ARTIE: Artificial Intelligence in Education - challenges and opportunities of the new era: development of a new curriculum, guide for educators and online course for students
Project co-funded by European Union under Erasmus+ Programme, 2020-1-HR01-KA201-077800

TITLE: Programing a robot - object following

LEARNING SCENARIO			
School:		*Duration (minutes):*	90
Teacher:		*Students' age:*	13-14

Essential Question:	How to make a robot ready for object following

Topics:

- Programing robot for object following using object tracking and object recognition

Aims:

- To learn how to program a robot for object following

Outcomes:

- Knowing how to write a program for a robot to be able to follow an object

Work forms:

- work in pairs, group work

Methods:

- presentation, talk, discussion, interactive exercise

ARTICULATION

Course of action (duration, minutes)

INTRODUCTION

We learned how to make a robot move in our previous lesson.
Check with your students if they understand everything from the last class and if they are ready for the next step.
Let's move our robots toward a specific object now. First, we must detect that object and track it. Tracking a moving object requires visual object tracking technology as well as manual operation.

ERASMUS+ ARTIE
ARTIFICIAL INTELLIGENCE IN EDUCATION

Co-funded by the Erasmus+ Programme of the European Union

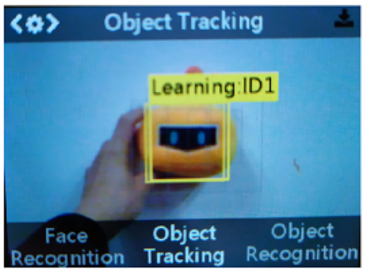
ARTIE: Artificial Intelligence in Education - challenges and opportunities of the new era: development of a new curriculum, guide for educators and online course for students
Project co-funded by European Union under Erasmus+ Programme, 2020-1-HR01-KA201-077800

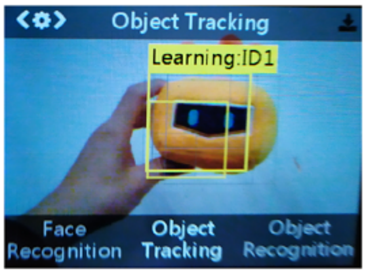
MAIN PART

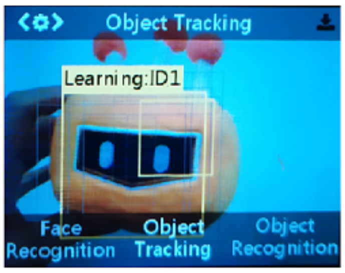
Object tracking is an important assignment in computer vision. It refers to the process of continuously inferring the state of objects in video sequences. The image is collected by a single camera and the image information is transmitted to a microcontroller. After analysing and processing, the relative position of the moving object is calculated. At the same time, the robot carrying camera is controlled to rotate and track the object in real time.
When the object tracking system performs the tracking function, it is mainly divided into 4 steps:

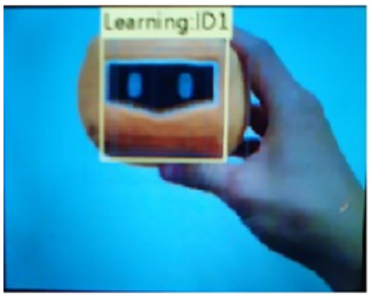
- object recognition
- object tracking
- object movement analysis
- controlling the robot (or any other system) with camera

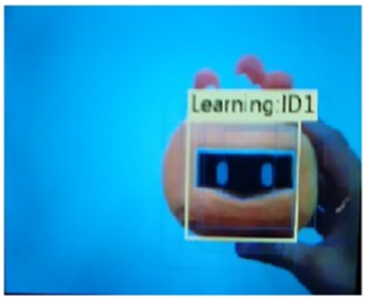
Object recognition - Learning
Connect the micro:bit or Arduino UNO with HuskyLens camera to your laptop or desktop computer. Point HuskyLens to the target object, adjusting the distance and until the object is contained in the orange bounding box of the centre of the screen. It is also acceptable that only part of the object included in the box but within distinct features.
Then long press "learning button" to learn the object from various angles and distances. During the learning process, the orange box with words "Learning: ID1" will be displayed on the screen.

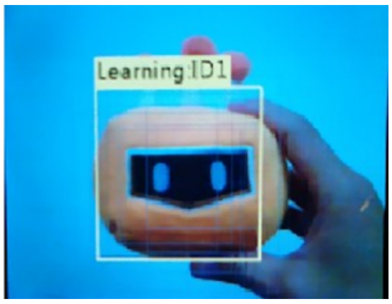




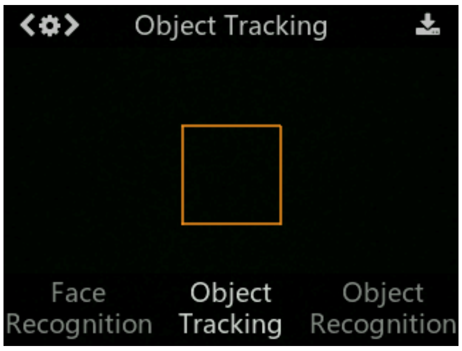




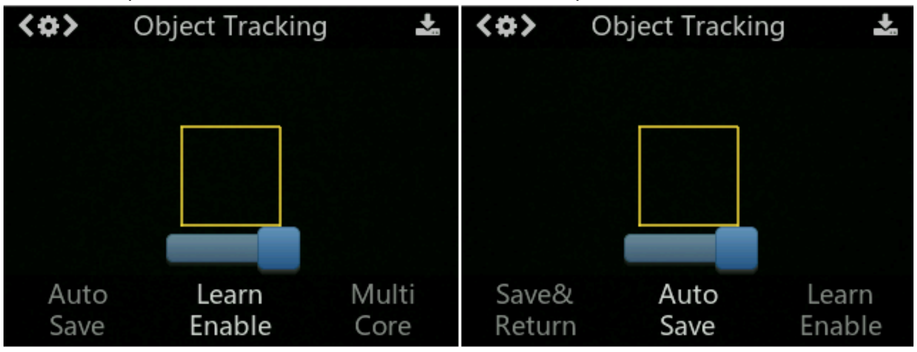




When HuskyLens can track the object at different angles and distances, then release the "learning button" to complete the learning. If there is no orange box on the centre of the screen, it means that the HuskyLens has already learned an object. If you want to track another object - select “Forget Learned Object” and learn again.
Under the object tracking function, HuskyLens can keep learning, that is, as long as the camera sees the learned object, it will keep learning the current state of the object, which is conducive to capturing dynamic object. Operation method: Long press the function button to enter the parameter setting of the object tracking function.

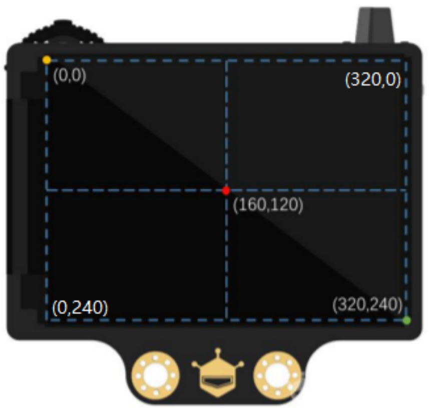


Dial the function button to the right to select "Learn Enable", then short press the function button, and dial it to the right to turn the "Learn Enable" ON, that is, the square icon on the progress bar is turned to the right. Then short press the function button to confirm this parameter.



When restarting HuskyLens, the last learned object is not saved by default, and you can turn on the switch to save models automatically.
Operation method: the same as above, after entering the parameter setting, switch "Auto Save" ON. In this way, you only need to learn the object once. Restarting the camera, the object you learned last time will be saved.

Object tracking
HuskyLens sensor screen resolution is 320*240, as shown in the following picture.

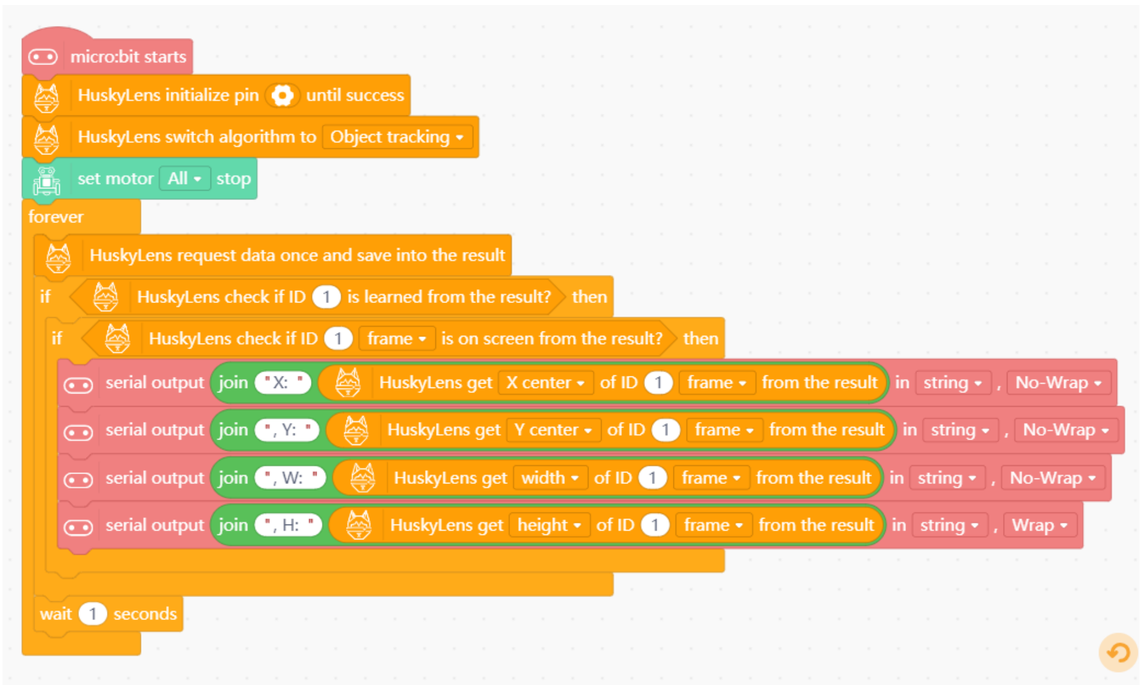


The coordinates of the object centre point obtained through the program are also within this range. For example, if the coordinate values obtained are (160, 120), the object being tracked is in the centre of the screen.
"X coordinates" and "Y coordinates" refer to the position of the box centre point in the screen coordinate. "Object width" and "Object height" refer to the size of the frame. Under the object tracking function, the frame is square, so the width and height are equal.

Test the object tracking - Option 1 (Maqueen Plus/HuskyLens)

Open your Mind+ and load extensions for work with Maqueen Plus and HuskyLens camera.

Use this code:



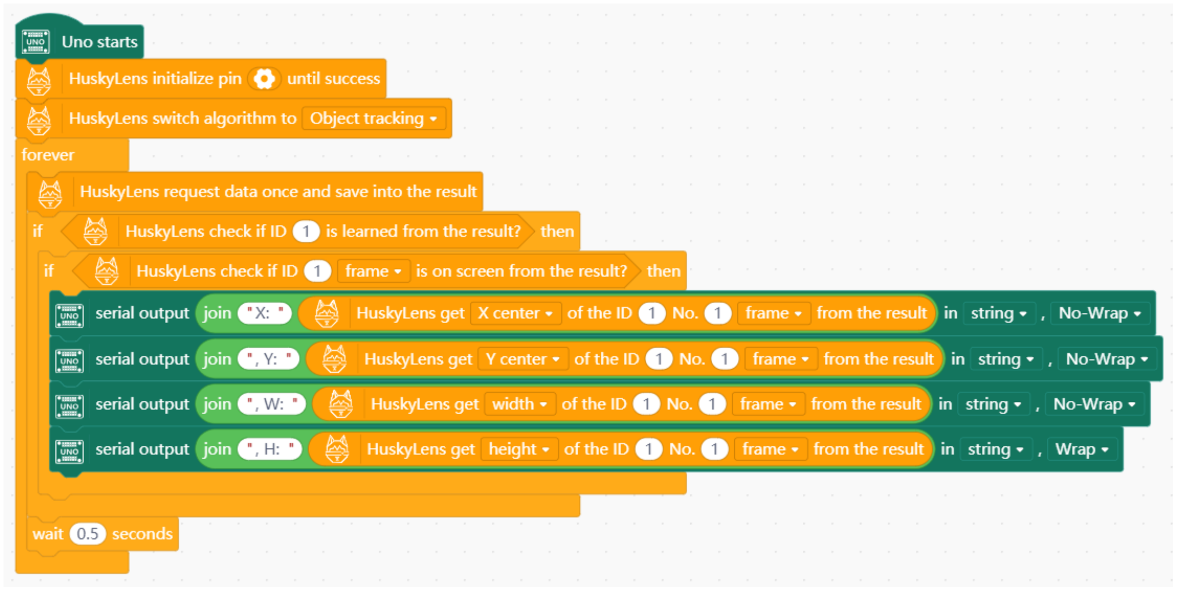
```
micro:bit starts
  HuskyLens initialize pin ⚙ until success
  HuskyLens switch algorithm to [Object tracking ▾]
  set motor [All ▾] stop
forever
  HuskyLens request data once and save into the result
  if < HuskyLens check if ID (1) is learned from the result? > then
    if < HuskyLens check if ID (1) [frame ▾] is on screen from the result? > then
      serial output join ("X: ") (HuskyLens get [X center ▾] of ID (1) [frame ▾] from the result) in [string ▾] , [No-Wrap ▾]
      serial output join (", Y: ") (HuskyLens get [Y center ▾] of ID (1) [frame ▾] from the result) in [string ▾] , [No-Wrap ▾]
      serial output join (", W: ") (HuskyLens get [width ▾] of ID (1) [frame ▾] from the result) in [string ▾] , [No-Wrap ▾]
      serial output join (", H: ") (HuskyLens get [height ▾] of ID (1) [frame ▾] from the result) in [string ▾] , [Wrap ▾]
  wait (1) seconds
```

Skip to **Checking results**

Test the object tracking - Option 2 (Arduino UNO/HuskyLens)

Open your Mind+ and load extensions for work with Arduino UNO and HuskyLens camera.

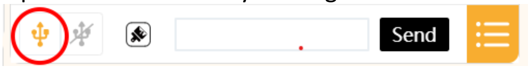
Use this code with Arduino/HuskyLens:



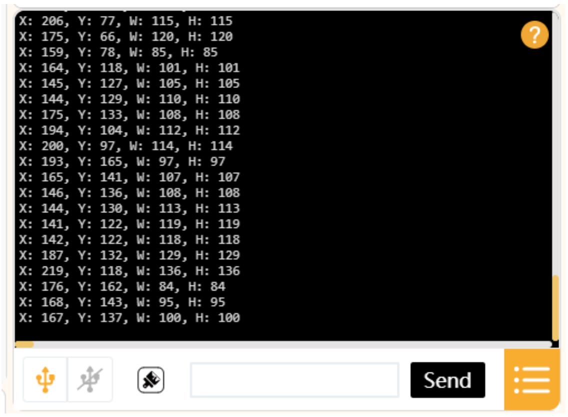
```
Uno starts
HuskyLens initialize pin until success
HuskyLens switch algorithm to Object tracking
forever
    HuskyLens request data once and save into the result
    if HuskyLens check if ID 1 is learned from the result? then
        if HuskyLens check if ID 1 frame is on screen from the result? then
            serial output join "X: " HuskyLens get X center of the ID 1 No. 1 frame from the result in string , No-Wrap
            serial output join ", Y: " HuskyLens get Y center of the ID 1 No. 1 frame from the result in string , No-Wrap
            serial output join ", W: " HuskyLens get width of the ID 1 No. 1 frame from the result in string , No-Wrap
            serial output join ", H: " HuskyLens get height of the ID 1 No. 1 frame from the result in string , Wrap
    wait 0.5 seconds
```

Checking results on serial monitor (both options)

Open serial monitor by clicking on USB icon in lower right part of the Mind+ screen.



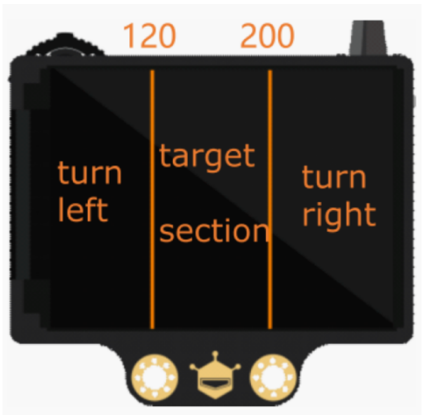
Try to move the object left and right to observe the numerical change of the X centre. Move the object up and down to observe the numerical change of Y centre. Move the object back and forth to observe the numerical change of width and height.



```
X: 206, Y: 77, W: 115, H: 115
X: 175, Y: 66, W: 120, H: 120
X: 159, Y: 78, W: 85, H: 85
X: 164, Y: 118, W: 101, H: 101
X: 145, Y: 127, W: 105, H: 105
X: 144, Y: 129, W: 110, H: 110
X: 175, Y: 133, W: 108, H: 108
X: 194, Y: 104, W: 112, H: 112
X: 200, Y: 97, W: 114, H: 114
X: 193, Y: 165, W: 97, H: 97
X: 165, Y: 141, W: 107, H: 107
X: 146, Y: 136, W: 108, H: 108
X: 144, Y: 130, W: 113, H: 113
X: 141, Y: 122, W: 119, H: 119
X: 142, Y: 122, W: 118, H: 118
X: 187, Y: 132, W: 129, H: 129
X: 219, Y: 118, W: 136, H: 136
X: 176, Y: 162, W: 84, H: 84
X: 168, Y: 143, W: 95, H: 95
X: 167, Y: 137, W: 100, H: 100
```

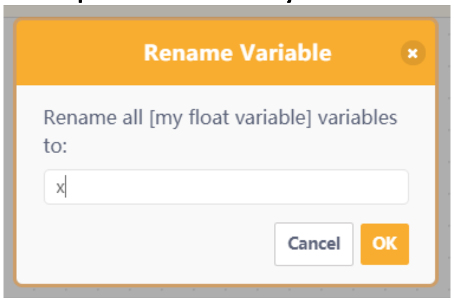
Object motion analysis

As shown in the following picture, the screen is divided into 3 sections according to the X axis of the camera screen coordinate system, and the middle section is our target section.



When the camera continuously detects the state of the target object in the picture, its X centre is 120-200, which means that the target is in the centre of the field of vision and robot does not need to adjust its position; its X centre is 0-120, our robot needs to adjust by turning right; its X centre is 200-320, ARTIEbot needs to turn left to adjust.

Now it’s time to write the main part of the code to turn the robot toward the object.
Both options - Rename **my float variable** to **x**. Right click on variable -> Rename numeric variable.

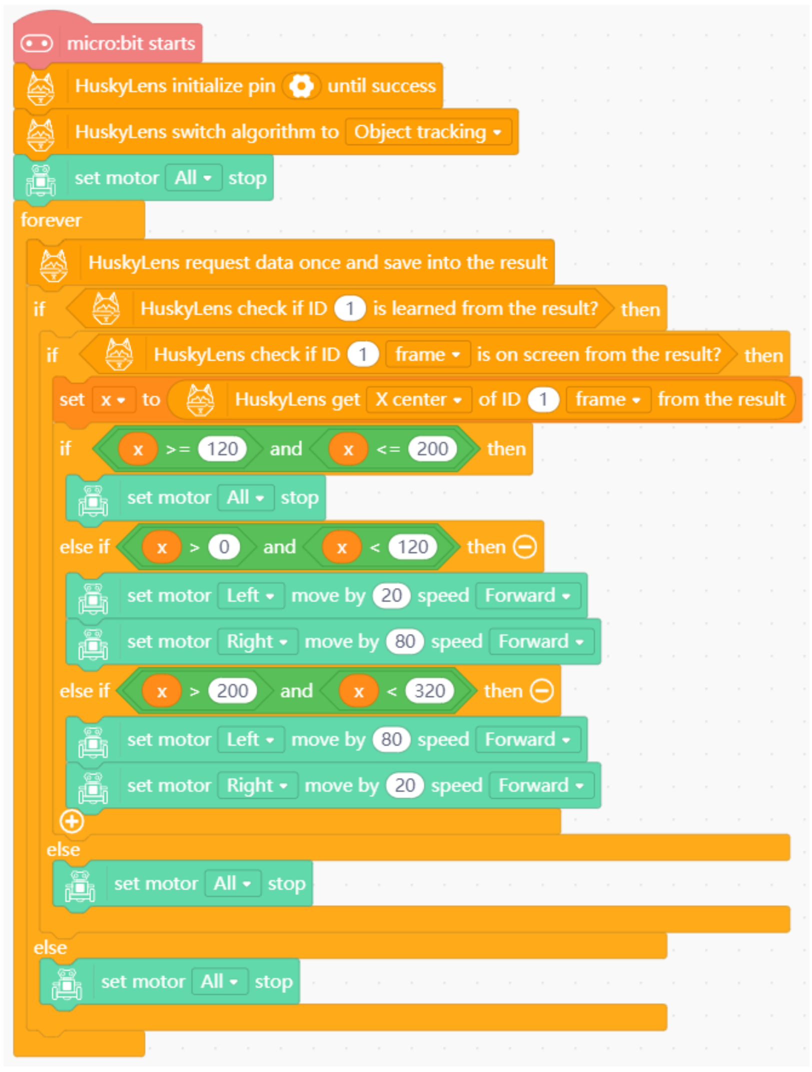

Rename Variable

Rename all [my float variable] variables to:

x

Option 1 - track the object with Maqueen Plus

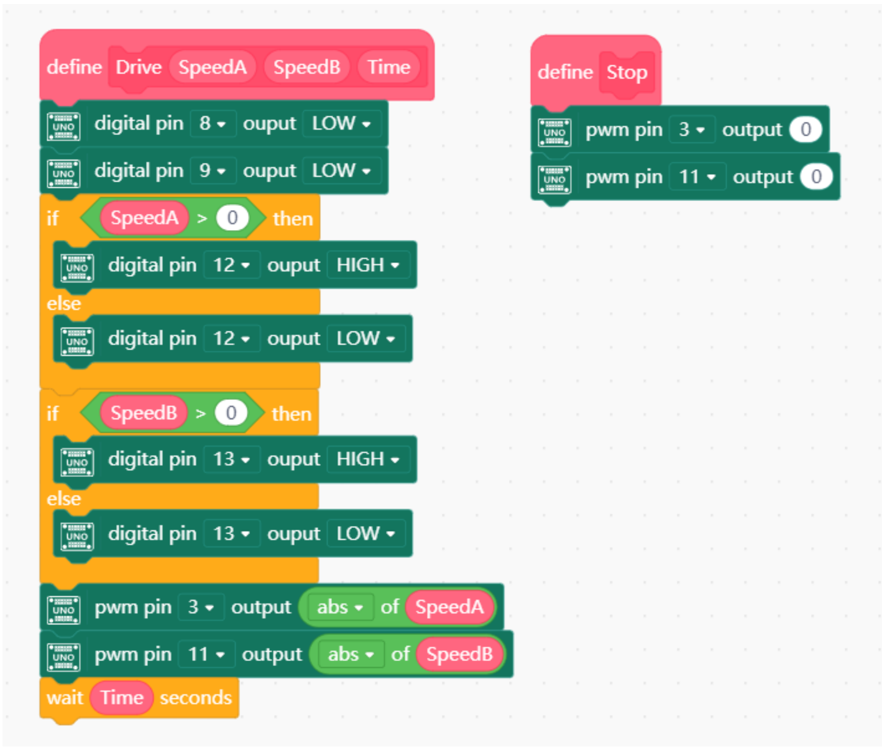
Use and configure blocks as on the picture below:



```
micro:bit starts
HuskyLens initialize pin ⚙ until success
HuskyLens switch algorithm to Object tracking ▾
set motor All ▾ stop
forever
    HuskyLens request data once and save into the result
    if HuskyLens check if ID 1 is learned from the result? then
        if HuskyLens check if ID 1 frame ▾ is on screen from the result? then
            set x ▾ to HuskyLens get X center ▾ of ID 1 frame ▾ from the result
            if x >= 120 and x <= 200 then
                set motor All ▾ stop
            else if x > 0 and x < 120 then ⊖
                set motor Left ▾ move by 20 speed Forward ▾
                set motor Right ▾ move by 80 speed Forward ▾
            else if x > 200 and x < 320 then ⊖
                set motor Left ▾ move by 80 speed Forward ▾
                set motor Right ▾ move by 20 speed Forward ▾
            ⊕
        else
            set motor All ▾ stop
    else
        set motor All ▾ stop
```

Option 2 - track the object with ArtieBot

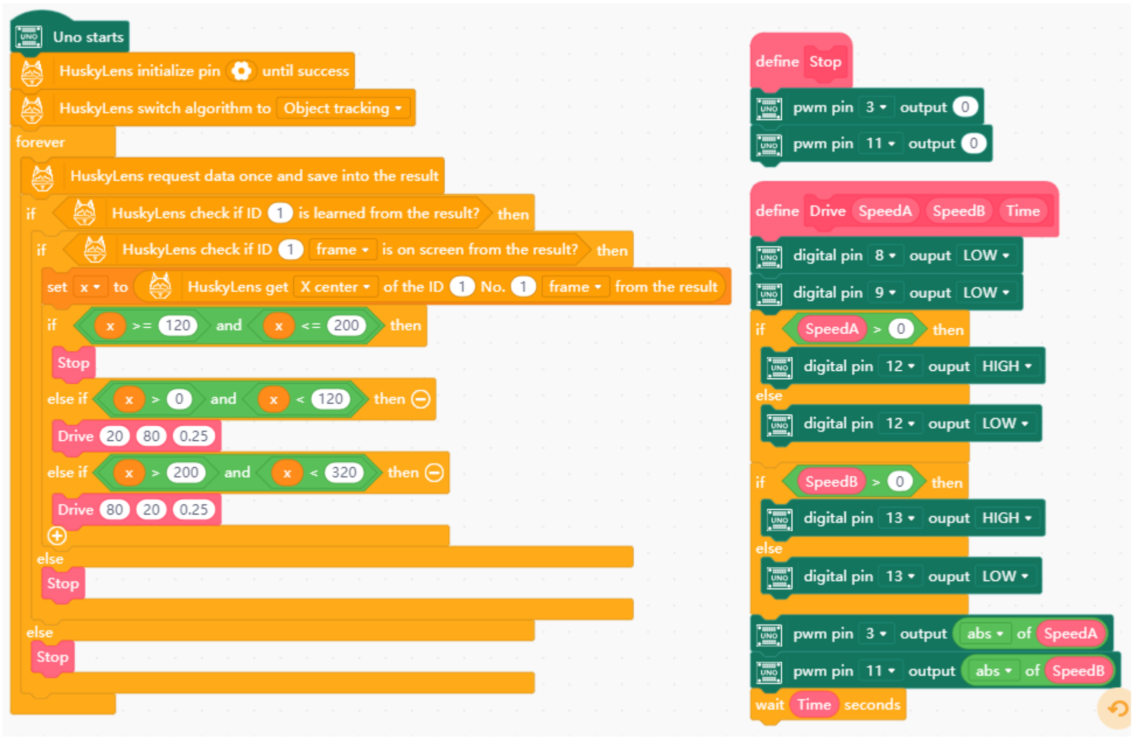
First define blocks Drive and Stop as described in the previous lesson (Programing the robot)



```
define Drive SpeedA SpeedB Time
  digital pin 8 ouput LOW
  digital pin 9 ouput LOW
  if SpeedA > 0 then
    digital pin 12 ouput HIGH
  else
    digital pin 12 ouput LOW
  if SpeedB > 0 then
    digital pin 13 ouput HIGH
  else
    digital pin 13 ouput LOW
  pwm pin 3 output abs of SpeedA
  pwm pin 11 output abs of SpeedB
  wait Time seconds

define Stop
  pwm pin 3 output 0
  pwm pin 11 output 0
```

Use and configure blocks as on picture below:



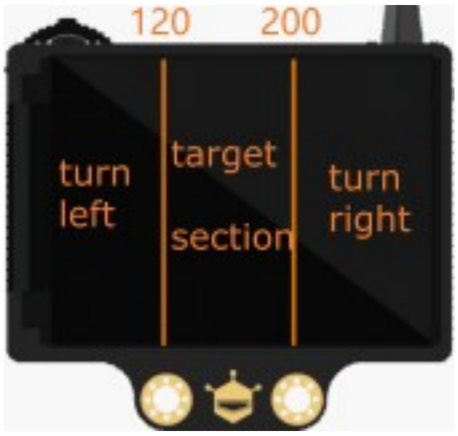
```
Uno starts
HuskyLens initialize pin until success
HuskyLens switch algorithm to Object tracking
forever
  HuskyLens request data once and save into the result
  if HuskyLens check if ID 1 is learned from the result? then
    if HuskyLens check if ID 1 frame is on screen from the result? then
      set x to HuskyLens get X center of the ID 1 No. 1 frame from the result
      if x >= 120 and x <= 200 then
        Stop
      else if x > 0 and x < 120 then
        Drive 20 80 0.25
      else if x > 200 and x < 320 then
        Drive 80 20 0.25
    else
      Stop
  else
    Stop

define Stop
  pwm pin 3 output 0
  pwm pin 11 output 0

define Drive SpeedA SpeedB Time
  digital pin 8 ouput LOW
  digital pin 9 ouput LOW
  if SpeedA > 0 then
    digital pin 12 ouput HIGH
  else
    digital pin 12 ouput LOW
  if SpeedB > 0 then
    digital pin 13 ouput HIGH
  else
    digital pin 13 ouput LOW
  pwm pin 3 output abs of SpeedA
  pwm pin 11 output abs of SpeedB
  wait Time seconds
```

BOTH options - Check how it works

Upload the program to your robot.
Make the adjustments to speed of MotorA or MotorB if necessary.
When the box of identified object is in the centre of the screen, the robot stops.
When the box is on the left or right side of the screen, the robot automatically adjusts the position left or right until the box is located in the target section of the screen.

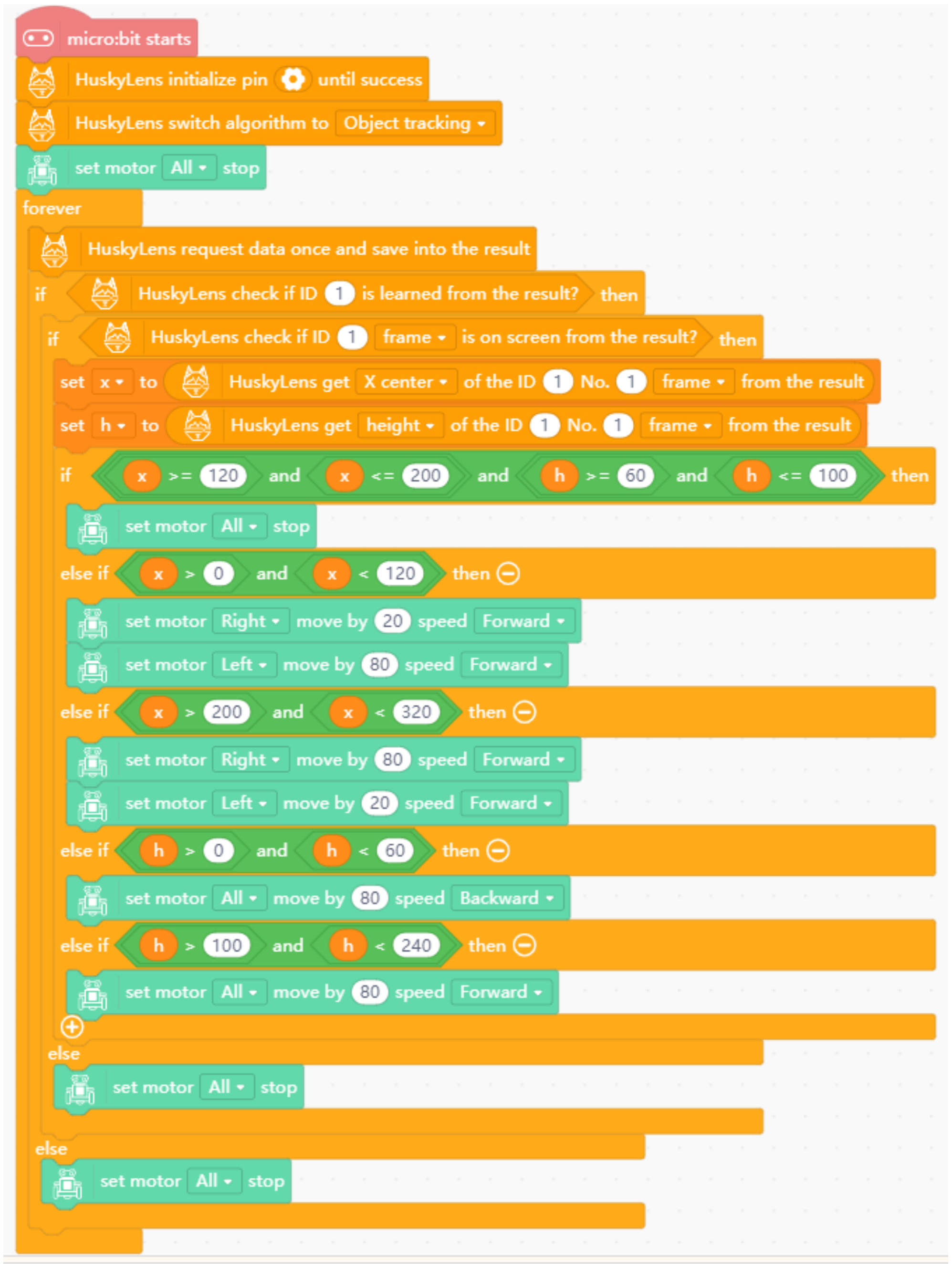


Follow the object

We got our robots turning to object but still not following. To achieve this, we'll have to detect the size of the object to find out if it's large (close to the camera) or small (far from the camera).

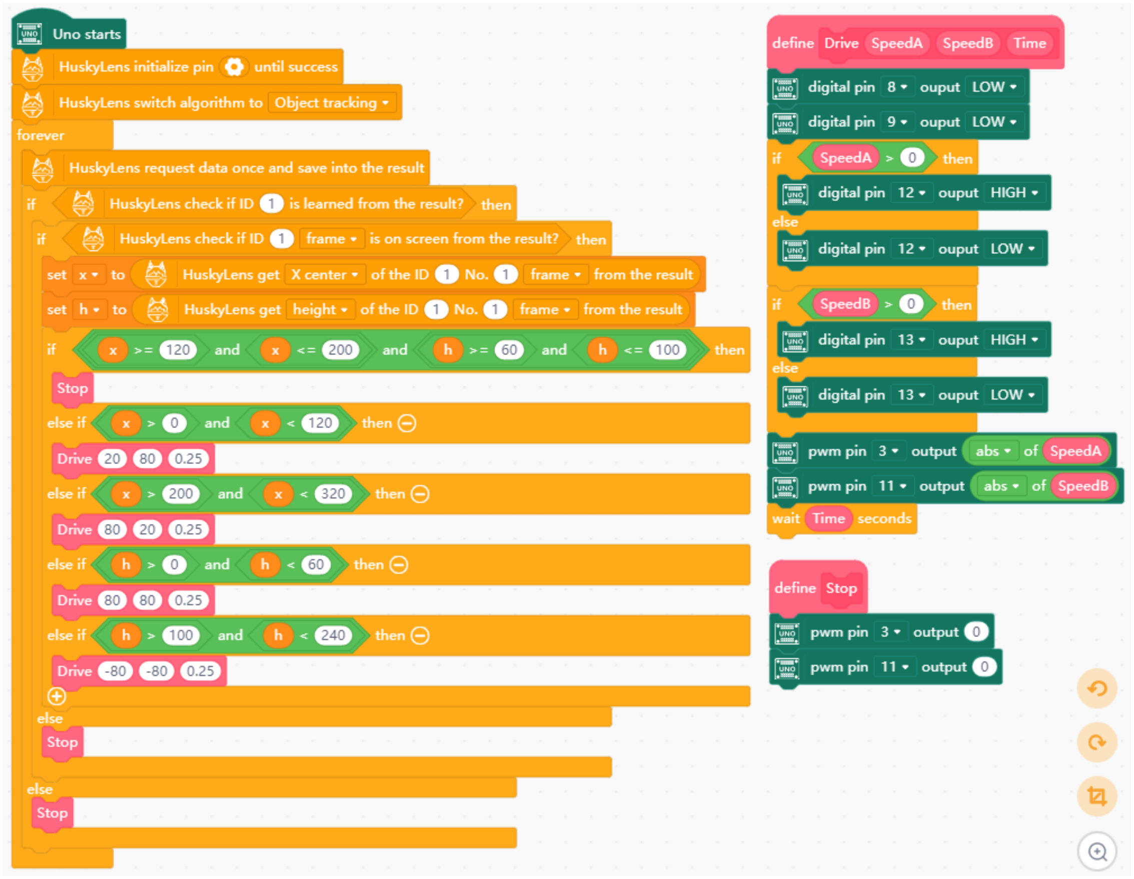
Make a new variable (numeric) and name it **h**. It will keep the height of the object we are tracking. If the object height is between 60 and 100, the robot will keep the current position. If it's smaller than 60, it's far and we need to make the robot drive forward. If it's higher than 100, the robot should move backward.

Option 1 - Code for Maqueen Plus



```
micro:bit starts
  HuskyLens initialize pin ✪ until success
  HuskyLens switch algorithm to Object tracking
  set motor All stop
forever
  HuskyLens request data once and save into the result
  if HuskyLens check if ID 1 is learned from the result? then
    if HuskyLens check if ID 1 frame is on screen from the result? then
      set x to HuskyLens get X center of the ID 1 No. 1 frame from the result
      set h to HuskyLens get height of the ID 1 No. 1 frame from the result
      if x >= 120 and x <= 200 and h >= 60 and h <= 100 then
        set motor All stop
      else if x > 0 and x < 120 then
        set motor Right move by 20 speed Forward
        set motor Left move by 80 speed Forward
      else if x > 200 and x < 320 then
        set motor Right move by 80 speed Forward
        set motor Left move by 20 speed Forward
      else if h > 0 and h < 60 then
        set motor All move by 80 speed Backward
      else if h > 100 and h < 240 then
        set motor All move by 80 speed Forward
    else
      set motor All stop
  else
    set motor All stop
```

Option 2 - Code for Arduino (ArtieBot):



```
Uno starts
HuskyLens initialize pin ⚙ until success
HuskyLens switch algorithm to Object tracking
forever
  HuskyLens request data once and save into the result
  if HuskyLens check if ID 1 is learned from the result? then
    if HuskyLens check if ID 1 frame is on screen from the result? then
      set x to HuskyLens get X center of the ID 1 No. 1 frame from the result
      set h to HuskyLens get height of the ID 1 No. 1 frame from the result
      if x >= 120 and x <= 200 and h >= 60 and h <= 100 then
        Stop
      else if x > 0 and x < 120 then
        Drive 20 80 0.25
      else if x > 200 and x < 320 then
        Drive 80 20 0.25
      else if h > 0 and h < 60 then
        Drive 80 80 0.25
      else if h > 100 and h < 240 then
        Drive -80 -80 0.25
    else
      Stop
  else
    Stop

define Drive SpeedA SpeedB Time
  digital pin 8 ouput LOW
  digital pin 9 ouput LOW
  if SpeedA > 0 then
    digital pin 12 ouput HIGH
  else
    digital pin 12 ouput LOW
  if SpeedB > 0 then
    digital pin 13 ouput HIGH
  else
    digital pin 13 ouput LOW
  pwm pin 3 output abs of SpeedA
  pwm pin 11 output abs of SpeedB
  wait Time seconds

define Stop
  pwm pin 3 output 0
  pwm pin 11 output 0
```

Both options - Check how it works

Upload the program to micro:bit/Arduino UNO to check how it works.
Do the corrections to make the movement smooth by adjusting motors speed and time of driving.
After HuskyLens finishes learning an object, robots will automatically follow the object and move forward, backward, left and right, keeping the object box in the centre of the screen and at a suitable distance.

When the robot is used as a tracking robot, it can be programmed to locate any target with HuskyLens camera. It means that you can turn this project into a person follower and make the robot follow people.

CONCLUSION

Object tracking is the task of taking an initial set of object detections, creating a unique ID for each of the initial detections, and then tracking each of the objects as they move around frames in a video, maintaining the ID assignment. State-of-the-art methods involve fusing data from RGB and event-based cameras to produce more reliable object tracking.
Now we understand the basic principles of object tracking, learn the HuskyLens object tracking function.
We also know how to use HuskyLens to make our robot follow the target.

Do the K.W.L. (**K**now, **W**ant, **L**earned) chart with your students .

What I **K**now	What I **W**ant to Know	What I **L**earned

Methods	**Work forms**
presentation *interactive exercise / simulation on the computer*	*work in pairs* *group work*

Material:

- http://mindplus.cc/download-en.html

Literature

-

PERSONAL OBSERVATIONS, COMMENTS AND NOTES