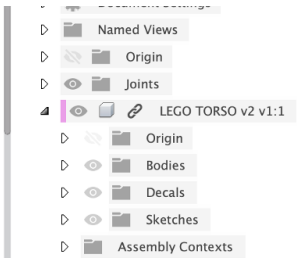
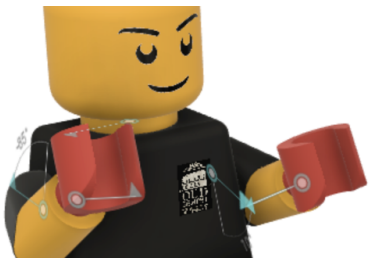
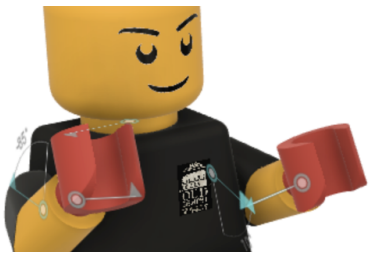
IoT Project: Self-Driving-Car





Koen Verbeeck

Table of Contents



Koen Verbeeck

Description Assignment

Mercedes-Benz wants to automate its museum in Stuttgart. To do this, they want a system, which will allow a visitor to take a seat in a self-driving car. At the beginning of the tour, the visitor can take place in one of these cars that will drive around the entire museum. At the end, the visitor gets out and the trolley returns to the beginning of the tour so that new visitors can embark.

The route that the cars have to follow is fixed with a white tape on the floor, which makes it possible to adapt the tour if new exhibits are added. At each exhibit, a transverse white line will cross the tour line, this way the car knows where a museum piece is exhibited and that it has to stop. If the driver wants to continue, he or she can press a push-button inside the car and the tour will continue to the next exhibited item.

To avoid congestion, the car will automatically continue the tour after 5 minutes. Optionally, they also want to operate the cars remotely, for which the Control Center could take over the function of the push-button with an app.

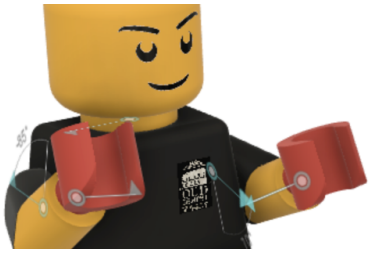
Of course it is also important that the cars do not collide, therefore there should always be a minimum distance of 1.5 meters between the trolleys (Corona measures). If a car detects that it is too close to the one in front, it must stop and only continue if the one in front also continues to drive.

The cars are also equipped with LEDs. Using this signalization, the staff present in the control room can check cameras should there be a problem.

- A green LED will light up if there are no issues.
- A red LED will light up if the car has lost track of the white line.
- A orange LED will light up if the car detects an obstacle (other car or anything else in its path).

Requirements:

- The car can autonomously follow a track indicated by a white line.
- The car must stop at a white line and can only continue driving if:
    - The driver pushes the drive-through push-button.
    - The trolley has been stationary for 5 minutes.
    - Optional: when the control center activates the car with their custom made application.
- The cars must not collide.
    - If a car is halted, the cars behind it will keep their distance and only continue their route when the car in front of them continues its path.



- Should there occurre problems with a car, it will be forced to stop and inform the control center by means of LEDs mounted on the cars.
    - Signalization in case of problems:
        - Green LED: No issues.
        - Orange LED: Obstacle (other car or anything else in its path).
        - Red led: car has lost track of the white line.

What does Mercedes-Benz expect:

- A brief description of the used hardware.
    - Files to produce the enclosure.
    - Electronic components.
    - Schematics and PCB (if applicable).
- Documented code.
- Brief description of to produce these cars themselves.
    - Tutorial.

All of this will be provided to them through a GitHub page.

Follow-up:
Mercedes-Benz wants to take the cars into production after 7 weeks, there will be 3 evaluations.

In 3 weeks time the car should be driving and have basic functionality (line following).

In 5 weeks they want to be sure that the cars have the correct functions to :

- Follow the white line.
- Stop and continue at each transverse white line.
- Signalization.

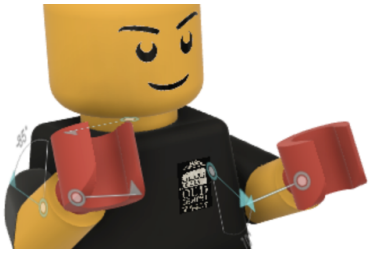
In 7 weeks a working prototype should be presented. This prototype will be tested for functionality and speed. The prototype may not contain any loose components, so the use of a breadboard is not allowed. At this time, all documentation should also have been provided.

Github Page with Project Details

https://github.com/kverbeeck/IoT_Project_Line_Following_Car

YouTube Video Project Demo

https://www.youtube.com/watch?v=Afq3bGS6EhM

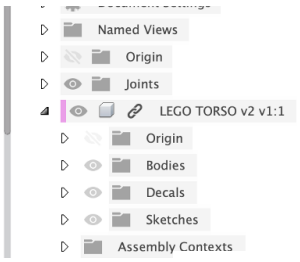


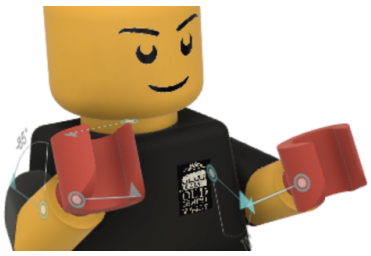
Koen Verbeeck

Required Hardware

Below the required hardware for this project, of course you are free to buy your components wherever you like.

	Item Name	Quantity	Price €	Item Description
1	LEGO 42093	1	39.99	Lego 42093 Technic Chevrolet Corvette ZR1
2	2 x L298N and 4 x DC-motors	1	12.99	KYYKA Motor Drive Controller (2x) for Arduino Dual H Bridge DC-steppermodule L298N with 4 DC-engines
3	2 x 18650 Battery	1	15.89	Battery Double Pack US18650VTC5A (KONION) - MURATA 18650
4	Mega 2560 Microcontroller Board	1	13.79	SUNFOUNDER Mega 2560 R3 ATmega2560 ATmega16U2 Microcontroller Board Compatibel met Arduino Genuino MEGA 2560
5	Servo HS-422	1	12.04	Hitec RCD 31422S HS-422 Deluxe Servo
6	Mini-Traffic Light	1	3.99	AZDelivery Led-lightmodule Creative DIY mini-traffic light 3,3-5 V 5 mm compatibel with Arduino!
7	Push-Button	1	7.99	RUNCCI-YUN Momentary Tactile Push Buttons, 12 x 12 x 7,3 mm, Micro Momentary push-buttons, 4-pins microswitches for Arduino, 80 pieces
8	Resistor-Set	1	8.99	AZDelivery Resistor-set 525 pieces resistors assortiment, 0 ohm -1 Mohm
9	Capacitors-Set	1	11.99	BOJACK 15 Wert 600 Stück Keramik kondensator 10pF 20pF 30pF 47pF 56pF 68pF 100pF 220pF 330pF 680pF 1nF 4.7nF 10nF 47nF 100nF Sortiments kit
10	TRCT5000	1	4.99	AZDelivery 3 x TRCT5000 IR infrared line-tracking module compatibel with Arduino
11	LED EL 5-7150WW	2	0.31	LED EL 5-7150WW LED, 5 mm, leaded, warm white, 7150 mcd, 50 °
12	LED EL SF 14RT	2	0.24	LED EL SF 14RT Super-Flux-LED, 7.6x7.6 mm, 70 °, red, 8660 mcd, 85 °
13	HC-SR04	1	3.99	AZDelivery HC-SR04 Ultrasonic Module Rangefinder Sensor for Raspberry Pi and Arduino
14	ESP-01	1	5.29	AZDelivery esp8266 ESP-01S WiFi Parent module
14	**Total**		**130,24**	





Koen Verbeeck

Links to hardware

Lego 42093 Technic Chevrolet Corvette ZR1

KYYKA Motor Drive Controller (2x) for Arduino Dual H Bridge DC-steppermodule L298N with 4 DC-engines

Battery Double Pack US18650VTC5A (KONION) - MURATA 18650

Hitec RCD 31422S HS-422 Deluxe Servo

AZDelivery Led-lightmodule Creative DIY mini-traffic light 3,3-5 V 5 mm compatibel with Arduino!

RUNCCI-YUN Momentary Tactile Push Buttons, 12 x 12 x 7,3 mm, Micro Momentary push-buttons, 4-pins microswitches for Arduino, 80 pieces

AZDelivery Resistor-set 525 pieces resistors assortiment, 0 ohm -1 Mohm

BOJACK 15 Wert 600 Stück Keramik kondensator 10pF 20pF 30pF 47pF 56pF 68pF 100pF 220pF 330pF 680pF 1nF 4.7nF 10nF 47nF 100nF Sortiments kit

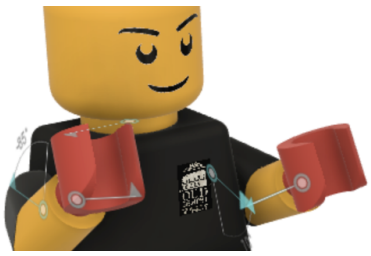
AZDelivery 3 x TRCT5000 IR infrared line-tracking module compatible with Arduino

LED EL 5-7150WW LED, 5 mm, leaded, warm white, 7150 mcd, 50 °

LED EL SF 14RT Super-Flux-LED, 7.6x7.6 mm, 70 °, red, 8660 mcd, 85 °

AZDelivery HC-SR04 Ultrasonic Module Rangefinder Sensor for Raspberry Pi and Arduino

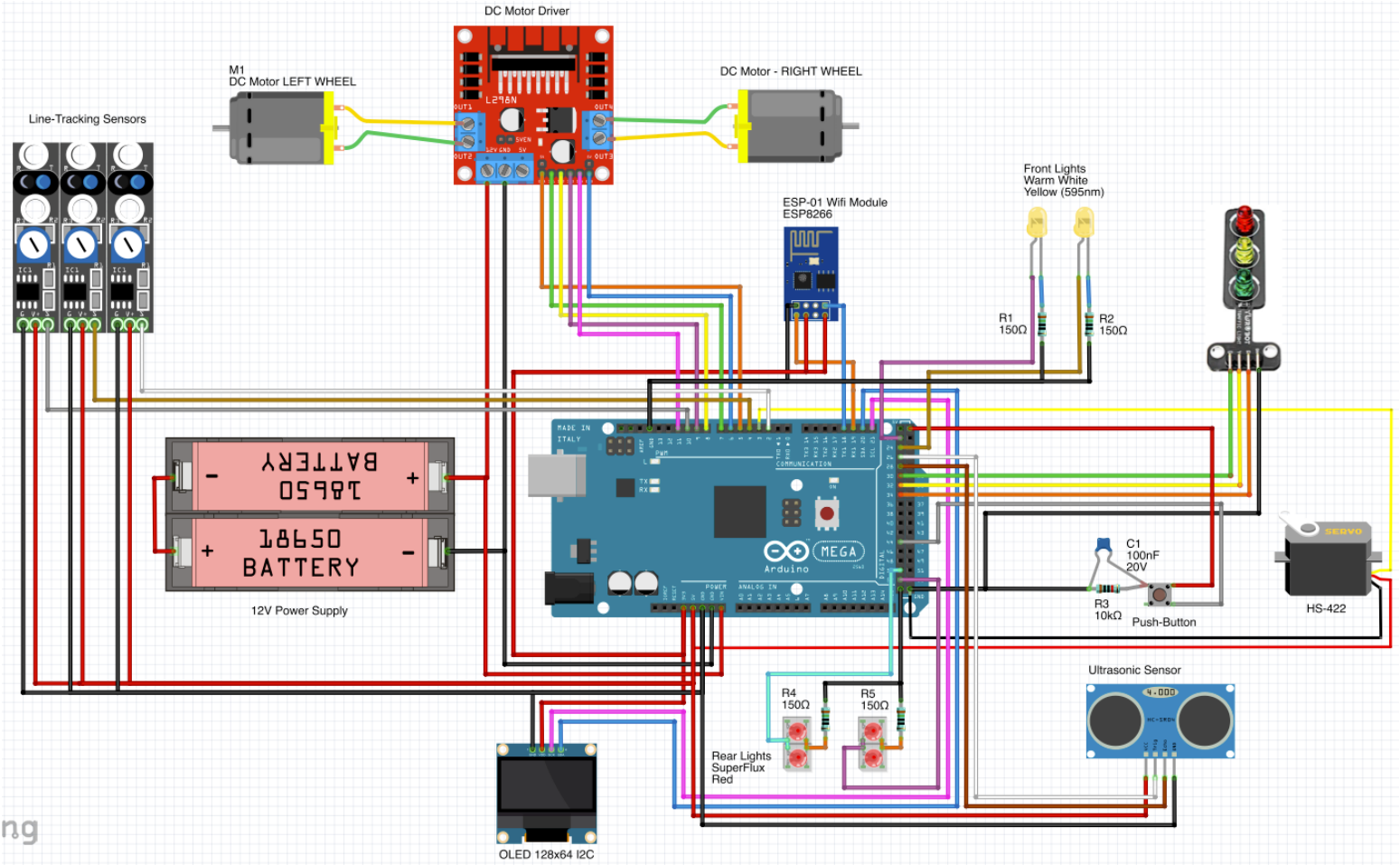
AZDelivery esp8266 ESP-01S WiFi Parent module

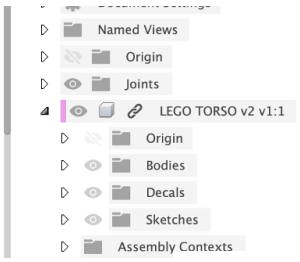


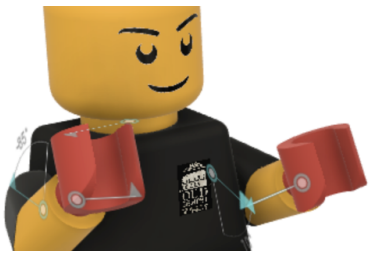
Koen Verbeeck

Schematics - Fritzing

Below the design for the electronics and their connections to the MEGA2560 Micro-Controller when using dupont wires.



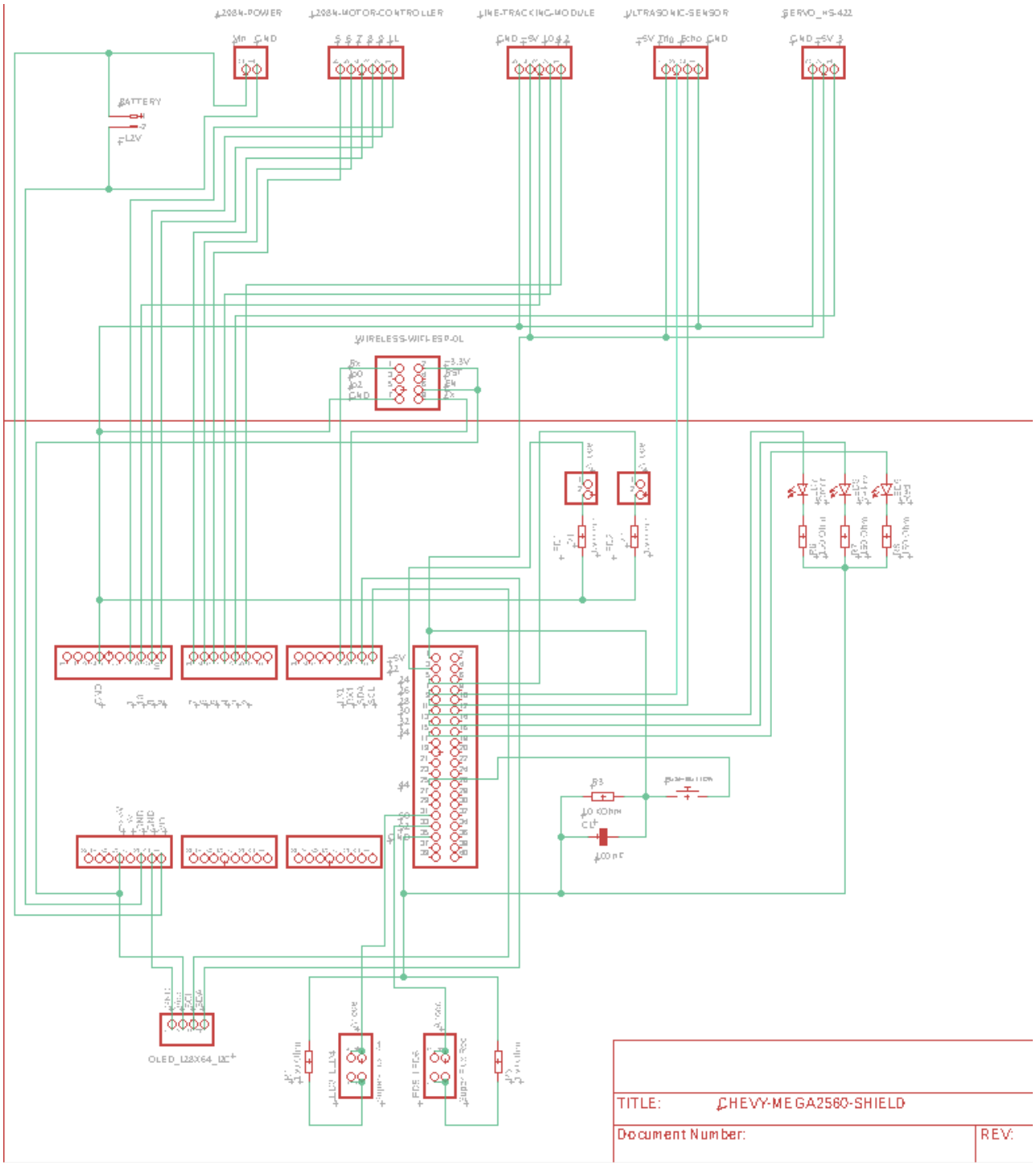




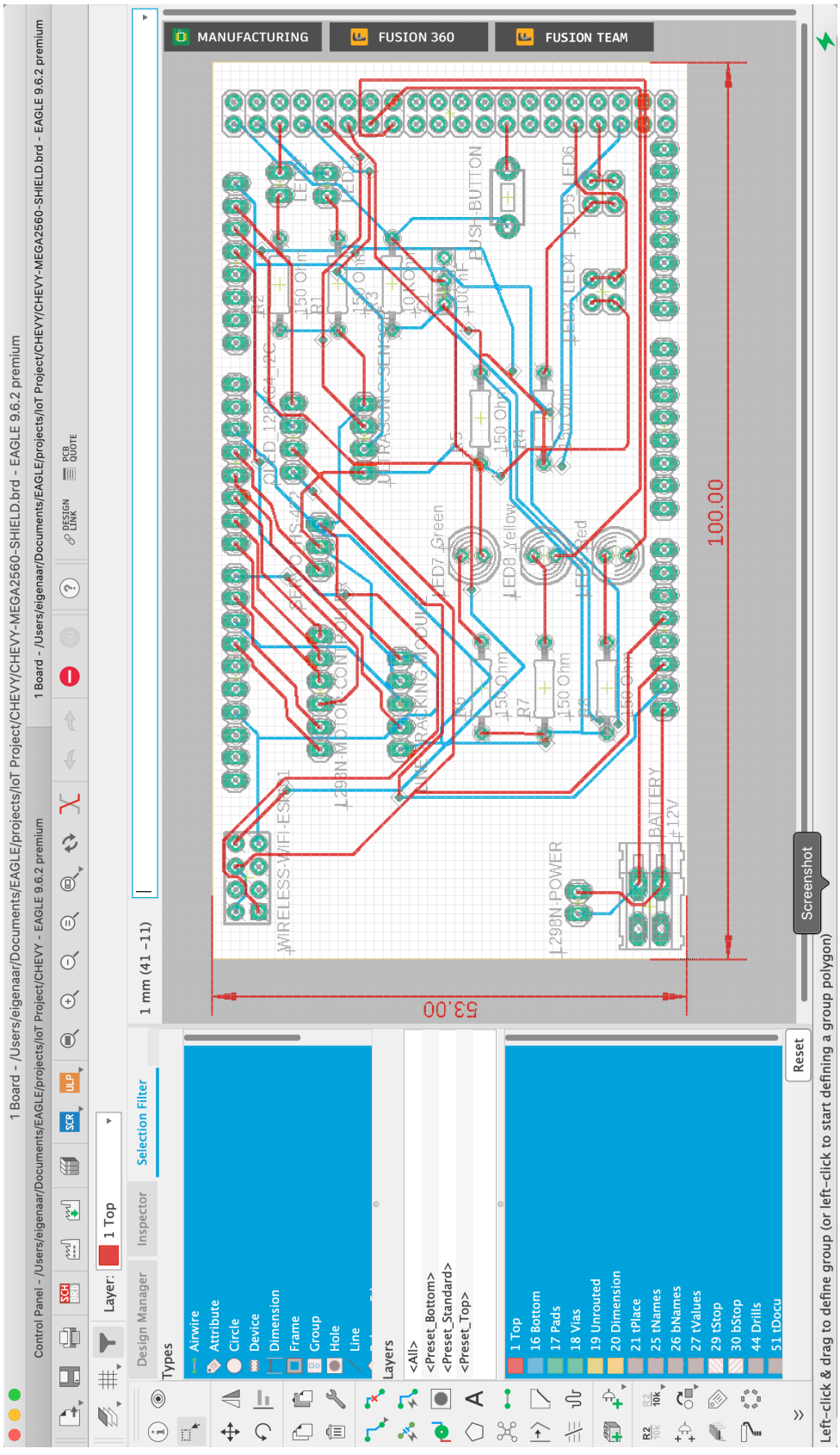
Koen Verbeeck

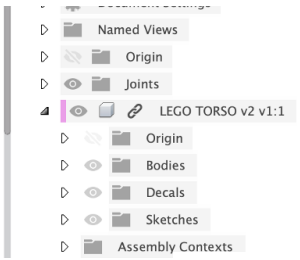
Schematics - Eagle - CHEVY

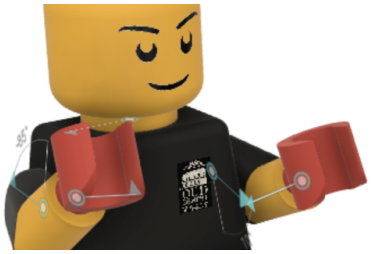
Below the designs for the PCB shield that should be attached to the MEGA2560 Micro-Controller.



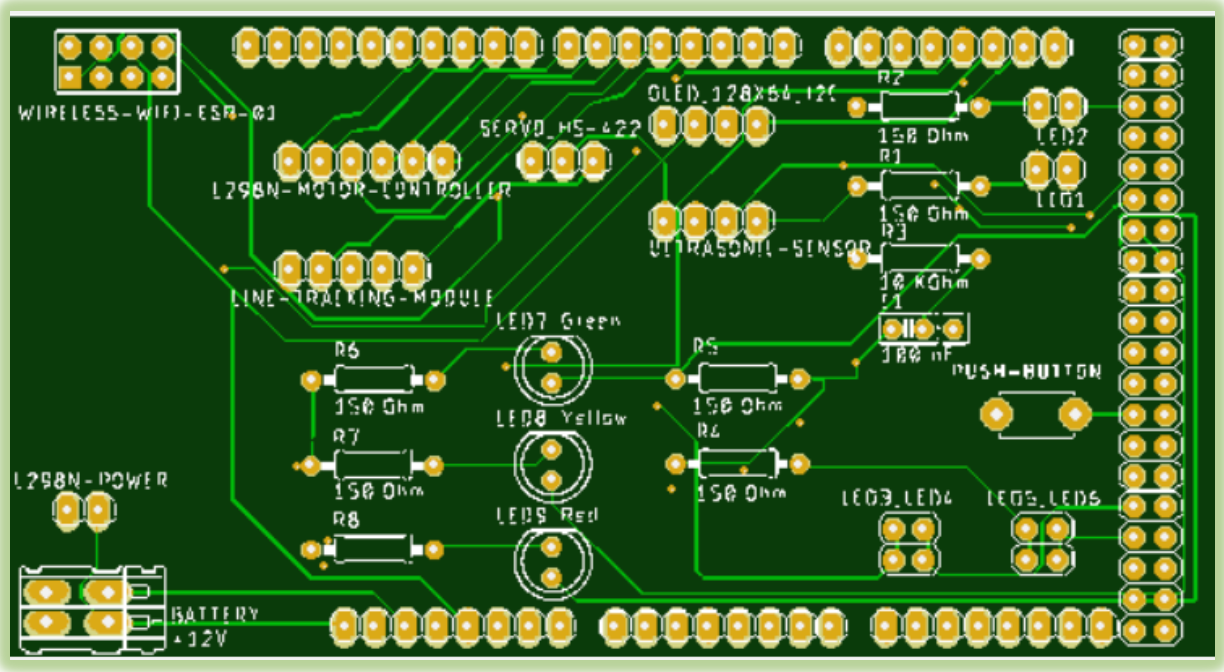
Koen Verbeeck

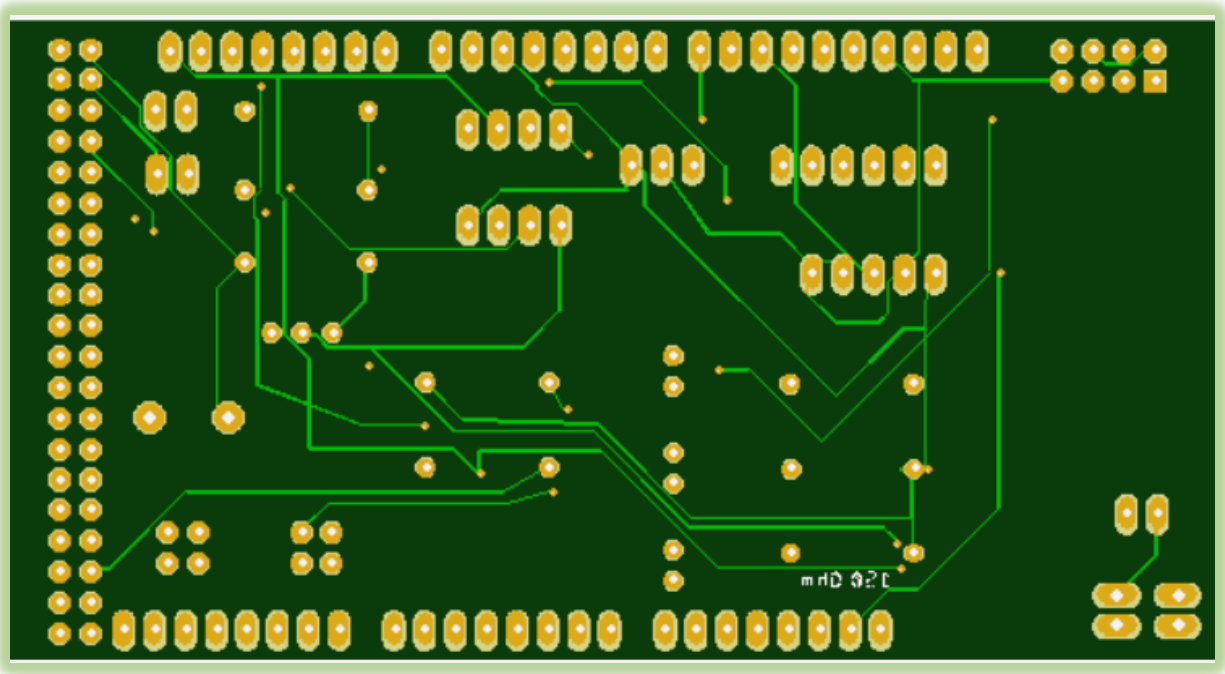


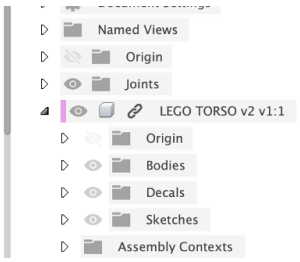


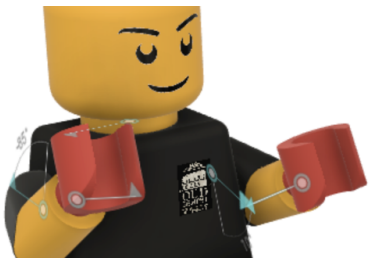


Koen Verbeeck





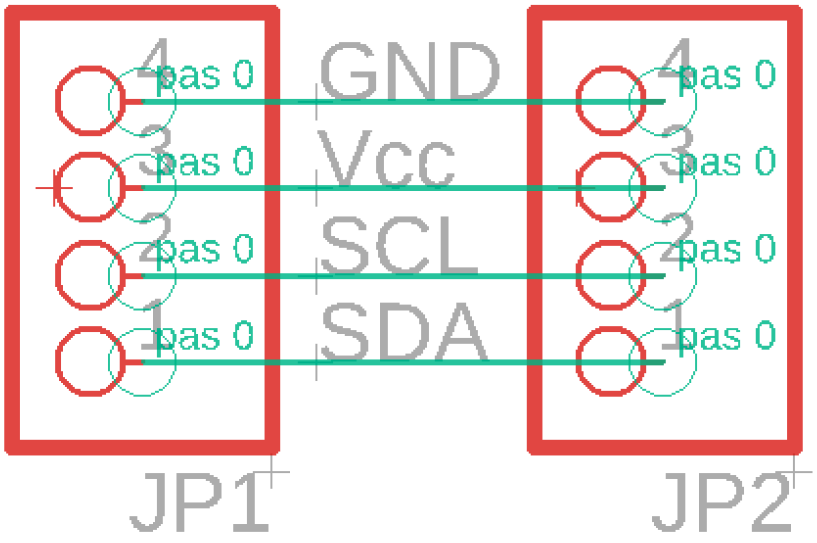


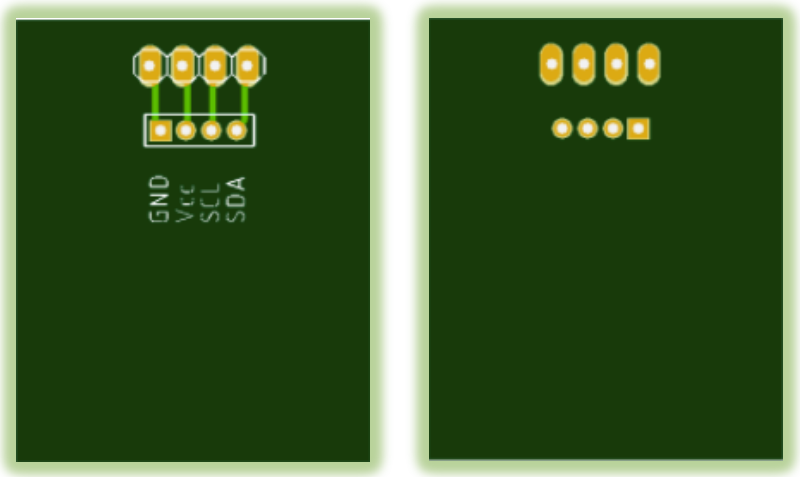


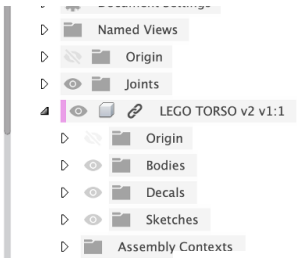
Koen Verbeeck

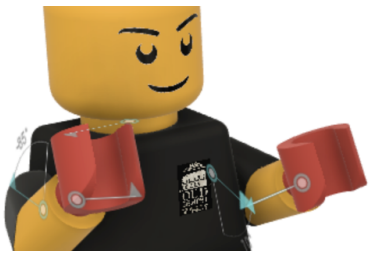
Schematics - Eagle - OLED

Below the designs for the PCB that should hold the OLED 128x64 screen.





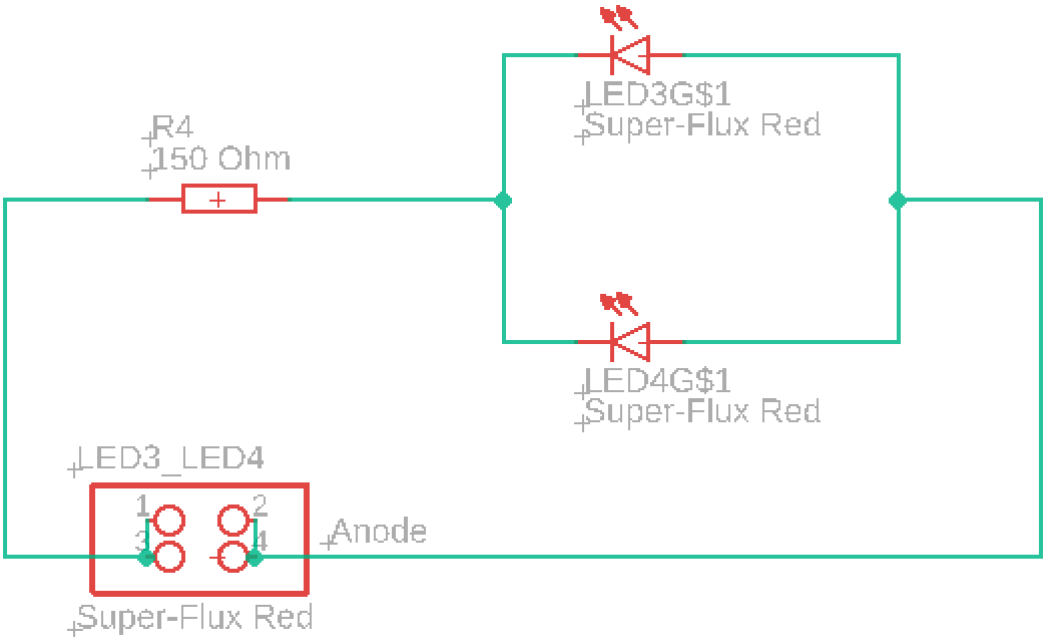


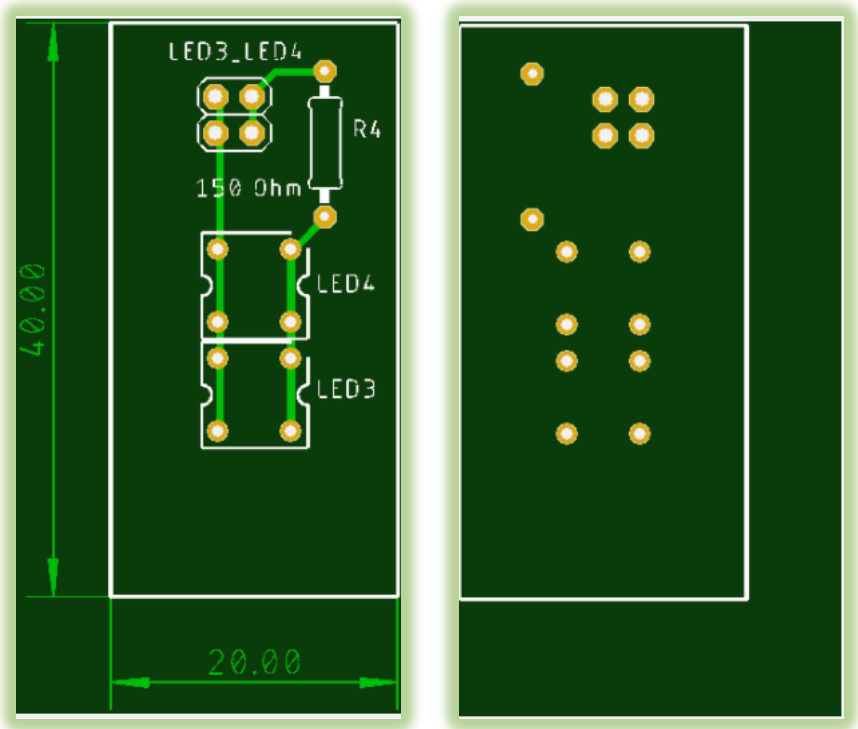


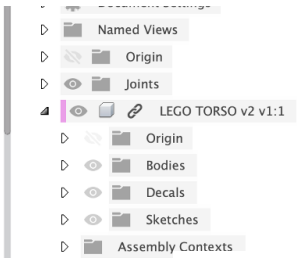
Koen Verbeeck

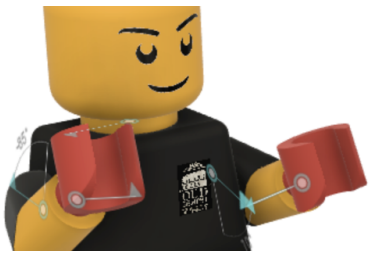
Schematics - Eagle - BACKLIGHTS LEFT

Below the designs for the PCB that should hold the Left Backlights.





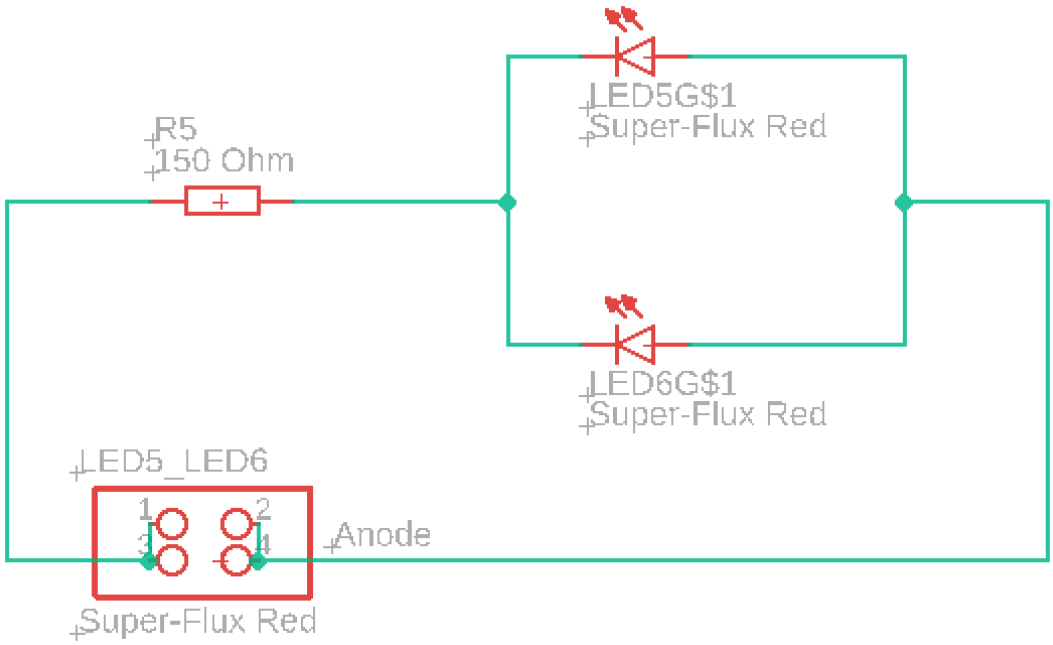


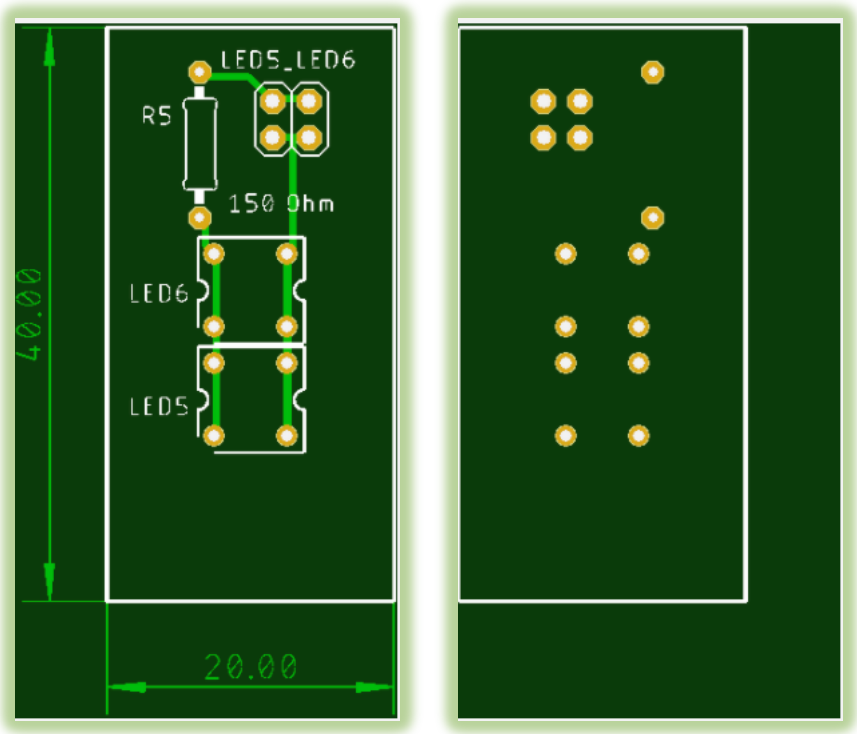


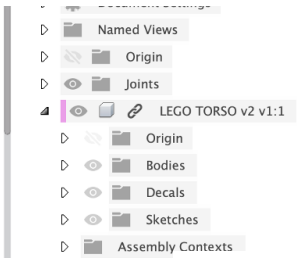
Koen Verbeeck

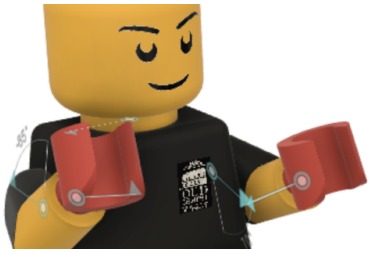
Schematics - Eagle - BACKLIGHTS RIGHT

Below the designs for the PCB that should hold the Right Backlights.





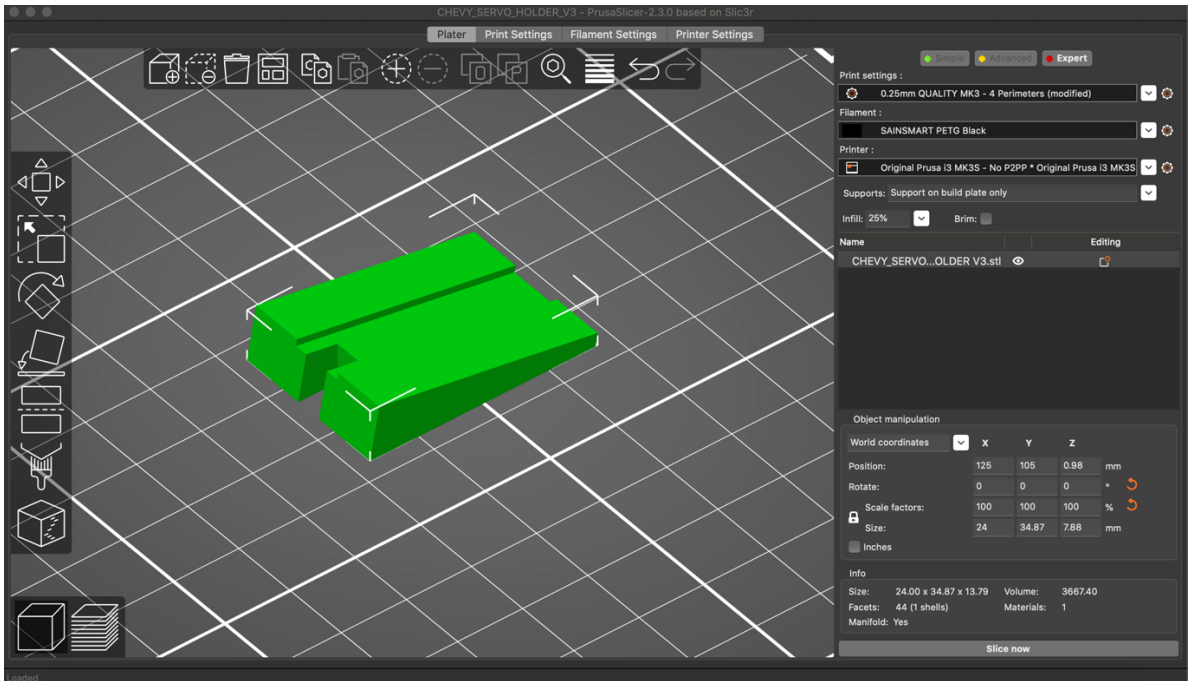


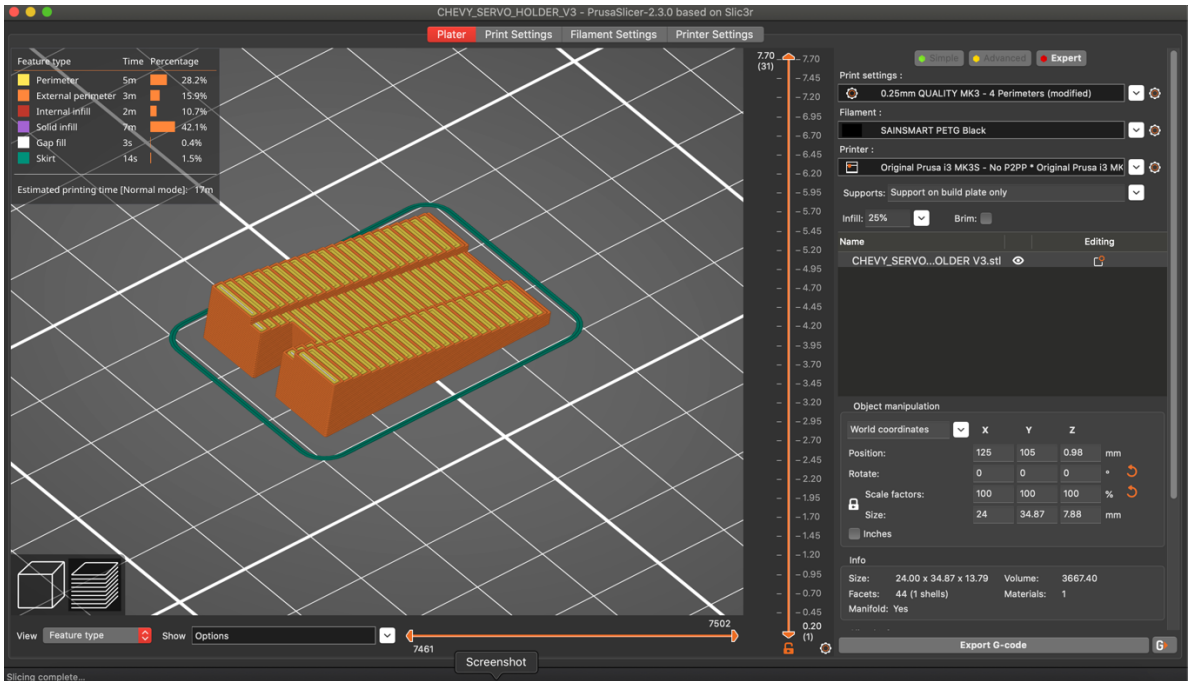


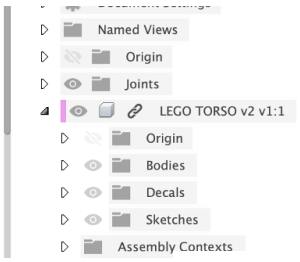
Koen Verbeeck

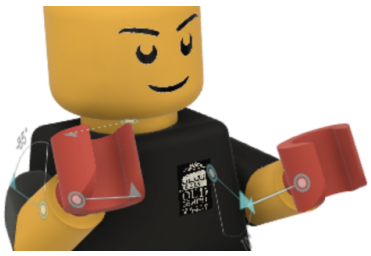
3D Printing

Following part should be printed to mount the HS-422 Servo into the car. This will tilt the Servo some degrees and provide it a snug fit with the steering gear. See my Github page for more details.







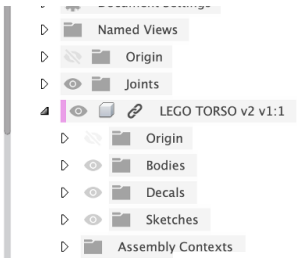


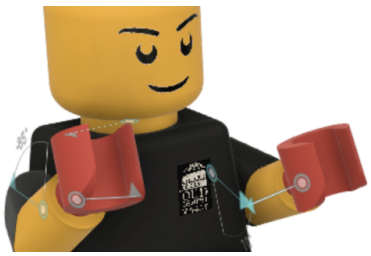
Koen Verbeeck

Code - C++

CHEVY_Turning_Using_DC_Motors_SERVO_BLYNK_WORKING.ino on Github page. See the purple comments in the code for detailed information.

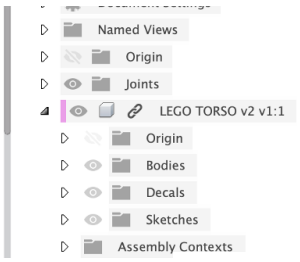
```
/*
  You will need all 4 control signals, even if you have only 1 DC motor right and 1 DC motor left !!!!

  Left Motor truth table
  ------------------------------------------------------------------------
  ENA       IN1            IN2         Description
  LOW   Not Applicable   Not Applicable   Motor is off
  HIGH      LOW            LOW         Motor is stopped (brakes)
  HIGH      LOW            HIGH        Motor is on and turning forwards
  HIGH      HIGH           LOW         Motor is on and turning backwards
  HIGH      HIGH           HIGH        Motor is stopped (brakes)

  Right Motor truth table
  ------------------------------------------------------------------------
  ENB       IN3            IN4         Description
  LOW   Not Applicable   Not Applicable   Motor is off
  HIGH      LOW            LOW         Motor is stopped (brakes)
  HIGH      LOW            HIGH        Motor is on and turning forwards
  HIGH      HIGH           LOW         Motor is on and turning backwards
  HIGH      HIGH           HIGH        Motor is stopped (brakes)

  OLED Screen 128x64 connected to MEGA2560: SDA (20); SCL (21)

  BIG NOTE*: BLYNK Server will interfere with the servo's response time, so try to avoid sending data while driving !!!!!!!!!!!!
*/

// BLYNK info in Serial Monitor (Comment this out to disable prints and save space.
#define BLYNK_PRINT Serial

/*
  Include the necessary libraries
*/
#include <Wire.h> // I2C Library
// Core graphics library for all Adafruit displays, providing a common set of graphics primitives (points, lines, circles, etc.).
#include <Adafruit_GFX.h>
```

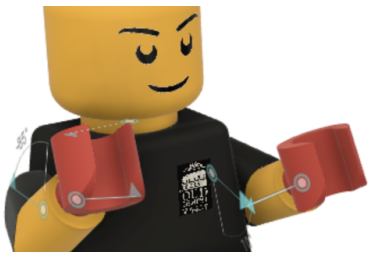




Koen Verbeeck

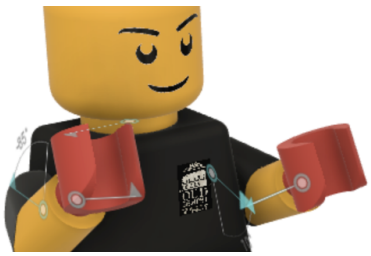
```
#include <Adafruit_SSD1306.h> // Library for Adafruit Monochrome OLEDs based on SSD1306 drivers.
// The fonts that work better with the OLED display are the 9 and 12 points size.
#include <Fonts/FreeSerifBoldItalic12pt7b.h>
#include <Servo.h> // Servo library
#include <ESP8266_Lib.h> // Library for ESP-01 Wi-Fi Module
#include <BlynkSimpleShieldEsp8266.h> // Library for ESP-01 Wi-Fi Module talking to BLYNK Server

/*
   BLYNK Server
*/
char server[] = "192.168.10.122";
#define port 8080

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "**********************************";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "***********";
char pass[] = "**********";

// Hardware Serial on Mega, Leonardo, Micro...
#define EspSerial Serial1

// or Software Serial on Uno, Nano...
#include <SoftwareSerial.h>
//SoftwareSerial EspSerial(2, 3); // RX, TX

// Your ESP8266 baud rate:
#define ESP8266_BAUD 115200

ESP8266 wifi(&EspSerial);

BlynkTimer timer;

// This function sends Arduino's up time every second to Virtual Pin (5).
// In the app, Widget's reading frequency should be set to PUSH. This means
// that you define how often to send data to Blynk App.
void myTimerEvent()
{
  // You can send any value at any time.
```



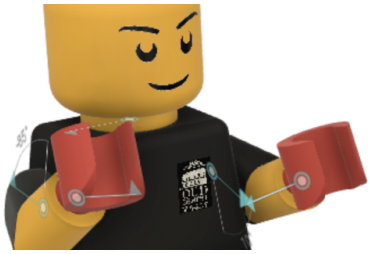


Koen Verbeeck

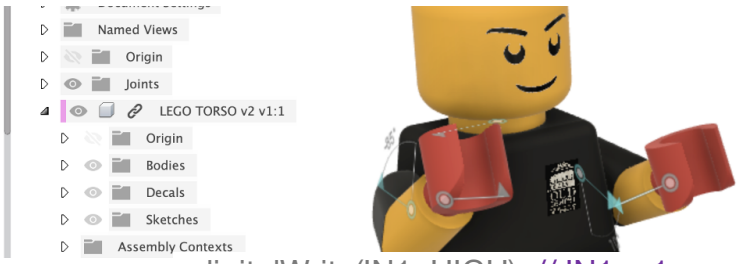
```
  // Please don't send more that 10 values per second !!!!!!!!!
  Blynk.virtualWrite(V5, millis() / 1000);
}

/*
   128x64 OLED Screen
*/
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET -1 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

/*
   SERVO Motor: HS-422 (20Kg)
*/
int servoPin = 3; // Port to control the servo Motor
// Start position of the Servo, stright wheels (only executed at begining)
int servoStartPos = 85;
int servoCenterPos = 85; // Drive Forward
int servoLeftmax = 110; // Turn Left // 105
int servoRightmax = 60; // Turn Right // 65

Servo myServo; // Create a virtual object into your Arduino so you can interact with it.
// Also give it a name (myServo).

/*
   LEDs
*/
#define redLED 34 // red LED on top of car (signalisation)
#define yellowLED 32 // yellow LED on top of car (signalisation)
#define greenLED 30 // green LED on top of car (signalisation)
#define LED_LF 22 // LED Left Front
#define LED_RF 24 // LED Right Front
#define LED_LB 50 // LED Left Back
#define LED_RB 52 // LED Right Back

// Red LED On
void RedLEDsOn(){
  digitalWrite(redLED, HIGH), digitalWrite(yellowLED, LOW), digitalWrite(greenLED,
LOW);
}
```



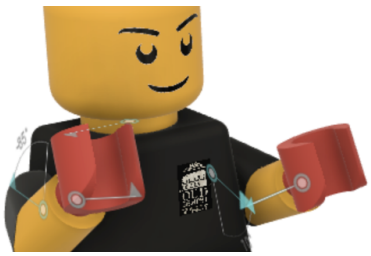

```
// Yellow LED On
void YellowLEDsOn(){
  digitalWrite(redLED, LOW), digitalWrite(yellowLED, HIGH), digitalWrite(greenLED,
LOW);
}

// Green LED On
void GreenLEDsOn(){
  digitalWrite(redLED, LOW), digitalWrite(yellowLED, LOW), digitalWrite(greenLED,
HIGH);
}

/*
   Push-Button
*/
#define pushButton 44
int museum_timer;

/*
   Line Tracking Sensor
*/
#define LT_L 2 // Left Sensor
#define LT_M 4 // Middle Sensor
#define LT_R 10 // Right Sensor
int linelost_timer = 0;

/*
   HC-SR04 Distance Sensor
*/
int trigPin = 26;    // Trigger
int echoPin = 28;    // Echo
// The pingTravelDistance devided by 2 (/2)
float cm;

// The total pingTravelTime (RTT, two-way)
int pingTravelTime;

void measure_distance() {
  // The sensor is triggered by a HIGH pulse of 10 or more microseconds.

  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  digitalWrite(trigPin, LOW);
  // Delay to make sure that the triggerpin has been set to low.
```



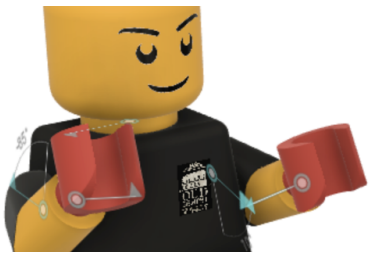

```
  delayMicroseconds(5);
  // Read the signal from the sensor: a HIGH pulse whose
  // duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  // (RTT, two-way)
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pingTravelTime = pulseIn(echoPin, HIGH);

  // Convert the time into a distance
  // distance = (traveltime/2) x speed of sound
  // We need to divide the traveltime by 2 because we have to take into account that
the wave was sent,
  // hit the object, and is then returned back to the sensor.
  // The speed of sound is: 343m/s = 0.0343 cm/uS = 1/29.1 cm/uS
  // Divide by 29.1 or multiply by 0.0343 (Speed of sound)
  cm = (pingTravelTime / 2.) / 29.1;
}

/*
   DC Motors - L298N I/O Pins
*/
int ENA = 5;  // Control Channel A for left wheel
int ENB = 6;  // Control Channel B for right wheel
int IN1 = 7;  // Control Signal 1 for left wheel
int IN2 = 8;  // Control Signal 2 for left wheel
int IN3 = 9;  // Control Signal 3 for right wheel
int IN4 = 11; // Control Signal 4 for right wheel

// set car speed
int CAR_SPEED_F = 90; // Car Speed for Driving Forward 255 is Max. , Min. is 70.
int CAR_SPEED_T = 120; // Car Speed for Turning 255 is Max. , Min. is 70.

void stop_car() {
  digitalWrite(IN1, LOW); // IN1 = 0
  digitalWrite(IN2, LOW); // IN2 = 0
  digitalWrite(IN3, LOW); // IN3 = 0
  digitalWrite(IN4, LOW); // IN4 = 0
}

void forward(int car_speed) {
  analogWrite(ENA, car_speed);
  analogWrite(ENB, car_speed);
```



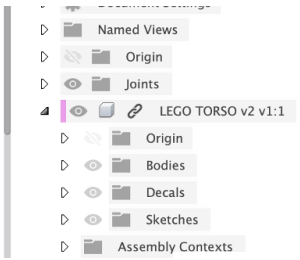

```
  digitalWrite(IN1, HIGH); // IN1 = 1
  digitalWrite(IN2, LOW); // IN2 = 0
  digitalWrite(IN3, LOW); // IN3 = 0
  digitalWrite(IN4, HIGH); // IN4 = 1
}

void turnLeft(int car_speed) {
  analogWrite(ENA, car_speed);
  analogWrite(ENB, car_speed);
  digitalWrite(IN1, LOW); // IN1 = 0
  digitalWrite(IN2, HIGH); // IN2 = 1
  digitalWrite(IN3, LOW); // IN3 = 0
  digitalWrite(IN4, HIGH); // IN4 = 1
}

void turnRight(int car_speed) {
  analogWrite(ENA, car_speed);
  analogWrite(ENB, car_speed);
  digitalWrite(IN1, HIGH); // IN1 = 1
  digitalWrite(IN2, LOW); // IN2 = 0
  digitalWrite(IN3, HIGH); // IN3 = 1
  digitalWrite(IN4, LOW); // IN4 = 0
}

/*
   128x64 OLED Screen
*/
void OLED_okdrive() {
  display.clearDisplay();
  // Display the measured distance on the OLED
  display.setCursor(0, 0);
  // Display static text
  display.println("Distance:");
  display.print(cm);
  display.println("cm");
  display.setCursor(0, 45);
  display.println("OK, DRIVE");
  display.display();
}

void OLED_stopcar() {
  display.clearDisplay();
  // Display the measured distance on the OLED
  display.setCursor(0, 0);
  // Display static text
```

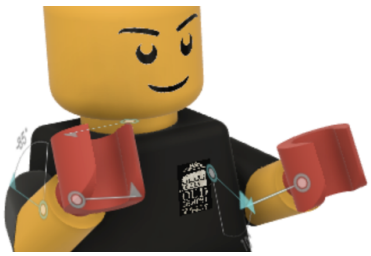



```
  display.println("Distance:");
  display.print(cm);
  display.println("cm");
  display.setCursor(0, 45);
  display.println("STOP CAR !");
  display.display();
}

// Function while halting at an exhibit.
void OLED_sightseeing() {
  display.clearDisplay();
  // Display the measured distance on the OLED
  display.setCursor(0, 0);
  // Display static text
  display.println("Exhibit");
  display.setCursor(0, 30);
  display.println("Time Left:");
  display.print(museum_timer/1000);
  display.println(" Sec.");
  display.display();
}

//-----------------------------------------------------------------------------------------------
--

void setup() {

  // Debug console Serial Monitor
  Serial.begin(115200);

  /*
   BLYNK Server
  */

  // Set ESP8266 baud rate
  EspSerial.begin(ESP8266_BAUD);
  delay(10);

  // You can also specify server:
  // Blynk.begin(auth, wifi, ssid, pass, "blynk-cloud.com", 80);
  // Own Server and port
  Blynk.begin(auth, wifi, ssid, pass, server, port);

  // Setup a function to be called every second
  timer.setInterval(1000L, myTimerEvent);
```



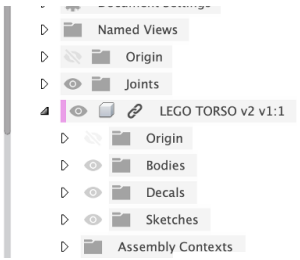

```
/*
  128x64 OLED Screen
*/

// SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
// 0x3C is I2C address
if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
  Serial.println(F("SSD1306 allocation failed"));
  for (;;); // Don't proceed, loop forever
}

// Clear the buffer.
display.clearDisplay();
// display.setTextSize(1);
display.setFont(&FreeSerifBoldItalic12pt7b);
display.setTextColor(WHITE);
display.invertDisplay(true);
display.setCursor(20, 40);
// Display static text (CHEVY WELCOME SCREEN)
display.println("CHEVY");
display.display();
delay(1000);
// Prepare the Display for dynamic text
display.setFont();
display.setTextSize(2);
display.invertDisplay(false);
delay(1000);

/*
  SERVO Motor
*/
// Attach the virtual Servo (object) to your Arduino using the control pin to interact
with it.
myServo.attach(servoPin);
// The Servo will start at the configured Start position of the Servo.
myServo.write(servoStartPos);

/*
  Line Tracking Sensor
*/
pinMode(LT_L, INPUT);
pinMode(LT_M, INPUT);
pinMode(LT_R, INPUT);
```

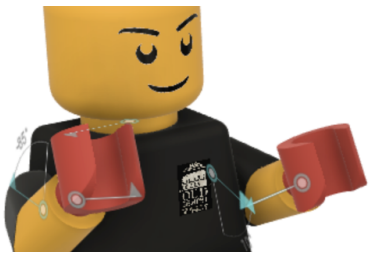




Koen Verbeeck

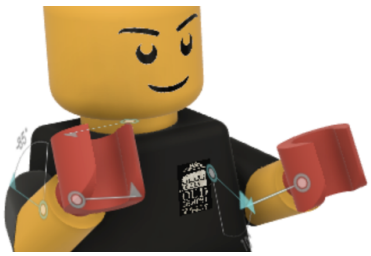
```
  /*
    LEDs
  */
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(LED_LF, OUTPUT);
  pinMode(LED_RF, OUTPUT);
  pinMode(LED_LB, OUTPUT);
  pinMode(LED_RB, OUTPUT);
  digitalWrite(LED_LF, HIGH);
  digitalWrite(LED_RF, HIGH);
  digitalWrite(LED_LB, HIGH);
  digitalWrite(LED_RB, HIGH);

  /*
   Push-Button
  */
  pinMode(pushButton, INPUT);

  /*
    DC Motors
  */
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  /*
     HC-SR04 Distance Sensor
  */
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

//-------------------------------------------------------------------------------------------
--

void loop() {
```



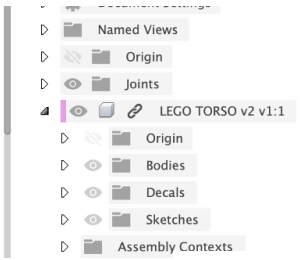


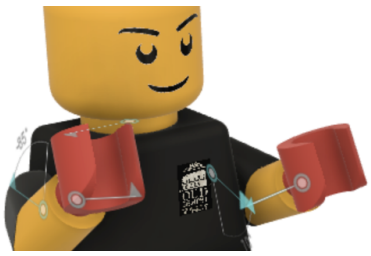
Koen Verbeeck

```
/*
 BLYNK Server
*/
Blynk.run(); // Maintain the connection and send & receive data
timer.run(); // Initiates BlynkTimer

/*
   Line Tracking Sensor
*/

// Print Line Tracking Sensor State: 0 is Black Line, 1 is no Black Line

//  Serial.print("LT_L: ");
//  Serial.print(digitalRead(LT_L));
//  Serial.print("  LT_M: ");
//  Serial.print(digitalRead(LT_M));
//  Serial.print("  LT_R: ");
//  Serial.println(digitalRead(LT_R));

// If no Black Line is detected by Left (LT-L) or Right (LT-R) sensor.
if (digitalRead(LT_L) && digitalRead(LT_R)) { // Drive Forward
  forward(CAR_SPEED_F);
  myServo.write(servoCenterPos);
  GreenLEDsOn();
  linelost_timer += 1;
  Serial.println(linelost_timer);
  if (linelost_timer >= 50) {
    stop_car();
    RedLEDsOn();
  }
}

// If a Black Line is detected by the Left Sensor.
else if (!(digitalRead(LT_L)) && digitalRead(LT_R)) { // Turn Left
  turnLeft(CAR_SPEED_T);
  myServo.write(servoLeftmax);
  GreenLEDsOn();
  linelost_timer = 0;
  Serial.println(linelost_timer);
}

// If a Black Line is detected by the Right Sensor.
else if (digitalRead(LT_L) && !(digitalRead(LT_R))) { // Turn Right
  turnRight(CAR_SPEED_T);
```



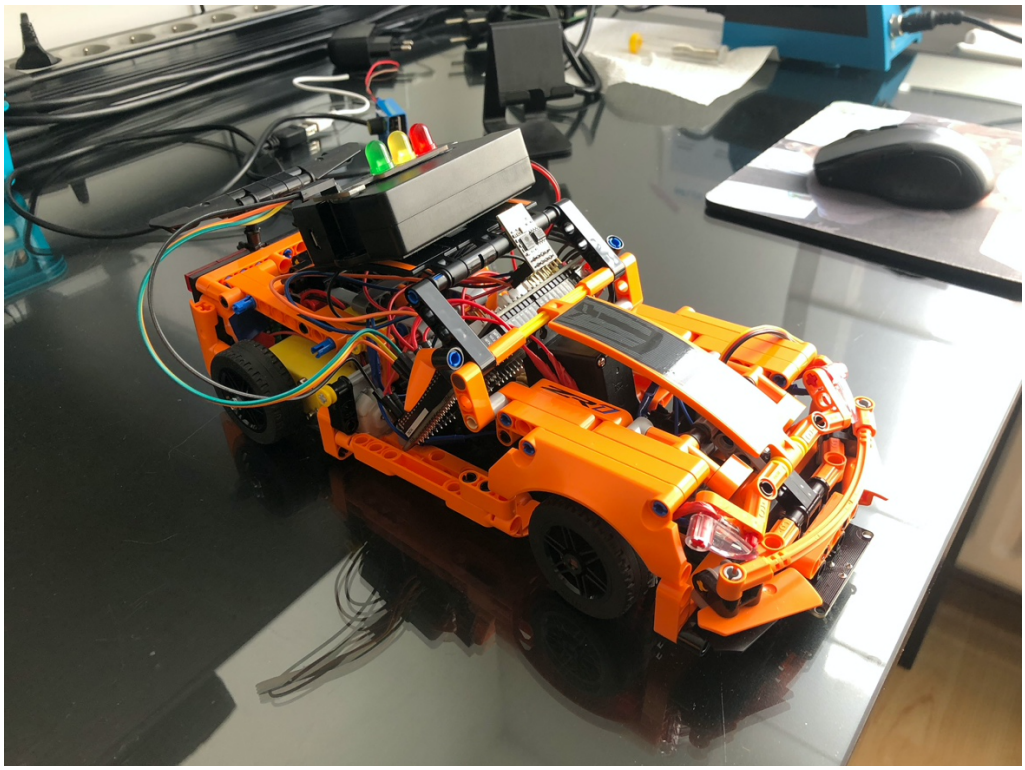

```
    myServo.write(servoRightmax);
    GreenLEDsOn();
    linelost_timer = 0;
    Serial.println(linelost_timer);
  }

  // If black perpendicular line is detected
  // Halt Car for 20 seconds, unless hardware or Virtual (BLYNK) push-button is pushed
  else if (!(digitalRead(LT_L)) && !(digitalRead(LT_M)) && !(digitalRead(LT_R))) {
    YellowLEDsOn();
    delay(500); // Wait 500ms before halting the car
    myServo.write(servoCenterPos);
    stop_car();
    museum_timer = 20000; // Halt Car for 10 seconds
    linelost_timer = 0;
    Serial.println(linelost_timer);
    while (museum_timer >= 0){
      museum_timer -= 200; // subtract 200ms from timer
      OLED_sightseeing();
      delay(200); // delay 200ms after subtrackting from timer
      measure_distance(); // Not needed while halting for sight seeing
      Blynk.run(); // Otherwise BLYNK will crash while in the loop
      timer.run(); // Otherwise BLYNK will crash while in the loop
      if (digitalRead(pushButton) == 1){
        break;
      }
    }
  }

  measure_distance();

  // While the distance to the object is smaller or equal to 20, then STOP the CAR !!!
  while (cm < 20.0) {
    OLED_stopcar();
    stop_car();
    YellowLEDsOn();
    measure_distance();
    Blynk.run(); // Otherwise BLYNK will crash while in the loop
    timer.run(); // Otherwise BLYNK will crash while in the loop
  }

  OLED_okdrive();
}
```

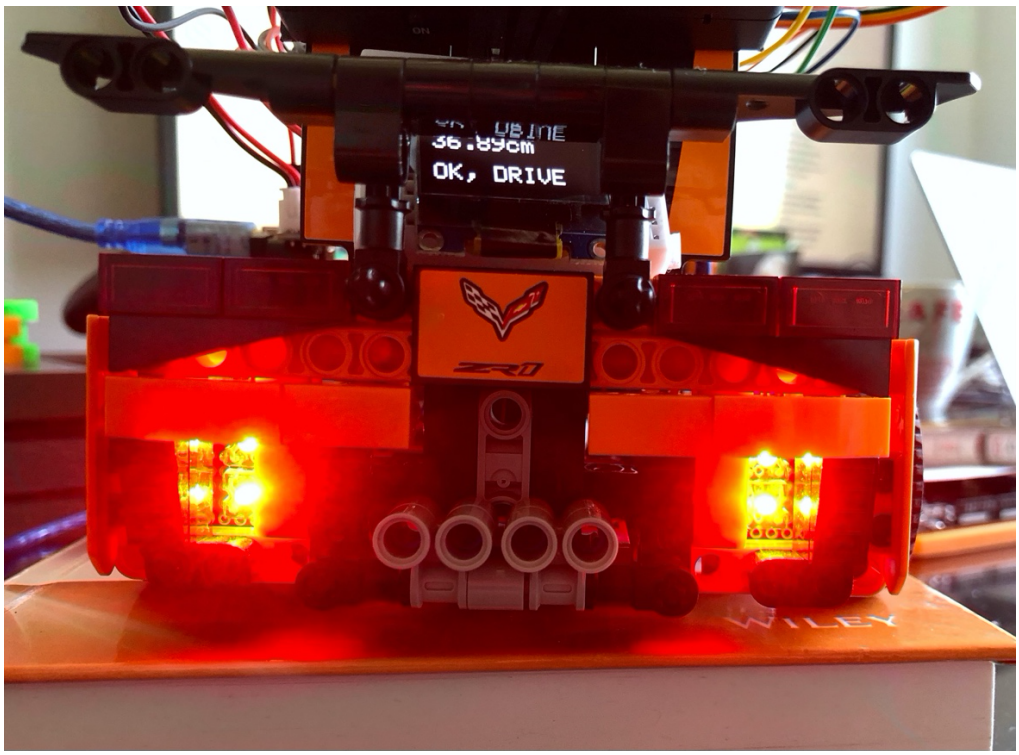


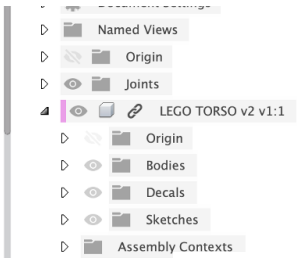


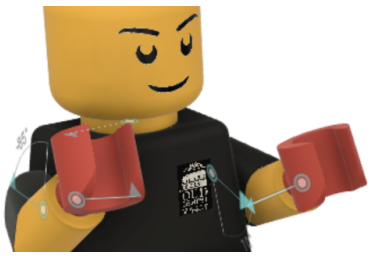
Koen Verbeeck

Pictures

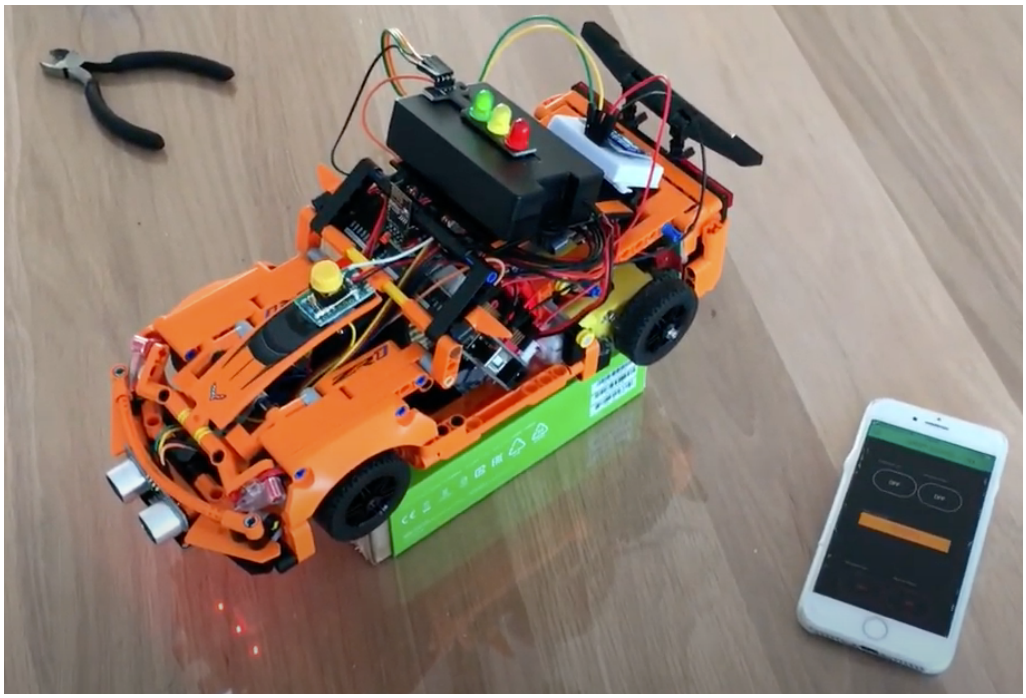








Koen Verbeeck



More pictures displaying installation steps can be found on my Github page.
Video can be found on my YouTube Page.