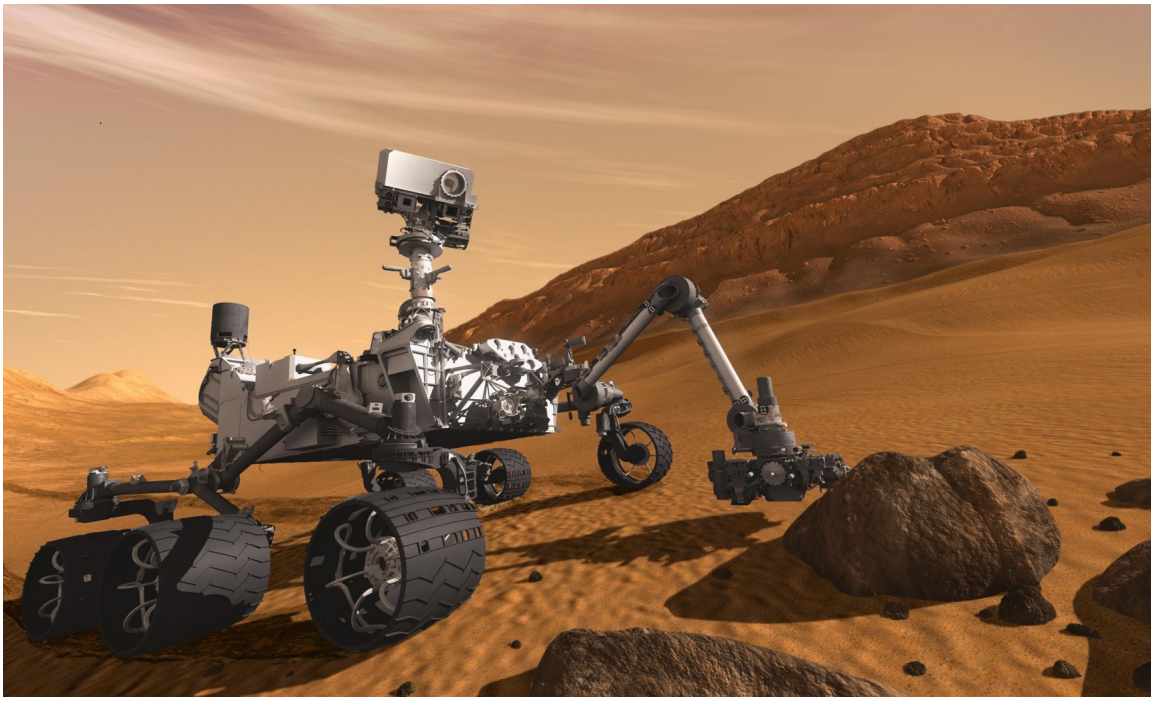
Robotics Club Werenfridus



Manual MER caterpillar-ar

Ton Schuckman
Physics amanuensis/instructor

Tabor College Werenfridus

index	page
the rectangular baseplate	03
slots for the servo	03
middle wheels and side-plates	04
mounting the motors	04
front wheels and caterpillar tracks	04
soldering the wires to the motors	04
connecting the baseplate and side-plates	04
assembling the roof beams	05
working of the electronic circuit	05
diodes and capacitors	05
placing the microcontroller	06
wires on the microcontroller	06
polarity - batteries and storage	06
assembling the circuit board	06
the battery pack	06
electrical diagram	07
table 1 abbreviations on the circuit board	08
table 2 connections on the circuit board	09
attachment of the servo	09
mounting the ultrasonic sensor	09
connecting the ultrasonic sensor	10
installing the Arduino program	10
software to test the engines, servo, u.s. sensor	11
software for autonomous driving	12
table 3 parts list	14
the mer (mars exploration rover) is finished	15
warranty and ownership	15
disposal and recycling	15

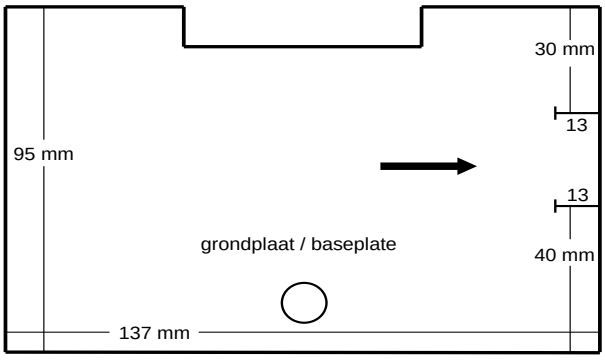
The manual is divided in three parts:

- Mechanical part
- Electronic circuit with connections
- Programming of the microcontroller

Mechanical part

The rectangular baseplate
While drilling use a pair of safety glasses against small flying pieces of metal. In case of a long hairstyle, tie your hair up at the back of your head to avoid entanglement with the drill head. Mark the holes on the rectangular baseplate. The black arrow marks the upper and front side of the baseplate. The circular hole with a diameter of 12 mm is situated at the right side. Put the side plates centered in the middle under the baseplate at such a way that the motors are situated at the back. Turn the assembly upside down and mark the spots at the four places where the holes must be drilled. Rear left is located 10 mm further to the rear. Place the microcontroller 8 mm from the rear and mark these holes as well. When you're not sure, take a look at the demo. Now you can drill al the marked holes with a 3,0 mm drill. Splinters can be removed by hand with a 10 mm drill. Don not assemble the parts yet, this will be done later.

Slots for the servo
Two slots are made at the front of the baseplate to facilitate the servo. Use a metal saw and make a slot measured 30 mm from the left side with a depth of 13 mm. Do the same measured 40 mm from the right side.



Figuur 1. The baseplate

Middle wheels and side-plates
Put an m4 bolt trough the wheel and screw a nut against the inner site of the wheel. Then loosen it half a turn so that the wheel can rotate without friction. Now you can attach the wheels on the side- plates with another m4 nut. Tighten them by turning the two nuts with two wrenches.

Mounting the motors
The motors are placed with the shortest bolts at the back of the side plates. The shortest bolts are essential; otherwise the gears inside the motor can be damaged. Screw gently by hand the three bolts in the motor, this way damage of the screw-thread is avoided, then tighten them with a screw-driver. The driving wheel can be connected to the motor axis with a bolt. The bolt comes at the flat side of the motor axis. In some cases, the driving wheel most be assembled with the cylindrical aluminum part.

Front wheels and caterpillar tracks
Sometimes the caterpillar track has to be adjusted to the right length. Put the caterpillar track around the not yet mounted front wheel and then around the rear driving wheel. Attach the front wheel the same way you attached the middle wheels.

Soldering the wires to the motors
Do not inhale the soldering smoke, it consists of heavy metals. Take care not to press the hot soldering iron against the plastic parts of the motors. In general: for the **+** wires use red and for the **–** a black wire. Cut the wires at the right length (± 20 cm) and solder on the left motor a red wire at the **+** eyelet and a black wire at the **–** eyelet. On the right motor it is reversed, a red wire at the **–** eyelet and a black wire at the **+** eyelet. For those who have never soldered: hold the end of the hot soldering iron for two seconds against the eyelet, which has already the wire in it, then push the solder at the same point and you will see the melting of the solder in the eyelet with wire. Then remove the soldering iron and wait a while to let it cool before you check the firmness of your connection. The other end of the wires are equipped with so called header pins.

Connecting the baseplate and side-plates
Now the two completed side plates are placed under the baseplate with the 75 mm long m3 bolts and nuts. Do this in such a way that the two motors are situated at the back of the baseplate. The 75 mm long bolts will be used to mount the roof beams and protect the electronics.

Assembling the roof beams
Saw two pieces of aluminum: (10 mm – 160 mm long). Drill 38 mm from the end of the beams three holes of 3,0 mm. The fourth hole is 28 mm from an end. Connect the two beams at the top of the 75mm bolts. This way the electronics are protected and there is place for the power bank on top of it, and you can lift the robot from the ground for testing.

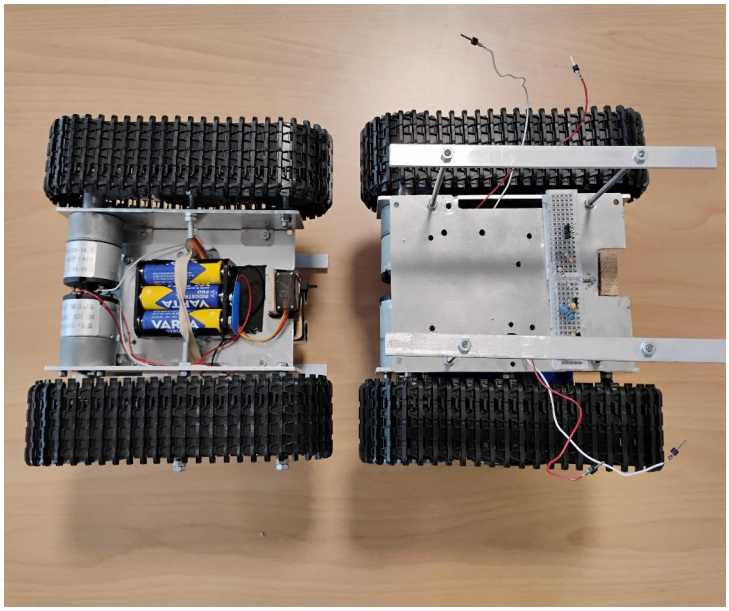


Figure 2. Under side and top side

Electronic circuit

Working of the electronic circuit – transistors

The microcontroller works, like most electronics, at a voltage of 5 volt, which is delivered by six AA batteries. However, this voltage (max. 9 volt) is to high for the microcontroller but it has a voltage regulator on board which makes 5 volt out of it. So, the following scheme is going to happen: when through the program that you will write a voltage of 5 volt is put on pin 5, a current will flow through the resistor, decreased by it, to the basis B of the transistor. A transistor has three connections: b/basis, c/collector and e/emitter. Normally, when there is no voltage on the basis the transistor is closed. This is possible because the transistor contains a semi-conductor that will only conduct when a small current goes from the basis to the emitter. Only then the main current from the collector to the emitter can flow. The motor will run because an electrical circuit has been made with the battery pack. Try to follow the electrical diagram in figure 3. Note: crossing lines are not connected with each other.

Diodes and capacitors

The diode, a part that allows the current to flow in one direction is a safety precaution. When an electromotor is rotated by hand it will act as a dynamo and could damage the microcontroller. The diode takes care that this induction current can only flow to the battery. The capacitors function is to reduce the electromagnetic distortion from the motors.

Wires on the microcontroller
Take good care not to mistakenly wrongly connect the plus and minus wires from the power plug on the microcontroller on the circuit-board. The microcontroller might be damaged!

Polarity
The capacitor and resistor are connected with disregard of the plus or minus side. The diode however must be connected with the gray band (the blockade side) at the plus side. The transistors are mounted with the metal side to the front of the vehicle so that the e/emitter, c/collector and b/basis are connected properly.

Assembling the printed circuit board
First saw with an iron saw the circuit-board length-wise in two parts. Paste them 30 mm from the front side of the base-plate. Don't make the wires to long to maintain a good overview and when available you can also use jumper-wires. Cut the ends of the resistors, diodes, and capacitors to a length of 12 mm. Use the scheme of table 1 and 2 of the plug-in circuit-board to connect first al the electronic components at the right place. After this is done put the wires a +, b +, c ─ and d ─ in the right places. Each rectangle in table 2 stands for one hole of the plug-in circuit-board. When you take a close look, you will see that the holes at the back are connected in pairs of five. This way the components are connected with each other. The remaining wires are placed after the servo is fastened.

Placing the microcontroller
Attach 4 M2.5 x 20 bolts with nuts on the microcontroller. Screw 4 extra nuts on the M2.5 bolts. Then fasten the microcontroller with the connectors at the rear on the baseplate with 4 nuts. Take good care to give the bolts the same height, the microcontroller could break! After this handle with care to avoid damage to the microcontroller! See figure 4.

Batteries and storage
When you didn't already buy them, now is the time to purchase the six AA 1,5V batteries. When stowing the vehicle away, disconnect the power plug from the microcontroller and put it on a plastic caterpillar to avoid short-circuit and fire! Storage: take the batteries out their compartment to avoid leakage and short-circuit.

The battery pack
First, put layer of duct-tape at the underside of the baseplate to avoid a short-circuit of the battery pack. Take a rubber band and wrap this on the left roof beam, put it through the open space and around the battery holder and up again on the right side through the 15 mm hole of the baseplate. Use a match to hold it in place. The blank ends of the black and red wires from the battery pack are soldered at separate male headers.

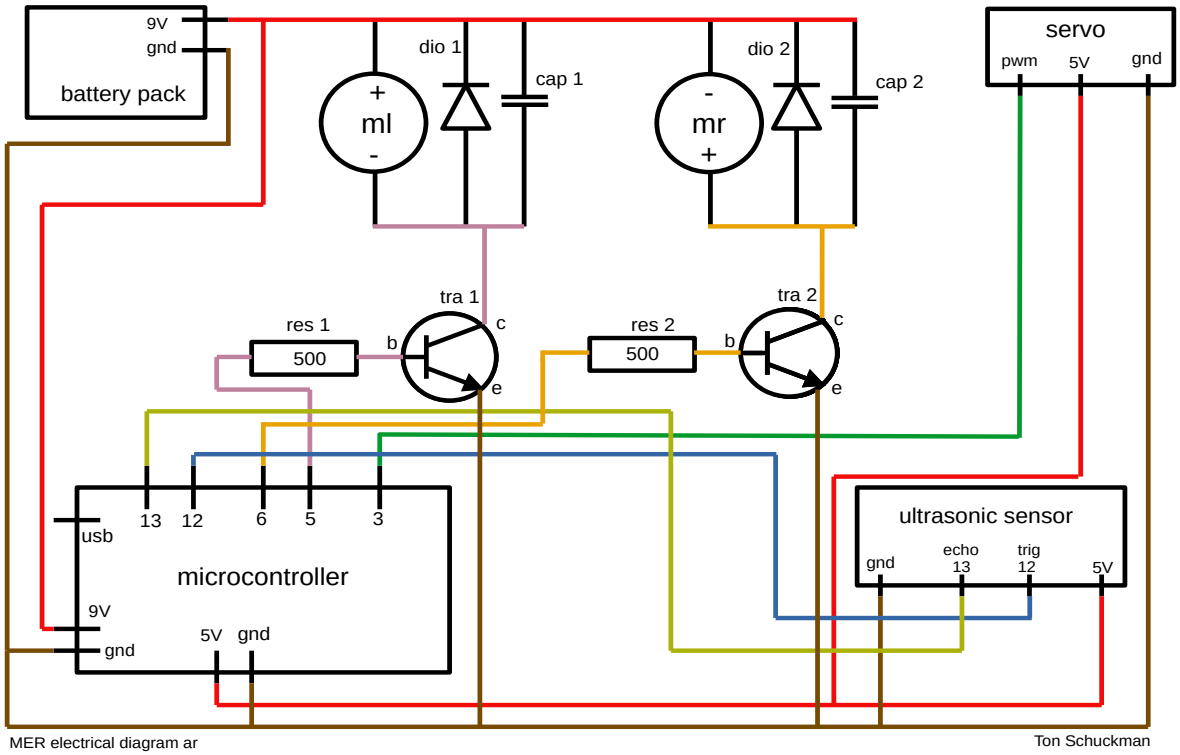


Figure 3. Electrical diagram

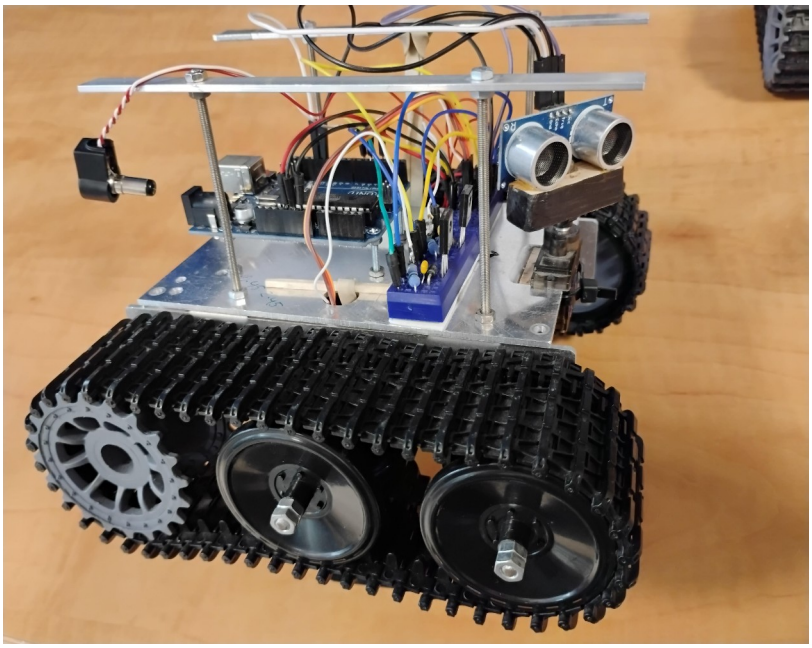
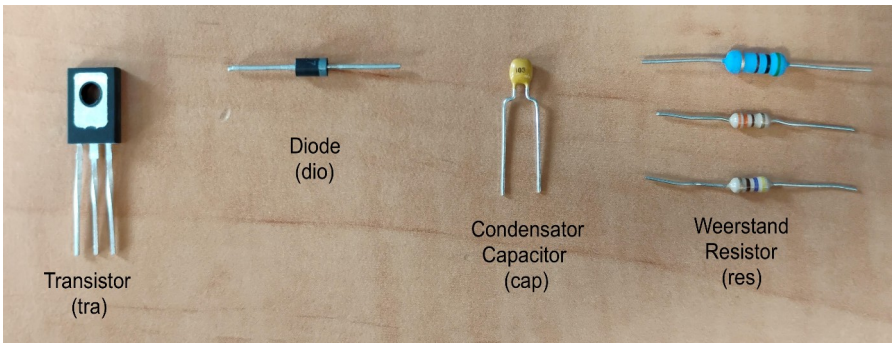
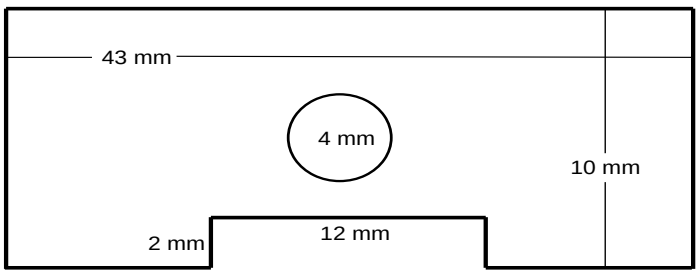


Figure 4. Components configuration

bas	basis (transistor)	m.l.	motor left
bat	battery holder	m.r.	motor right
bro	brown wire (servo)	pin	connection on the mic
cap	capacitor	res	resistor
col	collector (transistor)	ser	servo
dio	diode	tra	transistor
ech	echo (u.s. sensor)	tri	trigger (u.s. sensor)
emi	emitter (transistor)	u.s.	ultrasonic sensor
gnd	ground (—)	wir	wire
mer	mars exploration rover	yel	yellow wire (servo)
mic	micro controller		

Table 1. Abbreviations on the circuit-board





Bracket ultrasonic sensor
Voetstukje ultrasoon sensor

	wir e +	wir e +			wir c —				u.s. 5v	u.s. gnd						
	wir a +	wir b +			wir d —											
	m.l. +	m.r. +			mic —			ser yel	ser 5v	ser gnd						
		mic +			bat —											
	bat 9v+				pin gnd			pin 3	pin 5v	pin gnd						

Table2 Connections on the circuit-board **left**

		tra emi	tra col	tra bas						tra emi	tra col	tra bas				
			dio 1 —			dio 1 +					dio 2 —			dio 2 +		
			cap 1 —			cap 1 +		res 500			cap 2 —			cap 2 +		res 500
		wir c —	m.l. —	res 500		wir a +		pin 5		wir d —	m.r. —	res 500		wir b +		pin 6

Table 2 Connections on the circuit-board **right**

u.s. tri goes from the sensor to pin 12 on the microcontroller

u.s. ech goes from the sensor to pin 13 on the microcontroller

Attachment of the servo

Put the servo at the front of the MER so it can turn around to avoid obstacles. Two slots where already sawed for this at the start of this manual. Turn the piece of aluminum between the slots 90° downwards so the servo can be fastened with a tie-wrap to the baseplate.

Mounting the ultrasonic sensor

Saw a piece of aluminium 10 - 43 mm and drill in the middle a hole of 4 mm. Make with the help of a file in the middle a recess for the aluminium rectangle of the u.s. sensor. Use a little glue to attach the bracket of the servo at the bottom of the sensor base plate, take care to align the holes! The sensor is glued with the pins up at the upper side.

Connecting the ultrasonic sensor
Screw the assembly with a small screw on the servo. Take four male/female wires to connect the u.s. sensor according to table 2. Check with table 2 of al the wires are connected.

Programming the microcontroller

Installing the Arduino program
The Arduino software can be downloaded for free at the Arduino website. The programs that are already written for you underneath this page contains the code to test the engines, servo and ultrasonic sensor. Behind two // explanations can be add that will not interfere with the program, in other words the microcontroller does not see this text. Just type this program into the Arduino IDE (integrated development environment) and you have written your first computer program! When the microcontroller is connected to the computer and installed, the program can be uploaded. Possibly a port has to be assigned: click at the top of the Arduino IDE at → hulpmiddelen → ports, and see which port is used. Then click at your pc on → start → right mouse button computer → beheren → apparaat beheer, and see if the right port is used. Adjust this setting when necessary.

Software to test the engines

```
const byte motorLeft = 5; const byte motorRight = 6; // motors are assigned to pin 5 and 6.
void setup() // put your setup code here, to run once:
{ pinMode(motorLeft, OUTPUT); pinMode(motorRight, OUTPUT); } // pins set as output
void loop() // put your main code here, to run repeatedly:
{ delay(5000); // time to put the robot on the ground. (5000) = 5 seconds.
// right.
digitalWrite(motorLeft, HIGH); delay(1000); digitalWrite(motorLeft, LOW); delay(1000);
// left.
digitalWrite(motorRight, HIGH); delay(1000); digitalWrite(motorRight, LOW); delay(1000);
// straight.
digitalWrite(motorLeft, HIGH); digitalWrite(motorRight, HIGH); delay(1000);
digitalWrite(motorLeft, LOW); digitalWrite(motorRight, LOW); delay(1000); }
```

Software to test the servo

```
#include <Servo.h>
Servo myServo;  // make a servo object.

void setup()
{ myServo.attach(2); } // put the signal wire to pin 3.
void loop()
{ myServo.write(0);     delay(1000);   // turn left and Wait one second.
  myServo.write(90);   delay(1000);   // turn forward.
  myServo.write(180); delay(1000); } // turn right.
```

Software to test the ultrasonic sensor

```
const int trigPin = 12; const int echoPin = 13; float duration, distance;
void setup()
  { pinMode(trigPin, OUTPUT); pinMode(echoPin, INPUT);  Serial.begin(9600); }
void loop()
  { digitalWrite(trigPin, LOW); delayMicroseconds(2);
    digitalWrite(trigPin, HIGH); delayMicroseconds(10);
    digitalWrite(trigPin, LOW); duration = pulseIn(echoPin, HIGH);
    distance = (duration * 0.0343) / 171;
    Serial.print("Distance in meters: "); Serial.println(distance); delay(1000); }
```

When the above programs are working, we continue with the main program that implements the servo and ultrasonic sensor. Because of the size we do this with copy/paste.

Software for autonomous driving with ultrasonic sensor and servo

```
#include <Servo.h> // create servo object to control a servo.
Servo myservo;
// pin configuration.
const byte servoPin = 3; const byte motorLeft = 5; const byte motorRight = 6;
const byte trigPin = 12; const byte echoPin = 13;  // send = trigger pin, receive = echo pin.
int distanceLeft, distanceStraight, distanceRight;
// delay determines how long the MER turns, increase to make the turning angle larger.
const int turnDelay = 700; const int objectDelay = 900;
// standard angle for looking straight ahead, compensate for construction errors.
const int standardAngle = 90;
// angle in degrees that controls the direction in which the sensor looks.
const int viewAngle = 45;
// delay to let servo turn, not optimized; for smaller angles smaller delay could be used.
const int servoDelay = 270;
// object distance in cm at which the MER decides to turn another way.
const int lineOfSight = 45;
// pwm duty cycle of the motor, use a value from 0 (no movement) to 255 (full speed).
const int motorSpeedLeft = 80;
const int motorSpeedRight = 80;
void setup() // setup pin configuration and initialize the sensor position.
{ delay(3000); // delay to give us some time to place the MER on the ground.
myservo.attach(servoPin); // initialize the servo and look straight ahead.
Serial.begin(9600); // initialize serial communication.
```

```
turn_servo(standardAngle);
pinMode(motorLeft, OUTPUT); pinMode(motorRight, OUTPUT);
pinMode(trigPin, OUTPUT); pinMode(echoPin, INPUT); drive_mer(); } // start the mer.

void drive_mer() // this function lets the MER drive forward.
{ analogWrite(motorLeft, motorSpeedLeft); analogWrite(motorRight, motorSpeed); }

void stop_mer() // this function stops the mer.
{ digitalWrite(motorLeft, LOW); digitalWrite(motorRight, LOW); }

void turn_servo(int angle)
// turn servo to given angle, do not return until angle is reached.
// if the servo is already at the correct angle, we don't have to do anything.
{ if (angle == myservo.read()) { return; }
else { myservo.write(angle); delay(servoDelay); } }


int look() // looking function, returns distance in the observed direction as an integer.
{ int duration, distance;
digitalWrite(trigPin, LOW); delayMicroseconds(50); digitalWrite(trigPin, HIGH);
delayMicroseconds(10); digitalWrite(trigPin, LOW); duration = pulseIn(echoPin, HIGH);
distance = (duration / 2) / 29.1; return distance; } // in cm.

// turning function: makes MER turn in preferred direction. 0 for left and 1 for right.
void turn_mer(int direction)
{ if (direction == 0) { digitalWrite(motorLeft, LOW); digitalWrite(motorRight, HIGH); }
else { digitalWrite(motorLeft, HIGH); digitalWrite(motorRight, LOW); }
delay(turnDelay); stop_mer(); } // stop the engines when the MER has turned far enough.

// this function turns the MER depending on the observed distances.
void handle_measurements(int distanceLeft, int distanceStraight, int distanceRight)

// if objects in each direction, stop the mer.
{ if(distanceLeft < lineOfSight and distanceStraight < lineOfSight and
distanceRight < lineOfSight) { stop_mer(); }

// if we see something on the left side, turn right.
else if (distanceLeft < lineOfSight) { stop_mer(); delay(objectDelay);  turn_mer(1);
stop_mer(); delay(objectDelay); drive_mer(); }
```

```
// if something straight ahead go to best direction.
else if (distanceStraight < lineOfSight) { if (distanceRight > distanceLeft) { stop_mer();
delay(objectDelay); turn_mer(1); delay(objectDelay); } else { stop_mer();
delay(objectDelay(); turn_mer(0); } delay(objectDelay); drive_mer(); }

// if we see something on the right side, turn left.
else if (distanceRight < lineOfSight) { stop_mer(); delay(objectDelay); turn_mer(0);
delay(objectDelay); drive_mer(); } }

void loop() // look each direction starting left.
{ int distanceLeft, distanceStraight, distanceRight;
turn_servo(standardAngle + viewAngle); distanceLeft = look();
turn_servo(standardAngle); distanceStraight = look();
turn_servo(standardAngle - viewAngle); distanceRight = look();

Serial.print(“ Left: “); Serial.println(distanceLeft);
Serial.print(“ Straight: “); Serial.println(distanceStraight);
Serial.print(“ Right: “); Serial.println(distanceRight);
Serial.println(“ “);
handle_measurements(distanceLeft, distanceStraight, distanceRight); }
```

part	type	amount
tank chassis	he 0102-460	1
micro controller	arduino uno	1
circuit - board	he 0109-002	1
battery holder	co 615587-89	1
bolts and nuts	m3 x 75 mm 4x	< >
bolts and nuts	m2.5 x 20 mm 4x	< >
aluminium strip	bm 10 - 800 mm	1
diodes	co 50v 1 amp.	2
transistors	npn bd139	2
resistors	500 ohm	2
capacitors	100nf 50v	2
wires male/male	he 0108-002	< >
wires male/female	he 0166-001	4
ultrasonic sensor	hc-sr04	1
servo	sg90	1
5.5 mm plug haaks	oc	1
battery plug	co 624691-89	1
jumper wires	he 0167-001	1
male headers	he 0236-600	1
batteries	bm 1,5 volt aa	6
usb cable	a to b	1 *
computer	arduino software	1 *

Table 3. Parts list * available at home or to be purchased

When finished
This is the end of the support from the physics department but you're always welcome to attend the robotics afternoons to expand the robot with your own ideas, build a solar power station, upgrade to a Raspberry Pi version to control the MER with a mobile phone and use the MER as a desktop pc, or even use the Arduino platform for something totally different. You can also take a look on the internet for ideas.

Warranty and Ownership
During all your years on Werenfridus, you have a full warranty on the MER, excluding the microcontroller, it has its own factory warranty. So, if your MER doesn't work anymore and you can't solve the problem by yourself, bring it to the physics cabinet to have it repaired. When finished and taken home, the MER becomes your property, therefore the school is not accountable for damage/accidents caused by the MER during its use and storage. Always remove the batteries to avoid leakage.

Disposal and Recycling
In the (unhappily) event that you want to dispose the MER, please don't throw it away but bring it back to the physics cabinet so that it can be taken apart for spare parts. This is also valid for the situation when the Mer is not finished. Their will be no refund.

Ton Schuckman
Physics amanuensis/instructor
aj.schuckman@tabor.nl

Tabor College Werenfridus