PLANETARY MOTION – KEPLER'S 2ND LAW

STEMJAM Teaching Guide

Developing make spaces to promote creativity around STEM in schools

Acronym: STEMJAM
Project no. 2016-1-ES01-KA201-025470

www.stemjam.eu

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

PLANETARY MOTION – KEPLER'S 2nd LAW

ABSTRACT

Kepler's Laws are not always easily understood, especially by younger students or students from professional schools, who tend to see it as a technicism far from their interest. mBot and its flame sensor allow to simulate this motion so it can be easily grasped at once, and the supporting material suggests a way to pose the topic in the larger context of cosmological research, in order to better understand its importance.

The activity uses the flame sensor – when the mbot is closer to fire, it runs faster.

DIDACTIC OBJECTIVES

While playing the activity you will learn about:

- Science: Kepler's first and second laws.
- Philosophy / Science History: its historical importance for the development of astronomy in the Renaissance age.

While implementing or inspecting the code

- Physics: the conservation of angular momentum.
- Technology: the response curve of a sensor.

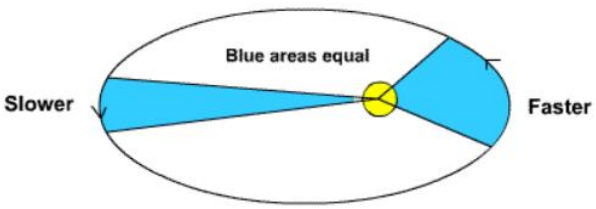
STEM Subject: Science☒ Technology ☒ Engineering☐ Mathematics☐

Education Level: 12-14 years☒ 14-16 years☒

PROBLEM STATEMENT

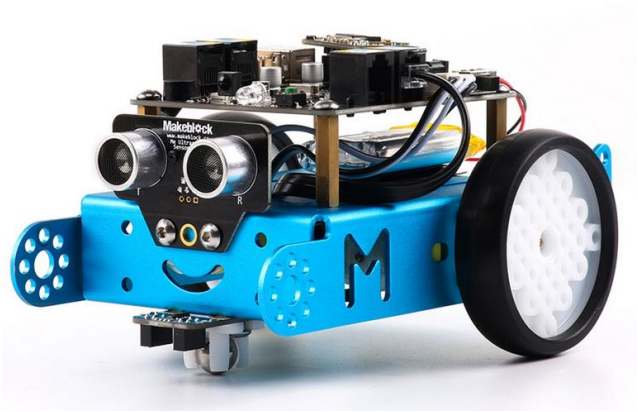
Simulate the motion of a planet around the sun, following an elliptic pathway with speed variable according to Kepler's second law (actually a consequence of the conservation of angular momentum).

"The line joining a planet and the Sun sweeps out equal areas during equal intervals of time "

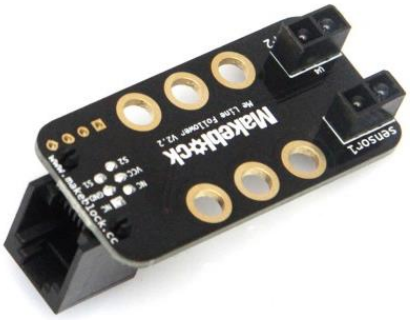


BOM (Bill of Materials Needed)

- mBot => Ref. 90054

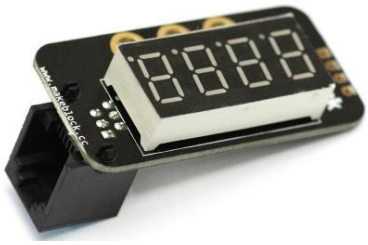


- Me **Line Follower**:



- Me **Flame Sensor**:

- Me **7-Segment Serial Display - Red**:



- A large sheet of paper with an elliptic black path on white background.
- A slim candle to simulate the sun.

ELEMENT	ID	CABLE	AMOUNT	PORT 1			PORT 2			PORT 3				PORT 4				P.MOT 1	P.MOT 2
				Y	B	W	Y	B	W	Y	B	W	Bl	Y	B	W	Bl	W*	W*
Mbot Robot 2´4G			2																
Motor 1	W*																	W*	
Motor 2	W*																		W*
Me 7-Segment serial display	B		1					B											
Me Line Follower	B		1		B														
Me Flame sensor	Bl		1														Bl		
RJ25 cables			3																
Structures and beams																			
Laptops			1																
Attrezzo (not essential)																			

ACTIVITY DESCRIPTION

The flame sensor is exploited to simulate the motion of a planet around the sun. From the sensor output, related to distance, the power of the motor is selected. In the following, we list the simple steps needed to play the activity, discuss the characteristics of the Flame sensor and give reasons for our choice of the algorithm. Finally we comment the code and the variables used therein. Through the text, useful tips leading to better results are mentioned and highlighted in orange color.

1. Turn on the candle and position it in a focus of the ellipse. It is better to use an ellipse with relatively large eccentricity to see easily the change in speed. (We used an ellipse with semi-axes of 40 cm and 25 cm).
2. Place the mBot on the path, oriented to run clockwise.
3. Press the on board button to start the simulation.
4. Mbot's speed will change during the experiment accordingly with the distance from the candle.

The experiment works better if you turn off the lights and, most important, avoid sunlight (or it is low at least). Indeed remember that not light but infrared radiation (=heat) is detected.

<u>The MeFlame sensor and its application to the present experiment</u>

The Me Flame Sensor is an infrared radiation detector. According to the documentation available [1] it is able to detect radiation with wavelength in the range from 760 nm to 1100 nm, with the highest sensitivity reached near 940 nm. It should be able to detect radiation up to a distance of 1 m and within an angle of 60°. When a flame is detected, its blue indicator will light on.

The flame sensor has both analog and digital outputs. The possible digital values are only Fire and NoFire; Analog reading returns values from 10 to 1023: a smaller number means that the sensor is closer to fire. In a dark room you will get 1023.

An experimental test to detect the sensor output as a function of distance, returns the data plotted in the figure below. A small Arduino code to run similar tests is also made available.

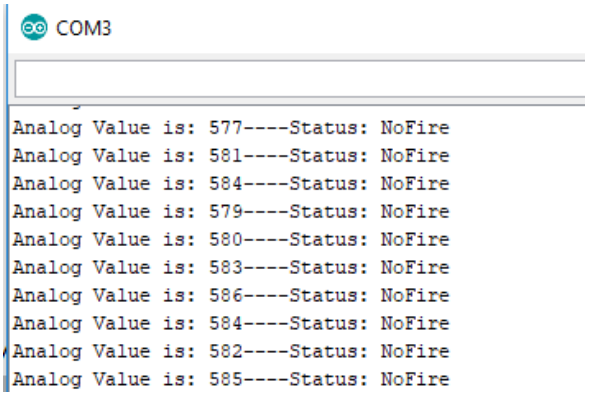
Arduino Code for sensor testing

Connect flame sensor to port 4, and 7-segment display to port 2.

Write the program in Arduino and upload it to the board:

```
//Libraries.   MeMCore  is the main library to control  mBot and other Makeblock products
#include <Arduino.h>
#include <Wire.h>
```

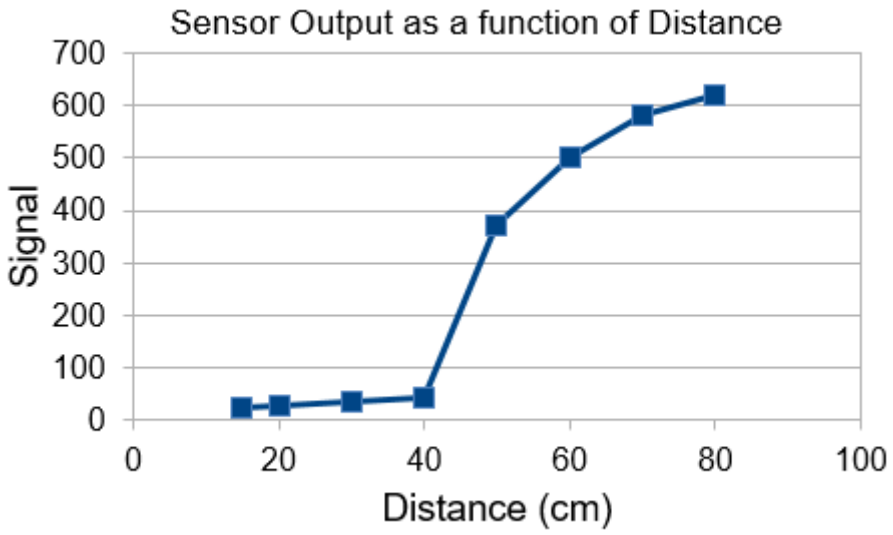
```
#include <SoftwareSerial.h>
#include <MeMCore.h>

// Global variables declaration.  Special classes are made available for motor control
and sensor and other mBot  components. In particular:
MeRGBLed for the onboard led,  MeFlameSensor for the flame sensor, Me7SegmentDIsplay for the Liquid crystals display
MeFlameSensor FlameSensor1(PORT_4);
Me7SegmentDisplay disp(PORT_2);



void setup(){              //Commands that are  run once at program start
{
  Serial.begin(9600);
}

void loop(){    //Commands run repeatedly till program ends
{
  Serial.print("Analog Value is: ");
  Serial.print(FlameSensor1.readAnalog());
  Serial.print("----Status: ");
  if(FlameSensor1.readDigital() == Fire)
   {
     Serial.println("Fire");
   }
  else if(FlameSensor1.readDigital() == NoFire)
   {
     Serial.println("NoFire");
   }
  disp.display(FlameSensor1.readAnalog());
  delay(200);
}
```



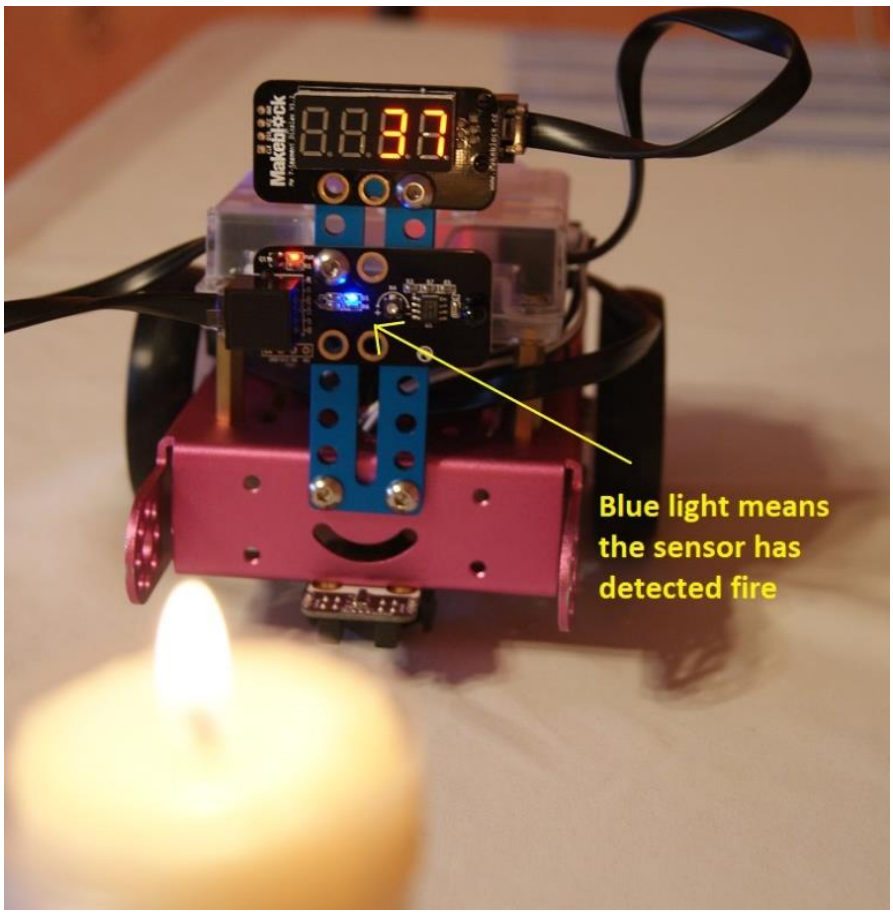
Your output in the Arduino serial monitor (open Tools --> Serial monitor) will be similar to the one reported above. The analog value will be seen on 7-segment display, so you can also use mbot without PC.

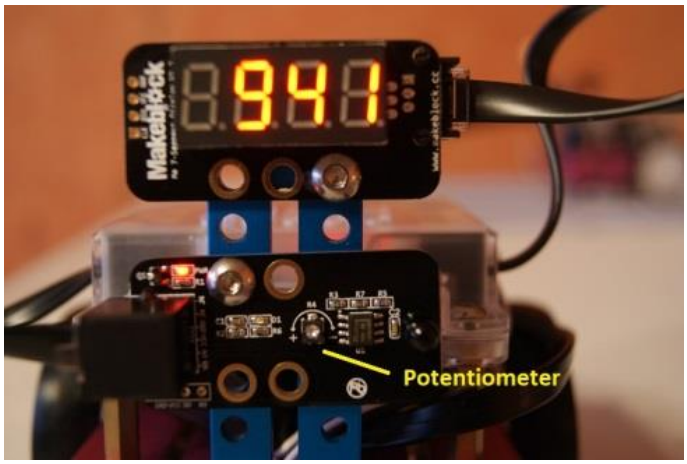
When the “Fire” status will be active the blue led on the sensor board will turn on.

You can also change the sensitivity of the sensor using the onboard potentiometer. In particular you can change the fire detection range (For example: the robot detects fire from a distance of 15 cm; when you turn the potentiometer towards the minus, the robot will detect fire from a distance of 20 cm).

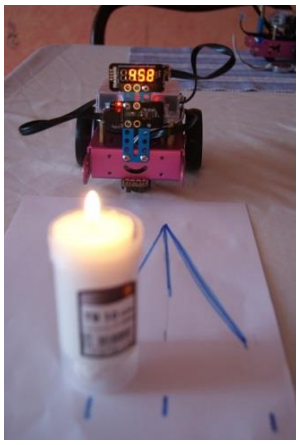


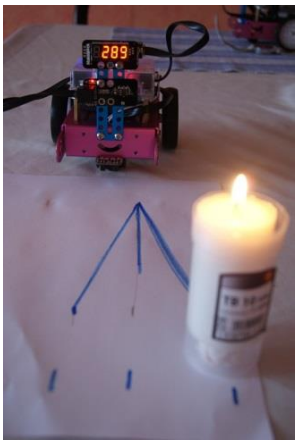
The sensor results linear up to 40 cm and then behaves differently (no logarithmic, exponential or square dependence were found).





If you would like to check the angle of detection too, please move the candle and draw a line where the sensor loses the fire: in our case the angle resulted 40°.





The proposed solution

The original algorithm conceived included:

- Measurement from the sensor.
- Calculation of the corresponding speed according to the conservation of angular momentum (the product of distance and speed should be constant).
- Evaluation of the power of the motors from the desired speed, according to the data obtained in the "Speed" activity.

Due to the sensor non-linear response and to its limited accuracy (repeated results give slightly different values) we decide to skip the intermediate calculation for speed and just made a correspondence table (sensor output in a certain range) ---> (select a power) . The power increase when approaching, and decrease when going away from the "sun". As already stated in the introduction, the accuracy is limited as the flame sensor detects not only to the flame but also to the light in the room, especially sunlight. It is therefore very difficult to calculate the exact speed according to Kepler's Law and the activity results more qualitative than quantitative. Nevertheless in a classroom dark enough you will appreciate the speed change and get a satisfying picture of what's going on during the planetary motion.

We may suggest two possible improvements:

- Use a smaller ellipse, with maximum Sun-Earth distance of 40 cm, in order to be within the linear response region of the sensor. This might allow to use the sensor exact value and evaluate the corresponding speed (distance*speed = costant) without using the correspondence table, which cannot be so accurate. A smaller ellipse, however, may be suitable to show the simulation only to small groups of students.

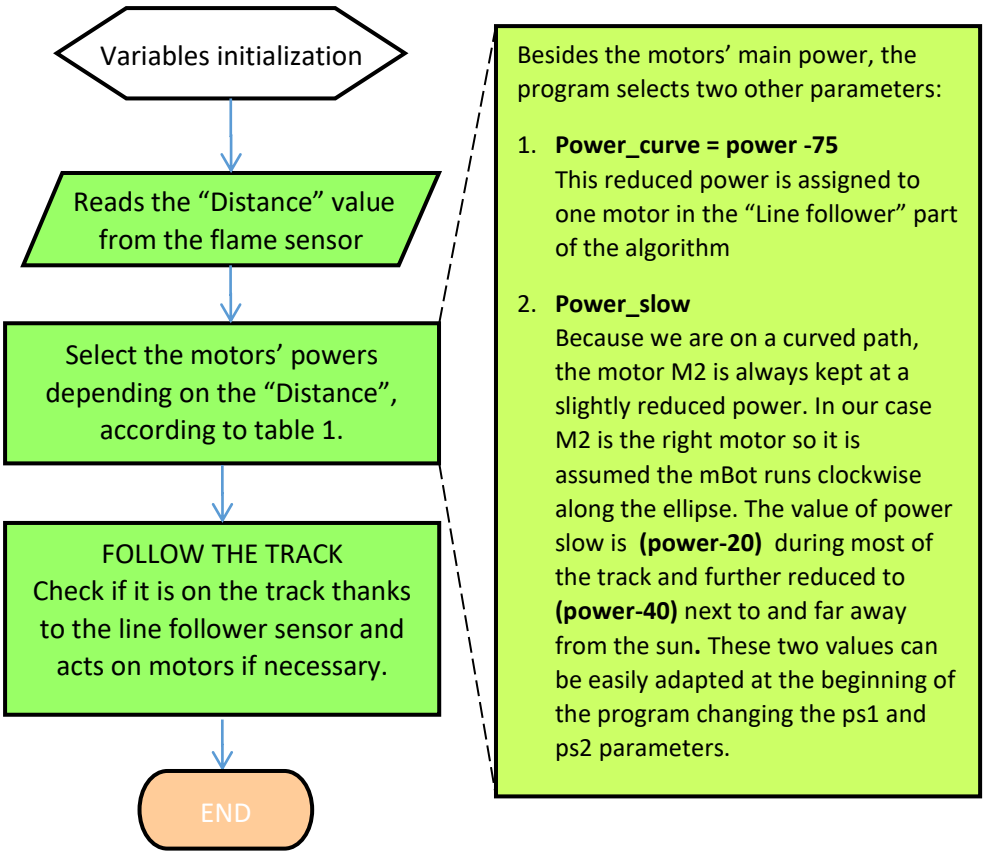
- Go back to the "speed" experiment and select the engine's power in relation with effective speed (again the relation proved to be not linear). This small correction might be appreciated only in the linear response region of the sensor.

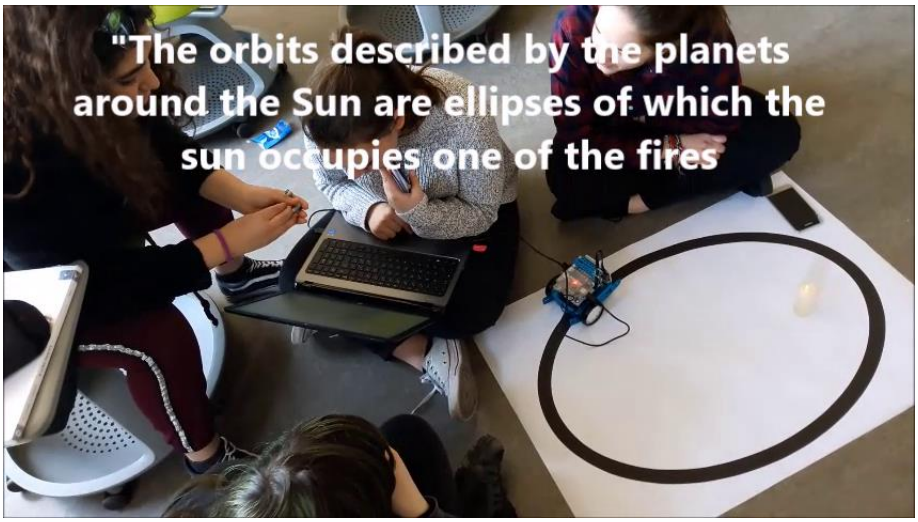
Algorithm workflow and Comments to code

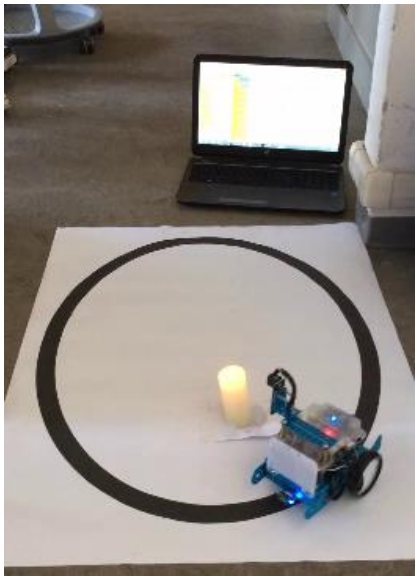
The algorithm is represented in the workflow below.

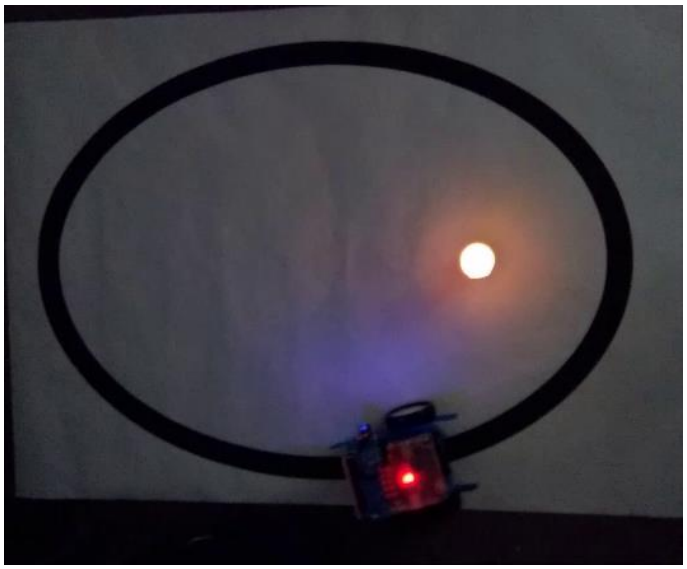
As already discussed above, depending on the sensor output (related to the "Sun-Earth" distance) the motors' powers are selected according to the following table:

Sensor Output in the range	0 - 15	15 -30	30 - 40	40 - 50	50 - 60	60 -70	70 - 80	> 80
Power	230	190	170	150	130	110	90	80

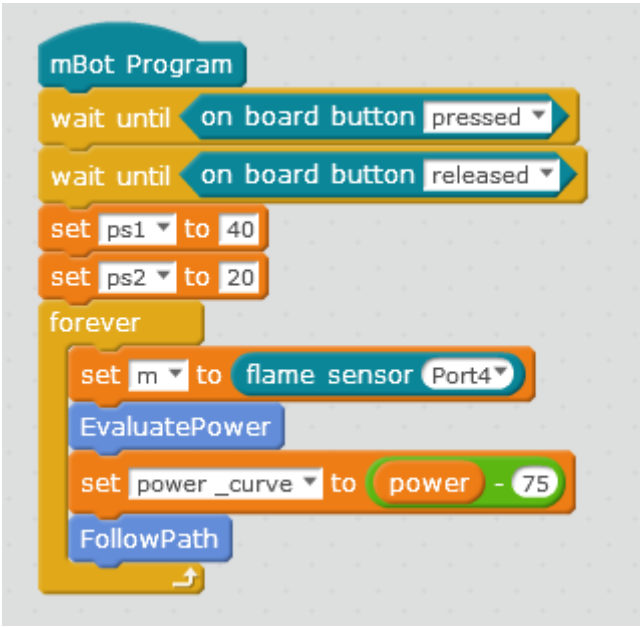


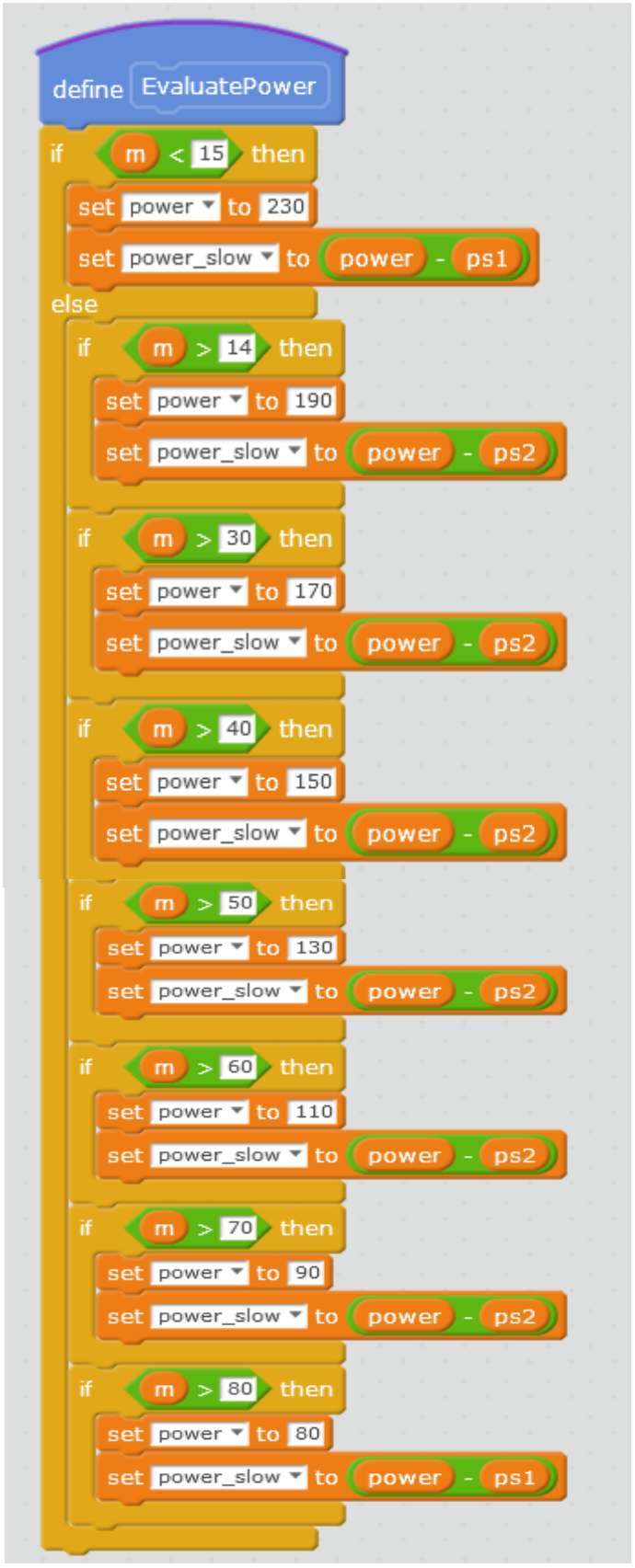


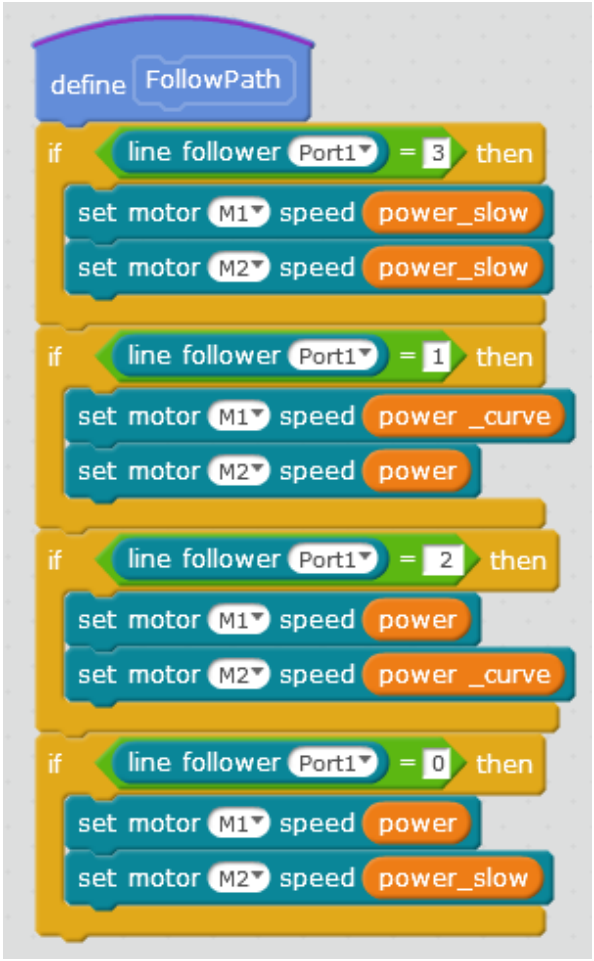




Makeblock (scratch) Code




define EvaluatePower
if m < 15 then
set power to 230
set power_slow to power - ps1
else
if m > 14 then
set power to 190
set power_slow to power - ps2
if m > 30 then
set power to 170
set power_slow to power - ps2
if m > 40 then
set power to 150
set power_slow to power - ps2
if m > 50 then
set power to 130
set power_slow to power - ps2
if m > 60 then
set power to 110
set power_slow to power - ps2
if m > 70 then
set power to 90
set power_slow to power - ps2
if m > 80 then
set power to 80
set power_slow to power - ps1

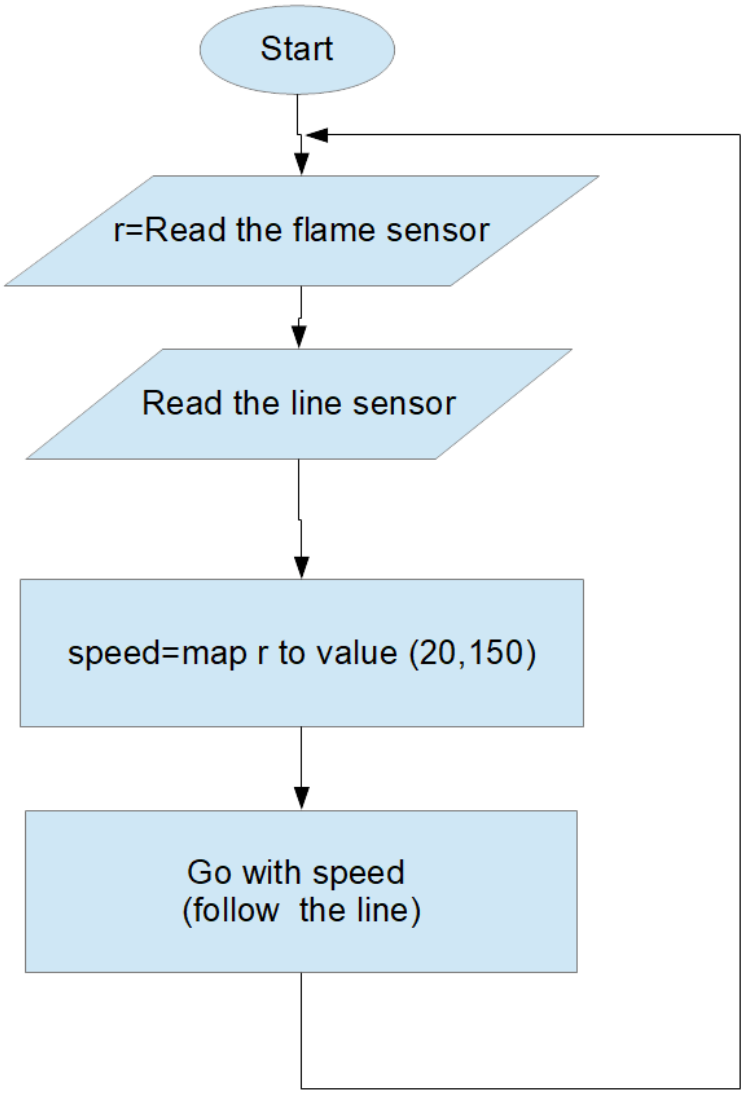

define FollowPath
if line follower Port1 = 3 then
set motor M1 speed power_slow
set motor M2 speed power_slow
if line follower Port1 = 1 then
set motor M1 speed power _curve
set motor M2 speed power
if line follower Port1 = 2 then
set motor M1 speed power
set motor M2 speed power _curve
if line follower Port1 = 0 then
set motor M1 speed power
set motor M2 speed power_slow

Arduino code

```
//Libraries
#include "MeOrion.h"

// Global variables declaration.  Special classes are made available for motor control and sensor and other mBot  components. In particular:
MeRGBLed for the onboard led,  MeFlameSensor for the flame sensor, Me7SegmentDIsplay for the Liquid crystals display
MeFlameSensor FlameSensor1(PORT_4);
MeDCMotor motor1(M1);
MeDCMotor motor2(M2);
MeLineFollower lineFinder(PORT_1);

void setup(){               //Commands run once at program start
{
}

void loop(){                //Commands run repeatedly till program ends
{
 int speed = map(FlameSensor1.readAnalog(),900,0,20,150);
 int sensorState = lineFinder.readSensors();
 switch(sensorState)
 {
  case S1_IN_S2_IN:
  motor1.run(-speed);
  motor2.run(speed);
  break;
  case S1_OUT_S2_IN:
  motor1.run(-speed);
  motor2.run(-speed);
  break;
  case S1_IN_S2_OUT:
  motor1.run(speed);
  motor2.run(speed);
  break;
```

```
    case S1_OUT_S2_OUT:
    motor1.run(-speed);
    motor2.run(speed);
    break;
    default: break;
  }
}
```

FLOW CHART



STUDENT'S EVALUATION

Indicators for students' evaluation may include:

- Science: She/He states correctly and understand Kepler's second law.
- Philosophy and History: She/He has an overview of the so-called "Astronomical revolution" started by Kepler and Kopernik in the 16th century.
- Physics: She/He has learned the idea Angular momentum and its conservation and is able to evaluate the speed of a planet at different distance from the sun.
- Computer Science: iterations, counters, logical operators, IF statement.

BIBLIOGRAPHY

[1] Sensor Description http://learn.makeblock.com/en/me-flame-sensor/

SCALABILITY

The activity is suitable for students aged 12 or higher.

With older students (14-15) increasing mathematical details can be included.