1 Robo-Camp 2019: Lesson Plan and Teaching Materials

1.1 Theme project – Automation (12 x 45 min.)

Briefly about content: The project deals with the automation of a coffee bean center, for which two types of robots are to be developed. A robot that sorts the beans according to quality (color sorting), as well as a robot that transports them around the warehouse (line tracking). The students choose which of the two robots they want to work with, after which it is intended that they work relatively independently with the project in small groups (it is recommended that the students are two students per group).

Learning outcomes: Students learn to apply what they have learned from previous lessons in a practical context, to solve concrete problems inspired by the industrial workplaces of reality. At the same time, this gives them a deeper insight into and understanding of the development of complex automated systems, as well as their limitations.

Materials: In the developed material, LEGO has been used as a platform for the developed robots. In order to facilitate the integration of electronics in the LEGO platform, a number of 3D models that are compatible with it have therefore been developed at the same time. Manuals for this, as well as 3D models are, as previously described material, made available on the website. The same applies to a complete overview of the components used in the project, solution proposals, as well as proposals for further development of the robots etc. It is important to note that the robots can be constructed in many different ways and from many different materials, the developed LEGO manuals are therefore only an offer for a solution to this.

1.1.1 Introduction to the Projekt:

Introduce the students to the following project, which is based on a coffee bean center, where the beans are first sorted by quality, after which they are distributed around the associated warehouse. At the center, the beans have previously been hand-sorted, after which they have been distributed by hand in the warehouse. This involves hard physical work, while at the same time taking a long time. The center's desire is therefore for these two processes to be automated, which is why two types of robots must be built and programmed, each of which can handle one of these functions.

> **Coffee bean sorting (color sorting):** The beans are sorted according to quality, which is determined by their color, in connection with the project, the beans consist of 3D printed bricks (red, green, blue and yellow). The robot must therefore be able to receive a bean, register which color category it belongs to, rotate four associated boxes around so that the corresponding box faces the conveyor belt, and then deliver the bean therein and rotate the boxes back to the starting point again.
>
> **Transport (line tracking):** The sorted beans are packed in boxes, which in connection with the project consist of 3D printed boxes (red, green, blue and yellow), the color indicates which of four corresponding areas in the warehouse they are to be delivered to. The warehouse has been equipped with black lines in the floor, which the robots must navigate by. In addition, to avoid

accidents, the robots must stop if there is an object in front of them blocking the road, this could be human employees or other robots.

The control panel is already equipped with an area where the sorting takes place and in addition, the employee has just completed the work of drawing the lines the robots must navigate according to, see Figure 23.

Once the students have chosen which of the two robots they want to work with, they can follow the corresponding procedure from the following two sections.

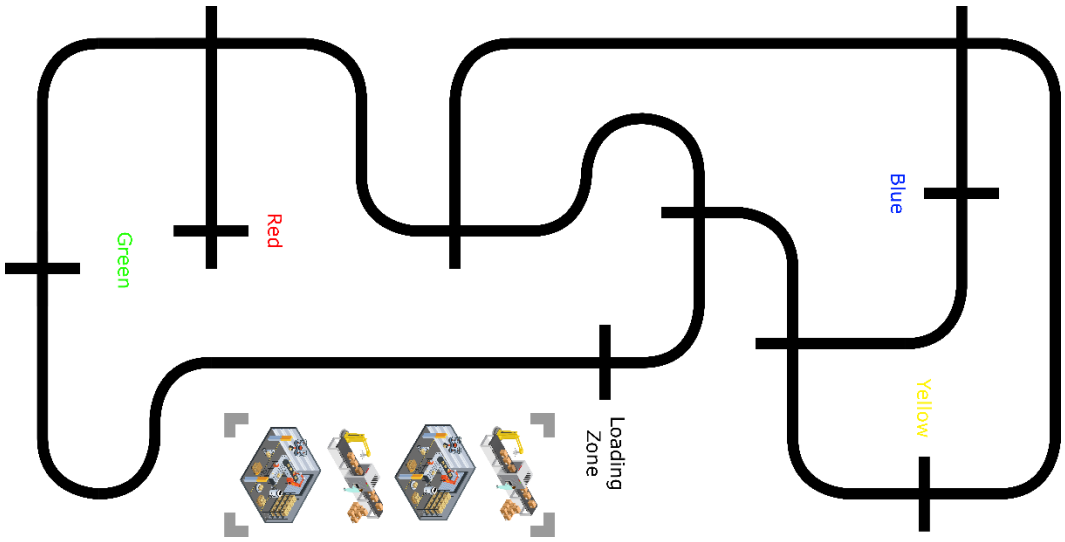

Fig. 23: Overview of the warehouse sorting/loading zone and the transportation routes.

In the videos below, you can see examples from the real world, of robots designed for automatic sorting of coffee beans, as well as transportation.

Coffebean sorting: https://www.youtube.com/watch?v=i0nmIl-bNLI

Transportation: https://www.youtube.com/watch?v=WIlS3vNSuQ4

1.1.2 Coffee bean sorting:

The robot consists of two main parts, which are combined into one. The first part is the conveyor belt, on which the coffee beans are first placed, then they are transported under a color sensor after which their color is registered (red, green, blue or yellow), then the task of the conveyor belt is to transport them the last piece to and down in a corresponding box. The second part is the mechanism that rotates the corresponding boxes, so that when the beans reach the end of the conveyor belt, they fall into the correct box. See Figure 24 below.

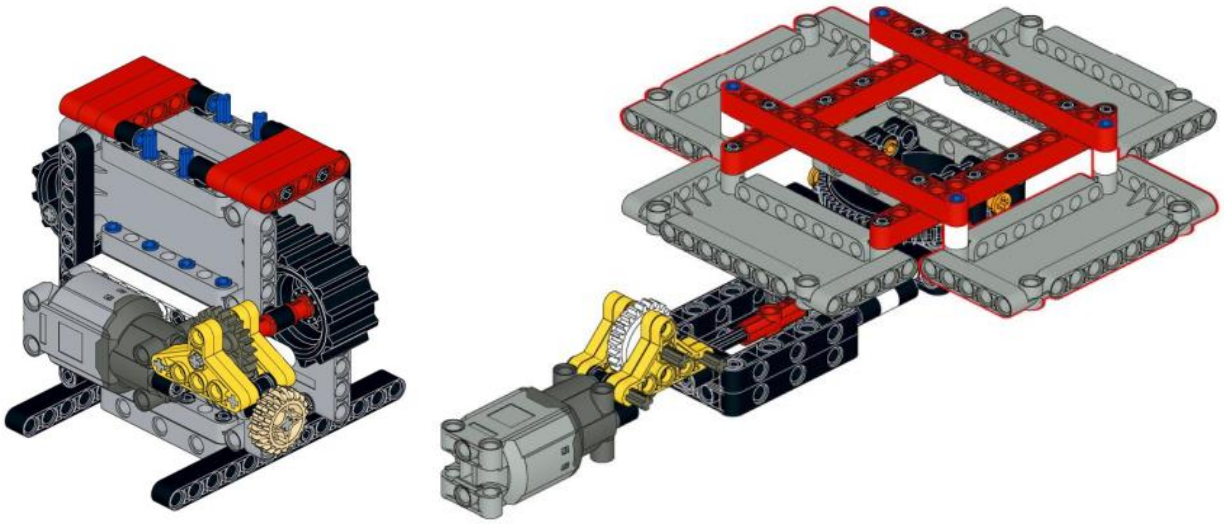

Fig. 24: The LEGO-models for the belt and the box rotator.

1.1.2.1 The initial steps:

1. Construct the robot:

The first thing to do is to design the robot itself so that there is something to work from.

Task:

- Follow the LEGO manual "Bean Sorter - Belt" and build the conveyor belt.
- Follow the LEGO manual "Bean Sorter - Rotator" and build the rotation mechanism for the boxes.
- Follow the LEGO manual "Bean Sorter - Combine" and build the two parts together.
- Now assign the boxes each their individual color, possibly. with a felt-tip pen and choose which, by default, should face the conveyor belt.

Materials:

- 2 pcs. LEGO Power Functions Motor (Large).

2. Integrate the Arduino platforms:

Now the Arduino must be added, which is the microcontroller that will be responsible for controlling our robot, as well as the associated motorshield, which is an extension to the Arduino that is used to control the motors, as well as a breadboard.

Task:

- Follow the manual "Bean Sorter - Integrate the Arduino platform" and add the Arduino, motor shield, breadboard and battery pack.

Materials:

- Arduino Uno.
- Motor shield. R3
- Breadboard (half size).
- Battery pack + Cord.
- Arduino module (3D printed).

- Breadboard module (3D printed).
- Battery pack module (3D printed).
- 2 pcs. he / he wires.

3. Test the engines:
It is now time to connect the motors and at the same time carry out a number of smaller tests to make sure that they work as intended.

Task:
- Connect the motors to the motor shield.
- Make the conveyor belt travel forward for 2 seconds, at full speed.
- Make the conveyor belt run backwards for 2 seconds, at half speed.
- Make the boxes rotate clockwise for 2 seconds, at full speed.
- Make the boxes rotate counterclockwise for 2 seconds, at half speed.

Cheat Sheets:
- Effectors - DC Motor.
- Code - pinMode.
- Code - digitalWrite.
- Code - analogWrite.
- Code - delay.

1.1.2.2 Transportation belt:
It is recommended to create a new program file for this part, but please save it earlier from the Initial Steps.

1. Use functions to control the conveyor belt:
Features are a great way to encapsulate a piece of code so that it can be used many times without having to type the code every single time. At the same time, it makes the code easier for people to read.

Task:
- Create a "startBelt" function that starts the conveyor belt.
- Create a “stopBelt” function that stops the conveyor belt.
- Create a "deliverBean" function that delivers the prayer (starts the conveyor belt and lets it run for 5 seconds, after which it is stopped again).
- Test that all three functions work.

Cheat Sheets:
- Code - Functions.

2. Integrate the color sensor:
Before the conveyor belt can be used, the color sensor must first be integrated in it, and it must be tested whether it works as intended.

Task:
- Follow the manual "Bean Sorter - Integrate Color-Sensor" and integrate the color sensor.

- Place a “bean” on the conveyor belt under the color sensor and print to the screen what color it is. Do this for all the colors (red, green, blue and yellow).

Materials:

- Color sensor (TCS3200).
- Color sensor module (3D printed).
- 7 pcs. he / she wires.

Cheat Sheets:

- Sensors - Color-Sensor (TCS3200).
- Code - Variables.
- Code - Serial Print.

3. Reacts when reading colors:

The conveyor belt can now be checked, as well as use the color sensor to detect colors, but so far it does the same all the time. It is therefore time to create a structure for the program that uses conditions to allow the program to perform different actions, depending on what color the color sensor detects (red, green, blue or yellow).

Task:

- Start the conveyor belt.
- If the color sensor reads "red":
- "red" to the screen.
- Stop the conveyor belt and deliver the "prayer".
- Test that this works when it does, then make an “if else” for each of the other colors (green, blue and yellow) that contains the same functionality as that for the red color.

Cheat Sheets:

- Code - Conditionals.

1.1.2.3 The boxes:

It is recommended to create a new program file for this part, but save it earlier from the Conveyor Belt.

1. Use functions to rotate the boxes:

Use, as with the conveyor belt, once again functions to encapsulate the engine control code so that it can be reused again and again.

Task:

- Create a “startRotation” function that starts the boxes rotating counterclockwise.
- Create a "stopRotation" function that stops the rotation of the boxes.
- Create a function "rotatePosition" that takes an int as a parameter (call the parameter "position"), except to print the parameter to the screen, this function is left blank.
- Test that the functions work.

Cheat Sheets:

- Code - Functions (with parameters).

2. Integrate the switch:

Until now, the boxes could only rotate back and forth, based on time, but because the time it takes to rotate a lap varies depending on how much power is left on the batteries, how much dirt has entered the gearing etc. is this not a good solution. Therefore, integrate the switch in the construction so that it can be used to register each time the boxes have rotated one space.

Task:
- Follow the manual "Bean Sorter - Integrate Switch" and integrate the switch.
- Test that the switch works by printing its mode to the screen.

Materials:
- Switch.
- Switch module (3D printed).
- 1KΩ resistor.
- 3 pieces. he / he wires.

Cheat Sheets:
- Sensors - Switch.

3. Control the number of rotations:
Because the switch is automatically pressed each time a box passes it, it can use this to control how many positions the boxes have moved, by counting the number of times the switch is pressed. This way you can always make sure that the boxes move exactly the number of positions desired. For this purpose, the function "rotatePosition" must be completed so that it can be called with the number of rotations desired.

Task:
- Inside the function "rotatePosition":
- Start the rotation of the boxes.
- A new counter variable called "i" and give it the value 0.
- Do a while loop with the condition (in <position).
- If the button is pressed:
- Increment "in" by 1.
- Wait approx. 200ms (depending on the rotation speed).
- Stop rotationen af kasserne.
- Test at funktionen virker, ved at kalde den med forskellige parametre (1, 2, 3 osv).
- Tæl nu hvor mange gange hver af kasserne skal rotere en position, førend de står foran transportbåndet, samt hvor mange gange de skal rotere førend de er tilbage ved deres udgangspunkt igen. Skriv disse tal ned på et stykke papir.

Combine the transportation belt and the box rotator:

It is recommended to create a new program file for this part, but please save it earlier from the Initial Steps.

1. Combine it all:
It is now time to combine the conveyor belt with the boxes so that the robot is fully automated. Fortunately, almost all the work required for this has already been done, through our use of features.

Assignment:

- Start the transportation belt.
- If the color sensor reads "red":
    - Print "red" to the screen.
    - Stop the conveyor belt.Roter kasserne sådan at den røde kasse står ud for transportbåndet.
    - Deliver the bean.
    - Rotate the boxes so that the red box is next to the conveyor belt.
    - Rotate the boxes so that the red box is at its starting point again.
    - • Test that this works and then expand the program to include the other colors (green, blue and yellow).

1.1.2.4 Ideas to expand upon the sorter:

Feel free to come up with some ideas for potential expansions, for example, to make different LEDs light up to indicate what actions the robot is currently doing. is about to undertake. It can also be to build a holder that, with the help of a servo motor, can even place new beans on the conveyor belt, one at a time.

1.1.3 Transportation:

The robot rests on two front wheels, which act as its driving force, as well as a ball wheel. See Figure 25 below.

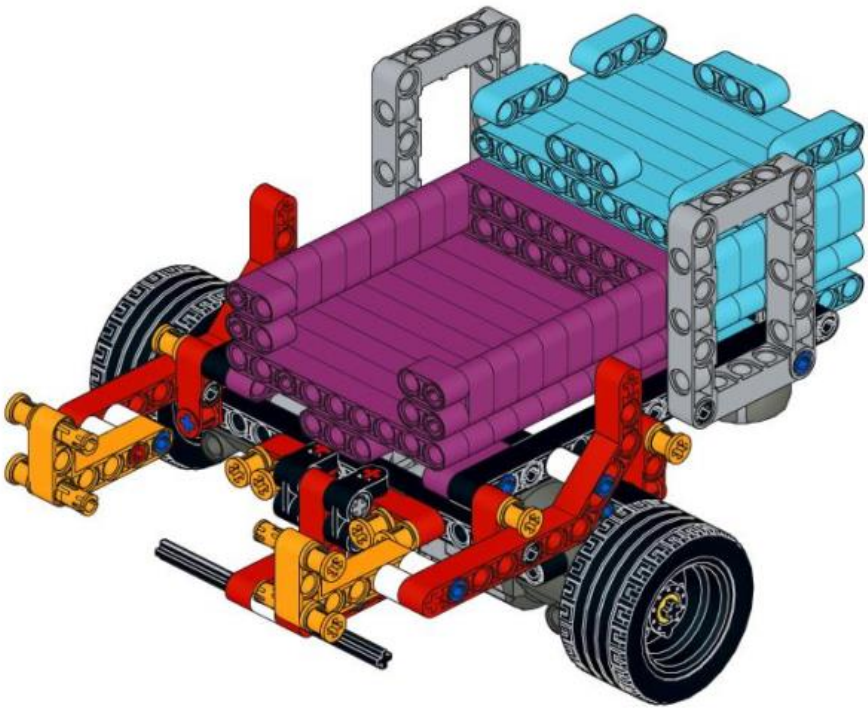

Fig. 25: The LEGO-model for the transporter.

1.1.3.1 The initial steps:

1. Design the robot:

The first thing to do is to construct the robot itself so that there is something to work from.

Task:

- Follow the LEGO manual "Transporter - Transporter" and build the transporter.

Materials:

- LEGO + 2pcs. LEGO Power Functions Motor (L).

2. Integrate the Arduino platforms:

Now the Arduino must be added, which is the microcontroller that will be responsible for controlling our robot, as well as the associated motorshield, which is an extension to the Arduino that is used to control the motors, as well as a breadboard.

Task:

- Follow the manual "Transporter - Integrate the Arduino platform" and add the Arduino, motor shield, breadboard and battery pack.

Materials:

- Arduino Uno.
- Motor shield. R3
- Breadboard (half size).
- Battery pack + Cord.
- Arduino module (3D printed).
- Breadboard module (3D printed).
- Battery pack module (3D printed).
- 2 pcs. he / he wires.

3. Test the engines:

It is now time to connect the motors and at the same time carry out a series of smaller tests to make sure that they work as intended.

Task:

- Connect the motors to the motor shield.
- Make the conveyor move forward for 2 seconds, at full speed.
- Have the conveyor reverse for 2 seconds, at half speed.
- Turn the conveyor to the left for 2 seconds, at full speed.
- Turn the conveyor to the right for 2 seconds, at half speed.

Cheat Sheets:

- Effectors – DC Motor.
- Code – pinMode.
- Code – digitalWrite.
- Code – analogWrite.
- Code – delay.

1.1.3.2 Navigate the lines:

It is recommended to create a new program file for this part, but please save it earlier from the Initial Steps.

Use functions to control the conveyor:

Features are a great way to encapsulate a piece of code so that it can be used many times without having to type the code every single time. At the same time, it makes the code easier for people to read.

Task:

- Create a “driveForward” function that drives the robot forward.
- Create a function "crossLine", which runs the robot approx. 5cm forward and stop it again.
- Create a "turnLeft" function that turns the robot a quarter turn to the left.
- Create a "turnRight" function that turns the robot a quarter turn to the right.
- Create a "uTurn" function, which rotates the robot half a turn, the direction does not matter.
- Create a “stopRobot” function that stops the robot for 1 minute.
- Test that all six functions work.

Cheat Sheets:

- Code – Functions.

2. Integrate the IR-sensors:

Before the carrier can follow the lines in the floor, its two IR sensors must first be integrated and it must be tested whether they work as intended.

Task:

- Follow the manual "Transporter - Integrate IR-Sensors" and integrate the IR sensors.
- First read the value of the left IR sensor and print it to the screen, then write down the values for white and black. Then repeat this for the right IR sensor.

Materials:

- IR sensor (QRE1113, Analog).
- IR sensor module (3D printed).
- 6 pieces. he / he wires.

Cheat Sheets:

- Sensors – IR-sensor (QRE1113, Analog).
- Code – Variables.
- Code – analogRead.
- Code – Serial Print.

3. Follow the line:

It is now time to use the IR sensors to control the robot so that it can follow a line. For this, for each IR sensor, the value that lies directly between the value for white and the value for black must be calculated. This value is called the threshold. To follow a line, the motors must run if their respective IR sensor looks white, otherwise they must stop. That way, the robot will automatically correct if it is about to skew and thereby cross in over the line, in the same way it will also automatically stop when it comes to an intersection because both sensors now look black. Feel free to start by simulating this, by controlling the robot by hand.

Task:

- Make the left motor run if the left IR sensor reads less than the threshold, otherwise it must stop.
- Make the right motor run if the right IR sensor reads less than the threshold, otherwise it must stop.
- Test that the robot can now follow the line, and that it stops when it reaches an intersection.

Cheat Sheets:

- Code – Conditionals.

4. Follow the line as a function:

So far, the robot stops automatically when it reaches an intersection, but it also starts again by itself if you manually push it past the intersection, or put it on a new line. Therefore, if you want to use the rotate commands stored in the functions "turnLeft" and "turnRight", to navigate in a cross, this creates a problem. Feel free to call these from the code - now the robot just stands and turns around and does not follow the line at all. This is because the lines of code that cause it to follow the line are executed in a split second, after which the turn command is called again. Therefore, an independent function must be created that causes the robot to follow the line until it reaches an intersection, after which the function ends.

Task:

- **Create a function "followLine".**
- **Use a while loop with the condition (true).**
- **Make the left motor run if the left IR sensor reads less than the threshold, otherwise it must stop.**
- **Make the right motor run if the right IR sensor reads less than the threshold, otherwise it must stop.**
- **If both IR sensors read less than the threshold, stop both motors and disconnect while looping.**
- **Test if the function works.**
- **Now test whether it works together with the turning functions, as well as the function for driving straight out at an intersection, for example by calling the functions below and then observing whether the robot is running the expected route. Remember that when a minute has passed, the function "stopRobot" is complete, the sequence also starts again.**
- **followLine.**
- **turnLeft.**
- **followLine.**
- **crossLine.**
- **followLine.**
- **turnRight.**
- **followLine.**
- **uTurn.**
- **stopRobot**.

Cheat Sheets:

- Code – While loop (with break).

1.1.3.3 Navigate the surroundings:

It is recommended to create save a copy of the program, then continue working on the copy.

1. Integrate the distance sensor:

The robot is now able to navigate along the lines on the warehouse floor, but what if there is another robot in front of it, or that an employee has overlooked that it is coming right towards them. To avoid the problems and damage that can occur, a distance sensor can be integrated, which allows the robot to measure the distance to the nearest object. This allows it to be programmed to stop afterwards, should an object be too close to it.

Task:

- Follow the manual "Transporter - Ultrasound Sensor" and integrate the distance sensor.
- Print the distance to the nearest object on the screen.

Materials:

- Distance sensor (HC-RS04).
- Distance sensor module (3D printed).
- 4 pieces. he / he wires.

Cheat Sheets:

- Sensors – Ultrasonic-Sensor (HC-RS04).

2. Stop for objects blocking the robot:

Upgrade function "followLine", so that the robot stops if the distance to an object in front is less than 20cm.

Task:

- **Upgrade the "followLine" function so that:**
- **The first thing that happens in the while loop is to read the distance to the nearest object.**
- **Upgrade the conditions for the engines to run so that the condition now reads:**
- **If the left IR sensor is below the threshold and the distance is more than 20cm, run the left motor, otherwise stop it.**
- **If the right IR sensor is below the threshold and the distance is more than 20cm, run the right motor, otherwise stop it**.

1.1.3.4 Deliver the box with the beans:

It is recommended to create save a copy of the program, then continue working on the copy.

1. Integrate the servo motor:

Employees can now put a box on the robot and program it to follow a route, but it cannot yet hand in the box itself when it has arrived. Therefore integrate a servo motor that can be used to tip the box off the barn, after which the robot can drive back for a new box.

Task:

- Follow the manual "Transporter - Servo-Motor" and integrate the servo-motor.

- Test the servo motor as intended by setting it in different positions.

Materials:

- Servo motor.
- 6 pieces. he / he wires.

Cheat Sheets:

- Effectors – Servo-Motor.

2. Deliver the box:
Used a function to let the robot deliver a box when it reaches the correct area.

Task:

- Create a “deliverBox” function that, with the help of the servo motor, tips the load so that the box falls off, after which the servo motor runs back to its starting point.
- Test that the function works, and that the robot can now run a route where it at the correct place, can now deliver the box it is transporting.

1.1.3.5 Ideas for expanding upon the transporter:

Feel free to come up with some ideas for potential expansions, for example, to make different LEDs light up to indicate what actions the robot is currently doing. is about to undertake. It may also be to use an LDR (Light Dependent Resistor) to check if the robot has a box on the bed.