Otto Cardy Worksheet #1

Guided exercises to learn the how to use block coding to control the movement and sounds of the **Otto Cardy** robot

S 1.0 Lesson plan for the teacher

1. Prepare
   - You can prepare same movement examples (*sketch*) chosen from OttoBlockly **Examples**.
   - Identify different task to assign to each group
2. Engage (5 min.)
   - Get an overview of the movements and sounds provided by the program that you can add.
   - Present the function of the "dept sensor" and the distance it detects
   - Point out the pitch of the notes and the length of the sound.
3. Explore (15 min.)
   - Invite students to work in pairs or groups
   - Invite to elaborate a work strategy using the **sound-movement-dance-distance** functions.
4. Elaborate (20 min.)
   - Invite students to program movements using **sounds** and **gestures**.
   - Ask them to insert musical sequences that their like between the movements of the robot.
5. Evaluate
   - Create an evaluation form for the work of individual students taking a cue from the evaluation provided.

# 8 Educational Worksheet	
Primary Secondary	35-40 min

Teacher Support

Key objectives
The pupils will be able to control the movements and add the sounds

Time: 35-40 min
Working group: the whole class

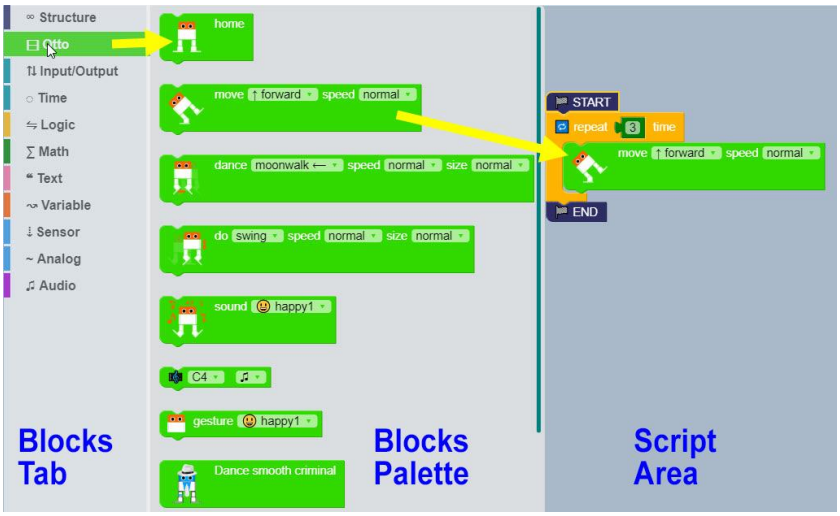
Materials:Otto + OttoBlockly

Key Competences
Problem Solving
- identify the appropriate problem-solving strategies

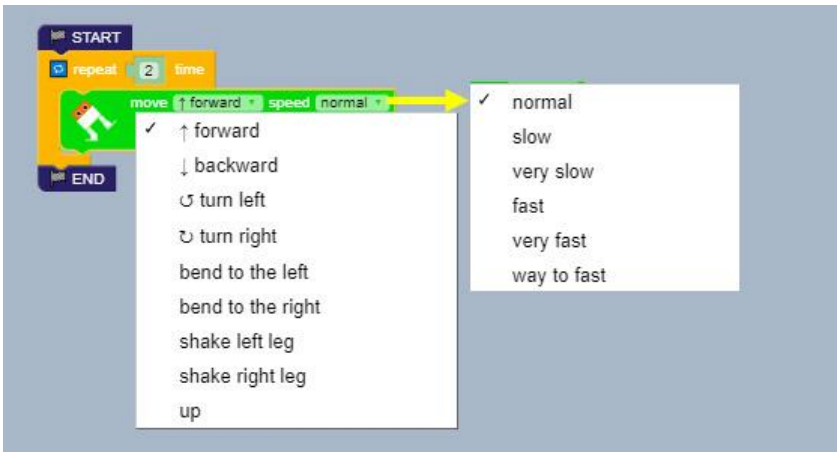
Decision Making
- choose strategies to help you solve a problem
- elaborate simple instructions to control the robot's behavior

S1.1 Let's teach the robot to walk

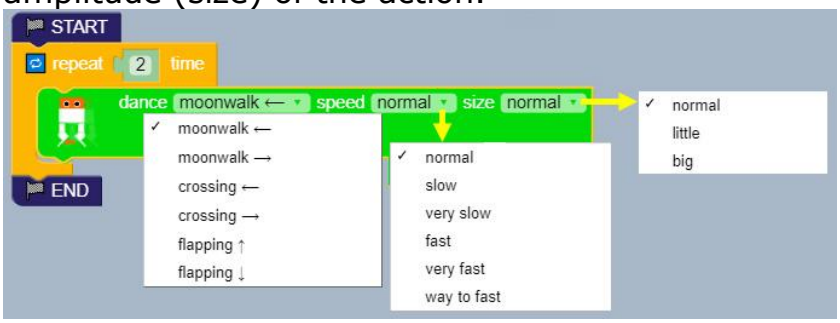
On the *Blocks Tab* menu, in the ***Blocks*** Palette, click ***Otto*** which provides some blocks with already arranged gestures.



Movements associated with **the MOVE** block where each movement can be assigned a speed of execution.



Movements associated with the **DANCE** block**:** in addition to the speed you can also control the amplitude (size) of the action.

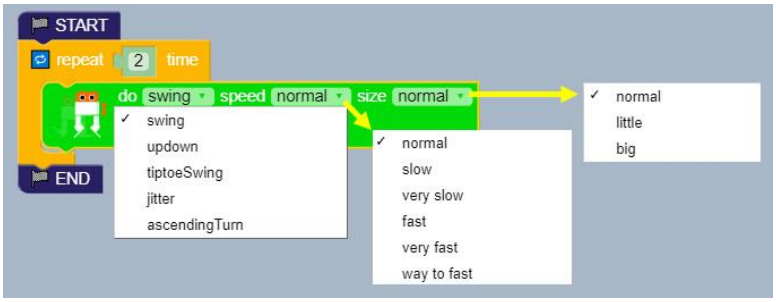


Teacher Support

OttoBlockly coding

- **move**: moves forward, in the back, bends and lifts on the tips;
- **dance**: there are # 3 types of dance such as *moonwalk, crossing* and *flapping;*
- **do**: moves his legs with the following movements of *swing*, *updown*, *tiptoeSwing*, *Jitter* e *ascendingTurn*;
- **gesture** e **sound**: there are two lists of gestures and sounds that can be used and combined.

The **DO** block adds simple gestures such as crawling, getting up on toes and others: each movement can be assigned a speed and amplitude of execution.



The **SOUND** and **GESTURE** blocks allow you to insert some sounds of the buzzer and gestures that can be coordinated with each other.

Movements of the **dance** block: in addition to speed, you can also control the size of the action.

Teacher Support

Coordinated sounds and movements

If you associate **movements** with **sounds**, you can create a challenge and at the same time an opportunity for discussion on what meaning to give to each sound and movement.

For example, what movement can you associate with the "**surprise**" sound, while you can execute only the movement of the **legs** and **feet**?

You could start with the **real life simulation** by associating Otto's sounds with the real leg movements of the students.
It would also be fun to associate the **sounds** with real **leg** and **foot movements**.

What if the robot also had arms? What movements could you associate with each sound?

S 1.2 Insert movements

If you insert a single block in the *Script* area, for example **MOVE**, the program executes the instruction indefinitely.

To interrupt an instruction or a serie of instructions, the **END** instruction must be inserted.
The block instruction generally has a duration of one second.

The **REPEAT** (LOOP) block provides a loop for a specified number of times. At the end, it moves on to the next instruction and if it doesn't find it, it is being repeated indefinitely.
To interrupt the sequence, the **END** block must be inserted. The START block, on the other hand, is not necessary.

You can also insert two loops one inside the other (nested)

Arduino Code

```
void loop() {
  Otto.walk(1,1000,1); //
FORWARD
}
```

```
void loop() {
  Otto.walk(1,1000,1); //
FORWARD
  while(true);
}
```

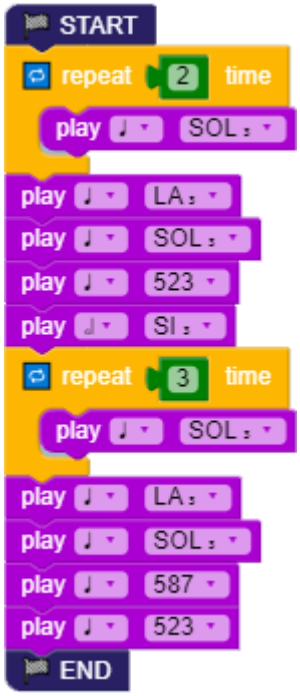
```
void loop() {
  for (int count=0 ; count<3 ;
count++) {
    Otto.walk(1,1000,1); //
FORWARD
  }
 while(true);
}
```

1. Insert two movements with a 1 second pause between them;
2. How many times is the note played in the sketch shown next to this?

S 1.3 Insert sounds

The **Audio Menu** allows you to enter notes through the buzzer, but they don't have great sound quality, or you can load an mp3 file if a player with speaker is installed.

The **PLAY** block allows you to insert a note controlling both its duration and pitch. The pitch of the notes is achieved in two octaves.
In the following example, the notes corresponding to the **Happy Birday** musical motif are displayed in sequence. Note that the **REPEAT** block is used twice to prolong the notes of the chorus.



Arduino Code

```
void loop() {
  tone(13,523,2000);
  delay(2000);
}
```

```
void loop() {
  for (int count=0 ; count<2
; count++) {
tone(3,392,500);delay(500);}
tone(13,440,500);delay(500);
tone(13,392,500);delay(500);
tone(13,523,500);delay(500);
tone(13,493,1000);delay(00);

for (int count=0 ; count<3 ;
count++) {
tone(13,392,500);delay(500);}
tone(13,440,500);delay(500);
tone(13,392,500);delay(500);
tone(13,587,500);delay(500);
tone(13,523,500);delay(500);
  while(true);
}}
```

Exercise 1.3

1. Add the musical notes that complete the tune
2. Compose a new motif from the notes you like

S 1.4 Blocks rules

To test the different movements it may be useful to insert notes at the beginning of each action.Unlike **Scratch** and other block programs, all the blocks presented in the ***Script Area*** are processed even if they are not connected to each other as shown in the figure.

By right-clicking on a block, a selection window opens with some selection functions, such as **Duplicate** or **Disable** block to deactivate the instructions of a block without deleting it.

To duplicate a block it is always possible to use the Windows copy shortcuts **Ctrl + C** and **Ctrl + V** for **copy-paste**.

Arduino Code

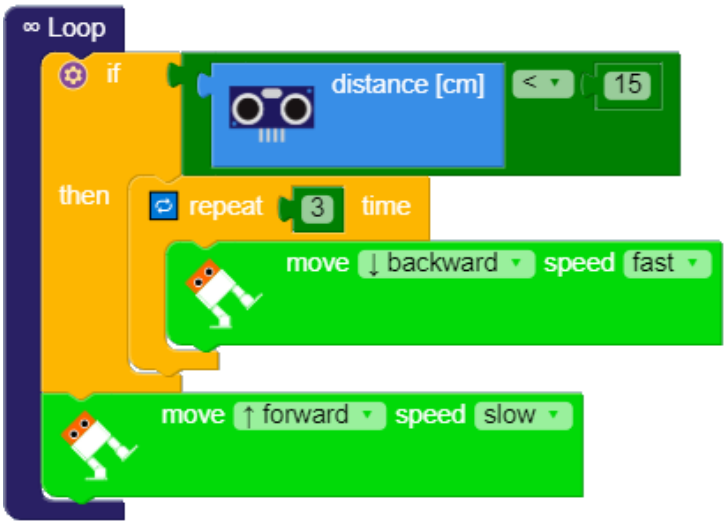
```
void loop() {
// first block
  tone(13,261,250); delay(250);
  tone(13,293,250); delay(250);
  Otto.swing(1, 1000, 25);

// second block
  tone(13,392,250); delay(250);
  Otto.updown(1, 1000, 25);

// third block
  for (int count=0; count<3 ; c
ount++) {
    tone(13,440,250);delay(250)
;
    Otto.jitter(1, 1000, 25);
    }

}
```

Exercise 1.4

1. Insert a comment in a block
2. What does the "External inputs" function change?

S 1.5 Distance sensor

OttoBlockly also provides the distance block to control the distance, in centimeters, through the proximity sensor which detects, using the ultrasound, the space between the sensor and an object.



To check the **distance**, we can use the **IF** block which checks if the distance is less than a given value, as shown in the figure *15 cm*.
If the condition is **true** then it executes the internal instructions otherwise it jumps to the following blocks.

In this case, you can reverse the walk by going backwards for 3 seconds (time).
If this does not happen, that is, if there are no obstacles, Otto continues his walk.

The sketch code is provided alongside with two different notes that warn you when it detects the obstacle and then when the back mark ends.

Arduino Code

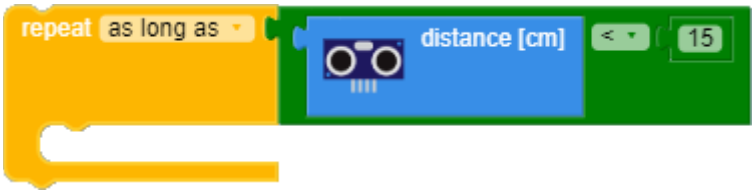
```
void loop()
  {if (Otto.getDistance() < 15)
     {for (int count=0 ;
count<3 ; count++)
     {Otto.walk(1,750,-1); //
BACKWARD }
        }
    Otto.walk(1,2000,1); //
FORWARD
  }

void loop() {
  if (Otto.getDistance() < 15)
     {tone(13,261,1000);
delay(1000);
       for (int count=0 ;
count<3 ; count++)
 {Otto.walk(1,750,-1);//
BACKWARD
}
       tone(13,349,1000);
delay(1000);}
      Otto.walk(1,2000,1); //
FORWARD
  }
```

Exercises 1.5

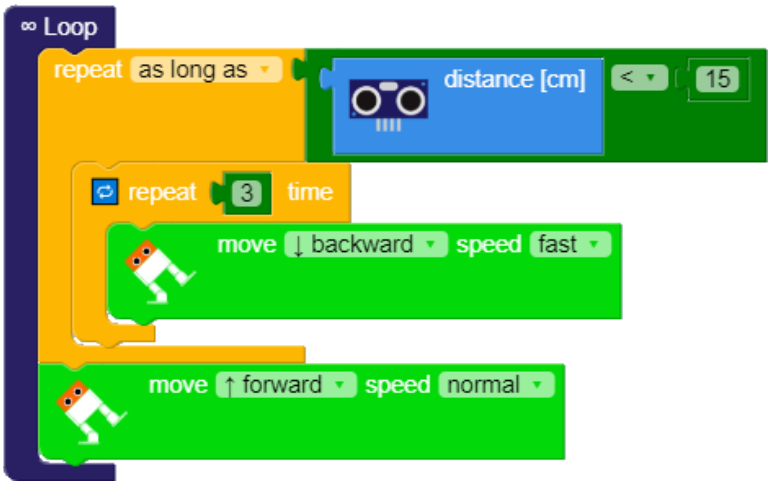
1. Insert an audible warning (note) when Otto detects an obstacle.
2. Enter another notice (different note) when the block ends

S 1.6 Distance condiction

To repeat a series of blocks depending on a condition, such as until the distance is less than 15 cm, you can use the repeat **as long as** block instead of the previous **IF** block. It corresponds to the common programming function **WHILE**.

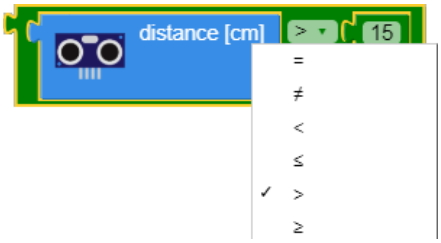


In our case, instead of the **IF** block, we can use the **REPEAT** block to make Otto move backwards until the distance within the object increases.



The structure is similar to the previous one where only the **REPEAT** block has been changed. In the code alongside, note that the **IF** statement has been replaced with **WHILE**.

It is also possible to change the **distance conditions** to change the movement when the condition occurs.



Arduino Code

```
while (Otto.getDistance() < 15
) {
  Otto.walk(1,1000,1); // FORW
ARD
 }
```

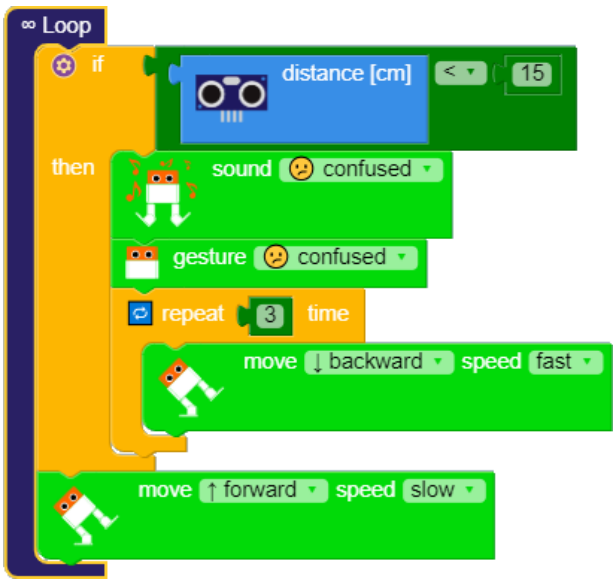
```
void loop() {
  while (Otto.getDistance() <
15) {
  for (int count=0 ; count<3 ;
count++) {
  Otto.walk(1,750,-1); // BACK
WARD
    }
  }
 Otto.walk(1,1000,1); // FORWA
RD
}
```

Exercises 1.6

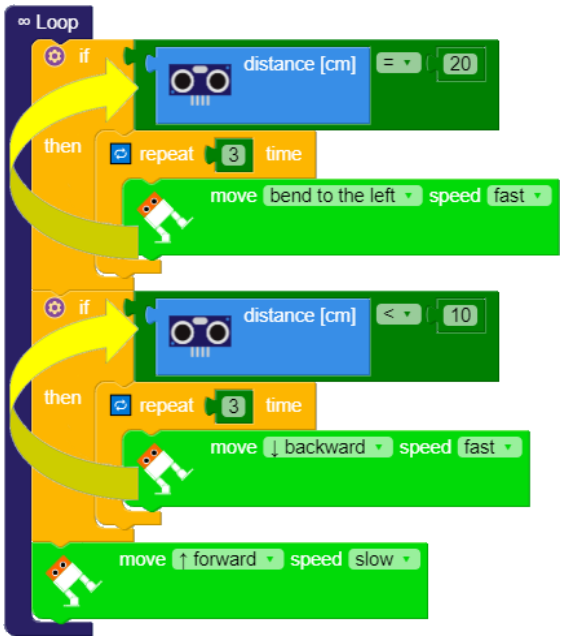
1. Change Otto's direction and change the distance
2. If we modify the distance condition with **> 15** what does it change?

S 1.7 *Sound* and *gesture* blocks

To emphasize the encounter with an obstacle, more articulated sounds can be added, such as a sound that is among those available sounds and a gesture of surprise due to the inconvenience of encounter with the the obstacle.



If we want to insert two distance conditions, for example at 20 cm it performs an action while at 10 cm another, then we must insert two IF blocks as in the figure:



Arduino Code

```
void loop()
{if (Otto.getDistance() < 15)
   {Otto.sing(S_confused);
Otto.playGesture(OttoConfused);
    for (int count=0 ; count<3 ;
count++)
   {Otto.walk(1,750,-1); //
BACKWARD}}
    Otto.walk(1,2000,1); //
FORWARD
}
```

```
void loop()
{if (Otto.getDistance() == 20)
   {for (int count=0 ; count<3 ;
count++)
   {Otto.bend(1,750,1);  }}
  if (Otto.getDistance() < 10)
     {for (int count=0 ; count<3
; count++)
    {Otto.walk(1,750,-1); //
BACKWARD }}
  Otto.walk(1,2000,1); //
FORWARD
}
```

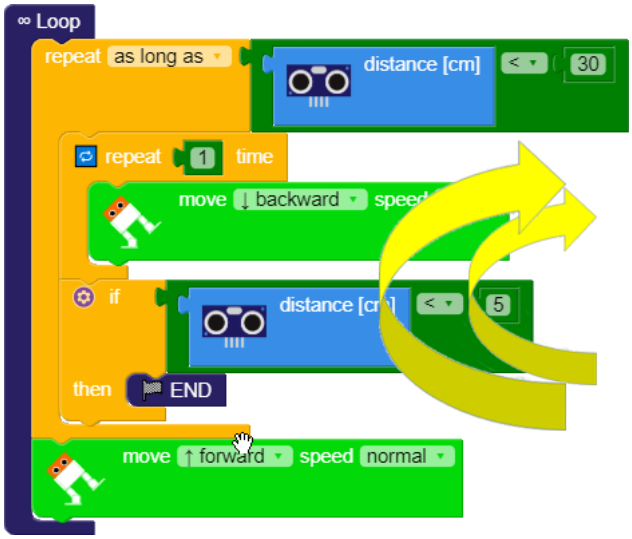
Exercices 1.7

1. Why was the condition = 20 and not <20 inserted in the second figure?
2. Propose another solution to have two distance conditions: at **10** and **20** cm.

S 1.8 Conditions of nested cycles

It is possible to use two or more cycles, one inside the other (*nested cycles*) to repeat a series of integrated operations.

In our case we have inserted an internal **END** block: if the distance is less than 5 cm. the robot stops.




```
void loop()
{ while
  (Otto.getDistance() < 30)
     {for (int count=0 ;
count<3 ; count++)
     {Otto.walk(1,750,-1); //
BACKWARD }
  if (Otto.getDistance() < 5)
      {while(true);}}
  Otto.walk(1,1000,1); //
FORWARD
}
```

The example is not very elegant from a programming point of view but it is an example of how to stop the execution program.

The conditions included are the following:

- **IF** the distance is <(*less*) than 30 cm. then back off;
- **IF** the distance is <(*less*) than 5 cm. then it stops (**STOP**).

Note that the first condition is that of the greatest distance that is checked first and after, it only checks if the second condition is also true, distance <5.

Exercises 1.8

1. Is it possible to insert another repetition as long as block, as in the first condition, is in the place of the IF block?
2. Find another solution for the two nested conditions.

S 2.0 Project Applications

Otto Cardy's project lends itself to many didactic applications not only in the subjects of coding, but also for math, physics, art or electronics.

Here I provide just a few ideas that teachers will surely know how to expand:

1. Add to the robot the ability to **detect** distances through a series of different sounds depending on the distance detected;
2. Associate new **movements** and **sounds** created by the students;
3. Create **variables** that control sounds and movements;
4. Add a **led** that lights up when the robot moves forward;
5. To create clothing for a robot use fabrics or paper according to **stories** that can invent;
6. Work in groups, write **stories**, fantastic or historical, where the characters are robots;
7. Add the **arms** to the robot with two other servomotors;
8. Add an 8x8 **display** to view messages and texts;
9. Add a **Bluetooth control** to be able to control the movements with the mobile phone app;
10. Develop applications that make the robot perform **complex movements** by interacting with the various sensors.

Project Development

The construction of the Otto Cardy robot allows children to acquire the basic knowledge of the construction of an object that works with Arduino.

There are many other ideas and information available: here we provided some links to websites where you can find technical and educational information and ideas for the development of various applications.

Otto Project sites:
-www.ottodiy.com/
-ottoschool.com/scratch/
-wikifactory.com/+OttoDIY/otto-diy

On the site you can find information on how to purchase the components that can be added to the basic robot.

Site of Otto Cardy Project:
www.cad-tutor.com/otto_cardy