Computer Software in Junior Cycle Engineering

This resource was developed as part of the Spring Webinar 'Classroom Practice and Computer Software in Junior Cycle Engineering' which took place during the 2020/2021 school year. All materials used during this webinar can be viewed in the Technologies section of www.jct.ie within the Elective Workshops section of the Engineering CPD Supports. Click on the link below to access those materials.

Link: Spring Webinar 2021 Classroom Practice and Computer Software in Junior Cycle Engineering - YouTube

The design challenges showcased during this webinar focused on how the JCt4 associates developed problem solving and computer software skills in Engineering through engagement with students learning from home. The associates planned the design challenges with their students with their school context in mind. They reflected on what they learned outlining changes they may make for future learner experiences. This resource can be viewed as a stimulus to generate student creativity in Junior Cycle Engineering and may assist a teacher to plan and develop materials suitable for the Engineering classroom.

What is included in this PDF?

Included in this document are design challenges which build on the some of the modules in the 2020/2021 online CPD cluster workshops and align with the design challenges which the JCt4 Engineering associates planned for their students. Throughout the 2020/2021 webinar titled, 'Classroom Practice and Computer Software in Junior Cycle Engineering', the design challenges are discussed through the lens of 'Key Learning', 'Evidence of Learning' and the 'Learner Experience' to emulate the planning process.

2020/2021 CPD Workshop

2020/2021 Webinar

A big thank you to the JCt4 associates for their commitment and involvement in making this resource available to the JCt4 Engineering team.

Note: It is recommended that the 2020/2021 webinar and the 2020/2021 CPD online cluster workshop materials are viewed in conjunction with using this resource to contextualise the resource and develop a better understanding of how these resources were developed.

The following design challenges were presented to first- and second-year Engineering students by JCt4 associates who are practicing Engineering teachers. The challenges aimed to develop the students Engineering and computer software skills. The students participated in a combination of live and pre-recorded remote learning lessons in 2021. A broad range of solutions were evidenced by the students due to the pedagogical approach taken by the JCt4 Engineering associates. The design challenges have been evolved by the JCt4 Engineering team to further develop Engineering teachers' understanding of computer software and stimulate creativity in the Junior Cycle Engineering classroom.

Some of the challenges also reference and build on the challenges teachers engaged with in the 2020/2021 CPD workshops. The contents of the learning log can be found here.

2020/2021
Learning Log

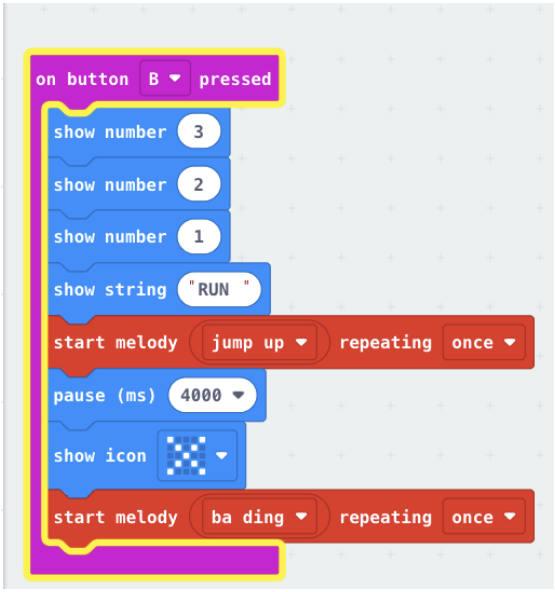
The following Design challenges were completed as part of the classroom experiences referenced in the videos. The intention is to further the learning in each challenge:

Design Challenge 1

Title – Traffic Lights

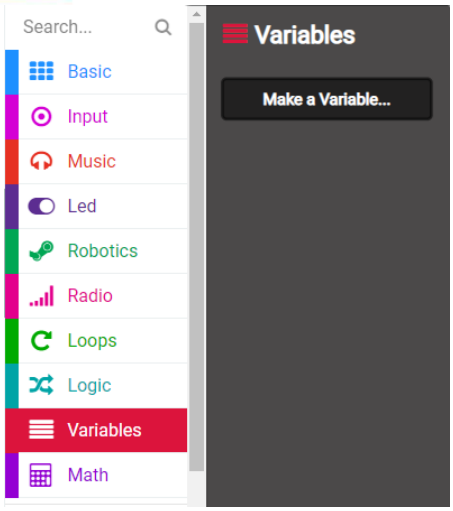
Brief: Show a countdown for a pedestrian crossing to include:

- A countdown
- 'Show String' to cross the road
- A suitable time length to cross the road
- 'Show Icon' to indicate it is unsafe to cross
- Include a melody



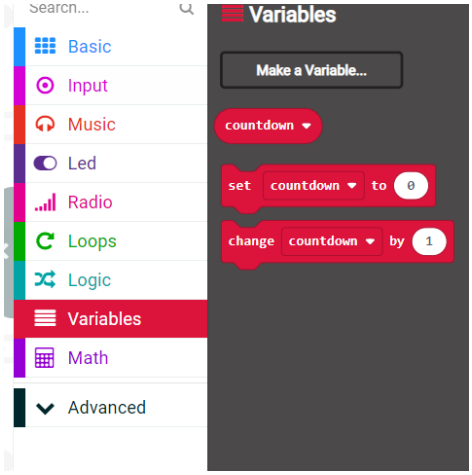
This was engaged with in Module 1 as part of Activity 2 and 3 of the 2020/2021 online CPD cluster workshop.

In order to further enhance student learning, the next code will develop a larger countdown and identify another method of how this can be created.



1:

- When creating the countdown, a variable will need to be created.
- Click on the 'variables' tab and then the 'Make a Variable' button
- Just like in the subject maths, a variable can have any value
- It can be called 'countdown' when asked to name the variable



2:

- Name the variable something that will help make it identifiable, in this case 'countdown'
- When the variable is named, three blocks appear
- 'countdown' allows the variable to be used to show numbers
- 'set countdown to ___' allows the variable to be set a given value.
- 'Change countdown by __' allows the variable to be changed which can be useful for a countdown using a loop.

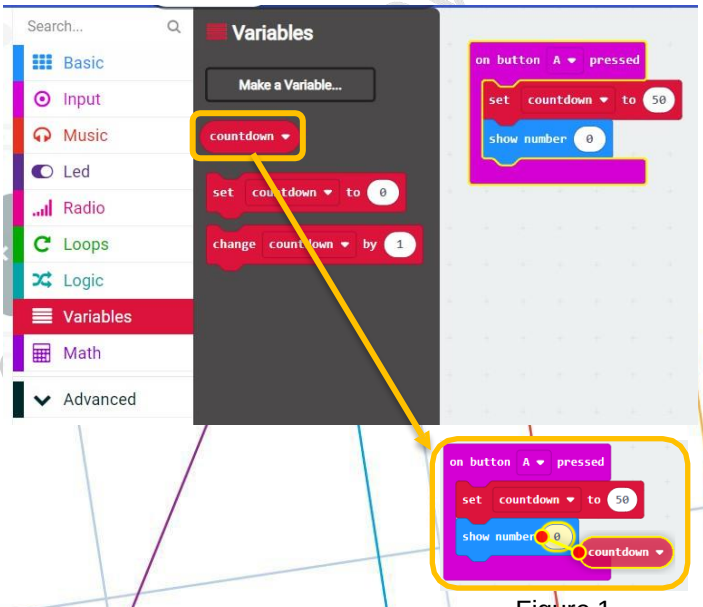
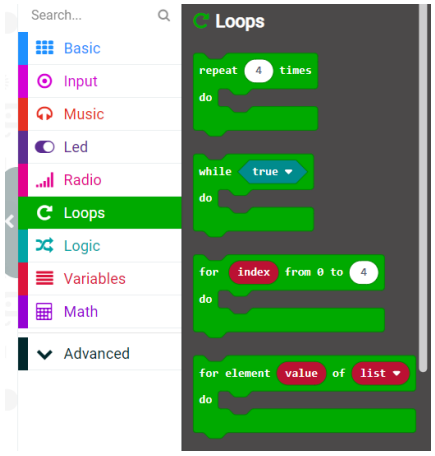


Figure 1

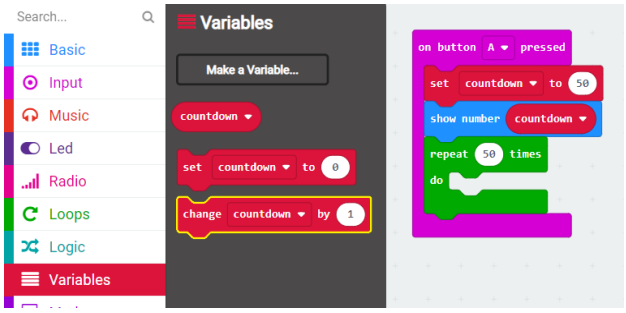
3:

- Using 'inputs' add a button pressed command
- 'set countdown to __' can be used to start the countdown at a higher number, in this case 50
- Using 'basic' we'll add 'show number'
- 'countdown' allows for the use of the variable to show the countdown so add it to 'show number' as shown opposite (Figure 1)



4:

- To make the countdown decrease, use a repetitive 'loop' where the variable can 'change countdown by_'
- The loop can be repeated for the duration of the countdown, in this case, 50

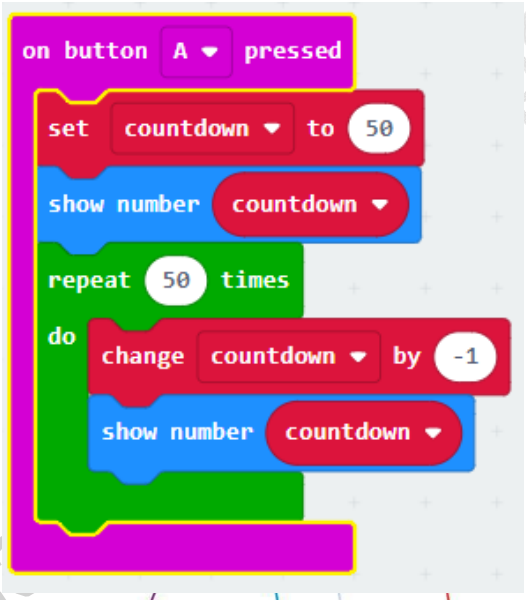


5:

- Using the 'change countdown by _' and adding -1 into the number option the countdown can be decreased by a value of 1, 50 times

*If that number is positive, it will add one to the countdown each time.

- To show the countdown decreasing each time, add 'show number', and add 'countdown' as shown in step 3



6:

To explain again what has happened:

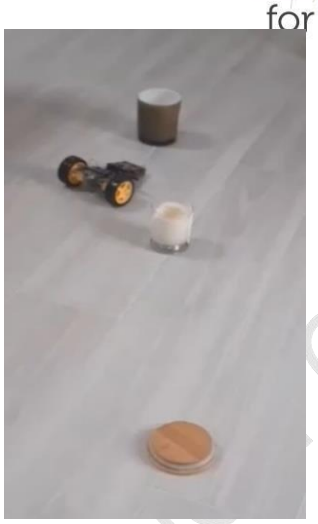
- The countdown was set to a value and then shown on the screen
- Using the loop, the countdown was changed by -1, and then showed the new number on screen 50 times until it eventually reaches 0

*If the countdown is required to stop at 1 then repeat the loop 1 less time than the initial number; in this example that would be 49

**If the requirement is to decrease the countdown faster, it is possible to make the countdown change -2 and repeat it 25 times

How could you use this information in class and in future experiences with your students?

Design Challenge 2



Title – Car Obstacle Course

Brief: Design and build an obstacle course and a car to negotiate the obstacle course.

When engaging with the design challenge, the following could be considered:

1. Obstacle course design and layout.
2. Motor control to turn and make it go forward.
3. A suitable time between movements.
4. Indications as to where it is turning.
5. What coding blocks do you need to make it move.
6. Are the controls automatic or is the vehicle going to be remote controlled?

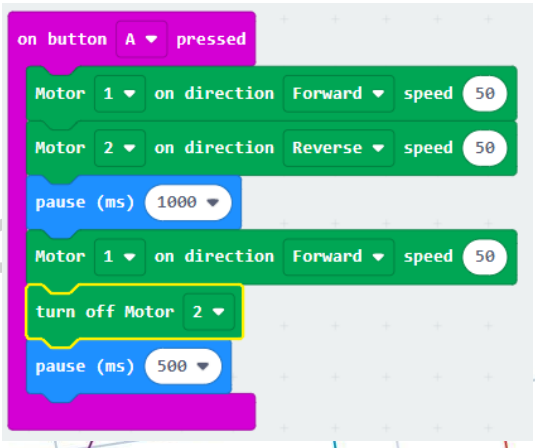
Some other questions which could be considered:

1. When looking at the ways in which to make the vehicle move, should servos or DC motors be used?
2. Is an 'all-in-one robotics' board required to design a solution?
3. Is an 'Extension' on 'Makecode' required?
4. How will the motors be fixed to the car?
5. How will the voltage of the battery impact on the timings in the program?

The following looks at an automatic run using the blocks mentioned in Module 3 (DC Motors) from the Engineering 2020/2021 CPD workshop.

Configuring motors

Understanding the relationship between how motors are configured and how a program works is necessary prior to solving any coding problem involving DC Motors. Two approaches may be used to understand this relationship.

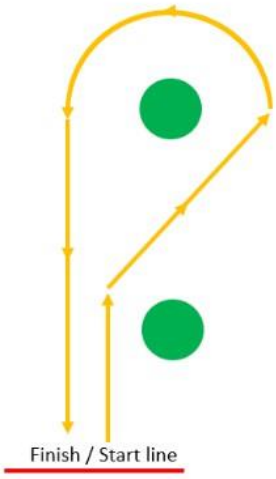


Program 1

For the purposes of this activity, Motor 1 is the left-hand side motor and Motor 2 is the right-hand side motor of the car.

- Both motors may be facing different directions so it may be necessary to program one motor to run forward and the second to run in reverse to make them go the same direction
- The pause in the code says that it will be 1 second until the next set of commands happen. It does not run the motor for 1 second and then stop. In this code the vehicle turns right

A possible obstacle course layout



Program 2

An alternative approach may be taken to ensure a vehicle is propelled forward while **both motors rotate forward**.

- This can be done by testing a program such as the one to the left
- For the purposes of this activity, Motor 1 is the left-hand side motor and Motor 2 is the right-hand side motor of the car
- It may be necessary to change the polarity of the wires by swapping their configured position on a board
- For the purposes of designing a solution in this activity, this is the process that is used to configure the motors

- Further considerations include a possible course design for a car propelled by two motors, the coding blocks and approaches to controlling the propulsion of the vehicle with the motors

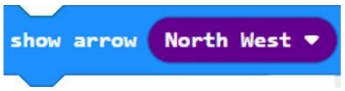
- It should be noted, that when interrogating a possible solution such as the code opposite, the progress of the code is from top to bottom

- The code also includes a range of motor movements which include, ½ a second, 1 second, 2 seconds and 3 seconds. All are timed by the 'pause' blocks

- A 9v battery was used in this program

- The first turn of the vehicle is done using a 'turn off motor _' block.
- The large turn at the top of the obstacle course could be done using both motors rotating at different speeds.

Are there other possible approaches to changing the direction of the car?



The block opposite, which is located in the 'Basic' tab, can show an arrow on the LED matrix to indicate the direction which the vehicle could turn.

It can be added into the code at any point. Is there another way?

How could you use this information in class and in future experiences with your students?

Design Challenge 3

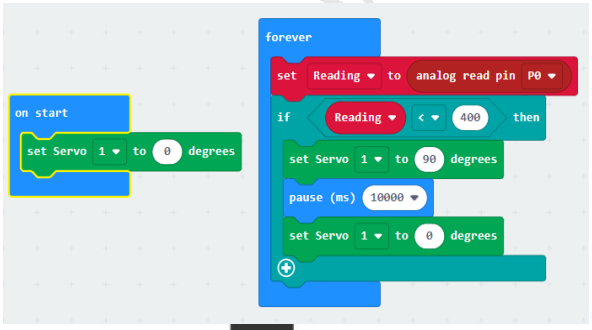
Title – Car Park Barrier

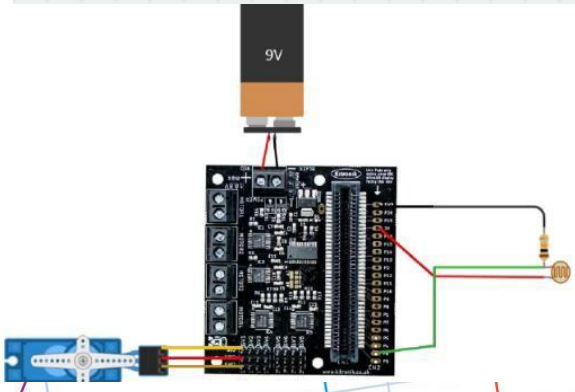
Brief: Design a code to open and close a Car Park Barrier with a Servo. The Barrier should close when a sensor is activated and should contain a sound to indicate the barrier is closing.

This activity was engaged with on the 2020/2021 online CPD Cluster Workshop

Module 2 Activity 1 showed us how to simulate the movement.
Students could also engage with the challenge using module 5 and a potentiometer.

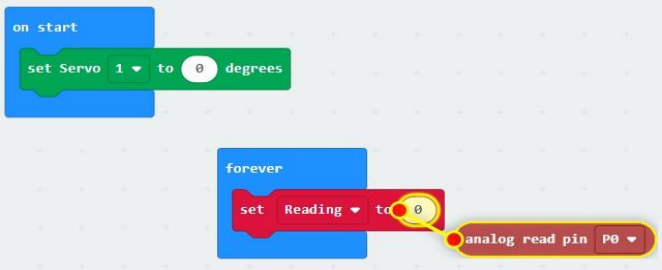
Is there another way this challenge could be engaged with?
Another way we could engage in this design challenge is using an LDR sensor. The codes from Module 6 in the Learning Log (p.26-31) for the 2020/2021 online CPD cluster workshop can be evolved to activate the barrier as a vehicle drives over an LDR.





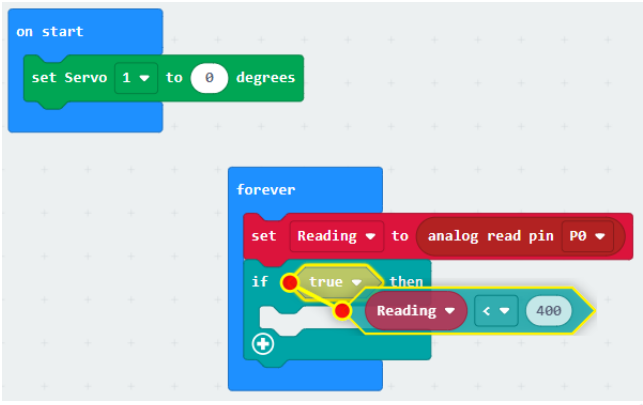
*Before this activity is started, add the 'All-in-One' Robotics board to the tabs in the Makecode menu.

- Revisit Module 6, Activities 1-3 on pages 26-31 in the Learning Log from the 2020/2021 online CPD cluster workshop
- This will explain how to calibrate the LDR and get the reading for when you want the barrier to open.
- The video for calibration can be found here.



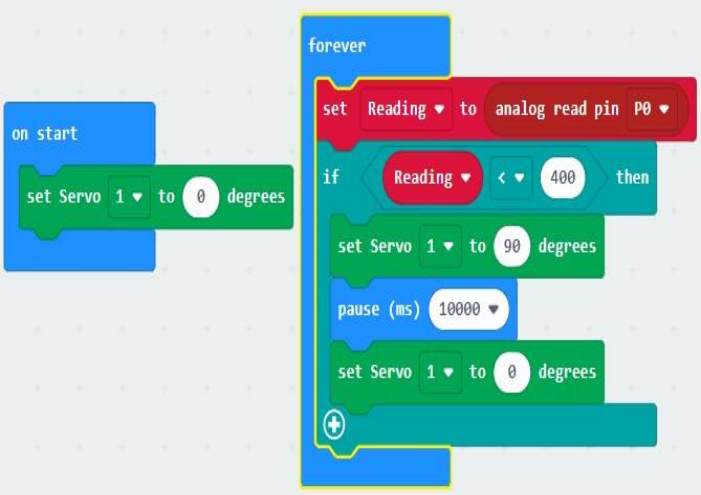
On start setting the servo to 0 is good practice as the servo will move to its original position

Add the 'set reading to _' block from the variables tab to the 'forever' block. It was shown earlier in this document how to create a 'variable' from the 'pins' tab. Add an 'analog read pin p0' into the block as shown.



- Add an if logic gate from the 'logic tab' and a 'Comparison' as shown. Also add in the 'reading' block as shown in the picture.
- Edit the value in the 'Comparison' block to say if 'Reading' is less than whatever value you found when calibrating it earlier.

*When the LDR is covered and the value goes below the value put in this box, the barrier will open.



When the comparison recognises that the LDR has a light reading greater than '400', the barrier should remain closed.

When the comparison recognises that the LDR has a light reading less than 400, the barrier should open with a Servo and stay open for 10 seconds then close again.
To activate this process:

- Add ' set servo ___ to ___ degrees' and a 'pause' block after this
- Set the servo angle to 90 degrees and the pause to 10 seconds
- ' set servo ___ to ___ degrees' set the servo to 0 degrees this time.

This will complete the code.