bulb to turn on.

You will need to take ambient light readings from the Port View in order for your program to work properly.

NB: You could explore the possibility of substituting the Display Block with the Brick Status Light Block – or even use both!

Blocks to Consider



Plan your program first. Write it in pseudo-code below:

Activity 4
Student Worksheets

CHALLENGE 2

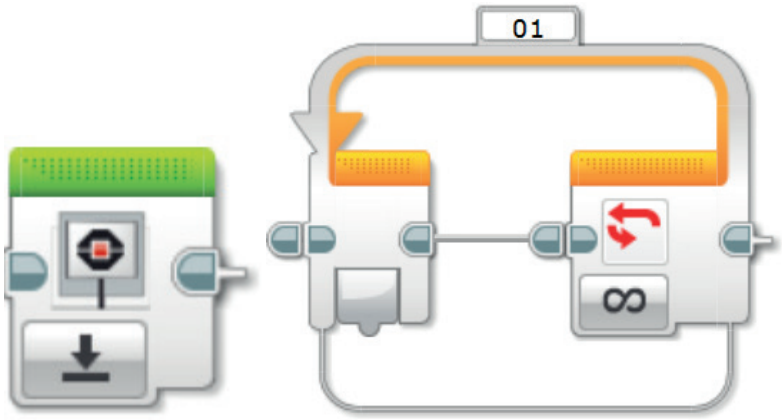
Now that your automatic lights come on successfully, you will need to extend your program so that they switch off when it becomes 'light' again.

In order to do this, you will need to create a program that repeats itself so that you don't need to keep restarting it.

NB: Again, you could explore the possibility of substituting the Display Block with the Brick Status Light Block – or even use both!

Blocks to Consider

Use the same blocks that you used in programming task 1, but also consider using the following:



Plan your program first. Write it in pseudo-code below:

Activity 4
Student Worksheets

CHALLENGE 3

What if you, as a driver, wanted more control over the automatic lights and wanted to be able to switch the lights on and off manually?

Many modern cars have this function, which gives the driver the option to override the automatic program.

Can you simulate this functionality in your program by using parallel programming or multitasking?

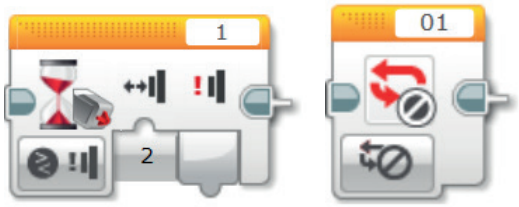
You could use a Touch Sensor in order to simulate the manual switch.

Tip: You will also need to use the Loop Interrupt Block in order to override the automatic control.

NB: Again, you could explore the possibility of substituting the Display Block with the Brick Status Light Block – or even use both!

Blocks to Consider

Use the same blocks that you used in programming tasks 1 and 2, but also consider using the following:



Plan your program first. Write it in pseudo-code below:

Activity 4
Student Worksheets

After a programming activity, it is important to note down your thoughts and observations. Consider the following points and then in the box below record how the activity went.

- How could you improve your program?
- Could your program have been more streamlined? Have you used too many blocks? Is there a more efficient way of building your program?
- What examples of real-world applications could you see your program being used in?

Thoughts and Observations

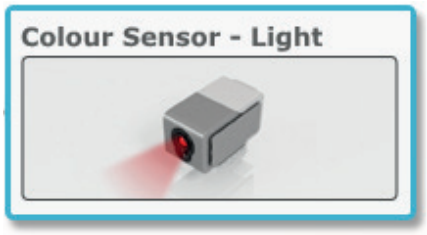
Activity 4
Teacher Notes

ROBOT EDUCATOR TUTORIALS

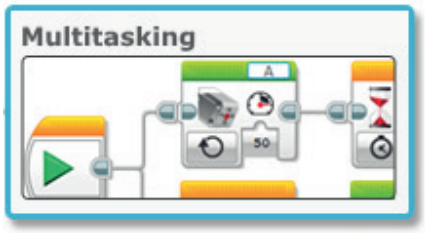
The following Robot Educator Tutorials will help teachers and their students to solve the challenges.

NEW ROBOT EDUCATOR TUTORIALS

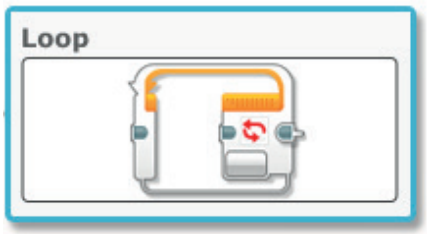
Hardware > Colour Sensor – Light



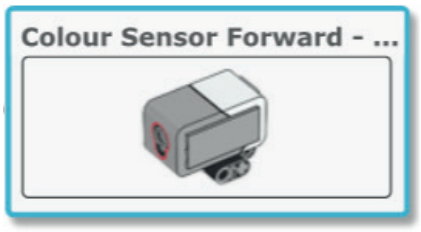
Beyond Basics > Multitasking



Beyond Basics > Loop

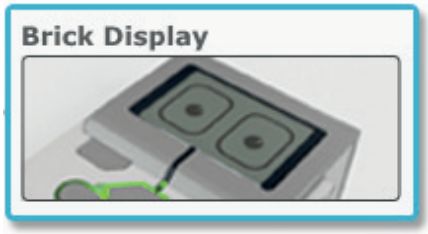


Building Instructions > Colour Sensor Forward – Driving Base



ROBOT EDUCATOR TUTORIALS PREVIOUSLY COVERED

Hardware > Brick Display



Activity 4
Appendix

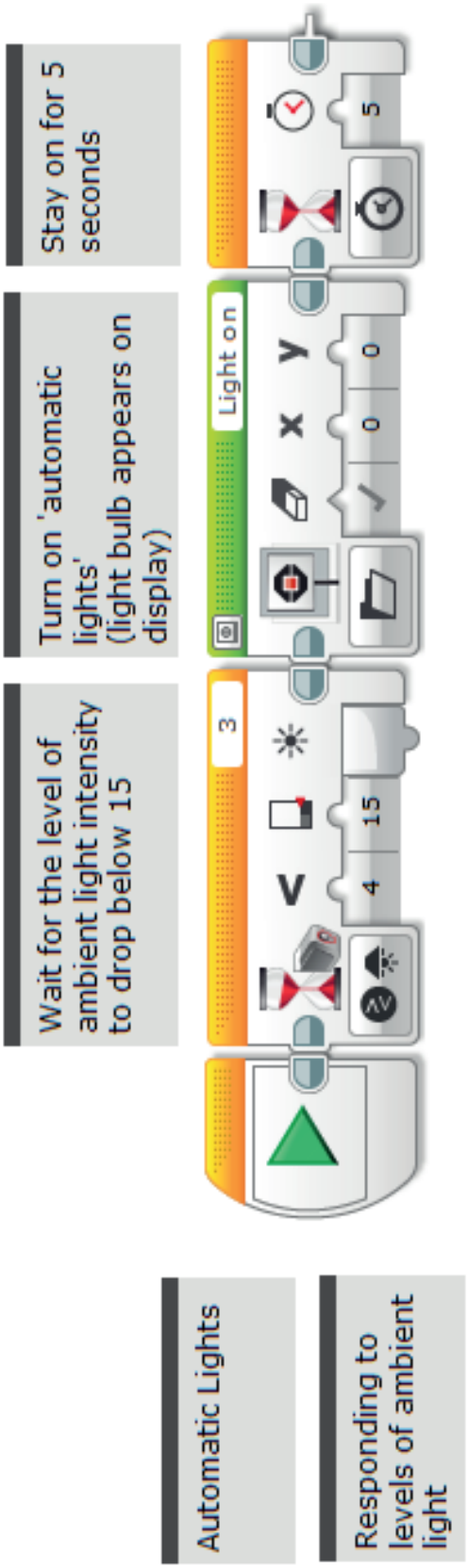
APPENDIX FOR ACTIVITY 4: LARGE IMAGES AND PROGRAMS

Activity 4
Appendix

Activity 4
Appendix

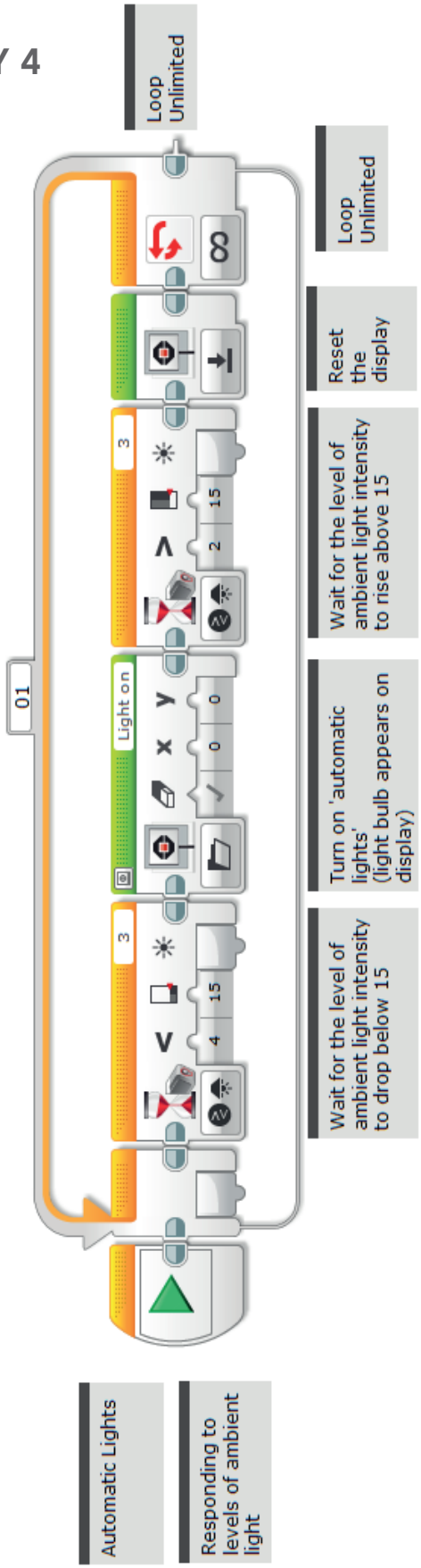
Activity 4
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 4
TAB: MAIN 1



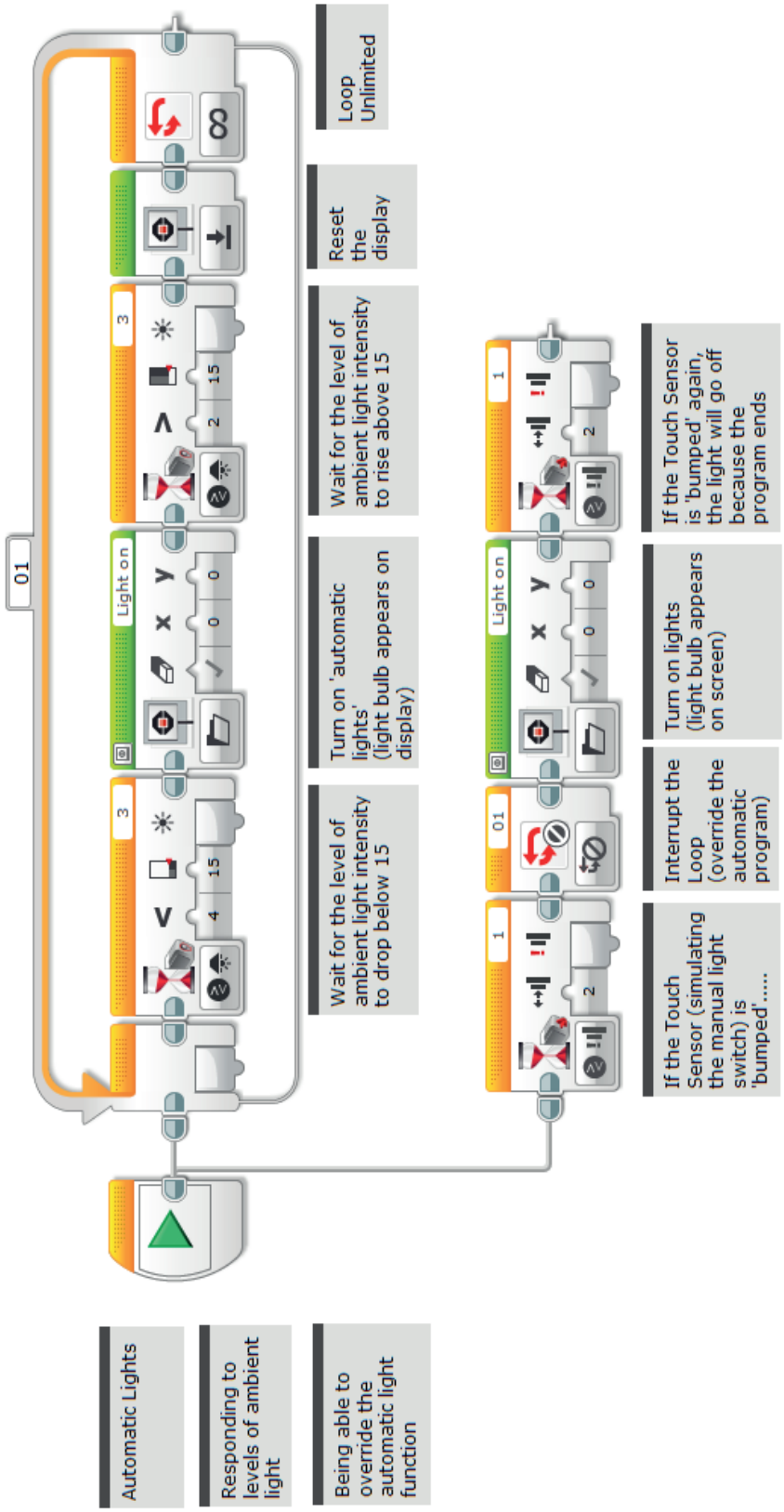
Activity 4
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 4
TAB: MAIN 2



Activity 4
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 4
TAB: MAIN 3



Activity 4
Appendix

Possible ROBOTC Solution
FILENAME: Activity4_1.c

Activity4_1.c

```
1  #pragma config(Sensor, S1,     touchSensor,    sensorEV3_Touch)
2  #pragma config(Sensor, S2,     gyroSensor,     sensorEV3_Gyro)
3  #pragma config(Sensor, S3,     colorSensor,    sensorEV3_Color, modeEV3Color_Ambient)
4  #pragma config(Sensor, S4,     sonarSensor,    sensorEV3_Ultrasonic)
5  #pragma config(Motor,  motorB,          rightMotor,    tmotorEV3_Large, PIDControl, driveRight, encoder)
6  #pragma config(Motor,  motorC,          leftMotor,     tmotorEV3_Large, PIDControl, driveLeft, encoder)
7  //*!!Code automatically generated by 'ROBOTC' configuration wizard !!*//
8
9  /*
10 Create a program that turns on the lights automatically using the Color Sensor
11 when the ambient light level drops below a certain level.
12 */
13
14 task main()
15 {
16   //Wait for the level of ambient light intensity to drop below 15.
17   while(getColorAmbient(colorSensor) >= 15)
18   {
19     //Do nothing while we're waiting.
20     sleep(1);
21   }
22
23   //Display Image "Light on" on the LCD Screen.
24
25   drawBmpfile(0, 127, "Light on");
26
27   //Stay on for 5 seconds.
28   sleep(5000);
29 }
30
```

This code may be subject to change as ROBOTC is updated periodically.

Activity 4
Appendix

Possible ROBOTC Solution
FILENAME: Activity4__2.c

Activity4_2.c*

```
#pragma config(Sensor, S1,     touchSensor,    sensorEV3_Touch)
#pragma config(Sensor, S2,     gyroSensor,     sensorEV3_Gyro)
#pragma config(Sensor, S3,     colorSensor,    sensorEV3_Color, modeEV3Color_Ambient)
#pragma config(Sensor, S4,     sonarSensor,    sensorEV3_Ultrasonic)
#pragma config(Motor,  motorB,    rightMotor,    tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC,    leftMotor,     tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Create an autonomous program that uses the Color Sensor to switch the light on and off
depending on the ambient light levels.
*/

task main()
{
  //Loop forever.
  while(true)
  {

    //Wait for the level of ambient light intensity to drop below 15.
    while(getColorAmbient(colorSensor) >= 15)
    {
      //Do nothing while we're waiting.
      sleep(1);
    }

    //Display Image "Light on" on the LCD Screen.
    drawBmpfile(0, 127, "Light on");

    //Wait for the level of ambient light intensity to rise above 15.
    while(getColorAmbient(colorSensor) <= 15)
    {
      //Do nothing while we're waiting.
      sleep(1);
    }

    //Erase the display to clear the LCD screen.
    eraseDisplay();
  }
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 4
Appendix

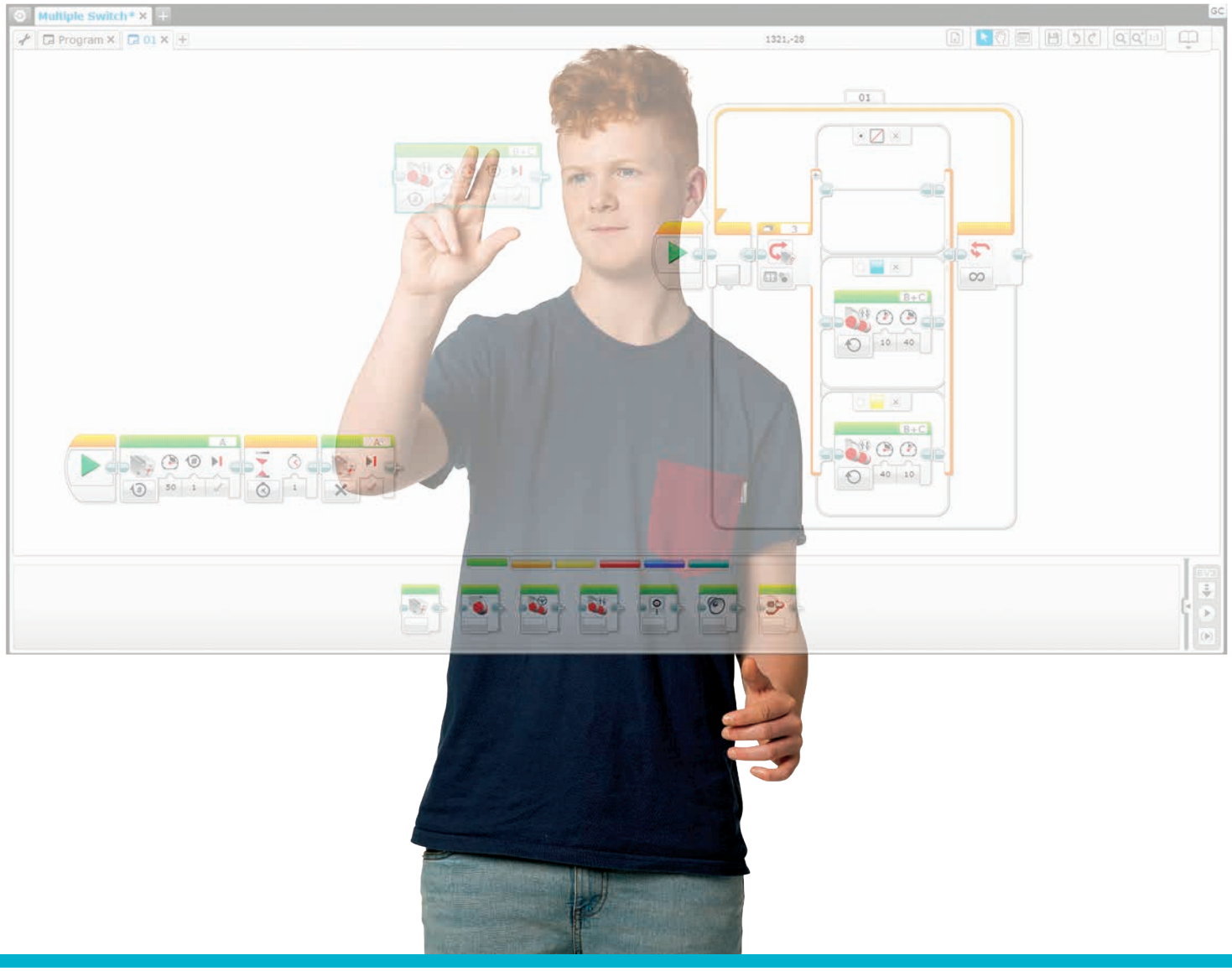
Possible ROBOTC Solution
FILENAME: Activity4_3.c

Activity4_3.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Create an autonomous program that uses the Color Sensor to switch the light on and off
depending on the ambient light levels. Pressing the Touch Sensor overides the loop
turning on the light regardless of the ambient light levels until the Touch Sensor is
pressed again.
NOTE: A sleep in the monitorTouchSensor task is used to replace the bump function in
the EV3 software.
*/

task monitorTouchSensor()
{

  //Wait for Touch Sensor to be touched.
  while(getTouchValue(touchSensor) ==0)
  {
    //Do nothing until touched.
    sleep(10);
  }

  //Stop task "main" (which is task #0)
  stopTask(0);

    //Display Image "Light on" on the LCD Screen.
  drawBmpfile(0, 127, "Light on");

  sleep (1000);

  //Wait for Touch Sensor to be touched.
  while(getTouchValue(touchSensor) == 0)
  {
    //Do nothing until touched.
    sleep(10);
  }
}

task main()
{
  startTask(monitorTouchSensor);

  //Loop forever
  while(true)
  {
    //Wait for the level of ambient light intensity to drop below 15.
    while(getColorAmbient(colorSensor) >= 15)
    {
      //Do nothing while we're waiting.
      sleep(1);
    }

  //Display Image "Light on" on the LCD Screen.
  drawBmpfile(0, 127, "Light on");

    //Wait for the level of ambient light intensity to rise above 15.
    while(getColorAmbient(colorSensor) <= 15)
    {
      //Do nothing while we're waiting.
      sleep(10);
    }

    //Erase the display to clear the LCD screen.
    eraseDisplay();
  }
}
```

This code may be subject to change as ROBOTC is updated periodically.

Activity 5
Traffic Lights and Automated Navigation



Activity 5

Traffic Lights and Automated Rail Systems

During this activity, the students will extend their understanding of the Colour Sensor and draw on their experiences of using the Loop Block by creating a program that simulates a 'stop – go' traffic light system.

In addition to this they will program their wheeled robot to follow a pre-determined path (or line) and they will be introduced to the Switch Block.

Activity 5
Lesson Plan

OUTCOMES

Students will be able to:

- understand that algorithms are capable of carrying out a series of instructions in order
- extend their understanding of Boolean logic and its uses
- use the Wait Block in relation to the Colour Sensor
- understand that the Colour Sensor has several functions and that it can measure a range of parameters
- extend the use of the Colour Sensor to recognise LEGO® system colours and reflected light intensity
- extend their understanding of the Loop Block
- understand the concept of a switch and how to use it for 'true' and 'false' commands

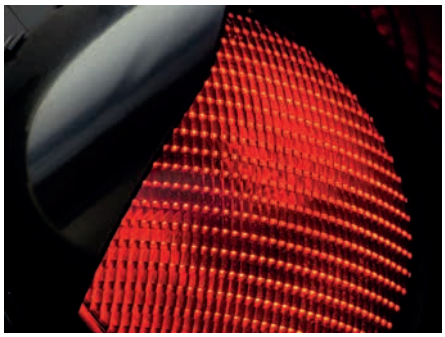
VOCABULARY

input, output, algorithm, wait, Colour Sensor, debug, ambient light, reflected light, loop, Boolean logic, switch

INTRODUCTION

- Explain to the students that they will once again be using the Colour Sensor. This time, though, they will be exploring its capacity to recognise and react to LEGO® system colours. They will also extend their understanding of how this sensor reacts to light. This time they will use reflected light intensity in order to create a program that will drive their wheeled robot along a given track. This will occur over the course of three challenges.
- Tell them that they will use the Colour Sensor to make their wheeled robot more autonomous and to simulate how a 'robot car' might respond to traffic lights, and that they will create a program that will make their wheeled robot drive around a given course or track.
- Ask them to think of real-life situations in which autonomous vehicles might operate, e.g. trains, robots in warehouses and driverless cars, etc.
- Explain that they will be exploring how switches and loops work, and how they could incorporate these into their programs.

Activity 5
Lesson Plan

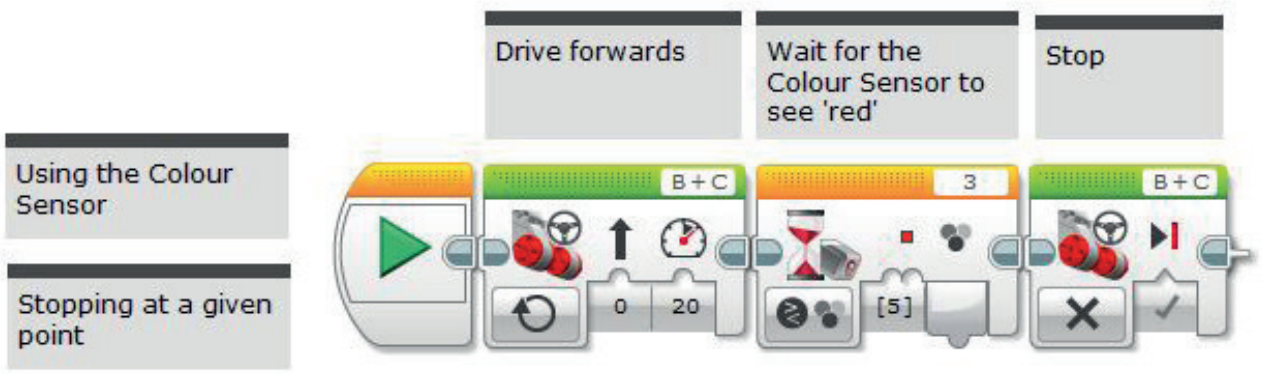


MAIN CHALLENGE 1

- The students will begin exploring the function of the Colour Sensor that recognises LEGO® system colours by programming their wheeled robot to drive along the table and stop at a red 'traffic light'.
- They will need to use the Wait Block in order to do this. Point out that the Wait Block can be configured to be triggered by multiple colors, or just one.
- The students will create a program that uses the Colour Sensor to stop the motors when the sensor sees red.
- They will need to 'tell' the Colour Sensor to recognise the colour red.
- You may wish for the students to further explore this concept and to ensure that their programs work as required by experimenting using a range of colours.

Activity 5
Lesson Plan

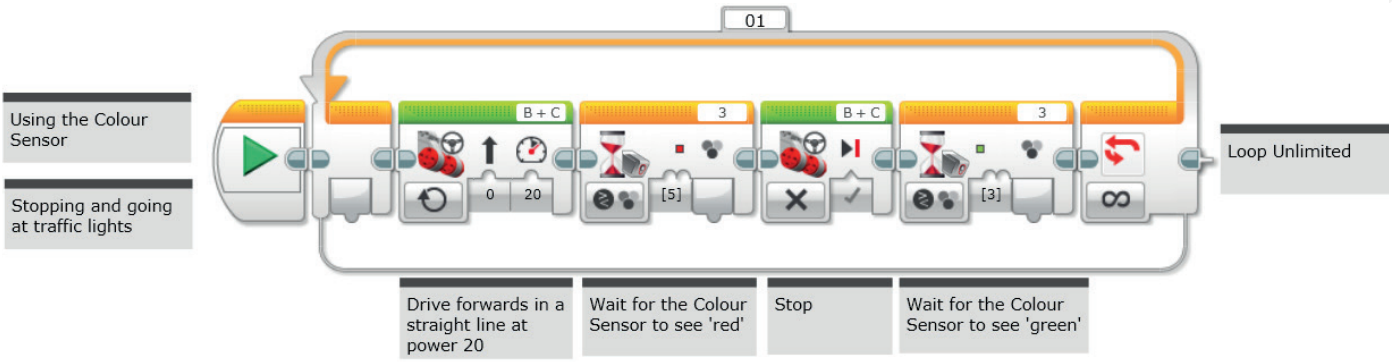
POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 5
TAB: MAIN 1



MAIN CHALLENGE 2

- This challenge will require the students to simulate traffic lights more closely, by having their wheeled robot respond to a series of green and red signals.
- Re-introduce the Loop Block and reiterate the concept of a loop and its ability to make a program run indefinitely until it is manually stopped.
- Putting this inside a loop allows for the possibility of multiple 'traffic lights' along a track.
- Point out to the students that they will need to make sure that all other colours are deselected in order for the Colour Sensor to respond most effectively to the colours they choose (red and green).

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 5
TAB: MAIN 2



Activity 5
Lesson Plan

MAIN CHALLENGE 3

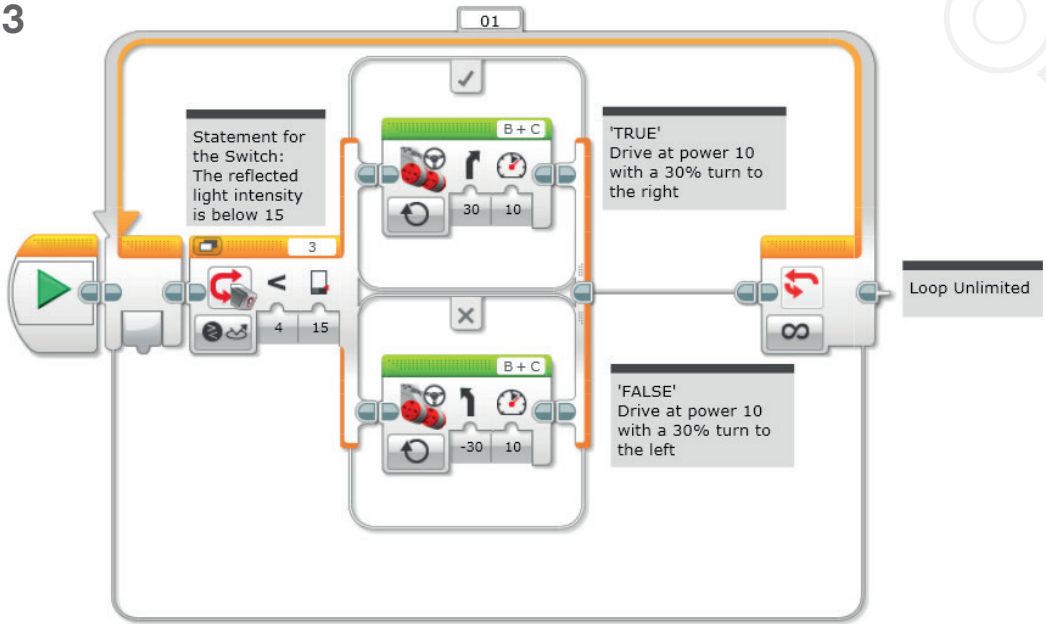
- In this activity, the students will contemplate how an automated vehicle might be guided along a road or track. How can they produce a line-following wheeled robot by using reflected light?
- The students will need to be introduced to the Switch Block, which will operate inside of a loop. Explain that a Switch Block can be used to automate a program that allows the wheeled robot to operate autonomously.
- Explain that the Switch Block is used to control the flow of a program and that the default Switch Block, using the Touch Sensor, is a classic example of Boolean logic. Demonstrate to the students how to change the block to the Colour Sensor and explain the trigger point. The trigger point is used to create the true / false statement (i.e. above the trigger point do one thing, below it do another).
- Point out that in order to create the line-following program, they will need to 'wiggle' the wheeled robot along the line. In other words, the wheeled robot will turn left and then right depending on whether the trigger has been crossed. Point out that the Move Steering Blocks that are used in the program will need to be set to 'On' and not 'On for' (e.g., Seconds, Degrees or Rotations).
- Once the wheeled robot is following the line, can it be improved so that it behaves more like a car, i.e. a straighter line rather than a wiggle?
- You will need to spend some time explaining the concept of a switch and how it is an example of Boolean logic.
- The students will once again use the Colour Sensor, but this time they will need to program it so that it responds to reflected light intensity.
- Tip: They will need to take reflected light intensity readings from the Port View in order to gauge which value to input into the Wait Block.
- Tip: This will work best using black tape on a very light (or white) surface.
- A possible extension from here would be to add a second Colour Sensor and to combine the line-follow and traffic light programs in order to simulate automated passenger services, such as a train system.

Activity 5
Lesson Plan

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 5
TAB: MAIN 3

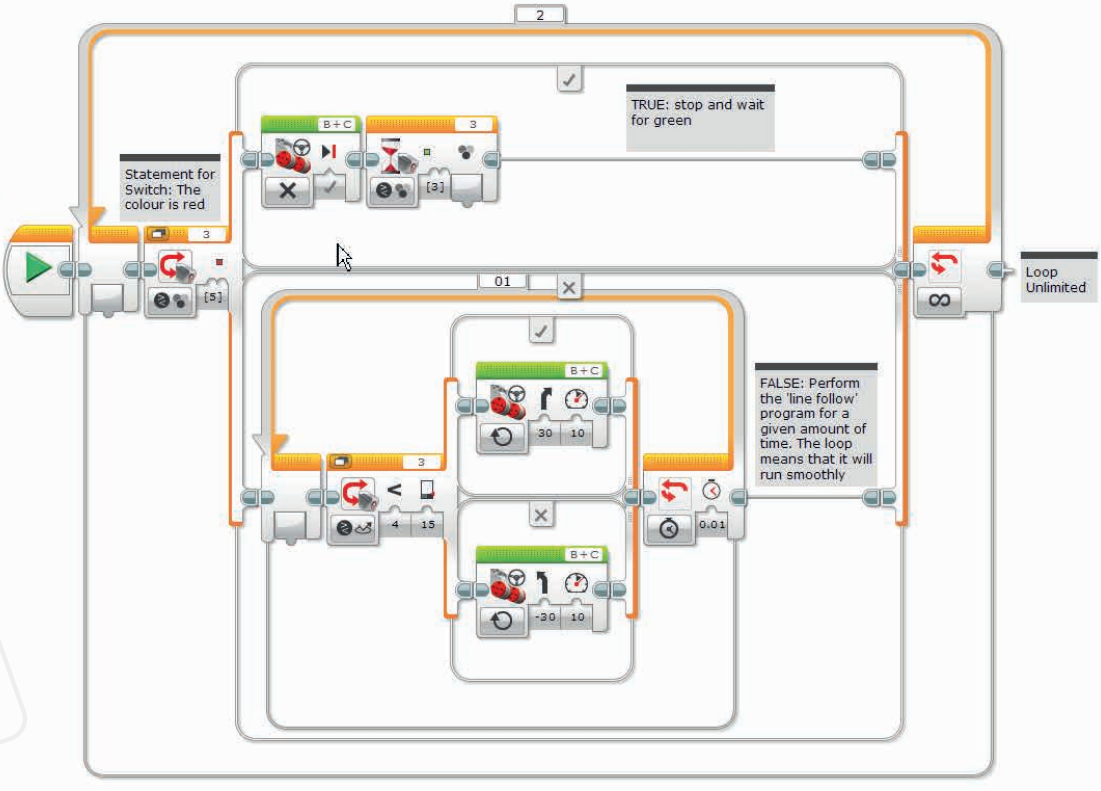
Using the Colour Sensor

Building on understanding of ambient light, using reflected light intensity to follow a line



POSSIBLE SOLUTION FOR EXTENSION ACTIVITY
FILENAME: CS ACTIVITY 5
TAB: EXTENSION

Combine the concepts from the previous challenges to make the car travel along a black line, while stopping at traffic lights along the way



Activity 5
Student Worksheets

CHALLENGES FOR TODAY

Today you are going to use the Colour Sensor and the Switch Block in order to make decisions using Boolean logic. These two blocks will allow the wheeled robot to make choices based on the colours that it sees.

CHALLENGE 1

When driving a car, it is important to recognise and abide by the rules of the road.

What should a motorist do when they are approaching traffic lights?

If cars were automated, they would need to use some sort of sensor in order to recognise and respond to traffic lights automatically.

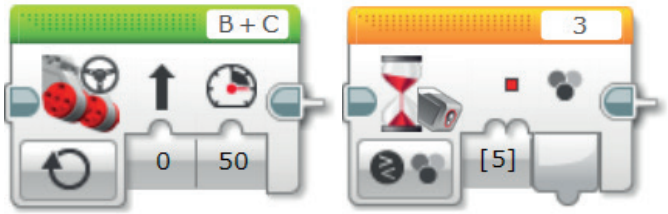
For this challenge, you will need to program your wheeled robot so that it responds to a 'stop' command. Which colour should you use in your program?

Use the Wait Block to program the Colour Sensor so that it recognises red and stops the wheeled robot.

Refine your program by making your wheeled robot stop at an appropriate distance from the 'traffic lights'.

Make sure that the wheeled robot is only responding to red by eliminating the other colours.

Blocks to Consider



Plan your program first. Write it in pseudo-code below:

Activity 5
Student Worksheets

CHALLENGE 2

Now that you have programmed your wheeled robot to stop at traffic lights, you need to make sure it goes again!

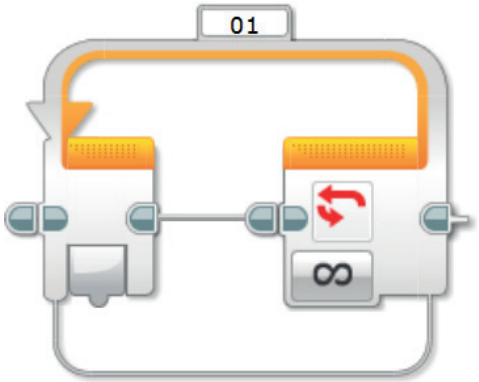
Create a program that uses the Colour Sensor to recognise and respond to both 'stop' and 'go' commands.

Which colours will you use?

What if there were multiple sets of traffic lights along the street? Can you change your program so that the 'stop – go' algorithm is repeatable?

Blocks to Consider

Use the same blocks that you used in programming task 1, but also consider using the following:



Plan your program first. Write it in pseudo-code below:

Activity 5
Student Worksheets

CHALLENGE 3

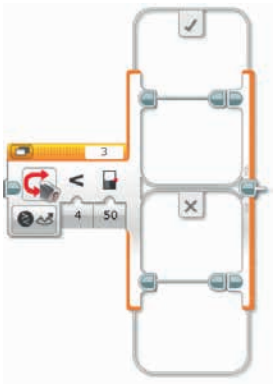
For this challenge, you will begin to make your wheeled robot even more autonomous. You will need to alter your model slightly so that the Colour Sensor is pointing downwards. Imagine that cars can drive on 'autopilot' along a given route, a little bit like driverless trains.

Your challenge is to program your wheeled robot to do just that. You will need to create a program that recognises and responds to the black line that has been laid out for you. Your wheeled robot will need to travel along that line without losing contact with it. You will need to constantly debug your program in order to make your wheeled robot travel as smoothly as possible along the line.

Tip: In the Port View, you will need to change the Colour Sensor settings so that it measures reflected light intensity.

Blocks to Consider

Use the same blocks that you used in programming tasks 1 and 2, but also consider using the following:



Plan your program first. Write it in pseudo-code below:

Activity 5
Student Worksheets

After a programming activity, it is important to note down your thoughts and observations. Consider the following points and then in the box below record how the activity went.

- How could you improve your program?
- Could your program have been more streamlined? Have you used too many blocks? Is there a more efficient way of building your program?
- What examples of real-world applications could you see your program being used in?

Thoughts and Observations

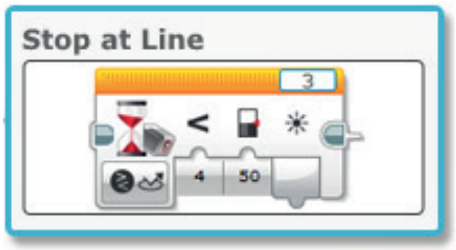
Activity 5
Teacher Notes

ROBOT EDUCATOR TUTORIALS

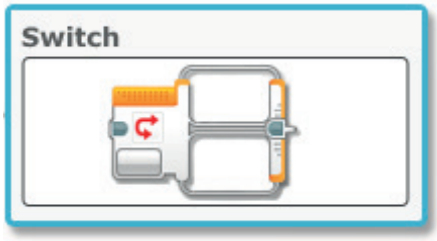
The following Robot Educator Tutorials will help teachers and their students to solve the challenges.

NEW ROBOT EDUCATOR TUTORIALS

Basics > Stop at Line

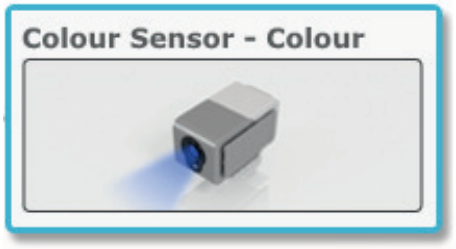


Beyond Basics > Switch

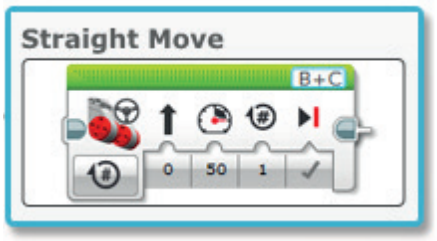


ROBOT EDUCATOR TUTORIALS PREVIOUSLY COVERED

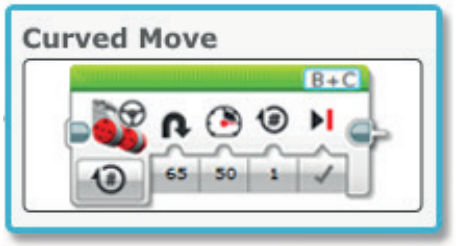
Hardware > Colour Sensor – Colour



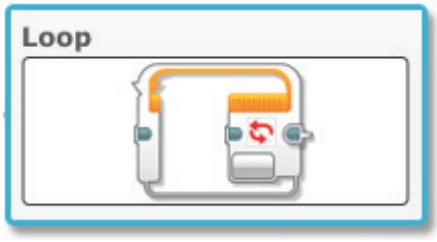
Basics > Straight Move



Basics > Curved Move



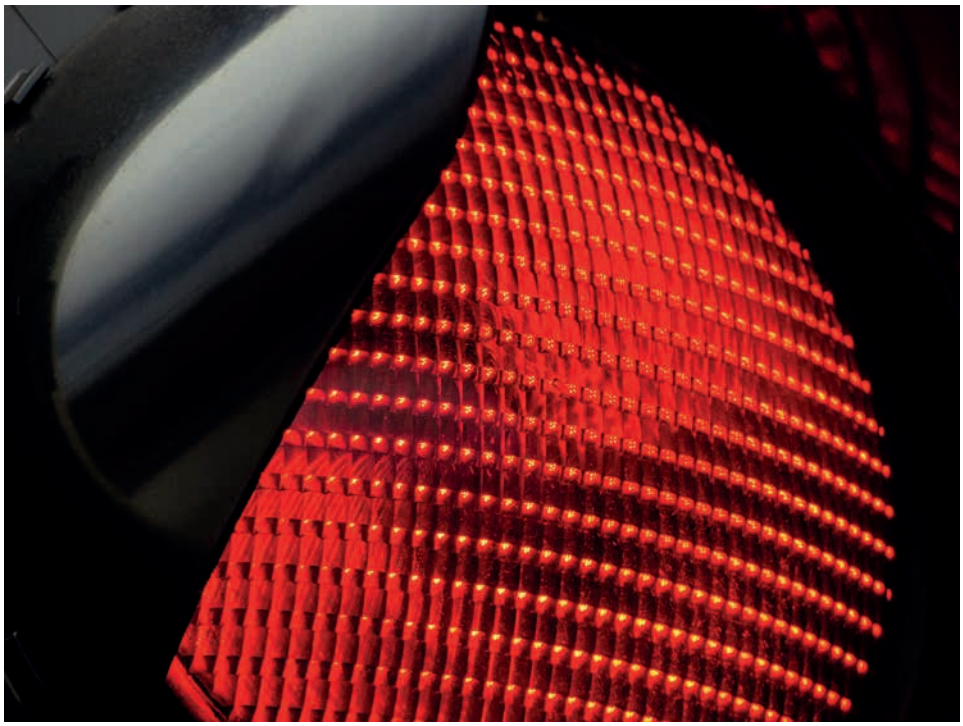
Beyond Basics > Loop



Activity 5
Appendix

APPENDIX FOR ACTIVITY 5: LARGE IMAGES AND PROGRAMS

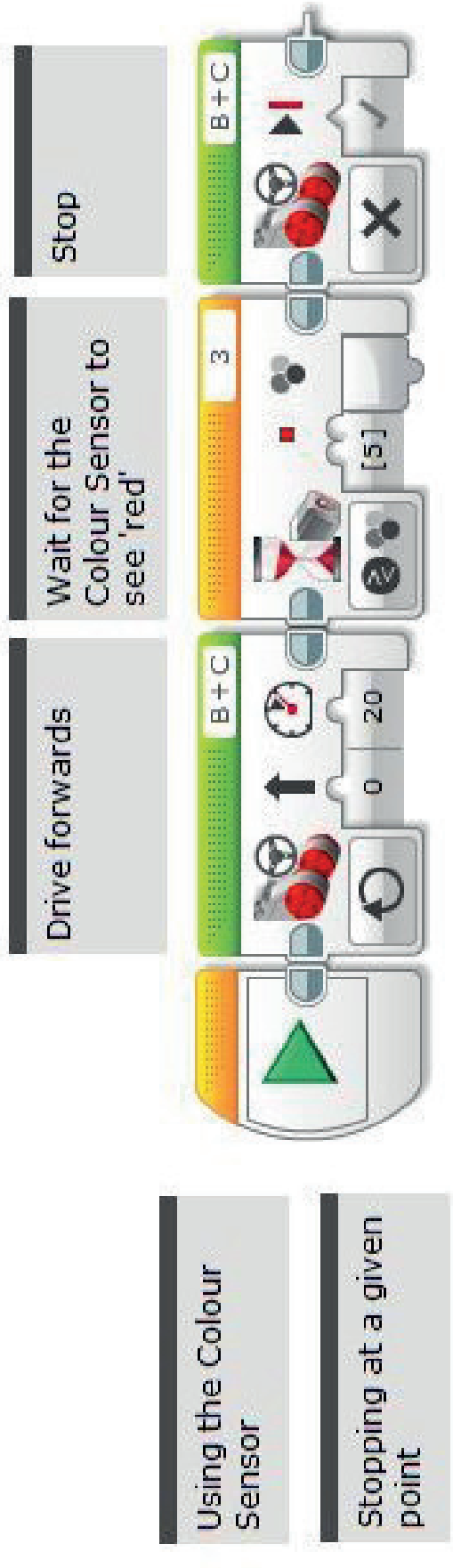
Activity 5
Appendix



Activity 5
Appendix

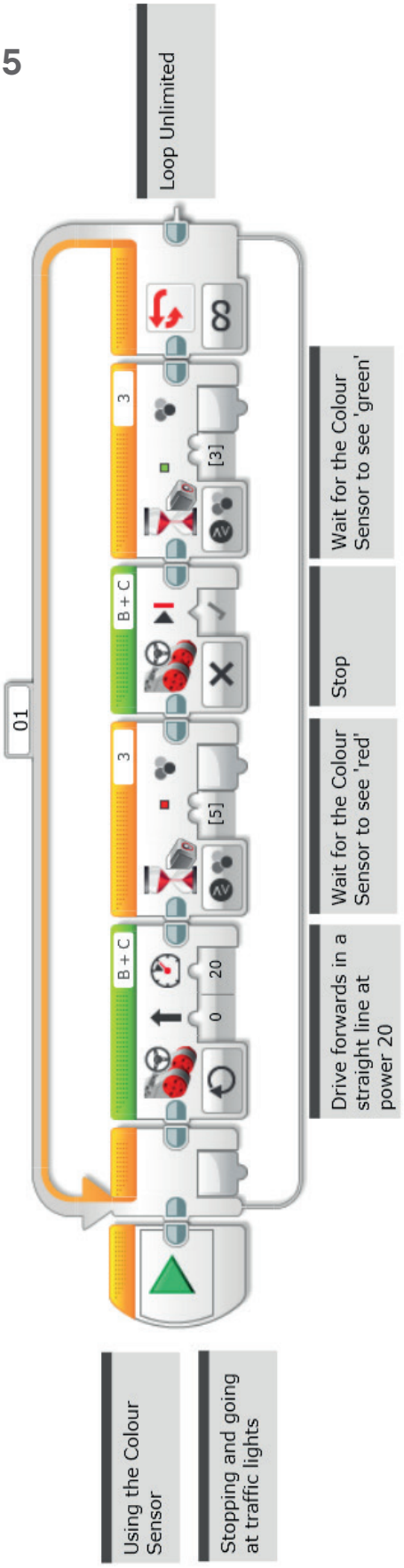
Activity 5
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 5
TAB: MAIN 1



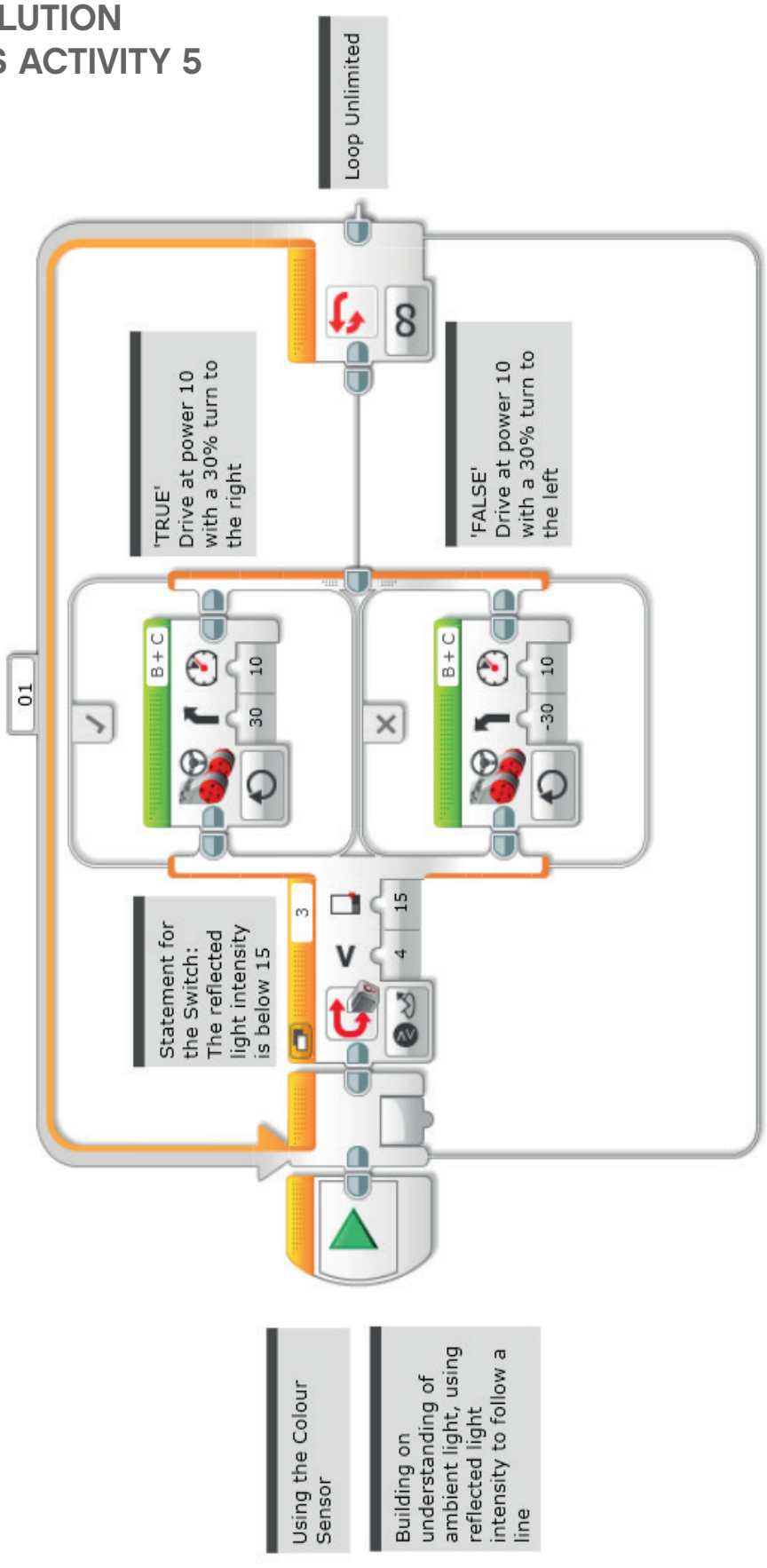
Activity 5
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 5
TAB: MAIN 2



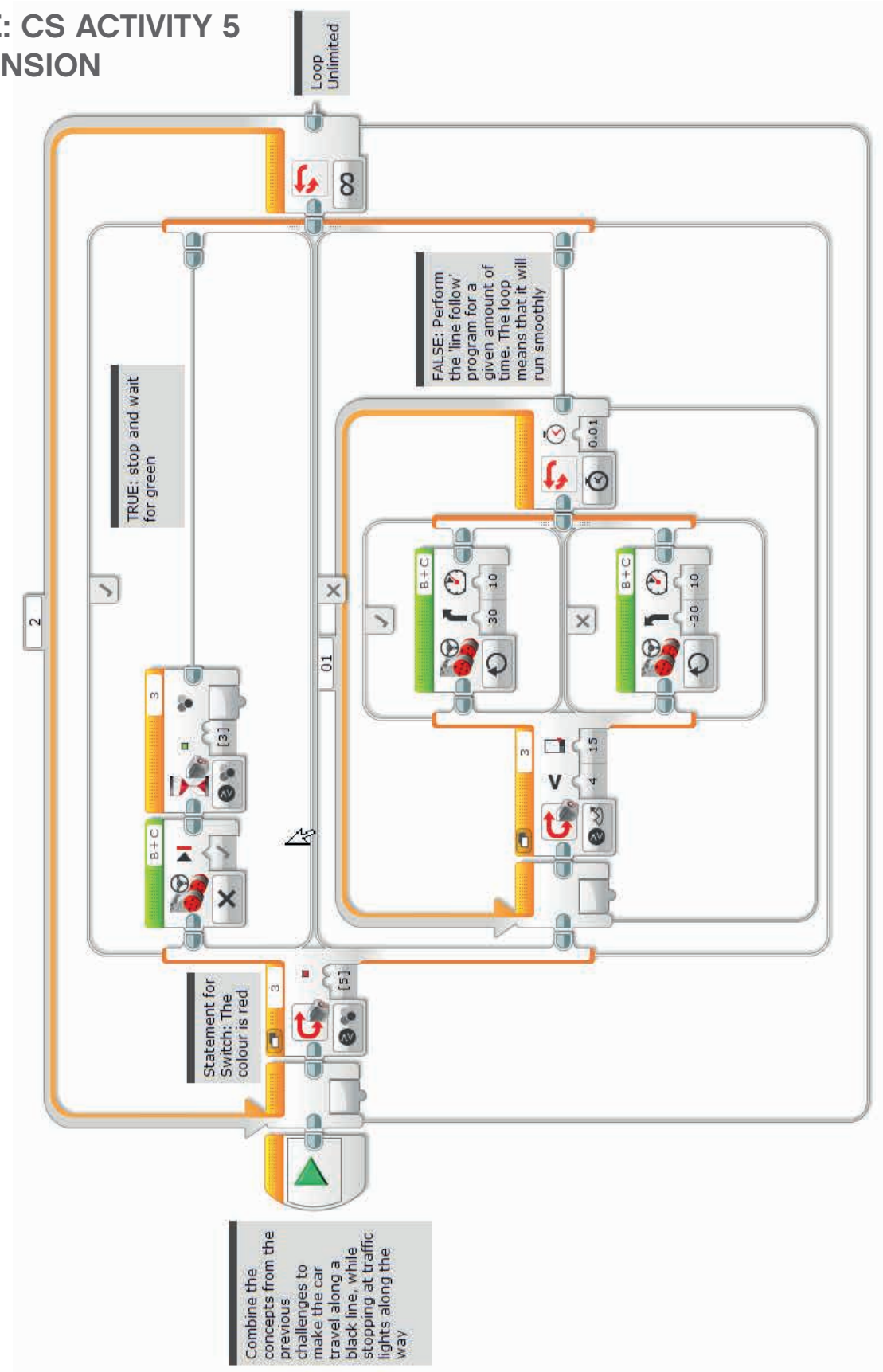
Activity 5
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 5
TAB: MAIN 3



Activity 5
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 5
TAB: EXTENSION



Activity 5
Appendix

Possible ROBOTC Solution
FILENAME: Activity5_1.c

Activity5_1.c*

```
#pragma config(Sensor, S1,     touchSensor,    sensorEV3_Touch)
#pragma config(Sensor, S2,     gyroSensor,     sensorEV3_Gyro)
#pragma config(Sensor, S3,     colorSensor,    sensorEV3_Color, modeEV3Color_Color)
#pragma config(Sensor, S4,     sonarSensor,    sensorEV3_Ultrasonic)
#pragma config(Motor,  motorA,     armMotor,      tmotorEV3_Large, PIDControl, encoder)
#pragma config(Motor,  motorB,     leftMotor,     tmotorEV3_Large, PIDControl, driveLeft, encoder)
#pragma config(Motor,  motorC,     rightMotor,    tmotorEV3_Large, PIDControl, driveRight, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Create a program that drives the robot forward until the Color Sensor sees red.
The robot then stops.
*/

task main()
{
  //Set motor speed at 20% (Drive Forwards).
  setMotorSpeed(motorB, 20);
  setMotorSpeed(motorC, 20);

  //Loop while the Color Sensor does not see red.
  while(getColorName(colorSensor) != colorRed)
  {
    //Keep driving while the Color Sensor does not see red.
    sleep(10);
  }

  //Set motor speed to 0% (Stop).
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
}
```

This code may be subject to change as ROBOTC is updated periodically.

Activity 5
Appendix

Possible ROBOTC Solution
FILENAME: Activity5__2.c

Activity5_2.c*

```
#pragma config(Sensor, S1,     touchSensor,    sensorEV3_Touch)
#pragma config(Sensor, S2,     gyroSensor,     sensorEV3_Gyro)
#pragma config(Sensor, S3,     colorSensor,    sensorEV3_Color, modeEV3Color_Color)
#pragma config(Sensor, S4,     sonarSensor,    sensorEV3_Ultrasonic)
#pragma config(Motor,  motorB,   rightMotor,    tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC,   leftMotor,     tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Create a program where the robot drives forward stopping at the red brick and
moving off when the Color Sensor sees the green brick. This happens continuously.
*/

task main()
{
  //Repeat Forever.
  while(true)
  {
    //Set motor speed to 20%. (Drive Forwards).
    setMotorSpeed(motorB, 20);
    setMotorSpeed(motorC, 20);

    //Loop while the sensor does not see red.
    while(getColorName(colorSensor) != colorRed)
    {
      //Keep driving while the sensor does not see red.
      sleep(10);
    }

    //Set the motor speed to 0% (Stop).
    setMotorSpeed(motorB, 0);
    setMotorSpeed(motorC, 0);

    //After we've seen red, we need to wait until we see green.
    while(getColorName(colorSensor) != colorGreen)
    {
      sleep(10);
      //Do nothing while the sensor does not see green.
      //When the robot sees green loop back to start of program.
    }
  }
} //End of program - turn off all motors automatically.

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 5
Appendix

Possible ROBOTC Solution
FILENAME: Activity5_3.c

Activity5_3.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Create a line following program that uses the Color Sensor in reflected mode to
'wiggle' along a line.
NOTE: ***THERE ARE SLIGHT MOTOR POWER DIFFERENCES BETWEEN ROBOTC & EV3 PROGRAMS***
*/

task main()
{
  //Repeat Forever.
  while(true)
  {
    //If we see a dark value.
    if(getColorReflected(colorSensor) < 15)
    {
      //Steer to the right.
      setMotorSpeed(motorB, 30);
      setMotorSpeed(motorC, 12);
    }
    //If we see a light value (or not dark).
    else
    {
      //Steer to the left.
      setMotorSpeed(motorB, 12);
      setMotorSpeed(motorC, 30);
    }
  }
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 6
Reversing Beeps

Activity 6

Reversing Beeps

In this activity, the students will be re-introduced to the Ultrasonic Sensor and the concept of parking sensors (sensors that emit a warning sound as they approach an obstacle).

As the students work their way through the challenges, they will create programs that will simulate the parking sensors found on modern cars. Their final challenge will require them to create a program that emits warning 'beeps', which will become faster as their wheeled robots get closer to an object.

The students will need to use a range of programming techniques and blocks, including parallel programming (multitasking), loops and switches. They will also be introduced to the Maths Block and Data Wires.

Online resources:

You may wish to find some online videos that demonstrate how parking sensors work.

Activity 6
Lesson Plan

OUTCOMES

Students will be able to:

- understand that algorithms are capable of carrying out a series of instructions in order
- extend their understanding of Boolean logic and its uses
- use the Wait Block in relation to the Colour Sensor
- understand that the Ultrasonic Sensor works by 'bouncing' waves off of objects and that it can be programmed to respond to distance
- program their wheeled robots to reverse, emit a sound dependent on its distance from an object and stop at a given distance from that object
- extend their understanding of the Loop Block
- understand the concept of a switch and how to use it for 'true' and 'false' commands
- understand the Maths Block and its functions
- understand that readings can be taken from one block and passed to another through the use of Data Wires

VOCABULARY

input, output, algorithm, wait, Ultrasonic Sensor, debug, loop, Boolean logic, switch, Maths Block, Data Wire, interrupt

INTRODUCTION

- Explain to the students that over the course of this activity they will be writing and developing programs that will closely simulate rear parking sensors.
- Point out that rear parking sensors work by emitting a beeping noise that starts slowly but gets quicker as the car gets closer to an object (usually a wall). This is made possible through the use of an ultrasonic sensor.
- Tell the students that in this activity, they will use the same Ultrasonic Sensor that they used in Activity 1, but that this time they will greatly extend its use and their understanding of how it works.
- How does this sensor work? Ask the students to explain it. They should explain that ultrasonic waves are sent out via the 'eye' of the sensor and that these bounce off an object. The amount of time that it takes for the wave to come back determines the distance of the sensor from the object.
- You may wish to take some time to let the students experiment with the sensor and to work out how accurate it is. Does speed affect the accuracy of the sensor?

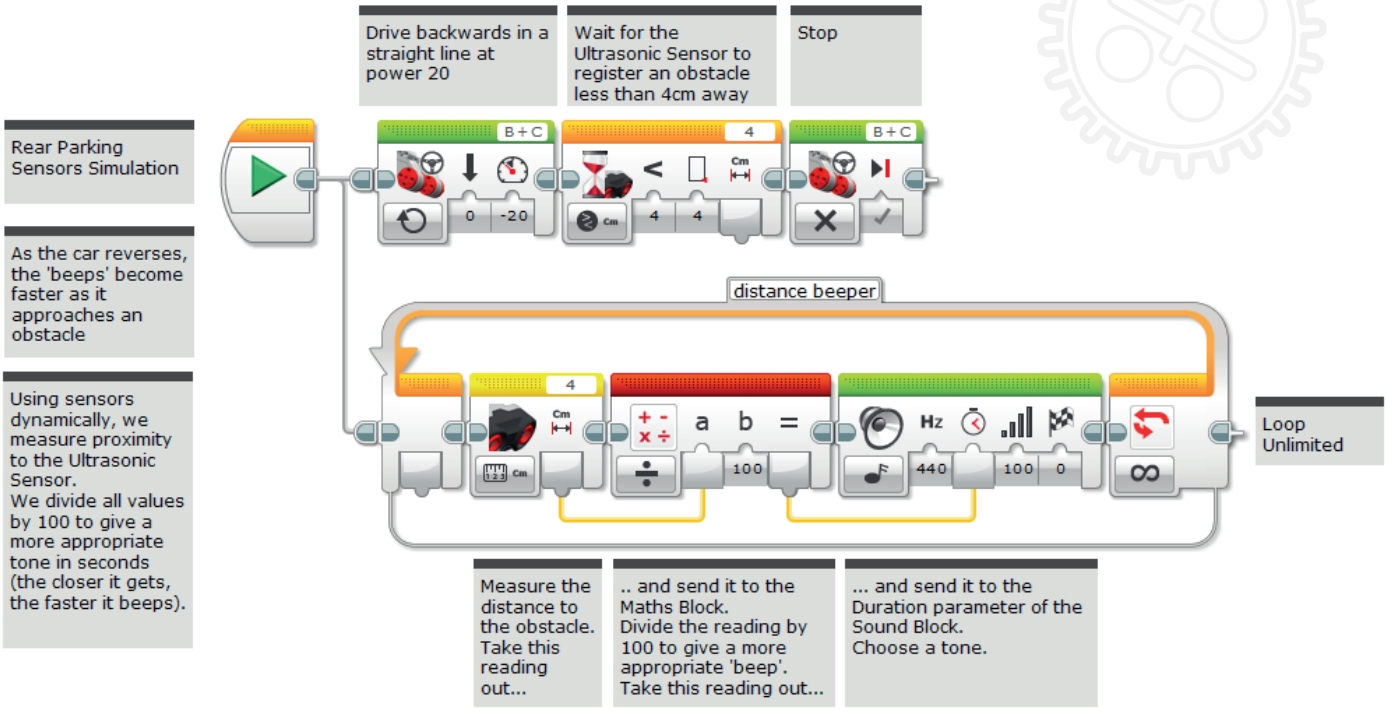
Activity 6
Lesson Plan

MAIN CHALLENGE 1

- This activity will require the students to create a program that makes their wheeled robot emit a beeping sound as it reverses towards an obstacle.
- The closer the wheeled robot gets to the object, the faster the beeps should become. The wheeled robot should stop automatically as it reaches a certain distance from the obstacle.
- The beeping sound (and particularly the speed of the beeps) is made by using the Maths Block and Data Wires.
- You can see the explanation of how this works in the example program below.
- Take some time to show the students the Maths Block and how it works.
- The students will need to attach the Ultrasonic Sensor to the rear of the base model.
- This activity revisits parallel programming (multitasking). You may need to remind the students of how to drag a Data Wire from the Start Block in order to enable this.

Activity 6
Lesson Plan

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 6
TAB: MAIN 1

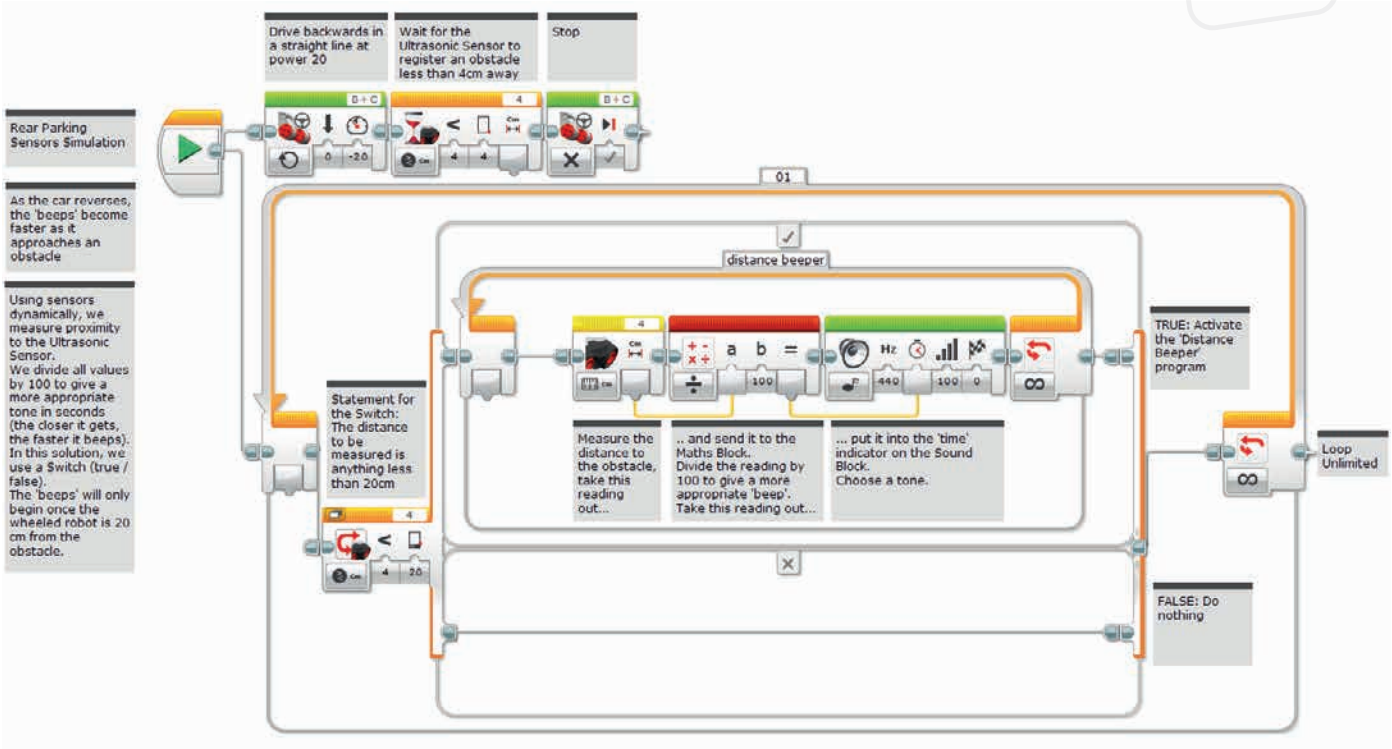


MAIN CHALLENGE 2

- This activity builds directly upon Activity 1.
- Ask the students what is 'wrong' with the program so far.
- They will need to point out that the sound emitted from the wheeled robot starts straight away, regardless of how far away from the obstacle it is.
- How can they improve this and make their wheeled robot behave more like a real car?
- The sound / beeping could start at a given distance away from the obstacle.
- In order to do this, they will need to use a Switch Block with a true / false statement.

Activity 6
Lesson Plan

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 6
TAB: MAIN 2

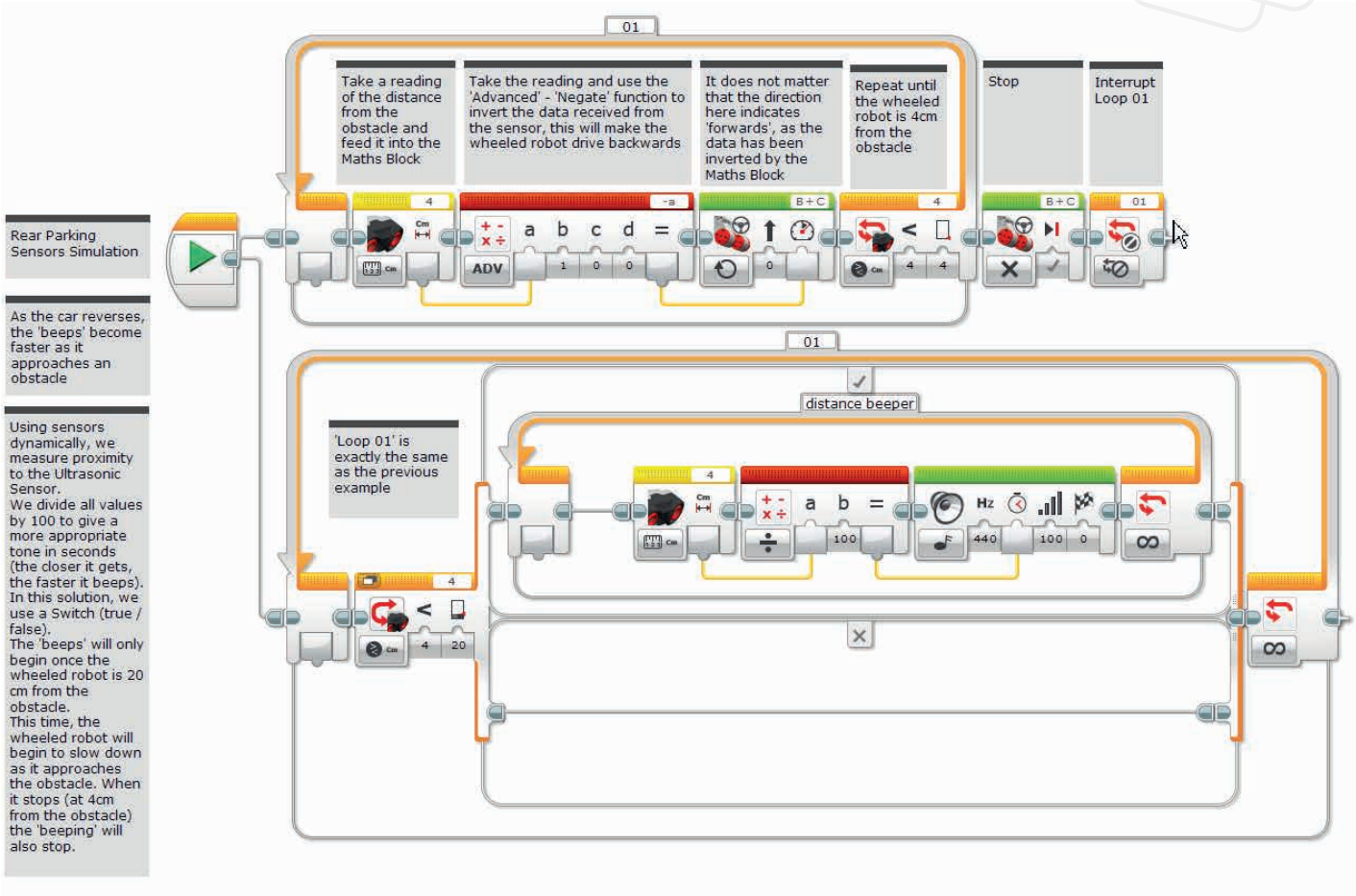


MAIN CHALLENGE 3

- The students' wheeled robot should now be behaving very similarly to cars when reversing.
- The beeping sound should begin at a certain distance from the obstacles and the wheeled robot should stop at a given distance.
- Ask the students how they could further improve the program.
- Point out that when cars reverse and stop, they do two things that the students' wheeled robots currently don't do – they slow down as they approach an obstacle, and that when they brake, the warning beeps stop.
- The students will now need to further extend their programs in order to simulate these two additional behaviours.
- In order to do this, they will need to use another Maths Block and map the Ultrasonic Sensor to speed in order to slow the wheeled robot down as its distance from the obstacle decreases.
- They will also need to use a Loop Interrupt Block in order to stop the sound as the wheeled robot stops.

Activity 6
Lesson Plan

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 6
TAB: MAIN 3



CLASS DISCUSSION

- This is a good opportunity for all of the groups to share their programming simultaneously.
- You could have all of the wheeled robots reversing towards the same obstacle at once in order to see which ones most accurately simulate a car.
- Have the students make a video of their wheeled robot in order to evaluate them at a later date.

Activity 6
Student Worksheets

CHALLENGES FOR TODAY

Today you are going to explore how the yellow sensor blocks are used in conjunction with the Maths Block. You will also use the Loop Block.

CHALLENGE 1

Over the course of three challenges, you will be programming your wheeled robot to simulate a car's parking sensor.

What happens when some cars reverse? There is a beeping sound, which becomes quicker as the car gets closer to an obstacle.

Can you devise a program that drives your wheeled robot backwards, emits beeping noises as it approaches an obstacle and then stops automatically at a set distance away from the object?

Tip 1: You will need to use parallel programming (multitasking).

Tip 2: You will need to use what you have learnt about the Maths Block and Data Wires, in order to increase the frequency of the beeps as your wheeled robot gets closer to the obstacle.

Blocks to Consider

Plan your program first. Write it in pseudo-code below:

Activity 6
Student Worksheets

CHALLENGE 2

What have you noticed about your program and in particular the beeping sounds?

They should become quicker as your wheeled robot approaches the obstacle.

However, in real life, the warning sounds only begin when the vehicle is a certain distance from an obstacle.

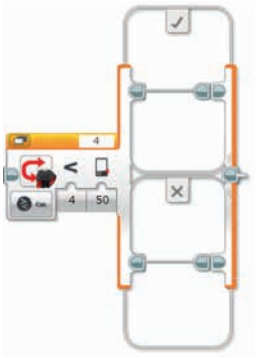
Can you simulate this in your program?

You will need to build on the program you have already created, but alter it slightly so that the beeping begins at a given distance from the obstacle.

Tip: You will need to utilise a true/false statement and Boolean logic. Which programming block do you need to use for this?

Blocks to Consider

Use the same blocks that you used in programming task 1, but also consider using the following:



Plan your program first. Write it in pseudo-code below:

Activity 6
Student Worksheets

CHALLENGE 3

By now your wheeled robot should be simulating rear parking sensors more and more.

Now it's time to take your programming one step further.

You will need to add two more features:

1. Can you make the beeping sound stop when your wheeled robot stops at a given distance away from the obstacle?
2. Can you make your wheeled robot slow down as the beeping sound starts?

Tip 1: In order for the beeping sound to stop, you will need to interrupt the loop.

Tip 2: You will need to map speed to distance by using a second Maths Block somewhere in your program. Can you work out where?

Blocks to Consider

Use the same blocks that you used in programming tasks 1 and 2, but also consider using the following:

Plan your program first. Write it in pseudo-code below:

Activity 6
Student Worksheets

After a programming activity, it is important to note down your thoughts and observations. Consider the following points and then in the box below record how the activity went.

- How could you improve your program?
- Could your program have been more streamlined? Have you used too many blocks? Is there a more efficient way of building your program?
- What examples of real-world applications could you see your program being used in?

Thoughts and Observations

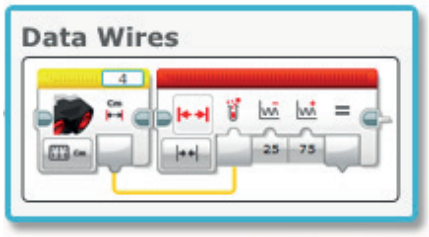
Activity 6
Teacher Notes

ROBOT EDUCATOR TUTORIALS

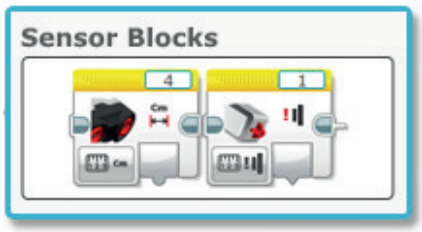
The following Robot Educator Tutorials will help teachers and their students to solve the challenges.

NEW ROBOT EDUCATOR TUTORIALS

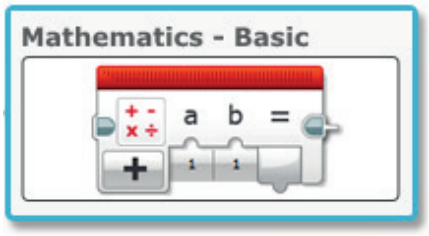
Beyond Basics > Data Wires



Beyond Basics > Sensor Blocks

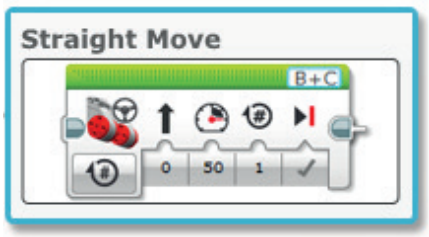


Beyond Basics > Mathematics – Basic

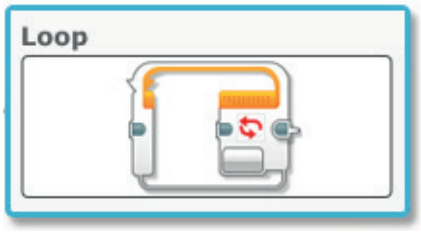


ROBOT EDUCATOR TUTORIALS PREVIOUSLY COVERED

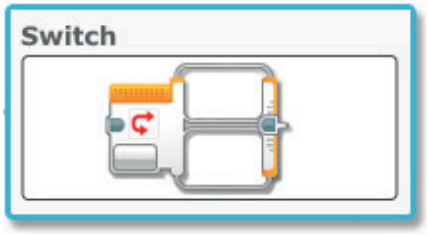
Basics > Straight Move



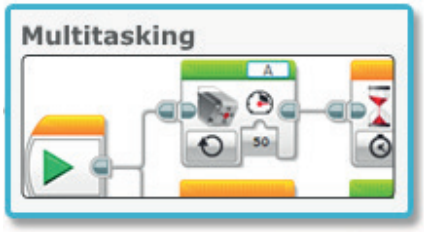
Beyond Basics > Loop



Beyond Basics > Switch



Beyond Basics > Multitasking



APPENDIX FOR ACTIVITY 6: LARGE IMAGES AND PROGRAMS

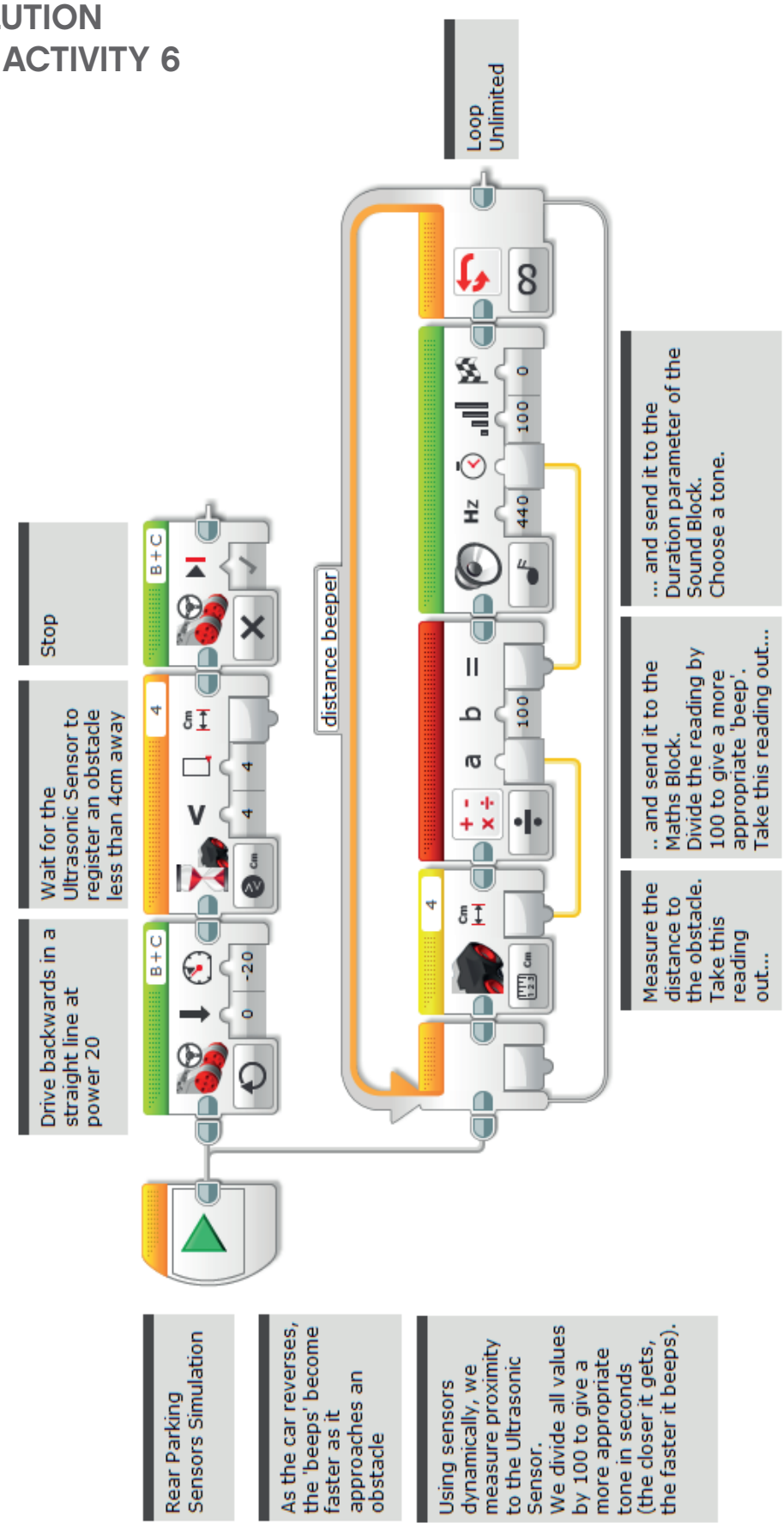
Activity 6
Appendix

Activity 6
Appendix

Activity 6
Appendix

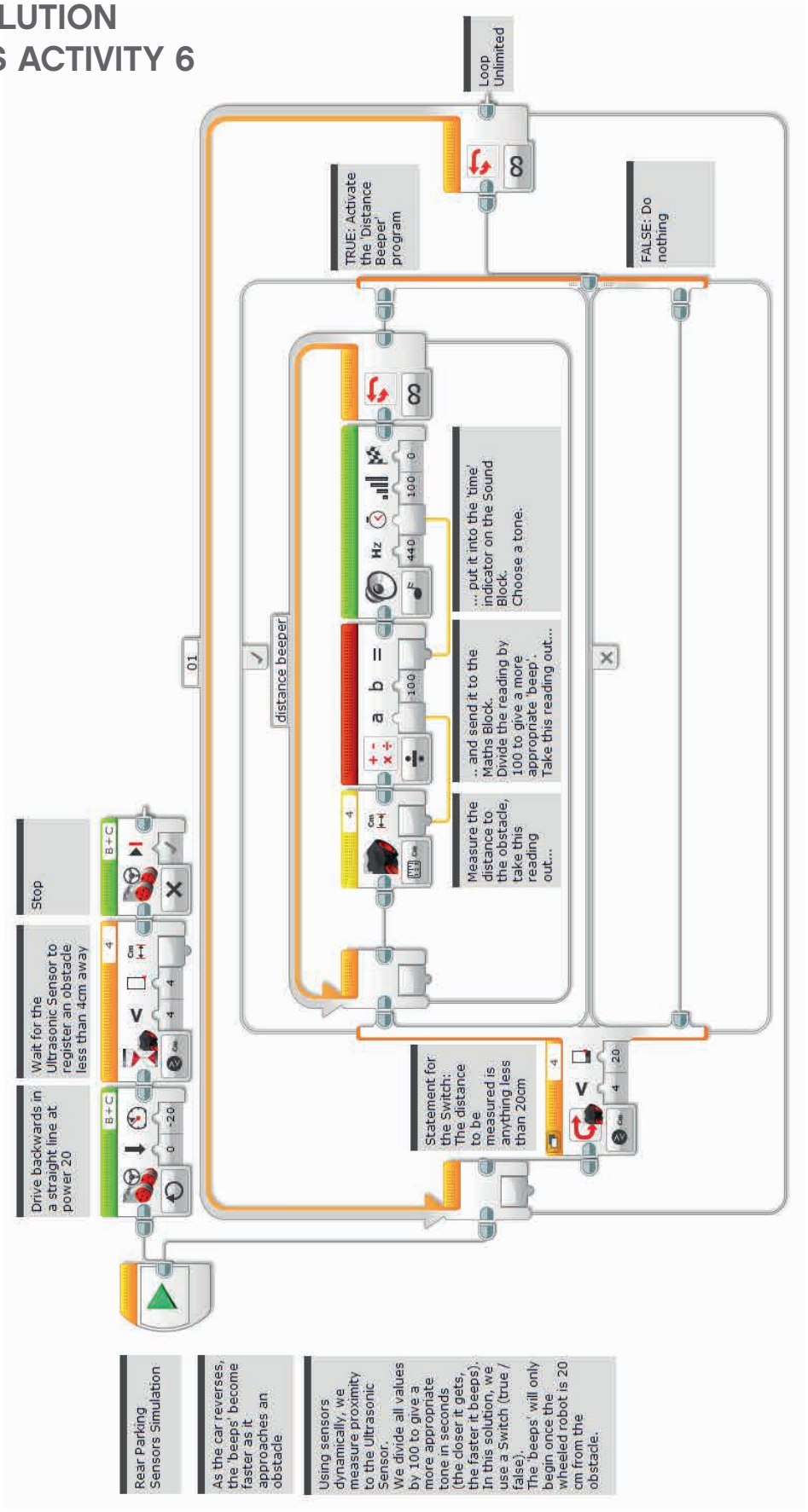
Activity 6
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 6
TAB: MAIN 1



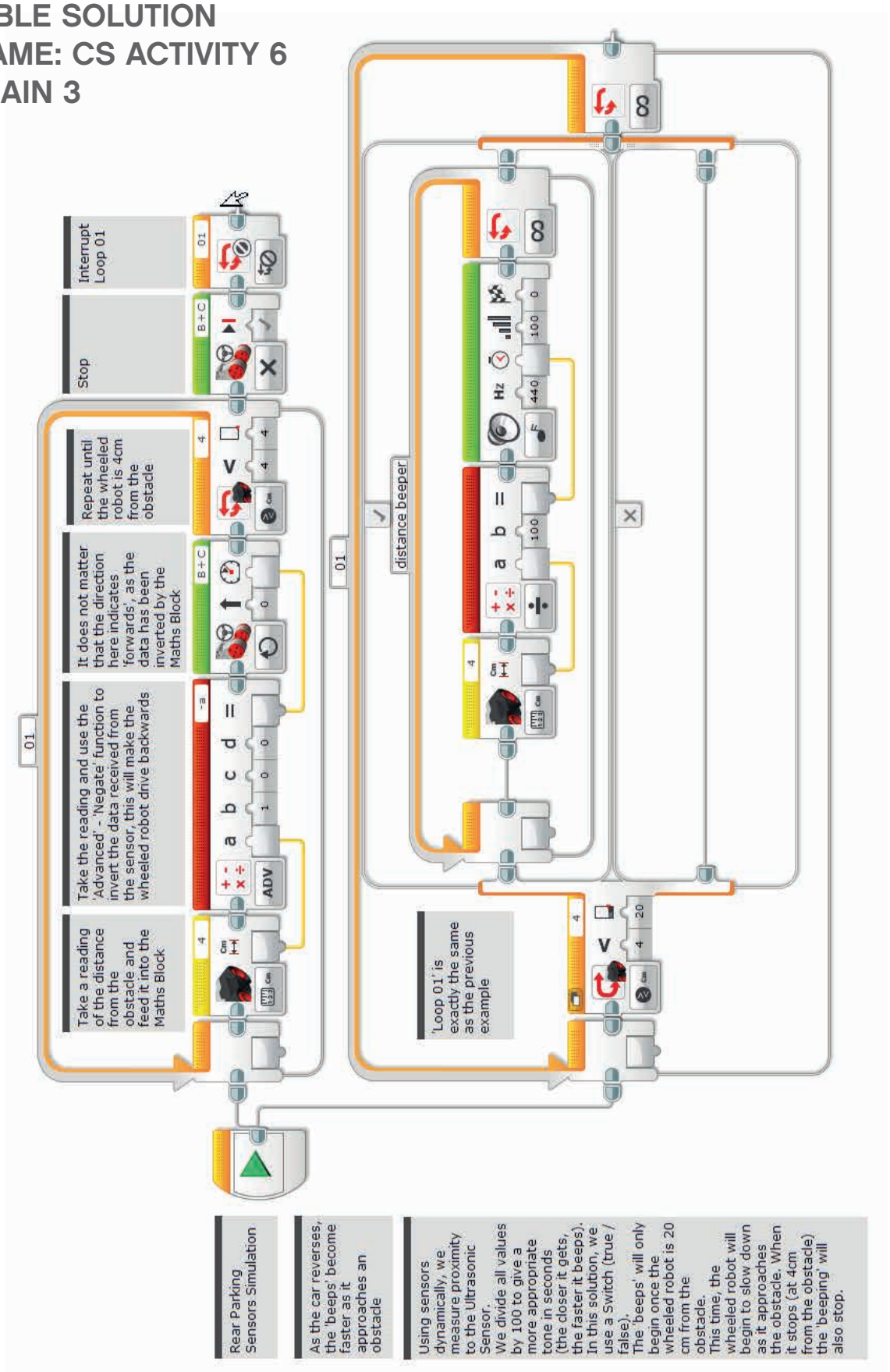
Activity 6
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 6
TAB: MAIN 2



Activity 6
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 6
TAB: MAIN 3



Activity 6
Appendix

Possible ROBOTC Solution
FILENAME: Activity6_1.c

Activity6_1.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Create a program to drive the robot backwards with a warning beep getting quicker
as the Ultrasonic Sensor sees an approaching object. The robot stops when a distance
of less than 4cm is reached.
*/

task distanceBeeper()
{
  while(true)
  {
    //Start playing a tone (in tens of milliseconds).
    playTone(440, getUSDistance(sonarSensor));

    // Delay while the sound plays.
    while(bSoundActive)
    {
      sleep(10);
    }

    //Add 50ms of "silence" to avoid a constant tone.
    sleep(50);
  }
}

task main()
{
  //Start the "distance beeper" task.
  startTask(distanceBeeper);

  // Set the motor speed.
  setMotorSpeed(motorB, -20);
  setMotorSpeed(motorC, -20);

  // Wait while the distance is great than 4.
  while(getUSDistance(sonarSensor) >= 4)
  {
    sleep(10);
  }
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 6
Appendix

Possible ROBOTC Solution
FILENAME: Activity6_2.c

Activity6_2.c

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Create a program to drive the robot backwards with a warning beep getting quicker
as the Ultrasonic Sensor sees an approaching object. The beep only sounds at a distance
less than 20cm. The robot stops when a distance of less than 4cm is reached.
*/

task distanceBeeper()
{
  while(true)
  {
    if (getUSDistance(sonarSensor) < 20){
      //Start playing a tone (length determined by distance).
      playTone(440, (getUSDistance(sonarSensor)));

      // Wait for the tone to be done playing.
      while(bSoundActive)
      {
        sleep(10);
      }

      //Add 50ms of "silence" to avoid a constant tone.
      sleep(50);
    }
  }
}

task main()
{
  //Start the "distance beeper" task.
  startTask(distanceBeeper);

  setMotorSpeed(motorB, -20);
  setMotorSpeed(motorC, -20);

  while(getUSDistance(sonarSensor) >= 4)
  {
    //Keep driving.
    sleep(10);
  }
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 6
Appendix

Possible ROBOTC Solution
FILENAME: Activity6_3.c

Activity6_3.c*

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard   !!*//

/*
Create a program to drive the robot backwards with a warning beep
getting quicker as the Ultrasonic Sensor sees an approaching object.
The beep only sounds at a distance less than 20cm. The robot
slows down as it nears the object. The robot stops when a distance
of less than 4cm is reached as does the beeper.
*/

task distanceBeeper()
{
  while(true)
  {
    if (getUSDistance(sonarSensor) < 20){

      //Start playing a tone (length determined by distance).
      playTone(440, (getUSDistance(sonarSensor)));

      //Delay for the length of the tone.
      //Wait while the sound is playing.
      while (bSoundActive)
      {
        sleep(10);
      }

      //Add 50ms of "silence" to avoid a constant tone.
      sleep(50);
    }
  }
}

task main()
{
  //Start the "distance beeper" task.
  startTask(distanceBeeper);

  while(SensorValue(sonarSensor) >= 4)
  {
    setMotorSpeed(motorB, -getUSDistance(sonarSensor));
    setMotorSpeed(motorC, -getUSDistance(sonarSensor));
    sleep(50);
  }

  //Set motor speed to 0 and stop the distanceBeeper loop.
  setMotorSpeed(motorB, 0);
  setMotorSpeed(motorC, 0);
  stopTask(distanceBeeper);
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 7
Keyless Starting of a Car

LEGOeducation.com.au

LEGO mindstorms education EV3

Activity 7

Keyless Starting of a Car

Did you know that it's now possible to start a car without a key? The 'car key' or fob stays in your pocket or handbag and allows the engine to be started remotely. The driver just needs to press a combination of buttons and the clutch in order to start the car.

In this activity, the students will learn how they can do the same with their wheeled robot by using a combination of sensors and algebraic logic. For example, has something come into range of the Ultrasonic Sensor and has the Touch Sensor been pressed? If the answer is 'true' to both of these questions, the program will trigger a response. The students will use this logic in order to control their wheeled robot.

Activity 7
Lesson Plan

OUTCOMES

Students will be able to:

- understand several key algorithms that reflect computational thinking
- understand simple Boolean logic (such as AND, OR and NOT) and some of its uses in circuits and programming
- use the Logic Operations Block in conjunction with the Switch Block
- use several sensors, in combination, to activate a program on the EV3 Brick

VOCABULARY

input, output, Boolean logic, algebraic, switch, true, false

INTRODUCTION

- Explain to the students that during the course of this lesson, they will adapt their wheeled robot to move just like a real car does when it is started using a button.
- For much of the students' programming, they have concentrated on using one sensor at a time. But what about using two sensors? Why would there be a need? Make sure that it is mentioned that in a burglar alarm system, different sensors should be able to trigger the alarm. But what happens when two sensors both send out messages?
- Modern cars with keyless start require three actions in order for the car to start. What are they? Make sure that the students say that the three things that are needed are: the key, clutch down and button press. How can they do this with the wheeled robot? For this, they will need to investigate logic.
- The challenge today is to create a program that only gives control to the wheeled robot after the right start combination is activated.

INTRODUCTION EXPLORATION TASK

- The students will work in pairs or small groups to construct the Touch Sensor module (Robot Educator > Building Instructions > Touch Sensor – Driving Base)
- The students will also work in pairs or small groups to construct the Ultrasonic module (Robot Educator > Building Instructions > Ultrasonic Sensor – Driving Base)

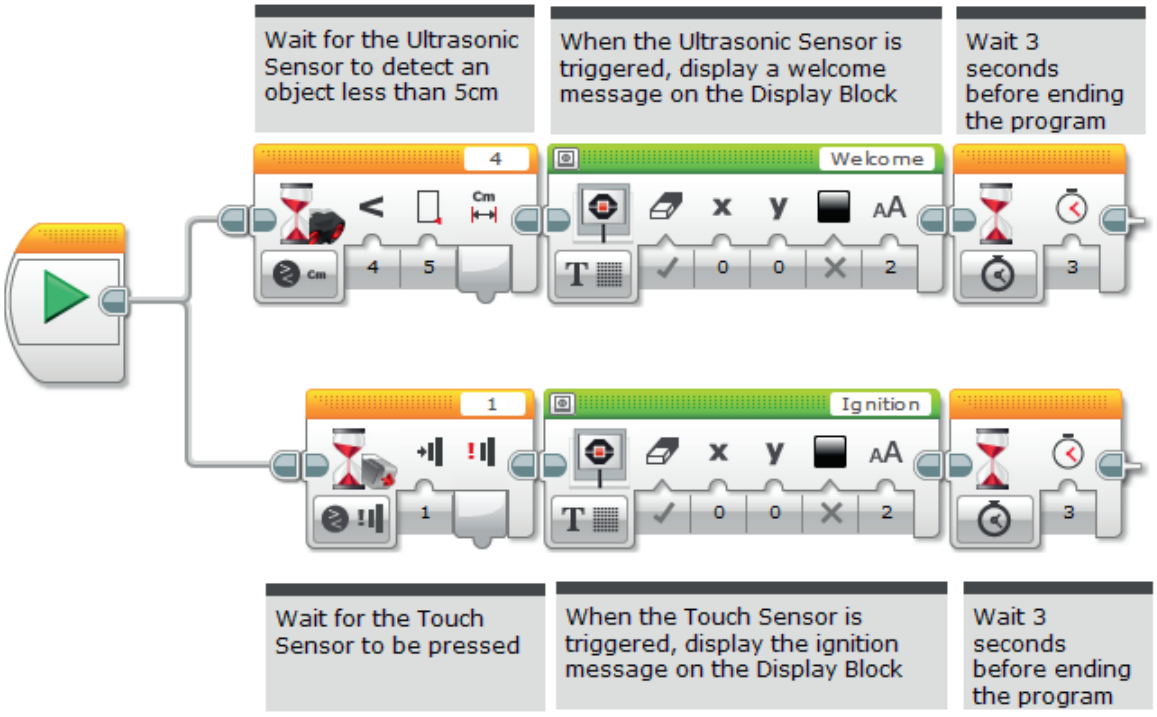
Activity 7
Lesson Plan

MAIN CHALLENGE 1

- The students will need to have some experience of working with more than one sensor. Make sure that you allow enough time in the lesson for this.
- Ensure that the base model has both an Ultrasonic Sensor and Touch Sensor connected to it. The builds for these can be found in the build guide book or in the Robot Educator building instructions. Explain that they are going to create a program that displays messages on the EV3 Brick Display when one of the sensors is triggered.
- Once the students have got their wheeled robot to display messages, ask them to put their program inside a loop in order to repeat those messages.

Activity 7
Lesson Plan

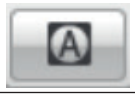
POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 7
TAB: MAIN 1



MAIN CHALLENGE 2

- Get feedback from students on the first program. Ask them how they could integrate the two sensors into one programming line. Explain that they will need to use a programming block called the *Logic Operations Block*.
- Explain to the students that the Logic Operations Block is similar to a Venn diagram in the way that it behaves. Draw a Venn diagram on the board with two circles overlapping or show a Venn diagram from the Internet. Explain how it works. Point out the many conditions that this diagram has and how it relates to the Logic Operations Block in the EV3 Software. Explain the following image and how the results are created.

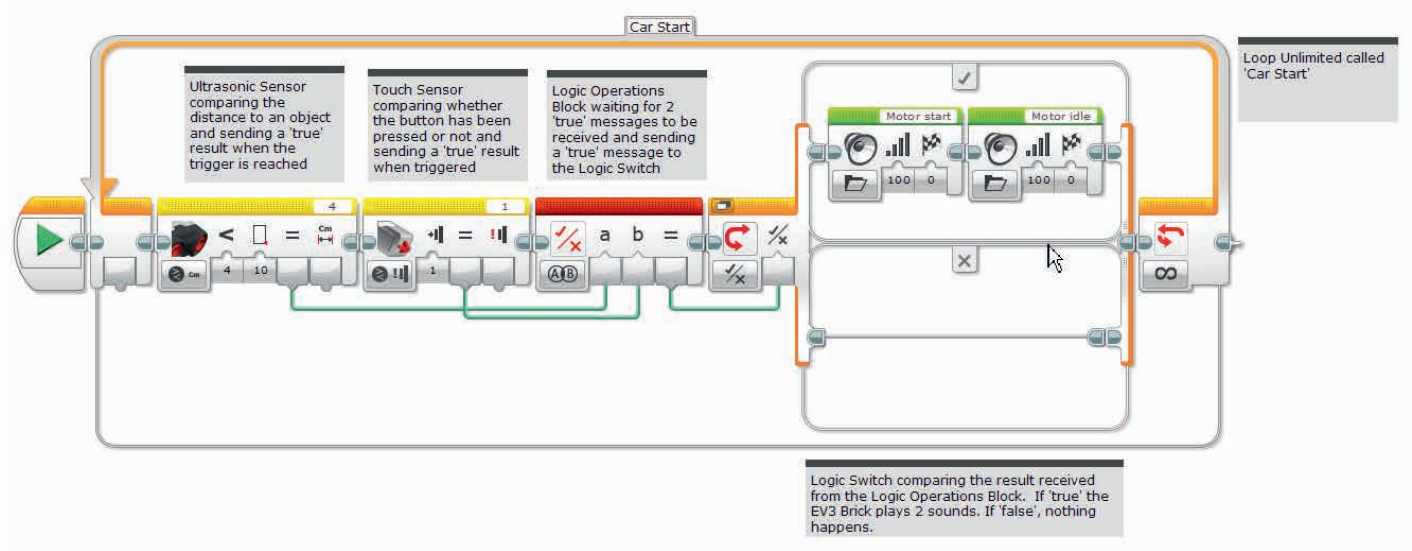
Activity 7
Lesson Plan

Modes		Inputs Used	Result
	AND	A, B	True if both A and B are True, otherwise False
	OR	A, B	True if either A or B (or both) are True, False if both A and B are False
	XOR	A, B	True if exactly one of A and B is True, False if both A and B are True, False if both A and B are False
	NOT	A	True if A is False, False if A is True

MAIN CHALLENGE 2

- The EV3 Software Logic Operations Block creates a true / false output depending on the criteria that have been set in the block. Most people will use it for A and B, giving a true output. That's what will be used today.
- In the example below, the Ultrasonic Sensor is used to simulate the keys in a driver's pocket when the driver is in the car. This is known as 'entering the bubble'. The Touch Sensor is used to turn on the car. Ask the students to create this program.

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 7
TAB: MAIN 2



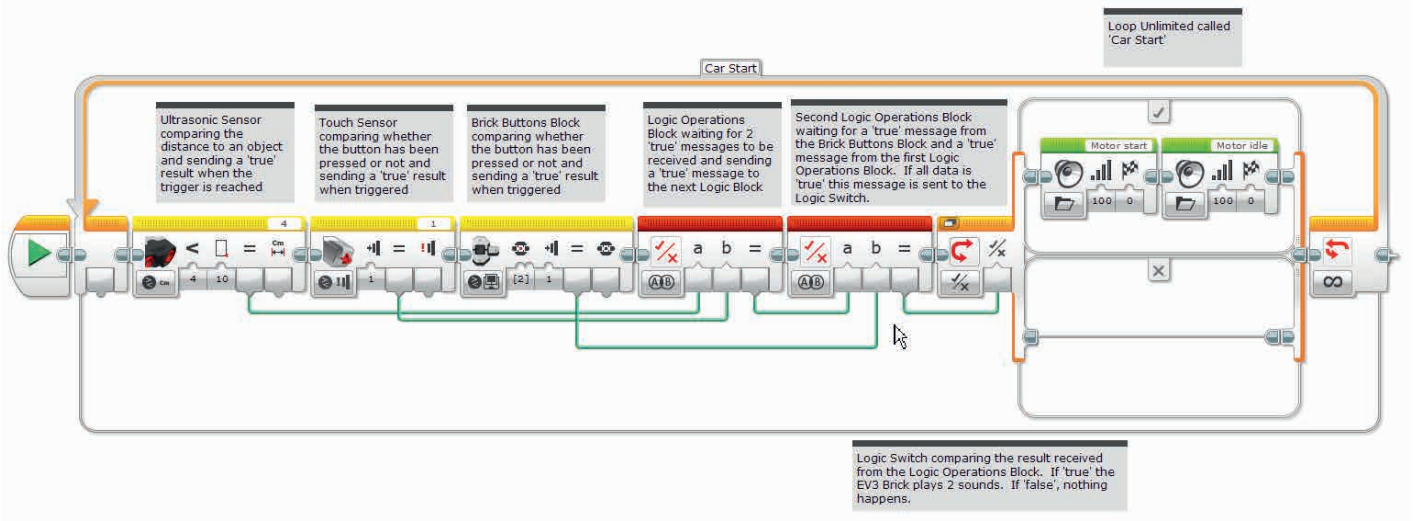
- Ask the students if this is what happens in real life.

Activity 7
Lesson Plan

MAIN CHALLENGE 3

- Make sure that the students understand that three criteria need to be true in order for the car to start.
- First, the key or fob itself needs to be inside of the car.
- Second, the clutch needs to be depressed.
- Third, the button must be pressed in order to turn the ignition on.
- Ask the students to design a program that simulates this. Which blocks will be needed?
- NB: Some cars only require two of the above criteria to be met in order to start. Consider asking certain students / groups to design a program which only has to meet two conditions.

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 7
TAB: MAIN 3



CLASS DISCUSSION

- What did the students think about using the Data Wires? Ask them for their thoughts. Point out that wires can be moved around and tidied up, which is important to ensure that programs don't become too messy.
- Possible extension activity: Could the students design a program that displays an error message if the criteria for turning on the car aren't met?
- Revise the use of a Wait Block in programming and point out that these are generally used for one sensor at a time. The Logic Operations Block allows users to use multiple sensors simultaneously.
- Ask the students to compare the text versions of the challenges to the EV3 Software versions. Ask them to follow the course of each solution so that they understand how the two versions work. On their student worksheet ask them to record their thoughts about using the two different languages.

Activity 7
Student Worksheets

CHALLENGES FOR TODAY

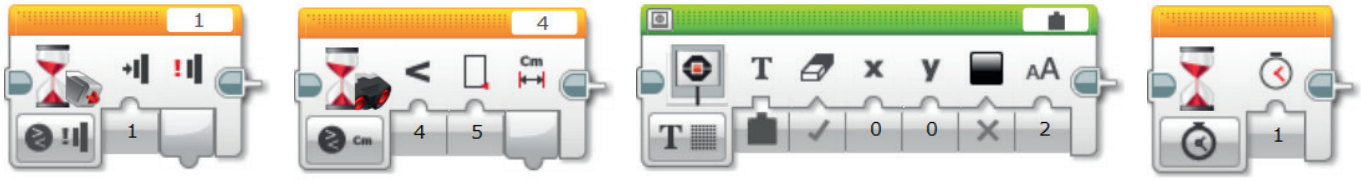
Today you are going to create a keyless entry system for your wheeled robot. When a combination of sensors is activated, your drive program will be executed. In order for you to succeed with these challenges, you will need to use a number of different sensors, and challenges 2 & 3 will require you to use one or more Logic Operations Blocks. On this sheet there are no answers – we give possible programming blocks for you to explore while you are solving the problem. There is room for you to write pseudo-code and to record your observations.

CHALLENGE 1

Program your wheeled robot to display the text 'Welcome' when the Ultrasonic Sensor sees an object and to display the text 'Ignition' when the Touch Sensor is pressed.

Tip: Ensure that the Ultrasonic Sensor's parameter is set to **less than (<).**

Blocks to Consider



Plan your program first. Write it in pseudo-code below:

Activity 7
Student Worksheets

CHALLENGE 2

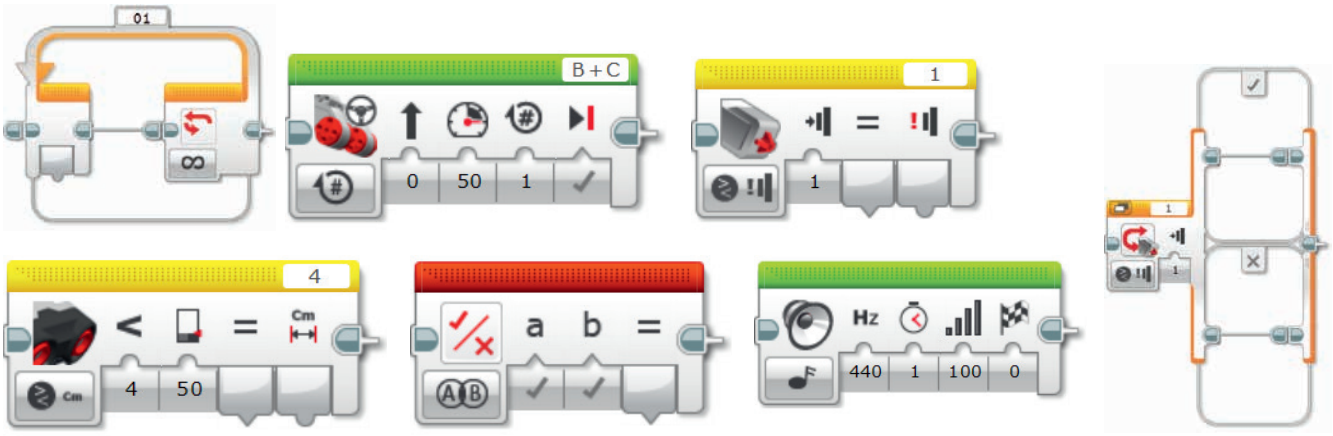
Challenge 2 is all about using the Logic Operations Block and making sure that two sensors work together in order to provide information to another block.

So how does the keyless car work? For this task, the Touch Sensor is the 'ignition' and the Ultrasonic Sensor is used to 'see' the key in the car. Both sensors will need to be activated correctly in order for your wheeled robot to start!

Now that you understand how to use more than one sensor in your programs, you will need to use the sensor blocks (yellow ones) in order to create logic for the Logic Operations Block. Each sensor block will be used to create a true output. This output is taken from the sensor block to the Logic Operations Block. When the right conditions are met, an output from the Logic Operations Block can be taken to the Switch Block. In this program, two sensor blocks will feed the one Logic Operations Block.

Blocks to Consider

Use the same blocks that you used in programming task 1, but also consider using the following:



Plan your program first. Write it in pseudo-code below:

Activity 7
Student Worksheets

CHALLENGE 3

You will now need to program your wheeled robot so that it reacts when the conditions of three different sensors are met.

The sensors will be:

- Touch Sensor = Ignition
- Ultrasonic Sensor = Detect key in car
- Brick Buttons = Clutch

The Logic Operations Block can receive two inputs. But what happens if you want three inputs? Think about using two Logic Operations Blocks in order to achieve this.

Two sensors will need to enter the first Logic Operations Block. The output is then taken to the next Logic Operations Block with the third input (sensor).

That result is then taken to the Switch Block.

Blocks to Consider

Use the same blocks that you used in programming tasks 1 and 2, but also consider using the following:

Plan your program first. Write it in pseudo-code below:

Activity 7
Student Worksheets

After a programming activity, it is important to note down your thoughts and observations. Consider the following points and then in the box below record how the activity went.

- How could you improve your program?
- Could your program have been more streamlined? Have you used too many blocks? Is there a more efficient way of building your program?
- What examples of real-world applications could you see your program being used in?
- Comparing text-based programming with visual programming, which is easier to follow? Try writing in the other program to see which is more efficient.

Thoughts and Observations

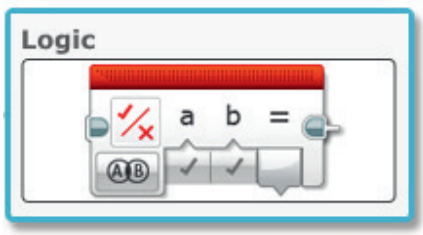
Activity 7
Teacher Notes

ROBOT EDUCATOR TUTORIALS

The following Robot Educator Tutorials will help teachers and their students to solve the challenges.

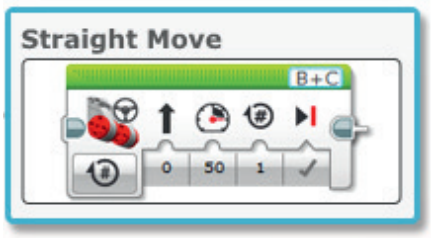
NEW ROBOT EDUCATOR TUTORIALS

Beyond Basics > Logic

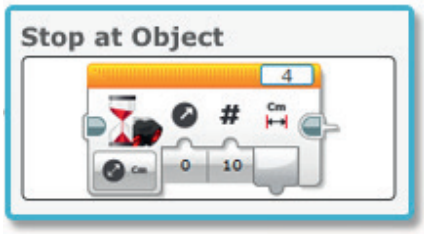


ROBOT EDUCATOR TUTORIALS PREVIOUSLY COVERED

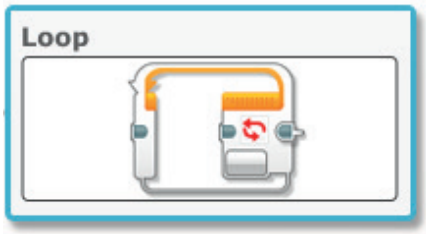
Basics > Straight Move



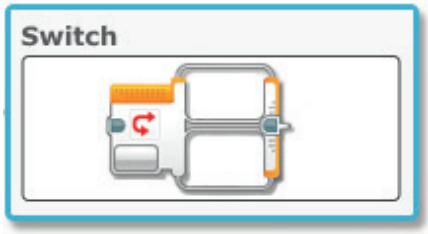
Basics > Stop at Object



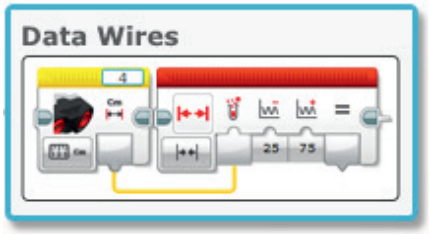
Beyond Basics > Loop



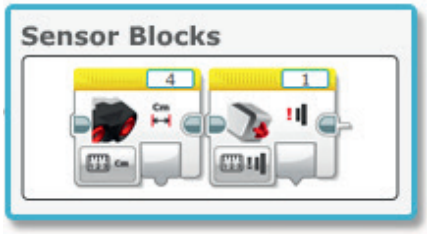
Beyond Basics > Switch



Beyond Basics > Data Wires



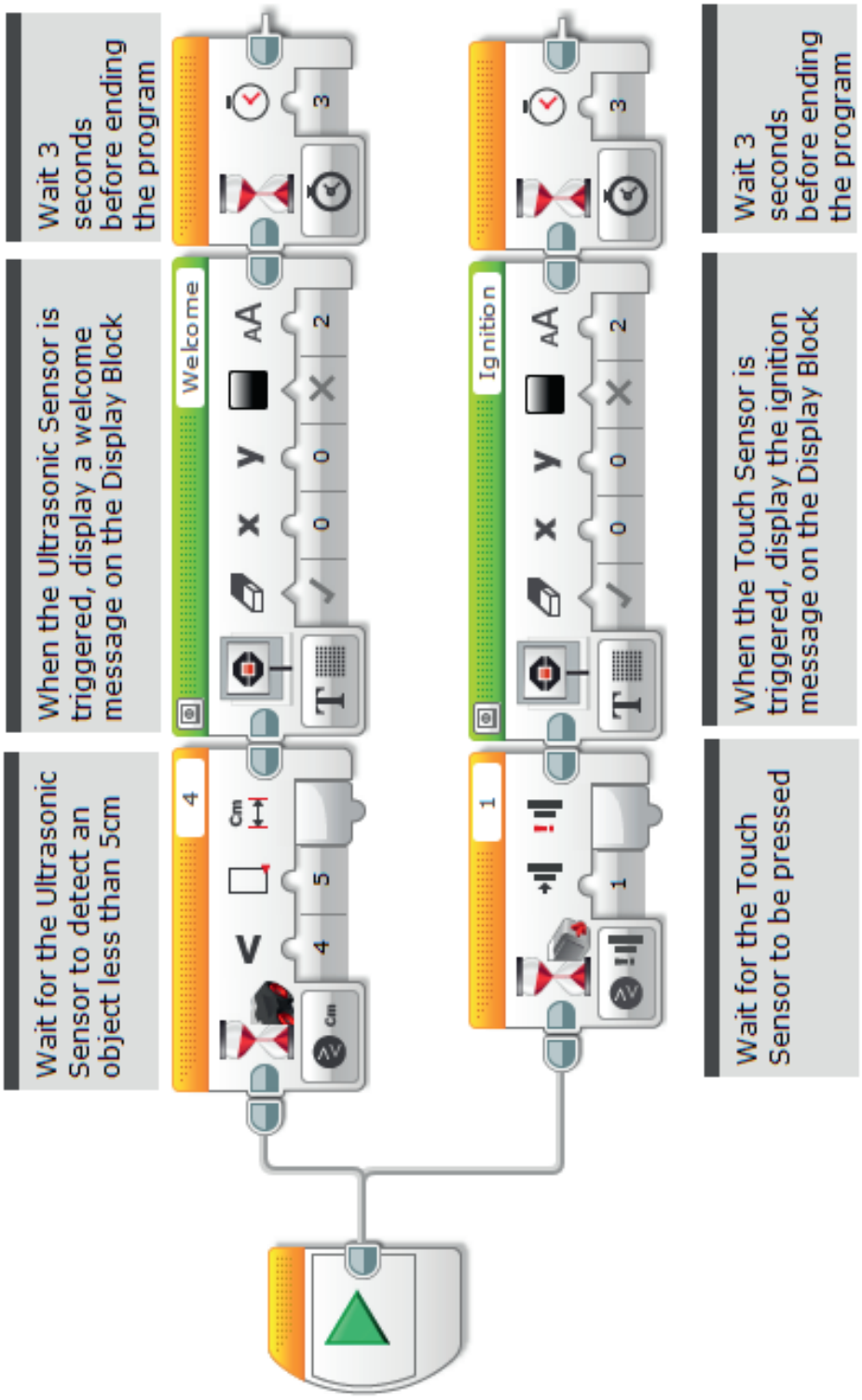
Beyond Basics > Sensor Blocks



APPENDIX FOR ACTIVITY 7: LARGE IMAGES AND PROGRAMS

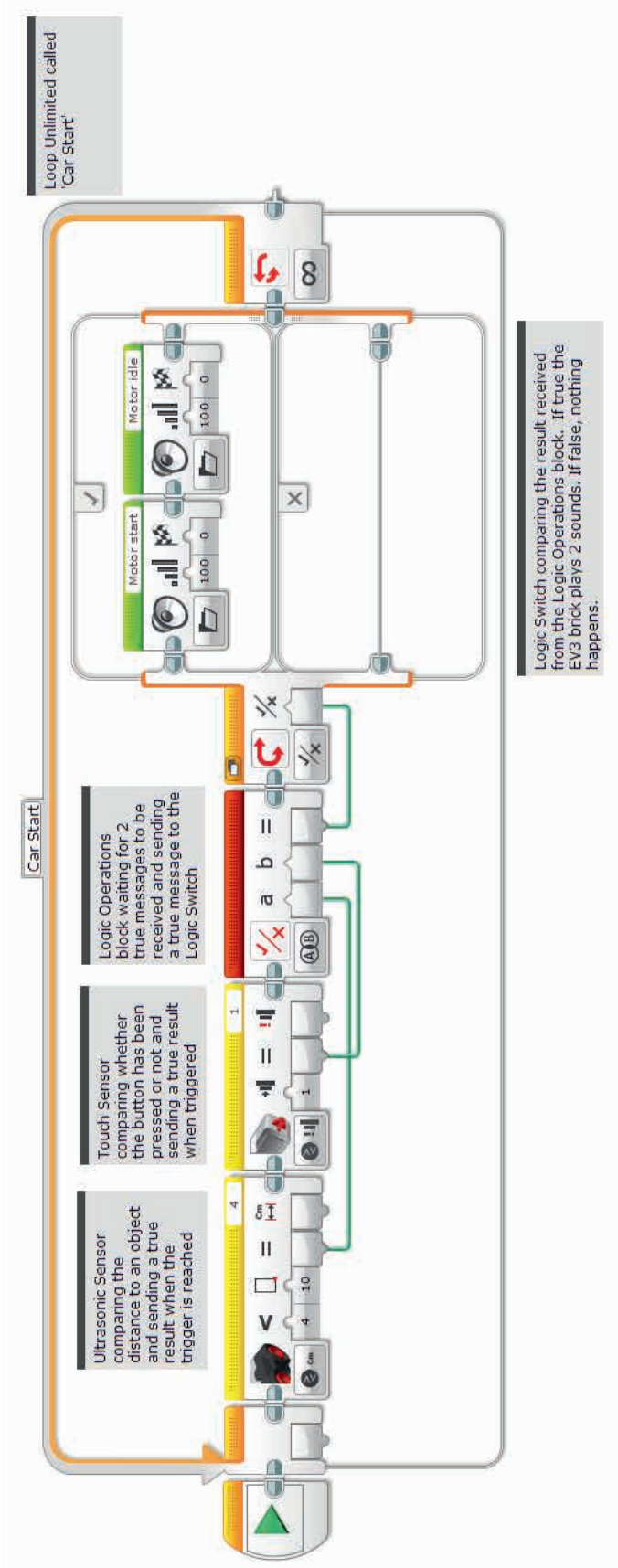
Activity 7
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 7
TAB: MAIN 1



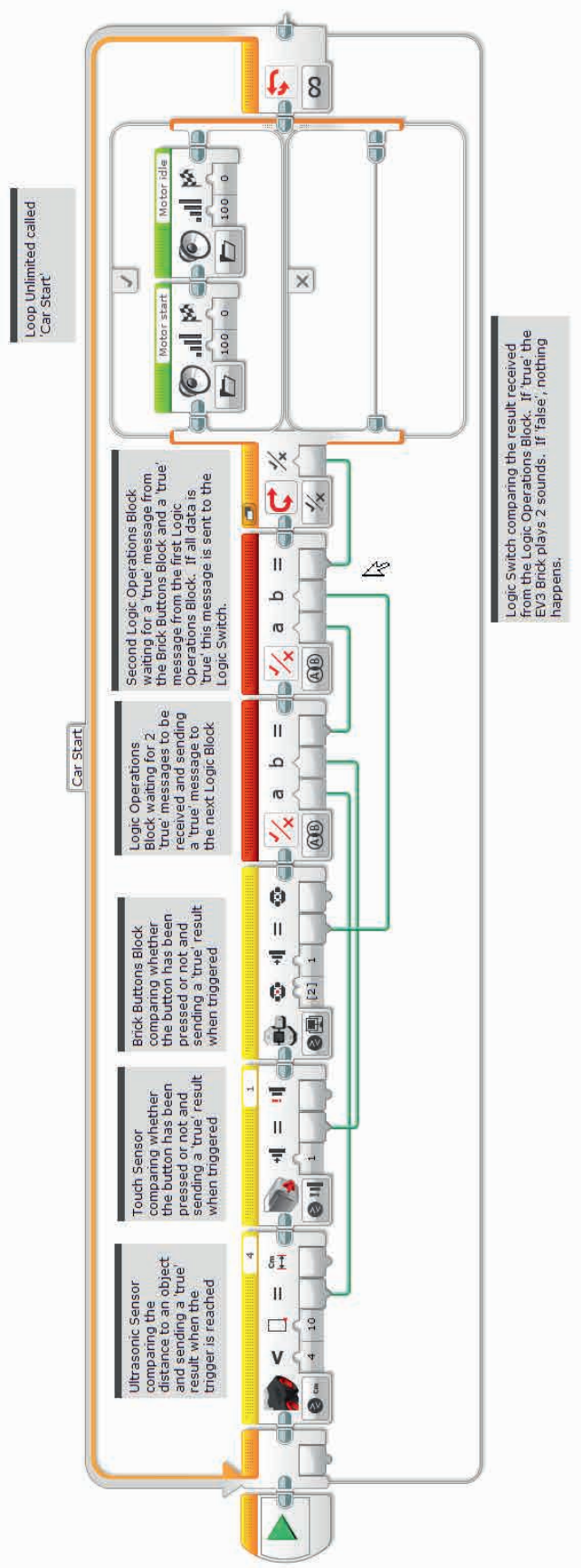
Activity 7
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 7
TAB: MAIN 2



Activity 7
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 7
TAB: MAIN 3



Activity 7
Appendix

Possible ROBOTC Solution
FILENAME: Activity7_1.c

Activity7_1.c*

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard  !!*//

/*
Create a program that shows how a keyless ignition works.
The Ultrasonic Sensor welcomes the driver. The Touch Sensor works
the ignition.
*/

task touchTask()
{
  //Wait for the touch sensor to be pressed.
  while(getTouchValue(touchSensor) == 0)
  {
    //Do Nothing.
    sleep(10);
  }

  //Display Text on the LCD Screen.
  eraseDisplay();
  displayCenteredBigTextLine(4, "Ignition");

  //Display text for 3 seconds.
  sleep(3000);
}

task main()
{
  //Start the second task to monitor the touch sensor.
  startTask(touchTask);

  //Wait for the sonar sensor to see an object that is less than
  //or equal to 5cm away.
  while(getUSDistance(sonarSensor) >= 5)
  {
    //Do Nothing.
    sleep(10);
  }

  //Display text on the LCD Screen.
  eraseDisplay();
  displayCenteredBigTextLine(4, "Welcome");

  //Display text for 3 seconds.
  sleep(3000);
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 7
Appendix

Possible ROBOTC Solution
FILENAME: Activity7_2.c

Activity7_2.c*

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard   !!*//

/*
Create a program that starts the engine running when two conditions
are met. The Sonar Sensor detects the driver in the seat and
the Touch Sensor is the press of the ignition.
*/

task main()
{
  //Repeat continuously.
  while(true)
  {
    //If Sonar Sensor is less than 10 AND Touch Sensor is pressed.
    if(getUSDistance(sonarSensor) < 10 && getTouchValue(touchSensor) == 1)
    {
      playSoundFile("Motor start");

      // Wait for the tone to be done playing.
      while(bSoundActive) sleep(10);


      //After the engine starts - keep playing the sound.

      playSoundFile("Motor idle");

      // Wait for the tone to be done playing.
      while(bSoundActive) sleep(10);
    }
  }
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 7
Appendix

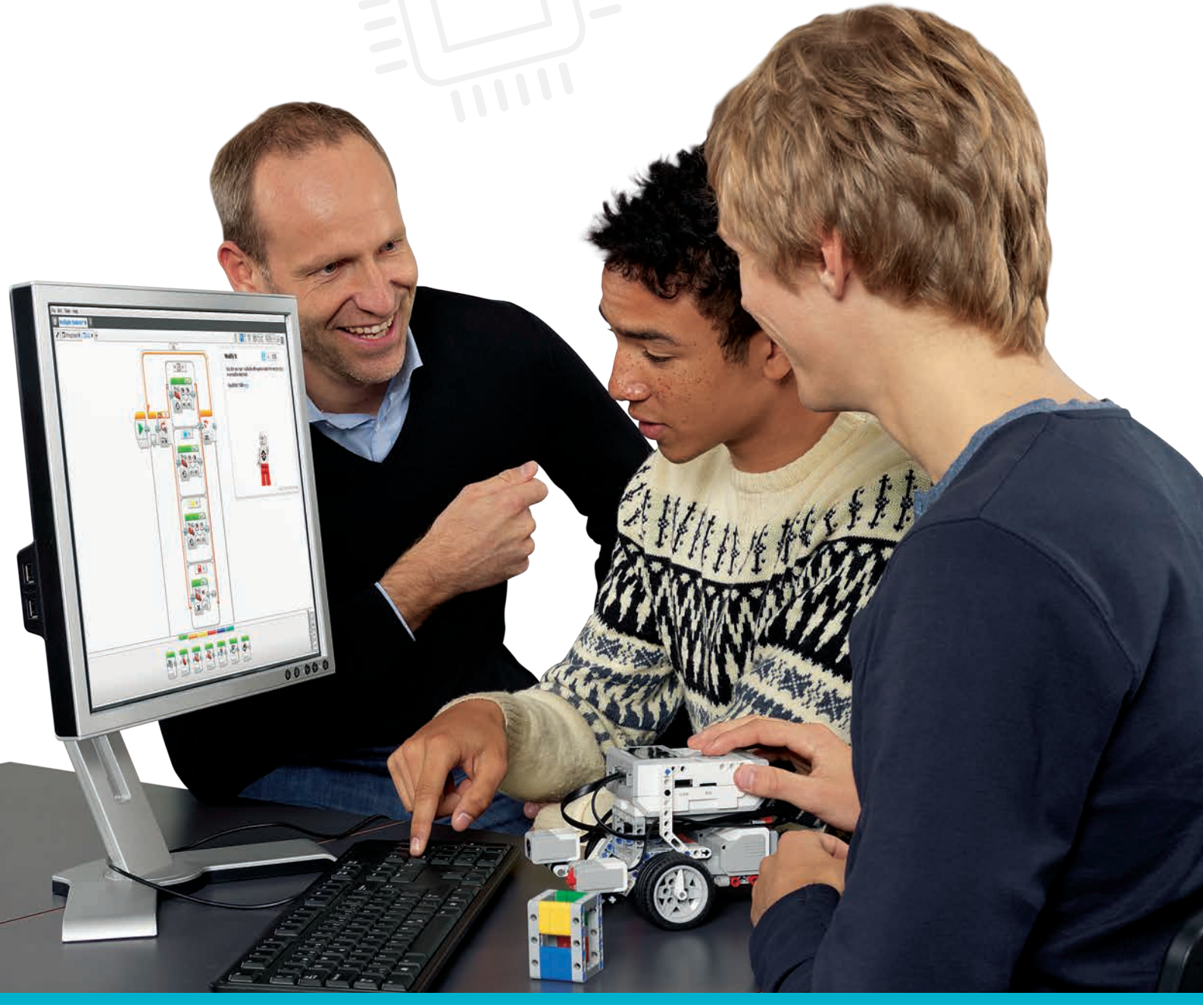
Possible ROBOTC Solution
FILENAME: Activity7_3.c

Activity7_3.c*

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard    !!*//

/*
Create a program that starts the engine running when three conditions
are met. The Sonar Sensor detects the key in the car, the Button
press represents the clutch being pressed and the Touch Sensor
is the press of the ignition.
*/

task main()
{
  //Repeat continuously.
  while(true)
  {
    //If Sonar Sensor is less than 10 AND Touch Sensor is pressed
    //AND Enter Button is pressed.
    if(getUSDistance(sonarSensor) < 10 &&
       getTouchValue(touchSensor) == 1  &&
       getButtonPress(buttonEnter) == 1)
    {
      playSoundFile("Motor start");
      // Wait for the tone to be done playing.
      while(bSoundActive) sleep(10);

      //After the engine starts - keep playing the sound.
      playSoundFile("Motor idle");

      // Wait for the tone to be done playing.
      while(bSoundActive) sleep(10);
    }
  }
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 8
Cruise Control



LEGOeducation.com.au

LEGO mindstorms education EV3

Activity 8

Cruise Control

During this activity, you will introduce the students to variables. By the end of the activity, the students will have created a cruise controller for their wheeled robot.

Pressing the Touch Sensor will increase the speed of the wheeled robot.

Point out that in real cars, although cruise control is automated, it is often the case that speed can be easily increased or decreased by using steering-wheel controls. This concept can be simulated using the Variable Block. The speed is set by pressing the Touch Sensor.

Activity 8
Lesson Plan

OUTCOMES

Students will be able to:

- understand several key algorithms that reflect computational thinking
- use the Variable Block to store information
- develop multi-level programs
- create *My Blocks*

VOCABULARY

input, output, variable, constant, loop, wait, motor, Touch Sensor

INTRODUCTION

- Play a video of a car travelling while using cruise control. Ask the students what is happening in the video. Make sure that the students realise that the computer on board is controlling the speed of the car automatically, a little like an autopilot on an aeroplane.
- Use the images below to reinforce this.

Image 1

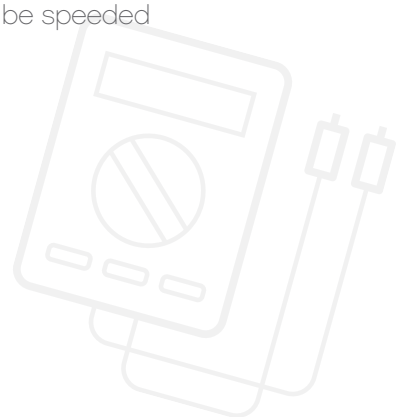
Image 2

Activity 8
Lesson Plan

- Look at image 1. Ask the students what they think the plus and minus buttons mean. Make sure that they realise that this can adjust the speed of the car.
- Tell the students that today's task will utilise two Touch Sensors in order to control and maintain the speed of their wheeled robot.
- Look at image 2. What do the 'Accel' and 'Decel' buttons do? Make sure that the students realise that these buttons can speed up and slow down the car.
- Explain that they will be using the Variable Block today.
- You might wish to explain to the students the difference between a constant and a variable.
- A constant is used to provide the same values over and over again within a program. These fixed values can only be edited by the user when the program is not running.
- You may wish to show the Constant Block in action along with the Move Block and the Display Block.
- A variable is another way of storing values within a program. The difference between a variable and a constant is that the value of a variable can be overwritten time and time again as the program is running.
- Show the students how to use the Variable Block and then get them to create a program that contains a variable.

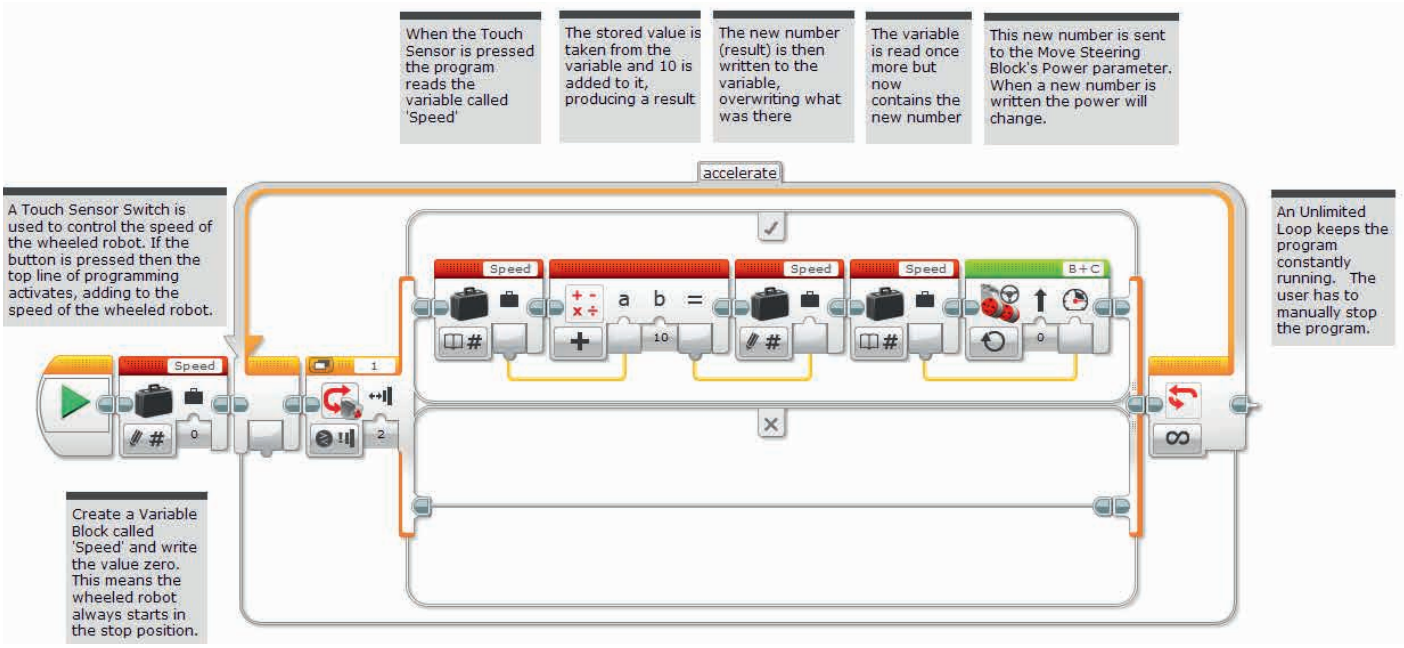
MAIN CHALLENGE 1

- By now, the students should be familiar with the Robot Educator Driving Base model. If they have not already built it, they will need to build it at this stage. Remember that the students will need to attach two forward-facing Touch Sensors to the model.
- Expand on the students' understanding of the Variable Block. Explain that it is a programming block that can store data (text, logic, a numeric value or array), which can be overwritten at any time while the program is running.
- Explain that this block has to be read and then written to, using either the Maths, Text or Array Operations Blocks.
- The students will need to program their wheeled robot so that once it is moving, it can be speeded up by pressing the Touch Sensor.



Activity 8
Lesson Plan

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 8
TAB: MAIN 1



MAIN CHALLENGE 2

- With the first program written and the wheeled robot accelerating, ask the students to theorise how they could extend the program in order to slow it down.
- After hearing their thoughts, make sure that the students realise that one solution could be to have a second unlimited loop that simply changes the Touch Sensor port (another sensor added), with the Maths Block changing to 'subtract' rather than 'add' mode.
- Remind the students about multitasking and tell them that they will need to utilise their knowledge of this type of programming when completing this activity. Inform them that a Loop Block must be dragged into the programming area before taking the second Data Wire from the Start Block.

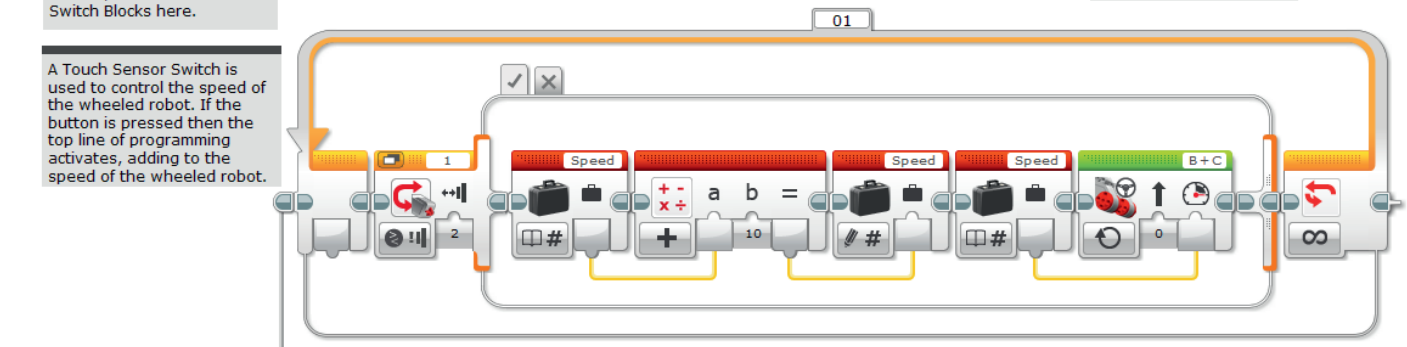
Activity 8
Lesson Plan

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 8
TAB: MAIN 2

NOTE:

The Switches have been set to Tabbed View in order to save space. Only the 'true' tab is used for the programming. The 'false' tab is always blank for both Switch Blocks here.

A Touch Sensor Switch is used to control the speed of the wheeled robot. If the button is pressed then the top line of programming activates, adding to the speed of the wheeled robot.



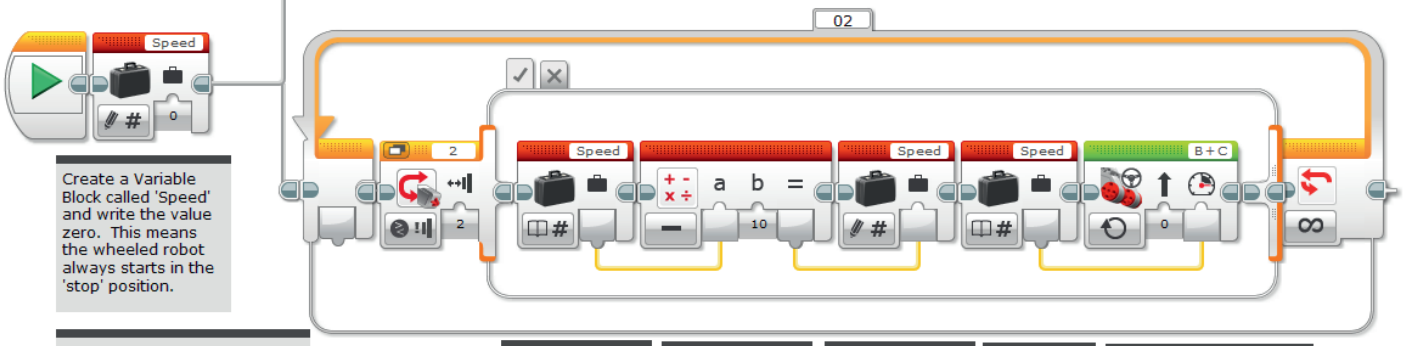
When the Touch Sensor is pressed the program reads the variable called 'Speed'

The stored value is taken from the variable and 10 is added to it, producing a result

The new number (result) is then written to the variable, overwriting what was there

The variable is read once more but now contains the new number

This new number is sent to the Move Steering Block's Power parameter. When a new number is written the power will change.



Create a Variable Block called 'Speed' and write the value zero. This means the wheeled robot always starts in the 'stop' position.

A second Loop is added with a second Touch Sensor to reduce the speed of the wheeled robot. In essence, the only difference is that the Maths Block subtracts rather than adds.

When the Touch Sensor is pressed the program reads the variable called 'Speed'

The stored value is taken from the variable and 10 is subracted from it producing a result

The new number (result) is then written to the variable overwriting what was there

The variable is read once more but now contains the new number

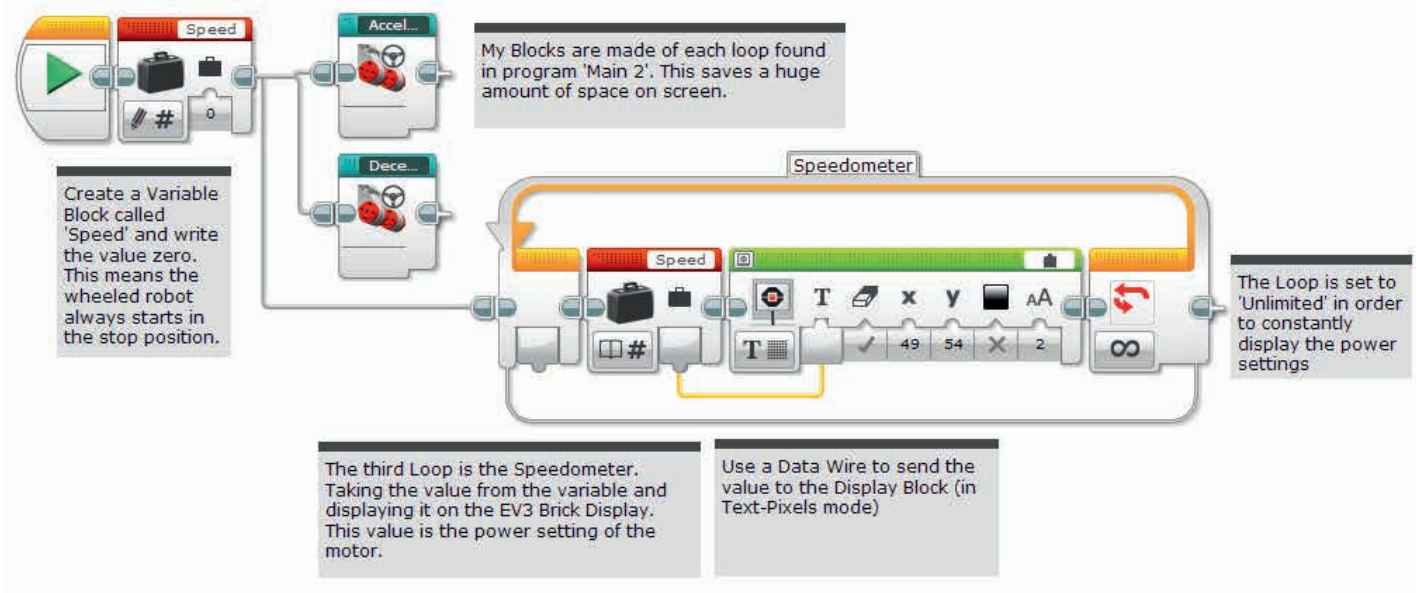
This new number is sent to the Move Steering Block into the power setting. When a new number is written the power will change.

Activity 8
Lesson Plan

MAIN CHALLENGE 3

- With the wheeled robot now accelerating and decelerating at the touch of a button (or two), the students can now extend their programming so that the EV3 Brick Display shows how fast the wheeled robots are going, thus resembling a real car more closely.
- Ask the students to think about their use of the Variable Block. How can they use this block in order to display this information?
- One possible solution is provided below.
- A new skill to be learnt in this activity is the creation of a *My Block*. Two of these can be seen in the solution below. My Blocks allow users to create subroutines of programs they have already written. In the case below, we have taken the acceleration and deceleration loops and created My Blocks from these programs. There are two reasons to do this: to save space and to allow these subroutines to be reused in other programs.

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 8
TAB: MAIN 3



Activity 8
Lesson Plan

CLASS DISCUSSION

- Ask the students to describe some ways in which the Constant Block could be used to control speed. Thinking of how this block works as compared to the Variable Block, make sure that they know that fixed numbers could be useful if they wanted their wheeled robot to stick to a 'speed limit'.
- Activating the first Touch Sensor could use a variable set to 30%, which is comparable to 30kph on a road. A second Touch Sensor could set the speed to 70% by using another Constant Block.
- Ask the students to compare the text versions of the challenges to the EV3 Software versions. Ask them to follow the course of each solution so that they understand how the two versions work. On their student worksheet, ask them to record their thoughts about using the two different languages.

Activity 8
Student Worksheets

CHALLENGES FOR TODAY

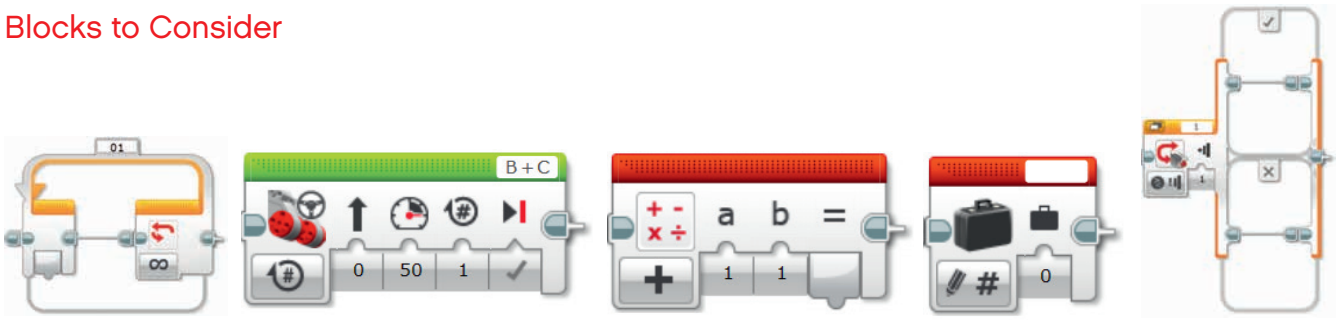
Today you are going to create a 'cruise control' for your wheeled robot that is similar to the cruise control that is found in many cars today. You will need to use the two Touch Sensors in the EV3 set in order to simulate the buttons that are found on the steering wheel of a car that is equipped with cruise control.

CHALLENGE 1

Program the car to speed up in increments of 10. Use the Variable Block as the set speed that can be added to.

Tip: Ensure that the Move Block mode is set to 'On'.

Blocks to Consider



Plan your program first. Write it in pseudo-code below:

Activity 8
Student Worksheets

CHALLENGE 2

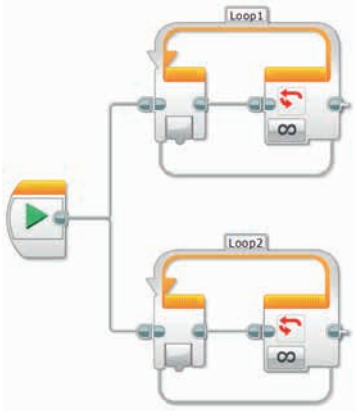
Now that you have created a program that can accelerate your wheeled robot, a new subsection needs to be written in order to decelerate the car. This can be done by simply adding a second Loop and Switch Block.

Inside the extra loop there will be a second Touch Sensor Block and a Maths Block which has been set to 'subtract' rather than 'add' mode.

Remember that you will be using multitasking with two lines of programming running simultaneously.

Blocks to Consider

Use the same blocks that you used in programming task 1, but also consider using the following:



Plan your program first. Write it in pseudo-code below:

Activity 8
Student Worksheets

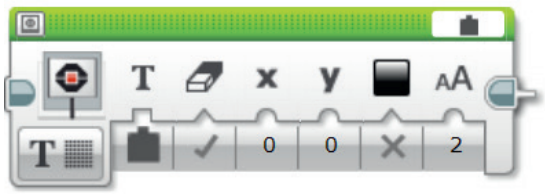
CHALLENGE 3

Now that the speed of your wheeled robot can be controlled by using two Touch Sensors, it would be great to be able to read its speed (motor power) and show this speed on the EV3 Brick Display. Your teacher will have shown you how to create My Blocks using the programs that you have already written. These are useful in two ways. The first is to save room on the programming screen and the second is that these subroutines can be used again within other programs that you write, as they are saved in their own Programming Palette category.

To create a visual power reading, take the value of the variable that controls the motor power and display it on the EV3 Brick using a Display Block that is set to Text – Pixels mode.

Blocks to Consider

Use the same blocks that you used in programming tasks 1 and 2, but also consider using the following:



Plan your program first. Write it in pseudo-code below:

Activity 8
Student Worksheets

After a programming activity, it is important to note down your thoughts and observations. Consider the following points and then in the box below record how the activity went.

- How could you improve your program?
- Could your program have been more streamlined? Have you used too many blocks? Is there a more efficient way of building your program?
- What examples of real world application could you see your program being used in?

Thoughts and Observations

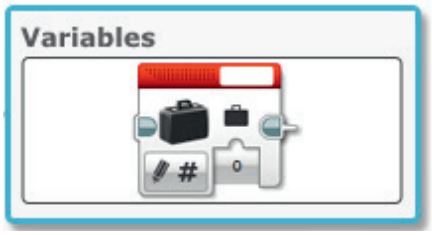
Activity 8
Teacher Notes

ROBOT EDUCATOR TUTORIALS

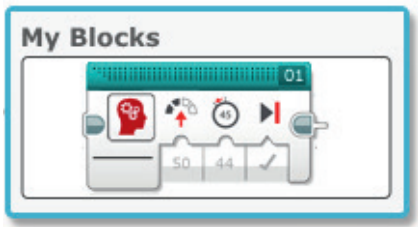
The following Robot Educator Tutorials will help teachers and their students to solve the challenges.

NEW ROBOT EDUCATOR TUTORIALS

Beyond Basics > Variables

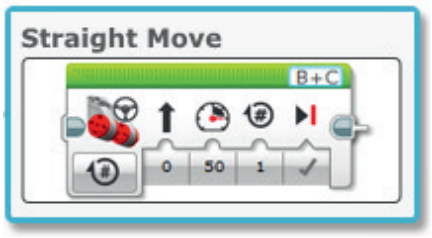


Tools > My Blocks

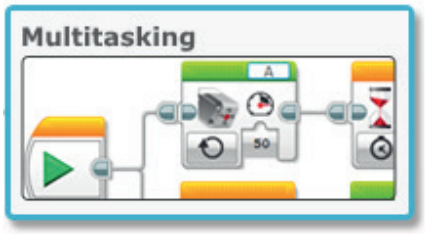


ROBOT EDUCATOR TUTORIALS PREVIOUSLY COVERED

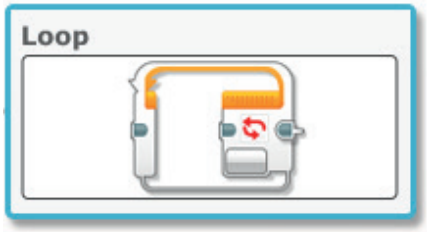
Basics > Straight Move



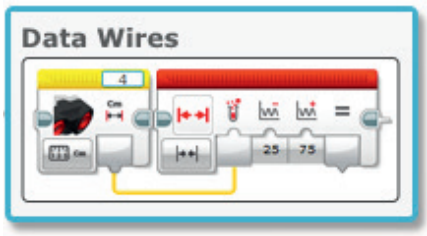
Beyond Basics > Multitasking



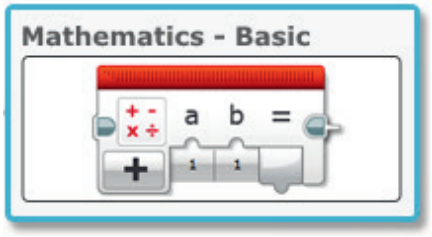
Beyond Basics > Loop



Beyond Basics > Data Wires



Beyond Basics > Mathematics – Basic

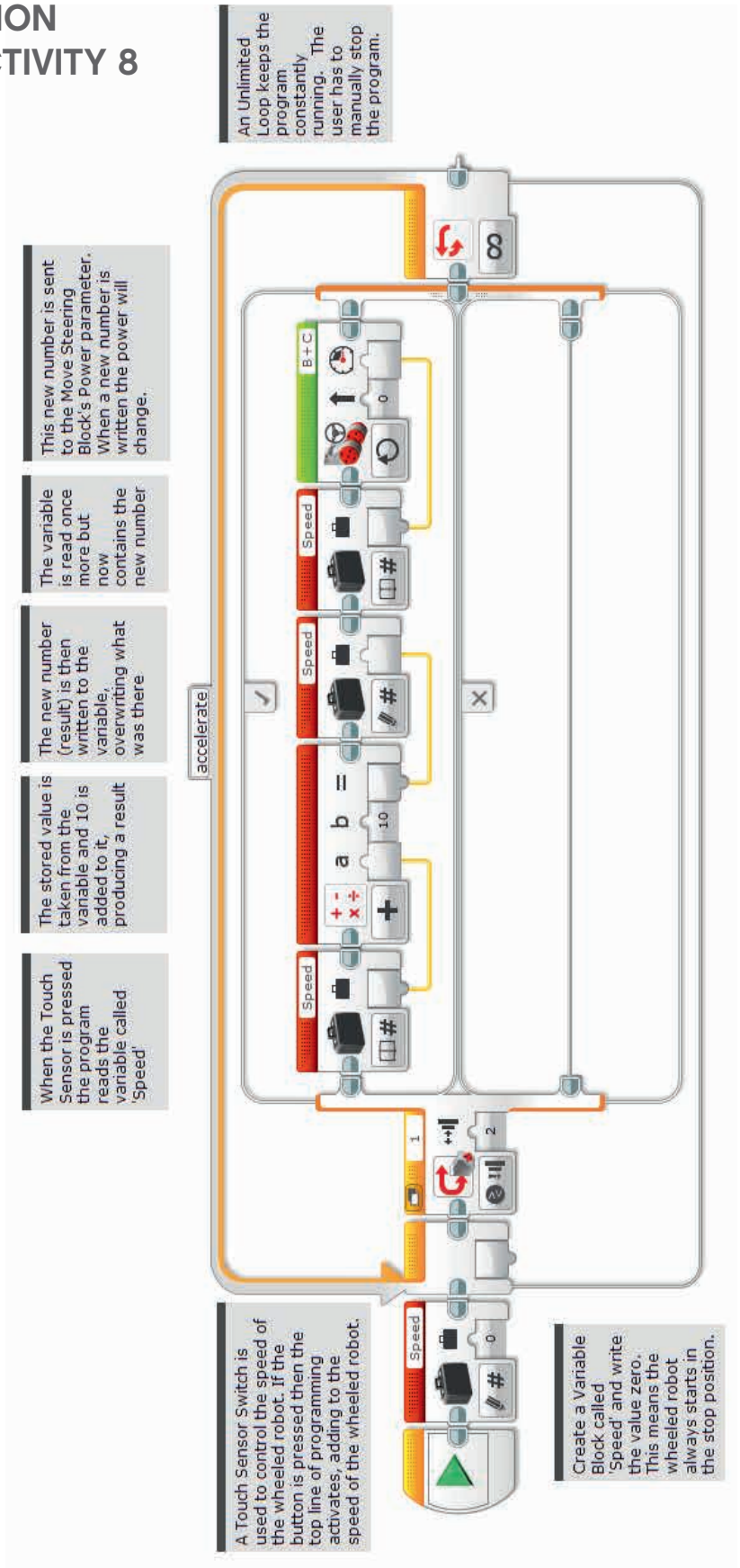


APPENDIX FOR ACTIVITY 8: LARGE IMAGES AND PROGRAMS

Activity 8
Appendix

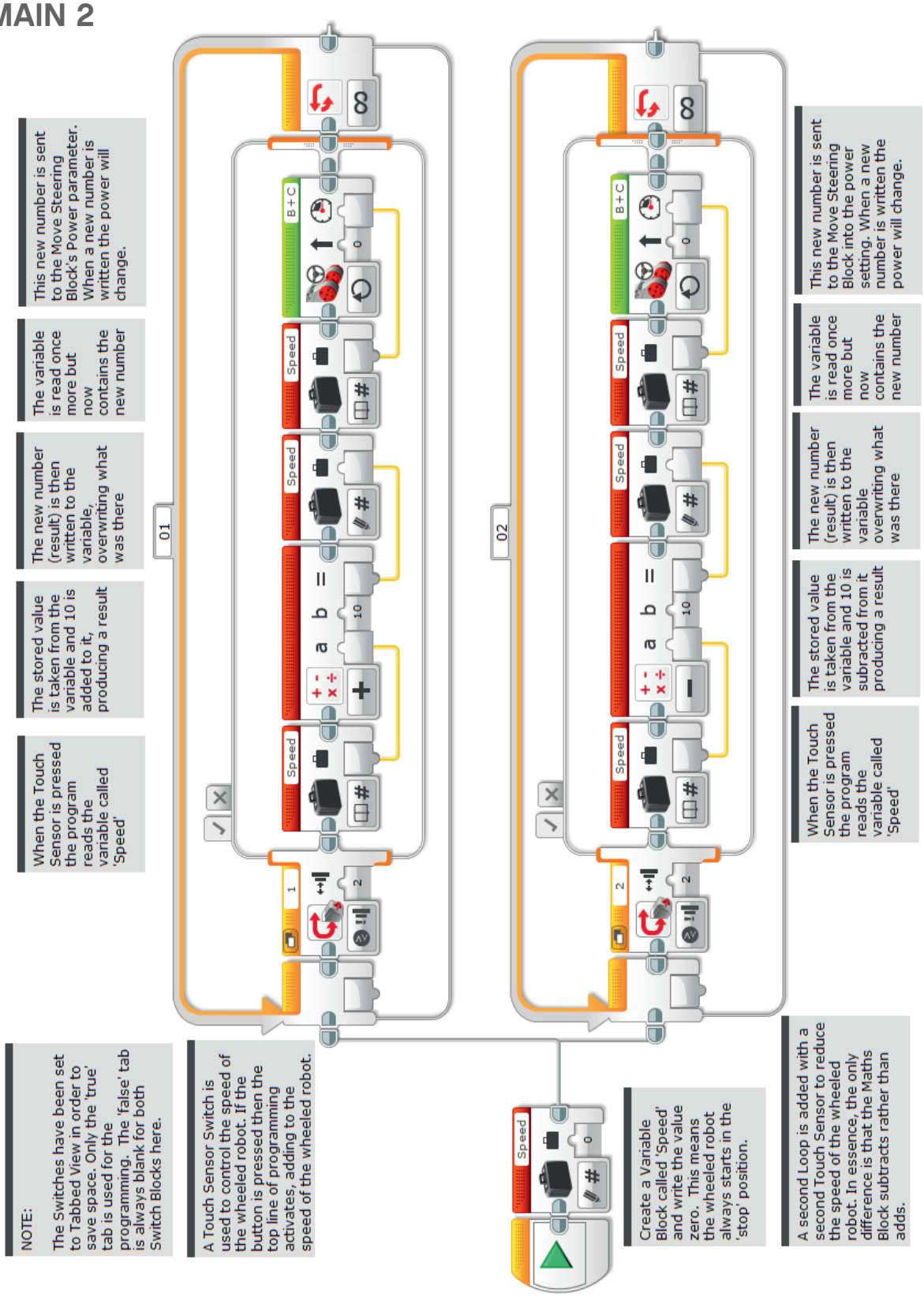
Activity 8
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 8
TAB: MAIN 1



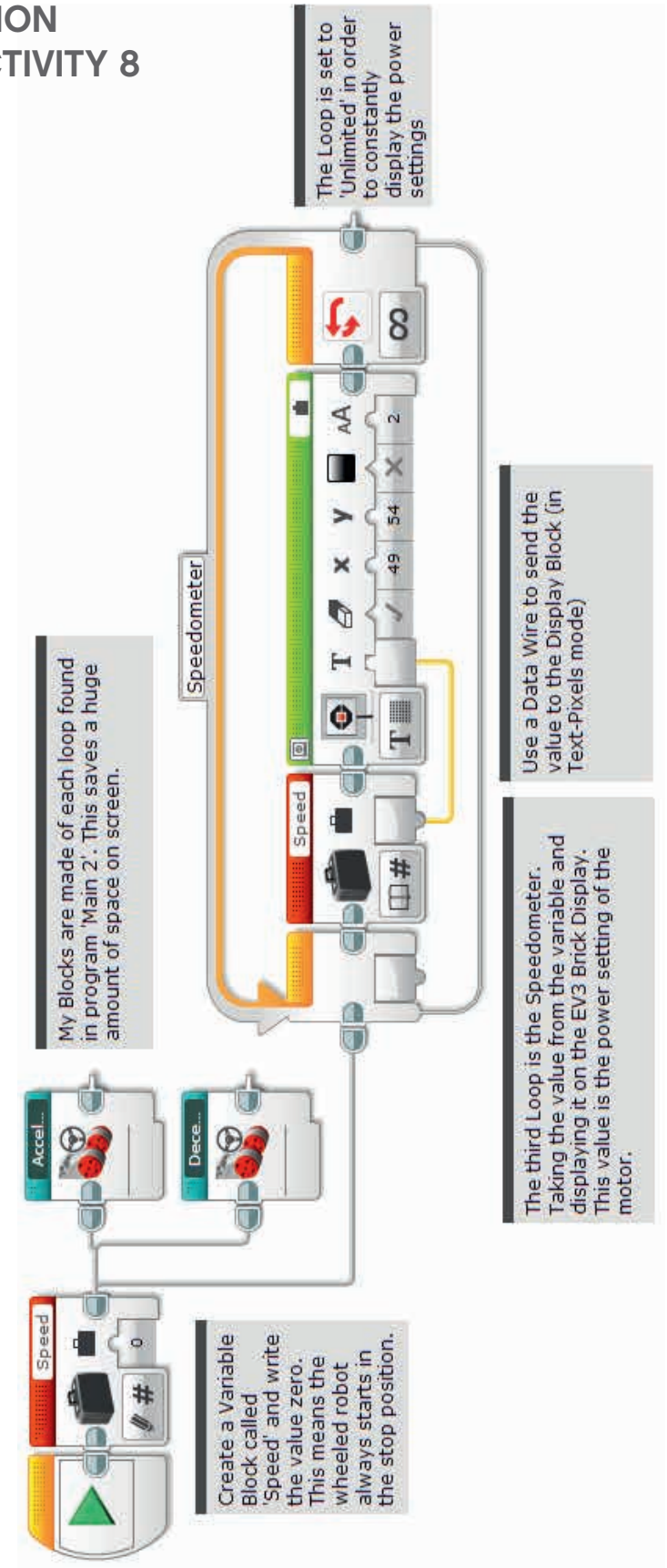
Activity 8
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 8
TAB: MAIN 2



Activity 8
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 8
TAB: MAIN 3



Activity 8
Appendix

Possible ROBOTC Solution
FILENAME: Activity8_1.c

Activity8_1.c*

```
1  #pragma config(Sensor, S1,     touchSensor,    sensorEV3_Touch)
2  #pragma config(Sensor, S2,     touchSensor2,   sensorEV3_Touch)
3  #pragma config(Sensor, S3,     colorSensor,    sensorEV3_Color)
4  #pragma config(Sensor, S4,     sonarSensor,    sensorEV3_Ultrasonic)
5  #pragma config(Motor,  motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
6  #pragma config(Motor,  motorC, leftMotor,  tmotorEV3_Large, PIDControl, driveLeft, encoder)
7  //*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
8
9  /*
10 Create a program to control the positive speed of the robot by a press of a
11 Touch Sensor.
12 */
13
14 task main()
15 {
16   //Create an integer (whole number) variable to store our speed value.
17   int speed = 0;
18
19   //Repeat our control loop forever.
20   while(true)
21   {
22     //When I press the touch sensor button.
23     if(getTouchValue(touchSensor) == 1)
24     {
25
26     //Add 10 to our 'speed' variable
27       if(speed < 100) speed = speed + 10;
28
29       //Create a loop to wait for the touch sensor button to be released.
30       while(getTouchValue(touchSensor) == 1)
31       {
32         //Wait for button to be released.
33       sleep(10);
34       }
35
36         //Set motorB and motorC speed to the value of the 'speed' variable
37     setMotorSpeed(motorB, speed);
38     setMotorSpeed(motorC, speed);
39     }
40   }
41 }
42
```

This code may be subject to change as ROBOTC is updated periodically.

Activity 8
Appendix

Possible ROBOTC Solution
FILENAME: Activity8_2.c

Activity8_2.c*

```
#pragma config(Sensor, S1,     touchSensor,    sensorEV3_Touch)
#pragma config(Sensor, S2,     touchSensor2,   sensorEV3_Touch)
#pragma config(Sensor, S3,     colorSensor,    sensorEV3_Color)
#pragma config(Sensor, S4,     sonarSensor,    sensorEV3_Ultrasonic)
#pragma config(Motor, motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor, motorC, leftMotor,  tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Create a program to control the positive and negative speed of the robot
by the press of two Touch Sensors.
*/

task main()
{
  //Create an integer (whole number) variable to store our speed value.
  int speed = 0;

  //Repeat our control loop forever.
  while(true)
  {
    //When I press the touch sensor button.
    if(getTouchValue(touchSensor) == 1)
    {
      //Add 10 to our 'speed' variable
      if(speed < 100) speed = speed + 10;

      //Create a loop to wait for the touch sensor button to be released.
      while(getTouchValue(touchSensor) == 1)
      {
        //Wait for button to be released
        sleep(10);
      }
      //Set motorB and motorC speed to the value of the 'speed' variable.
      setMotorSpeed(motorB, speed);
      setMotorSpeed(motorC, speed);
    }

    //When I press the touch sensor #2 button.
    if(getTouchValue(touchSensor2) == 1)
    {
      //Subtract 10 from our 'speed' variable.
      if(speed > -100) speed = speed - 10;

      //Create a loop to wait for the touch sensor #2 button to be released.
      while(getTouchValue(touchSensor2) == 1)
      {
        //Wait for button #2 to be released.
        sleep(10);
      }
      //Set motorB and motorC speed to the value of the 'speed' variable.
      setMotorSpeed(motorB, speed);
      setMotorSpeed(motorC, speed);
    }
  }
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 8
Appendix

Possible ROBOTC Solution
FILENAME: Activity8_3.c

Activity8_3.c*

```
#pragma config(Sensor, S1,     touchSensor,    sensorEV3_Touch)
#pragma config(Sensor, S2,     touchSensor2,   sensorEV3_Touch)
#pragma config(Sensor, S3,     colorSensor,    sensorEV3_Color)
#pragma config(Sensor, S4,     sonarSensor,    sensorEV3_Ultrasonic)
#pragma config(Motor, motorB, rightMotor, tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor, motorC, leftMotor, tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Create a program to control the positive and negative speed of the robot
by a press of two Touch Sensors. Display the speed on the EV3 screen.
*/

//Create an integer (whole number) variable to store our speed value.
int speed = 0;

void Accelerate()
{
  //When I press the touch sensor button.
  if(getTouchValue(touchSensor) == 1)
  {
    //Add 10 to our 'speed' variable
    if(speed < 100) speed = speed + 10;

    //Create a loop to wait for the touch sensor button to be released.
    while(getTouchValue(touchSensor) == 1)
    {
      //Wait for button to be released
      sleep(10);
    }
    //Set motorB and motorC speed to the value of the 'speed' variable.
    setMotorSpeed(motorB, speed);
    setMotorSpeed(motorC, speed);
  }
}
void Decelerate()
{
  //When I press the touch sensor #2 button.
  if(getTouchValue(touchSensor2) == 1)
  {
    //Subtract 10 from our 'speed' variable.
      if(speed > -100) speed = speed - 10;

    //Create a loop to wait for the touch sensor #2 button to be released.
    while(getTouchValue(touchSensor2) == 1)
    {
      //Wait for button to be released
      sleep(10);
    }
    //Set motorB and motorC speed to the value of the 'speed' variable.
    setMotorSpeed(motorB, speed);
    setMotorSpeed(motorC, speed);
  }
}
task main()
{
  //Create a string to store our text to be displayed to the LCD.
  string displaySpeed;

  //Repeat our control loop forever.
  while(true)
  {
    Accelerate();
    Decelerate();
    //Format our string to display "Speed: 50" or similar values.
    stringFormat(displaySpeed, "Speed: %d   ", speed);
    //Draw the string on the LCD screen.
    drawTextAt(49, 54, displaySpeed);
    //Set motorB and motorC speed to the value of the 'speed' variable.
    setMotorSpeed(motorB, speed);
    setMotorSpeed(motorC, speed);
  }
}

```

This code may be subject to change as ROBOTC is updated periodically.

Activity 9
Roaming Robots

LEGOeducation.com.au

LEGO mindstorms education EV3

Activity 9

Roaming Robots

This activity will focus on arrays and how they can be used in order to control the actions of the students' wheeled robot.

During the course of the activity, the students will discover what arrays are, how they work and why they are important to computer programming. The students will also learn how to incorporate an array into their program.

The students will create a program that mimics the program that is used in certain programmable toys and robots, such as the BeeBot.

During the introduction, you will use the Colour Sorter to demonstrate an array in action. The students can watch the video found in the 'Model Core Set' section of the software or the model can simply be built. If you are building the model, make sure that this model is built and programmed prior to beginning the activity.

The Colour Sorter model can be found in the 'Model Core Set' section of the EV3 Software. Click on 'Model Instructions' and you will see the Colour Sorter model near the top of the list.

Click on this tab and then click 'Open' to launch the model tutorial (you can also simply double-click the tab to open it). Here, you will find the building instructions for the model along with a sample program. Once the model is built, simply download the program and run it using the video on the first page as a guide.

This activity differs from the others as it has only two main activities, each of which will take a considerable amount of time for the students to complete.

Activity 9
Lesson Plan

OUTCOMES

Students will be able to:

- make appropriate use of data structures such as lists, tables and arrays
- understand simple Boolean logic (such as AND, OR and NOT) and some of its uses in circuits and programming
- use the Brick Buttons to control the movement of their wheeled robot
- use the Variable Block to store information
- use the Array Operations Block

INTRODUCTION

- Explain to the students that they are going to program their wheeled robot to respond (i.e. move) according to the instructions that are given to it through the buttons on the EV3 Brick (forwards, backwards, left and right).
- Point out that an array is a temporary way of storing a collection of numbers in sequence. These numbers can then be used later in the program.
- Run the Colour Sorter model a number of times and ask the students to write pseudo-code to describe what is happening.
- Liken programming the students' wheeled robot to programming a satellite navigation device (or sat nav). How can they do this using their wheeled robot? In groups, ask the students to discuss how they could devise a solution in order to program their wheeled robot to move around a course. Ask how a sat nav works.

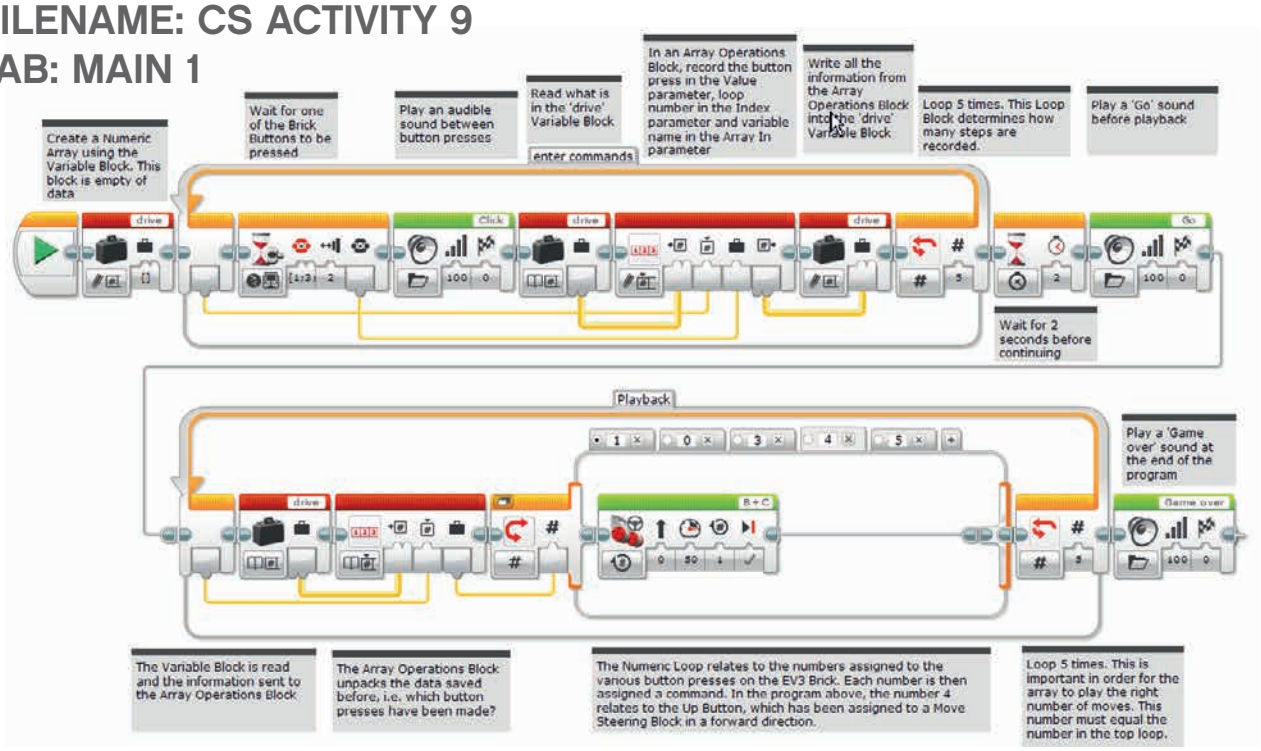
Activity 9
Lesson Plan

- Make sure that the students know that the Array Operations Block will be needed. Explain that the block is used in order to store data within the Variable Block, which is also needed. This data can then be retrieved and used in a program.
- Show the Colour Sorter model to the students again. Ask the students how they think it works. Make sure that they understand that the program 'remembers' the order of the coloured tiles and then places them in the predefined cups.
- Is it possible to do this for their wheeled robot?

MAIN CHALLENGE 1

- Explore the Colour Sorter program. Ask the students, in groups, to explain what is happening within the program. Does it make sense to them? Ask the students to write out the program.
- Take the students through the program, making sure that they understand the basics of creating a variable while using the Array Operations Block, and the need to read and then write to the variable. The students need also to understand that the process is as follows: *read the variable, write to it, then read once more and use the information.*
- Challenge the students to write a program (similar to a BeeBot) that will drive their wheeled robot around a course. In the beginning, limit the program to five steps.
- Explain that the program will need to be divided into two sections. The first section is to gather the information and the second is to use the information that has been gathered – just like the Colour Sorter does in the example.

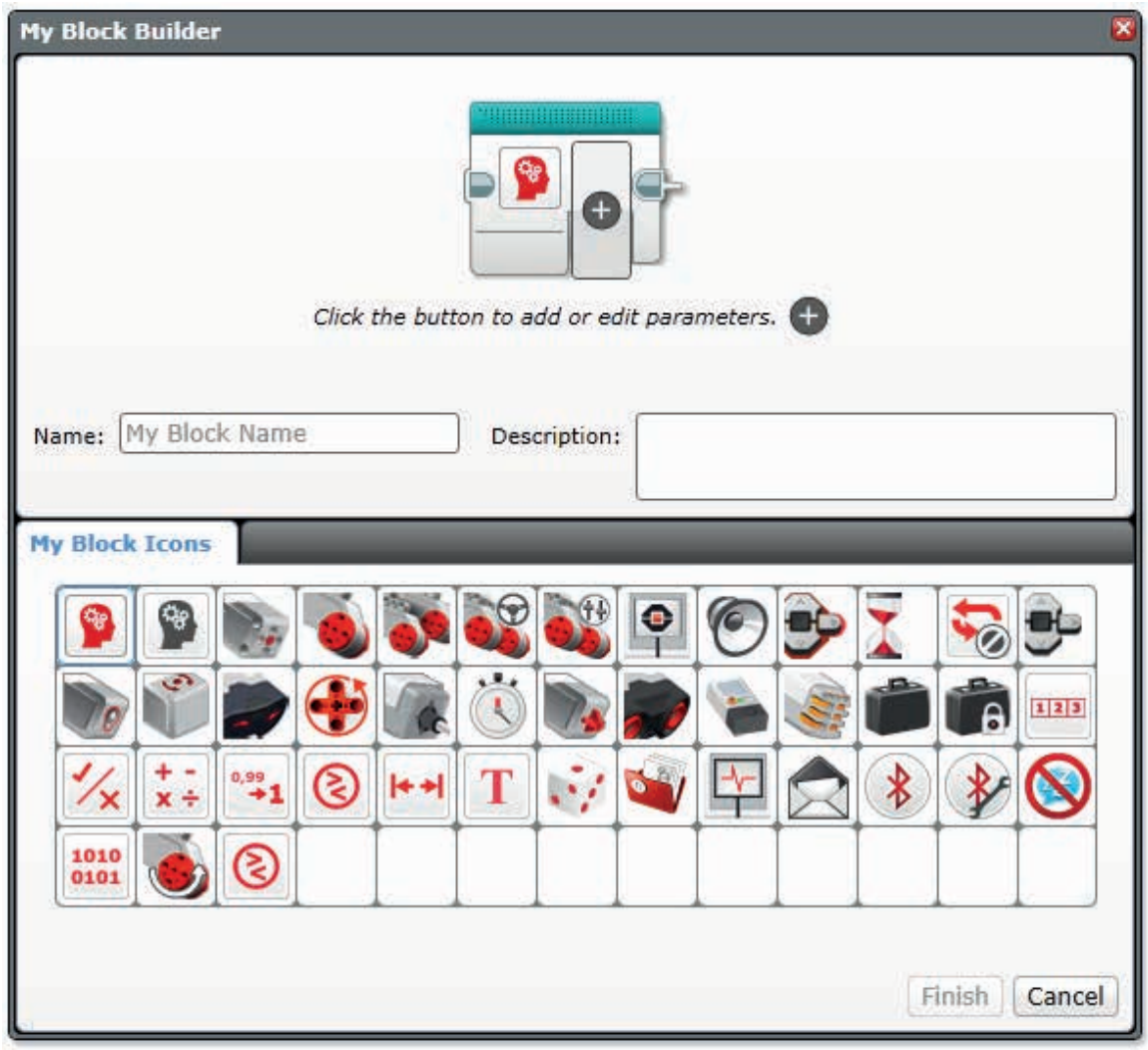
POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 9
TAB: MAIN 1



Activity 9
Lesson Plan

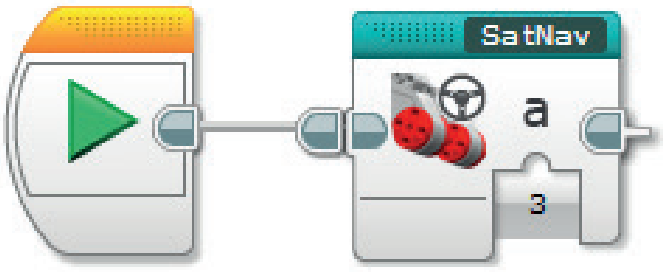
MAIN CHALLENGE 2

- In Activity 1, we were limited to the number contained within the loop.
- Demonstrate to the students how to create a My Block and then add a parameter to it.
- Notice that a new tab is added to the Programming Palette showing what is contained within the My Block. To complete the My Block, the user needs to connect the 'a' parameter to two inputs within the program. In our program, this will be to the two Loop Blocks. This allows data to be directly entered into the My Block rather than into the program contained within it.
- Point out that this allows the students to enter the desired number of steps easily. Make sure that the students understand that adding a parameter in the My Block Builder allows them to enter data on the My Block that is created.

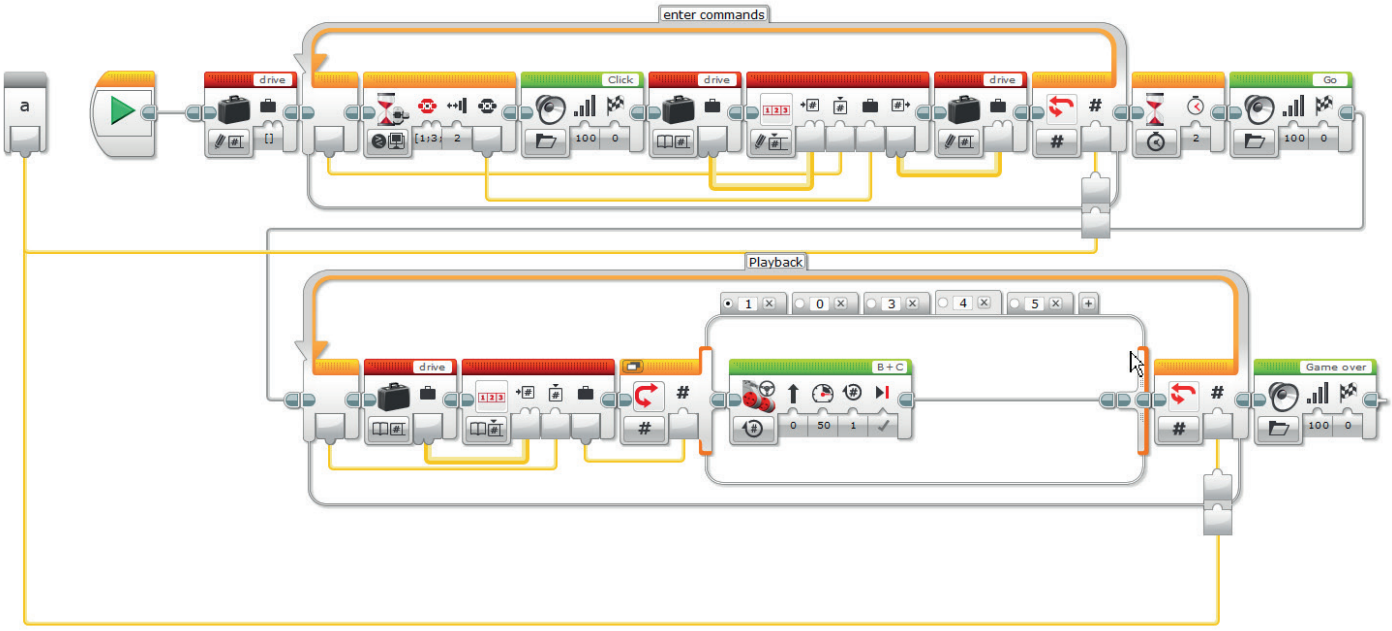


Activity 9
Lesson Plan

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 9
TAB: MAIN 2



POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 9
TAB: SATNAV



CLASS DISCUSSION

- Bring the group together to share their programming successes. Ask how they could have improved their programs.
- Does creating a My Block with a parameter make it easier for the students to program the right number of steps?
- Ask the students to consider alternative solutions. Can they come up with a new way of programming their wheeled robot to move around the room?

Activity 9
Student Worksheets

CHALLENGES FOR TODAY

Today you are going to learn how to use an array. The Array Operations Block is an important block that allows lots of information to be stored and then reused when it is needed. You will create an automated car that has been programmed to move in a series of steps. The EV3 Brick Buttons will allow directions to be chosen. Have fun!

CHALLENGE 1

Having watched the Colour Sorter in action, it's now your turn to create an array so that you can program your wheeled robot to move around the room using the buttons on the EV3 Brick. The four Brick Buttons can be used as controls (left, right, backwards and forwards).

To start, limit the program to five commands by entering '5' into the Loop Block.

Tip 1: Your program will have two distinct phases:

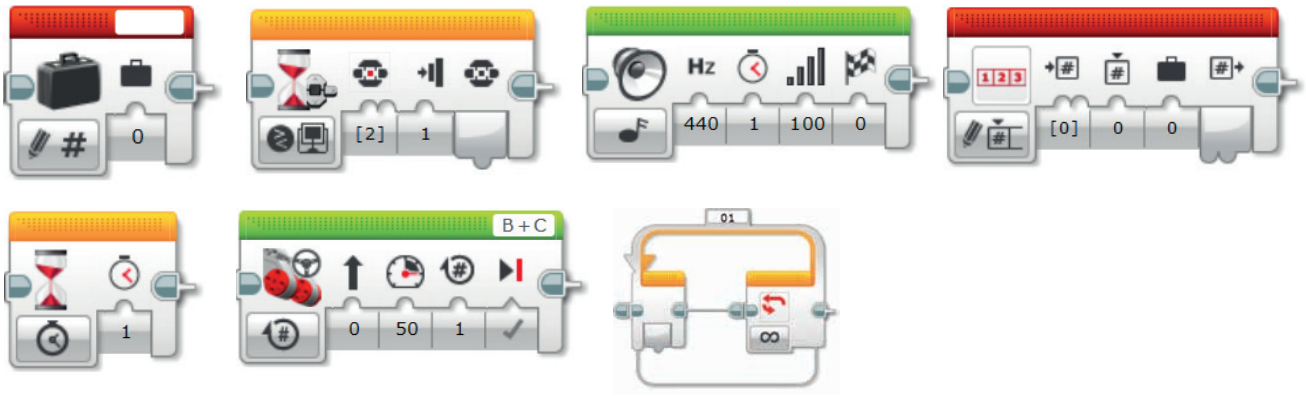
1. Collecting the data
2. Using that data

Tip 2: Two Loop Blocks will be needed for this activity in order to allow for the two phases mentioned above.

Tip 3: Using the Variable Block often requires a three-step process:

read the Variable Block, add information to it and then write to the Variable Block to save the new data.

Blocks to Consider



Plan your program first. Write it in pseudo-code below:

Activity 9
Student Worksheets

CHALLENGE 2

Create a MyBlock in order to easily be able to edit the number of steps in the program.

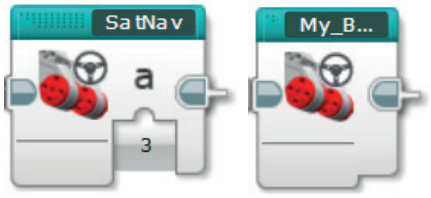
To change the number of movement steps from 5 to something else will require you to edit both Loop Blocks within the program. This can be made much simpler by creating a My Block with a parameter. The My Block will allow the number of loops to be changed easily and clearly. Your task is to create a My Block of the program that you created in Challenge 1.

Tip 1: When creating a My Block, highlight the blocks that need to be included but NOT the Start Block.

Tip 2: When you need to enter parameters at a later stage, ensure that a parameter has been added to the My Block as shown below. Use the '+' key when creating the block.

Tip 3: The parameter must be joined to the input on the block within the program. In our case, the two loops.

Blocks to Consider



Plan your program first. Write it in pseudo-code below:

Activity 9
Student Worksheets

After a programming activity, it is important to note down your thoughts and observations. Consider the following points and then in the box below record how the activity went.

- How could you improve your program?
- Could your program have been more streamlined? Have you used too many blocks? Is there a more efficient way of building your program?
- What examples of real-world applications could you see your program being used in?

Thoughts and Observations

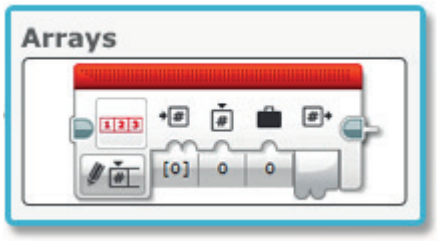
Activity 9
Teacher Notes

ROBOT EDUCATOR TUTORIALS

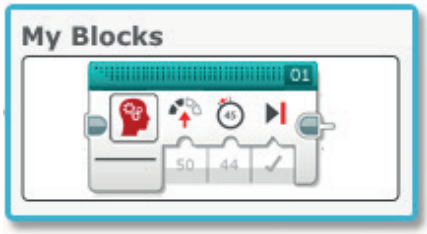
The following Robot Educator Tutorials will help teachers and their students to solve the challenges.

NEW ROBOT EDUCATOR TUTORIALS

Beyond Basics > Arrays

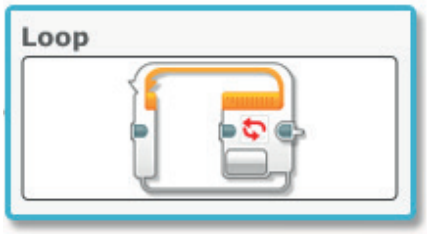


Tools > My Blocks

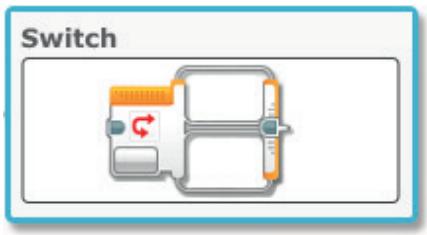


PREVIOUSLY COVERED ROBOT EDUCATOR TUTORIALS

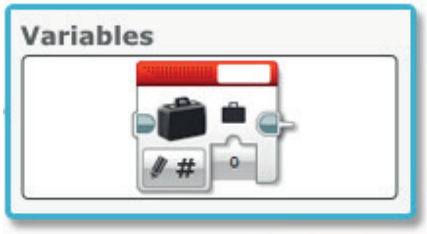
Beyond Basics > Loop



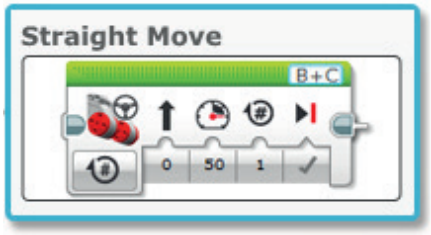
Beyond Basics > Switch



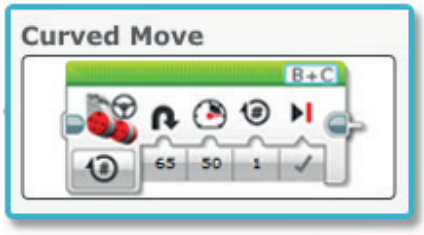
Beyond Basics > Variables



Basics > Straight Move



Basics > Curved Move

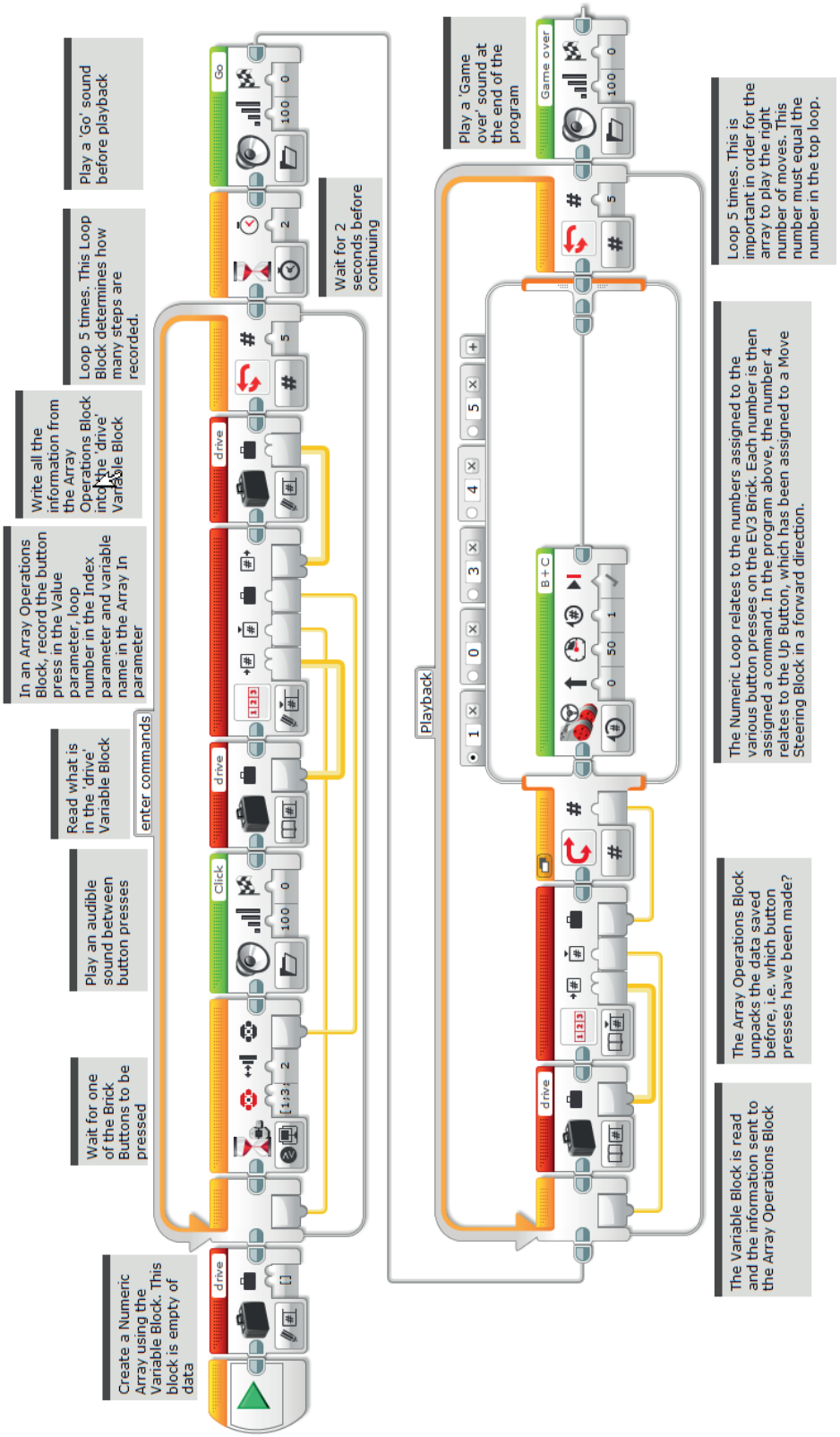


Activity 9
Appendix

APPENDIX FOR ACTIVITY 9: LARGE IMAGES AND PROGRAMS

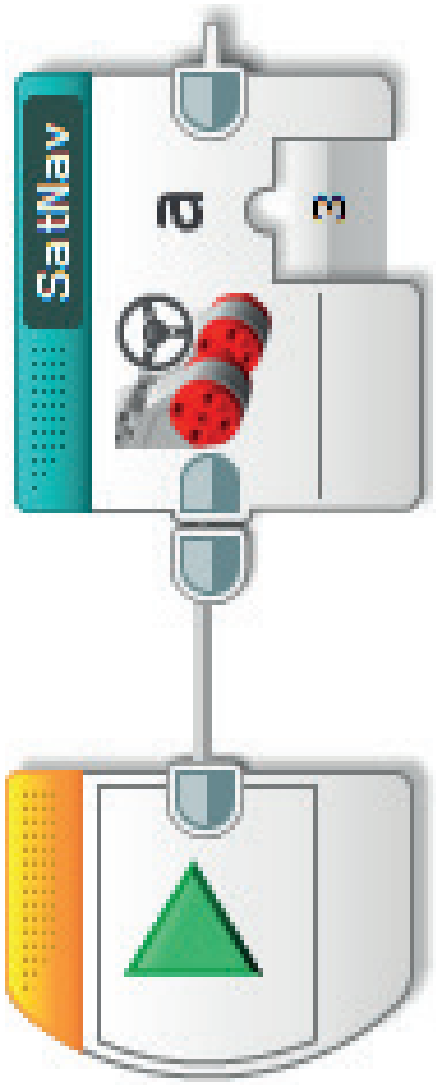
Activity 9
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 9
TAB: MAIN 1



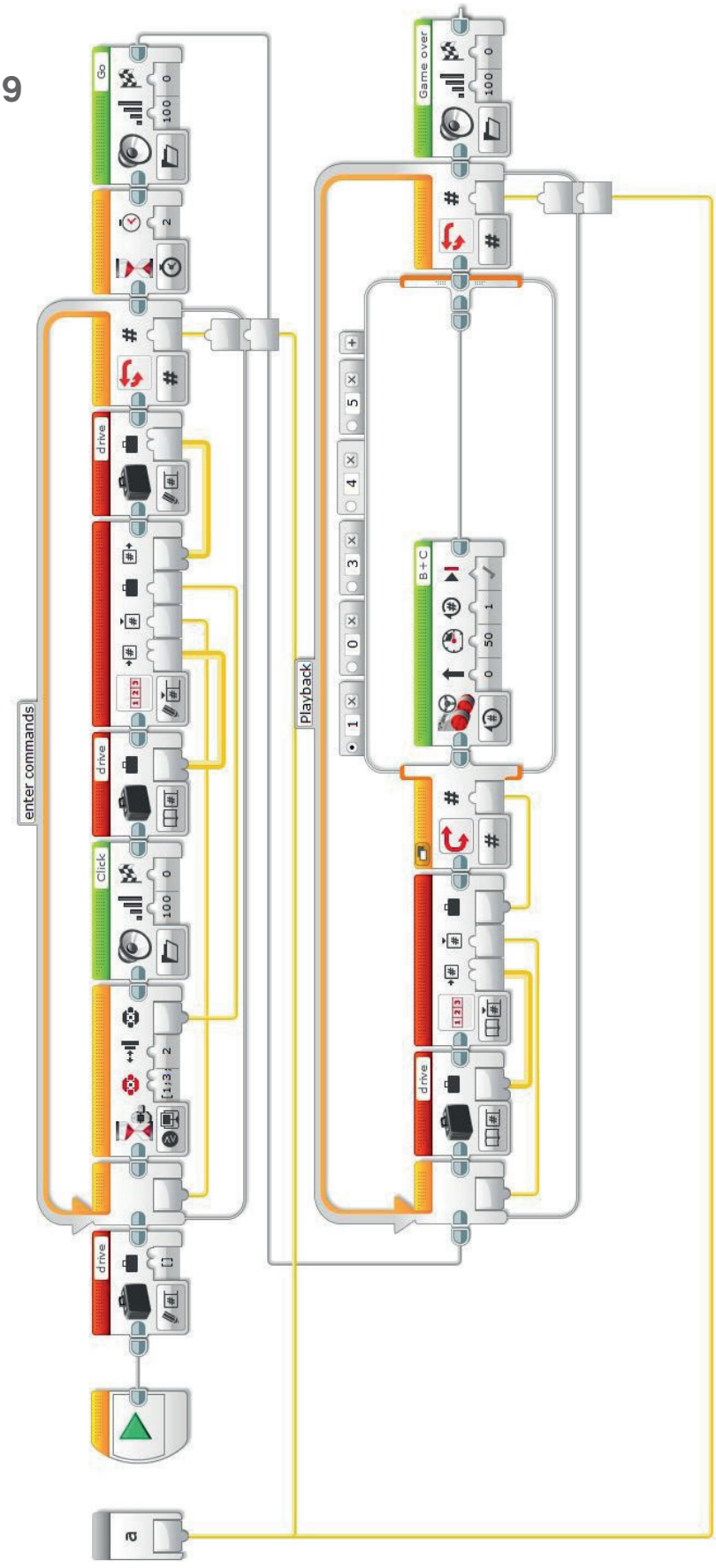
Activity 9
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 9
TAB: MAIN 2



Activity 9
Appendix

POSSIBLE SOLUTION
FILENAME: CS ACTIVITY 9
TAB: SATNAV



Activity 9
Appendix

Possible ROBOTC Solution
FILENAME: Activity9_1.c

Activity9_1.c*

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*
Use the buttons on the EV3 brick to program the robot to move around.
5 commands can be entered into the EV3 brick.
Left Button   = 1
Right Button  = 3
Up Button     = 4
Down Button   = 5
*/

int drive[5];

task main()
{
  for(int i = 0; i < 5; i++) //i = i + 1
  {
    while(getButtonPress(buttonAny) == 0)
    {
      //Wait for Any Button to be pressed.
    }
    if(getButtonPress(buttonLeft) == 1)           drive[i] = 1;
    else if(getButtonPress(buttonRight) == 1)     drive[i] = 3;
    else if(getButtonPress(buttonUp) == 1)        drive[i] = 4;
    else if(getButtonPress(buttonDown) == 1)      drive[i] = 5;

    playSoundFile("Click");
    while(bSoundActive)
    {
      sleep(10);
    }

    while(getButtonPress(buttonAny) == 1)
    {
      //Wait for All Buttons to be released.
      sleep(10);
    }
  }

  sleep(2000);
  playSoundFile("Go");
  while(bSoundActive)
  {
    sleep(10);
  }

  for(int i = 0; i < 5; i++)
  {
    if(drive[i] == 1)
    {
      //Turn Left Code.
      moveMotorTarget(motorC, 360, 50);
      waitUntilMotorStop(motorC);

    }
    else if(drive[i] == 3)
    {
      //Turn Right Code.
      moveMotorTarget(motorB, 360, 50);
      waitUntilMotorStop(motorB);
    }
    else if(drive[i] == 4)
    {
      //Forward Code.
      moveMotorTarget(motorB, 360, 50);
      moveMotorTarget(motorC, 360, 50);
      waitUntilMotorStop(motorB);
      waitUntilMotorStop(motorC);
    }
    else if(drive[i] == 5)
    {
      //Backward Code.
      moveMotorTarget(motorB, -360, -50);
      moveMotorTarget(motorC, -360, -50);
      waitUntilMotorStop(motorB);
      waitUntilMotorStop(motorC);
    }
  }

  playSoundFile("Game over");
  while(bSoundActive)
  {
    sleep(10);
  }
}
```

This code may be subject to change as ROBOTC is updated periodically.

Activity 9
Appendix

Possible ROBOTC Solution
FILENAME: Activity9_2.c

Activity9_2.c*

```
#pragma config(StandardModel, "EV3_REMBOT")
//*!!Code automatically generated by 'ROBOTC' configuration
//*!!Wizard !!*//

//Left Button =1; Right Button =3; Up Button =4; Down Button =5
//Maximum of 100 steps
int drive[100];

void recordSteps(int numberOfSteps)
{
  for(int i = 0; i < numberOfSteps; i++) //i = i + 1
  {
    while(getButtonPress(buttonAny) == 0)
    {
      //Wait for Any Button to be pressed
      sleep(10);
    }

    if(getButtonPress(buttonLeft) == 1)          drive[i] = 1;
    else if(getButtonPress(buttonRight) == 1)    drive[i] = 3;
    else if(getButtonPress(buttonUp) == 1)       drive[i] = 4;
    else if(getButtonPress(buttonDown) == 1)     drive[i] = 5;

    playSoundFile("Click");
    while(bSoundActive) sleep(10);

    while(getButtonPress(buttonAny) == 1)
    {
      //Wait for All Buttons to be released.
      sleep(10);
    }
  }
}

void playSteps(int numberOfSteps)
{
  for(int i = 0; i < numberOfSteps; i++)
  {
    if(drive[i] == 0)
    {
      //No Step Found, End Code.
      return;
    }

    else if(drive[i] == 1)
    {
      //Turn Left Code
      moveMotorTarget(motorC, 360, 50);
      waitUntilMotorStop(motorC);

    }
    else if(drive[i] == 3)
    {
      //Turn Right Code
      moveMotorTarget(motorB, 360, 50);
      waitUntilMotorStop(motorB);
    }
    else if(drive[i] == 4)
    {
      //Forward Code
      moveMotorTarget(motorB, 360, 50);
      moveMotorTarget(motorC, 360, 50);
      waitUntilMotorStop(motorB);
      waitUntilMotorStop(motorC);
    }
    else if(drive[i] == 5)
    {
      //Backward Code
      moveMotorTarget(motorB, -360, -50);
      moveMotorTarget(motorC, -360, -50);
      waitUntilMotorStop(motorB);
      waitUntilMotorStop(motorC);
    }
  }
}

task main()
{
  recordSteps(8);

  sleep(2000);
  playSoundFile("Go");
  while(bSoundActive) sleep(10);

  playSteps(8);

  playSoundFile("Game over");
  while(bSoundActive) sleep(10);
}
```

This code may be subject to change as ROBOTC is updated periodically.

Final Project

Activity 10:
Designing a Driverless, Automated, Wheeled Robot

Activity 11:
Building and Programming a Driverless, Automated, Wheeled Robot

Activity 12:
Reviewing, Revising, and Presenting Your Driverless, Automated, Wheeled Robot

Activity 10

Designing a Driverless, Automated, Wheeled Robot

LEGOeducation.com.au

LEGO education

Activity 10

Designing a Driverless, Automated, Wheeled Robot

Over the past nine activities, the students have spent time learning about the different aspects involved in automating a wheeled robot, from the information passed to passengers and pedestrians to the car being able to follow a pre-determined route. Now it is time for the students to design their own 'automated car', while applying as many of the features they have experienced over the course of this project as they can.

As with real cars, the students' wheeled robot will be judged based on the features it has. For example, many car manufacturers offer their cars in a range of models, from the very basic to the 'top of the range' models with a host of features.

For this project, the students will follow the engineering process that is defined in the LEGO® MINDSTORMS® Education EV3 Design Engineering Projects activity pack – an add-on pack designed especially for the STEM curriculum (but which can also be used in a computing context if practical activities are required).

This activity is about bringing together all of the aspects of programming that the students have experienced so far. In groups, the students will mind map which aspects are required in order to develop the driverless, automated, wheeled robot. The students should be encouraged to follow the engineering process found in the EV3 Design Engineering Projects pack. The process for today is as follows:

- **Design Brief**
- **Brainstorm**
- **Select the Best Solution**
- Build and Program a Solution
- Test and Analyse
- Review and Revise
- Communicate

The first three bullet points will be covered during this activity.
The remaining bullet points will be covered in subsequent activities.

Activity 10
Lesson Plan

OUTCOMES

Students will be able to:

- design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems

VOCABULARY

design, mind mapping, brainstorm, solution, design brief

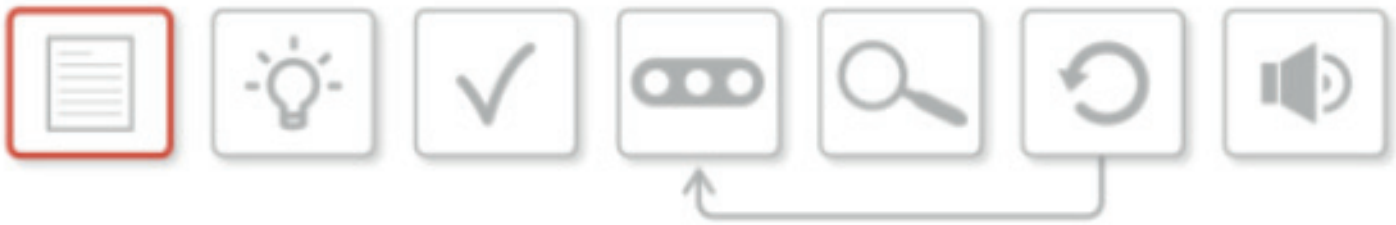
INTRODUCTION – GIVE THE DESIGN BRIEF

- Explain the design engineering process as shown above. Point out the seven phases of the process and explain to the students that they will work through this process over the next three activities.
- Activity 10 (this activity) – Design Brief, Brainstorm (mind map) and Select the Best Solution
- Activity 11 – Build and Program a Solution, Test and Analyse
- Activity 12 – Review and Revise, Communicate Your Design
- The design brief for this project is as follows:

  Design and build a driverless, automated, wheeled robot that can get from point A to point B while avoiding obstacles.

If we think back to the classifications of cars, this would be the standard, basic model. However, the students can add any of the following features to their designs in order to raise the 'class' of their wheeled robot. The classification names may be changed to suit your students.

- Standard: The wheeled robot avoids obstacles
- Enhanced: Standard, plus the wheeled robot responds to traffic signals and pedestrian warnings
- Superior: Enhanced, plus keyless start
- Excelsior: Superior, plus cruise control



Activity 10
Lesson Plan

The Final Project: Main Challenge

BRAINSTORMING THE IDEA (MIND MAPPING)

- Look at the design brief. Ask the students to consider which class of car they intend to design.
- Discuss with the students the need to work out the programming skills needed for the design.
- Questions for the students to ask themselves: Which sensors will be needed? Do they have enough sensors? What compromises are needed?
- Ask the students to research, explore and share ideas for solving the design brief.
- Ask them to look at current driverless vehicles such as:

  - Automated trains

  - BMW, Google, and Tesla's driverless concept cars

  The students can search online for videos and articles about each of these examples and compare their features, and then discuss which features from these systems could be applied / integrated into their designs.
- Encourage the students to explore the other LEGO® bricks within the EV3 kits in order to get ideas and inspiration. Most of the students will follow the Robot Educator model, but some groups may be more adventurous.

ASSIGNING ROLES

- What roles will be needed in order to make sure that the design brief becomes reality? Ask the students to explore the roles that are needed within the group. In every group there will be different skill sets. We all possess a range of skills. Encourage the students to discuss their strengths and weaknesses. Who has strengths as a builder? Who has particular programming strengths? Roles can then be assigned.
- **Possible roles within the team:**

  - Designer

  - Model Builder

  - Researcher

  - Marketer

  - Programmer

Activity 10
Lesson Plan

SELECTING THE BEST SOLUTION

- The students must be encouraged to consider the pros and cons of each idea they have come up with. They should then decide on a final design to carry forward.
- The students will then present their final choice to the class and explain the reasons for their decision.

The students should record every step of the process on their student worksheets or in the Content Editor within the EV3 Software.

CLASS DISCUSSION

- What are the next steps? Now that the solution has been chosen, the teams need to start building and programming their wheeled robot.
- Inform the students that in the next activity they will be building, programming and testing their wheeled robot.

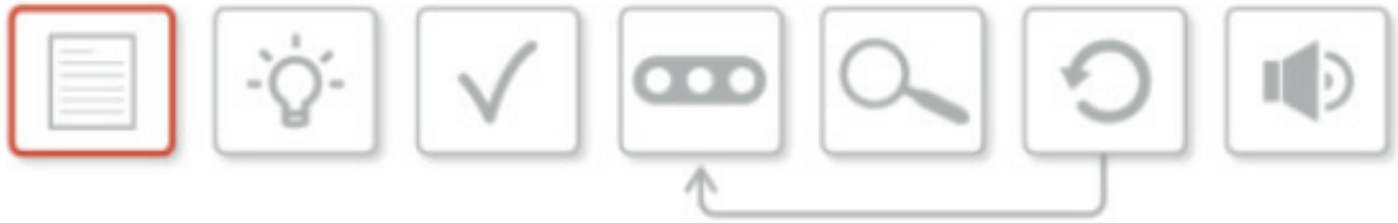
Activity 10
Student Worksheets

YOUR FINAL PROJECT: CHALLENGES FOR TODAY

Think about the design process.

Today you will need to address the first three stages: Receiving the **Design Brief, Brainstorming** within your group and **Selecting a Solution**.

Your teacher will have given you the design brief. Here it is again to remind you:



DESIGN AND BUILD A DRIVERLESS, AUTOMATED, WHEELED ROBOT THAT CAN GET FROM POINT A TO POINT B WHILE AVOIDING OBSTACLES.

The first part of the design process is working in your team to come up with a great idea. Once you have considered all of the pros and cons for your different ideas, you will need to select one of them and give reasons for your choice. Teamwork is important here and it is often one of the most challenging parts of a project. It may not always be your idea that is chosen. You will need to reach an agreement on the best solution and then present the reasons for choosing it.

Activity 10
Student Worksheets

BRAINSTORMING THE DESIGN BRIEF

Discuss the design brief. Which version of the automated, wheeled robot would your group like to design?

- **Standard:** The wheeled robot avoids obstacles
- **Enhanced:** Standard, plus the wheeled robot responds to traffic signals and pedestrian warnings
- **Superior:** Enhanced, plus keyless start
- **Excelsior:** Superior, plus cruise control

Now brainstorm ideas for the design. Which features do you intend to include in the build and consequently in the programming? Will changes be needed to the physical design of the robot? Note down all of your thoughts and sketches below.

Activity 10
Student Worksheets

ASSIGNING ROLES

Every project needs a team and you are all part of that team.

What tasks will need to be done?

Note down the different roles you think are needed within your team and assign those roles to your team members.

Activity 10
Student Worksheets

CHOOSING AND PRESENTING THE BEST SOLUTION

It's time to select the best solution from your brainstorming.

Prepare a short presentation, explaining to your teacher and fellow students which design option you have opted for, and give the reasons for your choice.

You should also explain the roles that have been assigned to each group member.

Explain how you have reached this conclusion, highlighting the strengths and weaknesses within the group.

Your presentation should be no longer than five minutes. Who presents and how it is presented is entirely up to you.

Use the box below to make notes.

Activity 11

Building and Programming a Driverless, Automated, Wheeled Robot

LEGOeducation.com.au

LEGO mindstorms education EV3

Activity 11

Building and Programming a Driverless, Automated, Wheeled Robot

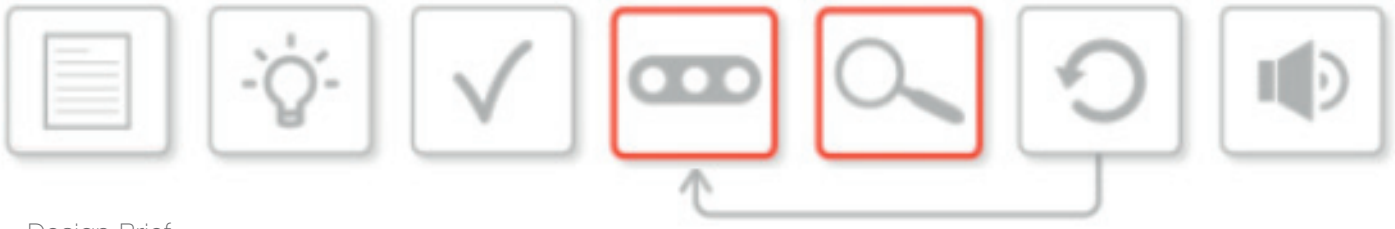
Over the past ten activities, the students have spent time learning about the different aspects involved in automating a wheeled robot, from the information passed to passengers and pedestrians to the car being able to follow a pre-determined route. Now it is time for the students to design their own 'automated car', while applying as many of the features they have experienced over the course of this project as they can.

As with real cars, the students' wheeled robot will be judged based on the features it has. For example, many car manufacturers offer their cars in a range of models, from the very basic to the 'top of the range' models with a host of features.

For this project, the students will follow the engineering process that is defined in the LEGO® MINDSTORMS® EV3 Design Engineering Projects activity pack – an add-on pack designed especially for the STEM curriculum (but which can also be used in a computing context if practical activities are required).

This activity is about building the students' wheeled robot, programming it and then analysing the build and programs.

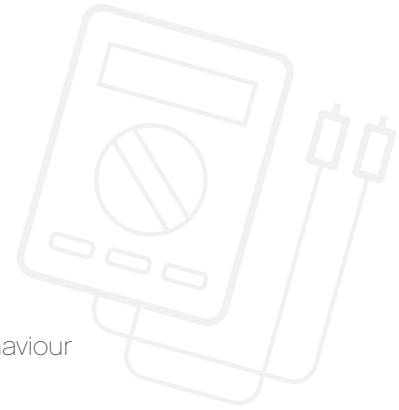
The students should be encouraged to follow the engineering process found in the EV3 Design Engineering Projects pack. The process for today is as follows:



- Design Brief
- Brainstorm
- Select the Best Solution
- **Build and Program a Solution**
- **Test and Analyse**
- Review and Revise
- Communicate

The highlighted bullet points will be covered during this activity.
The remaining bullet points will be covered in the next activity.

Activity 11
Lesson Plan



OUTCOMES

Students will be able to:

- design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- use objectivity when analysing their work
- work together as a team

VOCABULARY

design, solution, design brief, test, analyse

INTRODUCTION

- Recap on the work from Activity 10 and ensure that the students are at the stage where they can confidently move on to the requirements of the engineering process that are associated with this activity.
- Explain to the students that during this activity they will be addressing the next two stages of the process, namely building and programming their wheeled robot, and testing and analysing whether it is fit for the purpose.
- Remind the students of the overall design brief:

  Design and build a driverless, automated wheeled robot that can get from point A to point B while avoiding obstacles.

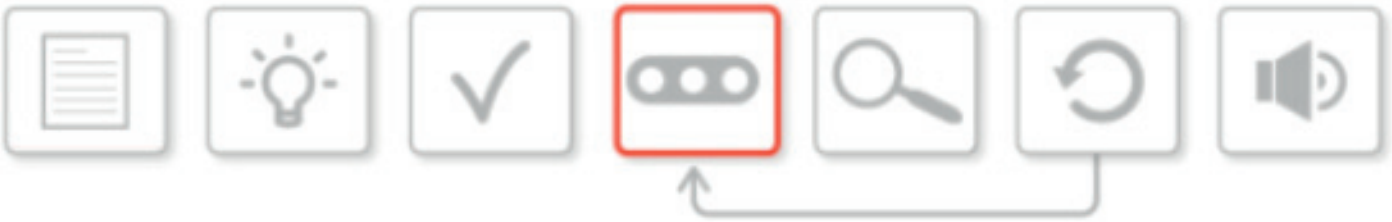
If we think back to the classifications of cars, this would be the standard, basic model. However, the students can add any of the following features to their designs in order to raise the 'class' of their wheeled robot. The classification names may be changed to suit your students.

- Standard: The wheeled robot avoids obstacles
- Enhanced: Standard, plus the wheeled robot responds to traffic signals and pedestrian warnings
- Superior: Enhanced, plus keyless start
- Excelsior: Superior, plus cruise control

Activity 11
Lesson Plan

The Final Project: Main Challenge

BUILD AND PROGRAM A SOLUTION



- The students will need to look back at the work they did in Activity 10, and in particular their design for their wheeled robot build.
- Encourage the students to recap and discuss the design and 'class' of build they opted for.
- They may wish to use the time here to begin dividing tasks according to the roles they were assigned in Activity 10.
- You may wish to encourage the groups to split their tasks, so that a small group are building while the rest focus on programming.
- Remind the students that they should not be expecting to solve the design brief with the first build / program and that they will be required to test and analyse what they build.

TEST AND ANALYSE

- This process will be ongoing, and the students will need reminding of this.
- It is important to point out to the students that they should be constantly analysing their builds and programs.
- Some students may wish to complete the whole build and program before testing, while others may wish to take a more iterative approach. Students should feel free to employ whichever technique works for them. This is a chance for them to be self-directed learners.
- Remind the students during this phase that they should be constantly referring to the design brief and to their own designs and ideas, and that those ideas may change as the activities progress, particularly in Activity 12.

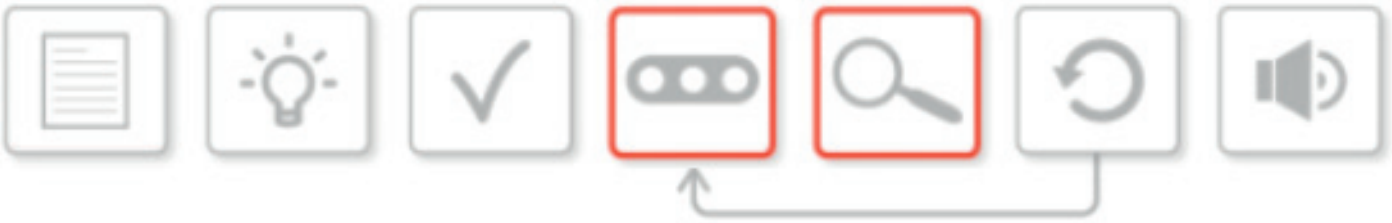
Activity 11
Lesson Plan

The students should record every step of the process on their student worksheets or in the Content Editor within the EV3 Software.

CLASS DISCUSSION

- What are the next steps? Now that the groups have built and programmed their wheeled robots, the next activity will focus on reviewing what they have done, revising their build and program if necessary, and presenting what they have done to the rest of the group.

YOUR FINAL PROJECT: CHALLENGES FOR TODAY



In this activity, you will need to refer back constantly to the design brief and to the build and programming solutions that you worked on last time.

Remember, your design brief was:

Design and build a driverless, automated wheeled robot that can get from point A to point B while avoiding obstacles.

Look at the diagram of the engineering process. Today, you will be focusing on building and programming a solution, and testing and analysing.

Remember to record your build and programming solutions in the Content Editor of the EV3 Software or here on your worksheet.

Activity 11
Student Worksheets

BUILD AND PROGRAM A SOLUTION

Look back at the work you did in Activity 10, in particular the design for your wheeled robot build.

Which version did you opt for (Standard, Enhanced, Superior or Excelsior)?

Recap on your thoughts from Activity 10 and discuss the design and programming implications.

You may wish to reassign the roles that you had in Activity 10, and use the time wisely by dividing your tasks.

Do you want to split your group, so that some of you are building while the rest are focused on programming?

You will probably not 'solve' the design brief or even your own design with your first attempt. You will need to constantly evaluate your build and debug your programs in order to achieve what you have set out to do today.

Record your work below or in the Content Editor:

Activity 11
Student Worksheets

TEST AND ANALYSE

This is an ongoing process and it should not be left until the end of the activity. Testing and analysing are very important factors in the engineering process.

You may wish to complete the whole build and program before testing it. Alternatively, you may wish to go through the process one small step at a time.

You should discuss this within your group and employ whichever technique works best for you.

During this phase, you should be constantly referring to the design brief and to your own designs and ideas.

When you test, you will need to ask yourselves the following questions:

- Does our wheeled robot fulfil the design brief?
- Does it look like our design?
- Does it do what we want it to, i.e. does our program work?

You may find that your ideas change and evolve as the activity progresses. This is all part of the process, and it should be recorded either on this worksheet or in the EV3 Software Content Editor.

Activity 12

Reviewing, Revising, and Presenting Your Driverless, Automated, Wheeled Robot

LEGOeducation.com.au

LEGO mindstorms education EV3

Activity 12

Reviewing, Revising, and Presenting Your Driverless, Automated, Wheeled Robot

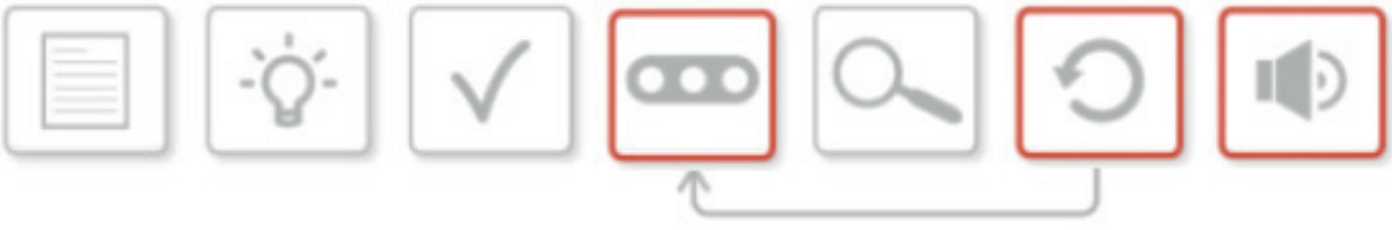
Over the past eleven activities, the students have spent time learning about the different aspects involved in automating a wheeled robot, from the information passed to passengers and pedestrians to the car being able to follow a pre-determined route. Now it is time for the students to design their own 'automated car', while applying as many of the features they have worked with over the course of this project as they can.

As with real cars, the students' wheeled robot will be judged based on the features it has. For example, many car manufacturers offer their cars in a range of models, from the very basic to the 'top of the range' models with a host of features.

For this project, the students will follow the engineering process that is defined in the Design Engineering Pack, an add-on pack designed especially for the STEM curriculum (but which can also be used in a computing context if practical activities are required).

This activity is a continuation and extension of Activity 11. During this activity, the students will be required to revisit their design and programming solutions from the last lesson, and critically review what they have done. They will need to work together in order to review what has worked well and what could be improved. From this activity, they will be given time to revise and refine their builds and programs. Finally, when all of the programming and testing is complete, they will need to give a presentation to their peers, focusing on the engineering process and how well they feel that they have achieved the design brief.

The students should be encouraged to follow the engineering process found in the EV3 Design Engineering Projects pack. The process for today is as follows:



Activity 12
Lesson Plan

- Design Brief
- Brainstorm
- Select the Best Solution
- **Build and Program a Solution**
- Test and Analyse
- **Review and Revise**
- **Communicate**

The highlighted bullet points will be covered during this activity.

OUTCOMES

Students will be able to:

- design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- use objectivity when analysing their work
- be able to critically review what they have done and work together to improve it
- present their journey through the engineering process to their peers
- work together as a team

VOCABULARY

design, solution, design brief, test, analyse, review, revise

INTRODUCTION

- Recap on the work from Activity 11 and ensure that the students are at the stage where they can confidently move on to the requirements of the engineering process that are associated with this activity.
- Explain to the students that they will begin by running the program they created during the last lesson, and that their tasks for today will focus on reviewing what they have done, the conclusions from which will determine which changes or improvements they might make to either their build or program.
- Point out to the students that any changes that they make to builds and programs may result in their wheeled robot moving from a more basic model to one that is more advanced.
- Tell the students to record any changes that they make to the builds and programs on either their student worksheets or in the Content Editor of the EV3 Software, as this will contribute towards their final presentation.

Activity 12
Lesson Plan

- Remind the students of the overall design brief:

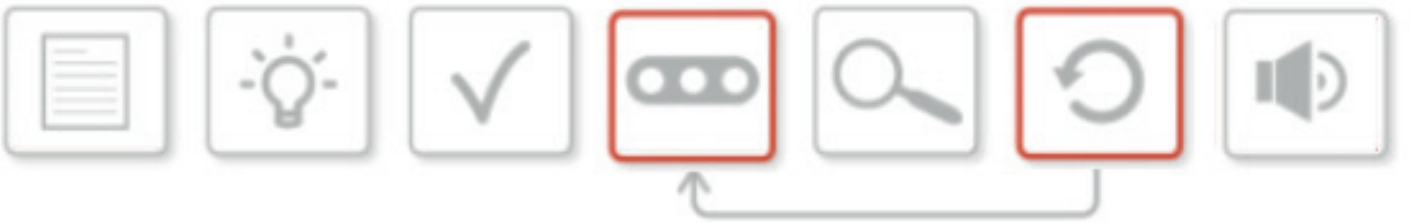
  Design and build a driverless, automated wheeled robot that can get from point A to point B while avoiding obstacles.

If we think back to the classifications of cars, this would be the standard, basic model. However, the students can add any of the following features to their designs in order to raise the 'class' of their wheeled robot. The classification names may be changed to suit your students.

- **Standard:** The wheeled robot avoids obstacles
- **Enhanced:** Standard, plus the wheeled robot responds to traffic signals and pedestrian warnings
- **Superior:** Enhanced, plus keyless start
- **Excelsior:** Superior, plus cruise control

The Final Project: Main Challenge 1

REVIEW AND REVISE/BUILD AND PROGRAM A SOLUTION



- Tell the students that during this part of the activity, they will be continually reviewing, testing and (possibly) rebuilding their wheeled robot.
- You may wish to devise success criteria that the students can refer to. The students could be involved in this process.
- It is important to remind the students of the original design brief.
- The students will work within their teams in order to run their programs and to discuss how they can improve how their wheeled robot behaves, and how this addresses the original design brief.
- They will then need to explore further programming options and where appropriate, changes to their build.
- Point out to the students that by continually reviewing, revising and redesigning, their wheeled robot could improve and change its 'classification'.
- Allow some time towards the end of this section for each group to run their program one final time. Bring all of the students together for this. When the wheeled robots are running, you and the students should be evaluating them against the success criteria that you have set.

Activity 12
Lesson Plan

The Final Project: Main Challenge 2

COMMUNICATE

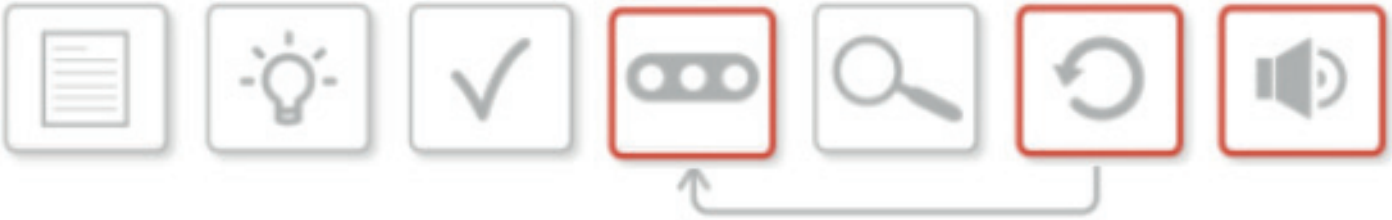
- Now that the students have gone through the engineering process thus far, have built the definitive version of their wheeled robot and programmed it to the best of their abilities, the final stage is to present their 'journey' to their peers.
- The students should have been keeping a record of their progress since Activity 10, either on their student worksheets or in the Content Editor of the EV3 Software. These records could include text notes, video, photographs and screenshots of their programming solutions.
- Explain to the students that they are to prepare a short presentation that explains their journey and decisions throughout the engineering process. The presentations should highlight their successes and failures, and communicate the classification that they feel their wheeled robot has achieved (Standard to Excelsior). They should also present the extent to which they have answered the design brief.
- The presentations may be in any format the students wish. They may wish to use presentation software, e.g. PowerPoint / Keynote / Prezi, or they may wish to use the EV3 Software Content Editor.

CLASS DISCUSSION

- In many ways, the student presentations are the classroom discussion for the last three activities. In presenting their work and appraising how they have addressed the design brief, the students are self-evaluating.
- After each presentation, it is important for each group to receive feedback from their peers and from you.

Activity 12
Lesson Plan

YOUR FINAL PROJECT: CHALLENGES FOR TODAY



This is the final activity in this particular unit of work. Once again, you will need to refer constantly to the design brief and to the build and programming solutions that you worked on last time.

Remember, your design brief was:

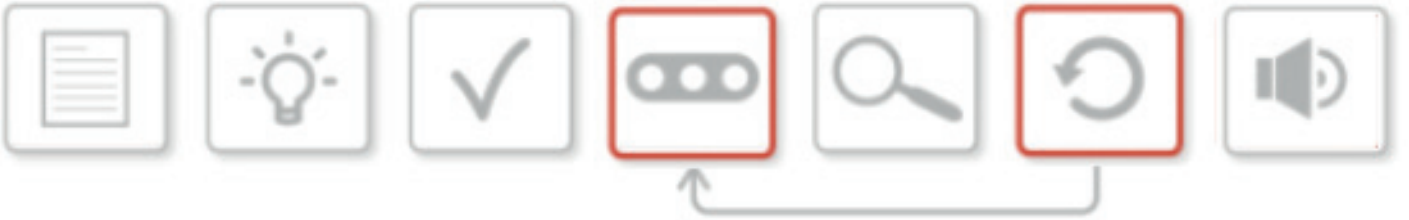
Design and build a driverless, automated wheeled robot that can get from point A to point B while avoiding obstacles.

Look at the diagram of the engineering process. Today, you will be focusing on **reviewing, revising (leading to further building and programming) and communicating.**

Remember to record your work in the Content Editor of the EV3 Software or here on your worksheet.

Activity 12
Student Worksheets

REVIEW AND REVISE/BUILD AND PROGRAM A SOLUTION



During this challenge, you will need to continually review, test and (quite possibly) rebuild your wheeled robot.

It is possible that you, your fellow students and your teacher have devised success criteria for this challenge. If so, you must remember to refer to them at all times.

What 'class' of wheeled robot did you settle on and build during the previous two activities? It is quite possible that, during this revision and review stage, you will redesign and reprogram your wheeled robot so that it evolves into a higher 'class'.

In your teams, run the program that you created during the last lesson. Does it fulfil the design brief? Could it be improved either by reprogramming or rebuilding (or both)? Explore further programming options and where appropriate, change your build.

Towards the end of this activity, you will demonstrate your wheeled robot to the rest of the group and measure it against the design brief and your success criteria.

Record your work below or in the Content Editor.

Activity 12
Student Worksheets

COMMUNICATE

Now that you have come this far through the whole engineering process and have built and programmed the definitive version of your wheeled robot, the final stage is to present what you have done to the other groups.

You should have been keeping a record of what you have been doing since Activity 10, either on these student worksheets or in the EV3 Software Content Editor.

Your record will probably include text notes, video, photographs and screenshots of your programming solutions.

You are to prepare a short presentation that explains your journey through the engineering process and the decisions that you made as a group that led you to the design for your final wheeled robot and program.

The presentation should communicate how you have met the design brief and which classification of wheeled robot you have built and programmed.

It should highlight your successes and failures, and how these failures were overcome.

The presentations may be in any format you wish. You may wish to use presentation software, e.g. PowerPoint / Keynote / Prezi, the Content Editor in the EV3 Software or something else of your choosing.