Development of an Autonomous Car for Folkrace at Robot-SM 2019

Birger Markusson, DevPort Väst AB

1.1 Development Team Members

David Anderson, Maaz Ullah Arshad, Misha Jafar, Birger Markusson, Jenny Mottare, Louise Rudbäck.

In the beginning, the team worked as a scrum team with Louise as a product owner and Maaz as a scrum master. Atlassian Jira, introduced to the team by David, was used for project control and documentation. The team also started a slack communication channel as an alternative to email. One member after the other left the team work for various reasons such as e.g. other assignments.

1.2 Starting Point 2019

The team continued from the autonomous car of 2018 in the beginning of February 2019. This year the car was named DevCar.

1.3 Hardware

Figure 1 shows DevCar at the day of robot SM 2019. The main driving motor is a DC motor that drives all four wheels via a front and a rear differential. There is also a servo motor for controlling the angle of the front wheels.

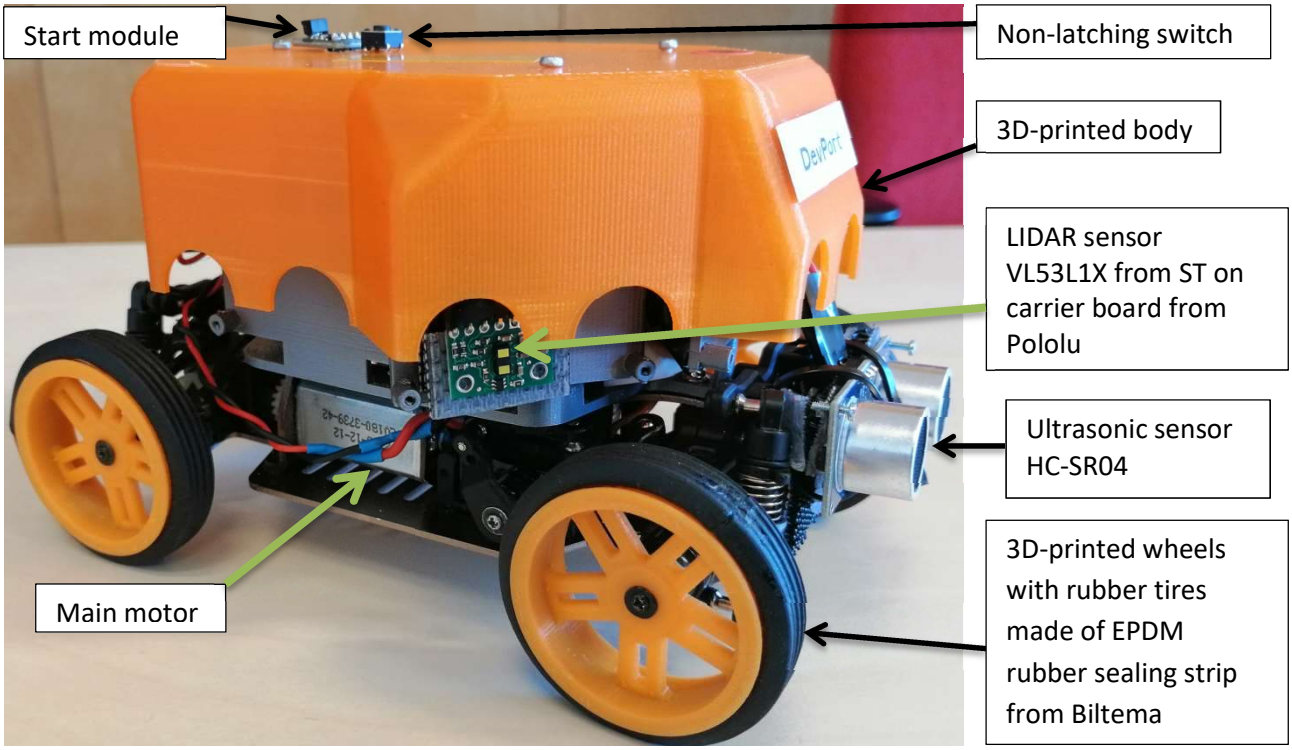

Figure 1: DevCar appearance at the day of competition, 2019-04-06.

1.3.1 Start module

The start module on the roof can be bought from e.g. http://startmodule.com/ or http://chalmersrobotics.se/. They are also available for 80 SEK in the basement office at Sven Hultins gata 8, 412 58 Göteborg, https://goo.gl/maps/htJDj2YC4hv . The electrical connection could be according to Figure 2. Also a remote control for the start module is available. By means of the switch, the car can be started without the remote control although the Arduino code is the same as the one used during the competition for remote control.

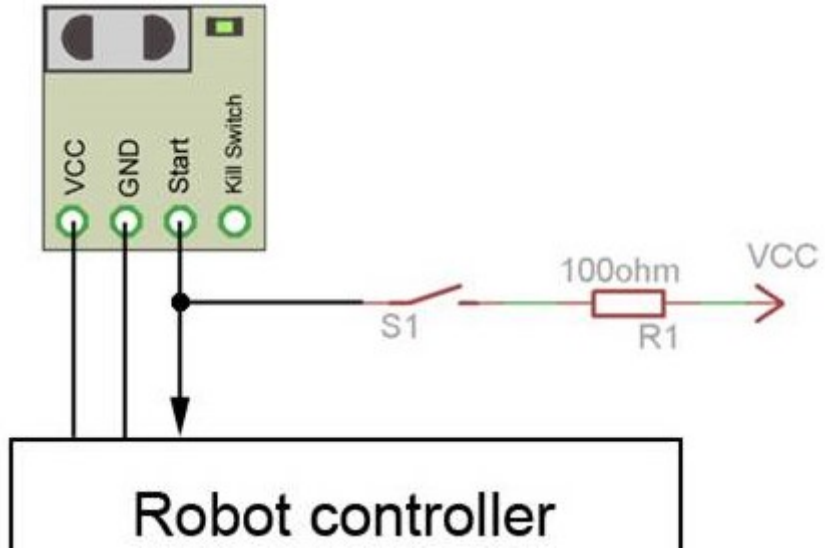

Figure 2: Prebuilt start module.

1.3.2 Original Car Specifications

Judging by photos such as in Figure 3, Figure 4 and specifications, the original car was Model A232, A242 or A252 made by WL-Toys. Banggood gives the following specifications:

Brand: WL-Toys, model A232, Scale: 1/24, Length: 203 mm, Width: 114 mm, Height: 70 mm, Type: 4WD 2.4GHz Rc car Short Course Truck style, high quality car, four-wheel independent suspension, 4-wheel shock absorbers, proportional steering function, Top speed 35 km/h, 100 m control distance, Motor: 180 brushed, ESC: 2 in 1 integrated, Battery type: 2S LiPo, Voltage: 7.4V, Capacity: 500mAh, Run time: about 20 min, Charging time: 1/3 hours.

Figure 3: WL-Toys model A232 without the body.

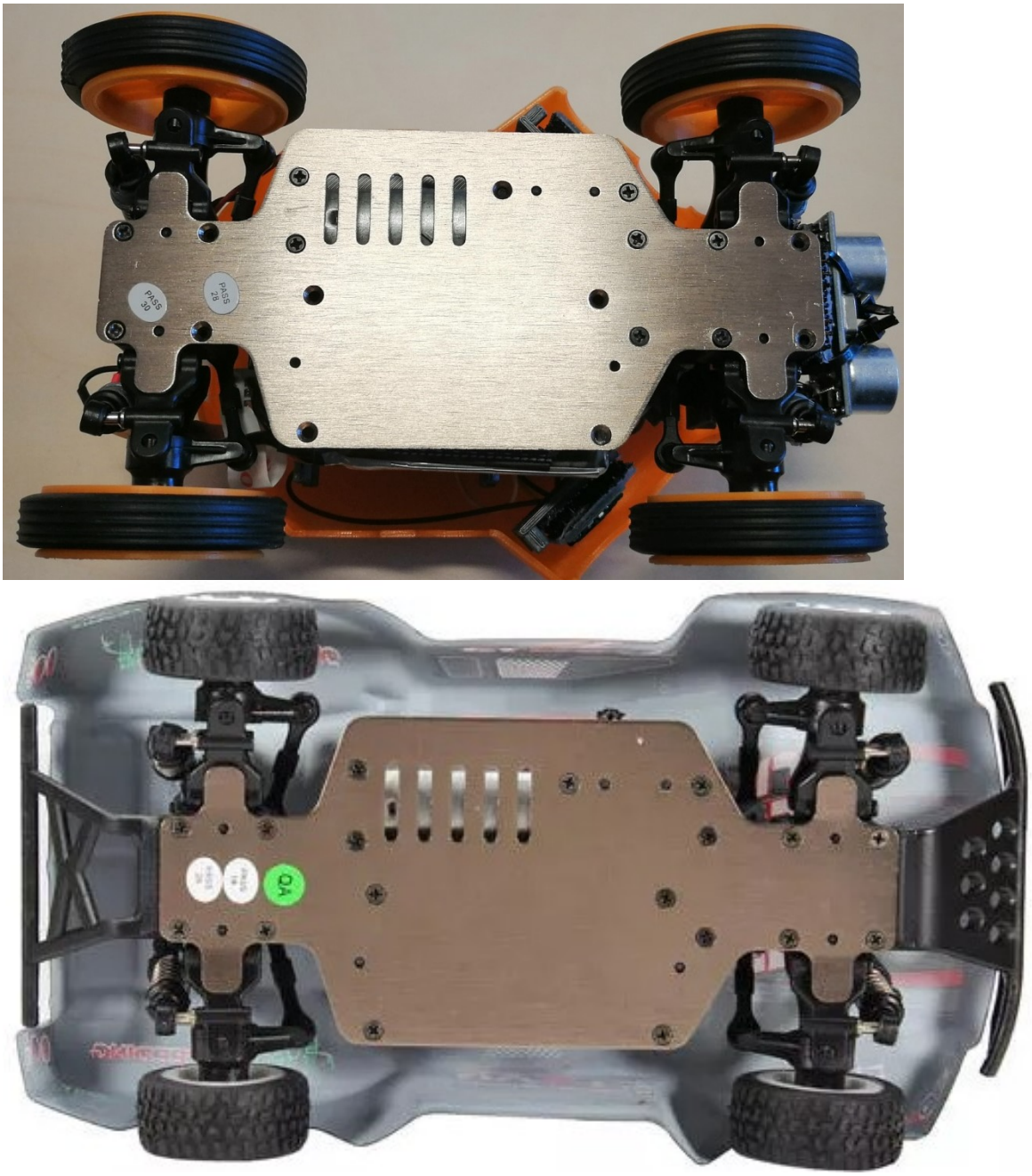

Figure 4: The upper figure shows DevCar. The lower figure shows *WL-Toys A232 1/24 2.4G 4WD Brushed RC Car Short Course RTR*. Currently, Banggood sells such an RC car for 578 SEK.

1.3.3 Servo Mechanism

The (original) construction of the steering mechanism is a disappointment since a lever arm (servo rod) is stopped by the servo motor casing when the front wheel angle reaches about 15° to the left. That can be a severe restriction on a narrow racing track with obstacles. On DevCar Misha put in a plastic washer to get the swing arm a bit further out so that the servo rod does not bump into the servo motor. See Figure 5. After this modification, DevCar can turn the front wheels about 27° both to the left and to the right, but 1° turn to one side in the software code may no longer give 1° turn of the front wheels. Originally, servo angle 90° in the code (myservo.analogWrite(90);) put the front wheels straight ahead. After modification, straight ahead is obtained approximately by the code angle 110° depending somewhat on the previous angle. WL-Toys sells an upgrade servo for $14.99 (https://www.wl-toys.com/Wltoys-A232-A242-A252-Parts-37-Upgrade-Servo-More-Steering-angle-More-Cheap-price-12924.html) that allows a larger steering angle than the original 5 g servo permits.

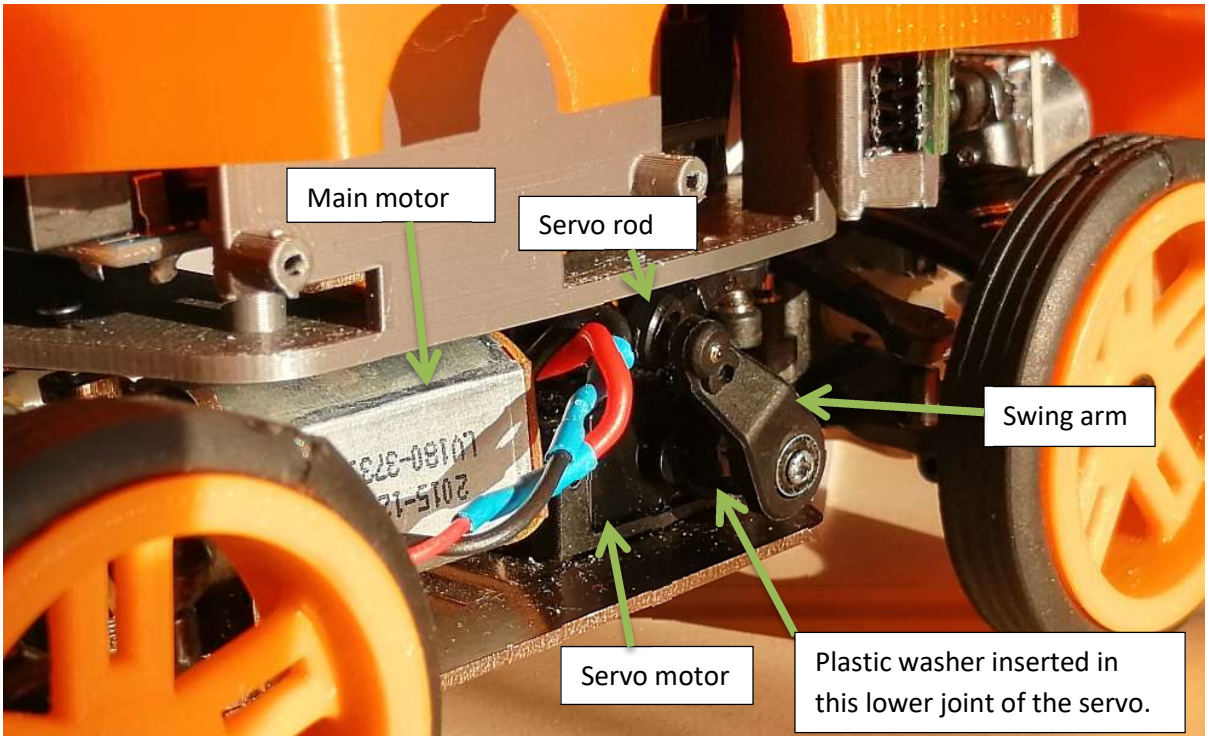

Figure 5: Main motor and servo.

1.3.4 Microcontroller Board and Current Supply for the Main Motor

Inside DevCar is one Arduino Uno Rev3 microcontroller breakout board. One Arduino Motor Shield is mounted on top of Uno, as shown in Figure 6. The main function of the motor shield is as a current supply for the main motor. The contact pins protruding downwards from the shield fits the corresponding ports in Uno. Hence, the shield is largely an extension of the Uno. Uno is enough for the servo, the sensors and the start module.

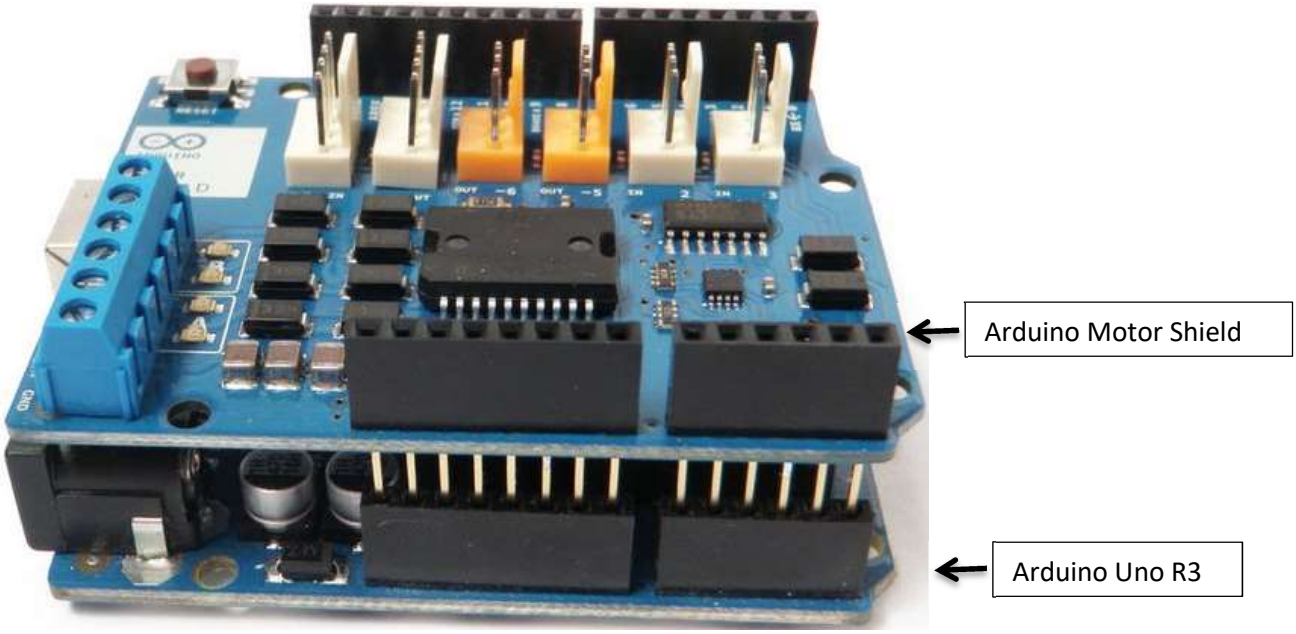

Figure 6: Arduino Motor Shield inserted on the top of an Arduino Uno board. Photo from https://www.instructables.com/id/Arduino-Motor-Shield-Tutorial/

For mass reduction, the shield with mass around 20-25 g can be replaced by Pololu's carrier board for Texas Instruments' driver DRV8838 in Figure **7** for a single brushed DC Motor. Pololu's tiny board has the mass 0.3 g without header pins. In Figure **7**, MCU can be represented by Uno on which any of the digital pins 3, 5, 6, 9, 10 and 11 can give a PWM signal that can be used to control the motor speed via the ENABLE port of the motor

driver. The PHASE port is used for setting the main motor direction. (Presently on DevCar, pin 3 is used for main motor speed control, and pin 6 is used for servo angle control.) DRV8838 can supply a motor with about 1.7 A continuously which is sufficient for DevCar's main motor which could be similar to a 30000 rpm, 7.4V, 480 mA brushed type 180 motor available at https://www.ebay.com. However, there are type 180 motors with much lower or much higher speed at 7.4 V.

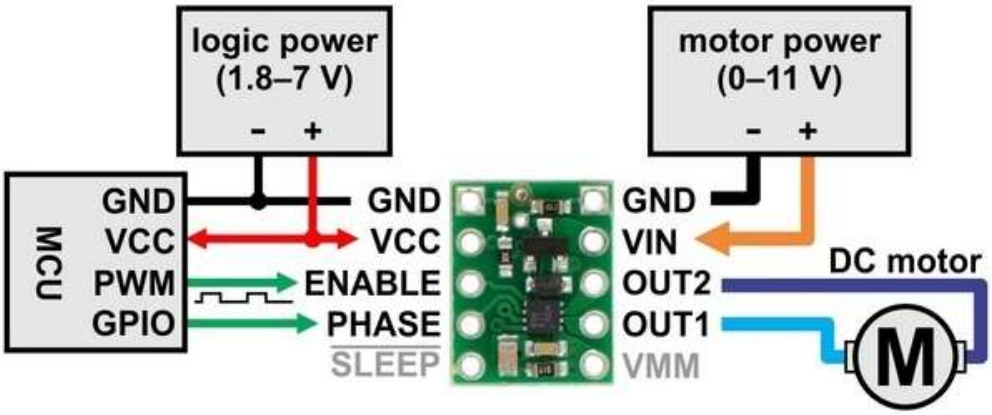

Figure 7: Minimal wiring diagram for connecting a microcontroller to a DRV8838 Single Brushed DC Motor Driver Carrier. See https://www.pololu.com/product/2990. The board size is about 12.7 mm x 10.2 mm. This is a suggested replacement of the bulky Arduino Motor shield.

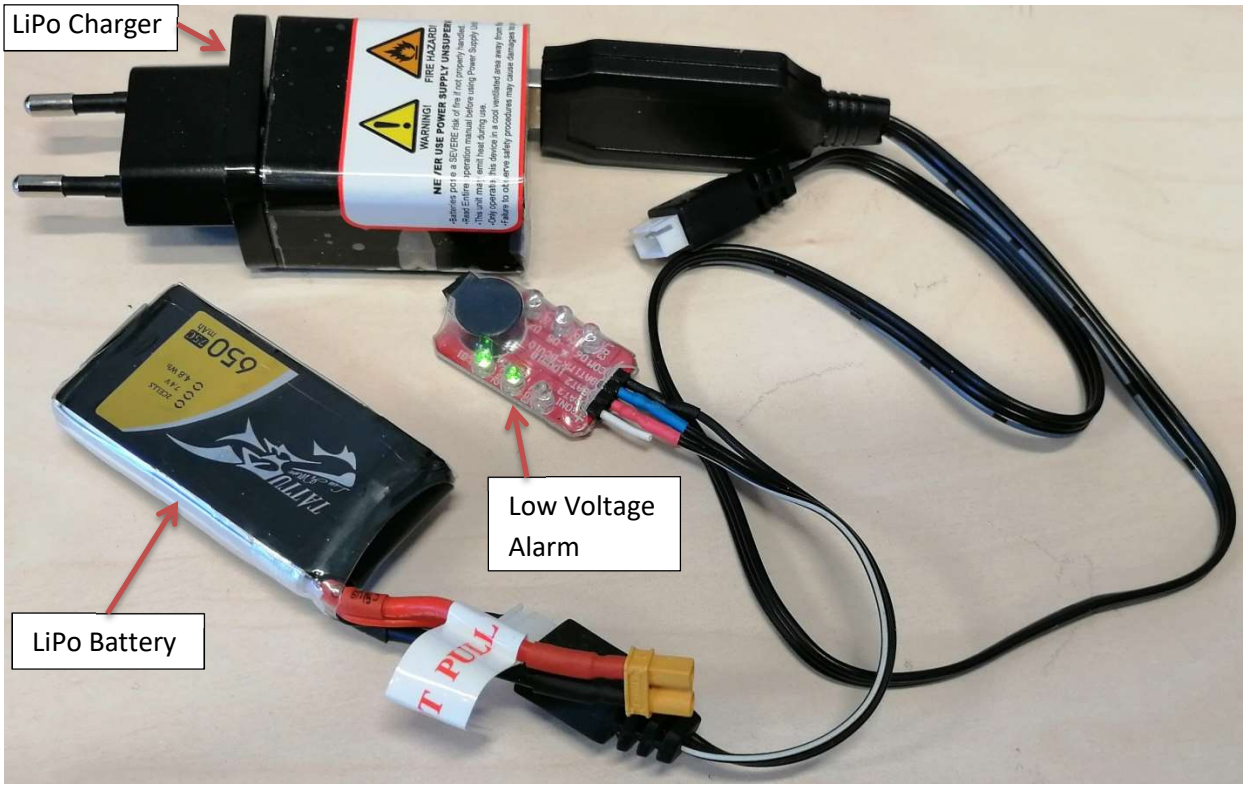

Figure 8: LiPo (Lithium Polymer) battery charger, LiPo two cell battery of capacity 650 mAh, 7.4 V, 75 C, 4.8 Wh attached to a low voltage alarm from HobbyKing, https://hobbyking.com/en_us/hobbykingtmlipoly-low-voltage-alarm-2s-3s.html .

Figure 8 shows a Tattu 650 mAh, 7.4 V, 75 C, 4.8 Wh LiPo (Lithium Polymer) battery with 2 cells in series. The capacity is 650 mAh. The number 75 C is the so called C rating. For LiPo batteries, the maximum continuous safe discharge current is

$$\max(I_{Discharge}) = Capacity \times C\ Rating$$

Hence, the max discharge current of the battery in Figure 8 is about 49 A. See https://www.rotordronemag.com/c-rating-drone-lipo-battery-packs/. The nominal voltage per cell is 3.7 V. The voltage per cell should be between 3.3 V and 4.2 V. The cable and contact attached to the low voltage alarm was taken from an old LiPo charger. The alarm is designed for batteries with two cells in series (2S) and three cells in series (3S). The alarm has one red and one green light diode per battery cell. The red diode instead of the green is lit, and the alarm sounds as soon as the voltage of a cell becomes lower than 3.3 V. Rubber sealing was put into the alarm bell to dampen the sound.

Figure 9 shows the sensor fixture, battery location and some electrical connections. None of the five fixtures for ultrasonic sensors was used for such sensors at the competition. The battery and the sensors are fixed with velcro. The use of digital pins/ports is specified in the Arduino code.

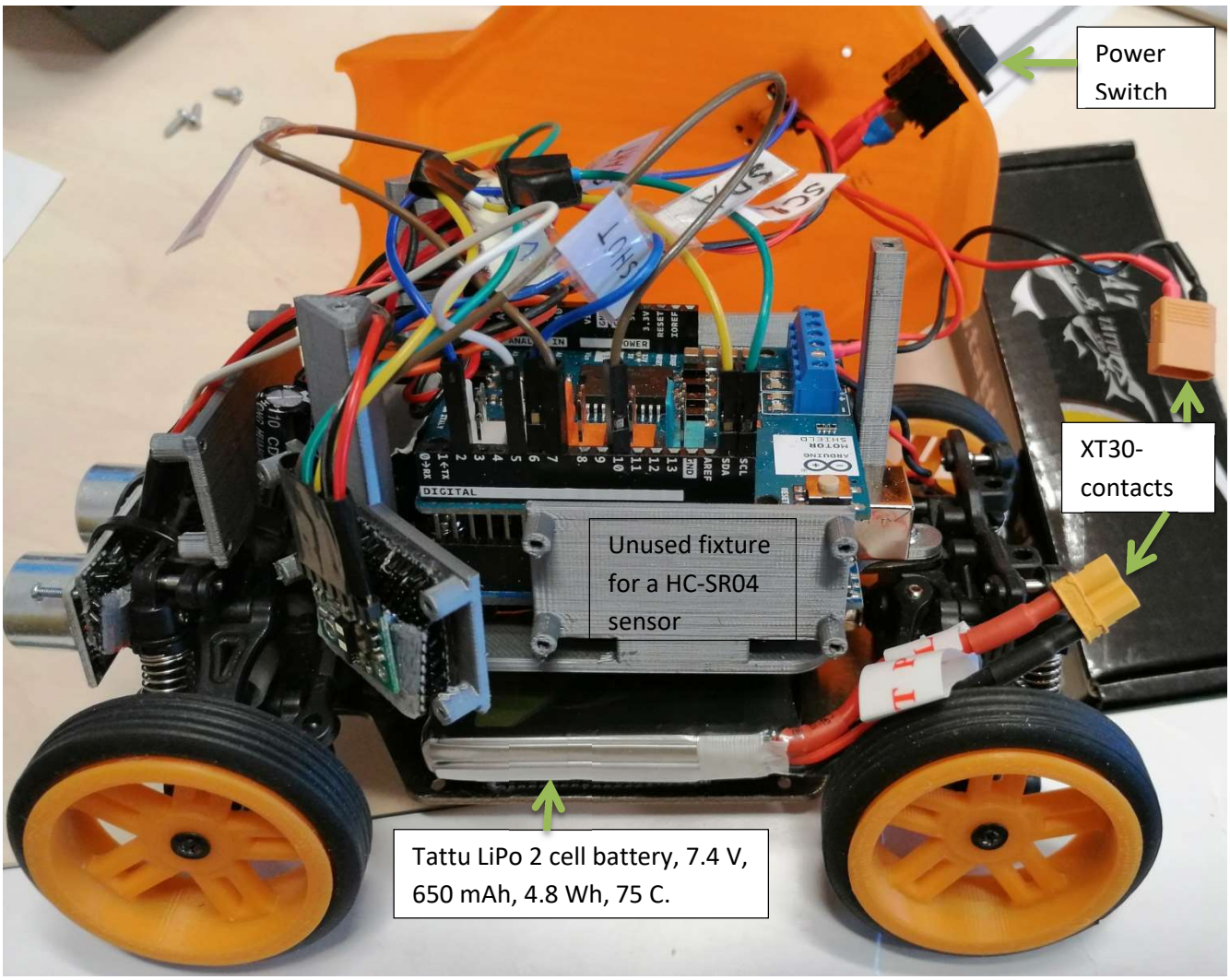

Figure 9: Sensor fixture, battery location and some electrical connections.

Figure 10 shows a top view of DevCar without the body. I order to get a 5 V power supply to all sensors, the servo motor and the start module, ground and 5 V from the shield was connected to a branch contact made of two single row female headers soldered to a piece of experiment board. Two electrolytic 460-470 uF capacitors between 5 V and ground in the branch contact are used to stabilize the voltage. The most suitable capacitance has not been determined yet. Power cables are twisted to reduce EMI (Electromagnetic Interference). In 2018, the car had a heavy ferrite ring instead around the main motor cables.

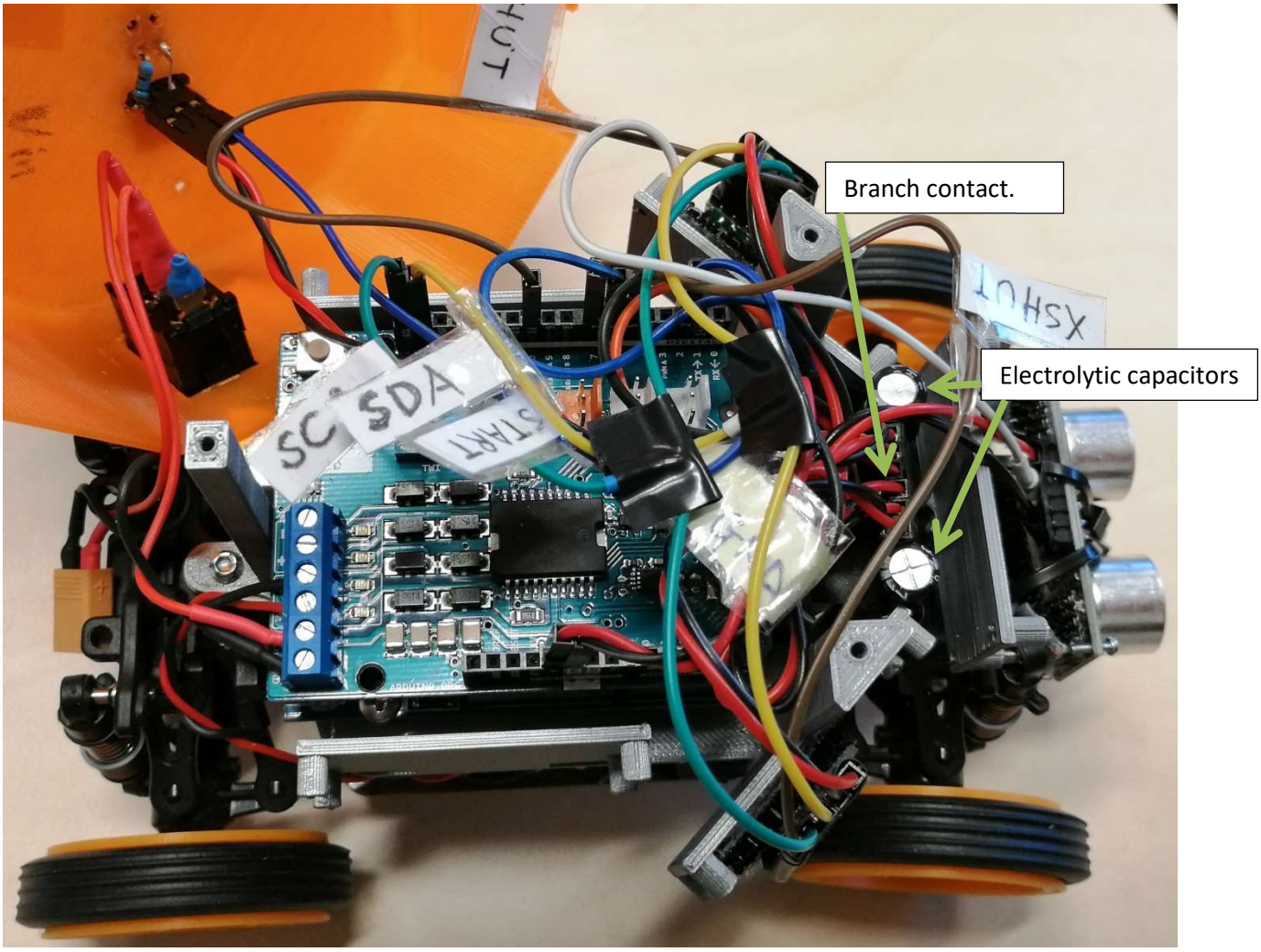

Figure 10: The branch contact for 5V and ground is required to power the sensors, the servo and the start module.

In order to further reduce the mass of DevCar, it is recommended that Arduino Uno Rev3 with mass 25 g is replaced by a more compact Arduino board such as Arduino Nano Rev3 with mass 7 g. Nano Rev3 is a compact version of Uno Rev3. Nano Rev3 also supports I2C communication. For I2C, analog port A4 is SDA, and A5 is SCL. See specifications on https://store.arduino.cc/arduino-nano and https://store.arduino.cc/arduino-uno-rev3 .

The speed of the winning car Pink Panic can be estimated by the track length and the number of turns Pink Panic managed to complete during one heat in 3 minutes. Pink Panic made at least 22 turns in 3 minutes. If the track length is of the order of 10 m, the average speed is just 1.2 m/s. Without obstacles, Pink Panic would make a few more turns, maybe around 30. This corresponds to an average speed of about 1.6 m/s. If DevCar had a max speed of around 2-3 m/s and sufficient torque for the hills, DevCar could win. Presently, the top speed of DevCar with its enlarged wheels is estimated to be around 50/37.5*35 km/h ≈ 13 m/s if air friction is neglected. Then DevCar should have its gear ratio increased from 29:17 ≈ 1.7 to about 7.4-11.1, but WL-Toys only supplies the original gear wheels among the spare parts. Another approach is to estimate the speed DevCar can have in a curve without sliding. At the limit of radial sliding in a curve, the centripetal force equals the frictional force, i.e.

$$\frac{mv^2}{r} = \mu mg \Rightarrow v = \sqrt{\mu r g}$$

where m is the mass, v is the speed, r is the curve radius, g is the gravitational acceleration, and μ is the coefficient of friction. The sliding coefficient of friction between rubber and dry asphalt is 0.5-0.8 according to http://www.engineershandbook.com/Tables/frictioncoefficients.htm. The material of the racing track is not asphalt but painted wood. Assumed parameter values μ =0.5-0.8, r = 1 m, and g = 9.81 m/s^2 give v = 2.2-2.8 m/s. Without a way to increase the friction, this confirms that there is not much to gain on giving DevCar a maximum speed above 3 m/s since the straight parts of the racing track are short. One way to increase the friction is to add a helicopter motor and rotor, but it gives more mass, more air friction and requires more power. Sliding is not the only problem with a high speed. Another problem is that DevCar must have time to measure distances often enough in order to avoid obstacles.

Figure 11 shows the two white plastic gear wheels with about 17 and 29 teeth respectively that can be replaced in order to give DevCar a much higher torque and a much lower maximum speed suitable for folkrace at Robot-SM. Grey, plastic hollow cylinders pre-compress the wheel suspension springs to give DevCar a more upright position in curves. The USB port has been used for uploading the Arduino code to Arduino Uno and for monitoring sensor data via the serial monitor that can be opened from the Arduino IDE. The computer can supply sufficient voltage and power for the servo and the sensors via the USB but not for the main motor. With the XT30 contacts connected, DevCar including the main motor can be started via the power switch. Optionally, the Arduino code can be written to wait for a start signal given to the start module.

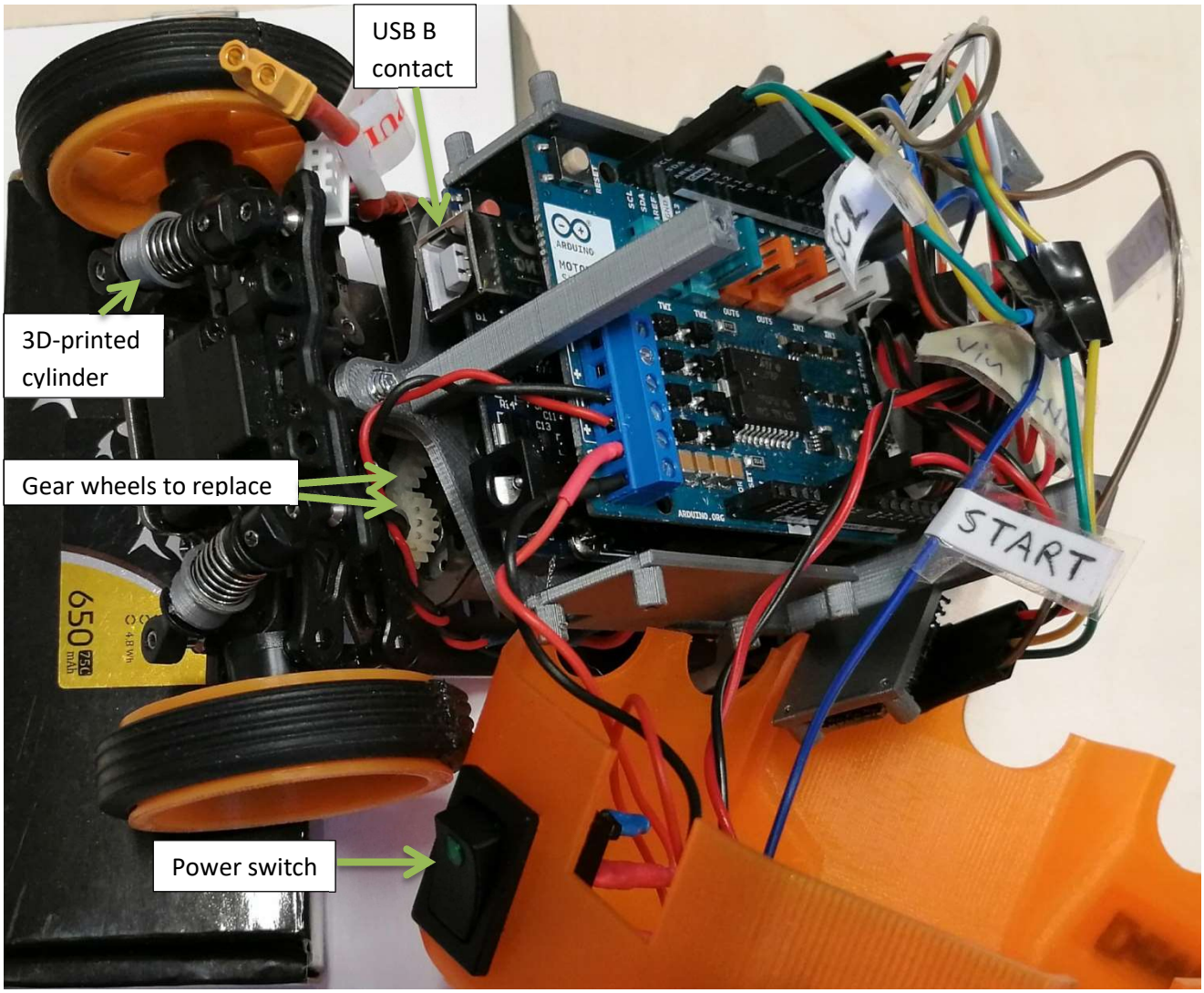

Figure 11: Rear view of DevCar showing the main motor gear wheels, 3D-printed plastic cylinders for spring pre-compression, USB contact as part of the Arduino Uno, and a battery power switch for DevCar when the battery XT30 contacts are connected.

1.3.5 Sensors

The body of DevCar was designed to fit five ultrasonic sensors of the type HC-SR04, but because of the difficulties with reliability of measured distances to the low walls of the racing track, about a week before the competition Birger decided to try two LIDAR sensors that had been bought a few weeks earlier.

1.3.5.1 Ultrasonic Sensors

Ultrasonic sensors have either the transmitter (T) and the receiver (R) at two clearly separate locations, such as in the cheap HC-SR04, or the transmitter and receiver at the same location, such as in the expensive sensors from Maxbotix. See Figure 12. A problem with multiple use of ultrasonic sensors is crosstalk, i.e. one sensor can receive the signal of another sensor. For robots, Maxbotix recommends their high resolution HRLV-MaxSonar-EZ rather than their LV-MaxSonar-EZ sensors, because the HRLV sensors have the best reduction of external noise sources. However, the HRLV is slow, 10 Hz, and objects closer than 30 cm are reported as being at distance 30 cm. See https://www.maxbotix.com/tutorials1/031-using-multiple-ultrasonic-sensors.htm.
The EZ0, EZ1, EZ2, EZ3, and EZ4 versions of the MaxSonar sensors have progressively narrower beam angles.
The ultrasonic sensors require a stable voltage supply. According to Maxbotix, a capacitor of typically 100 µF between the V+ pin and the ground pin solves most power related noise problems.

Figure 12: The upper photo shows a HC-SR04 sensor. The lower photo shows a HRLV-MaxSonar-EZ. Both photos are from https://www.electrokit.com.

One reason for the difficulty with the ultrasonic sensor HC-SR04 is the wide ultrasonic beam, about 30°. In addition, it is recommended that a Parallax Ping))) sensor that is similar to a HC-SR04 is mounted high above the floor for long distance measurements. That is not suitable for DevCar that needs to detect other vehicles and the 10 cm high walls of the racing track. Another problem with HC-SR04 is that very little of the beam is reflected back to the sensor from a plane surface if the angle of incidence is larger than 45°.

1.3.5.2 LIDAR Sensor VL53L1X

LIDAR (Light Detection and Ranging) sensors are usually expensive but the relatively new and very small time of flight measuring laser sensors of the types VL6180X, VL53L0X and VL53L1X from ST on carrier boards from Pololu in the USA were available for $ 8-12. See https://www.pololu.com/product/3415. The VL53L1X emits an eye safe 940 nm VCSEL (LASER) beam and receives reflections by a matrix of 4x4 to 16x16 single photon avalanche diodes (SPAD). By default, the receiver field of view is 27°. This is obtained when all 256 SPADs are used. Advantages of the LIDAR sensors are that they have a very small mass and can measure distances to oblique surfaces. In short distance mode, the VL53L1X can measure the distance up to 1.3 m independently of light conditions if the default field of view is used. The ports (pins) of the VL53L1X are shown in Figure 13. SDA and SCL are data and clock bus ports respectively that use the I2C communication protocol. Sensors communicating with a microcontroller via I2C are connected to the same SDA bus and the same SCL bus. That requires the possibility to set a unique address for each sensor. The XSHUT input port can be driven LOW (potential 0) initially to make it possible to first initialize and then set an I2C address to each LIDAR sensor, one after the other. SDA and SCL are level shifted to 5 V on the carrier board, but never try to set XSHUT high from a microcontroller that works with 5 V since VL53L1X was designed for 0-2.8 V. XSHUT and GPIO1 are not level shifted. The name GPIO (General Purpose Input Output) is a misnomer since GPIO1 is only an output. By default, GPIO1 is driven low when new ranging data are ready for the host microcontroller (e.g. Uno) to read, but with the Pololu library VL53L1X.h read() function, the GPIO1 is not needed, because read() by default waits until the sensor has completed its measurement. Of the seven ports in Figure 13, only VIN, GND, SDA, SCL and XSHUT need to be connected if more than one VL53L1X is connected to the microcontroller board via the SDA and SCL wires. XSHUT is not needed if I2C is used only for communication between one sensor and the microcontroller. Optionally, VDD can be connected to the supply voltage instead of VIN if the supply voltage is 2.6-3.5 V. GND should be connected to ground.

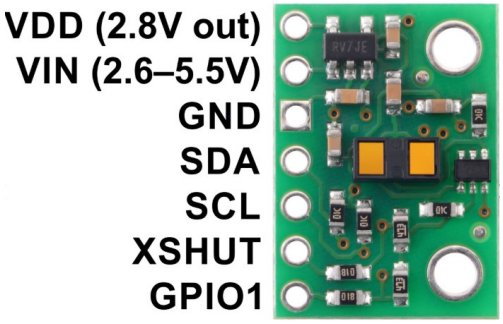

Figure 13: LIDAR sensor VL53L1X on a carrier board from Pololu.

1.3.5.3 IR Sensor Sharp GP2Y0A60SZLF

A fast alternative to the mentioned sensors is SHARP's infrared diode sensor GP2Y0A60SZLF on a carrier board by Pololu shown in Figure 14. The power supply voltage relative to ground (GND) is VCC between 2.7 and 5.5 V. The analog output voltage port OUT can be connected to one of Uno's analog ports A0-A5. The output voltage is approximately a linear function of the inverse of the distance within the measurement range. By default the enable port, EN, is at supply voltage (VCC) via a pull up resistor. EN can be driven low and thereby be used for saving energy between measurements. Otherwise, the sensor measurements are repeated 60 times per second with a typical average current 33 mA.

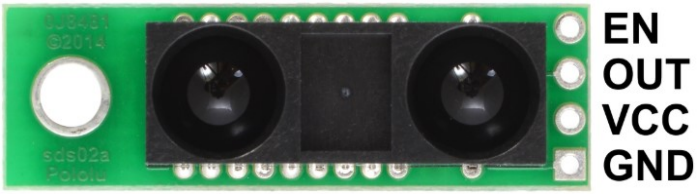

Figure 14: Sharp GP2Y0A60SZLF on a carrier board by Pololu.

1.3.5.4 Comparison of Sensors

Comparisons of some important parameters for some sensors for distance measurements:

Sensor	Max update rate (Hz)	Mass (g)	Average current during measurement (mA)	Distance range (cm)
HC-SR04	20	8.5-9	15 (2 mA quiescent) [1)]	2-500
LV-MaxSonar EZ	20	4.3	2	0 (15)-645 [2)]
HRLV-MaxSonar EZ	10	4.3	3.1 at 5V	0 (30)-500 [3)]
GP2Y0A60SZLF	60	1.1	33-50	10-150
VL53L1X	50	0.5	15 with peak at 40	4-400

1) The similar Parallax Ping))) sensor requires 30-35 mA.
2) Objects within the distance range 0-15 cm are registered as being at the distance 15 cm.
3) Objects within the distance range 0-30 cm are registered as being at the distance 30 cm.

1.4 Software

The Arduino programming language is much like C and C++. A program for controlling an embedded system with Arduino code can be edited in Arduino Integrated Development Environment (IDE) that is free to download and install. A program is called a sketch. The structure of a typical sketch contains headers for necessary libraries, definitions, variable and function declarations, function definitions, the functions void setup() and void loop(). A sketch is compiled and uploaded into the microcontroller board such as Arduino Uno or Nano via USB.

1.5 Folkrace Algorithm

For competence development, the Arduino code was completely rewritten. A relatively simple algorithm, aiming at keeping the car in the center of the track was considered but skipped because it could lead to collisions if different cars with different speeds try to stay midways between the walls and do not care to avoid other cars. However, the centering algorithm works also for centering the car between obstacles if there is time to detect them. Anyway, because of fun and challenge, Birger tried to make an algorithm that aimed at keeping DevCar at a constant distance from one wall except when an obstacle is avoided. Ideally, the algorithm should use measured distances to determine if DevCar has come to a curve or another obstacle. The difference between them is that at a curve without an obstacle, a turn in the direction of the curve does not have to be followed by a turn in the opposite direction. Unless DevCar is disoriented by a collision, DevCar would not turn back and drive in the direction it came from. This is not handled by a simple centering algorithm.

It is possible to write a simulator for the folkrace algorithm in e.g. MATLAB or C++. In MATLAB, the plot command can be used for drawing the race track walls and the position of the car.

The main steps in the algorithm are:

void setup(){

1. Start DevCar when the start signal is set to 1 remotely via the start module.
2. Drive slowly straight forward initially and measure the distances to the left and right wall and choose to follow to nearest wall at an approximately constant distance. Set a flag, a “wall_flag”, equal to 1 if DevCar should follow the left wall. Set the flag to 0 if DevCar should follow the right wall.
   }

   void loop(){
3. Check if the referee has set the start signal to 0. Continue if start signal is still 1. Check wall_flag. Set a medium high speed.
4. if wall_flag == 1, repeatedly, aim at approaching a constant distance to the left wall. Increase the speed to a high speed if DevCar is within a tolerance from a constant distance from the chosen constant distance.
   if wall_flag == 0, repeatedly, aim at approaching a constant distance to the right wall. Increase the speed to a high speed if DevCar is within a tolerance from a constant distance from the chosen constant distance.
5. Slow down or stop temporarily in front of an obstacle.
6. Avoid obstacles and turn in curves not already handled by step 4. Step 6 is subdivided into a number of scenarios.
   }

1.6 Preliminary Arduino Code in Need of Improvements for Use at Robot-SM

//Preliminary code for folkrace at RobotSM 2019. This code version does not wait for a signal from the start module. Just copy and paste this code into an empty Arduino IDE sketch editor window to work with it.

```
#include <Servo.h>
#include <NewPing.h>
#include <Wire.h>
#include <VL53L1X.h>

#define MaxDistAE 400 // Max distance to detect for sensors A and E.
#define MaxDist 60 // Max distance chosen for sensors B, C and D. The sonar function ping_cm() returns zero if there is no ping echo within set distance limit MaxDist.
Servo myservo; //myservo is an object of class Servo.
//NewPing sA(5,5,MaxDistAE); // sA is an object of class NewPing with trig_pin = echo_pin = 5
VL53L1X sB; // sB is a LIDAR sensor of the type VL53L1X.
//NewPing sB(7,7,MaxDist); // sB is an object of class NewPing
//NewPing sC(8,8,MaxDist); // sC is an object of class NewPing
NewPing sC(5,5,MaxDist); // sC is an object of class NewPing
VL53L1X sD; // sD is a LIDAR sensor of the type VL53L1X.
//NewPing sD(10,10,MaxDist); // sD is an object of class NewPing
//NewPing sE(11,11,MaxDistAE); // sE is an object of class NewPing. It has happened that sE reported e = 1149 cm > MaxDistAE. It should not happen.
int StartSig = 1; //Used with the start module for remote start. The car should not start until StartSig is set to 1 by a remote control.
int KillSig = 1; // Startsig = 0 combined with KillSig = 1 is considered as the power on state but not the start state.
int i;
int imax = 7; //
int dt_Pings2 = 30; //50; //25;  // 2*dt_Pings2 >= 50
int dt_Pings3 = 30; //34; //17;   // 3*dt_Pings3 >= 50. Use at least 50 ms between pings of the same sensor.
int T = 20; //The inter-measurement period that should be at least as long as the TimingBudget for LIDAR sensors.
int v_start = 100;
int v_straight = 120; //150; //Speed straight forward in 3aS and 3bS.
int v_corr = 100; //120; //Speed at small corrections of the direction in steps 3aL, 3aR, 3bL, 3bR.
int v_turn = 90; //100; //Speed in sharp curves. 80 gives too low torque. The motor needs other gear wheels.
int b; //
int c; //
int d; //
int bold; //
int cold = 51; //
int dold; //
//const int incMax = 10; //Max acceptable increase in b or d between measurements without turning sharply in step 4.
//const int cdecmax = 10*v_straight/120*(2*T+dt_Pings3)/70; //If c decreases more than this between 2 pings
//the cause can be the sudden appearance of another car or turning towards a wall or an erroneous reading.
//const int amin = 8; // DevCar can get stuck on a wall unless there is a minimum allowed distance.
const int cmin = 50; //Drive straight ahead if c > cmin. DevCar may turn the wrong way in scenario 5a, 5b if cmin is larger than the width of the track.
//Dev car has a turn radius of about 40 cm to the outer wheel.
const int bmin = 11; //Turn left if b < bmin, c < cmin and d > dmin. Tried 40, 20, 30
const int dmin = 11; //Turn right if d < dmin, c < cmin and b > bmin. Tried 40, 20, 30
int bmax = 0; //The max value is likely to become about sqrt(2) times the distance to the wall to the right
int dmax = 0; //The max value is likely to become about sqrt(2) times the distance to the wall to the left
//const int akeepmax = 45; //The width of the racing track must have room for emin + car_width + bkeep.
int bkeep;
```

```
const int bkeepmin = 20; //Larger than or equal to bmin + tol
const int bkeepmax = 30;
const int blarge = 80;
int bturn; //b directly after turning right.
int dkeep;
const int dkeepmin = bkeepmin; //
const int dkeepmax = bkeepmax;
const int dlarge = blarge;
int dturn; //d directly after turning left.
int dflag; //dflag = 1 if DevCar should stay at a constant d = dkeep, i.e. follow the left wall
const int tol2 = 12; //Tolerance for large direction corrections needed in sharp curves and at some obstacles
const int tol = 6; //Tolerance for small direction corrections
//Define functions outside void setup() and void loop()
int read_b(void){ //Read the distance b from a LIDAR sensor.
 int s1;
    sB.startContinuous(T);
    s1 = sB.read()/10;//Read b.
    sB.stopContinuous();
    if (sB.timeoutOccurred()) { Serial.print("TIMEOUT for sB: Distance out of range for timing budget"); }
    Serial.print("b:");
    Serial.print(s1);
    Serial.println("cm");
    return s1;
}
int read_c(void){ //Read distance c from the ultrasonic sensor.
 int s2;
    s2 = sC.ping_cm();//Ping and measure c.
    delay(dt_Pings3);
    if (s2 == 0 || s2 >= MaxDist) //The distance is either above the upper distance limit or the supplied power is
unstable.
      s2 = MaxDist; //An erroneous sensor reading should not cause an exit from the loop.
    Serial.print("c:");
    Serial.print(s2);
    Serial.println("cm");
    return s2;
}
int read_d(void){ //Read the distance d from a LIDAR sensor.
  int s1;
    sD.startContinuous(T);
    s1 = sD.read()/10;//Read d.
    sD.stopContinuous();
    if (sD.timeoutOccurred()) { Serial.print("TIMEOUT for sB: Distance out of range for timing budget"); }
    Serial.print("d:");
    Serial.print(s1);
    Serial.println("cm");
    return s1;
}

//PWM pins on Arduino UNO are digital pins 3, 5, 6, 9, 10 and 11.
void setup() {
Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
//Setup commands for LIDAR sensors:
Wire.begin(); //Initiate the Wire library and join the I2C bus as a master unless the adress is specified.
Wire.setClock(400000); // Set the clock frequency for I2C communication to 400 kHz (fast mode).
pinMode(7, OUTPUT); // XSHUT port on sB set as OUTPUT for subsequent digitalWrite.
pinMode(10, OUTPUT); // XSHUT port on sD set as OUTPUT for subsequent digitalWrite.
```

```
digitalWrite(7, LOW);
digitalWrite(10, LOW);
pinMode(7, INPUT); //Return to INPUT mode (high impedance). Don't use digitalWrite(7, HIGH) since the XSHUT port is not 5 V tolerant.
sB.init(); //Reset the sensor. It will also reset the address. Therefore, run init before setting the address.
sB.setAddress(1); //Arbitrary 7-bit address
pinMode(10, INPUT);//Return to INPUT mode (high impedance). Don't use digitalWrite(7, HIGH) since the XSHUT port is not 5 V tolerant.
sD.init();
sD.setAddress(2);
sB.setDistanceMode(VL53L1X::Short); //In short distance mode, distance up to 1.3 m can be measured almost independently of ambient light
sD.setDistanceMode(VL53L1X::Short);
sB.setMeasurementTimingBudget(20000); // Minimum time for each measurement. The minimum value is 20000 us, only allowed in Short distance mode.
sD.setMeasurementTimingBudget(20000);
//sB.startContinuous(T); // It may be energy saving to measure when needed instead of continuously.
//sD.startContinuous(T);// Start continuous readings at a rate of one measurement every 20 ms (the
                        // inter-measurement period). This period should be at least as long as the timing budget.
//End of setup command block for LIDAR sensors.
pinMode(12, OUTPUT); //Initiates Motor shield Channel A pin for rotation direction of the Main Motor. Channel B is not used.
pinMode(9, OUTPUT); //Initiates Motor shield Channel A pin for brake control of the Main Motor
pinMode(2, OUTPUT); // Start signal digital pin connected to the starting module. A referee can set the value remotely if the pin mode is INPUT.
pinMode(4, OUTPUT); // Kill signal digital pin connected to the starting module. A referee will set the value remotely if the pin mode is INPUT.
digitalWrite(2,StartSig); //Set a value to the start signal pin.
digitalWrite(4,KillSig); //Set a value to the kill signal pin.
//Step 1: Start:
myservo.attach(6); // Attach the signal pin of servo to digital pin 6 of arduino
StartSig = digitalRead(2); //Read Start Signal Pin
KillSig = digitalRead(4);  //Read kill Switch Pin
Serial.print("StartSig = ");
Serial.println(StartSig);
//Serial.print("KillSig = ");
//Serial.println(KillSig);
if(StartSig == 1 && KillSig == 1){
//if(StartSig == 1){
//Step 2: Drive straight forward briefly while checking distances to the walls and determining which wall to follow:
myservo.write(110); //Set front wheel angular direction to be straight forward which means angle 110°-111° for the modified servo mechanism!
digitalWrite(12, HIGH); //Establishes forward direction of the motor connected to shield Channel A
digitalWrite(9, LOW);   //Disengage the brake for the motor connected to Channel A
analogWrite(3, v_start); //Sets the motor on Channel A at a speed <= 255. We have to adapt the speed setting to the speed obtained in tests.
for (i = 1; i <= imax; i++){
  b = read_b();
  d = read_d();
  if (b > bmax)  //It may be better to rely on bmax than on an average value
     bmax = b;
  if (d > dmax)
     dmax = d;
}
if (dmax <= bmax) {
```

```
  dflag = 1;
  dkeep = min(dkeepmax,max(d,dmin));
}
else {
  dflag = 0;
  bkeep = min(bkeepmax,max(b,bmin));  //Try to keep b equal to bkeep
}
Serial.print("bmax =");
Serial.print(bmax);
Serial.println("cm");
Serial.print("dmax =");
Serial.print(dmax);
Serial.println("cm");
Serial.print("bkeep =");
Serial.print(bkeep);
Serial.println("cm");
Serial.print("dkeep =");
Serial.print(dkeep);
Serial.println("cm");
c = read_c();
b = read_b();
d = read_d();
}
}
void loop() {
//In folkrace the kill switch will not be used.
//StartSig = digitalRead(2); //Read Start Signal Pin
//KillSig = digitalRead(4);  //Read kill Switch Pin
//if(StartSig == 1 && KillSig == 1){
//Step 3: Set speed to medium high:
Serial.print("dflag:");
Serial.println(dflag);
analogWrite(3, v_corr); //Set a medium high speed.
//Step 4: Small or large direction corrections and straight forward:
if (dflag == 1) {//Scenario 4a. Keep d constant
 if(((dkeep-d) > tol2) && (c > cmin) && (b > bmin)){
  for(i = 105; i >= 85; i = i - 5){
   myservo.write(i); //Turn the wheels sharply to the right.
   delay(1);
  }
  Serial.println("Scenario 4aRR. Turn sharply right!");
  while(((dkeep-d) > tol2) && (c > cmin) && (b > bmin)) {
   cold = c;
   b = read_b();
   c = read_c();
   d = read_d();
   //if (cold-c > cdecmax){c = cold;}
  }
  Serial.print("dkeep after loop:");
  Serial.print(dkeep);
  Serial.println("cm");
 }
 if(((dkeep-d) > tol) && (c > cmin) && (b > bmin)){
  myservo.write(98); //Turn the wheels to the right.
  Serial.println("Scenario 4aR. Turn right!");
  while(((dkeep-d) > tol) && (c > cmin) && (b > bmin)) {
```

```
    cold = c;
    b = read_b();
    c = read_c();
    d = read_d();
    //if (cold-c > cdecmax){c = cold;}
   }
   Serial.print("dkeep after loop:");
   Serial.print(dkeep);
   Serial.println("cm");
  }
  if(((d-dkeep) > tol2) && (c > cmin) && (b > bmin)) {
   for(i = 115; i <= 135; i = i + 5){
    myservo.write(i); //Turn the wheels to the left.
    delay(1);
   }
   Serial.println("Scenario 4aLL. Turn sharply left.");
   while(((d-dkeep) > tol2) && (c > cmin) && (b > bmin)) {
    cold = c;
    b = read_b();
    c = read_c();
    d = read_d();
    //if (cold-c > cdecmax){c = cold;} //Not good
   }
   Serial.print("dkeep after loop:");
   Serial.print(dkeep);
   Serial.println("cm");
  }
  if(((d-dkeep) > tol) && (c > cmin) && (b > bmin)) {
   myservo.write(120); //Turn the wheels to the left.
   Serial.println("Scenario 4aL. Turn left.");
   while(((d-dkeep) > tol) && (c > cmin) && (b > bmin)) {
    cold = c;
    b = read_b();
    c = read_c();
    d = read_d();
    //if (cold-c > cdecmax){c = cold;} //Not good
   }
   Serial.print("dkeep after loop:");
   Serial.print(dkeep);
   Serial.println("cm");
  }
  if(((d-dkeep) <= tol) && ((dkeep-d) <= tol) && (c > cmin) && (b > bmin)) {
   myservo.write(110); // Drive straight forward
   analogWrite(3, v_straight); //Set a high speed
   Serial.println("Scenario 4aS. Drive straight forward.");
   while(((d-dkeep) <= tol) && ((dkeep-d) <= tol) && (c > cmin) && (b > bmin)) {
    cold = c;
    b = read_b();
    c = read_c();
    d = read_d();
    //if (cold-c > cdecmax){c = cold;}
   }
   Serial.print("dkeep after loop:");
   Serial.print(dkeep);
   Serial.println("cm");
  }
```

```
}
else {//Scenario 4b. Keep a constant a
 if(((bkeep-b) > tol2) && (c > cmin) && (d > dmin)) {
  for(i = 115; i <= 135; i = i + 5){
   myservo.write(i); //Turn the wheels to the left.
   delay(1);
  }
  Serial.println("Scenario 4bLL. Turn sharply left.");
  while(((bkeep-b) > tol2) && (c > cmin) && (d > dmin)) {
   cold = c;
   b = read_b();
   c = read_c();
   d = read_d();
   //if (cold-c > cdecmax){c = cold;}
  }
  Serial.print("bkeep after loop:");
  Serial.print(bkeep);
  Serial.println("cm");
 }
 if(((bkeep-b) > tol) && (c > cmin) && (d > dmin)) {
  myservo.write(120); //Turn the wheels to the left.
  Serial.println("Scenario 4bL. Turn left.");
  while(((bkeep-b) > tol) && (c > cmin) && (d > dmin)) {
   cold = c;
   b = read_b();
   c = read_c();
   d = read_d();
   //if (cold-c > cdecmax){c = cold;}
  }
  Serial.print("bkeep after loop:");
  Serial.print(bkeep);
  Serial.println("cm");
 }
 if(((b-bkeep) > tol2) && (c > cmin) && (d > dmin)) {
  for(i = 105; i >= 85; i = i - 5){
   myservo.write(i); //Turn the wheels to the right.
   delay(1);
  }
  Serial.println("Scenario 4bRR. Turn sharply right.");
  while(((b-bkeep) > tol2) && (c > cmin) && (d > dmin)) {
   cold = c;
   b = read_b();
   c = read_c();
   d = read_d();
   //if (cold-c > cdecmax){c = cold;}
  }
  Serial.print("bkeep after loop:");
  Serial.print(bkeep);
  Serial.println("cm");
 }
 if(((b-bkeep) > tol) && (c > cmin) && (d > dmin)) {
  myservo.write(98); //Turn the wheels to the right.
  Serial.println("Scenario 4bR. Turn right.");
  while(((b-bkeep) > tol) && (c > cmin) && (d > dmin)) {
   cold = c;
   b = read_b();
```

```
    c = read_c();
    d = read_d();
    //if (cold-c > cdecmax){c = cold;}
   }
   Serial.print("bkeep after loop:");
   Serial.print(bkeep);
   Serial.println("cm");
  }
  if(((b-bkeep) <= tol) && ((bkeep-b) <= tol) && (c > cmin) && (d > dmin)) {
   myservo.write(110); // Drive straight forward
   analogWrite(3, v_straight); //Set a high speed
   Serial.println("Scenario 4bS. Drive straight forward.");
   while(((b-bkeep) <= tol) && ((bkeep-b) <= tol) && (c > cmin) && (d > dmin)) {
    cold = c;
    b = read_b();
    c = read_c();
    d = read_d();
    //if (cold-c > cdecmax){c = cold;}
   }
   Serial.print("bkeep after loop:");
   Serial.print(bkeep);
   Serial.println("cm");
  }
 }
 //Exit from step 4 above should only take place if c < cmin or b < bmin or d < dmin.
 //Step 5: Slow down or stop temporarily to get time to measure distances:
 analogWrite(3, v_turn); //Slow down in front of an obstacle.
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 /////
 //Step 6: Avoid obstacles and turn in curves not already handled by step 4:
 if (d >= dmin && c <= cmin && d >= b) { //Scenario 6a
  for(i = 115; i <= 135; i = i + 5){
   myservo.write(i); //myservo.write(135); // Turn sharply left
   delay(1);
  }
  Serial.println("Scenario 6a. Turn sharply left.");
  while (c <= cmin){
   cold = c;
   c = read_c();
   //if (cold-c > cdecmax){c = cold;}
  }
  for(i = 130; i >= 110; i = i - 5){
     myservo.write(i); //Set wheels straight forward.
     delay(1);
  }
  b = read_b();
  if (b > blarge){ //An obstacle was passed and was not a curve.
   Serial.println("The car has passed the obstacle which was not a curve.");
   d = read_d();
   dturn = d;
   //The obstacle was likely to be another robot.
   for(i = 105; i >= 85; i = i - 5){
    myservo.write(i); //myservo.write(85); // Turn sharply right to get DevCar parallel to a wall again.
    delay(1);
   }
   while (b > bkeepmin && c > cmin && d < dturn*2){
```

```
    cold = c;
    b = read_b();
    c = read_c();
    d = read_d();
    //if (cold-c > cdecmax){c = cold;}
   }
   for(i = 90; i <= 110; i = i + 5){
    myservo.write(i); //Set wheels straight forward again.
    delay(1);
   }
   dflag = 1;
   dkeep = min(dkeepmax,max(d,dkeepmin));
   Serial.print("dkeep:");
   Serial.print(dkeep);
   Serial.println("cm");
  }
  else{//DevCar passed through a curve to the left. Follow the right wall.
   dflag = 0;
   bkeep = min(bkeepmax,max(b,bkeepmin));
   Serial.print("bkeep:");
   Serial.print(bkeep);
   Serial.println("cm");
   Serial.println("DevCar passed through a curve to the left.");
  }
 }
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 //////
 else if (b >= bmin && c <= cmin && b > d) { //Scenario 6b.
  for(i = 105; i >= 85; i = i - 5){
   myservo.write(i); //myservo.write(85); // Turn sharply right
   delay(1);
  }
  Serial.println("Scenario 6b. Turn sharply right.");
  while (c <= cmin){
   cold = c;
   c = read_c();
   //if (cold-c > cdecmax){c = cold;}
  }
  for(i = 90; i <= 110; i = i + 5){
   myservo.write(i); //Set wheels straight forward.
   delay(1);
  }
  d = read_d();
  if (d > dlarge){//The car has passed the obstacle which was not a curve.
   Serial.println("The car has passed the obstacle which was not a curve.");
   b = read_b();
   bturn = b;
   for(i = 115; i <= 135; i = i + 5){
    myservo.write(i); //myservo.write(135); // Turn sharply left to get DevCar parallel to a wall again.
    delay(1);
   }
   while (d > dkeepmin && c > cmin && b < bturn*2){
    cold = c;
    b = read_b();
    c = read_c();
    d = read_d();
```

```
    //if (cold-c > cdecmax){c = cold;}
   }
   for(i = 130; i >= 110; i = i - 5){
    myservo.write(i); //Set wheels straight forward again.
    delay(1);
   }
   dflag = 0;
   bkeep = min(bkeepmax,max(b,bkeepmin));
  }
  else {//DevCar has passed through a curve to the right.
   dflag = 1;
   dkeep = min(dkeepmax,max(d,dkeepmin)); //Follow the left wall.
   Serial.println("DevCar passed through a curve to the right.");
  }
 }
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 ////////
 else if (b <= bmin && c >= cmin && d > dmin){ //There is an obstacle to the right. Turn left somewhat.
  Serial.println("Scenario 6c. Turn somewhat left.");
  for(i = 115; i <= 125; i = i + 5){
    myservo.write(i); //Turn the wheels to the left.
    delay(1);
  }
  while(b <= bmin && c >= cmin && d >= dmin){
   cold = c;
   b = read_b();
   c = read_c();
   d = read_d();
   //if (cold-c > cdecmax){c = cold;}
  }
 }
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 /
 else if (d <= dmin && c >= cmin && b > bmin){ //There is an obstacle to the left. Turn right somewhat.
  Serial.println("Scenario 6d. Turn somewhat right.");
  for(i = 105; i >= 95; i = i - 5){
    myservo.write(i); //Turn the wheels to the right.
    delay(1);
  }
  while(d <= dmin && c >= cmin && b >= bmin){
   cold = c;
   b = read_b();
   c = read_c();
   d = read_d();
   //if (cold-c > cdecmax){c = cold;}
  }
 }
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 ///////////
 else if (b <= bmin && d <= dmin) { //Scenario 6e is unlikely
  digitalWrite(9, HIGH);   //Stop the car. Engage the brake for the motor connected to Channel A
  Serial.println("Scenario 6e. Stop and wait.");
  while (b <= bmin && d <= dmin) {
   b = read_b();
   d = read_d();
  }
```

```
  digitalWrite(9, LOW);   //Start the car.
}
//Step 7: Read sensor values outside all other steps to handle cases not covered by the previous steps
//b = read_b();
//c = read_c();
//d = read_d();
//else if(StartSig == 0 && KillSig == 0){
//  digitalWrite(9, HIGH);   //Stop the car. Engage the brake for the motor connected to Channel A
//  Serial.print("StartSig again = ");
//  Serial.println(StartSig);
//  Serial.print("KillSig again = ");
//  Serial.println(KillSig);
//}
}
```