World Robot Olympiad – Open Category (Senior)

Team: Benedek Szakali, **A**nna Eszter Nyíri, **D**ániel Mihalik

TEAM_BAD

Hungary, Bányai Júlia Secondary Grammar School

Coach: Robert Kiss

SKYNET CITY

Global City Controlling System

CONTENTS

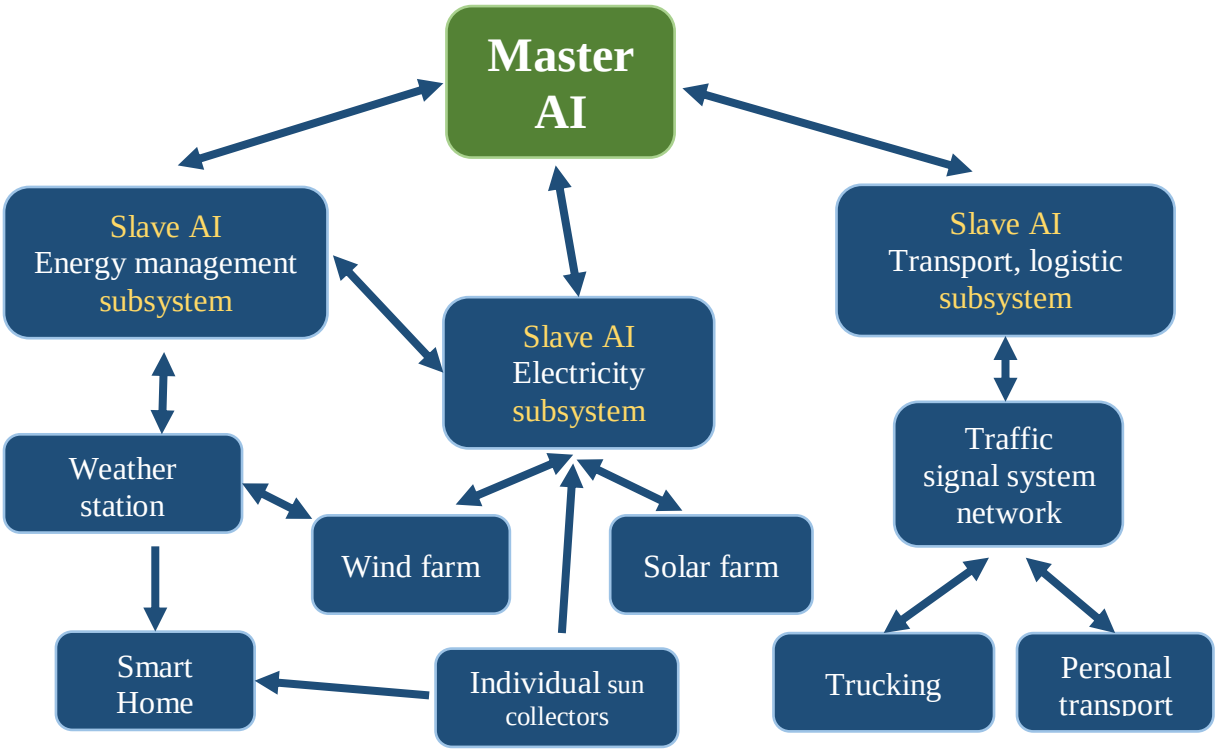
INTRODUCTORY STORY (SCIENCE FICTION?)

By 2050 the amount of exploitable fossil energy sources had decreased to such an extent that there followed a serious energy crisis. This certainly did not happen all of a sudden. Some researchers had already warned at the end of the 20th century that considering the population growth of the planet the energy crisis would be inevitable after the peak oil.
The problem was made worse by the fact that greenhouse gases produced by burning fossil energy resources had got into the air thus raising the average temperature of the atmosphere. At first reaching the symbolic 2 degrees Celsius (increase of average temperature compared to that of the beginning of the industrial revolution) was considered ideal. That seemed to be maintainable until the 2020s but politicians and economic policy reacted too late. True, they introduced restrictive measures concerning CO_2 emissions. Emissions trade was an economic idea (CO_2 quota); however, as long as a country or a company had their financial means, it was difficult to restrain them or to motivate them to introduce more efficient environmentally-conscious methods.
They saw the solution in the increase of the ratio of renewable energy resources. There were agreements with deadlines which prescribed the increase of ratios. By reshuffling money spent on researches there were serious efforts made to increase the efficiency of converting solar energy into electrical power. Brave estimates had envisaged 60% efficiency of solar power conversion but financial sources for research were scarce. Some of the developed industrialized countries were slow to realize the forthcoming dangers.
Explosion of population brought along energy crisis. There appeared a huge gap between developed and developing countries as to energy, food and drinking water supplies. Diseases, epidemics, malnutrition and poverty did not only lead to economic and political conflicts but they also resulted in slowing down population growth: the planet's capacities being an obstacle to the growth of population like a saturation curve but at the same time aggravating local and global conflicts.
By 2048 world population had reached 9 billion, 80% of whom lived in Asian and African countries. 50% of the planet's population did not have access to clear drinking water and never saw a mobile phone.
Of course research and development continued. Bigger than ever groups of scientists worked on developing more efficient methods of producing energy. From the 2010s financial resources grew amply. Reducing energy use became an important factor in every invention's case. The notion of sustainable development became part of the educational curriculum of developed countries. The young generation of that period could already speak about the problem and saw its components. All that gave hope concerning the future of the planet.
They started to connect the results of different science fields from the 2040s so as to implement a system that could handle and solve the crisis effectively. Linking technological advances into a system was made possible by the development of information technology. Standard handling of distinct control systems required such great capacities that it would have been impossible under human control. They introduced control centers based on artificial intelligence capable of overviewing and controlling a whole city autonomously. Towns with a population of a couple of a hundred thousand became ideal subjects for optimizing energy use. Towns bigger than that showed too much of an energy-wasting scheme, while it was not economical enough to work a system like that in case of towns with fewer people. People started to move from big cities to smaller residential areas. Towns where people can live in and the SkyNet systems appeared.
The ideas and new methods worked so the planet was saved from the dangers imminent in 2017.
…

Statistics the introductory story is based on and a summary of today's facts can be found in Attachments 1 and 2.

CONCEPTUAL PRESENTATION OF THE SKYNET SYSTEM

The system below is a conceptual outline of the envisioned intelligent and energy-saving city. The created working model is shown in the following chapter.
Our conception is that the governance of a city is controlled by AIs (artificial intelligences) that operate computer systems completely autonomously. They collect and store data, they send controlling signals to the units and they make decisions based on data analysis - bearing in mind human safety as the most important factor (corresponding to Isaac Asimov's laws of robotics) and using energy sources more efficiently and more environmentally friendly. Of course not every system can be controlled by algorithms but for most of them computer control is viable. Each subsystem is controlled by a Slave AI while the whole system is coordinated by a Master AI.



The system makes use of the option of exclusion of AIs that is if someone wants to leave the network for personal reasons, they can do so provided they do not risk others' safety. (This fourth principle is an addition to Isaac Asimov's principles.)
The diagram above is an outline of a system scheme. The diagram is not complete: it only shows the most important systems that we also created in our model.
Arrows in the diagram represent two-way communication. Data as well as controlling signals run in the network. In case of a real system it is advisable to use a Wi-Fi connection because of the numerous related elements. We used Bluetooth technology, landline and infrared connection in our model.
If the system starts to work in reality, safety will be a very important issue. As there are data and control signals running in the system, it is crucial to protect it from outside access. It is necessary to restrict computer access to data for the sake of personal security. The hierarchic build-up of shared data bases appears complicated but it is essential to restrict the system so that each AI has access only to data necessary to make decisions.
This system would be capable of controlling the entire working of a city a lot more economically compared to today. Cost-effectiveness and energy saving would be results of

reducing costs with the help of Smart Home technology as opposed to wasting energy management by households due to individual regulation. Information to operate the heating/cooling systems, doors and windows of houses is provided by actual data collected from a weather station. We measure the intensity of solar radiation, the strength and direction of wind, temperature, air pressure and precipitation. Data get to the decision-making system online so the systems of the house can be controlled by sensors on the basis of data from the center.

There are further possibilities of automation in traffic. Electricity-driven cars are run by robot pilots. Passengers only have to give the coordinates of their destination and the car will automatically take them there. Traffic control is automatic too as the signaling system communicates with the robot pilot.

In case of goods transport truck convoys piloted by robots based on the principle of swarm intelligence save fuel by keeping an ideal distance. When trucks approach the town, a Master AI takes control over the convoy and leads the trucks safely to the unloading site in town traffic.

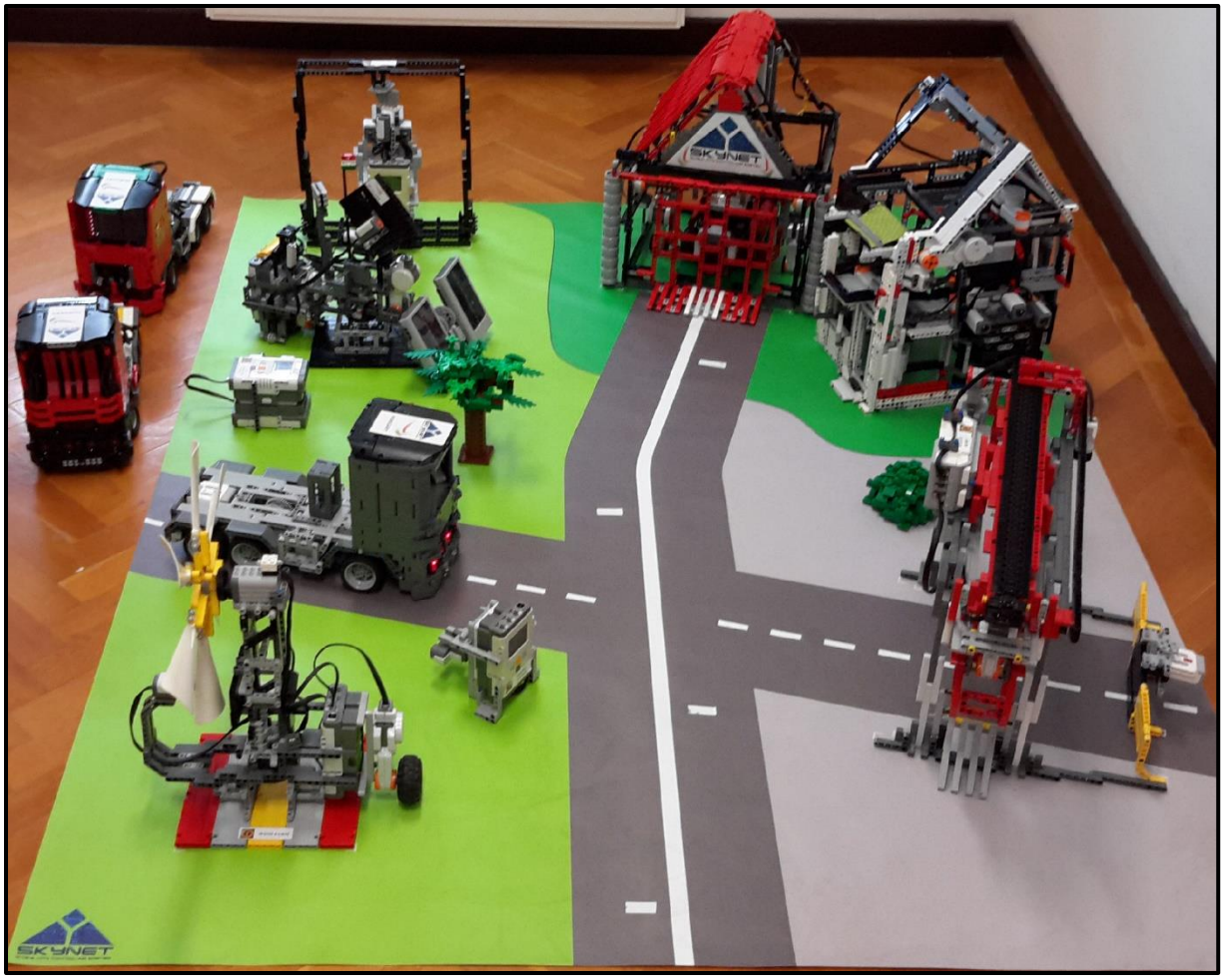
The entire energy production of the city is supplied by solar farms and wind farms (corresponding to geographical and climate conditions).

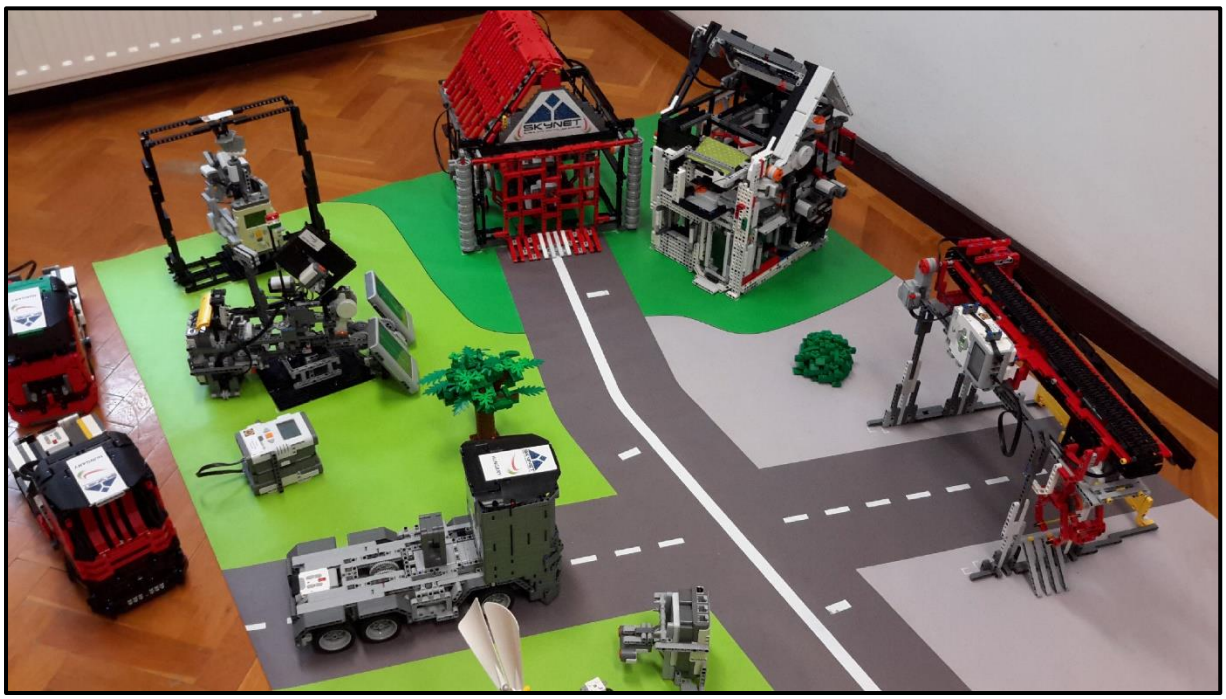
Solar panels suitable for producing further electricity, which can be directed or put in safety according to the data of the weather station, are placed on houses and public buildings. A house with such solar panels can produce enough electricity to recharge an electric car. In case of overproduction excess electricity can be diverted to the electricity network of the city.

In reality computer control is necessary for comprehensive, safe and efficient operation. The system can be automated but there are a lot of issues to work on before implementation. A few of these at present:

- Developing an AI program capable of controlling this complicated system.
- Development of legislation as background to the working of automated systems.
- Creating infrastructure (e.g. traffic signaling system, smart home technology, grey water collecting system, solar and wind farms to produce electricity).
- Increasing efficiency of solar panels.
- Retraining human resources required by demands in computer control.
- …

DEVELOPED MODEL OF THE SKYNET SYSTEM



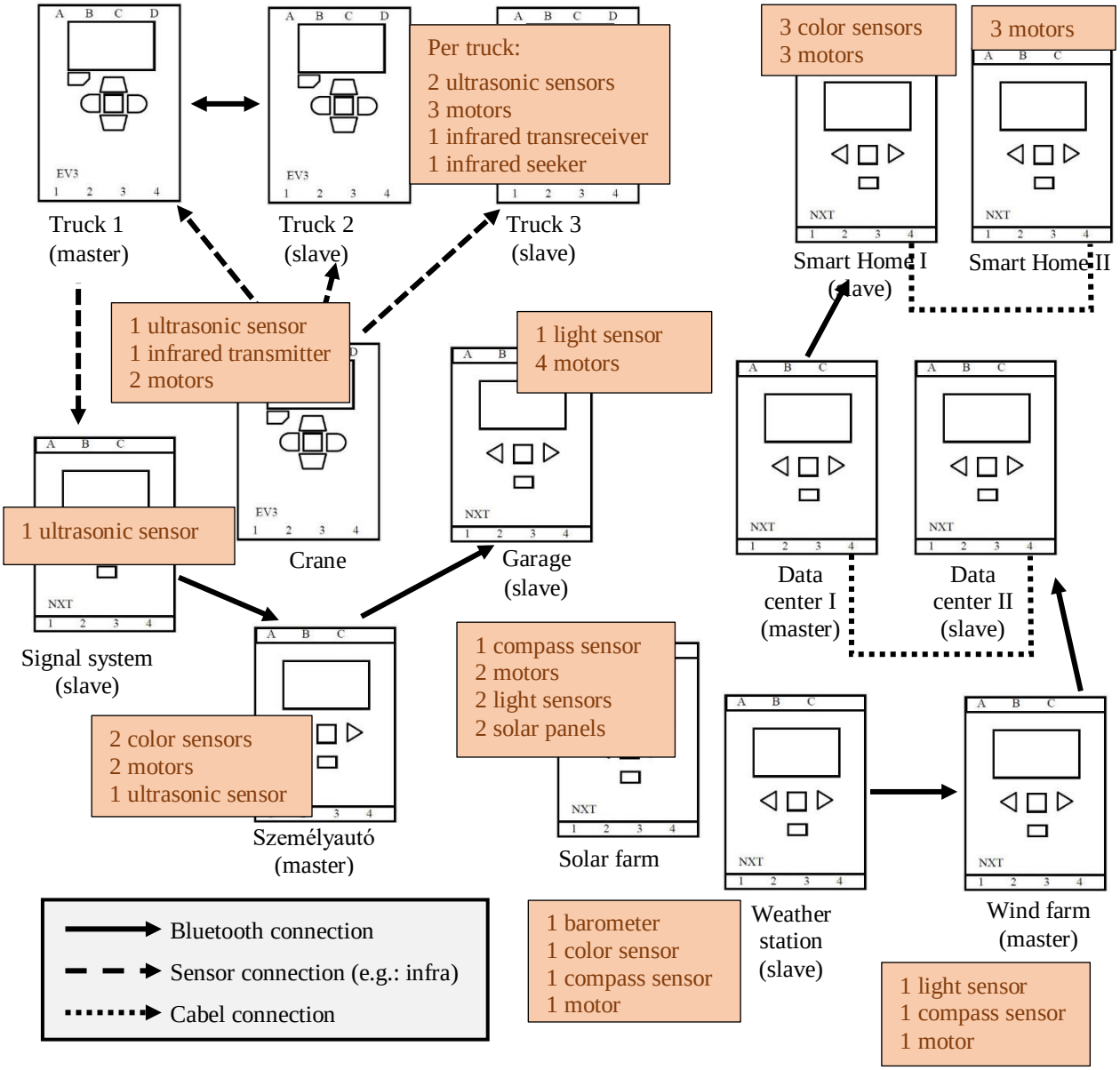


NETWORK CONNECTIONS OF NXT AND EV3 BLOCKS IN THE SYSTEM

In the developed system NXT and EV3 blocks together create the units connected to one another. The subsystems communicate with one another with the help of Bluetooth, landline connection or their sensors. The diagram below shows this network.

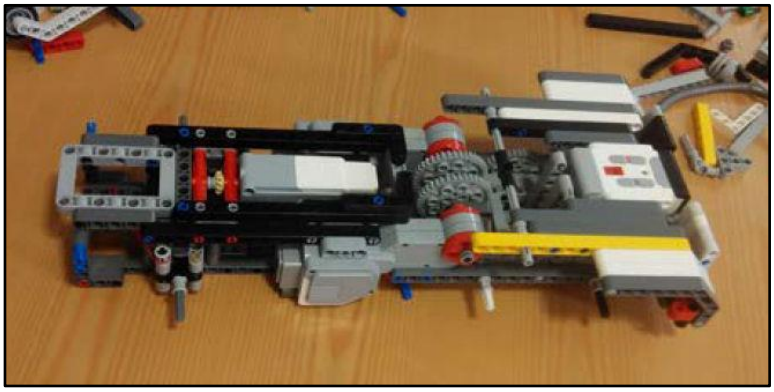
The system comprises 14 blocks altogether. Next to the blocks there are lists of corresponding connected sensors and motors. The construction is made up of altogether 27 motors (middle or large), 9 ultrasonic sensors, 10 color/light sensors, 3 compass sensors, 2 solar panels, 4 infrared transceivers and 3 infrared seekers.

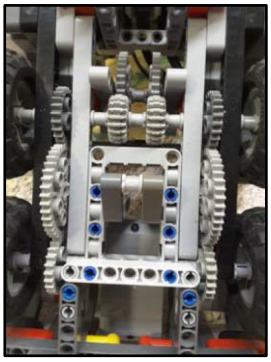
The NXT and EV3 systems are capable of communicating with each other only via a bridge (e.g. mobile application). This idea was not implemented in the model as recoding was too slow and reaction time too long.



SWARM INTELLIGENCE BASED TRUCK SYSTEM

The model includes a swarm truck system in which trucks can follow the one in front of them automatically, without human intervention thus creating swarms similar to the road trains common in Australia. Because it is not possible to have a human driver, we operate the first truck by remote control. While developing the trucks we tried to create as realistic ones as possible, so the constructions we made have robust steering systems. The rear axles B and C are responsible for driving. The chassis are identical, only the colors of cabs and spare parts are different. Electronic control is provided for by a LEGO MINDSTORMS EV3 set, two large motors are responsible for driving and the steering system is controlled by a middle motor. Each truck is equipped with two ultrasonic distance sensors, one infrared sensor and also one infrared transmitter at the back. Such a transmitter is used for remote control of trucks.

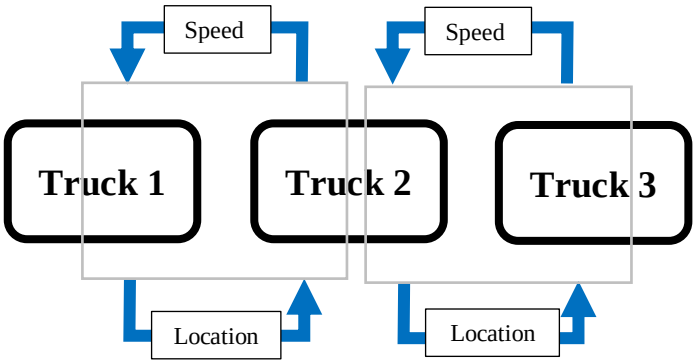




An infrared remote control controls the robot leading the swarm. This makes it possible to control speed and steering. The other trucks in the swarm primarily rely on the data obtained by their sensors. The two ultrasonic sensors measure the distance from and the infrared sensor detects the direction of the truck ahead. The robot determines the necessary speed based on the distance, and controls steering based on the direction given by the ultrasonic transmitter. It receives data necessary for precise control from connections to other robots.

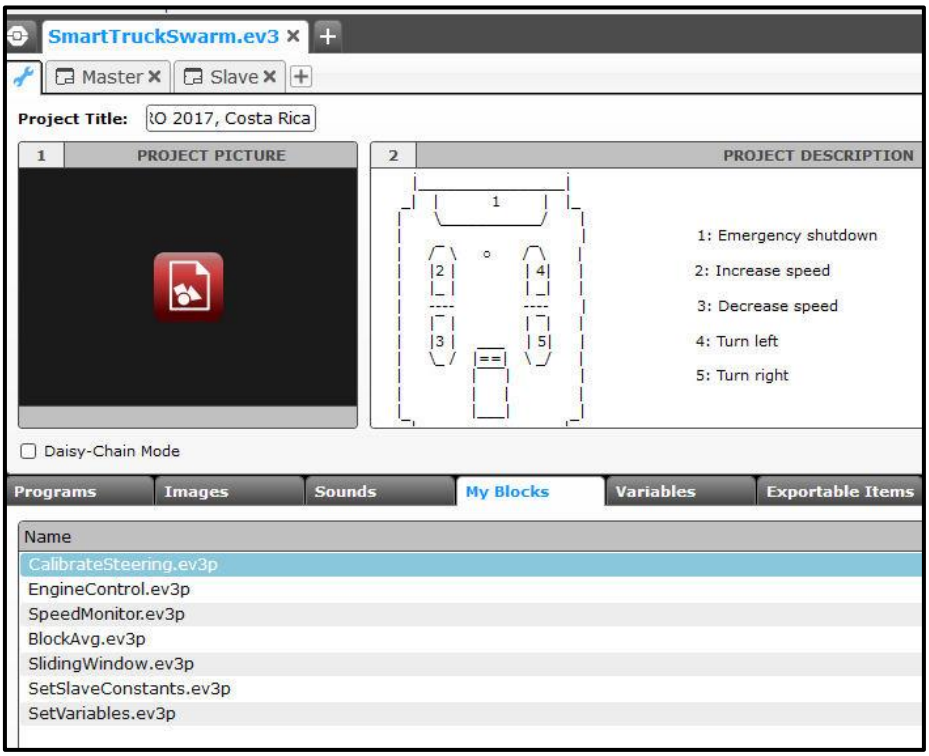
Adapting to EV3 robot capabilities, the model is based on Bluetooth communication. (The robot's system makes only master-slave based communication possible.)

Each robot communicates only with the truck in front and at the back which means that communication is fast: signals do not need intermediary transmitters and there is less risk of communication problems. This also makes it possible for a truck to join the convoy on the way because it only has to connect to two trucks. In addition, there is no risk of losing an important communication center: because there are no irreplaceable centers, there are no fatal errors. In our case communication has two important roles: it synchronizes the speeds of the trucks and informs the trucks about the situation of others.

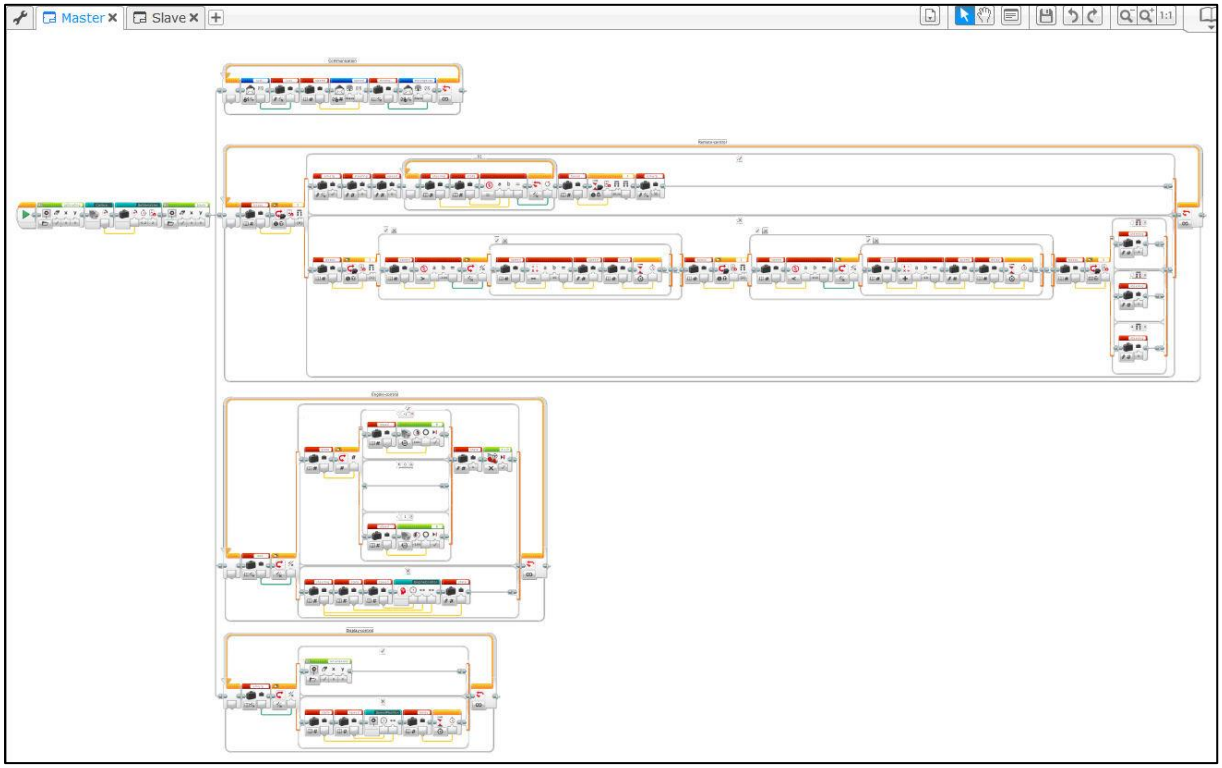


Every robot starting from the first one sends its speed to the following one which determines its own speed compared to that. If a truck is too close to the one ahead according to the data given by the distance sensor, it adjusts its speed to lower, if too far, then to higher respectively. There might be problems: if the infrared sensor loses the signal of the truck ahead, it sends a Stop request to it via Bluetooth. Then it will stop to wait for the lost one and sends on the signal to the following in front which means if one falls behind, the whole convoy will wait for it.

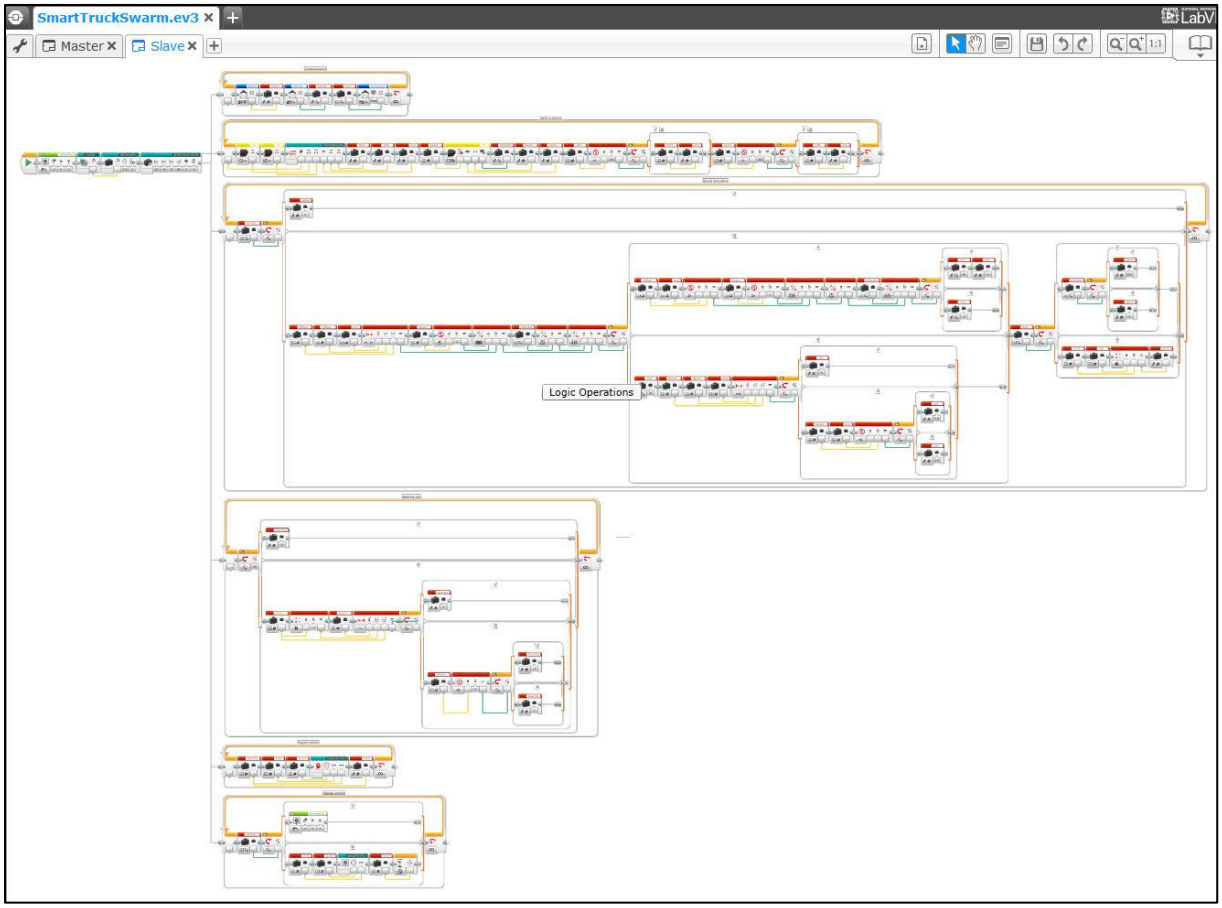
It is not possible to show the code in this document because of its size. It includes 7 My Blocks altogether. The picture below shows the project picture of the master and slave program and also the list of My Blocks.



Master program:



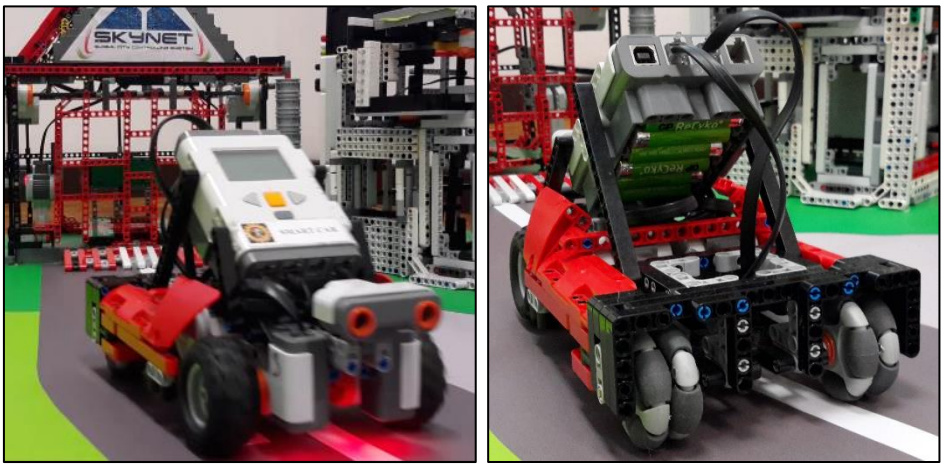
Slave program:



When the trucks get to a city, their control is taken over by the corresponding AI. In the model they wait outside the city and when they get the signal, they move on to the crane (following its infrared signal). Prior to that a calibration program straightens the truck's wheels and then it moves on following the crane's infrared signal. When it arrives at the crane, it stops and waits for its cargo to be unloaded. As passenger cars give way to trucks at the crossroads, it can travel without stopping.
After unloading it can join the convoy again and start for a new destination (this stage is not part of our project).

PERSONAL TRANSPORT MODEL

There is a car controlled by an NXT robot running on a track-based road in the model. The direction is shown by a grey lane, in the middle of which there is a 2cm white line. The important orientation points are represented by cross-shaped white lines. One is in front of the garage, another one is on two sides of the crossroads and a third one is at the end of the route. The program was written using character-based NXC language.



The car follows the white line with one of its two light sensors (S2), while the other senses the orientation points. For following we determine a *threshold g*lobal variable: the value below it is for the color of the road, the one above is for the white line. The variables *speedHigh* and *speedLow* determine the speeds of the two motors, which are different. If the light sensor senses white, then motor B rotates quicker, otherwise motor C. Thus the car follows the edge of the white line snaking along. This is handled by the *follow()* function in the program

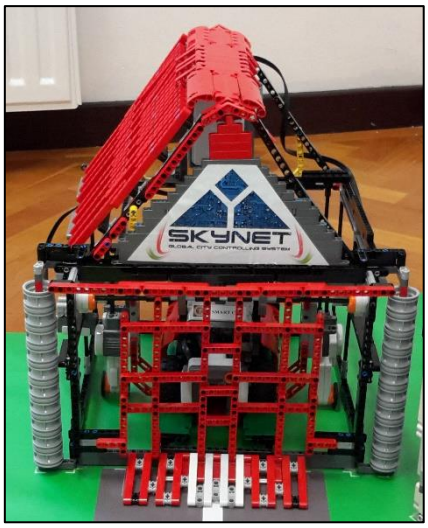
```
void follow(){
  while(Sensor(S1) < treshold){
    if(Sensor(S2) > treshold){
      OnFwd(OUT_B, speedHigh);
      OnFwd(OUT_C, speedLow);
    }else{
      OnFwd(OUT_C, speedHigh);
      OnFwd(OUT_B, speedLow);
    }
    NumOut(0, 0, Sensor(S1), true);
  }
  Off(OUT_BC);
}
```

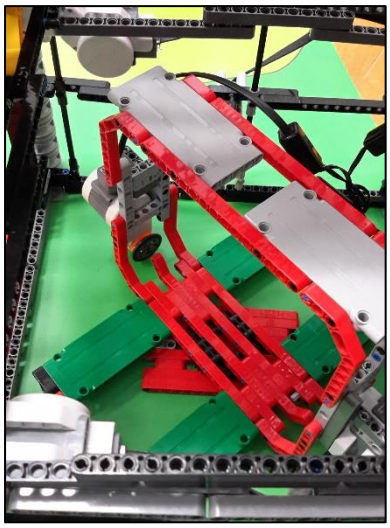
The car follows the line until its light sensor on the left senses white (S1), when the car stops and then it acts depending on its position.

If it is at the orientation point in front of the garage, it sends via Bluetooth a message to open the garage, and after moving in to close the door.

In the garage it gets onto a turntable which getting a Bluetooth message turns 180 degrees and so rotates the car opposite the door, which this way can move into and out of the garage without reverse mode.

```
void intogarage() {     //Into The garage
   SendRemoteNumber(1,1,1);   //Garage open
   Wait(2017);
   RotateMotor(OUT_BC,50,1300);
   OnFwd(OUT_BC,-50);
   Wait(100);
   Off(OUT_BC);
   SendRemoteNumber(1,1,1);   //Garage close
   Wait(500);
   SendRemoteNumber(1,1,2);   //Turn the car
   Wait(10000);
}
```





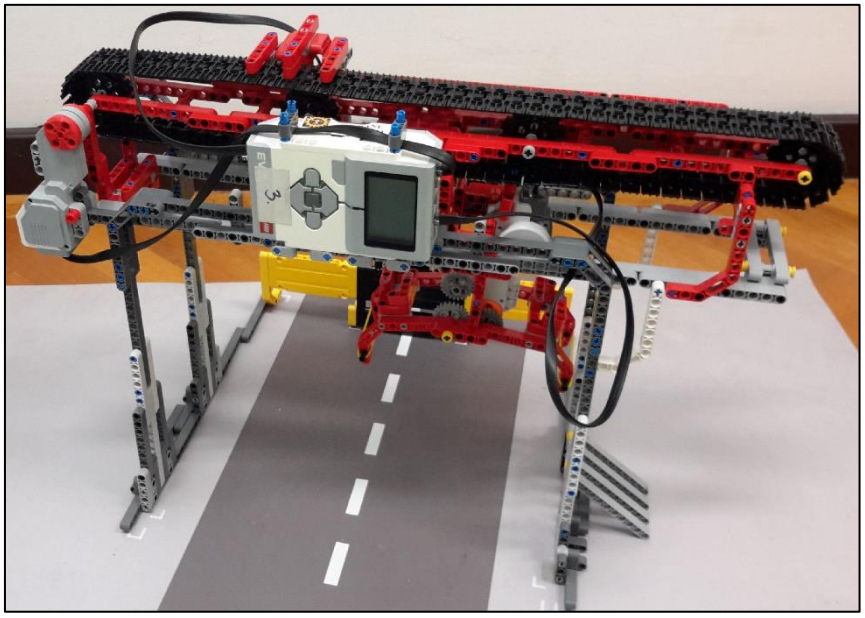
If the car reaches the crossroads, it connects to the signaling system and asks for its status through Bluetooth. If the value is 1, it carries on moving, if 0, it stops. The signal system observes the traffic at the crossroads and if there is a vehicle approaching from the crossing direction, it sends a stop signal (0). The car continues asking for data until it gets a signal to start: then it moves on for a period of time given in milliseconds shown in the variable *ms* to get across the crossing safely. The code is contained in the *waitForTheLamp()* function.

```
void waitForTheLamp(int ms) {
   temp = 0;
   while(temp == 0){
     ReceiveRemoteNumber(1,true,temp);
  }
   RotateMotor(OUT_BC,50,ms);
}      Off(OUT_BC);
}
```

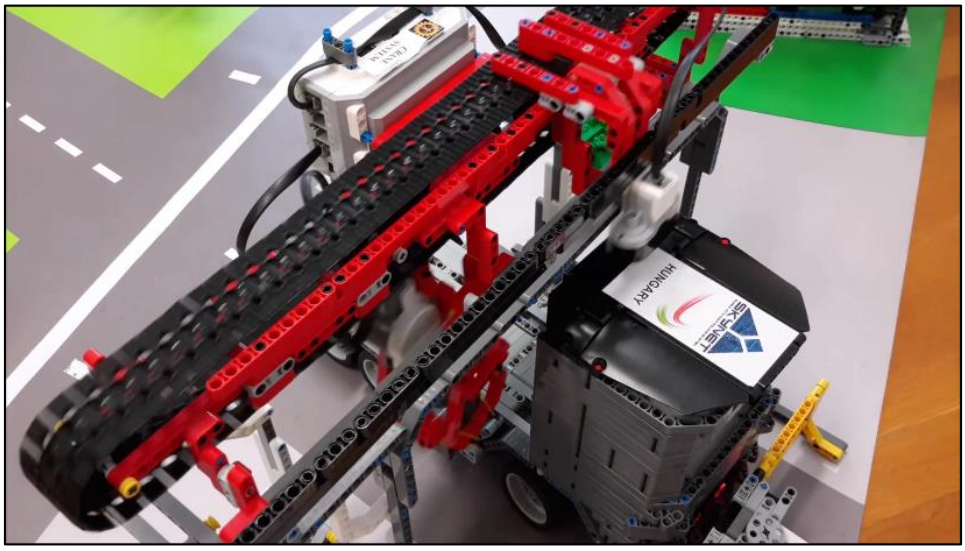
So the car communicates with the NXT block of the traffic lights and the garage via Bluetooth. In this relation the car is the master and automatically builds connections with the two slaves when starting the program.

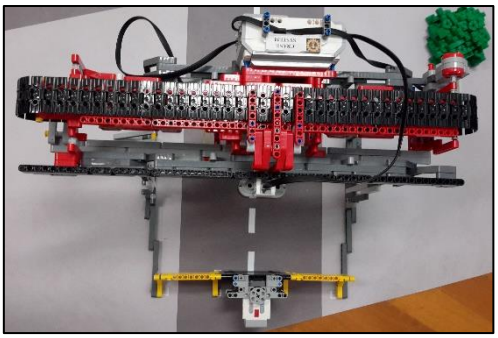
In the *main()*function of the algorithm controlling the car one can only find the timed function calls and the turn at the end of the route. (See Attachment 3)

TRUCKING MODEL, CRANE

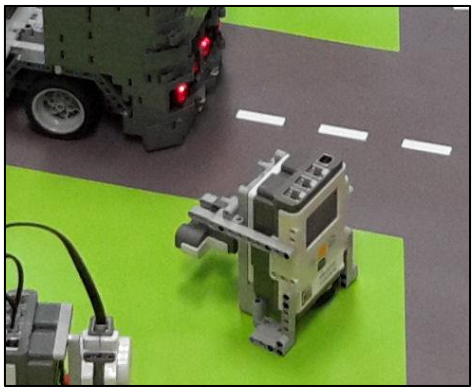


According to the conception of the city, automation of trucking includes directing robot controlled trucks to their destination as well as unloading cargo. The latter task is completed automatically by our crane construction. The truck follows the signal of the infrared transmitter on the crane and stops in an adequate position. When the crane perceives this with its ultrasonic sensor, it starts unloading the cargo. First it calibrates the position of the grabs (positioning it to the edge of the crane), then sensing the edge of the truck with an ultrasonic sensor it grabs the cargo and unloads it from the truck. Next it moves back to its starting position and waits for the next truck. The truck then can move on.





TRAFFIC CONTROL, LOGISTIC MODEL

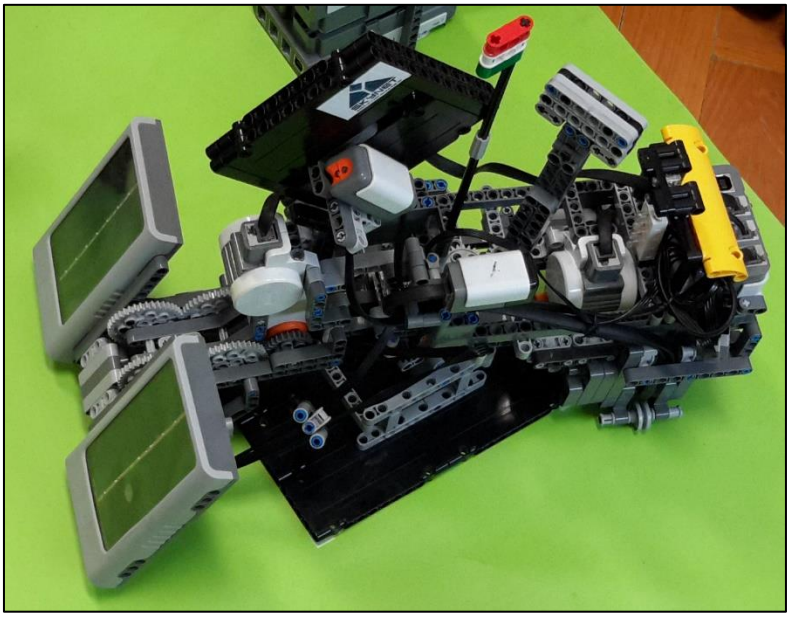


The traffic in the city is controlled by a "traffic lights" system which sends continuous go or stop signals to the cars in the crossroads. In reality timing is necessary just like in case of present-day traffic management. However, it is not needed to send set-time go and stop signals: depending on how busy the road is, it is possible to let a vehicle from a certain direction through just for a period that is necessary for it to go through. If the crossroad is free, the signaling system reacts automatically when a vehicle arrives. For this we need a monitoring system that can perceive the arrival of vehicles. There is a crossroad in the model. The trucks move along the main road (as braking and starting in their case requires more energy). So they have a continuous go signal (this is only relevant in the model). A car arrives on the side road and either stops and waits or carries on moving depending on the signal it receives from the signaling system.

```
task main(){
   SetSensorLowspeed(S2);
//Truck distance from the crossroad
   int distance = 30;
   while(true){
      if(SensorUS(S2) < distance){
         SendResponseNumber(1,0);
         NumOut(0,0,0,true);
      }else{
         SendResponseNumber(1,1);
         NumOut(0,0,1,false);
      }
      NumOut(0,10,SensorUS(S2),false);
      Wait(300);
   }
}
```

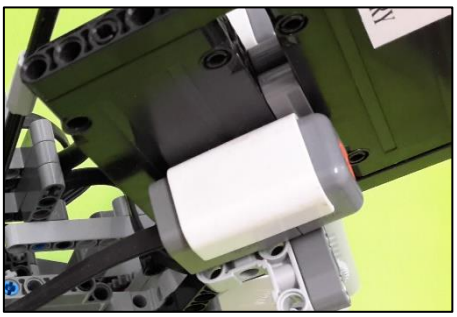
The program of the signaling system was written in simple character-based NXC language. The gate observes the main road with an ultrasonic sensor. If a truck arrives, it sends signal 0 through Bluetooth, otherwise signal 1. The car, perceiving this, either waits or goes on. The program runs in an infinite loop so signal sending is continuous.

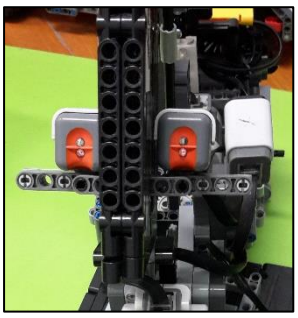
SOLAR FARM MODEL



We simulate conversion of solar energy into electricity with the model of a solar farm. The unit makes use of an NXT central element. Two solar panels follow the sun's movement both vertically and horizontally. When starting it finds south with the help of a compass sensor and adjusts the solar panels corresponding the orientation. After that according to the principle of real sun following solar panels it always turns the panels in the brightest direction so as to produce energy in the most efficient possible way. It works like this: there is a light sensor on both sides of a vertical panel, one of which is in shade during the sun's movement and the two sensors measure different light intensity. The algorithm turns the construction towards the brighter sensor until the two values are the same again. In reality solar panels do not need to be turned around completely as the sun's virtual movement in the sky only happens in a limited range.

```
while (true){       //Follow the light
        LeftLight=Sensor(IN_2);
        RightLight=Sensor(IN_3);
        if (abs(LeftLight-RightLight)>3){
           if (LeftLight>RightLight){
              OnFwd(OUT_C,-50);
           }
           else{
              OnFwd(OUT_C,50);
           }
        }
        else{
            Off(OUT_C);
        }
   }}
```

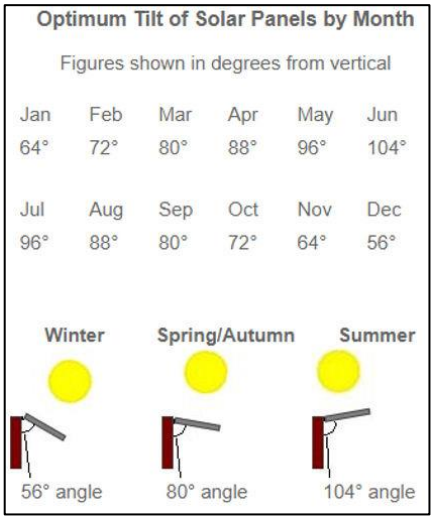




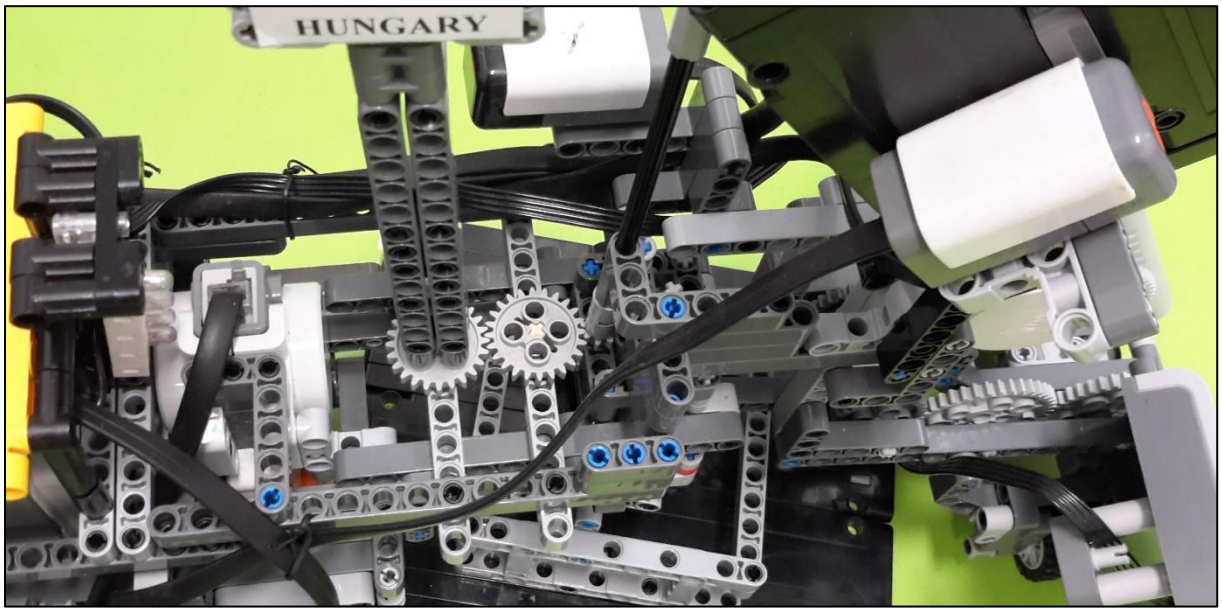
In different months the sun is at different heights above the horizon. Our model is capable of adapting to this as the most efficient setting is when sunrays reach the panels perpendicularly. To save energy we change the gradient of the panels monthly. In the model one month is 5 seconds. The calculated gradient values are optimized for Costa Rica.

```
task SunDirection(){   //Tilt of Solar panel
   int i=0, direction=1, Month=5000;
   while (true){
      RotateMotor(OUT_A,direction*20,14);
      Wait(Month);
      i++;
      if (i==6){
         direction*=-1;
         i=0;
      }
   }
```

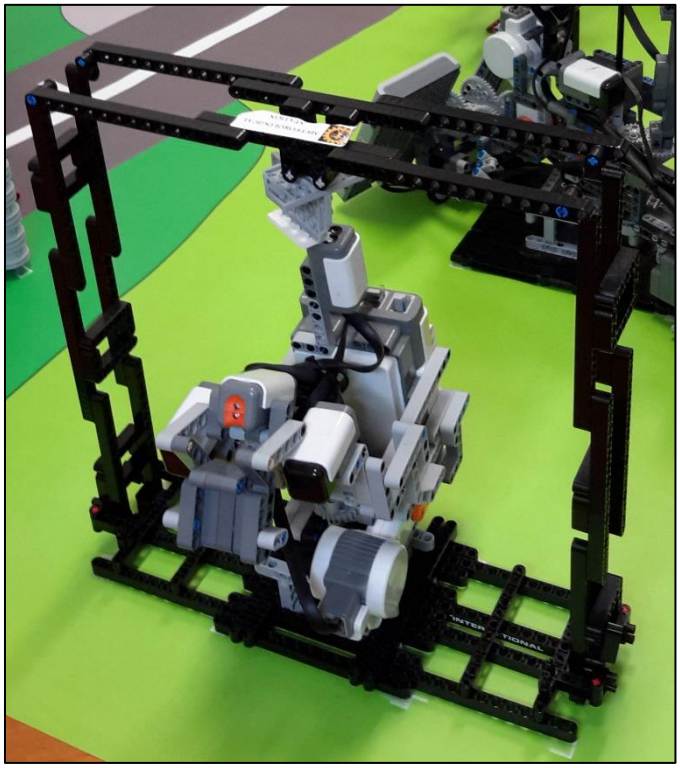
Costa Rica, San José
Time zone: GMT -6
Longitude: 09° 55’ 30” N
Latitude: 084° 05’ 00” W



The two-axle rotation of the structure is ensured by two motors. East-west rotation is done by rotation round a fixed axle, while adjusting the gradients of the panels is helped by cogs.



WEATHER STATION



The weather station is to measure different parameters of weather, which provides data for the smart home to automatically control its heating/cooling system and the working of its windows and doors and electric light sources. The data get to a data transfer center which sends the relevant data to the different subsystems of the model via Bluetooth.

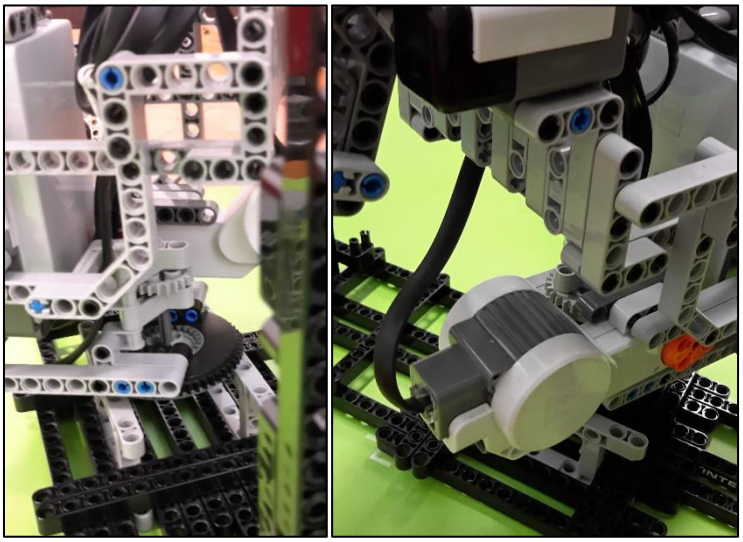
There are four sensors at the station. A barometric sensor measures the temperature and air pressure. A light sensor measures the light in the surroundings to decide whether it is day or night, whether it is cloudy or sunny. The weathervane on the construction can define the direction of the wind with the help of a color sensor. Sensing the move of the vane the construction rotates in the direction of the wind; the precise direction is measured with a compass sensor that is also used e.g. for turning the windmill in the wind's direction through the data transfer center.

The weather station collects the measured data into a string, thus being able to send all of it in one step via Bluetooth. The target

```
while (true){
      ReadSensorHTBarometric(IN_3, temp, press);
      temp/=10;               //Temperature
      press*=0.0254;          //Press
      do {                    //Direction
         direction1=SensorHTCompass(IN_1);
         Wait(300);
         direction2=SensorHTCompass(IN_1);
         Wait(300);
      } while (direction1!=direction2);
      if (direction1<10)
          dirstr="00"+NumToStr(direction1);
      else if (direction1<100)
               dir="0"+NumToStr(direction1);
           else dir=NumToStr(direction1);
      light=Sensor(IN_2);     //Light
      if (light<10)
          lightstr="0"+NumToStr(light);
      else
          lightstr=NumToStr(light);
      message=NumToStr(temp)+NumToStr(press)+dir+lig;
      if (Sensor(IN_4)!=6) Rotate();
   }
```

computer receives the string, and cutting and converting it, the computer gets the data.

Moving the construction happens by using a train attached to a fixed axle. Thus it is possible for the construction to turn around without the frame's movement or the cables getting tangled.



The weathervane is a construction attached to the frame, which can easily turn. It can be moved by wind and the construction turns into the direction with the smallest resistance which corresponds to the direction of the wind. A color sensor perceives this move, and the whole structure is rotated respectively.

The weather station is a slave in the Bluetooth network, while the wind farm is the master.

DATA TRANSFER CENTER

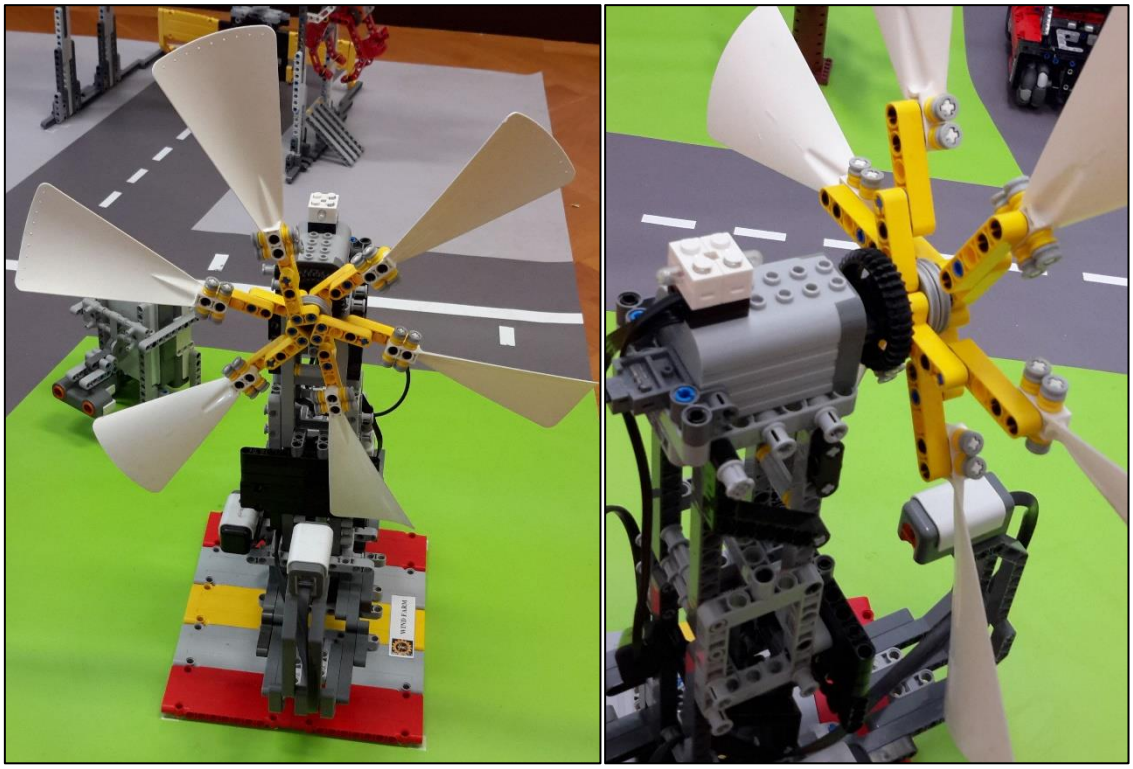
The data transfer center consists of two NXT bricks connected at I^2C ports. The so-called dual NXTs are capable of connecting two Bluetooth-based subsystems. Both bricks can connect to three further NXTs; thanks to the cable connections between them, all the three Bluetooth connections remain free, thus making it possible to create a system of up to eight bricks. One brick of the data transfer center connects as a slave to the wind farm (which is the master), and through that to the weather station. The data transfer center gets the text message including the value of wind speed added by the wind farm to the data it received from the weather station. The center displays the data on the screen and also sends it to the connected other NXT through the I^2C port. This second brick as a master makes up another subsystem which is in connection with the smart home and sends data to it.

The data on the screen is updated every half second. Converting text data and showing it on the screen happens e.g. in case of wind according to the following code:

```
TextOut(0,32,"Wind pow.: ",0);          //wind power
rpm=StrToNum(SubStr(message,10,1));
NumOut(60,32,rpm*20,0);
TextOut(80,32,"rpm",0);
```

In case of other data the code is similar but the data can be found at a different place of the string, so the substring can be obtained from the text with other parameters.

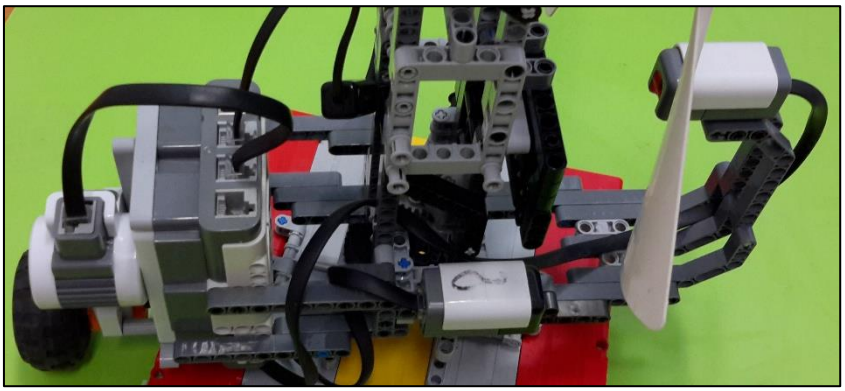
WIND FARM

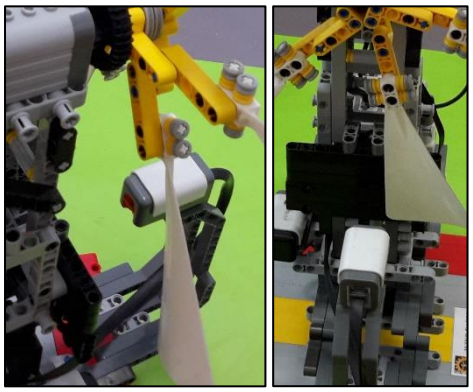


The wind farm is the master robot of the data collecting subsystem. When starting it automatically attaches the weather station and one of the data transfer center's bricks to the network. It can rotate around a fixed axle so that its blades are always in the actual direction of the wind. It receives the direction of the wind from the weather station and it uses a compass sensor to define the direction.
It has a light sensor to measure wind. It adds the measured value to the data received from the weather station and sends the data to the data collecting center via Bluetooth. It measures the value of strength of the wind in rpm (rotations per minute).

So a light sensor measures the wind. The blades of the windmill are white and they rotate in front of a black surface. When the wind blows, the light sensor perceives the variation of black and white colors, while in case of no wind this value is constantly the same (either white or black). The variation speed of the measured value is used by the program to create an *rpm* value which is in proportion with the speed of rotation. This value is added to the end of the data string received from the weather station and is sent to the data transfer center.

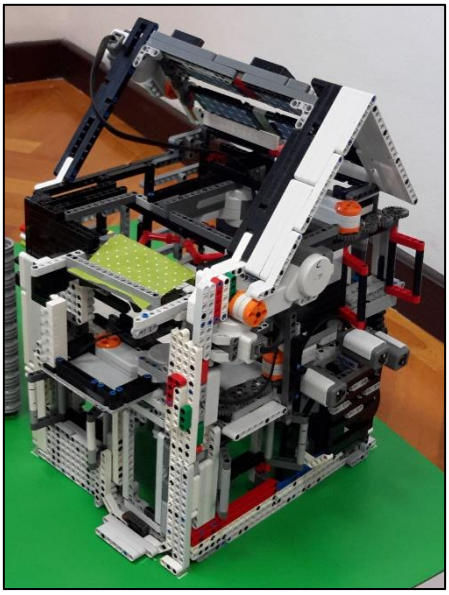
```
while(true){
      NumOut(0,50,rpm,0);
      if (rpm!=old){
         old=rpm;
      }
      if (rpm==0) old=-1;
      whitecount = 0;
      kezd = CurrentTick();
      while(CurrentTick() - kezd < 500){
         if(Sensor(IN_1) < 50 && isOnWhite){
            isOnWhite = false;
         }
         if(Sensor(IN_1) > 50 && !isOnWhite){
            isOnWhite = true;
            whitecount++;
         }
      }
      rpm = whitecount;
   }
```



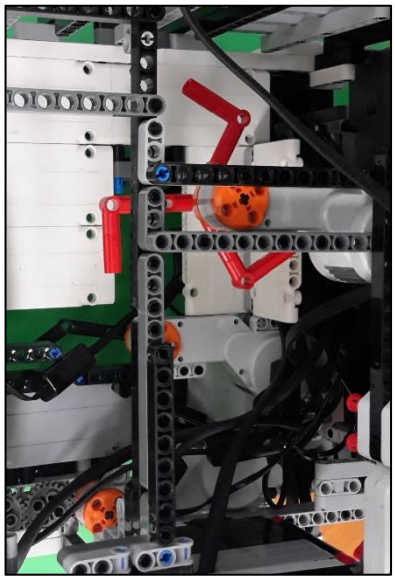


On top of the wind farm there is a led string which flashes continuously. In a real situation this informs air traffic of an actual height.

SMART HOME



The central unit of the smart home consists of two NXT bricks which are connected with cables via I^2C ports. One brick connects as a slave to the data transfer system from which it gets the data collected by the weather station.
What is new about this system is that in case of implementation at city level the smart homes have access to a single data system so there is no need for individual measuring solutions. The data provided by the weather station will serve as basis for working.
The house can open and close its windows and doors and also control its heating-cooling system. The cooling system is represented by a ventilator while heating is modelled by color sensors placed outside the house. Red light means high, green normal and blue low temperature. Of course in reality all this can be replaced by air-conditioning. Lights are modelled by the led string on the house. There are solar panels on the roof which in a real situation provide energy enough to recharge an electric car. The solar panels are rotatable so in sunny weather they can produce energy, but in dark or in stormy weather they can be rotated to face their surroundings with a surface that prevents the leaking of heat.



So the working of the systems of the house is influenced by temperature, wind, and the intensity of sunshine. There is part of a code below demonstrating the reaction of the house to different values.

```
void Operation(string msg) {
   switch(msg) {
   //Wind-Temp-Light   window-roof-heatleds-fan  heatled: 1-red; 2-green; 3-blue
      case "111" :        OnOff(0,0,1,1); break;
      case "112" :        OnOff(0,1,1,1); break;
      case "121" :        OnOff(0,0,3,0); break;
      case "122" :        OnOff(0,1,3,0); break;
      case "191" :        OnOff(0,0,2,0); break;
      case "192" :        OnOff(0,1,2,0); break;
      case "211" :        OnOff(0,0,1,1); break;
      case "212" :        OnOff(0,1,1,1); break;
      case "221" :        OnOff(1,0,3,0); break;
      case "222" :        OnOff(1,1,3,0); break;
      case "291" :        OnOff(1,0,2,0); break;
      case "292" :        OnOff(1,1,2,0); break;
   }
}
```

The *msg* string contains data referring to wind, temperature and light. If the first character is 2, there is no wind, if it is 1, there is. If the second character is 1, it is hot, if it is 2, it is cold and if it is 9, the temperature is normal. If the third character is 1, it is dark, otherwise it is light. The windows and doors work as the code above shows: if the wind blows (1), they are closed, if it does not blow but it is hot (1), they are also closed (as the air-conditioner turns on). By changing the values that are arranged like in a matrix, it is possible to customize the working of the systems of the house, based on the received data.

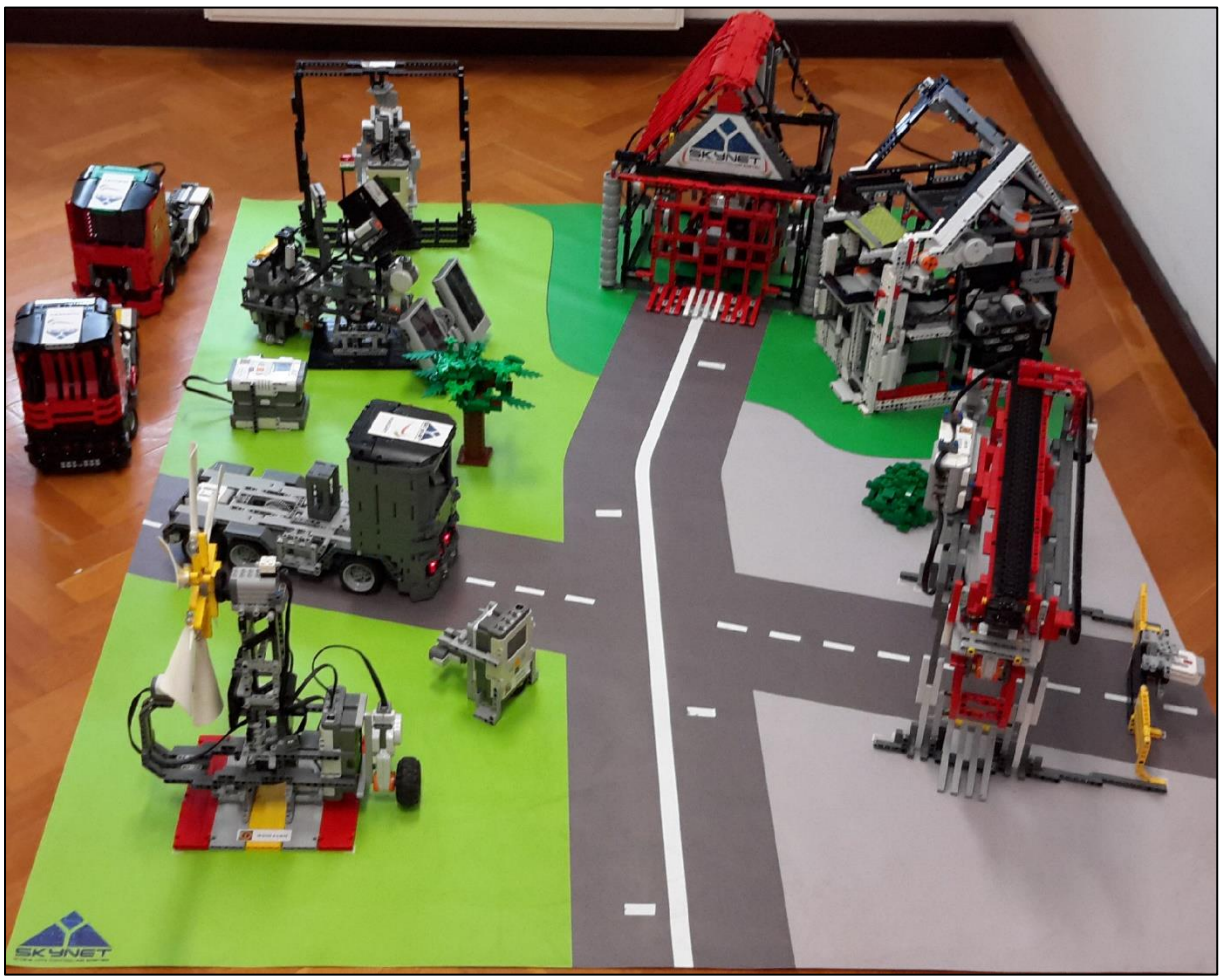
The picture shows a part of the code responsible for opening and closing the windows. The *wstate* logical variable contains the actual position of the window as working the motors used for opening-closing depends on whether the window is actually open or closed.

```
//window
   if(window!=wstate) {
      if(wstate==1){  //close
        OnFwd(OUT_A,-50); Wait(500); Off(OUT_A);
        wstate=1;
      }
      else {  //open
        OnFwd(OUT_A,50); Wait(500); Off(OUT_A);
          wstate=0;
      }
   }
```

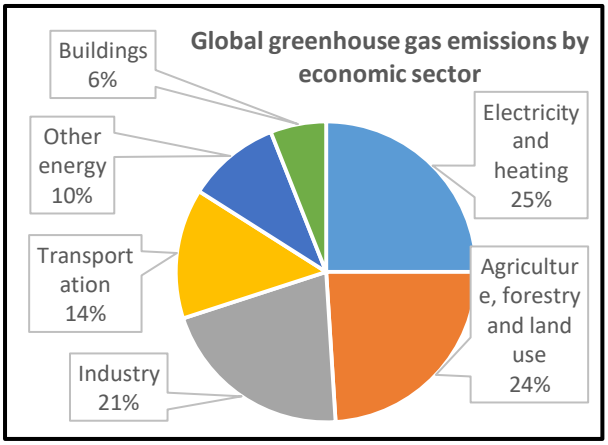
The second brick of the smart home controls the solar panels on the roof and the lights. It was necessary to involve an NXT robot because of the large number of motors.
The automated functions of the house can be further developed by supplementing the sensor set of the weather station, e.g. with a rain sensor. We could produce a rain sensor by using the technique applied with the windscreen of a car, or the idea that light reflects from a wet glass surface in a different way than from a dry surface.

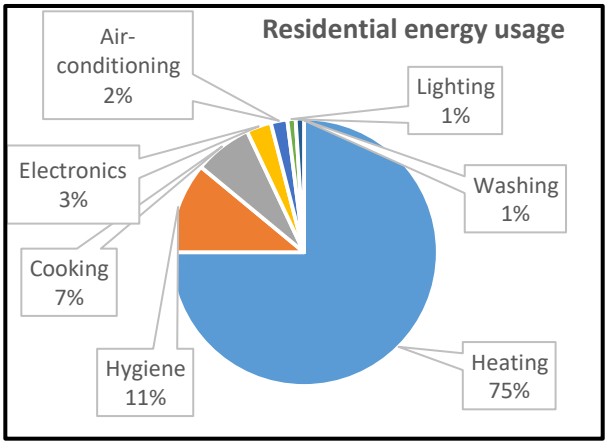
CONCLUSION

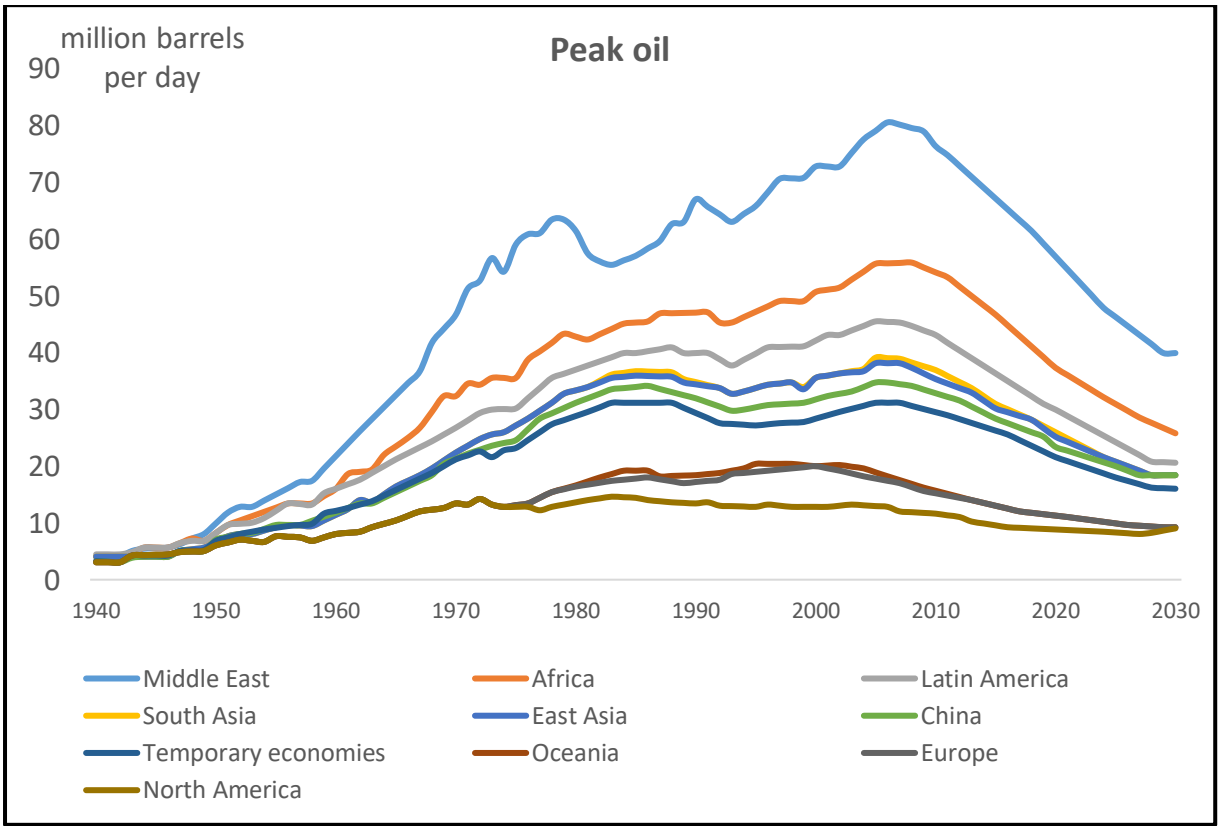
The modelled system works. This supports the idea that a global city controlling system can be implemented in reality too: a system which handles constructions that can be automated in an algorithmic and single way. All this is not solely for the purpose of comfort in terms of e.g. self-driven cars or smart homes, but it can also result in saving energy thanks to the central sources (for instance data collected and handled in a single way) used instead of numerous individual solutions. A further advantage of this system is its cost-effectiveness which does not only make smart homes and environmentally friendly cars available for everyone but also sustainable.
Renewable energy sources can primarily be used efficiently in case of electricity. True, solar panels are not really efficient yet and it is not so simple to store electricity as fossil energy sources. Tendencies show though that the future is electricity. And automated truck convoys have already proved to be able to reduce energy consumption.

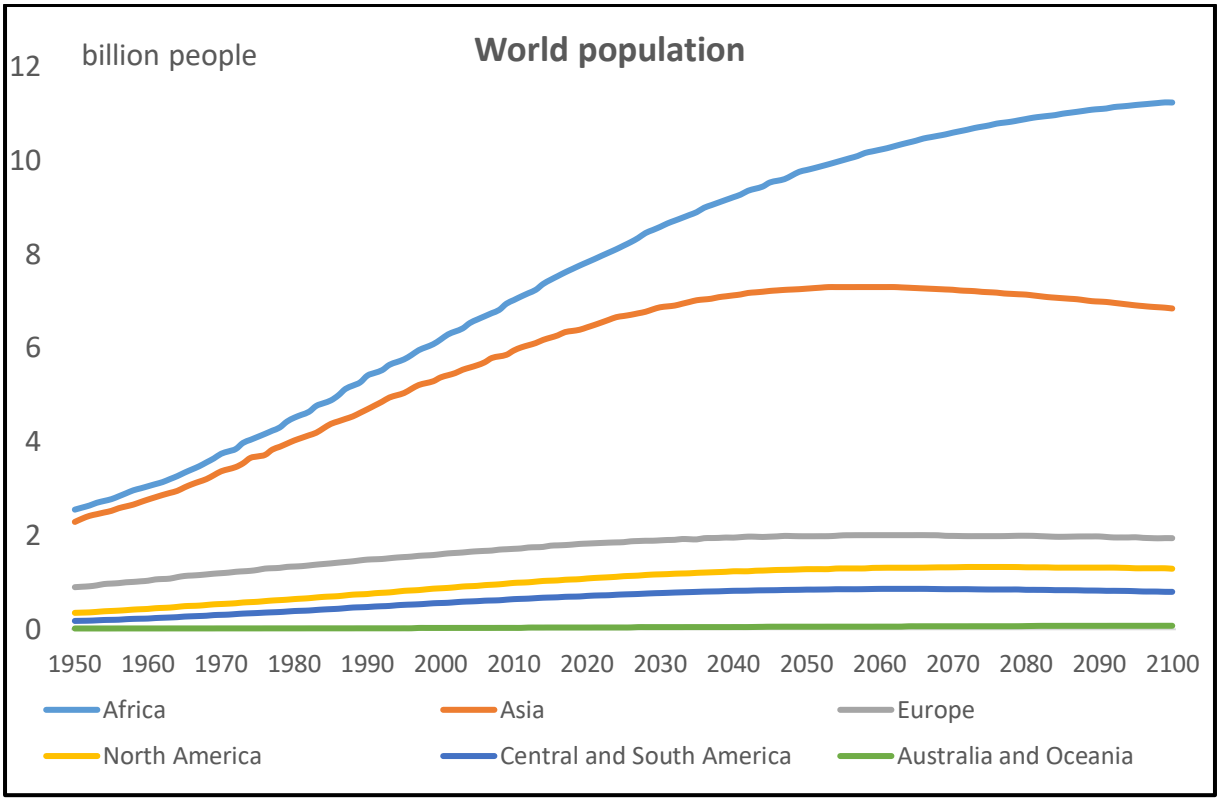


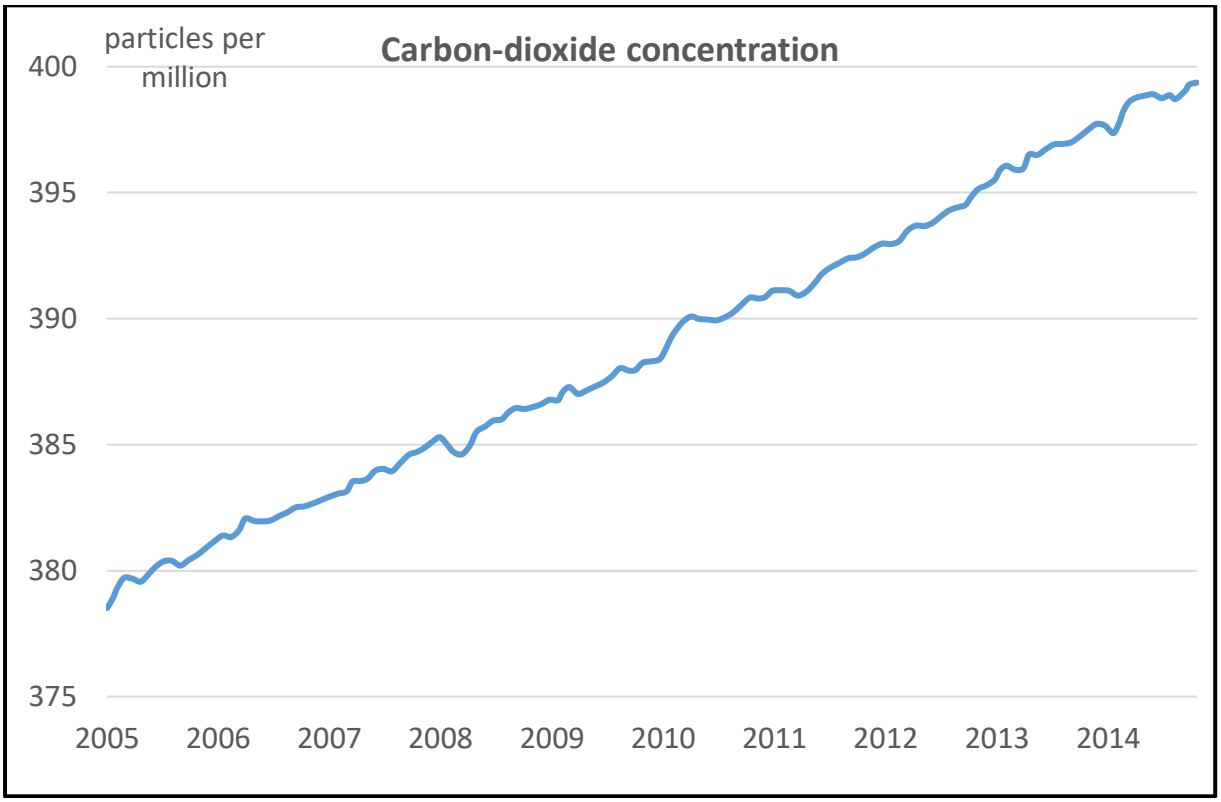
ATTACHMENT 1 - STATISTICAL BACKGROUND OF THE INTRODUCTORY STORY

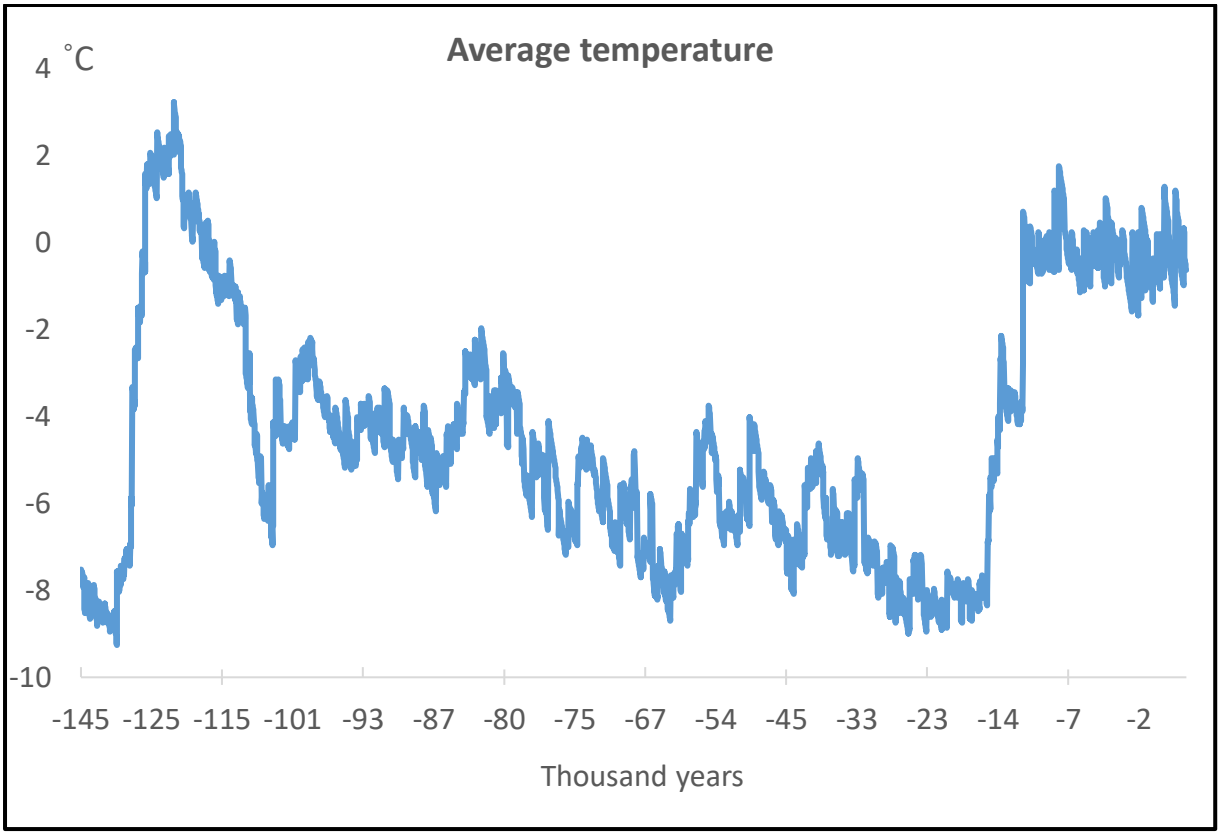


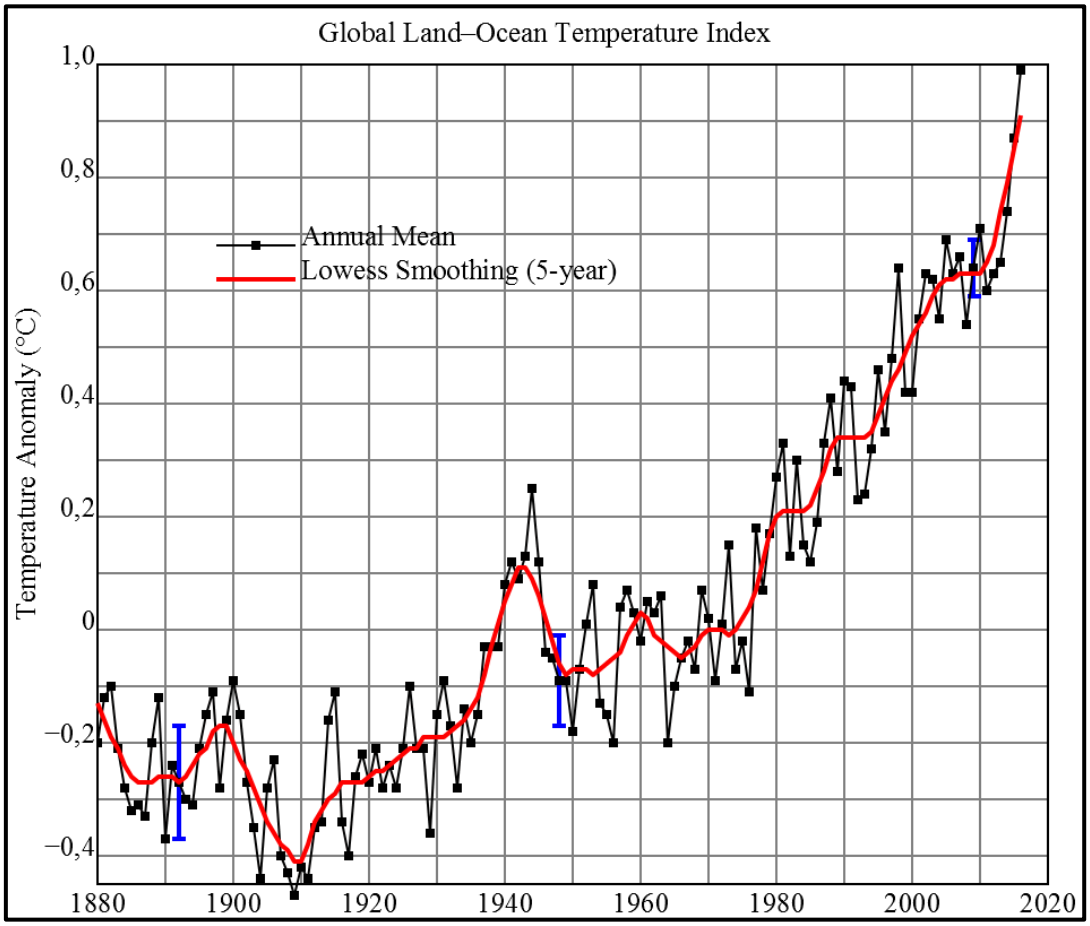


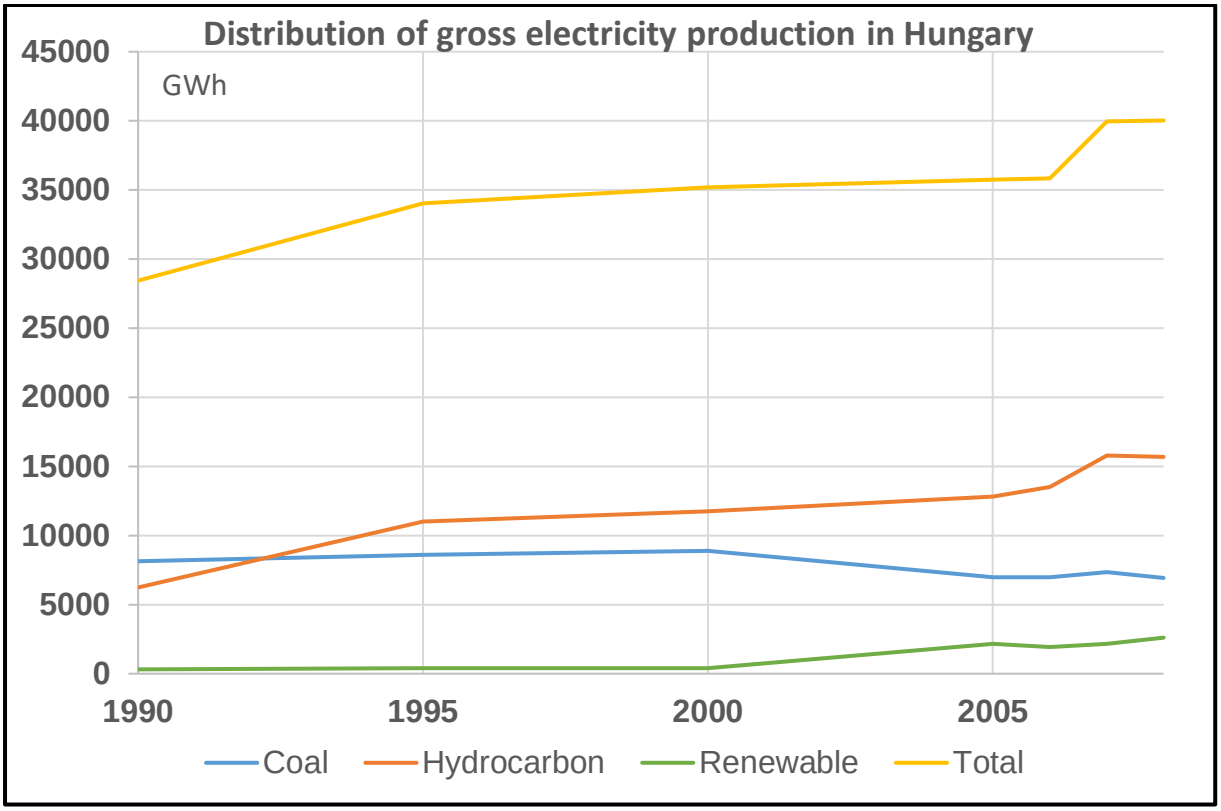



World population
billion people
12
10
8
6
4
2
0
1950 1960 1970 1980 1990 2000 2010 2020 2030 2040 2050 2060 2070 2080 2090 2100
Africa
Asia
Europe
North America
Central and South America
Australia and Oceania


Carbon-dioxide concentration
particles per million
400
395
390
385
380
375
2005 2006 2007 2008 2009 2010 2011 2012 2013 2014


Average temperature
°C
4
2
0
-2
-4
-6
-8
-10
-145 -125 -115 -101 -93 -87 -80 -75 -67 -54 -45 -33 -23 -14 -7 -2
Thousand years


Global Land–Ocean Temperature Index
Temperature Anomaly (°C)
1,0
0,8
0,6
0,4
0,2
0
−0,2
−0,4
Annual Mean
Lowess Smoothing (5-year)
1880
1900
1920
1940
1960
1980
2000
2020



Data sources:

http://fna.hu/vilagfigyelo/olajcsucs
http://fna.hu/vilagfigyelo/eghajlatvaltozas
https://www.ksh.hu/interaktiv/grafikonok/vilag_nepessege.html
https://www.epa.gov/ghgemissions/global-greenhouse-gas-emissions-data
http://kkft.bme.hu/sites/default/files

ATTACHMENT 2 – A SUSTAINABLE MODEL OF ENERGY USE AND TRAFFIC

The sustainable town model demonstrates a community working in a lot more environmentally-friendly and cost-effective way compared to traditional towns. The aim of creating such a city is to protect our planet, to use our energy resources efficiently, to make our society more environment-conscious and to minimize expenses. There are several ways to reach these goals but by all means we have to make alterations in cities and form a system capable of handling the problems arising while controlling the town and of processing and making use of the incoming data. Let's automate traffic (both public and personal transport) and freight transport, let's build Smart Homes and let's provide for energy use with renewable and sustainable resources instead of traditional energy sources!

In an ideal town vehicles are electric and hybrid and also self-driving. They are not widely-spread yet but they are legal to use in e.g. two US states. There is a great need for them as traffic is responsible for 17% of air pollution. There are quite a few misconceptions about these cars regarding fuel consumption, bearing capacity and working life. Reality and experience show, though, that electric cars can run 120-130 km on average after refill, and the best Tesla-made cars can even go as far as 300-400 km. Surveys show that because in these cars there are a lot fewer spare parts that tend to go wrong, they are in a much better condition after a few years' use than cars with internal combustion engines. Some say that electric cars are not environmental-friendly as producing batteries causes more pollution but this argument has little truth in it: these batteries do not emit any harmful gases and also lithium-ion batteries are comparatively environment-friendly. Among the major advantages of these vehicles is that they can accelerate surprisingly fast, that they recharge brake energy into the battery and also they have great turning stability thanks to their low center of gravity. A common problem is that they are very expensive, though it has dropped to 8 million HUF on average, and that many buy a second-hand car. As a result of different reductions and free recharges, it takes only about five years to recover the price of the car. Although there are free charging stations in quite a few towns by now (in Budapest 50 at present), these cars can be recharged at home with the help of a household plug. Recharging usually takes about 6 to 8 hours (depending on the type of the charger and the extent of charging) but there are rapid or fast chargers as well. Fast chargers can recharge the battery to 80% in about 30 minutes. Google cars are the best-known self-driving cars. Their development is a continuous success. There was only one accident during the test drives and even in that case it was the driver of the other vehicle who was responsible. However, these cars still need a driver because they are not "man-like" enough: they can only react as robots. Human intervention is indispensable in certain situations. They are planning to introduce completely self-driving cars in 2019. As to public transport, definite progress would be made by direct communication between vehicles and also complete automation. All these would lead to a considerable reduction of harmful exhaust emissions and at the same time we could save energy.

Trucking is a great challenge even for big companies. The expenses are high, there is a risk for goods to be damaged and also trucks can get involved in accidents. Fuel-loss is an even bigger problem than those mentioned before. That is why trucks often travel in convoys: they can go faster in each other's lee. However, this often causes accidents because the distances between the vehicles are shorter and so the drivers have a shorter time to react. This problem too can be solved by using robot pilots. The vehicles communicate directly and their reaction time is a lot (circa 14 times) shorter than man's, so there are no accidents to be feared. This method makes transport more cost-effective, safer and environmentally friendly. Our school has already done a research based on this idea (Smart Truck Swarm), but practice shows there are existing models already. Mercedes has successfully experimented with sending a convoy from Stuttgart to

Rotterdam. The members of the group communicated with one another via Wi-Fi (Highway Pilot Connect). They managed to shorten the distance between them to 15 meters and at the same time the method resulted in saving 10% of fuel.

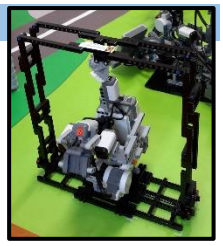
Smart Home technology means an automated technical unit of houses which cooperate to control the systems working in the houses effectively. Controllable subsystems include cooling/heating, operating the security system, opening and closing the doors and windows and operating household machines as well. In respect of saving energy the most important of these is regulating temperature. They usually use individual solutions; there is no connection to a central system like e.g. a weather station or the neighboring buildings. The control system is based on algorithmic principles. Systems controlled by artificial intelligence are still under development. It was in the 1960s that the first computer-operated houses appeared: they already worked with sensor-based program-control. Development has continued ever since and the number of built-in controllable elements is growing all the time. They estimate the number of apartments with smart home technology in the US will be 8 million by the end of 2017.

Solar energy as one of the renewable resources seems to be the most exploitable. Technology is developing all the time. Exploiting solar radiation for electricity is not efficient enough yet. Energy conversion efficiency at room temperature is around 27% but they consider efficiency over 60% to be possible. Today's technology is far from it but in many places around the world solar farms are created to largely contribute to the energy supply of towns.

ATTACHMENT 3 – SOURCE CODES

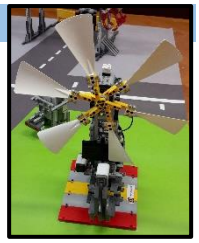
SOURCE CODES IN CHARACTER-BASED PROGRAMMING LANGUAGE (NXC)

Weather Station



```
string message="";

void Rotate(){
   while (Sensor(IN_4)!=6){
      OnFwd(OUT_B,100);
   }
   Off(OUT_B);
}

task Send(){
   float x;
   while (true){
      SendResponseString(1,message);
      Wait(200);
   }
}

task main(){
   int temp, press;
   int light, direction1, direction2;
   string dirstr, lightstr;
   SetSensorColorFull(IN_4);   //Wind direction measurement (weathercock)
   SetSensorLowspeed(IN_3);    //Compass sensor
   SetSensorLight(IN_2);       //Day-Night sensor
   SetSensorLowspeed(IN_1);    //Barometric sensor
   SetSensorType(IN_2,IN_TYPE_LIGHT_INACTIVE);

   StartTask(Send);

   while (true){
      ReadSensorHTBarometric(IN_3, temp, press);
      temp/=10;               //Temperature
      press*=0.0254;          //Press
      do {                    //Direction
         direction1=SensorHTCompass(IN_1);
         Wait(300);
         direction2=SensorHTCompass(IN_1);
         Wait(300);
      } while (direction1!=direction2);
      if (direction1<10)
          dirstr="00"+NumToStr(direction1);
      else if (direction1<100)
               dir="0"+NumToStr(direction1);
           else dir=NumToStr(direction1);
      light=Sensor(IN_2);     //Light
      if (light<10)
          lightstr="0"+NumToStr(light);
      else
          lightstr=NumToStr(light);
      message=NumToStr(temp)+NumToStr(press)+dir+lig;
      if (Sensor(IN_4)!=6) Rotate();
   }
}
```

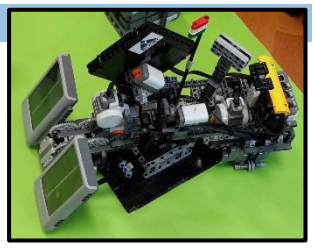
Wind Farm



```
int rpm=0, old=0;
string message;

void BT_fel_csatol(int x, string nev){
   CommBTConnectionType args;
   args.Name = nev;
   args.ConnectionSlot = x;
   args.Action = 1;
   SysCommBTConnection(args);
   while(BluetoothStatus(x)!=0);
}

task Communicate(){
   string received;
   while (true){
      do{
         ReceiveRemoteString(1,1,received);
      } while (received=="");
      message=received;
      message+=NumToStr(old);
      SendRemoteString(2,1,message);
      Wait(100);
   }
}

task Rotate(){
   SetSensorLowspeed(IN_2);
   int direction=0;
   int target_dir=0;
   int dir=1;
   while (true){
      target_dir=StrToNum(SubStr(message,5,3));
      target_dir = target_dir%360;
      direction=SensorHTCompass(IN_2)%360;
      if(abs((target_dir-360)-direction) < abs(target_dir-direction)){
        dir = 50;
      }else if(abs(target_dir+360)-direction < abs(target_dir-direction)){
        dir = -50;
      }else{
         if(target_dir-direction < 0){
            dir = 50;
         }else{
            dir = -50;
         }
      }
      OnFwd(OUT_B,dir);
       NumOut(0,0,target_dir,1);
       NumOut(0,10,direction,0);
       NumOut(0,20,dir,0);

      if(abs((target_dir%360)-(direction%360))>20){
         while (abs((target_dir%360)-(direction%360))>5){
            direction=SensorHTCompass(IN_2)%360;
            NumOut(0,0,target_dir,1);
            NumOut(0,10,direction,0);
            NumOut(0,20,dir,0);
         }
      }
      Off(OUT_B);
```

```
      Wait(100);
   }
}

task main(){
   BT_fel_csatol(1,"MetStat");
   PlayTone(440,200);
   Wait(200);
   BT_fel_csatol(2,"MetAI");
   PlayTone(440,200);
   Wait(200);

   SetSensorLight(IN_1);

   bool isOnWhite = false;
   int whitecount=0;
   long kezd;
   StartTask(Rotate);
   StartTask(Communicate);
   while(true){
      NumOut(0,50,rpm,0);
      if (rpm!=old){
         old=rpm;
      }
      if (rpm==0) old=-1;
      whitecount = 0;
      kezd = CurrentTick();
      while(CurrentTick() - kezd < 500){
         if(Sensor(IN_1) < 50 && isOnWhite){
            isOnWhite = false;
         }
         if(Sensor(IN_1) > 50 && !isOnWhite){
            isOnWhite = true;
            whitecount++;
         }
      }
      rpm = whitecount;
   }
}
```

Solar Farm



```
task SunDirection(){   //Tilt of Solar panel
   int i=0, direction=1, Month=5000;
   while (true){
      RotateMotor(OUT_A,direction*20,14);
      Wait(Month);
      i++;
      if (i==6){
         direction*=-1;
         i=0;
      }
   }
}

task LedFlash(){
   int power=0;
   while (true){
      OnFwd(OUT_B,power*10);
      Wait(500);
      power+=10;
```

```
      if (power==110) power=0;
   }
}

task main(){
   SetSensorLight(IN_2);
   SetSensorLight(IN_3);
   SetSensorType(IN_2,IN_TYPE_LIGHT_INACTIVE);
   SetSensorType(IN_3,IN_TYPE_LIGHT_INACTIVE);
   SetSensorLowspeed(IN_1);
   int LeftLight, RightLight, Compass;
   StartTask(SunDirection);
   StartTask(LedFlash);
   while (abs(SensorHTCompass(IN_1)-180)>2){
       OnFwd(OUT_C,60);
       NumOut(80,30,SensorHTCompass(IN_1),0);
   }
   Off(OUT_C);
   while (true){       //Follow the light
        LeftLight=Sensor(IN_2);
        RightLight=Sensor(IN_3);
        if (abs(LeftLight-RightLight)>3){
           if (LeftLight>RightLight){
              OnFwd(OUT_C,-50);
           }
           else{
              OnFwd(OUT_C,50);
           }
        }
        else{
            Off(OUT_C);
        }
   }
}
```

Weather Data Center (Dual NXT)

```
string message="32759123560";
int rpm=0;

string TextLight(int light){
   string text_light;
   if (light<20) text_light="Night";
   else if (light<50) text_light="Cloudy";
        else text_light="Sunny";
   return text_light;
}

string TextDirection(int direction){
   string text_direction;
   if (direction<23) text_direction=  "South    ";
   else if (direction<68) text_direction= "SouthWest";
        else if (direction<113) text_direction="West     ";
             else if (direction<158) text_direction="NorthWest";
                  else if (direction<203) text_direction="North    ";
                       else if (direction<248) text_direction="NorthEast";
                            else if (direction<293) text_direction="East   ";
                                 else if (direction<338) text_direction=
                                                          "SouthEast";
                                      else text_direction="South    ";
   if (rpm==0) text_direction="No wind  ";
```

```
   return text_direction;
}

void RefreshScreen(){
   int num;
   string answer;
   TextOut(0,0,"Temp.: ",1);                //temp
   TextOut(44,0,SubStr(message,0,2),0);
   CircleOut(75,5,2);
   TextOut(79,0,"C",0);

   TextOut(0,16,"Press: ",0);               //press
   TextOut(38,16,SubStr(message,2,3),0);
   TextOut(75,16,"Hgmm",0);

   TextOut(0,32,"Wind pow.: ",0);           //wind power
   rpm=StrToNum(SubStr(message,10,1));
   NumOut(60,32,rpm*20,0);
   TextOut(80,32,"rpm",0);

   TextOut(0,40,"Wind: ",0);                //wind direction
   num=StrToNum(SubStr(message,5,3));
   answer=TextDirection(num);
   TextOut(40,40,answer,0);

   TextOut(0,56,"Light: ",0);               //light
   num=StrToNum(SubStr(message,8,2));
   answer=TextLight(num);
   TextOut(38,56,answer,0);
   TextOut(75,56,"(",0);
   TextOut(80,56,SubStr(message,8,2),0);
   TextOut(92,56,")",0);
}

void SendRS485String(const string msg){
   byte mlen = ArrayLen(msg);
   SetHSOutputBuffer(0, mlen, msg);
   SetHSOutputBufferOutPtr(0);
   SetHSOutputBufferInPtr(mlen);
   SetHSState(HS_SEND_DATA);
   SetHSFlags(HS_UPDATE); //send it
}

void WaitForMessageToBeSent(){
   while(HSOutputBufferOutPtr() < HSOutputBufferInPtr())
   Wait(1);
}

task SmartHomeData(){
   SetSensorType(IN_4, SENSOR_TYPE_HIGHSPEED);
   SetHSState(HS_INITIALISE);
   SetHSFlags(HS_UPDATE);
   Wait(10);
   string HomeData;
   float temp, light;
   while (true)   {
      if ((rpm*20)>=40) HomeData="1";   //Window closed
      else HomeData="2";           //Window opened

      temp=StrToNum(SubStr(message,0,2));
      if (temp>25) HomeData+="1";     //Window closed, fan on
```

```
      if (temp<18) HomeData+="2";     //Window closed, heating on
      if ((temp<=25) && (temp>=18)) HomeData+="9"; //Window opened, fan off,
heating off

      light=StrToNum(SubStr(message,8,2));
      if (light<20) HomeData+="1";   //Light on
      else HomeData+="2";            //Light off
      SendRS485String(HomeData);
      WaitForMessageToBeSent();
   }
}

task WindData(){
   int data;
   while (true){
      do{
         ReceiveRemoteNumber(2,1,data);
         Wait(100);
      }while (data==0);
      if (data<0) rpm=0;
      else rpm=data;
   }
}

task RefreshData(){
   long begin;
   while (true){
      begin=CurrentTick()
      while (abs(CurrentTick()-begin)<1000);
      RefreshScreen();
   }
}

task main(){
   string received;

   StartTask(WindData);
   StartTask(RefreshData);
   StartTask(SmartHomeData);

   while (true){
      do{
         ReceiveRemoteString(1,1,received);
      } while (received=="");
      message=received;
      Wait(100);
   }
}
```

Smart Home Data Center (Dual NXT)

```
string message;

void BT_fel_csatol(int x, string nev){
   CommBTConnectionType args;
   args.Name = nev; // The slave robot's name
   args.ConnectionSlot = x; // Number og the communication channel
   args.Action = 1; // 1-> connect on; 0-> connect off
   SysCommBTConnection(args);
   while(BluetoothStatus(x)!=0);
}
```

```
task MetAIRead(){
   byte mlen;
   SetSensorType(IN_4, SENSOR_TYPE_HIGHSPEED);
   SetHSState(HS_INITIALISE);
   SetHSFlags(HS_UPDATE); // start with empty input buffer
   SetHSInputBufferInPtr(0);
   SetHSInputBufferOutPtr(0);
   Wait(10);
   while(true){ // wait for a message to arrive
      mlen = 0;
      while (mlen == 0)
         mlen = HSInputBufferInPtr();
      GetHSInputBuffer(0, mlen, message); // read it
      SetHSInputBufferInPtr(0); // clear the incoming buffer
   }
}

task main(){
   BT_fel_csatol(1,"sHome");
   PlayTone(440,200);
   Wait(200);

   StartTask(MetAIRead);

   while (true){
       TextOut(0,0,message,1);
       SendRemoteString(1,1,message);
       Wait(100);
   }
}
```

Smart Home I (Dual NXT)

```
bool wstate=0; //window, 0=closed, 1=opened
bool fstate=0; //fan, 0=off, 1=on
bool rstate=0; //roof, 0=solar, 1=black
string message="111";

task MetData(){
   while (true){
      ReceiveRemoteString(1,0,message);
      Wait(500);
   }
}

void SendRS485String(const string msg){
   byte mlen = ArrayLen(msg);
   SetHSOutputBuffer(0, mlen, msg);
   SetHSOutputBufferOutPtr(0);
   SetHSOutputBufferInPtr(mlen);
   SetHSState(HS_SEND_DATA);
   SetHSFlags(HS_UPDATE);
}

void WaitForMessageToBeSent(){
   while(HSOutputBufferOutPtr() < HSOutputBufferInPtr())
   Wait(1);
}

task SendData(){
```

```
   SetSensorType(IN_4, SENSOR_TYPE_HIGHSPEED);
   SetHSState(HS_INITIALISE);
   SetHSFlags(HS_UPDATE);
   Wait(10);
   while (true){
      SendRS485String(message);
      WaitForMessageToBeSent();
   }
}

void OnOff(bool window, bool roof, int heating, bool fan) {  //roof
   if(roof!=rstate) {
       RotateMotor(OUT_C,50,180);
       rstate=0;
   }

   switch (heating) {   //heating leds
     case    1    :    SetSensorColorRed(IN_1);    SetSensorColorRed(IN_2);
SetSensorColorRed(IN_3); break; //warm
     case    2    :    SetSensorColorBlue(IN_1);    SetSensorColorBlue(IN_2);
SetSensorColorBlue(IN_3); break;  //cold
     case    9:    SetSensorColorGreen(IN_1);    SetSensorColorGreen(IN_2);
SetSensorColorGreen(IN_3); break;  //normal
   }

   if(fan!=fstate) {   //fan
     if (fan==1) {  //fan on
       OnFwd(OUT_B,50);
       fstate=1;
     }
     else {    //fan off
       Off(OUT_B);
       fstate=0;
     }
   }

   if(window!=wstate) {   //window
      if(wstate==1){  //close
        OnFwd(OUT_A,-50); Wait(500); Off(OUT_A);
        wstate=1;
      }
      else {  //open
        OnFwd(OUT_A,50); Wait(500); Off(OUT_A);
          wstate=0;
      }
   }
}

void Operation(string msg) {
   switch(msg) {
   //Wind-Temp-Light   window-roof-heatleds-fan  heatled: 1-red; 2-green; 3-
blue
      case "111" :        OnOff(0,0,1,1); break;
      case "112" :        OnOff(0,1,1,1); break;
      case "121" :        OnOff(0,0,3,0); break;
      case "122" :        OnOff(0,1,3,0); break;
      case "191" :        OnOff(0,0,2,0); break;
      case "192" :        OnOff(0,1,2,0); break;
      case "211" :        OnOff(0,0,1,1); break;
      case "212" :        OnOff(0,1,1,1); break;
      case "221" :        OnOff(1,0,3,0); break;
```

```
      case "222" :        OnOff(1,1,3,0); break;
      case "291" :        OnOff(1,0,2,0); break;
      case "292" :        OnOff(1,1,2,0); break;
   }
}

task main(){
    StartTask(MetData);
    StartTask(SendData);

    while (true){
       TextOut(0,0,message,0);
       Operation(message);
    }
}
```

Smart Home II (Dual NXT)

```
int wind, temp, light;
int lampstate=0; //lamp, 1=turn on, 0=turn off
int nestate=0; //umbrella, 1=opened, 0=closed
string message="555";

task ReceiveData(){
   byte mlen;
   SetSensorType(IN_4, SENSOR_TYPE_HIGHSPEED);
   SetHSState(HS_INITIALISE);
   SetHSFlags(HS_UPDATE);
   SetHSInputBufferInPtr(0);
   SetHSInputBufferOutPtr(0);
   Wait(10);
   while(true){
      mlen = 0;
      while (mlen == 0)
         mlen = HSInputBufferInPtr();
      GetHSInputBuffer(0, mlen, message);
      SetHSInputBufferInPtr(0);
   }
}

void Operation(){
      int x;
      x=StrToNum(SubStr(message,0,1));
      if(x!=0){
        wind=x;
      }
      x=StrToNum(SubStr(message,1,1));
      if(x!=0){
        temp=x;
      }
      x=StrToNum(SubStr(message,2,1));
      if(x!=0){
        light=x;
      }

  if(wind==2){    //Umbrella
    if(light==2){
      if(nestate==0){
        OnFwd(OUT_C,50);
        Wait(1000);
        Off(OUT_C);
```

```
        OnFwd(OUT_B,50);
        Wait(500);
        Off(OUT_B);
        nestate=1;
      }
    }
    else{
      if(nestate==1){
        OnFwd(OUT_C,-50);
        Wait(1100);
        Off(OUT_C);
        nestate=0;
      }
    }
  }
  else{
    if(nestate==1){
      OnFwd(OUT_C,-50);
      Wait(1100);
      Off(OUT_C);
      nestate=0;
    }
  }

  if(light==1){    //lamp
    if(lampstate==0){
     OnFwd(OUT_A,100);
     lampstate=1;
    }
  }
  else{
    if(lampstate==1){
     Off(OUT_A);
     lampstate=0;
    }
  }
}

task main(){
  StartTask(ReceiveData);

  while(true){
    TextOut(0,24,message,0);
    Operation();
  }
}
```

Self-Driving Car

```
int treshold = 40, speedHigh=40, speedLow=25;
int temp = 0;

void BT_fel_csatol(int x, string nev){
   CommBTConnectionType args;
   args.Name = nev;
   args.ConnectionSlot = x;
   args.Action = 1;
   SysCommBTConnection(args);
   while(BluetoothStatus(x)!=0);
}
```

```
void follow(){
  while(Sensor(S1) < treshold){
    if(Sensor(S2) > treshold){
      OnFwd(OUT_B, speedHigh);
      OnFwd(OUT_C, speedLow);
    } else {
      OnFwd(OUT_C, speedHigh);
      OnFwd(OUT_B, speedLow);
    }
    NumOut(0, 0, Sensor(S1), true);
  }
  Off(OUT_BC);
}

void intoGarage() {    //Into The garage
   SendRemoteNumber(1,1,1);   //Garage open
   Wait(2017);
   RotateMotor(OUT_BC,50,1300);
   OnFwd(OUT_BC,-50);
   Wait(100);
   Off(OUT_BC);
   SendRemoteNumber(1,1,1);   //Garage close
   Wait(500);
   SendRemoteNumber(1,1,2);   //Turn the car
   Wait(10000);
}

void waitForTheLamp(int ms) {
   temp = 0;
   while(temp == 0){
     ReceiveRemoteNumber(1,true,temp);
  }
   RotateMotor(OUT_BC,50,ms);
}

task main(){
  SetSensorColorRed(IN_2);
  SetSensorColorRed(IN_1);

  BT_fel_csatol(1,"garazs");
  PlayTone(700, 0.1);
  Wait(500);
  BT_fel_csatol(2,"lampa");
  PlayTone(700, 0.1);
  Wait(500);

  while(true){
//Garage open
  SendRemoteNumber(1,1,1);
  Wait(2000);
  OnFwdSync(OUT_BC, 50, 0.5);
  Wait(2000);
  Off(OUT_BC);
  OnFwd(OUT_C, 40);
  OnFwd(OUT_B, 30);
  while(Sensor(IN_2) < treshold){};
  Off(OUT_BC);

  follow();

//Garage close
```

```
  SendRemoteNumber(1,1,1);

//Wait for the lamp (into front)
   waitForTheLamp(117);

   follow();

//Turn back
  OnFwd(OUT_B, 50);
  OnFwd(OUT_C, -50);
  Wait(500);
  while(Sensor(IN_2) < treshold){};
  Wait(200);
  while(Sensor(IN_2) > treshold){};
  Wait(100);
  Off(OUT_BC);

  follow();

//Wait for the lamp (into back)
   waitForTheLamp(117);
   follow();
   intoGarage();
   }
}
```

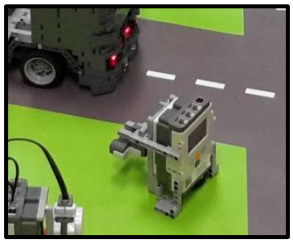
Garage

```
bool garageOpen;
bool rotated = false;

void ChangeGarageState(){
  if(garageOpen){
    RotateMotor(OUT_AC, -50, 100);
  }else{
    RotateMotor(OUT_AC, 50, 110);
    OnFwd(OUT_AC,10);
  }
  garageOpen = !garageOpen;
}

void Rotate(){
  if(rotated){
    OnFwd(OUT_B, 50);  Wait(1500); while (Sensor(IN_4)<40); Off(OUT_B);
  }else{
    OnFwd(OUT_B,  -50);  Wait(1500);  while  (Sensor(IN_4)<40);  Wait(100);
Off(OUT_B);
  }
  rotated = !rotated;
}

task main(){
  int val = -1;
  SetSensorColorRed(IN_4);
  while(true){
    ReceiveRemoteNumber(1,true,val);
    if(val != -1){
       if(val == 1){
        ChangeGarageState();
      }else if(val == 2){
        Rotate();
```

```
      }
      val = -1;
    }
  }
}
```

Traffic Lamp

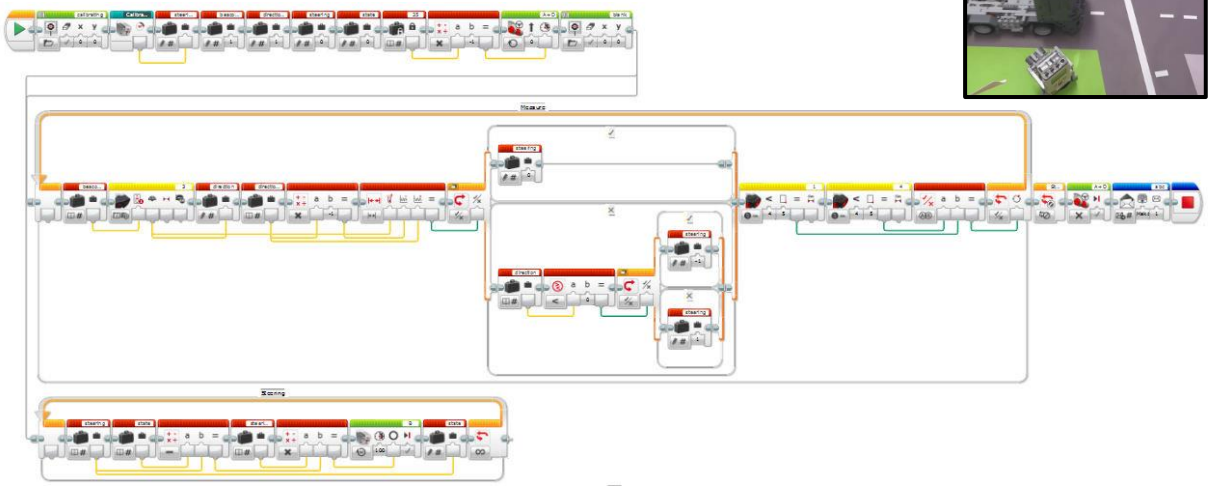
```
task main(){
  int distance = 30;      //Truck distance from the crossroad
  SetSensorLowspeed(S2);
  while(true){
    if(SensorUS(S2) < distance){
      SendResponseNumber(1,0);
      NumOut(0,0,0,true);
    }else{
      SendResponseNumber(1,1);
      NumOut(0,0,1,false);
    }
    NumOut(0,10,SensorUS(S2),false);
    Wait(300);
  }
}
```



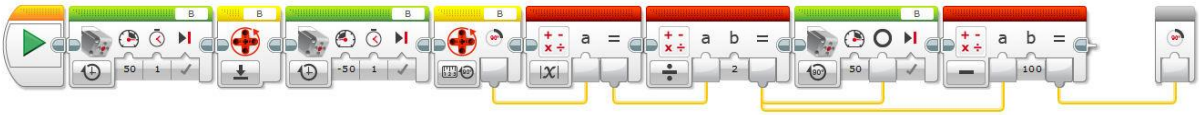
SOURCE CODES IN NODE-BASED PROGRAMMING LANGUAGE (EV3-G)

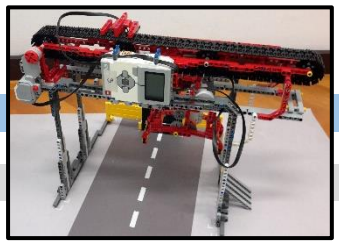
Cargo Truck

Main



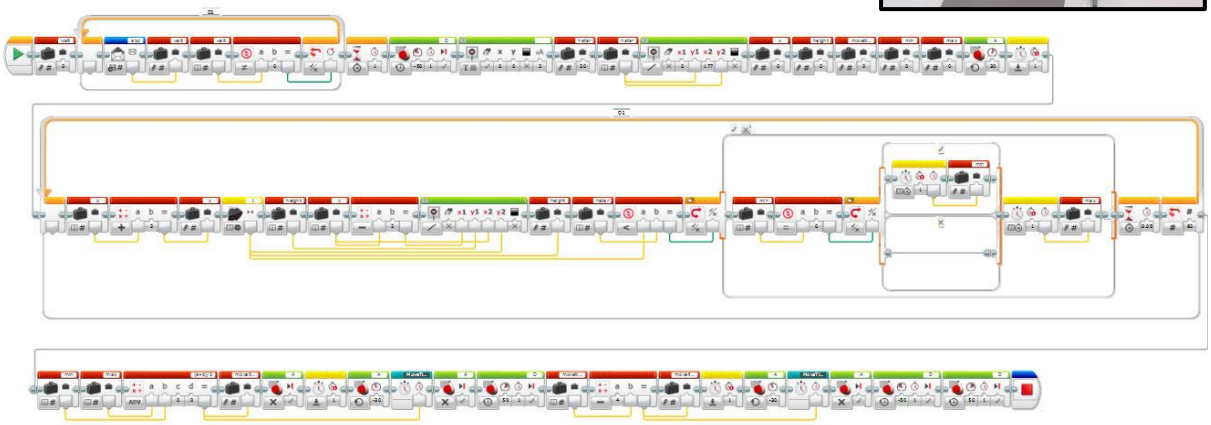
MyBlock – CalibrateSteering



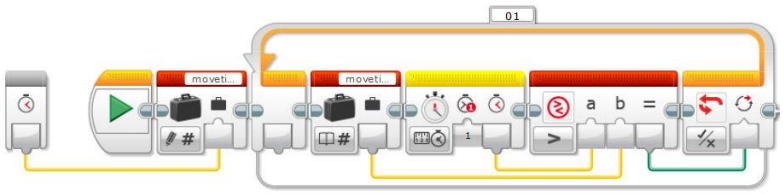


Crane

Main

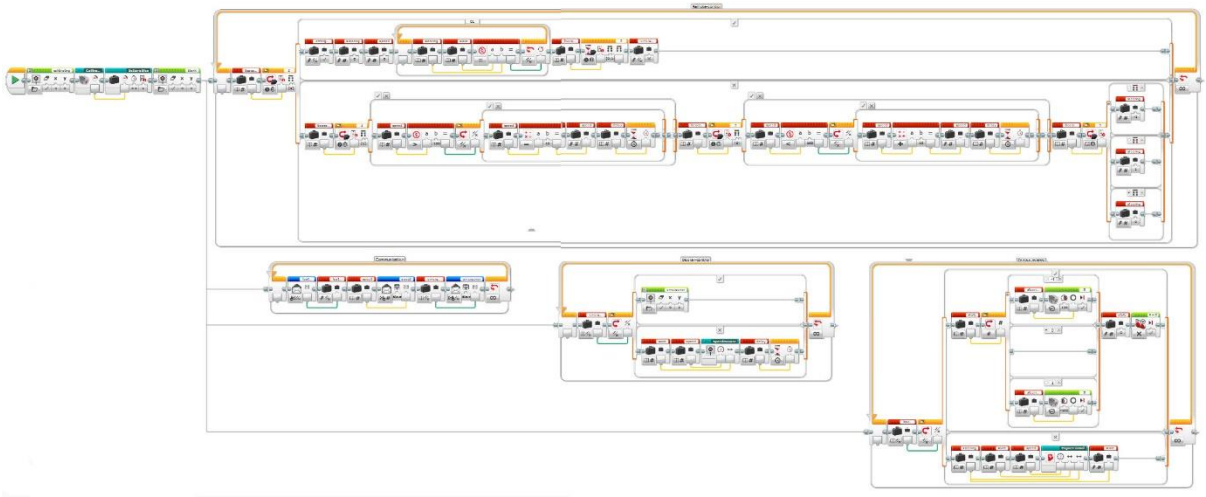


MyBlock – MoveTime

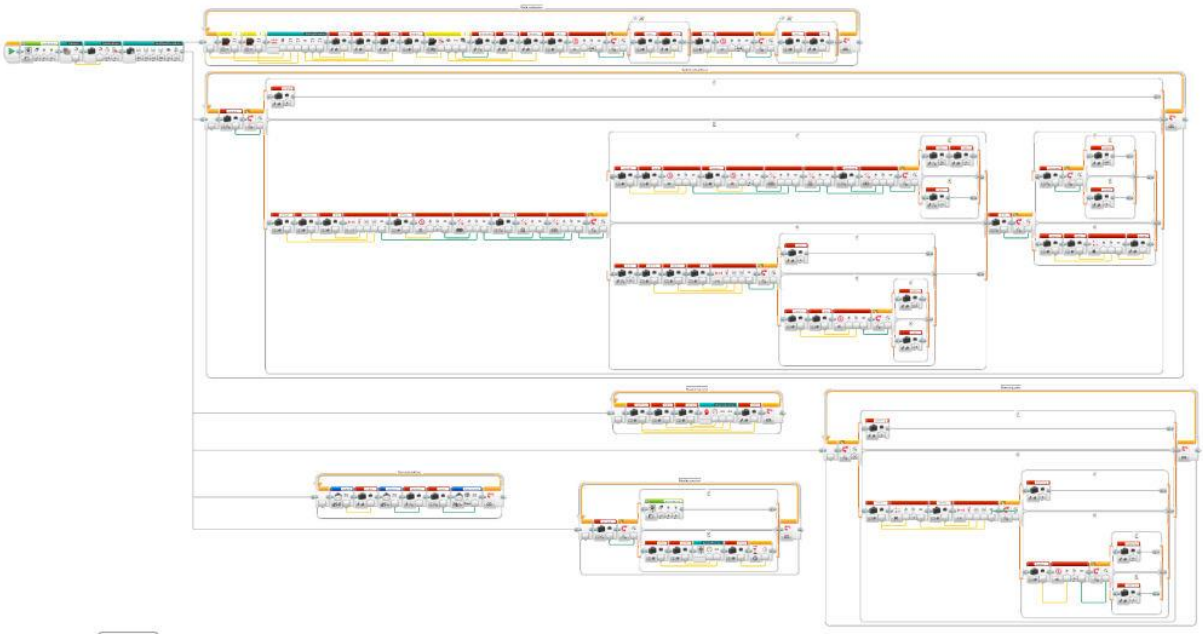


Swarm Truck System

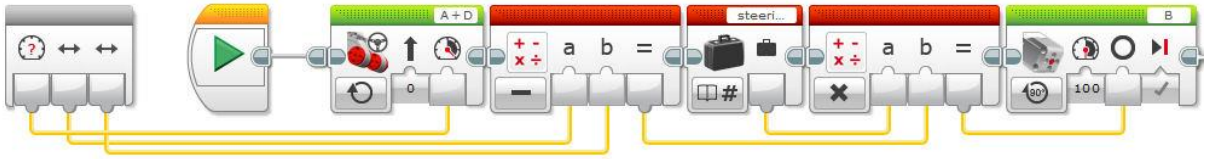
Main – Master



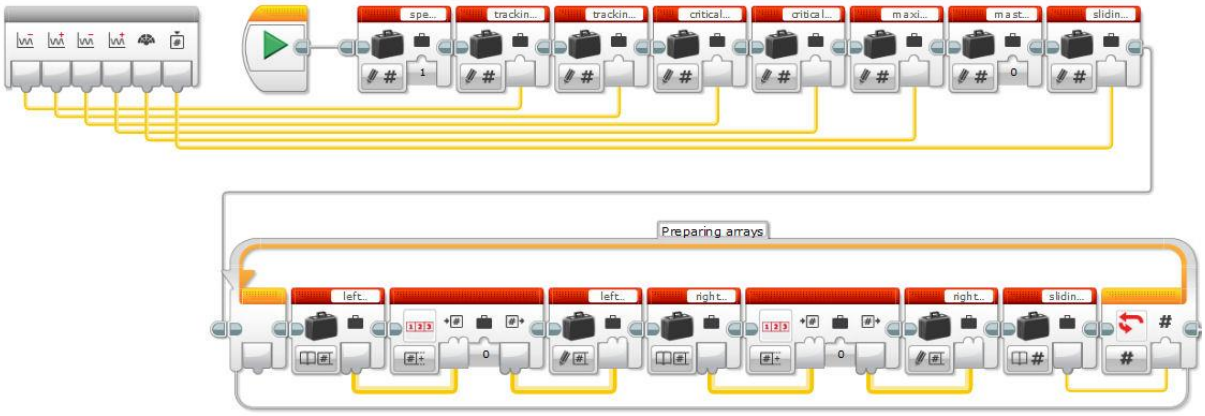
Main – Slave



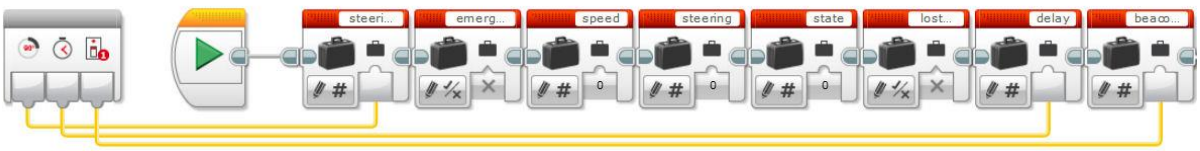
MyBlock – EngineControl



MyBlock – SetSlaveConstants

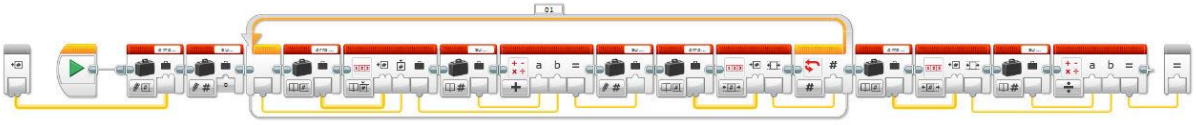


MyBlock – SetVariables



MyBlock – SlidingWindow

MyBlock – BlockAvg



MyBlock – SpeedMonitor