**Olimexino**

Model based software development

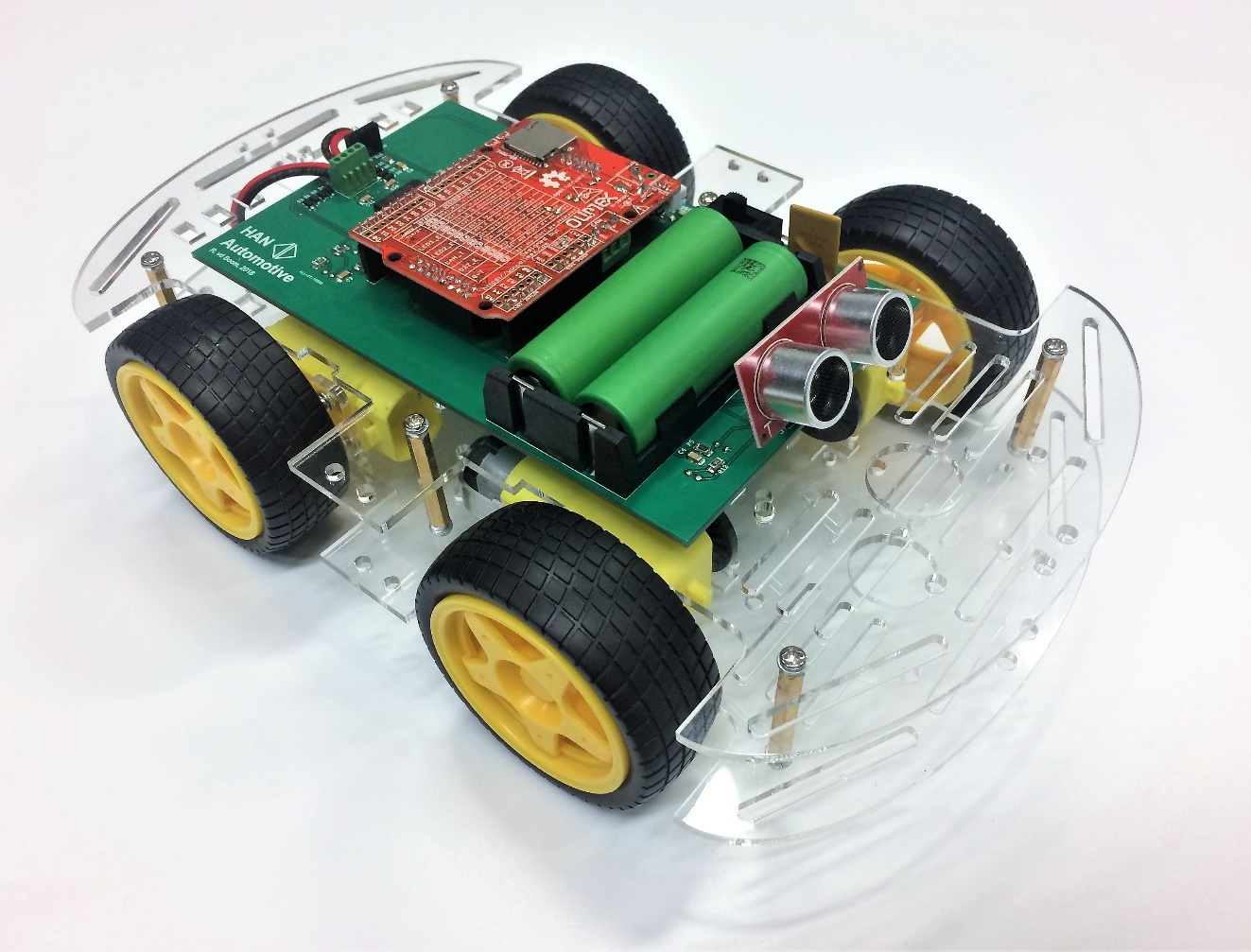
for a microcontroller system

Figure : Robot Car

Figure 2: Robot Car

**Traction Control**

Roel van den Boom

Version 1.1 English**Contents**

[1 Goal of the assignment 2](#_Toc509227413)

[2 Starting conditions 3](#_Toc509227414)

[3 Information about the software that’s being used 3](#_Toc509227415)

[4 The Simulink template model 4](#_Toc509227416)

[5 Assignment description 6](#_Toc509227417)

[5.1 Full Throttle 6](#_Toc509227418)

[5.2 Interrupt power when wheel slip has been detected 7](#_Toc509227419)

[5.3 Using a control system 8](#_Toc509227420)

[6 Compile the model and Program your Olimexino. 10](#_Toc509227421)

[7 View the variables in HANtune 11](#_Toc509227422)

**Images**

[Figure 1: Robot Car 1](file:///F:\HAN\04_Lesmateriaal\03_Au_2.0_Electro%20Practica\03_Robot%20Car\Redesign_Roel\Assignment\Olimexino%20Practicum%202%20English.docx#_Toc509227590)

[Figure 2: Simulink Model: Top level 5](#_Toc509227591)

[Figure 3: Simulink Model: Algorithm subsystem 6](file:///F:\HAN\04_Lesmateriaal\03_Au_2.0_Electro%20Practica\03_Robot%20Car\Redesign_Roel\Assignment\Olimexino%20Practicum%202%20English.docx#_Toc509227592)

[Figure 4: Position of the function button 7](file:///F:\HAN\04_Lesmateriaal\03_Au_2.0_Electro%20Practica\03_Robot%20Car\Redesign_Roel\Assignment\Olimexino%20Practicum%202%20English.docx#_Toc509227593)

[Figure 5: Simulink blocks 8](#_Toc509227594)

[Figure 6: Simulink Blocks 8](#_Toc509227595)

[Figure 7: Simulink blocks 9](#_Toc509227596)

[Figure 8: Simulink Build button 11](#_Toc509227597)

[Figure 9:Microboot waiting for target reset 11](#_Toc509227598)

[Figure 10: Add ASAP2 file to HANtune 12](file:///F:\HAN\04_Lesmateriaal\03_Au_2.0_Electro%20Practica\03_Robot%20Car\Redesign_Roel\Assignment\Olimexino%20Practicum%202%20English.docx#_Toc509227599)

[Figure 11: File location of ASAP2 file 12](file:///F:\HAN\04_Lesmateriaal\03_Au_2.0_Electro%20Practica\03_Robot%20Car\Redesign_Roel\Assignment\Olimexino%20Practicum%202%20English.docx#_Toc509227600)

[Figure 12: HANtune ASAP2 Elements 12](#_Toc509227601)

[Figure 13: COM port of connected Olimexino 13](#_Toc509227602)

[Figure 14: Setting the COM connection for HANtune 13](file:///F:\HAN\04_Lesmateriaal\03_Au_2.0_Electro%20Practica\03_Robot%20Car\Redesign_Roel\Assignment\Olimexino%20Practicum%202%20English.docx#_Toc509227603)

# 

# Goal of the assignment

The goal of this assignment is to make a traction control system for a model car that mimics the traction control system used in a modern car. As an optional extra the front ultrasonic sensor can be used to create a brake assist system.

Take a look at the bottom of the car, where you will find four sensors (light-slots) mounted to the PCB. A slotted wheel runs through this sensor, interrupting the light-slot each time the gap in the slotted wheel has passed. Each full revelation of the wheel will result in twenty pulses, which can be read by the microcontroller.

Also, you will find two electric motors, which can be controlled individually. A PWM signal (Pulse Width Modulation) will be used to regulate the motor power between 0% and 100%.

The main assignment is to compare the front wheel-speed with the rear wheel-speed while accelerating. If they are not equal, wheel slip has occurred. The only way to reduce wheel slip is to reduce the amount of power delivered by the motor.

**Create an algorithm which reduces the power output to the motors when wheel slip is detected.**

# Starting conditions

Make sure you have a laptop with the following software installed:

* Matlab Simulink 2014a or newer with the correct plug-ins (Getting started guide Olimexino)
* The HANcoder directory containing the Simulink template for this assignment
* A HANtune directory (optional: a HANtune.exe shortcut on your desktop)
* GNU ARM Toolchain
* OpenBLT USB driver

**If you have not yet done so or are missing any of the applications, follow the guide at:**

<http://openmbd.com/wiki/File:Getting_Started_Guide_Olimexino.pdf>

You will need an Olimexino STM32 board that can be connected to your laptop over USB. (USB to mini USB cable)

During the assignment, a robot car will be provided by the HAN.

Additional information necessary to conduct this assignment is provided within the Simulink template model.

# Information about the software that’s being used

For information with regards to HANcoder or HANtune, the openMBD wiki page can be consulted. On this page, you will find information about the different function blocks. There are also several examples as to how certain blocks can be used.

<http://openmbd.com/wiki/OpenMBD_Wiki>

# The Simulink template model

**Opening the Model**

1. Run MatLab as administrator (right click MatLab’s executable or shortcut -> choose ‘Run as Administrator’)
2. In MatLab navigate to the target folder of the assignment (..\Robot Car Practical\Robot\_Car\_Model\_2014a\Robot\_Car\_Algorithm\_2014a\Target)
3. In the MatLab workspace, double click the ‘HANcoder\_Olimexino\_ModelCar.mdl’ file to open the model.

When opening the template model (HANcoder\_Olimexino\_ModelCar.slx), the following subsystems will be visible:

*(if the olimexino picture shows, please double click on the picture)*

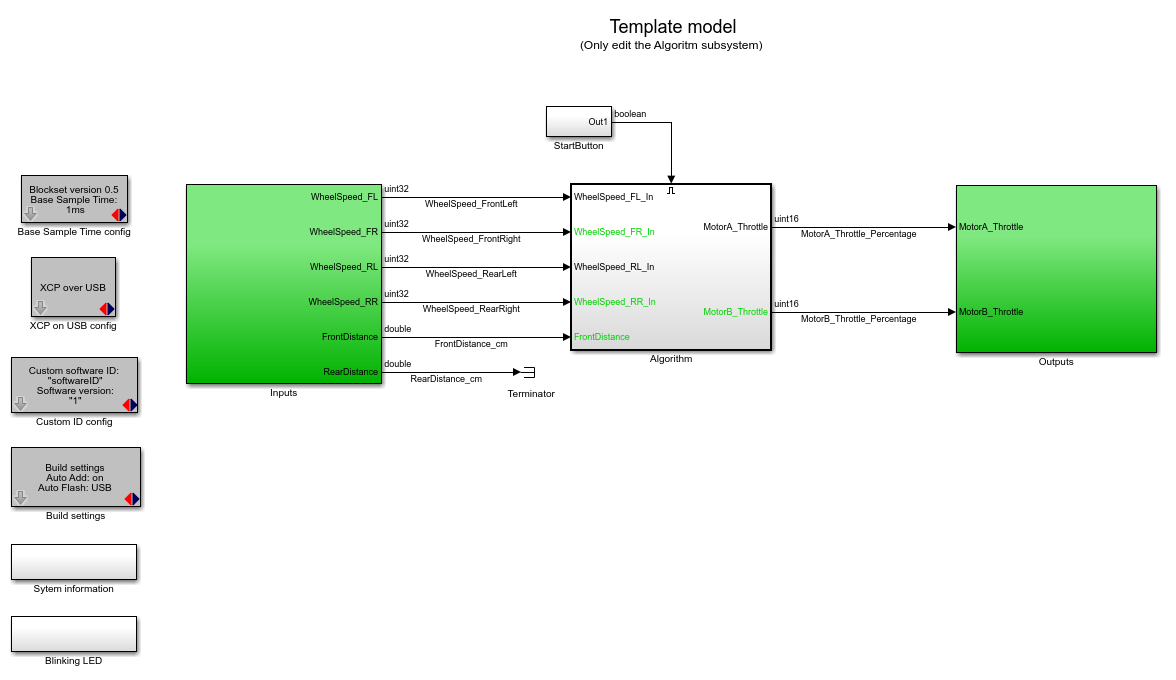


Figure 3: Simulink Model: Top level

Take a look at the contents of the Input and Output algorithms. The input and output subsystems are already finished and ready to use. The input subsystem translates the raw analog and digital measurements and processes them into values which are more convenient to use, like speed [cm/s] or distance [cm]. The output block does the same thing in reverse, namely calculating the output to the motor driver based on a percentage for motor power and a boolean for direction.

To start the assignment, please open the Algorithm block. Here you will have to place your own blocks to control the model car. When opened, you will see the following:

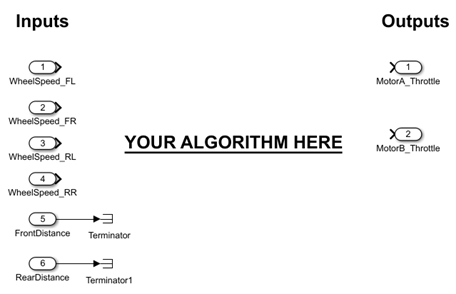


Figure 4: Simulink Model: Algorithm subsystem

The values for the speed of each wheel is at your disposal (in [cm/s]). Based on this information you will create different algorithms which act as your control system. The goal is to mimic a kick-down (full throttle, instantly) and make sure the wheels do not slip.

# Assignment description

## Full Throttle

First, we are going to observe what happens to the car when it gives full throttle, instantly.

Please add the following blocks to your model, inside the Algorithm subsystem:

Connect each wheel-speed signal to a terminator. This will ignore the measured wheel-speeds. Now connect a constant block to both throttle outputs and give them the value of 100. (which resembles 100%)

Compile the program and flash the Olimexino according to chapter 6.

When the program is flashed, push the start button on your Olimexino to make it accelerate. (Do not push the reset button which is on the other side)

Start button

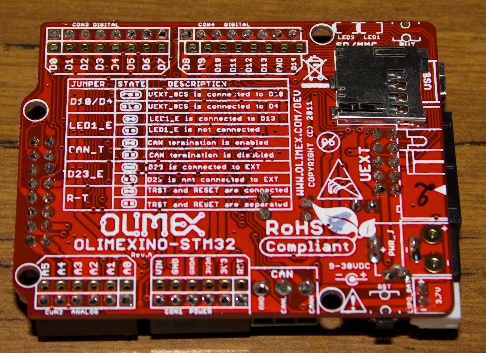


Figure 5: Position of the function button

There is a built-in delay of 1,5 seconds before taking off.

Observe the wheel-slip when accelerating.

**What happens when the car is accelerating with a weight attached to the rear of the car?**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Now prevent the car from moving. What happens when the car is trying to accelerate but is not able to?**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Interrupt power when wheel slip has been detected

Open the algorithm subsystem and remove the blocks placed in the previous sub-assignment.

First, calculate the average value of the rear wheels, using the following blocks:

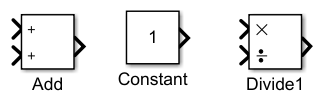


Figure 6: Simulink blocks

Use these blocks to calculate the sum of the rear wheels and divide the result by two.

Do the same thing for the front wheels.

Hint: to clean up your model, a subsystem can be created by selecting a part of your algorithm, right click and select “create subsystem from selection”

Now place the following blocks:

Figure 7: Simulink Blocks

The front wheels are driven, so are likely to run the fastest. Subtract the average rear wheel speed from the average front wheel speed. What remains is the difference in wheel speed front/rear.

(Please note: wheel speed is given in **centimeters per second! [cm/s]** )

* Use the Subtract block to determine the difference in wheel speed
* Connect the output of the subtract block to the Switch “condition” (middle input)
* Connect the output of the Switch block to the MotorA\_Throttle and MotorB\_Throttle

Two inputs of the Switch are still unconnected. When the Switch condition is true, the upper input will be used, when false, the lower input will be used. The Switch condition currently is the difference in wheel speed. When the wheel-speed exceeds 15[cm/s] we want to interrupt the power to the motor. Double-click the Switch block and enter the value “15” into the “Threshold” field.

* The upper input of the Switch will be used when the condition input exceeds 15. This is the case when wheel slip is detected. Connect a constant block to this input and provide it with the power output in percentage which you want the motors to produce in this case.
* The lower input of the Switch will be used when the condition input does not exceed 15. This is the case when no wheel slip is detected. Connect a constant block to this input and provide it with the power output in percentage which you want the motors to produce in this case.

Compile and flash your model into the Olimexino and observe its behaviour.

**What happens when the car is trying to accelerate when a weight is attached to the rear of the car?**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Using a control system

The switching behaviour of the previous control system is not ideal yet. It is possible to create an algorithm which doesn’t switch between two values but rather increases power output gradually when no wheel-slip has been detected or gradually decreases power output when wheel-slip has been detected. In this example it is possible to separately tune the increase and decrease of power based upon the detection of wheel-slip.

Delete the constant block which is attached to the first input of the switch and delete the line connecting the switch output to the MotorA\_Throttle and MotorB\_Throttle blocks.

Place the following blocks:

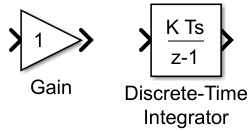


Figure 8: Simulink blocks

The Discrete Time Integrator is the digital representation of the integration algorithm known from mathematics, it behaves as follows:

When the input is positive and constant, the output will linearly increase over time. When the input is negative (and constant), it will linearly decrease over time.

Double click the Discrete Time Integrator and do the following:

* Set the initial condition to 25 (Power output %, to correct for internal friction of the motor)
* Upper saturation limit: 100 (cannot provide more power than 100%)
* Lower saturation limit: 25 (Power output % where the wheels start to turn, i.e. overcome their internal friction)
* Sample time -1 (-1 means inherited, 10ms in this case)

*Short recap: When the difference in average wheel-speed front and rear has exceeded 15 [cm/s] we have determined that wheel-slip has occurred. When that has happened, the power output should decrease. If no wheel-slip has occurred, the power output should increase. The key here is the Discrete Time Integrator, which steadily increases when the input is positive and decreases when the input is negative.*

So, when the input of the Switch is greater than 15 (upper input is used as output), wheel-slip has been detected and the power output should decrease. Place the gain to the left and above the Switch statement and connect the output of the Gain to the first input of the Switch. Connect the Gain input to the line which connects the Subtract to the Switch block. Double click the Gain and fill in: Ki\_Down. By filling in a name instead of a number we can later change its value without having to rebuild and flash the model. Connect the output of the switch to the input of the Discrete Time Integrator and the output of the Discrete Time integrator to the inputs of MotorA\_Throttle and MotorB\_Throttle.

Now double-click the Constant block which is connected to the third input of the Switch block. Fill in “Ki\_Up” in the Value field.

Please review your algorithm and try to understand what happens. When necessary connect any loose ends.

Now go back to the Matlab and enter the following commands in the Command Window, each line followed by pressing enter: *(hint: use copy-paste!)*

Ki\_Down = Simulink.Parameter

Ki\_Down.Value = -6

Ki\_Down.StorageClass = 'ExportedGlobal'

Ki\_Up = Simulink.Parameter

Ki\_Up.Value = 90

Ki\_Up.StorageClass = 'ExportedGlobal'

Click the “build” button () to compile the model and flash it to the Olimexino. Make sure your Olimexino is connected. When Microboot pops up, please reset your Olimexino using the reset button on the side of the Olimexino.

When the model is flashed and the Olimexino is mounted on the car, give it a test run. Observe its behaviour when it’s accelerating.

**What happens when the car is trying to accelerate but is unable to move due to an obstacle?**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**What happens while accelerating form standstill when a weight is mounted to the rear end of the car?**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Connect the Olimexino to HANtune and gradually change the values for Ki\_Up and Ki\_Down. See if you can make the car accelerate smoother when a weight is attached to the rear end. Consult Chapter 7 on how to use HANtune with your Olimexino.

## Optional extra: Brake Assist

The Robot Car has an ultrasonic sensor mounted on the front of the car. This can be used to interrupt the power output to the wheels when an obstacle has been sensed.

This particular sensor needs to be triggered by a pulse in order to perform a measurement. The distance measured will result in a pulse which is sent back to the microcontroller. The length of that pulse is proportional to the distance measured. For more information refer to the datasheet of the SR04 Ultrasonic Sensor.

The model is already equipped to periodically trigger the sensor and calculate the measured distance in cm according to the sensor’s response. The measured distance in cm is already available in the “Algorithm” subsystem.

**Try to modify your algorithm in such a way that the power output to the motor will be interrupted when an object has been detected within a range of 750 mm.**

*Hint: use a Switch just before the output to MotorA\_Throttle and MotorB\_Throttle.*

# Compile the model and Program your Olimexino.

**!!! BE AWARE !!!**

**After the flashing procedure is finished, the controller will automatically be reset. Ten seconds after the last reset the car will start accelerating for another 10 seconds.**

The Simulink model is now ready to compile and be programmed to the Olimexino.

Build your model.



Figure 9: Simulink Build button

After a licensing notification, Matlab will call up the GNU ARM Toolchain to convert your model to machine code. After the S file is ready, Matlab will start the Microboot program to flash the code to the Olimexino. A similar window to that of *Figure 9* should pop up. Reset your Olimexino to flash the program.

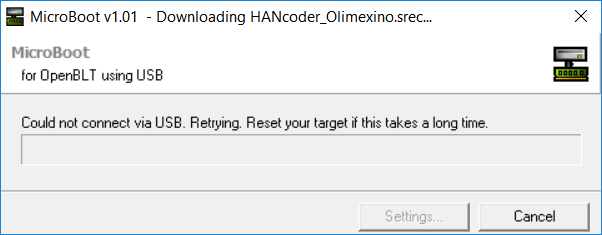


Figure 10:Microboot waiting for target reset

# View the variables in HANtune

In your model some signals and parameters are used. Now we want to visualize the signals and change the parameters using HANtune. Open HANtune from the directory you have saved it to.

Tip: Save the HANtune directory under “Documents” and save a HANtune.exe shortcut to your desktop.

When HANtune has finished loading, make a new project. *File 🡪 New project*

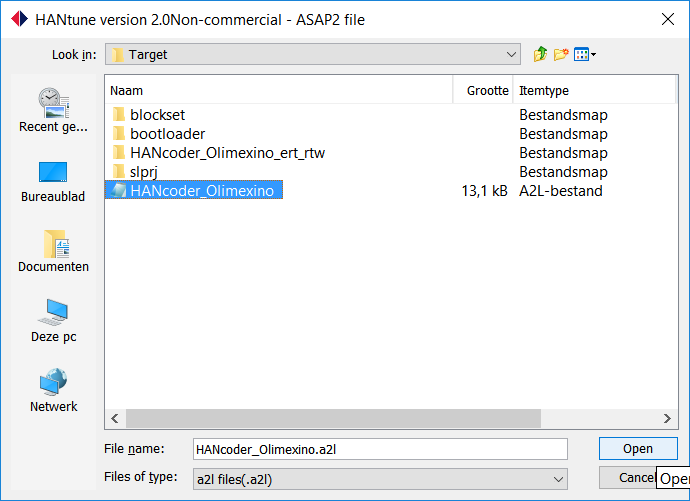
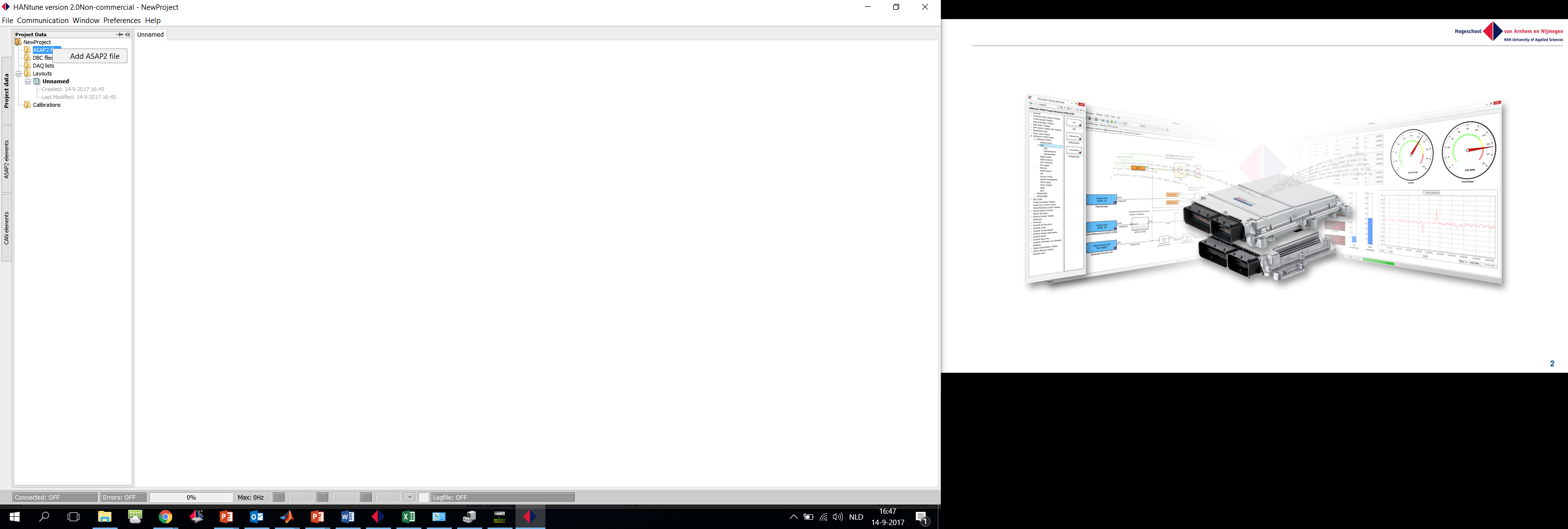
Now, open the ASAP2 file (a2l) by right clicking “ASAP2 files” and click “Add ASAP2 file”. The ASAP2 file can be found in the same folder as the Simulink template model.

Figure 11: Add ASAP2 file to HANtune

Figure 12: File location of ASAP2 file

The ASAP2 file needs to be loaded up before it can be used. Either right click the “HANcoder\_Olimexino\_ModelCar.a2l” ASAP2 file and click “(un)Load file” or double click it “HANcoder\_Olimexino\_ModelCar.a2l”.

Now the ASAP2 file will accommodate HANtune to use the added signals and parameters from Matlab. Next, you’ll put together a dashboard. Click the *ASAP* *Elements* tab on the left side of the HANtune window.

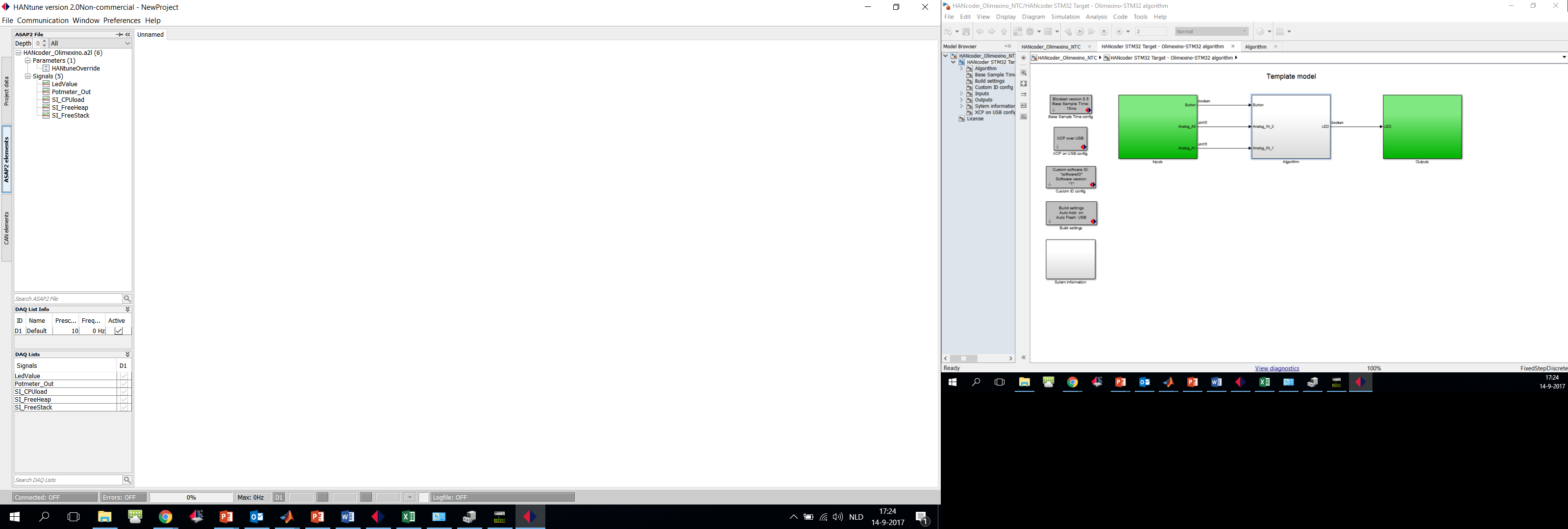


Figure 13: HANtune ASAP2 Elements

Drag and drop the parameters and signals which are needed into your HANtune Layout. Choose your editors and viewers to your liking.

Lastly, we need to connect the Olimexino to the computer to actually be able to read the value of the potentiometer. The Olimexino uses a serial USB connection. If you’ve installed the USB driver properly, your laptop should have assigned a COM port to your Olimexino. The first time you connect the Olimexino to HANtune, you’ll need to select the COM port which has been assigned to the Olimexino. To do this, press (F5) to connect and go to settings. Go to the USB/UART tab, select the proper COM port and click *OK*.

To figure out which COM port the Olimexino is using, right click the *Windows/Start* button on your screen and click *Device Manager*. Then go to *Ports (COM & LPT)* and click on it. Now look for *STMicroelectronics Virtual COM Port* and see which port has been assigned to it.

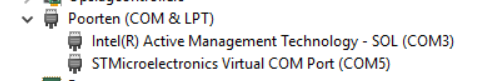


Figure 14: COM port of connected Olimexino

Choose *XCP on USB/UART* as your connection type.In this example, the Olimexino is connected to COM5. This is the port we will need to pick in *Settings*.

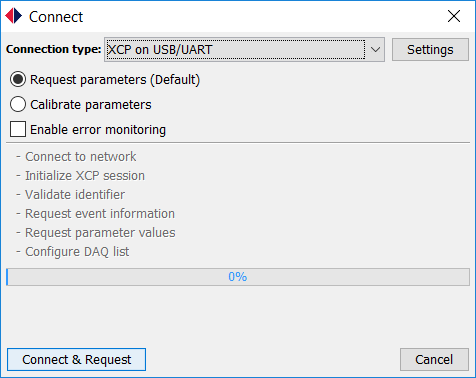
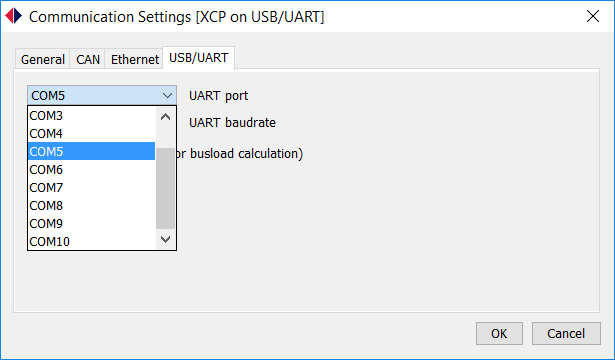


Figure 15: Setting the COM connection for HANtune

Now we can connect our Olimexino to HANtune by clicking *Connect & Request*. Once it’s connected, the data will become visible in real-time.