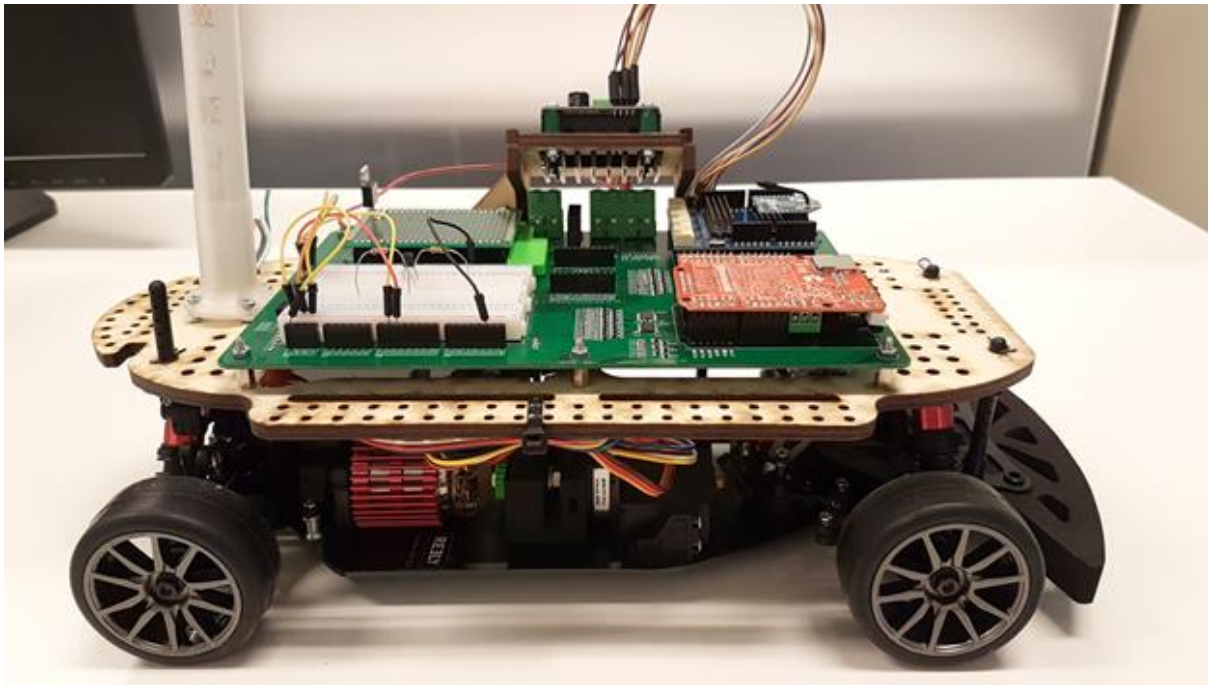
VEC S4 Report



HAN Automotive

Postbus 2217

6802 CE Arnhem, The Netherlands

Teachers: Stefan van Sterkenburg, Nijs van der Mei

25/06/2021

Version 1.1

Daniel Mirchev – 629183

Hassan Hotait – 593099

Mihail Markov – 636710

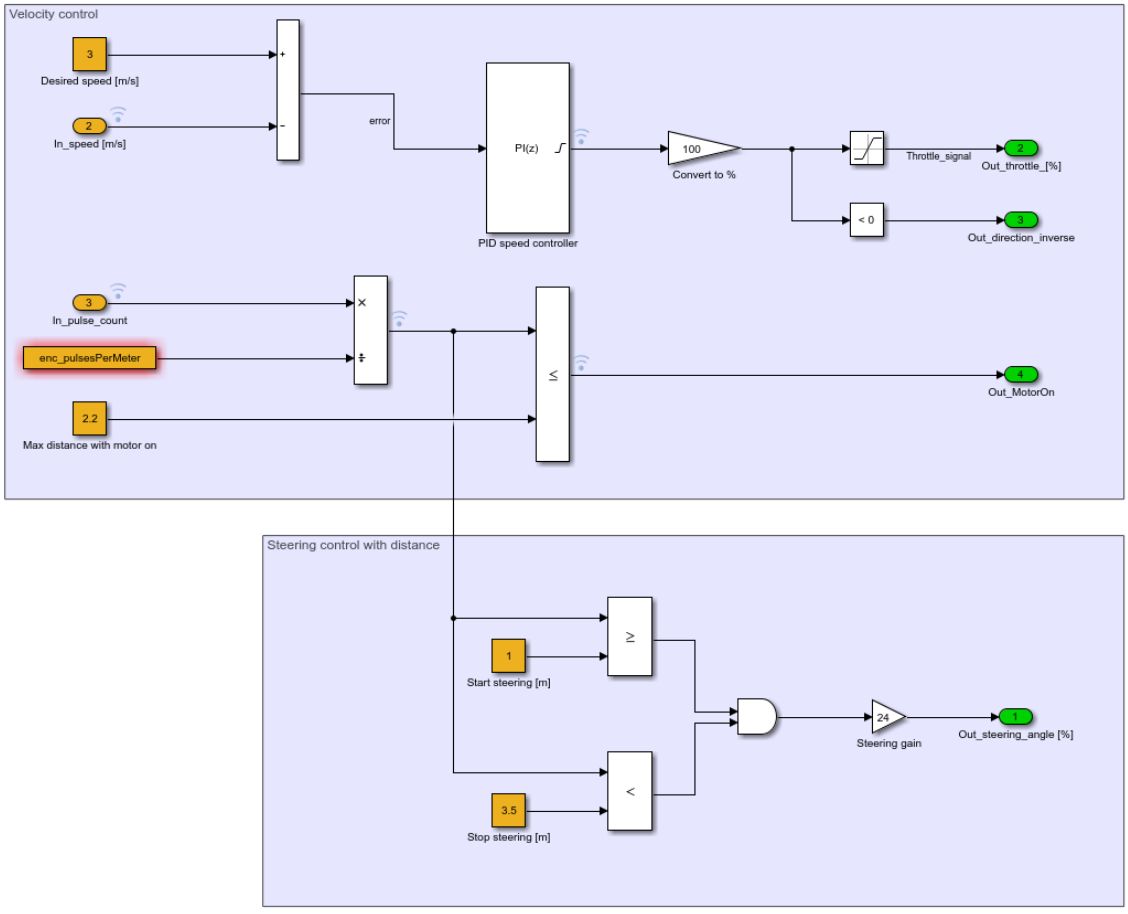
Nikolay Donchevski - 630132

Chapter 1 The vehicle model

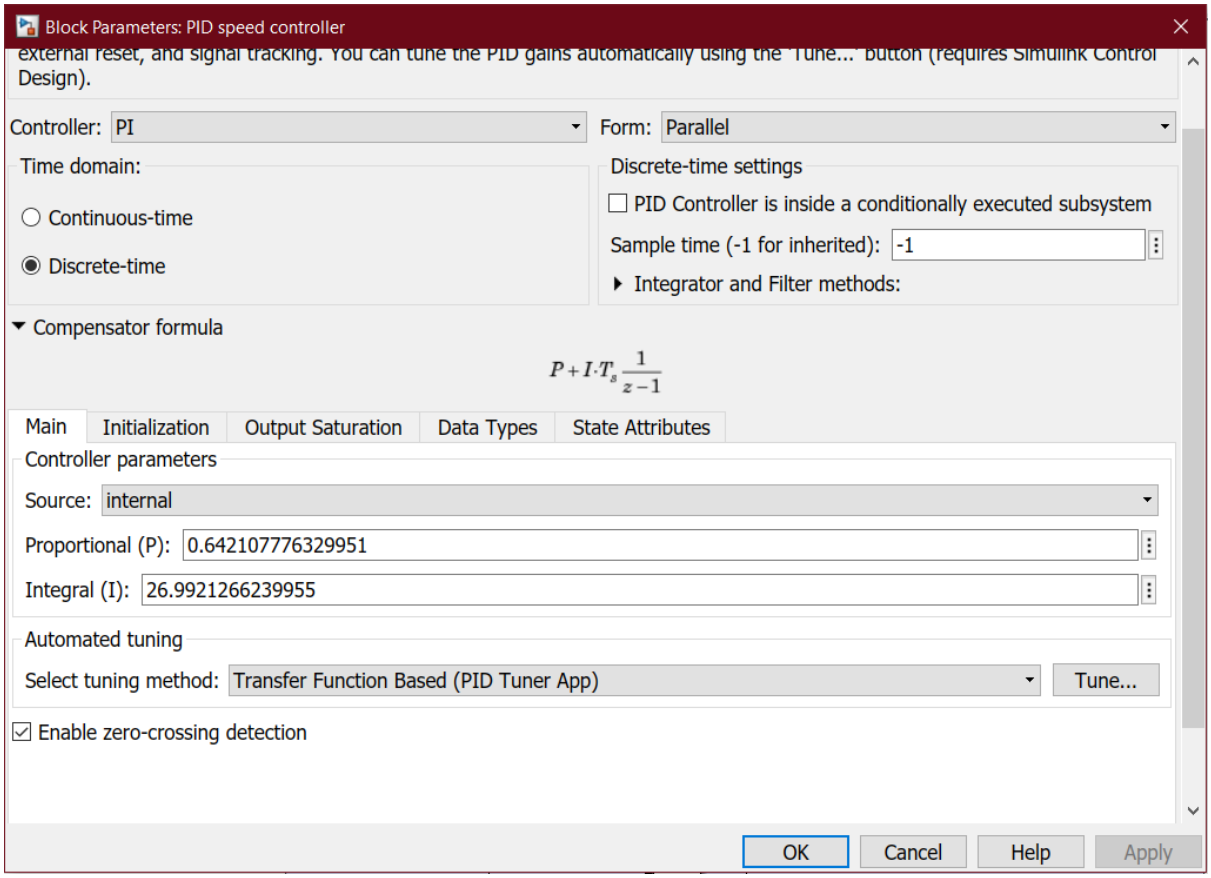
In the original model the conversion from steering percent to steering angle is done with a lookup table and it's linear. After we did measurements, we found that in the real world that is not the case and we changed the lookup table accordingly. Moreover, we neglect the friction in our model and we change a few parameters such as tyre radius, mass, distance to CoG, etc. Other than that our first model was the same as the original one.

Chapter 2 Control systems for the challenges

2.1 Intermediate challenge



For the intermediate challenge we are using a PI controller for the velocity control, as an input there is the speed of the car and the desired speed set by us. After the PI controller we convert the values to percentage and we use a saturation block to make sure it doesn't exceed the maximum values. Of course if the values are negative we reverse the direction.



The PI controller is auto tuned in Simulink.

We decided to use steering with distance for the intermediate challenge. We specify the starting and the ending of the steering. We use AND gate to get 1 when the distance travelled is between these values and 0 when not. After that we have a gain to control how much the car should steer.

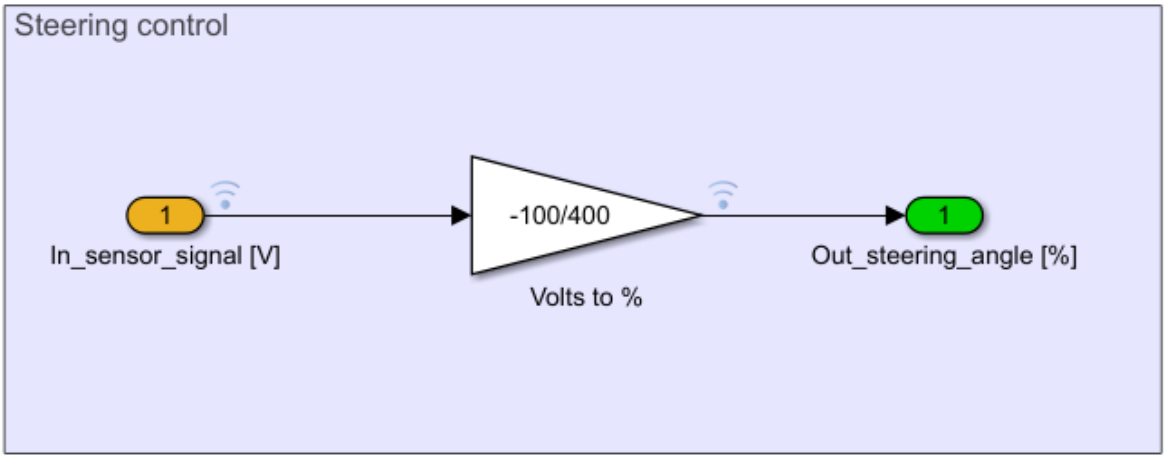
To get the distance travelled we take information from the encoder. We divide the signal by the settings of the encoder to get the distance in meters. We are using a maximum distance set to make the motor stop.

2.2 Final Challenge

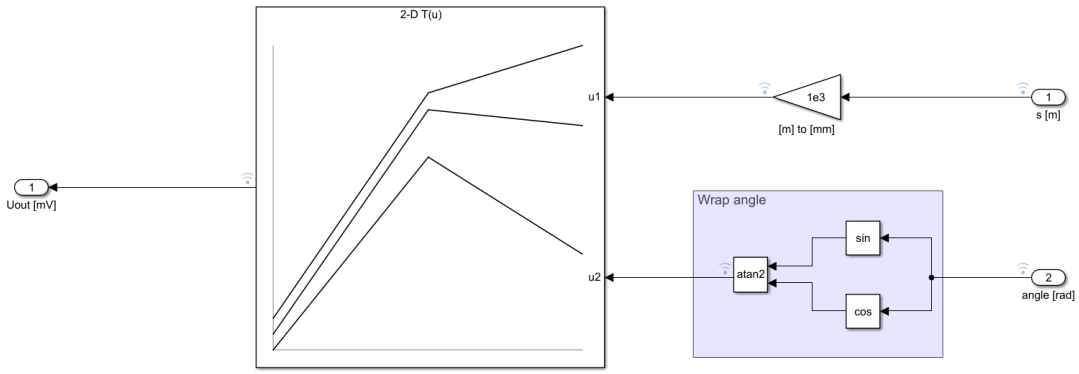
For the final challenge the velocity control is the same. The only difference is in the steering control because now we dont have a predefined track. In this challenge we have different approach in model-based and real-life attempts and they will be presented separately.

2.2.1 Model based challenge

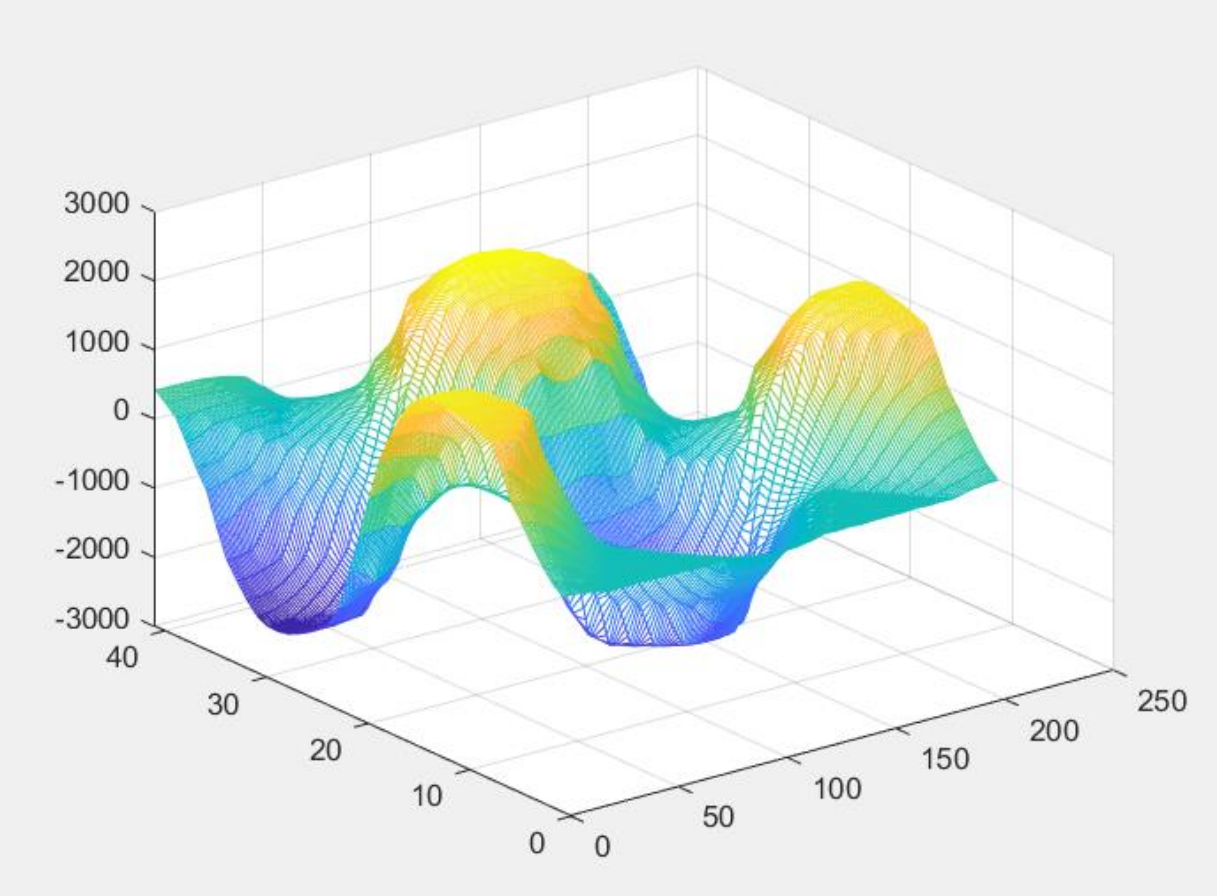
For the steering control we need voltage from the sensors (simulated signals) and then we use a gain to get a percentage of steering.



There is a subsystem developed by the teachers that defines the track. It outputs distance in meters from the track to the optical sensor and also it's angle. So these are the inputs for our optical path finding sensor subsystem.



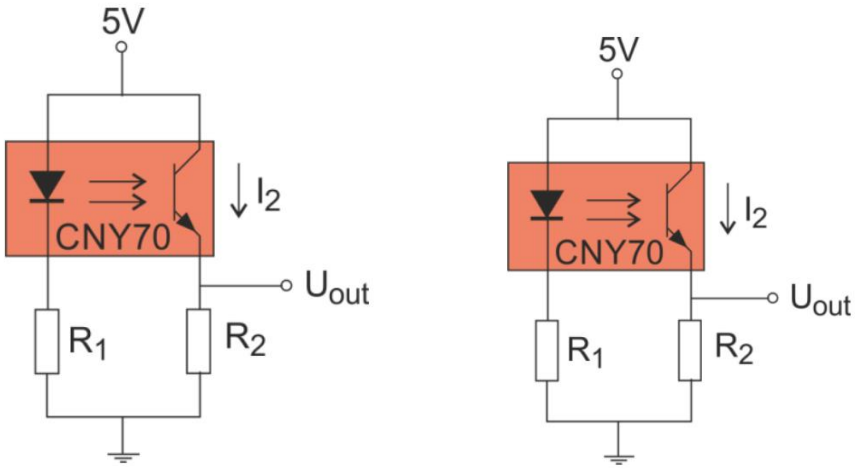
First we convert the meters in millimeters. We are using the formula for angle warping to make sure the angle is between -pi and pi in radians. This information is put into a 2-D lookup table with data measured from us (look Chapter 4).



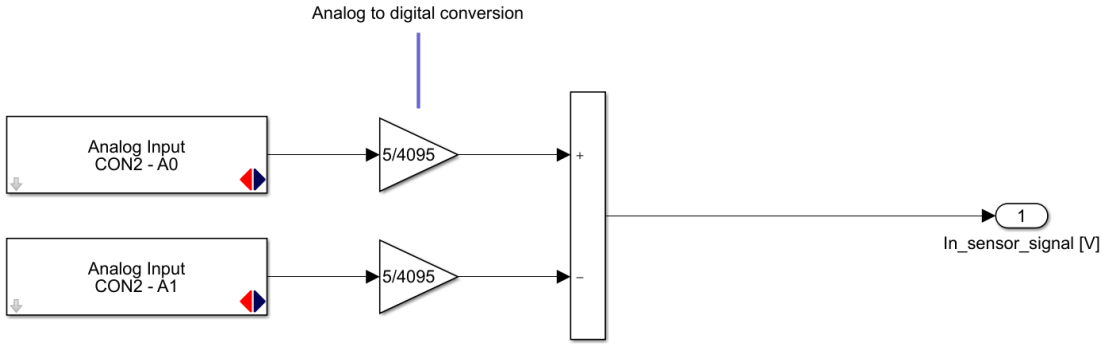
The output after the lookup table is voltage in mV, which goes into the aforementioned steering system.

2.2.2 Real world challenge

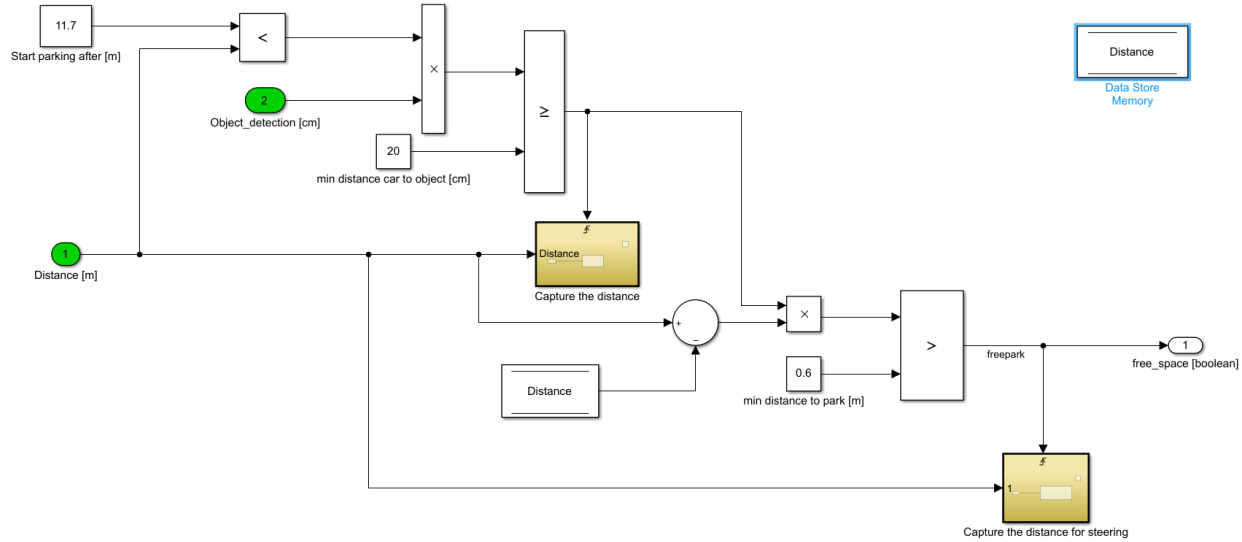
For the real world challenge we are using the simple configuration of the optical sensors.



They output two voltages and we take the difference between them to get information where the car is located according to the black tape.



The signal is then feeded to the same control system as the model based challenge. Furthermore a parking function is implemented in the real world challenge. First we have an algorithm which finds a free parking space. This algorithm activates after a certain distance, so the car doesn't start parking in the middle of the challenge.



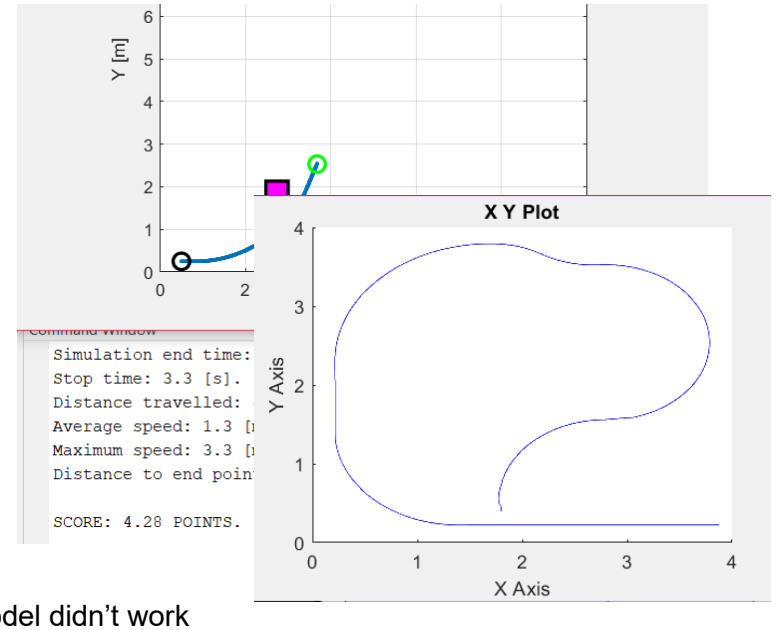
When the car has found a free space, a predefined parking manouver with lookup tables is performed.

Chapter 3 Results from challenges

3.1 Intermediate Challenge

For the intermediate challenge both the software and hardware models were able to make it successfully. For the software, we used a PI controller for the speed, with 3 m/s desired speed. In comparison, when we tried to implement the same control on the hardware, the wheels of the smart car were slipping at the beginning, which led to false measurement of the distance travelled and thus it could not stop at the right end spot. So we changed that, to go slowly at the first 0.3m with a fixed duty cycle of 30% (that was the upper limit before

wheels start to slip), then the maximum speed was set. Another thing that we needed to change from the software model was the steering. Since it is time based, we could not run it to the hardware (or at least we couldn't). So for the smart car, we used a logic to steer based on distance travelled. Overall, the models results were similar, with both finishing in 3 seconds. The smart car had an offset of 10 cm from the end point, while the software had only 1cm offset. The difference is likely to be due to the speed of the smart car, it would be more precise if we used lower overall speed, which will add 1-2 seconds but stop closer to the target.



3.2 Final Challenge

For the final challenge, the software model was quite good, making the track precisely every time. We used a speed of 1m/s with a PI controlled, which we were able to tune with the Tuner App. The gain for the steering, based on the optical sensor output was -3.125.

In comparison, the hardware model didn't work consistently unfortunately. We had to use a slower speed of 0.6 m/s. The PI controller had to be tuned manually, because it had a big overshoot at the beginning with the autotune setting from the software. Another difference was the gain for the steering, we used -0.0552. That huge difference comes from the fact that we didn't convert the xadc from the sensors into voltage for the car (because the range was better). The inconsistency mainly comes from the lack of a PI controller for the steering. So that is something that definitely can be improved. The parking was quite good during the challenge. Another point of improvement can be the elimination of the delay block that we used to delay the parking algorithm after the detection of the free spot.

Chapter 4 Optical sensor for path following

Question 1
Find in the datasheet the saturation voltage, maximum LED current, maximum collector current and maximum collector-emitter voltage.

Saturation Voltage	0.3V
Max LED current	50mA
Max Collector current	50mA
Max collector-emitter voltage	32V

Question 2
Calculate the resistor R1 in figure 3 to adjust the LED current to half of its maximum value (half of the maximum value gives a good sensitivity without having the risk that a voltage peak or other disturbance damages the sensor)

As the max LED current is 50mA, half of it is 25mA, so the resistance should be 5V/25mA = 200ohms

Question 3

a. Estimate the collector current of a CNY70 at the following condition:

- The sensor is placed above the blue linoleum floor in ARLA
- The reflection of the floor is 50% compared to that of the white side Kodak neutral card mentioned in the datasheet of the CNY70
- The distance from the sensor to the floor = 5 [mm]
- The LED current is 25 [mA]

Explain your answer. Mention which figures / tables you have used in the datasheet to get your answer.

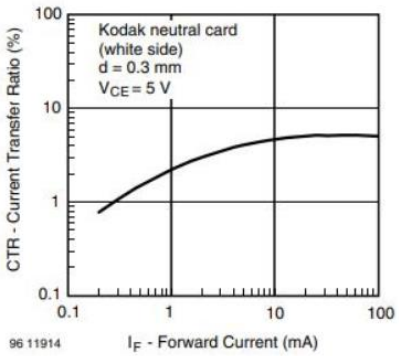


Fig. 7 - Current Transfer Ratio vs. Forward Current

Based on the graph on the right, the difference between 20mA and 25mA for the LED current is negligible. Therefore, the addition of 5mA won't have an effect of the collector current. With the help of the graph below, it can be seen that for 5mm, the collector current is around 0.2mA.

Because the blue floor has 50% of the reflection of the white card, the collector current will be around 0.1mA.

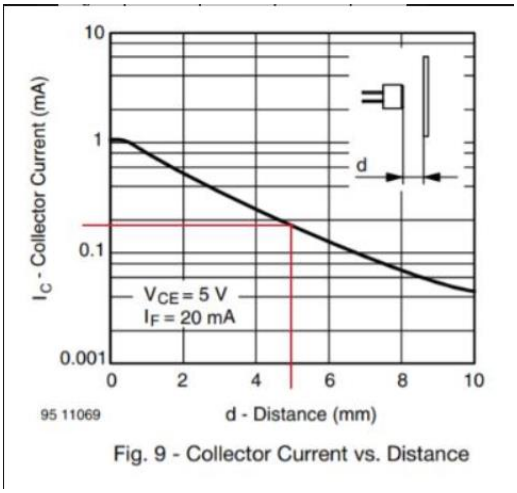


Fig. 9 - Collector Current vs. Distance

b. Estimate the collector current at the following condition:

- The sensor is placed above the black tape
- The reflection of the tape is 2% compared to that of the white side Kodak neutral card mentioned in the datasheet
- The distance from the sensor to the tape = 5 [mm]
- LED current is 25 [mA]

Explain your answer. Mention which figures / tables you have used in the datasheet to get your answer.

Using the same graphs as in question a and taking into account the 2% reflection, the collector current will be 0.2*0.02 = 0.004mA

Question 4

a. Calculate R2 in figure 3 to get a 3.0 [V] output voltage when the sensor is above the floor. Assume the collector current that you have calculated in question 3a.

Collector current = 0.1mA

Uout = 3V

R = 3V/0.1mA = 30kOhms

b. Calculate the output voltage of figure 3 when the sensor is above the black tape. Use the calculated value of R2 of the previous question. Assume the collector current that you have calculated in question 3b.
Collector current = 0.004mA
R2 = 30kOhms
U = R2*I = 0.12V

Question 5
a. Calculate R2 in figure 4 to get a 3.0 [V] output voltage when the upper sensor in figure 4 is located above the floor and the lower sensor is located above the black tape. Assume the collector currents that you have calculated in question 3.

So from the picture Ux is the same as Uout, and using Kirchoff, we can get that:

$$I_3 = \frac{5 - U_X}{R_2}$$

$$I_4 = \frac{U_x}{R_2} = I_1 - I_2 + I_3$$

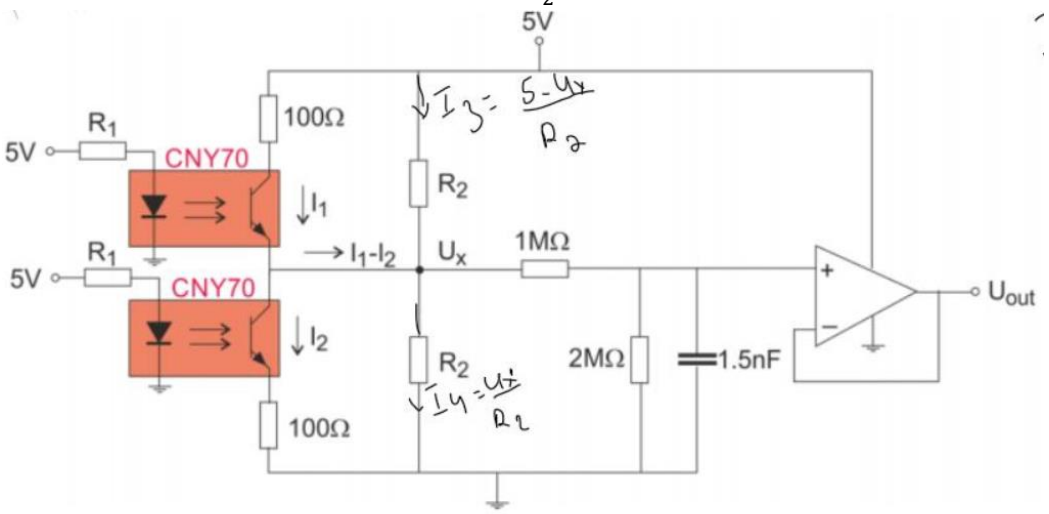
So based on these equations the Ux is:

$$U_x = (I_1 - I_2) * R_2 + 5 - U_x$$

After substituting for I1 and I2, we get:

$$\frac{(2U_x - 5)}{(0.1 - 0.004)} = R_2$$

$$R_2 = 10.4k\Omega$$



b. Calculate the output voltage of figure 4 when the lower sensor is located above the floor and the upper sensor is located above the black tape. Use the calculated value of R2 of the previous question. Assume the collector currents that you have calculated in question 3.

$$U_x = (I_1 - I_2) * R_2 + 5 - U_x$$

$$2U_X = (0.004 - 0.1) * 10.4 + 5$$

$$U_x = 2V$$

c. Calculate the output voltage of figure 4 when both sensors are located above the floor.

$$U_x = 2.5V$$

d. Calculate the output voltage of figure 4 when both sensors are located above the black tape.

$$U_x = 2.5V$$

Question 6

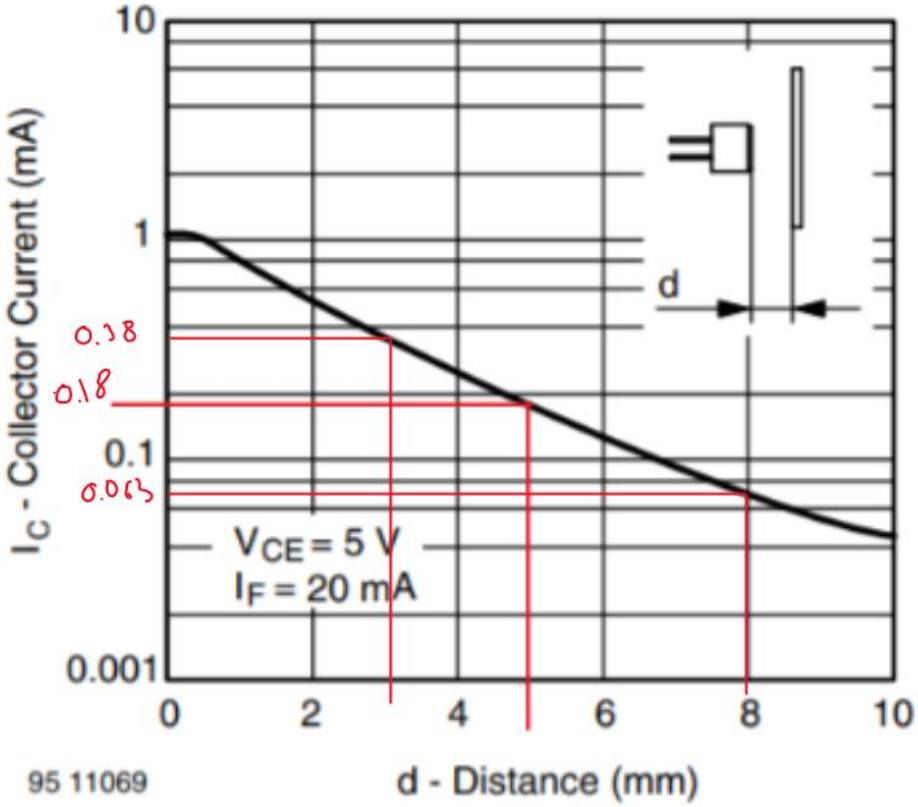


Fig. 9 - Collector Current vs. Distance

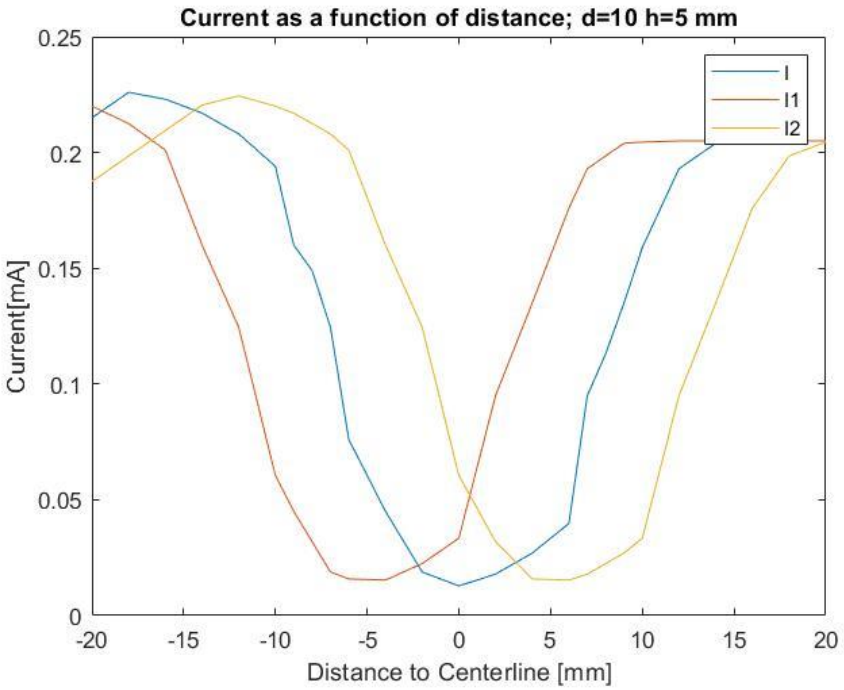
Optimal R2 is calculated to obtain a wide operating range, however the sensor must not be saturated. The saturation voltage is 0.3 V therefore the max voltage on R2 is 4.7 V. For Safety we will use max voltage as 4 V.
The maximum current at each height derived from the above graph. Then we assume 50% of that current since it's on blue floor.

$$R_{3mm} = \frac{U}{I} = \frac{4}{0.38 * 0.5 * 10^{-3}} = 21\ kOhms$$

$$R_{5mm} = \frac{U}{I} = \frac{4}{0.18 * 0.5 * 10^{-3}} = 44.5\ kOhms$$

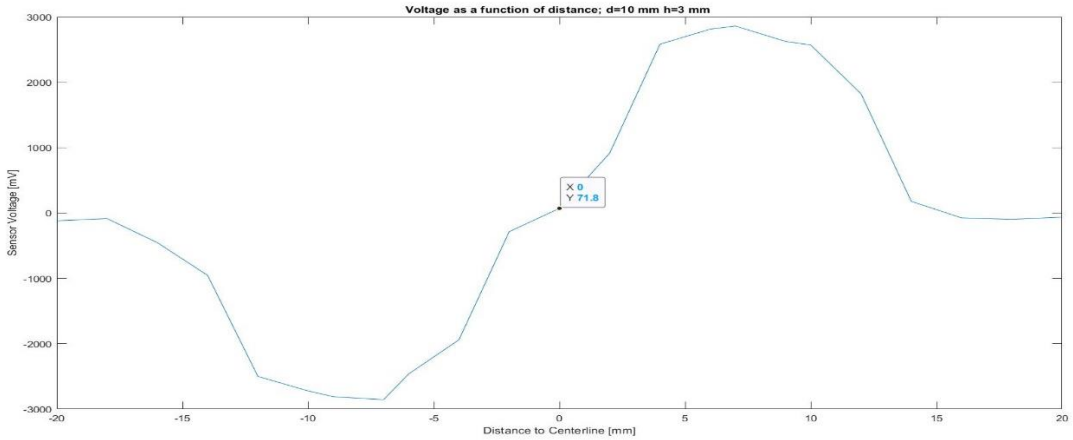
$$R_{8mm} = \frac{U}{I} = \frac{4}{0.063 * 0.5 * 10^{-3}} = 127\ kOhms$$

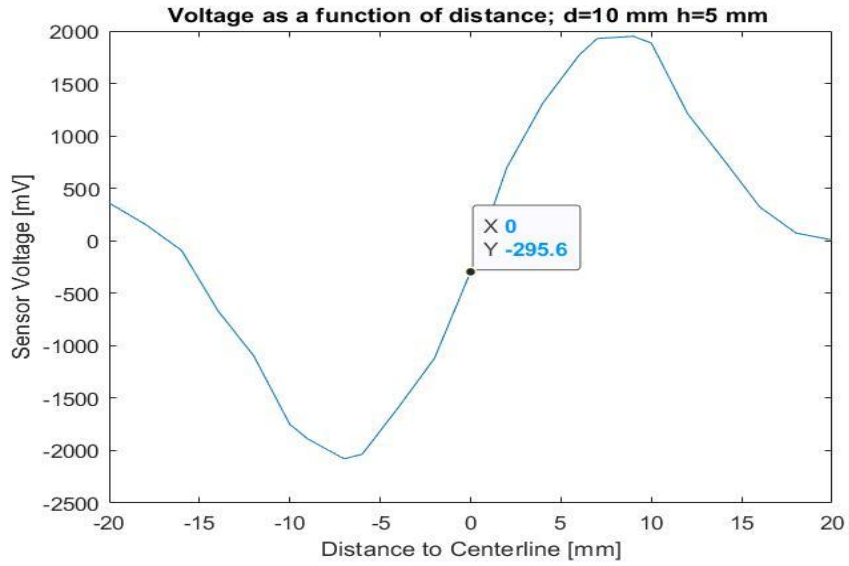
Question 7:

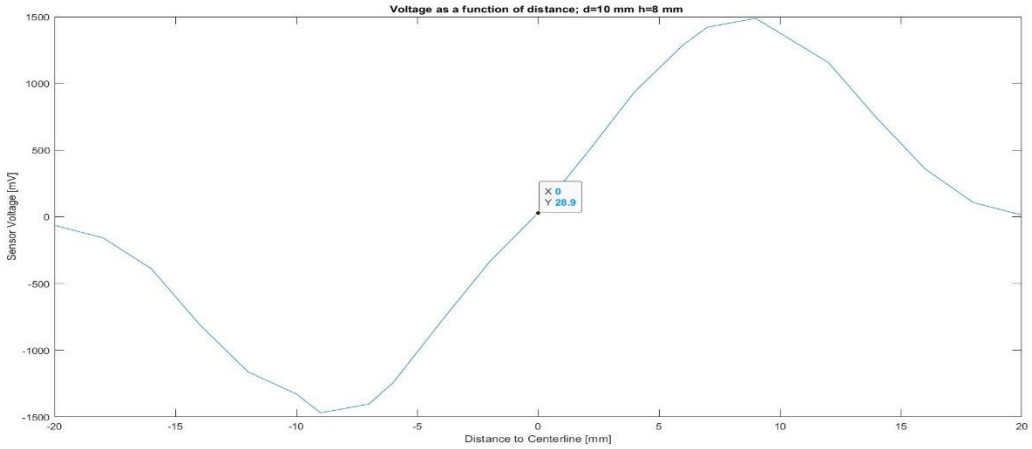



```
U_5mm=OpticalSensorMeasurementsS2(:,3).*22
I_5mm=U_5mm./R2
I1_5mm= interp1(s, I_5mm, s+d/2, 'linear', 'extrap');
I2_5mm= interp1(s, I_5mm, s-d/2, 'linear', 'extrap');
voltage_difference_5mm=((I1_5mm-I2_5mm).*R2+5)/2 % Based
on Equation from Question 5.b
```

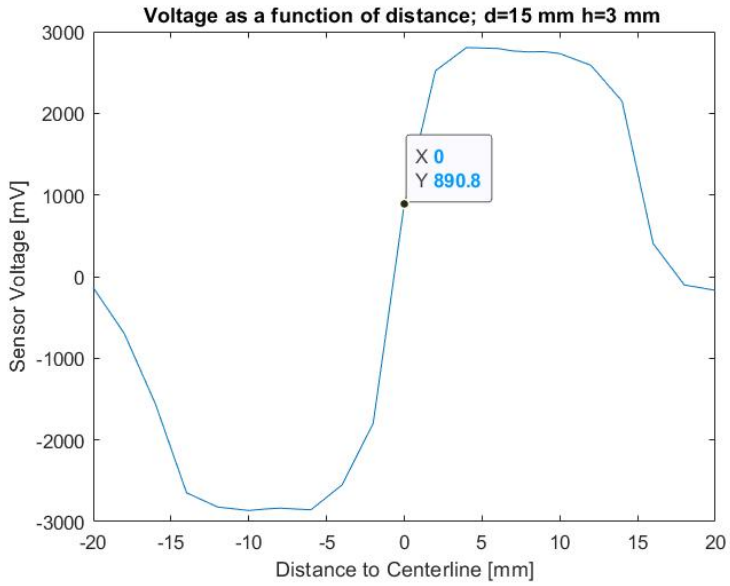
Question 8:
For d=10

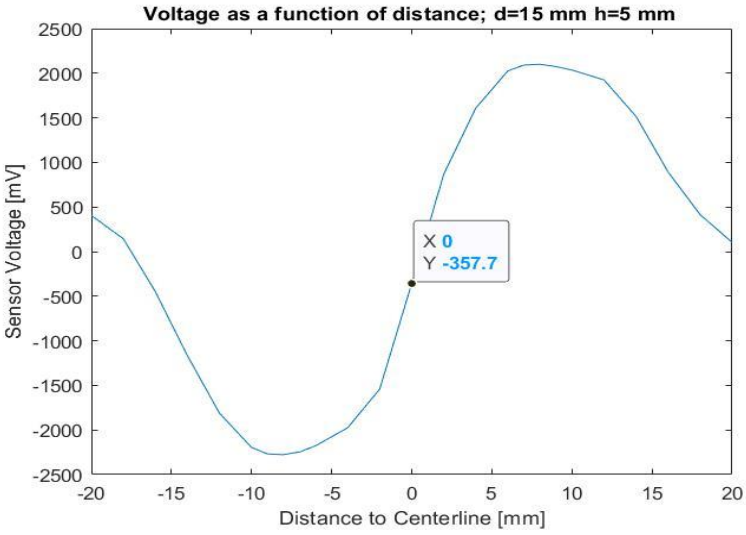


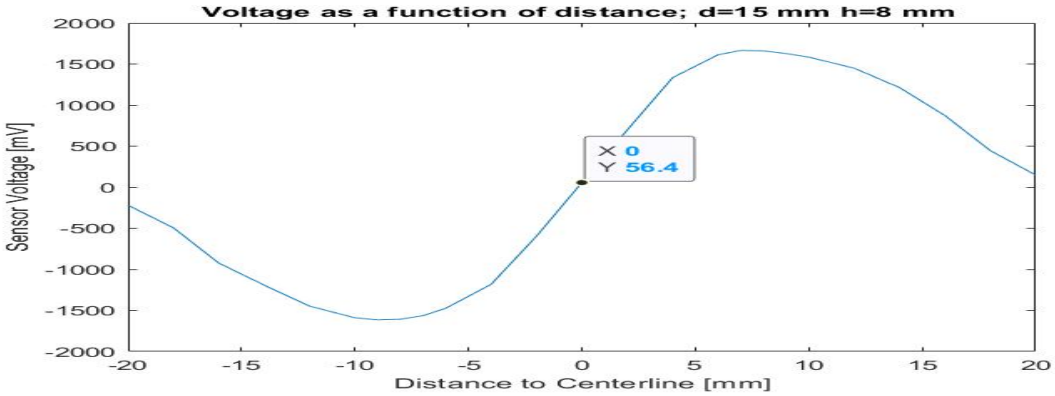

Voltage as a function of distance; d=10 mm h=5 mm
Sensor Voltage [mV]
Distance to Centerline [mm]
X 0
Y -295.6


Voltage as a function of distance; d=10 mm h=8 mm
Sensor Voltage [mV]
Distance to Centerline [mm]
X 0
Y 28.9

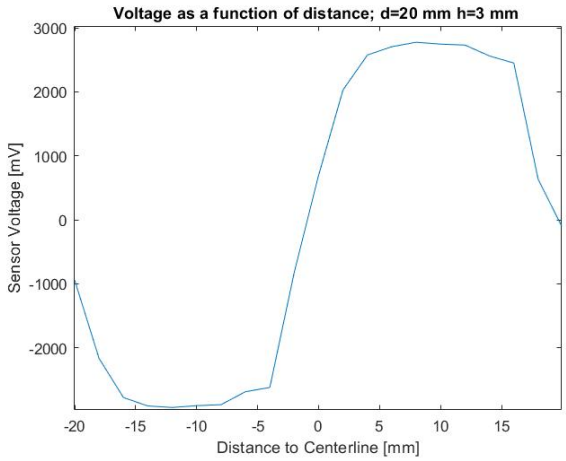
For d=15 mm

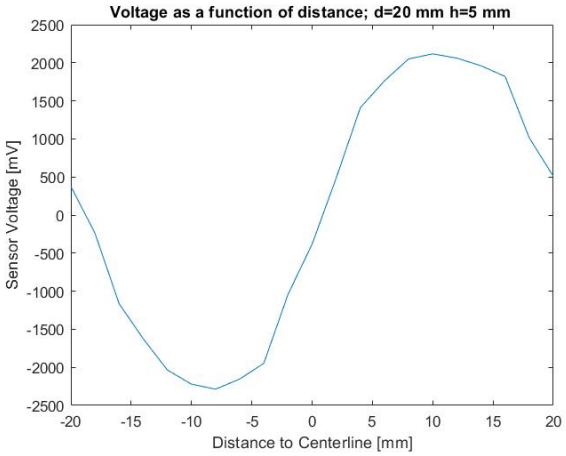

Voltage as a function of distance; d=15 mm h=3 mm
Sensor Voltage [mV]
Distance to Centerline [mm]
X 0
Y 890.8

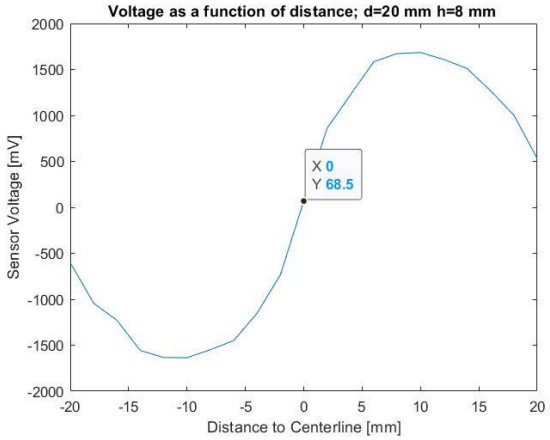

Voltage as a function of distance; d=15 mm h=5 mm
Sensor Voltage [mV]
Distance to Centerline [mm]
X 0
Y -357.7


Voltage as a function of distance; d=15 mm h=8 mm
Sensor Voltage [mV]
Distance to Centerline [mm]
X 0
Y 56.4

For d= 20 mm


Voltage as a function of distance; d=20 mm h=3 mm
Sensor Voltage [mV]
Distance to Centerline [mm]





Theoretically the optimal sensor, will change the voltage at a high rate for a minimal deviation in distance. Which means that the plot with the greatest slope around 0 mm has the optimal specs. However, in practice this wasn't ideal because the steering was very shaky and almost oscillating between left and right. Therefore, the most favorable sensor design was the one with greatest distance between the 2 sensors because then the car had a greater room for error without losing the black tape. Therefore our designed had d=30 mm and h = 5 mm.

Question 9:

Q9 has been implemented as a Simulink model and is under SVN.

Optical_Sensor_Lookup_Table_a.slx

Q9.m

Chapter 5 Validation

5.1 Ultrasonic sensor

In order to make the SMARTcar find a parking spot and park at the end of the track we are using Ultrasonic sensor. This sensor sends pulses which are reflected by the nearest object they reach. The sensor then listens for the reflected signal and calculates the distance by the time needed for the pulses to reach the object and return. The assumed speed is the speed of sound 340 m/s. The sensor doesn't take into account the differences in temperature and its range is from 2 to 200 cm. However, for our purpose high accuracy is not needed so this small deviation can be neglected.

Figure 1. Ultrasonic sensor connected to Olimex

In order to make the sensor work it needs to be connected to 5V supply, Trigger, Echo and Ground. The trigger needs to be high for 10 uS or more in order to send 8 pulses with 40 kHz. For this reason we use a Limited Counter block (limited to 10). It is compared with the value of 10 and the output, which is Boolean, is sent to the Digital Output connected to the trigger pin. (The model can be seen in the picture below).

In order for the sensor to listen for the reflected signals we use the only available timer we had left Timer 3. The duty cycle from the timer is used to calculate the distance from the object.

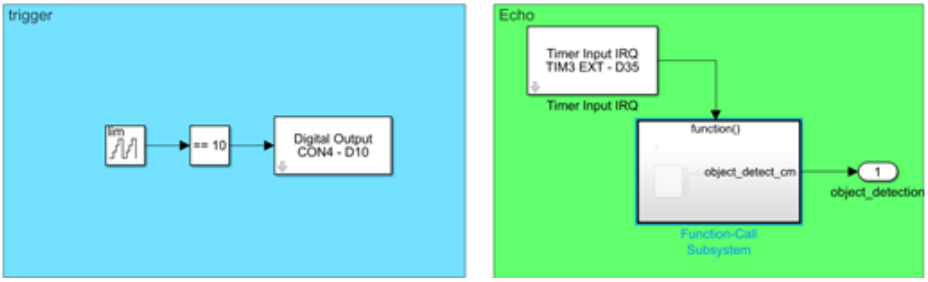


Figure 2. Ultrasonic sensor model

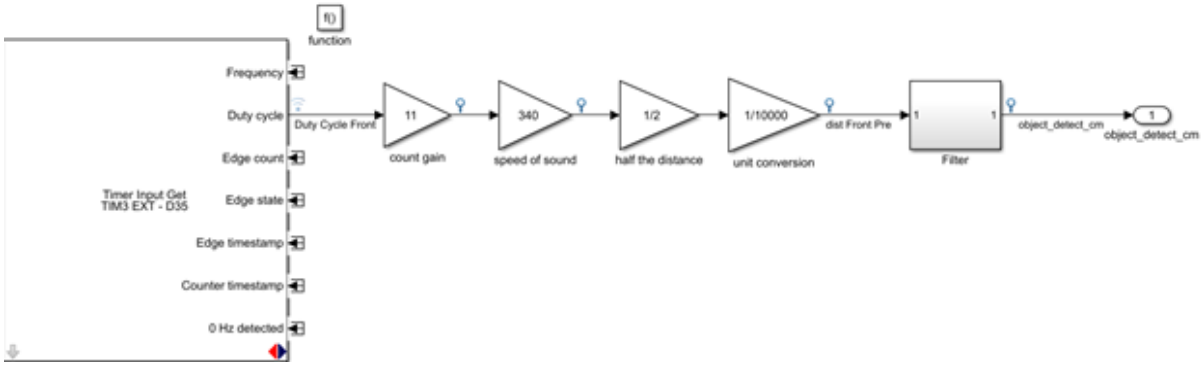


Figure 3. Ultrasonic sensor distance calculation with echo

After making the model and connections it was all tested with HANtune. To check if the readings of the sensor are right we used the box of the SMARTcar and a ruler. The accuracy of the sensor turned out to be less than +-0.5 cm for the range that we are going to use, which is enough. Example of the test made can be seen in the pictures below.

Figure 4. Comparing the distance between the ultrasonic sensor and the box to the real distance

Real distance [cm]	5	10	15	20	25	30	35
Ultrasonic distance[cm]	5.32	10.05	15.12	20.19	25.25	30.20	35.29

Table 1. Measured values with the ruler versus with the ultrasonic sensor

In order to be sure that the sensor will also work as intended in the car we mounted it on the car and connected it. We wanted to see if the car will be able to detect a free space of 60 cm length and perform a certain action after that. For simplicity we made the car switch to reverse after finding the free spot.

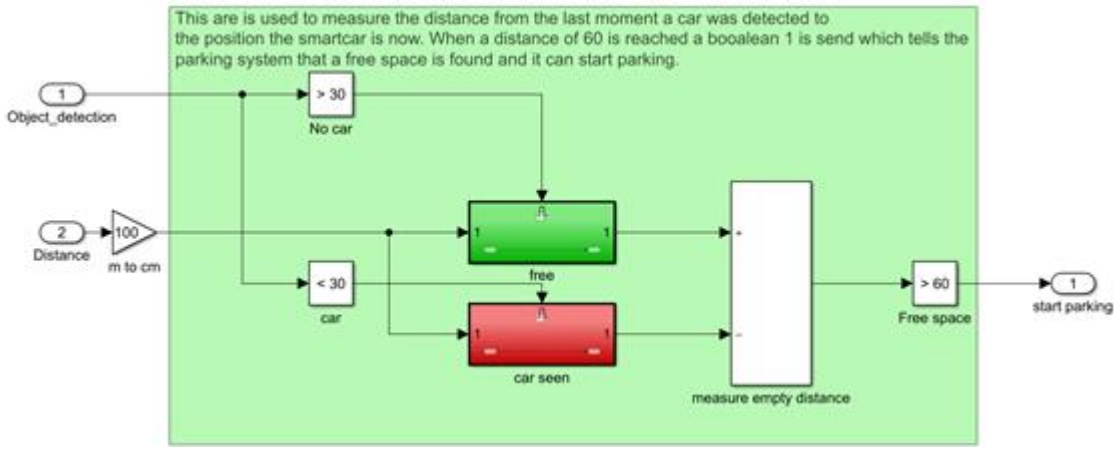
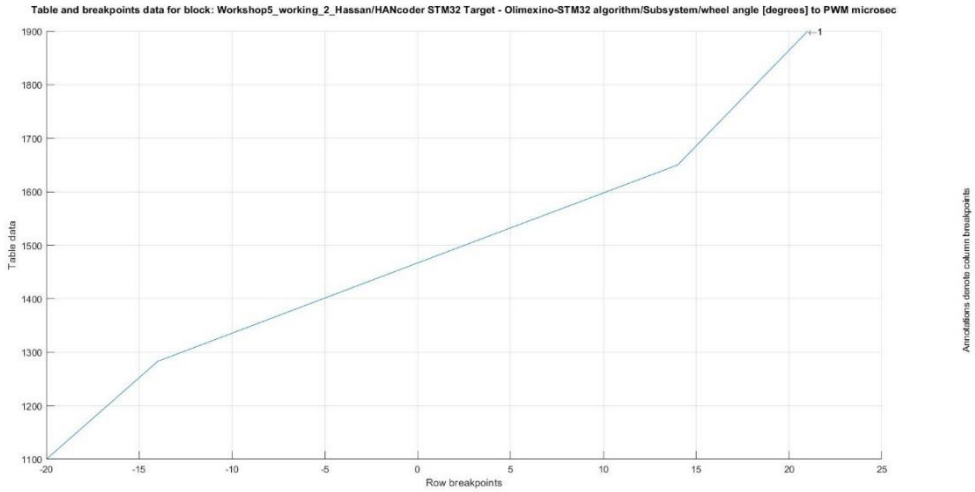


Figure 5. Simple algorithm to check the working of the ultrasonic sensor when mounted

We checked the working a couple of times with different distances between the boxes and also different positions of the free spot and all of them worked correctly, which meant that the sensor was validated completely.

5.2 Steering Validation

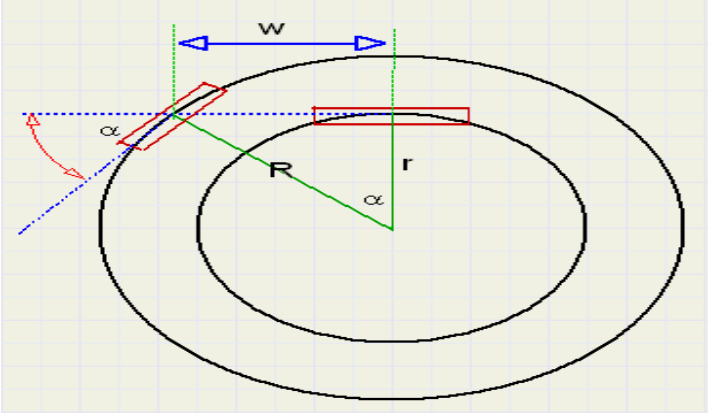
In the software model, the steering is described by the kinematic model for a 2-wheel vehicle. This model is relatively simple and assumes that the velocity at each wheel is in the direction of the wheel. However, our smart car has 4 wheels, which causes a difference especially at a low cornering radius. Also, the validity of this model is limited to relatively low speeds. For higher speed, a dynamic model for lateral motion should be assumed.
To ensure the correctness of the steering subsystem in the software model, the following aspects were validated:
1. Does the cornering radius of the smart car match the steering angle of the wheels as described in the kinematic model?
This is validated by measuring the cornering radius as a function of the wheel angle in the complete range of steering angles. The maximum steering limited by hardware design is +-20 degrees as shown in this lookup table.



In the tables found below, the cornering radius has been tested at different steering angles from left to right. The measured results are considered to match the simulation results since they only deviate by a small error. However, it is observed that there is a slight steering bias to the left. Perfectly aligning the servo should get us closer to the simulation results.

Simulation Testing Condition	Hardware Testing Conditions
Velocity = 1 m/s	Velocity = 1 m/s
Friction Enabled	Friction Present

Measurement Setup:
A constant speed was set and maintained using the PI speed controller, and a constant steering angle could be set from HANtune. Then the car ends up rotating in a circle as shown below. Then diameter of this circle is measured and then the cornering radius is obtained.



Wheel Steering Degrees	Cornering Radius in Simulation	Cornering Radius Measured	% error
6 Degrees Left	247.5 cm	263 cm	6 %
12 Degrees Left	122.3 cm	130 cm	6 %
18 degrees left	99 cm	109 cm	9 %

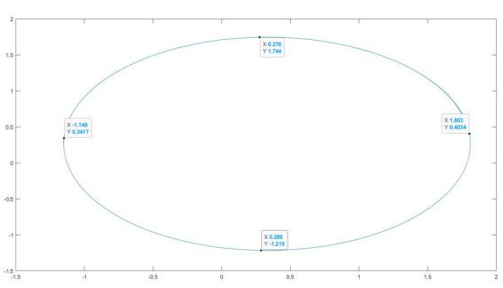
Wheel Steering Degrees @ V=1 m/s	Cornering Radius in Simulation	Cornering Radius Measured	% error
6 Degrees Right	247.5 cm	257 cm	4 %
12 Degrees Right	122.3 cm	126 cm	3 %
18 degrees Right	99 cm	105 cm	6 %

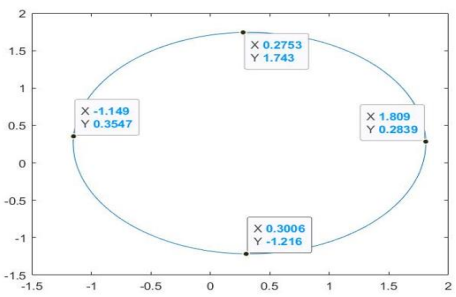
2. Is the cornering radius independent of the vehicle speed as assumed in your software model?

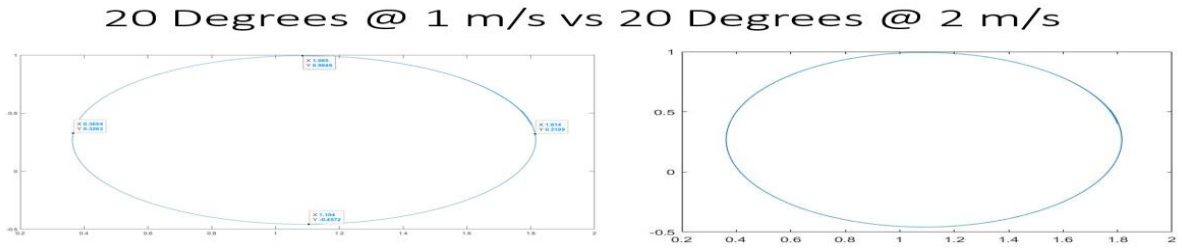
This is validated by measuring the cornering radius as a function of speed while maintaining constant steering.

The cornering radius is shown to be independent of vehicle speed in the simulation results below.

10 Degrees @ 1 m/s vs 10 Degrees @ 2 m/s







However, when measured at different speeds the cornering radius increases as a function of speed.

Wheel Steering Angle	Cornering Radius @ 1 m/s	Cornering Radius @ 2 m/s
10 Degrees left	1.54 m	1.67 m
20 Degrees left	0.82 m	1.13 m

The cornering radius increases slightly, therefore It can be assumed that the simulation sufficiently describes the motion of the vehicle. However, this validity is limited to low speeds since the Kinematic model we are using doesn't take in account dynamics of lateral motion.

6.0 Battery Management System

A battery management system can be used for different purposes but given the limited time we had we decided to limit ourselves into being able to measure the voltage per cell. That way we can make sure that if the discharge of the battery is not balanced between cells, we will not kill our battery. Also, it can be useful when testing the SMART car for a long amount of time and we can check if all the cells are at their safe voltage range even if the total voltage is not much. Then we will be able to extend our testing time.

To implement this idea, we researched on how the BMS connector is connected. It was interesting to see that pin Vb2 was a measurement of all the cells, Vb1 of two cells and Vb0 of one cell. Therefore, we need to subtract the measurement values in order to see each cells voltage. How we did it can be seen I our model.

In order to read the values using the olimex, we wanted to make sure that we don't exceed the maximum voltage of the pins. Therefore, we used voltage dividers. The goal was that every measurement value to be max at 2.1 volts.

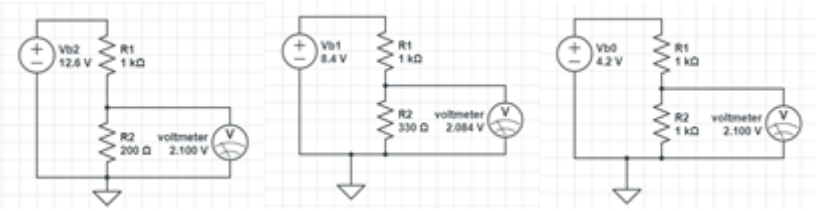


Figure 7. Voltage dividers for every measurement pin.

In order to be sure that the resistors we use will not cause a problem we calculated the power dissipation per resistor. The example calculation below shows the resistor which had the highest power dissipation.

$$I = \frac{U}{R\_total} = \frac{12.6}{1200} = 0.0105\,A \quad P = I^2 * R1 = 0.11025\,W$$

The resistors we used are rated to 0.25 W, so we are able to use them. All other resistors had way smaller power dissipation.

By using a multimeter, we validated that the measured voltage is the same as the simulated one. In software we converted the values from xadc to voltage and multiplied the values with gains, so we get the real ones. More detailed description can be seen in the picture below.

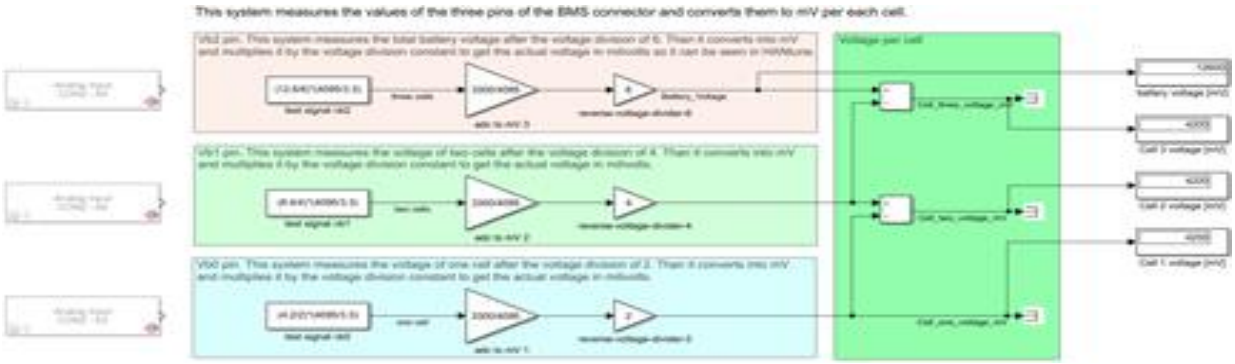
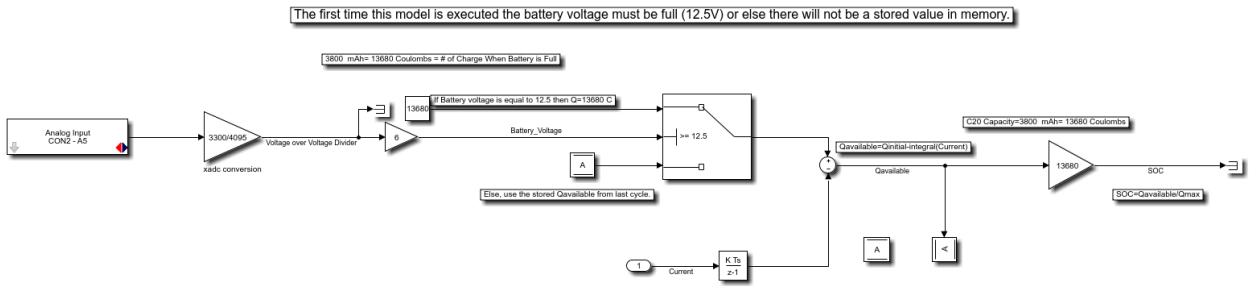


Figure 8. Screenshot from the BMS software model. We used test signal to check how it works.

Battery SOC

Since knowing the available charge in a battery is problematic, the problem is simplified by assuming that when battery is full Q=3800 mAh=13680 C, thus when the battery is on Q=13680-Qout. The charge going out of the battery is obtained by integrating the current as a function of time.



Important Files Under SVN:

Final Challenge Hardware Model:

Final_working_2.slx Workshop5_startup.m

Final Challenge Simulation Model:

Final_challenge_model_Nikolay.slx Model_Script.m

Question 9 Optical Sensor:

Q9.m Optical_Sensor_Lookup_Table_a.slx

BMS:

BMS_SOC_Hassan.slx